REAL: A Reciprocal Protocol for Location Privacy in Wireless Sensor Networks

Jia-Dong Zhang, and Chi-Yin Chow, *Member, IEEE*

Abstract—K-anonymity has been used to protect location privacy for location monitoring services in wireless sensor networks (WSNs), where sensor nodes work together to report k-anonymized aggregate locations to a server. Each k-anonymized aggregate location is a cloaked area that contains at least k persons. However, we identify an attack model to show that overlapping aggregate locations still pose privacy risks because an adversary can infer some overlapping areas with less than k persons that violates the k-anonymity privacy requirement. In this paper, we propose a reciprocal protocol for location privacy (REAL) in WSNs. In REAL, sensor nodes are required to autonomously organize their sensing areas into a set of non-overlapping and highly accurate k-anonymized aggregate locations. To confront the three key challenges in REAL, namely, self-organization, reciprocity property and high accuracy, we design a state transition process, a locking mechanism and a time delay mechanism, respectively. We compare the performance of REAL with current protocols through simulated experiments. The results show that REAL protects location privacy, provides more accurate query answers, and reduces communication and computational costs.

Index Terms—Location privacy, k-anonymity, wireless sensor networks, location monitoring systems, aggregate locations

1 Introduction

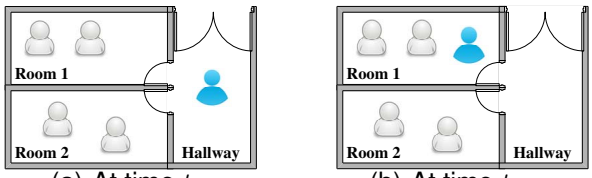
With the advancement of sensor and wireless communication technologies, location monitoring applications have been developed for surveillance and location systems. Basically, location monitoring applications use sensors to gather personal locations and provide location-based services [22], [39], [42]. However, with an untrustworthy server, an adversary may abuse its received location information to infer personal sensitive information. As a result, monitoring personal locations poses privacy threats to the monitored individuals [12], [21], [30], [39].

To tackle such privacy threats, an effective way is to use k-anonymity techniques [5], [12], [16], [19], [21], [25], [34], [44], [48]. Basically, a k-anonymity technique extends a person's location to a cloaked area that covers this person and at least $k-1$ other persons, so that this person is indistinguishable from the persons residing in the cloaked area. In wireless sensor networks (WSNs), this kind of cloaked areas is defined as **k-anonymized aggregate locations** [12]. A k-anonymized aggregate location R is represented in a form of a cloaked area A along with the number of persons (moving objects) N residing in A, where $N \geq k$, written as $R = \langle A, N \rangle$ hereafter. Previous work has defined the **reciprocity property** as a sufficient condition for spatial k-anonymity, i.e., the persons in a cloaked area share the same k-anonymized cloaked area [19], [20], [28]. In the context of WSNs, the reciprocity property requires that sensor nodes in the same aggregate location area share the same aggregate location, that is, each sensor node is included into one and only one aggregate location.

• *J.-D. Zhang and C.-Y. Chow are with Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: jzhang26-c@my.cityu.edu.hk, chiychow@cityu.edu.hk.*

Let us consider a typical application scenario. A location monitoring system can be deployed in a mall to monitor its visitors to provide various important services, including but not limited to (a) *density queries* — determining the number of persons within a certain region, (b) *safety control* — triggering an alarm if the number of persons in a region exceeds a predefined threshold, and (c) *resource management* — turning off some building facilities if the number of persons in a region is less than a predefined threshold. This kind of location monitoring applications usually can be implemented by four steps. (1) Wireless sensor nodes are deployed in the mall to communicate with a small wireless transmitter worn by persons in order to determine their exact locations and identities [22], [39]. (2) Each sensor node counts the number of persons in its sensing area; note that a certain person is counted by only one sensor node, since the transmitter worn by the person maintains only one connection to a certain sensor node, although it may be sensed by more than one sensor node [42]. (3) Each sensor node blurs its sensing area into a k-anonymized aggregate location and only reports the aggregate location to the server to offer monitoring services [12], in which the value of k is set by the system administrator. (4) The spatial histogram technique is employed to answer aggregate queries (i.e., "how many objects are located in a certain area") based on k-anonymized aggregate locations reported from sensor nodes [12].

Unfortunately, although in-network (or peer-to-peer) spatial cloaking algorithms [10], [11], [12], [18], [26], [27] have been designed for this kind of location monitoring applications in WSNs, they cannot guarantee that all aggregate locations can satisfy the reciprocity property, since a sensor node may be involved in more than one aggregate location. Thus, their generated aggregate locations still pose privacy breaches.



(a) At time t_i (b) At time t_{i+1}

Fig. 1: An example of a privacy breach

Fig. 1 depicts an example of a privacy breach, where three sensor nodes are installed in Room 1, Room 2 and Hallway to count the number of persons in their sensing area. Given the anonymity level $k = 3$ by the system administrator: (1) At time t_i (Fig. 1a), the three sensor nodes generate aggregate locations

$$R_1^{t_i} = \langle A_1^{t_i} = \{Room\ 1, Room\ 2\}, N_1^{t_i} = 4\rangle\ ,$$
$$R_2^{t_i} = \langle A_2^{t_i} = \{Room\ 2, Hallway\}, N_2^{t_i} = 3\rangle\ ,$$
$$R_3^{t_i} = \langle A_3^{t_i} = \{Room\ 1, Hallway\}, N_3^{t_i} = 3\rangle\ .$$

Let x, y and z be the number of persons in $Room\ 1$, $Room\ 2$ and $Hallway$, respectively. A system of linear equations can be formed: $x + y = 4$, $y + z = 3$, and $x + z = 3$. Since $R_1^{t_i}$, $R_2^{t_i}$ and $R_3^{t_i}$ do not meet the reciprocity property, i.e., overlapping each other, by solving these linear questions, an adversary is able to infer **two persons in Room 1, two persons in Room 2 and one person in Hallway at time** t_i. (2) At time t_{i+1} (Fig. 1b), the sensor nodes report

$$R_1^{t_{i+1}} = \langle A_1^{t_{i+1}} = \{Room\ 1\}, N_1^{t_{i+1}} = 3\rangle\ ,$$
$$R_2^{t_{i+1}} = \langle A_2^{t_{i+1}} = \{Room\ 1, Room\ 2\}, N_2^{t_{i+1}} = 5\rangle\ ,$$
$$R_3^{t_{i+1}} = \langle A_3^{t_{i+1}} = \{Room\ 1, Hallway\}, N_3^{t_{i+1}} = 3\rangle\ .$$

For the same reason, the adversary can infer **three persons in Room 1, two persons in Room 2 and no person in Hallway at time** t_{i+1}. By combining the obtained knowledge from aggregate locations at two consecutive time instances t_i and t_{i+1} with the background knowledge of Alice standing in Hallway at time t_i, the adversary knows that Alice left Hallway and entered Room 1 at time t_{i+1}. Such knowledge leakage may lead to privacy threats. For instance, knowing that a person has visited a certain clinical room may result in knowing her health records.

It is important to note that not all overlapping aggregate locations will result in privacy risks. In terms of linear algebra, if and only if the rank of a system of linear equations formed by a certain set of overlapping aggregate locations does not equal the number of sensing areas in the overlapping aggregate locations, an adversary cannot deduce the number of objects in the sensing areas of the overlapping aggregate locations by solving the system of linear equations, i.e., publishing these overlapping aggregate locations is still safe. Nevertheless, the state-of-the-art cloaking algorithms [10], [11], [12], [18], [26], [27] do not take effort to ensure that all generated aggregate locations satisfy this condition. Our experimental results showed that an adversary can break the k-anonymity privacy requirement with a likelihood higher than 0.1 based on aggregate locations generated by them.

In this paper, we are motivated to propose a **RE**ciprocal protocol for generating k-anonymized **A**ggregate **L**ocations (**REAL** for short hereafter) in WSNs. The objectives of REAL are to (a) partition the whole system area into a set of aggregate locations such that each aggregate location covers at least k persons and does not overlap with any other aggregate locations, and (b) minimize the areas of aggregate locations in order to maximize their accuracy and thus provide location-based services with better quality. However, this optimal problem subject to the reciprocity property is NP-hard [2], [16], [32]. Thus, we design a heuristic in-network algorithm that guarantees the reciprocity property and attempts to generate aggregate location areas as small as possible.

In general, REAL addresses three key challenges. (1) **Self-organization**. To avoid the single point of attack in the centralized cloaking approaches [16], [28], [34], [44], sensor nodes are required to autonomously organize their sensing areas into a set of aggregate locations. To this end, we design a state transition process for each sensor node to interact with other nodes. (2) **Reciprocity property**. A locking mechanism is proposed to ensure that each sensor node is involved in the generation of only one aggregate location and is excluded from the generation of any other aggregate locations at any time. (3) **High accuracy**. Aggregate locations with higher accuracy can provide location monitoring services with better quality. We employ a time delay method for aggregate location generation to reduce the size of the aggregate location area.

We evaluate REAL by comparing its performance with the state-of-the-art in-network spatial cloaking techniques [10], [11], [12], [18], [26], [27] in our simulated experiments. The improvement of REAL is threefold. (1) REAL can guarantee the reciprocity property, but the current in-network spatial cloaking methods cannot, and thus, REAL substantially strengthens location privacy. (2) Using aggregate locations from REAL, the server can estimate the distribution of monitored persons more accurately and hence provide better monitoring services for users. (3) The communication and computational costs of REAL are significantly lower than the existing spatial cloaking methods, so REAL is energy-efficient, which is essential for prolonging the lifetime of WSNs [46].

Organization of this paper: The system model, protocol and implementation of REAL are described in Sections 2, 3 and 4, respectively. Sections 5 and 6 evaluate the performance of REAL. Section 7 highlights related work. Section 8 concludes this paper.

