TinyCasper: A Privacy-Preserving Aggregate Location Monitoring System in Wireless Sensor Networks

Chi-Yin Chow Mohamed F. Mokbel Tian He
Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN
{cchow, mokbel, tianhe}@cs.umn.edu

ABSTRACT

This demo presents a privacy-preserving aggregate location monitoring system, namely, *TinyCasper*, in which we can monitor moving objects in wireless sensor networks while preserving their location privacy. *TinyCasper* consists of two main modules, *in-network location anonymization* and *aggregate query processing over anonymized locations.* In the first module, trusted *wireless sensor nodes* collaborate with each other to anonymize users' exact locations by a cloaked spatial region that satisfies a prespecified privacy requirement. On the other side, the *aggregate query processing* module collects and analyzes the cloaked spatial regions reported from the *wireless sensor nodes* to support *aggregate* and *alarm* queries over anonymized locations. The prototype of *TinyCasper* is implemented on a physical test-bed on the TinyOS/Mote platform with 39 MICAz motes.

1. INTRODUCTION

The emergence of the state-of-the-art devices and communication techniques in wireless sensor networks has resulted in many new applications for military and civilian purposes. Examples of these applications include surveillance and location monitoring systems. Such applications support wide variety of important functionalities that include. (1) *Density queries*, e.g., "*determine the number of moving objects within a specified region*", (2) *Safety control*, e.g., *"send an alarm if the number of persons in a certain area exceeds a prespecified threshold"*, and (3) *Resource management*, e.g., *"turn off some building facilities if the number of people in a certain area is less than a certain threshold"*. Such location monitoring applications rely on deploying wireless sensor nodes that are able to communicate with a small wireless transmitter attached to human bodies to determine human's exact locations and identities (e.g., see Bat [4], Active Badge [11], and Cricket [9]).

Although location monitoring systems promise convenience and safety, with untrustworthy server, adversaries could abuse the personal location information to track people or reveal some sensitive information. Figure 1 gives an example of a privacy threat that can take place in location monitoring systems. Figure 1a depicts a floor plan of a certain building at time t_i in which there are 11 sensor nodes installed in nine rooms R_1 to R_9 and two hallways C_1 and C_2. Each sensor node reports the exact location of each user within its monitoring area to a server. Figures 1b and 1c give the readings reported by the same sensor nodes at two consecutive time instances t_{i+1} and t_{i+2}, respectively. If an adversary knows that Alice is in room R_3 at time t_i, then the adversary can know that Alice has left her room at time t_{i+1} and went to C_2. Likewise, the adversary can reveal that Alice has left C_2 at time t_{i+2} and went to R_7. Such knowledge leakage may lead to several privacy threats. For example, tracking the personal behavior in a clinical building may reveal the personal medical record by knowing the particular clinic that the person has visited. Similarly, employers can use the location monitoring application to invade their employees' privacy by checking how long their employees are out of their cubes every day and where do they spend their time, e.g., chatting with other employees, in restrooms, or walking around [5, 6].

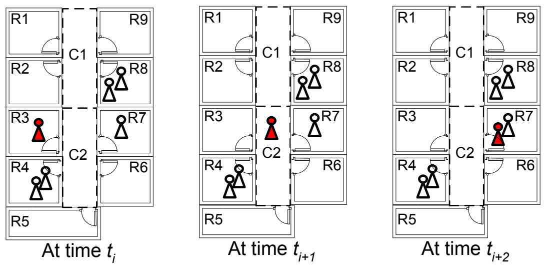

Figure 1: Privacy threats in surveillance systems

This demo presents the *TinyCasper* system that aims to enable all sensor-based monitoring functionalities without sacrificing the privacy of tracked objects. *TinyCasper* has two main modules, namely, *in-network location anonymization* and *aggregate query processing over anonymized locations.* The first module mainly aims to blur the exact user location within the monitoring area of each sensor node into a cloaked spatial region that contains at least k objects. If the number of users within the monitoring area of a certain sensor node is less than a prespecified privacy threshold k, that sensor node will exchange information with other sensor nodes to come up with a larger monitoring area that contains at least k objects. Figure 2 depicts the same example of Figure 1 when applying the anonymization module of *TinyCasper* with $k = 3$. At time t_i, both the sensor nodes