2 System Model

This section presents our problem, REAL's key entities (Fig. 2), communication and privacy models.

Problem definition. Given an anonymity level k, a set of moving objects $o_1, o_2, \ldots, o_L$ and a set of sensor nodes $s_1, s_2, \ldots, s_M$ with sensing areas $a_1, a_2, \ldots, a_M$, respectively, the goal of REAL is to cloak the whole system area into a set of k-anonymized aggregate

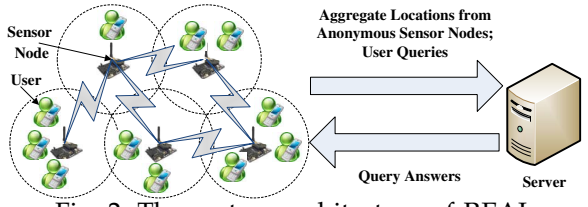


Fig. 2: The system architecture of REAL

locations $\Re = \{R_i = \langle A_i, N_i \rangle\}$ such that: (1) $\Re$ satisfies the reciprocity property, i.e., each $R_i = \langle A_i, N_i \rangle \in \Re$ covers at least k objects and does not overlap with any other $R_j = \langle A_j, N_j \rangle \in \Re$, formally $\forall i, N_i \geq k$ and $\forall i \neq j, A_i \cap A_j = \emptyset$. (2) $\Re$ minimizes the average size of all areas A_i in $\Re$ to maximize their accuracy and thus provide location-based services with better quality.

Sensor nodes. Sensor nodes are stationary after deployment, but routing paths may change over time due to node failure [3], [43]. In each reporting period, every sensor node is aware of its location and sensing area and responsible for determining the number of persons in its sensing area. All sensor nodes autonomously organize their sensing areas into a set of non-overlapping k-anonymized aggregate locations and report them to the server.

Server. The server collects k-anonymized aggregate locations from sensor nodes, estimates distribution of monitored persons using the spatial histogram method [12], and provides location-based services through answering aggregate queries from users, for instance, "what is the number of persons in a certain area?" The spatial histogram divides the whole monitored area into disjointed equal-sized grid cells and maintains an estimator of the number of objects within each grid cell. Further, only the system administrator can change the anonymity level k of the system by disseminating a message with a new value of k to all the sensor nodes.

Users. Users are the persons monitored by the system. They can also issue aggregate queries to the system via the sensor nodes. The server answers the queries based on the estimated object distribution.

Communication models. By maintaining a routing table, a sensor node knows how to communicate with others even if the network topology is changing due to node failure [3], [43]. Once a sensor node receives a message of any type, it immediately confirms the receipt by sending an acknowledgement message. Thus, if a message gets lost, the source sensor node will send it again until it receives the acknowledgement message [3], [43]. Sensor nodes use two communication paradigms: (1) *Broadcast*. All sensor nodes residing in the transmission range of a source node receive the broadcast message. (2) *Point-to-point* (P2P). There is only one destination node for the message being sent from a source node; the P2P communication can be implemented using multi-hop routing techniques [3]. In addition, our system (a) establishes a secure network channel for communication between sensor nodes to avoid internal network attacks like eavesdropping and malicious nodes [21] and a secure protocol to ensure data confidentiality, data authentication, data integrity and data freshness [36], and (b) employs anonymous communication techniques for communication between sensor nodes and the server [29]; hence, given a k-anonymized aggregate location R, the server only knows that R's sender is one of the sensor nodes within R.

Privacy model. First, through establishing secure network channels, the sensor nodes constitute a trusted zone in which they just behave as defined in our proposed REAL protocol. Second, through the anonymous communication techniques for communication between sensor nodes and a server [29], the server only knows that the sender of a k-anonymized aggregate location R is one of the sensor nodes within R, but cannot infer the exact identifier of the sender of R. Third, the system only allows sensor nodes to report k-anonymized aggregate locations to the server and these aggregate locations are publicly available. Lastly, an adversary is a user of the monitored system or a certain operator of the server who can arbitrarily analyze aggregate locations through the system terminal and the background knowledge (including the map layout of the system and the location and sensing area of each sensor node) in order to infer the location of a monitored person. The attacker model used by the adversary is defined in Section 5.

To prevent the adversary from inferring the number of persons in a certain sensing area, the REAL protocol partitions the whole system area into a set of cloaked areas such that each cloaked area covers at least k persons and does not overlap any other cloaked areas. As a result, the adversary cannot infer a person's exact location with any fidelity even if it has the background knowledge about the system.

3 The REAL Protocol

This section presents a state transition process, locking mechanism, and time delay mechanism to address the three challenges: *self-organization*, *reciprocity property*, and *high accuracy*, respectively.

3.1 State Transition for Self-Organization

We first introduce the basic concepts and then describe the process of the intra-sensor state transition.

3.1.1 Basic Concepts

Four states of sensor nodes. To meet the reciprocity property, the REAL protocol requires sensor nodes to autonomously cloak their sensing areas into a set of non-overlapping aggregate locations with high accuracy. To accomplish self-organization, each sensor node should follow the protocol and update its state accordingly. In our protocol, a sensor node s is in one of four states at any time, as described in TABLE 1. If s does not belong to any k-anonymized aggregate location, it is at the **ROAMER** state. Otherwise, if s owns an aggregate location R, s is at the **LEADER** state; otherwise, s is at the **FOLLOWER** state. In an

TABLE 1: Four states of a sensor node s

State	Necessary conditions
ROAMER	s does not belong to any aggregate location
FOLLOWER	s belongs to an aggregate location R but s is NOT R's owner
LEADER	s belongs to and owns an aggregate location R
END	s belongs to an aggregate location and s does not need any further collaboration with others

TABLE 2: Three message types for state transition

Type	Purpose
INVITE	To notify followers that they have been added into its aggregate location
JOIN	To notify a neighbor that the sender wants to join its aggregate location
FINISH	To notify a leader that the sender has transited into the **END** state

aggregate location, there is only one owner, called as a **leader** and other nodes are **followers**. Finally, s is at the **END** state when it is at the **FOLLOWER** or **LEADER** state but s does not need any further collaboration with other sensor nodes. It is important to note that each sensor node belongs to the exact one aggregate location at any time to avoid any overlapping relationship between aggregate locations; this also means that each node owns at most one aggregate location at any time.

Three message types for intra-sensor state transition. To cloak sensing areas into non-overlapping aggregate locations, sensor nodes have to collaborate with each other to update their own state. We design three message types to facilitate collaboration among sensor nodes, as defined in TABLE 2. At the start of a reporting period, all sensor nodes attempt to blur their sensing areas into k-anonymized aggregate locations. If a sensor node successfully creates an aggregate location, it immediately sends an INVITE message to each peer in this aggregate location through the point-to-point (P2P) communication. Reversely, if a sensor node fails to generate an aggregate location, it sends a JOIN message to one of its neighbors via the P2P communication in order to join an existing aggregate location. Furthermore, the neighbor node forwards the JOIN message to its leader via the P2P communication. Based on the INVITE and JOIN messages, a sensor node can be cloaked into an aggregate location in three ways: (1) it successfully creates an aggregate location, (2) it passively waits for an INVITE message, or (3) it actively joins into an existing aggregate location by sending a JOIN message to one of its neighbor nodes. When a sensor node transits into the **END** state, it immediately sends a FINISH message to its leader via the P2P communication. The FINISH message is necessary for a leader to know when it will not receive JOIN messages anymore and then it sends its aggregate location to the server.

Relations between states and messages. A sensor node only sends or receives certain (not all) types of messages based on its current state, as shown in Fig. 3, where the **ROAMER**, **FOLLOWER**, and **LEADER** are represented by **R**, **F**, and **L**, respectively. An INVITE message is sent and received by a sensor node at the **ROAMER** state (Fig. 3a). A JOIN message is sent by a sensor node at the **ROAMER** state to another node at the **FOLLOWER** or **LEADER** state. If the receiver is at the **FOLLOWER** state, the JOIN message is forwarded to its leader (i.e., at the **LEADER** state) (Fig. 3b). A FINISH message is only sent by a sensor node at the **FOLLOWER** state to its aggregate location's leader (i.e., at the **LEADER** state) (Fig. 3c).

3.1.2 Intra-Sensor State Transition

Fig. 4 presents the intra-sensor state transition diagram and the events triggering transitions.

At the ROAMER state. At the beginning of every reporting period, all sensor nodes are at the **ROAMER** state because none of them belongs to any aggregate location. We call a sensor node that attempts to create an aggregate location R as an **active node** for reference convenience. Initially, the active node generates R with itself (i.e., its sensing area and object count). The active node next iteratively conducts the *Peer Search Step* until R satisfies the k-anonymity privacy requirement or it is impossible for the active node to find k monitored objects to satisfy the privacy requirement. In the former case, the active node sends an INVITE message to other peer nodes in R (i.e., its followers) and transits itself into the **LEADER** state. However, in the latter case, the active node executes the *Aggregate Location Search Step*.

Peer Search Step: The active node broadcasts a request message to the neighbors of the last sensor node selected for the current aggregate location R and collects responses into a candidate set. Note that right after initialization, the active node is the last one selected node for R, so the active node just broadcasts a request message to its neighbors. Otherwise, the active node sends a request message to a peer lastly selected for R through the P2P communication and then the peer broadcasts the received request message to its neighbors. If a sensor node decides to reply to a received request, it sends a response message to the source active node via the P2P communication. After the active node receives responses, it determines a score for each sensor node in the candidate set. The score is defined as a ratio of the object count of the sensor node to the Euclidean distance between the sensor node and the active node [12]. The active node then selects the node with the highest score from the candidate set to R; if there are two or more nodes in the candidate set with the highest score, the active node will randomly pick up one of them. A candidate with the highest score usually means that it closes to the active node and its sensing area contains a large number of objects compared to the other candidates. Thus, the node with the highest score benefits the active node to restrict the area size of aggregate locations [12].

Aggregate Location Search Step: The active node broadcasts requests to its neighbors to enquire about

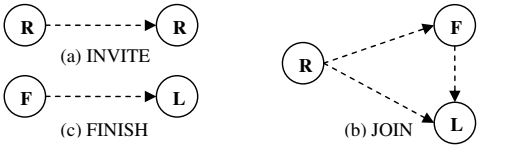


Fig. 3: Relations between states and message types

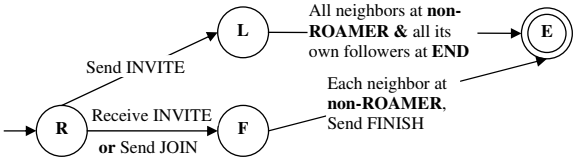


Fig. 4: The intra-sensor state transition diagram

their aggregate locations; the neighbors at the **non-ROAMER** state send a response message with the area size of their aggregate locations. The active node joins the aggregate location with the smallest area size to restrict the total area size by sending a JOIN message to the corresponding neighbor and transits itself into the **FOLLOWER** state. Finally, this neighbor forwards the JOIN message to its leader. During this step, the active node also responses to requests from other sensor nodes that are attempting to create a new aggregate location, and then watches out for an INVITE message from them. If it receives one, it is involved in the new aggregate location instead of joining an existing one, terminates the *aggregate location search step* in advance, and switches its state into the **FOLLOWER** state. Note that: (1) Creating a new aggregate location is much better than joining an existing one, since a new aggregate location only contains the necessary sensing areas whereas joining an existing one unnecessarily expands its cloaked area. (2) All sensor nodes in the candidate set not selected for some aggregate location in the *peer search step* should be released in time (i.e., unlocked in Section 3.2) such that one of these sensor nodes may get a chance to invite the active node to generate a new aggregate location.

At the FOLLOWER state. At this state, the sensor node performs two tasks. (1) When the node receives JOIN messages, it forwards them to its leader. (2) The node keeps checking the states of its neighbors. According to Fig. 3b and the *aggregate location search step*, a follower only receives JOIN messages from its neighbors at the **ROAMER** state. Thus, once the state of each of its neighbors changes to the **non-ROAMER** state, the node changes its state to the **End** state and sends a FINISH message to its leader.

At the LEADER state. At this state, the sensor node also performs two tasks. (1) When the node receives a JOIN message, the original sender is added to the node's aggregate location. (2) The node keeps tracking of the states of its neighbors and followers. According to Fig. 3b and the *aggregate location search step*, a leader only receives JOIN messages sent by its neighbors at the **ROAMER** state or forwarded by its followers. Thus, once all its neighbors are in the **non-ROAMER** state and all its followers are in the **END** state, the node changes its state to the **END** state because the node will not receive any JOIN message until a new reporting period starts. The node then sends its aggregate location to the server.

3.2 Locking Mechanism for Reciprocity Property

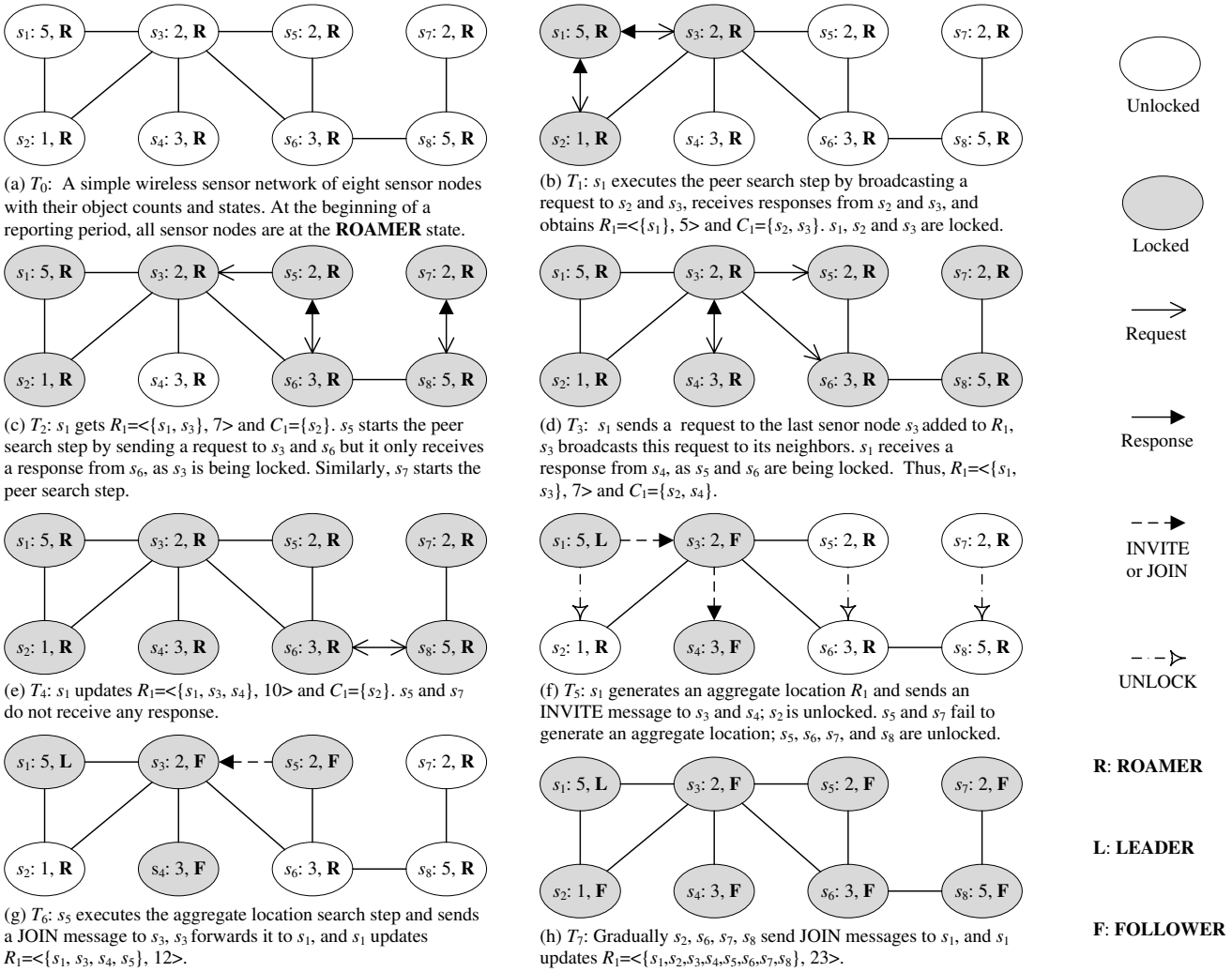
We have described how to generate k-anonymized aggregate locations. We now discuss how to avoid overlapping among aggregate locations to satisfy the reciprocity property. For example, when two neighbor sensor nodes send a request message to each other during the *peer search step* in Section 3.1.2, if they both respond to the other, their aggregate locations may overlap. The key idea of our solution is to require sensor nodes to use a locking mechanism that a *locked* sensor node is prohibited from responding to any request for certain situations until it is unlocked. For example, an active node locks itself before sending a request to its neighbors. Similarly, if some of these neighbors are currently unlocked, they have to lock themselves before replying to the active node. Note that a deadlock cannot occur even if some sensor nodes have crashed, since the remainder sensor nodes will be unlocked in a new reporting period.

There are two possible outcomes for the active node. (1) If it can generate an aggregate location R, it sends an INVITE message to each candidate peer selected for R. The active node and the selected peers will be kept locked until a new reporting period starts and changed to the **LEADER** and **FOLLOWER** states, respectively. For those candidate peers not selected for R, the active node sends an UNLOCK message to them through the P2P communication. They unlock themselves and remain in the **ROAMER** state. (2) If it fails to generate an aggregate location, it unlocks itself and all candidate peers by sending UNLOCK messages to them. The active node and all candidate peers remain in the **ROAMER** state.

Example. Consider a simple wireless sensor network given in Fig. 5a, where nodes and edges denote sensor nodes and their neighborhood relationships, respectively, and the figure and the letter along with a sensor node are its object count and current state, respectively. For discussion convenience, we further assume that the Euclidean distance between two sensor nodes is the number of edges in the shortest path between them, and the anonymity level is $k = 10$. We describe this example based on the eight snapshots of the network, from T_0 to T_7, as depicted in Fig. 5.

At T_0 (Fig. 5a): When a reporting period starts, all sensor nodes are *unlocked* and in the **ROAMER** state.

At T_1 (Fig. 5b): Sensor node s_1 initializes an aggregate location with itself $R_1 = \langle\{s_1\}, 5\rangle$ and performs the *peer search step*. s_1 locks itself and sends a request to its neighbors s_2 and s_3. Since s_2 and s_3 are *unlocked* and in the **ROAMER** state, they lock themselves (i.e., they will not respond to any request until they are unlocked by s_1) and respond to s_1. s_1 inserts them into its candidate set $C_1 = \{s_2, s_3\}$.



(a) T_0: A simple wireless sensor network of eight sensor nodes with their object counts and states. At the beginning of a reporting period, all sensor nodes are at the **ROAMER** state.

(b) T_1: s_1 executes the peer search step by broadcasting a request to s_2 and s_3, receives responses from s_2 and s_3, and obtains R_1=<{s_1}, 5> and C_1={s_2, s_3}. s_1, s_2 and s_3 are locked.

(c) T_2: s_1 gets R_1=<{s_1, s_3}, 7> and C_1={s_2}. s_5 starts the peer search step by sending a request to s_3 and s_6 but it only receives a response from s_6, as s_3 is being locked. Similarly, s_7 starts the peer search step.

(d) T_3: s_1 sends a request to the last senor node s_3 added to R_1, s_3 broadcasts this request to its neighbors. s_1 receives a response from s_4, as s_5 and s_6 are being locked. Thus, R_1=<{s_1, s_3}, 7> and C_1={s_2, s_4}.

(e) T_4: s_1 updates R_1=<{s_1, s_3, s_4}, 10> and C_1={s_2}. s_5 and s_7 do not receive any response.

(f) T_5: s_1 generates an aggregate location R_1 and sends an INVITE message to s_3 and s_4; s_2 is unlocked. s_5 and s_7 fail to generate an aggregate location; s_5, s_6, s_7, and s_8 are unlocked.

(g) T_6: s_5 executes the aggregate location search step and sends a JOIN message to s_3, s_3 forwards it to s_1, and s_1 updates R_1=<{s_1, s_3, s_4, s_5}, 12>.

(h) T_7: Gradually s_2, s_6, s_7, s_8 send JOIN messages to s_1, and s_1 updates R_1=<{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8}, 23>.

Fig. 5: Example of the locking mechanism for generating non-overlapping k-anonymized aggregate locations

At T_2 (Fig. 5c): s_1 computes the scores for candidate peers s_2 and s_3. Since the distances from s_1 to s_2 and s_3 are equal to one, the scores for s_2 and s_3 are 1 and 2, respectively. The candidate peer with the highest score (i.e., s_3) is selected for $R_1 = \langle\{s_1, s_3\}, 7\rangle$ and $C_1 = \{s_2\}$. At this step, s_5 also initializes an aggregate location with itself $R_5 = \langle\{s_5\}, 2\rangle$ and performs the *peer search step*. s_5 locks itself and sends a request to its neighbors s_3 and s_6. Since s_6 is *unlocked* and in the **ROAMER** state, s_6 responds to s_5's request; and hence, $C_5 = \{s_6\}$. However, because s_3 is *locked*, it neglects s_5's request. Similarly, s_7 starts the peer search step.

At T_3 (Fig. 5d): s_1 sends a request to s_3 which is the last one selected for R_1, s_3 broadcasts the request to its neighbors, i.e., s_4, s_5, and s_6. Only s_4 locks itself and responds to s_1, because s_5 and s_6 are being locked. s_1 then inserts s_4 into $C_1 = \{s_2, s_4\}$. For s_5, since there is only one peer s_6 in C_5, s_6 is selected for s_5's aggregate location R_5, and hence, $R_5 = \langle\{s_5, s_6\}, 5\rangle$ and $C_5 = \emptyset$.

At T_4 (Fig. 5e): For s_1, the score of s_4 is $3/2 = 1.5$ that is larger than that of s_2. Thus, s_4 is selected for R_1, i.e., $R_1 = \langle\{s_1, s_3, s_4\}, 10\rangle$ and $C_1 = \{s_2\}$. Although s_5 sends requests to the neighbors of s_6, but s_5 cannot receive any response. Hence, $C_5 = \emptyset$.

At T_5 (Fig. 5f): Since s_1's aggregate location R_1 meets the k-anonymity requirement, s_1 sends an INVITE message to s_3 and s_4 (i.e., the nodes selected for R_1). As a result, s_1 changes its state into **LEADER** while s_3 and s_4 change their state into **FOLLOWER** and they remain locked until a new reporting period starts. However, s_1 sends an UNLOCK message to the remaining peer in C_1, i.e., s_2. Since s_5's C_5 remains empty, it is impossible for s_5 to generate an aggregate location. Therefore, s_5 unlocks itself and sends an UNLOCK message to s_6. s_6 is unlocked again.

At T_6 (Fig. 5g): Since s_5 fails to generate an aggregate location, it executes the *aggregate location search step* and sends a JOIN message (at this point of time s_5 also locks itself) to its neighbor which is in the **LEADER** or **FOLLOWER** state (i.e., s_3). s_3 forwards the JOIN message to s_1. s_1 adds s_5 to its $R_1 = \langle\{s_1, s_3, s_4, s_5\}, 12\rangle$.

At T_7 (Fig. 5h): Similar to the case of s_5, s_6, s_7, and s_8 are gradually joined to s_1's aggregate location, i.e., $R_1 = \langle\{s_1, s_3, s_4, s_5, s_6, s_7, s_8\}, 23\rangle$.

3.3 Delay Mechanism for High Accuracy

As we have seen in Section 3.2, lock collisions may occur during the generation of aggregate locations.

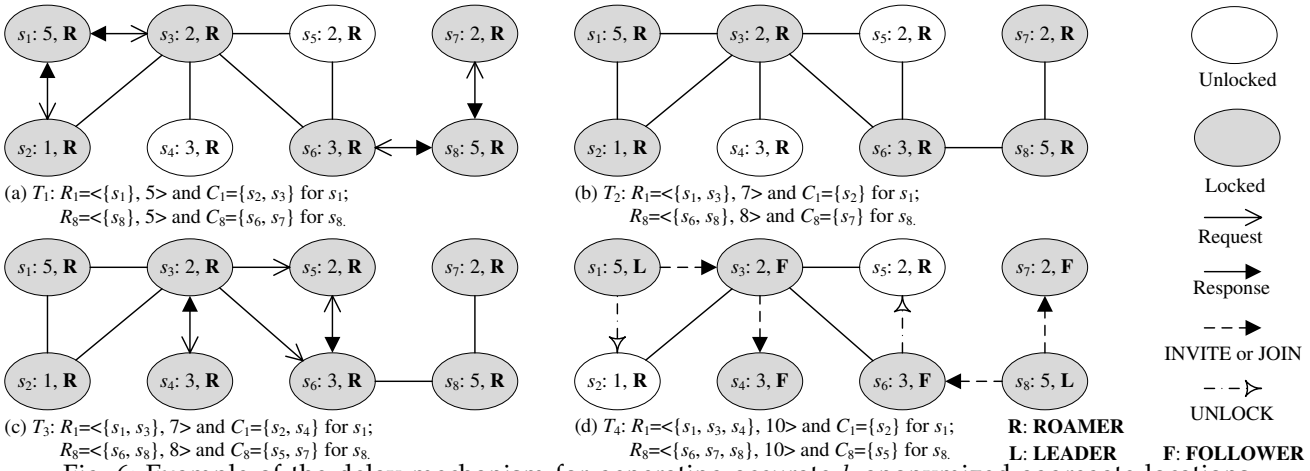


Fig. 6: Example of the delay mechanism for generating accurate k-anonymized aggregate locations

For example, at time T_2 (Fig. 5c), s_5 and s_7 generate their aggregate locations R_5 and R_7 with peers s_6 and s_8, respectively. Lock collisions occurred at time T_4 (Fig. 5e) when s_6 and s_8 send a request to each other. There will be very high possibility for lock collisions to take place if all sensor nodes attempt to generate their aggregate locations at the same time. Lock collisions not only increase communication and computational cost, but they also lead to aggregate locations with a large cloaked area size. For instance in Fig. 5h, at time T_7 if s_6 and s_8 attempt to generate aggregate locations at the same time, the lock collision will occur again when they send a request to each other; then s_6 and s_8 have to join s_1's aggregate location R_1. Likewise, s_5 and s_7 will also join s_1's R_1. Subsequently, the sensor network generates only one aggregate location consisting of all the sensing areas.

The location monitoring system requires accurate aggregate locations reported by sensor nodes to provide high-quality services. We should reduce the area size of the aggregate location to improve its accuracy. However, all of the existing approaches are essentially heuristic since the optimal k-anonymity is shown to be a NP-hard problem, even in the centralized systems [2], [16], [32]. To this end, we design a delay mechanism as a heuristic approach to minimize the aggregate location area size.

The main idea behind our delay mechanism is that a sensor node with a large object count has a higher priority to be a leader. Each sensor node conducts a *time delay step* that defers the start of the *peer search step* in order to reduce lock collisions. The delay interval for a sensor node relies on the number of objects residing in its sensing area. Clearly, if its object count satisfies the k-anonymity privacy requirement, it generates an aggregate location with only its sensing area and object count without any delay. This kind of aggregate locations has the smallest cloaked area size, since it only contains one sensing area. In general, a sensor node with a larger object count has a higher chance to generate an aggregate location with a small area size. Inspired by this observation, given the object count x and the anonymity level k, the delay interval is determined by

$$Interval(x) = \begin{cases} T(k-x)/(2k), & x < k; \\ 0, & x \geq k; \end{cases} \quad (1)$$

where T is the time of a reporting period. According to Equation (1), a sensor node with a smaller object count has to be delayed for a longer time period.

Example. We use our running example in Fig. 5a to illustrate our delay mechanism depicted in Fig. 6.

At T_1 (Fig. 6a): Since s_1 and s_8 have the largest object count, they finish the delay earlier than other peers and start generating aggregate location at the same time (based on Equation (1)). Hence, s_1 and s_8 construct their aggregate locations $R_1 = \langle\{s_1\}, 5\rangle$ and $R_8 = \langle\{s_8\}, 5\rangle$, and candidate sets $C_1 = \{s_2, s_3\}$ and $C_8 = \{s_6, s_7\}$, respectively.

At T_2 (Fig. 6b): s_1 and s_8 select peers for their aggregate locations, i.e., $R_1 = \langle\{s_1, s_3\}, 7\rangle$ $C_1 = \{s_2\}$ for s_1 and $R_8 = \langle\{s_6, s_8\}, 8\rangle$ and $C_8 = \{s_7\}$ for s_8.

At T_3 (Fig. 6c): s_1 finds a new peer s_4 through s_3 that is the last node added to R_1, i.e., $R_1 = \langle\{s_1, s_3\}, 7\rangle$ and $C_1 = \{s_2, s_4\}$. Likewise, s_8 finds s_5 through s_6, i.e., $R_8 = \langle\{s_6, s_8\}, 8\rangle$ and $C_8 = \{s_5, s_7\}$.

At T_4 (Fig. 6d): s_1 and s_8 select peers for their aggregate locations, i.e., $R_1 = \langle\{s_1, s_3, s_4\}, 10\rangle$ and $C_1 = \{s_2\}$ for s_1; $R_8 = \langle\{s_6, s_7, s_8\}, 10\rangle$ and $C_8 = \{s_5\}$ for s_8. Since the aggregate locations of s_1 and s_8 contain at least $k = 10$ objects, they issue UNLOCK messages to unlock the peers in their candidate sets.

Since none of s_2 or s_5's neighbors is at the **ROAMER** state, they execute the *aggregate location search step* to join one of their own neighbors' aggregate locations. As shown in Fig. 6, the delay mechanism reduces the number of lock collisions and generates two more accurate aggregate locations with smaller cloaked area sizes (e.g., $R_1 = \langle\{s_1, s_2, s_3, s_4\}, 11\rangle$ and $R_8 = \langle\{s_5, s_6, s_7, s_8\}, 12\rangle$), instead of having only one aggregate location with all the sensing areas in Fig. 5h.

4 Implementation and Analysis

In this section, we describe the implementation of our REAL protocol and prove its correctness.

Algorithm 1 The REAL protocol

```
1: function REAL (Sensor node s, Anonymity level k)
2: state ← ROAMER
3: isLocked ← false
4: Invoke Search-Thread(s, k) (Algorithm 2)
5: while state ≠ END do
6:     if state = ROAMER then {The roamer branch}
7:         if receive a request from a peer p then
8:             if isLocked = false then
9:                 isLocked ← true
10:                Send a response with s.ID, s.Area, s.count to p
11:            else
12:                Drop the request
13:            end if
14:        end if
15:        if receive an INVITE message from a peer p then
16:            s.leader ← p
17:            state ← FOLLOWER
18:            Terminate Search-Thread
19:        end if
20:        if receive an UNLOCK message then
21:            isLocked ← false
22:        end if
23:    else if state = FOLLOWER then {The follower branch}
24:        if receive a JOIN message from a peer p then
25:            Forward the JOIN message to s's leader, i.e., s.leader
26:            Return s.leader to p
27:        end if
28:        if none of s's neighbors is at ROAMER then
29:            Send a FINISH message to s.leader
30:            state ← END
31:        end if
32:    else if state = LEADER then {The leader branch}
33:        if receive a JOIN message from an original peer p then
34:            Send the s.leader (i.e., s.ID) to p
35:            R ← R ∪ {p}
36:        end if
37:        if none of s's neighbors is at ROAMER and s receives a FINISH
           message from each peer in R then
38:            Area ← the sensing areas of the sensor nodes in R
39:            N ← the total number of objects in R
40:            Send R = ⟨Area, N⟩ to the server
41:            state ← END
42:        end if
43:    end if
44: end while
```

4.1 Implementation

Algorithm 1 outlines the REAL protocol where the sensor nodes execute the REAL protocol for every reporting period to generate their k-anonymized aggregate locations and send them to the server.

Initially, all sensor nodes are at the **ROAMER** state and unlocked (Lines 2 and 3 in Algorithm 1). To increase system efficiency, the sensor node invokes an asynchronous sub-thread, called Search-Thread (Line 4 in Algorithm 1), while the main thread (i.e., Algorithm 1) consists of three main branches (determined by the sensor node's current state), namely, *roamer*, *follower*, and *leader* branches, to process received messages until its state is changed to the **END** state (Lines 5 to 44 in Algorithm 1).

The roamer branch in Algorithm 1. A sensor node at the **ROAMER** state processes requests, INVITE and UNLOCK messages (Lines 7 to 22 in Algorithm 1). (1) When it receives a request from a peer p, it checks its lock's state. If $isLocked$ is "false", it sets the value to "true" and sends a response to p; otherwise, it drops the request. (2) When it receives an INVITE message from p, it joins p's aggregate location (i.e., p becomes its leader), changes its state to **FOLLOWER** based on Fig. 4, and terminates the execution of Search-Thread. (3) When it gets an UNLOCK message, it sets $isLocked$ to "false" so as to reply requests hereafter.

Algorithm 2 The search-thread function

```
1: function Search-Thread (Sensor node s, Anonymity level k)
2: if s.count < k then {The time delay step}
3:     Wait for a time delay based on Equation (1)
4: end if
5: Wait until isLocked = false and resume execution
6: isLocked ← true {The peer search step}
7: R ← {s}
8: while the total number of objects in R < k do
9:     Send a request to each neighbor of the last node added to R
10:    Add responses into the candidate set C
11:    if C = ∅ then break
12:    p ← the peer with the largest score in C
13:    R ← R ∪ {p}
14:    C ← C − {p}
15: end while
16: if the total number of objects in R ≥ k then
17:    Send an INVITE message with s.ID to each peer in R
18:    s.leader ← s
19:    state ← LEADER
20: else {The aggregate Location search step}
21:    isLocked ← false
22:    Wait until a neighbor p is not at ROAMER and isLocked = false
       (isLocked may be set to true in Algorithm 1)
23:    isLocked ← true
24:    Send a JOIN message with s.ID, s.Area, and s.count to p
25:    Wait for the information of p's leader (i.e., p.leader)
26:    s.leader ← p.leader
27:    state ← FOLLOWER
28: end if
```

The follower branch in Algorithm 1. A sensor node at the **FOLLOWER** state processes JOIN messages (Fig. 3) and keeps track of the states of its neighbors (Lines 24 to 31 in Algorithm 1). (1) When the node receives a JOIN message from a peer p, it forwards the message to its leader, and returns its leader's ID to p. (2) Once the node detects that each of its neighbors is *not* at the **ROAMER** state, it sends a FINISH message to its leader, and transits its state into **End** (Fig. 4).

The leader branch in Algorithm 1. A sensor node at the **LEADER** state performs two tasks (Lines 33 to 42 in Algorithm 1). (1) When it receives a JOIN message from an original peer p, it adds p to its aggregate location R (Fig. 3). (2) After it detects that (a) none of neighbors is at the **ROAMER** state and (b) it receives a FINISH message from each of its followers, it sends R to the server and updates its state to **END** (Fig. 4).

Algorithm 2 describes the Search-Thread function that focuses on generating aggregate locations. The Search-Thread function consists of three steps.

The time delay step in Algorithm 2. At the beginning, to increase accuracy and reduce lock collisions, each sensor node sleeps for a certain time period based on its object count using Equation (1) (Line 3). After it wakes up, if it has responded to a request in Algorithm 1 during the time delay period, it is locked; therefore, it cannot proceed to the next step until it is unlocked by some peer (Line 5).

The peer search step in Algorithm 2. The sensor node first locks itself and initializes an aggregate location R with its sensing area and object count (Lines 6 and 7). If R does not meet the k-anonymity privacy requirement, it sends a request to each neighbor of the last sensor node added to R and adds the collected responses into its candidate set C (Lines 9 and 10). If C is not empty, a peer in C with the largest score

is removed from C and added to R (Lines 12 to 14). If R contains at least k objects (i.e., R satisfies the k-anonymity privacy requirement), the sensor node sends an INVITE message to each peer in R, becomes a leader for its R, and changes its state to **LEADER** (Lines 17 to 19). However, if R contains less than k objects, the sensor node proceeds to the next step.

The aggregate location search step in Algorithm 2. The sensor node first sets $isLocked$ to "false" (Line 21) to allow Algorithm 1 to respond requests from other peers. Meanwhile, it tries to join its neighbor's aggregate location. Once it detects that a neighbor is in the **non-ROAMER** state and $isLocked$ is "false", it locks itself, sends a JOIN message to the neighbor, and changes its state to **FOLLOWER** (Lines 22 to 27).

Note that the Search-Thread function can be terminated in advance by Algorithm 1 if the sensor node at the **ROAMER** state receives an INVITE message.

4.2 Proof of Correctness

Theorem 1. In REAL, the lock mechanism only allows each sensor node to belong to at most one k-anonymized aggregate location at any time.

Proof: There are two cases for each sensor node for a reporting period. (1) *A sensor node s has not responded to any request and s attempts to generate its aggregate location*. s first locks itself and then searches the network for other sensor nodes. When s is locked, it cannot respond to any request; and thus, s must not participate in the aggregate location generation of any other nodes. (2) *A sensor node s has responded to a request from a peer p.* Since s must lock itself before it responds to p, s is kept locked until it receives an UNLOCK message from p as p already gave up to form its aggregate location, or it will never be unlocked as it receives an INVITE message to join p's k-anonymized aggregate location. Since s cannot respond to any other requests until it is unlocked, s must engage in at most one peer's aggregate location generation. Therefore, each sensor node belongs to at most one k-anonymized aggregate location at any time. □

Theorem 2. If the total number of objects in the system is equal to or larger than the anonymity level (i.e., k), each sensor node belongs to one k-anonymized aggregate location by the end of the REAL protocol.

Proof: Because the total number of objects is not less than k, there exists at least one sensor node that can generate a k-anonymized aggregate location. From the *aggregate location search step*, all other sensor nodes at the **ROAMER** state will either receive an INVITE message from a peer p and join p's aggregate location or send a JOIN message to one of its neighbors at the **non-ROAMER** state to join its aggregate location. Therefore, every search node eventually belongs to one k-anonymized aggregate location. □

5 Experimental Evaluation

Here we describe an attacker model, evaluated techniques, performance metrics, and simulation settings.

Attacker model. To evaluate the privacy protection of the REAL protocol, we simulate an adversary attempting to infer the number of object residing in a sensing area using the aggregate locations reported from sensor nodes. Note that if the adversary is unable to infer the exact object count of the sensing area, it cannot infer the location information for an individual object [12]. Deriving the number of objects in a sensing area is equivalent to solving a system of linear equations in mathematics. The standard model for solving a system of large-scale linear equations is LU decomposition based on Gaussian elimination [37], which is used as our attacker model.

Evaluated techniques. The in-network protocols implemented in our experiments are listed below.

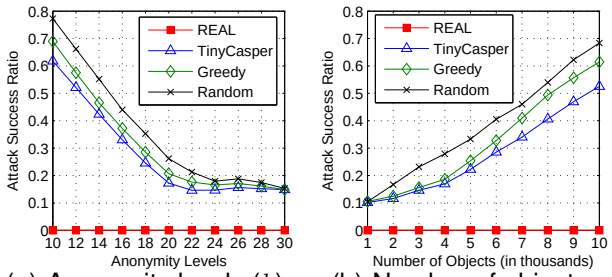
Random [18]: A sensor *randomly* chooses neighbors and *independently* generates an aggregate location.

Greedy [7], [10], [23], [26], [27]: A sensor *greedily* searches neighbors with the largest number of objects and also *independently* generates an aggregate location.

TinyCasper [11], [12]: This protocol is greedy-enhanced. That is, the cloaked area obtained by Greedy is iteratively refined based on extra communication among the sensor nodes until it reaches the minimal possible size. Specifically in TinyCasper: (1) A sensor node s extends its initial cloaked area A from Greedy into the search space S with the center at s and the radius as the maximum distance between sensor nodes residing in the cloaked area A. (2) The sensor node s communicates with other sensor nodes in the search space S to collect their sensing areas and object counts. (3) The sensor node s applies a progressive refinement approach for computing the minimal cloaked area by searching different combinations of the sensor nodes in S. Note that TinyCasper utilizes two optimization techniques to avoid enumerate all possible combinations in order to reduce computational cost. (a) The possible combinations of the sensor nodes in S are organized as a lattice structure. (b) If a combination C constitutes a cloaked area that is the same or larger than the current minimal cloaked area, other combinations that contain C at the higher levels of the lattice structure are pruned.

REAL: The sensor nodes *dependently* organize their sensing areas into a set of non-overlapping and high accurate k-anonymized aggregate locations based on the state transition process, locking mechanism, and time delay mechanism.

Performance metrics. We apply four popular performance metrics [11], [12]. (1) *Attack success ratio*. This metric measures the resilience of the k-anonymity protocols to the attacker model. Let M be the total number of sensing areas in an experiment, and m be the number of sensing areas that are derived with less than k objects by the attacker model. The ratio of successful attacks is measured as m/M. (2) *Query answer error*. This metric measures the quality of query answers, i.e., the accuracy of aggregate locations. Let



(a) Anonymity levels (k) (b) Number of objects

Fig. 7: Comparison of attack success ratios

$\hat{N}$ be the estimated number of objects within a query region using the spatial histogram, and N be the actual number of objects within the query region. If $N > 0$, the query answer error is $|\hat{N} - N|/N$; otherwise, the error is $\hat{N}$. (3) *Average number of messages*. This metric measures the communication cost by calculating the average number of messages sent by each sensor node per reporting period. (4) *Average execution time*. This metric measures the computational cost (i.e., average execution time) needed for each sensor node to be part of an aggregate location per reporting period.

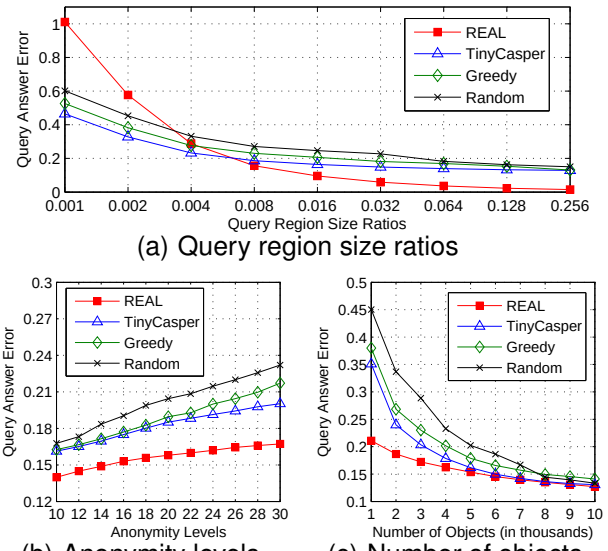
Simulation settings. Unless otherwise specified, we simulate 30×30 sensor nodes ($M = 900$) that are uniformly distributed in a 600×600 system space. Each sensor node monitors a 20×20 space and has an average of 5 neighbors. The performance of the k-anonymity protocols is tested with extensive experiments of various anonymity levels k (from 10 to 30) and numbers of moving objects (from 1,000 to 10,000) that freely roam around the system space at a random speed within a range of [0, 5] space unit(s) per time unit. Each experiment is evaluated for 100 reporting periods with different distributions of objects to make experimental results statistically significant and generalizable. For each experiment, we also issue 1,000 range queries whose query region size is specified by a ratio of the query region area to the system area, namely, a query region size ratio. The query region size ratio is from 0.001 (≈1/900) to 0.256.

6 Experimental Results

This section analyzes the experimental results regarding privacy protection, quality of aggregate locations, and communication and computational costs.

6.1 Attack Success Ratio

Fig. 7 depicts the resistance to the attacker model regarding the anonymity level and number of objects. The attack success ratios of TinyCasper, Greedy and Random decrease with larger anonymity levels, while they increase when there are more objects. They suffer from a high attack success ratio, i.e., above 10%, since they are not dedicated to evade the overlapping between k-anonymized aggregate locations. Conversely, REAL is completely resistant to the attacker model, with a zero attack success ratio. The reason is that REAL partitions the whole monitored area into a set of disjointed aggregate locations. As a result, no matter how we change the anonymity level and the number



(a) Query region size ratios

(b) Anonymity levels (c) Number of objects

Fig. 8: Comparison of query answer errors

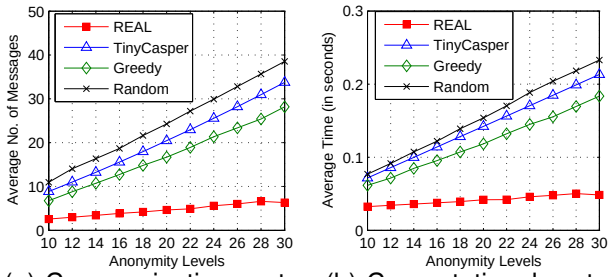
of objects, REAL is always immune to the attack model. All these results validate our REAL can protect location privacy rigorously and an adversary cannot infer personal location information with any fidelity.

6.2 Query Answer Error

Fig. 8 depicts the query answer errors of REAL, TinyCasper, Greedy and Random respecting various query region size ratios, anonymity levels, and numbers of objects. In Fig. 8a, the query region size ratio is the ratio of a query region area to the entire system area. When users issue range queries with a small query region size, the four protocols avoid the privacy leakage by providing low-quality location-based services, so an adversary could not use the query answer to track users. For example, at the ratio of 0.001 that corresponds to the size of a sensor node's sensing area, REAL generates the query error twice as large as TinyCasper, Greedy and Random, which indicates REAL doubles the strictness of privacy protection. On the other hand, for larger query regions, all protocols provide query answers with higher quality, but obviously REAL performs much better than the others. In sum, REAL promises the strictest privacy protection for small query regions and the most accurate answers for large query regions. This is an important feature to protect personal location privacy.

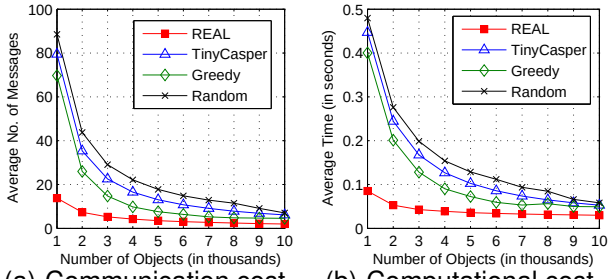
In Fig. 8b, the query answer error steadily gets higher with the increase of the anonymity level, because in general stricter anonymity levels result in the larger cloaked areas that lead to more inaccurate aggregate locations. By contrast to TinyCasper, Greedy and Random, REAL always maintains a lower level of query answer errors; especially, when the anonymity level becomes stricter, the gap gets larger.

In Fig. 8c, the query answer error gradually declines as the number of objects increases. The reason is that when there are more objects, sensor nodes can generate more accurate aggregate locations with smaller cloaked areas for the server to estimate the



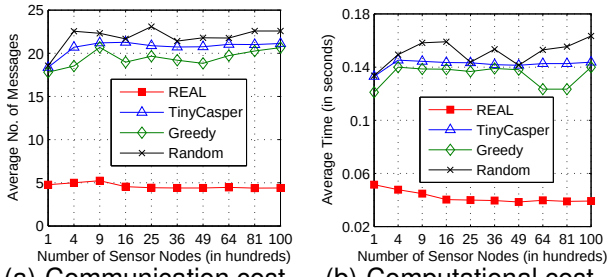
(a) Communication cost (b) Computational cost

Fig. 9: Costs regarding anonymity levels



(a) Communication cost (b) Computational cost

Fig. 10: Costs regarding numbers of objects



(a) Communication cost (b) Computational cost

Fig. 11: Costs regarding numbers of sensor nodes

distribution of objects. Interestingly, when the number of objects is smaller, REAL performs much better than TinyCasper, Greedy and Random; when there are many objects, the four methods perform similarly.

6.3 Communication and Computational Costs

Fig. 9 depicts the communication and computational costs of the four protocols with various anonymity levels. As the k-anonymity privacy requirement gets stricter, sensor nodes need to collaborate with more peers to blur their sensing areas into aggregate locations; thus, the communication and computational costs of all the protocols increase. Promisingly, it is observed that the increment is slight for REAL but steep for TinyCasper, Greedy and Random. This result shows that REAL is more scalable to strict k-anonymity privacy requirements than TinyCasper, Greedy and Random.

Fig. 10 depicts the communication and computational costs regarding various numbers of objects. When the number of objects increases from 1,000 to 10,000, the communication and computational costs of the four protocols decrease dramatically at first and then remain steady. This is because less cooperation among sensor nodes and cloaking computation are required to generate aggregate locations when there are many objects in the system. Importantly, the results show that REAL needs less communication and computational costs than TinyCasper, Greedy and Random, especially for the system with less objects.

Fig. 11 depicts the scalability of the four protocols, where we enlarge the system space by increasing the number of sensor nodes from 10×10 to 100×100. Obviously, the communication and computational costs are slightly affected by the number of sensor nodes, because a sensor node only blurs its sensing area with nearby peers, independent of the system space size.

6.4 Effect of Object Speed and Node Connectivity

Fig. 12 gives the performance of REAL, TinyCasper, Greedy and Random regarding increasing the maximum object mobility speed from [0, 5] and [0, 30]. The results show that increasing the object mobility speed only has little effect on the attack success ratio, query answer error, and communication and computational costs of the four protocols. Our explanation is that: the four protocols are only concerned with snapshot object locations in each reporting period and they have not considered the dependency between continuous object location updates in different reporting periods; hence, they are independent of the mobility speed of objects. Importantly, REAL is consistently better than the other protocols regarding various mobility speeds.

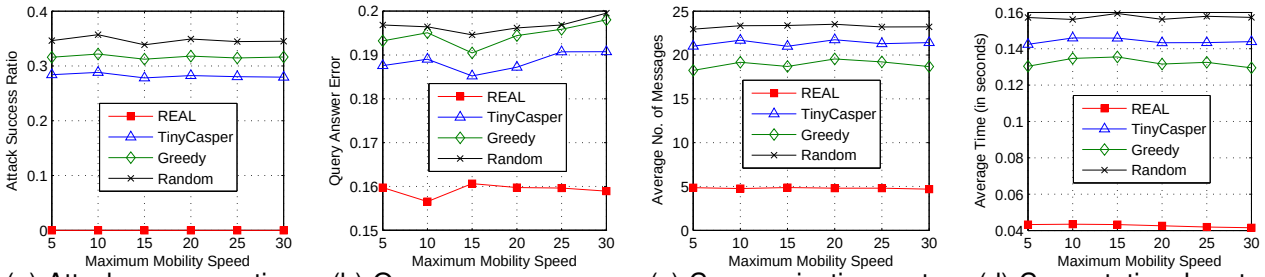
Fig. 13 depicts the performance of the four protocols about changing the neighbor connectivity among sensor nodes from 2 to 10 neighbors on average. As the connectivity increases, REAL still remains a zero attack success ratio, its query answer error decreases since it is able to select appropriate neighbors to generate more accurate aggregate locations with the smaller cloaked areas, and its communication and computational cost rises slowly since sensor nodes need to collaborate with more neighbors. In contrast, TinyCasper, Greedy and Random maintain a high level of attack success ratio, their query answer error increases gradually, and their communication and computational costs incline more quickly.

6.5 Discussion

Here we discuss the most important and general findings and their reasons in all experiments as follows.

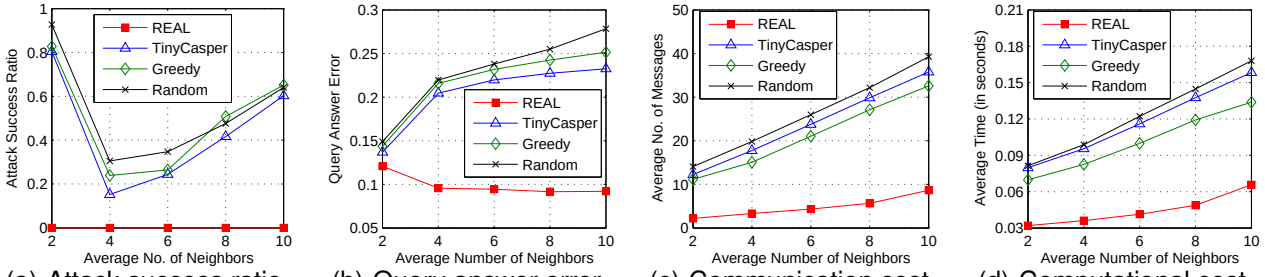
Random. Random is inferior to REAL, TinyCasper and Greedy, since in Random to create an aggregate location, each sensor node chooses neighbors in a very simple way, i.e., the random way. Subsequently, the sensor node fails to find appropriate neighbors to reduce the area size of the aggregate location and requires collaboration with more neighbors, which causes the higher attack success ratio, query answer error, and communication and computational costs.

Greedy. By using a heuristic way in neighbor search, i.e., each sensor node greedily selects neighbors with the largest number of objects to generate an aggregate location, Greedy can quickly discover appropriate neighbors to reduce the cloaked area size. Accordingly, Greedy decreases the attack success



(a) Attack success ratio (b) Query answer error (c) Communication cost (d) Computational cost

Fig. 12: Effect of mobility speeds of objects



(a) Attack success ratio (b) Query answer error (c) Communication cost (d) Computational cost

Fig. 13: Effect of changes in neighbor connectivity of sensor nodes

ratio, query answer error, and communication and computational costs compared to Random.

TinyCasper. TinyCasper enhances Greedy by iteratively refining the cloaked area from Greedy based on extra communication among the sensor nodes until it reaches the minimal possible size. Hence, TinyCasper reduces the attack success ratio and query answer error but needs more communication and computational cost, in comparison to Greedy.

REAL. Our REAL requires the sensor nodes to collaborate with each other based on the state transition process and always exhibits the best performance. Opposed to Random, Greedy and TinyCasper, the improvement of REAL is threefold. (1) REAL reduces the attack success ratio to zero, since REAL uses the locking mechanism to create aggregate locations that meet the reciprocity property and are completely resistant to the attacker model. (2) REAL reduces the query answer error because of two reasons. (a) The reciprocity property of aggregate locations removes the difficulty for the spatial histogram to accurately estimate the actual number of objects residing in an overlapping area. (b) By utilizing the time delay mechanism, REAL can alleviate lock collisions to generate aggregate locations with a smaller cloaked area size so as to provide location monitoring services with better quality. (3) REAL saves the communication and computational cost, i.e., the sensor nodes consume less power. Thus, REAL is energy-efficient, which is crucial for prolonging the lifetime of WSNs [46] and attributed to three reasons: (i) In REAL, most sensor nodes with fewer objects passively wait for an INVITE message to be included into an aggregate location, which only needs little communication and computation. (ii) In each reporting period, only the leaders of aggregate locations are required to communicate with the server. (iii) REAL alternately performs peer search and peer selection in the *peer search step* rather than executing two separate steps as in TinyCasper.

Privacy vs. utility. The anonymity level k achieves a trade-off between the strictness of privacy protection and the quality of monitoring services. A smaller k means less privacy protection but better monitoring services, because aggregate locations with smaller cloaked areas will be reported from sensor nodes; in contrast, a larger k results in aggregate locations with a larger cloaked area, which strengthens privacy protection but weakens the quality of monitoring services (Fig. 8b). Our system can avoid the privacy leakage by providing low-quality services for small query regions that the adversary may use to track users and high-quality services for larger query regions (Fig. 8a). Thus, REAL is not suitable for the monitoring system that needs high-quality services even for small query regions but relaxes the privacy requirement; in this case, TinyCasper, Greedy or Random can be applied.

7 Related Work

Privacy has attracted a lot of attention in data protection [35], information retrieval [40], and especially location-based services. Approaches for preserving location privacy include enforcing privacy policies [41], anonymizing identities of data [15], location obfuscation [31], [45], space transformation [17], differential privacy techniques [47], and spatial cloaking. Among these approaches, only the spatial cloaking technique can provide aggregate location information to the server and strive for a balance between privacy protection and the quality of services by tuning privacy requirements, e.g., k-anonymity. There are mainly four reasons. (1) Both privacy policy enforcement and identity anonymization approaches cannot prevent internal data thefts or inadvertent disclosure. (2) The location obfuscation approach reports n different locations with only one exact location and thus cannot

provide high-quality monitoring services due to a large amount of false location information. (3) The space transformation approach maps the location information into another space but still reveals the monitored object's exact location information in the mapped space; hence, this technique fails to provide privacy-preserving monitoring services. (4) The large noise added by the differential privacy techniques usually dominates the true object counts in a sensing area, which will severely deteriorates the quality of monitoring services. Therefore, we apply the spatial cloaking technique to preserve the monitored object's location privacy in our location monitoring system.

Based on system architecture, current spatial cloaking techniques for k-anonymity can be classified into centralized [5], [16], [25], [28], [34], [44], distributed [8], [19], [48], and peer-to-peer [7], [10], [11], [12], [18], [23], [26], [27]. (1) **Centralized.** The centralized approaches employ a central server that performs the aggregation on the behalf of individuals, e.g., users or sensor nodes. However, they suffer from two severe drawbacks: (a) All sensor nodes must trust the central aggregation server which is a single point of attack and poses a serious security threat. For example, if the server is compromised by an attacker, the history of all user movements may be revealed. (b) The server may become bottleneck since it must handle frequent location updates as users move. (2) **Distributed.** The distributed methods assume that there exist local infrastructures instead of using a global server. For instance, the techniques [8], [19] utilize base stations for users to communicate with each other and the study [48] collaboratively uses multiple servers and a third party to learn whether there are at least k persons in a certain area. None of these distributed methods is applicable to WSNs, in which there do not exist any local servers to assist sensor nodes to cloak their sensing areas into aggregate locations. (3) **Peer-to-peer (P2P).** The P2P protocols do not depend on any global or local servers and require sensor nodes to collaborate with one another to organize their sensing areas into aggregate locations. The P2P protocols can be grouped into *greedy*, *greedy-enhanced* and *random*. (i) *Greedy.* Most current protocols [7], [10], [23], [26], [27] use a greedy approach to find a cloaked area; in each step the greedy approach searches for a neighbor with the largest score calculated by a predefined function. (ii) *Greedy-enhanced.* In TinyCasper [11], [12] the cloaked area obtained by the greedy approach is iteratively refined based on extra communication among the sensor nodes until it reaches the minimal possible size. (iii) *Random.* The study [18] employs a random method to search neighbors rather than using the greedy approach. Although these P2P protocols have been used for WSNs, they cannot satisfy the reciprocity property, since a sensor node may be involved in more than one aggregate location due to the lack of a locking mechanism.

In this paper, that we partition a whole system area into a set of non-overlapping k-anonymized aggregate locations is a distributed clustering problem by nature. In WSNs, there are a few distributed clustering algorithms [1], [38]. They are proposed for various objectives including energy saving [6], [14], [24], [46], connectivity [4], management [33], fault tolerance [13], and load balancing [8], [9]. However, to the best of our knowledge, there is no research on distributed clustering algorithms for location privacy. These distributed clustering algorithms determine a set of cluster heads that are selected randomly from sensor nodes or are assumed as the special nodes with richer resources than the ordinary sensor nodes, allow the overlapping among clusters, and do not restrict the size of each cluster including the upper bound and the lower bound. Thus, none of the clustering algorithms can be applied to our problem for the following reasons: (1) In the context of privacy protection, our problem considers the k-anonymity requirement (the lower bound constraint) that is a much stricter constraint than the constraints of existing distributed clustering algorithms. (2) Our problem does not allow any overlapping between clusters (i.e., aggregation locations) to avoid location privacy breaches. (3) Our problem also aims at minimizing the spatial area size of a cluster (the upper bound constraint) to provide accurate aggregate locations. (4) Our problem chooses the sensor node with more objects as a leader (i.e., cluster head) with a higher probability, which is essentially different from the random selection method or the pre-assignment to the special sensor nodes with richer resources in the distributed clustering algorithms.

Our location privacy-preserving problem is also different other privacy related problems include: (a) source location privacy that hides the sender's location and identity, (b) aggregate data privacy that preserves the privacy of the sensor node's aggregate readings during transmission, (c) data storage privacy that hides the data storage location, and (d) query privacy that avoids disclosing the personal interests.

8 Conclusion

In this paper, we proposed the REAL protocol for privacy-preserving location monitoring services in WSNs. We defined an attack model that leads to a privacy breach in existing protocols because they generate overlapping aggregate locations. To avoid this privacy breach, REAL satisfies the reciprocity property by generating non-overlapping k-anonymized aggregate locations. In REAL, we designed the state transition process to accomplish self-organization among sensor nodes, the locking mechanism to guarantee the reciprocity property, and the delay mechanism to improve the accuracy of aggregate locations. By comparing with the state-of-the-art solutions, the experimental results show that REAL protects location privacy, provides more accurate query answers and saves communication and computational costs.

REFERENCES

[1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, 2007.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, 2005.

[3] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11(6):6–28, 2004.

[4] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE TCOM*, 29(11):1694–1701, 1981.

[5] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *WWW*, 2008.

[6] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM*, 2003.

[7] J. Bao, H. Chen, and W.-S. Ku. PROS: A peer-to-peer system for location privacy protection on road networks (Demo). In *ACM SIGSPATIAL*, 2009.

[8] P. Belsis and G. Pantziou. A k-anonymity privacy-preserving approach in wireless medical monitoring environments. *Personal and Ubiquitous Computing, accepted to appear*, 2012.

[9] H. Chan and A. Perrig. Ace: An emergent algorithm for highly uniform cluster formation. In *EWSN*, 2004.

[10] J. Chen, H. Xu, and L. Zhu. Query-aware location privacy model based on p-sensitive and k-anonymity for road networks. In *Internet of Things*. 2012.

[11] C.-Y. Chow, M. F. Mokbel, and T. He. TinyCasper: A privacy-preserving aggregate location monitoring system in wireless sensor networks (Demo). *ACM SIGMOD*, 2008.

[12] C.-Y. Chow, M. F. Mokbel, and T. He. A privacy-preserving location monitoring system for wireless sensor networks. *IEEE TMC*, 10(1):94–107, 2011.

[13] M. Demirbas, A. Arora, and V. Mittal. FLOC: A fast local clustering service for wireless sensor networks. In *DIWANS*, 2004.

[14] P. Ding, J. Holliday, and A. Celik. Distributed energy-efficient hierarchical clustering for wireless sensor networks. In *IEEE DCOSS*, 2005.

[15] J. Freudiger and et al. Non-cooperative location privacy. *IEEE TDSC*, 10(2):84–98, 2013.

[16] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE TMC*, 7(1):1–18, 2008.

[17] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *ACM SIGMOD*, 2008.

[18] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MOBIHIDE: A mobile peer-to-peer system for anonymous location-based queries. In *SSTD*, 2007.

[19] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVÉ: Anonymous location-based queries in distributed mobile systems. In *WWW*, 2007.

[20] G. Ghinita, K. Zhao, D. Papadias, and P. Kalnis. A reciprocal framework for spatial k-anonymity. *Information Systems*, 35(3):299–314, 2010.

[21] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy-aware location sensor networks. In *HotOS*, 2003.

[22] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *MobiCom*, 1999.

[23] T. Hashem and L. Kulik. Safeguarding location privacy in wireless ad-hoc networks. In *UbiComp*, 2007.

[24] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE TWC*, 1(4):660–670, 2002.

[25] B. Hoh and et al. Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. *IEEE TMC*, 11(5):849–864, 2012.

[26] B. N. Jagdale and N. S. Gawande. Hybrid model for location privacy in wireless ad-hoc networks for mobile applications. *IJCA*, 57(21):1–10, 2012.

[27] B. N. Jagdale and N. S. Gawande. Hybrid model for location privacy in wireless ad-hoc networks. *IJCNIS*, 5(1):14–23, 2013.

[28] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE TKDE*, 19(12):1719–1733, 2007.

[29] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *IEEE ICDCS*, 2005.

[30] G. Kaupins and R. Minch. Legal and ethical implications of employee location monitoring. In *HICSS*, 2005.

[31] M. Li, S. Salinas, A. Thapa, and P. Li. n-CD: A geometric approach to preserving location privacy in location-based services. In *IEEE INFOCOM*, 2013.

[32] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *ACM PODS*, 2004.

[33] R. Nagpal and D. Coore. An algorithm for group formation in an amorphous computer. In *ICPADS*, 1998.

[34] X. Pan, J. Xu, and X. Meng. Protecting location privacy against location-dependent attacks in mobile services. *IEEE TKDE*, 24(8):1506–1519, 2012.

[35] M. A. Pathak and B. Raj. Large margin Gaussian mixture models with differential privacy. *IEEE TDSC*, 9(4):463–469, 2012.

[36] A. Perrig and et al. SPINS: Security protocols for sensor networks. In *MobiCom*, 2001.

[37] D. Poole. *Linear algebra: A modern introduction*. Brooks/Cole, Belmont, CA, 2nd edition, 2006.

[38] S. R. B. Prabhu and S. Sophia. A survey of adaptive distributed clustering algorithms for wireless sensor networks. *IJCSES*, 2(4):165–176, 2011.

[39] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *MobiCom*, 2000.

[40] D. Rebollo-Monedero, J. Forné, and J. Domingo-Ferrer. Query profile obfuscation by means of optimal query exchange between users. *IEEE TDSC*, 9(5):641–654, 2012.

[41] E. Snekkenes. Concepts for personal location privacy policies. In *ACM EC*, 2001.

[42] B. Son, S. Shin, J. Kim, and Y. Her. Implementation of the real-time people counting system using wireless sensor networks. *IJMUE*, 2:63–80, 2007.

[43] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab. A lightweight secure scheme for detecting provenance forgery and packet drop attacks in wireless sensor networks. *IEEE TDSC, accepted to appear*, 2014.

[44] T. Wang and L. Liu. Privacy-aware mobile services over road networks. In *VLDB*, 2009.

[45] L. Yao, L. Kang, P. Shang, and G. Wu. Protecting the sink location privacy in wireless sensor networks. *Personal and Ubiquitous Computing*, 17(5):883–893, 2013.

[46] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE TMC*, 3(4):366–379, 2004.

[47] J.-D. Zhang, G. Ghinita, and C.-Y. Chow. Differentially private location recommendations in geosocial networks. In *IEEE MDM*, 2014.

[48] G. Zhong and U. Hengartner. A distributed k-anonymity protocol for location privacy. In *PerCom*, 2009.

Jia-Dong Zhang received the M.Sc. degree in computer science and engineering from Yunnan University, China, in 2009. He is currently working toward the Ph.D. degree with the Department of Computer Science, City University of Hong Kong. His research work has been published in journals such as *IEEE TSC*, *IEEE TITS* and *Pattern Recognition*. His research interests include location-based services and location privacy.

Chi-Yin Chow received the M.S. and Ph.D. degrees from the University of Minnesota-Twin Cities in 2008 and 2010, respectively. He is currently an assistant professor in Department of Computer Science, City University of Hong Kong. His research interests include spatio-temporal data management and analysis, GIS, mobile computing, and location-based services. He is the co-founder and co-organizer of ACM SIGSPATIAL MobiGIS 2012, 2013, and 2014.