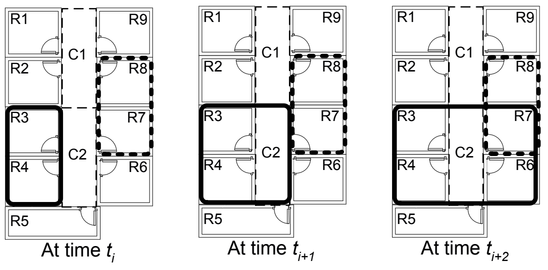

Figure 2: Anonymized locations in *TinyCasper*

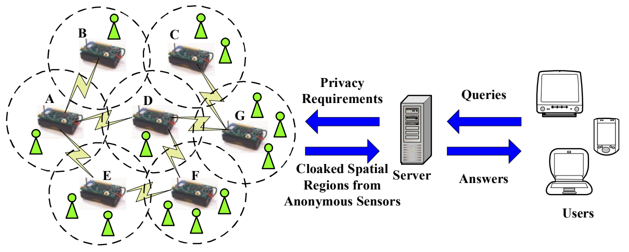

Figure 3: System architecture

at R_3 and R_4 report the same region with number of objects = 3 while the sensor nodes at R_7 and R_8 report the same region with also number of objects = 3. Then, at time t_{i+1}, an adversary would know that someone has entered C_2. However, the adversary cannot pinpoint who is that person, instead the adversary can say that this person is one of the three persons that were in R_3 and R_4 at time t_i. Thus, the areas and the count numbers reported from sensor nodes make all users k-anonymous, i.e., indistinguishable among k users ($k = 3$). Similarly, at time t_{i+2}, an adversary cannot infer that Alice has entered R_7. The second module of *TinyCasper*, i.e., *aggregate query processing*, aims to provide all monitoring functionalities (i.e., aggregate queries and alarm queries) based on the anonymized data received from Figure 2 instead of the actual data received from Figure 1.

TinyCasper can be viewed as applying similar functionalities to the *Casper* system [7, 8] on TinyOS/Mote platform [10]. The *Casper* system is designed to provide privacy-aware location-based services in environments where users detect their location information via GPS-like devices and send it to a centralized entity. Due to its centralized nature, *Casper* cannot be applied to the wireless sensor network environment. Up to the authors' knowledge, *TinyCasper* is the first sensor-based system that ensures users' location privacy through k-anonymity while providing location monitoring functionalities (e.g., aggregate and alarm queries).

2. SYSTEM ARCHITECTURE

Figure 3 depicts the system architecture of *TinyCasper*, in which only the wireless sensor nodes are trusted. As has been mentioned in Section 1, *TinyCasper* consists of two main modules, *in-network location anonymization* and *aggregate query processing over anonymized locations*. In the *location anonymization* module, the wireless sensor nodes collaborate with each other to find their cloaked spatial regions, and then send the cloaked spatial region along with the number of objects within the region to a server via secure and anonymous communication. *TinyCasper* can employ any existing anonymous and secure communication techniques designed for wireless sensor networks. On the other side, the *aggregate query processing* module is embedded inside the server to support aggregate query processing over anonymized locations. *TinyCasper* system administrators can change the privacy requirement, i.e., k-anonymity requirement, at anytime. Finally, system users can issue snapshot and continuous *aggregate* and *alarm* queries through the server terminals.

3. IN-NETWORK LOCATION ANONYMIZATION

The *TinyCasper* location anonymization module consists of two phases. (1) *Initial anonymization phase.* In this phase, sensor nodes with a non-zero object count broadcast a message containing their IDs, monitoring areas, and IDs of the detected objects. When the peers receive these messages, they rebroadcast them until all their neighboring sensor nodes have enough number of objects, i.e., at least k objects. Then, the cloaking sensor node calculates a score for each peer of the received messages. The score is defined as a ratio of the number of objects to the area of the minimum bounding rectangle covering the monitoring area of both the cloaking sensor node and the peer. Finally, the cloaking sensor node selects the peers with the highest score repeatedly until at least k objects are found. The cloaked spatial region is computed as the minimum bounding rectangle covering the monitoring area of the cloaking sensor node and the selected peers. The sensor node reports the cloaked spatial region with the number of objects within the region to the server. (2) *Incremental maintenance phase.* After the sensor node finds a cloaked spatial region, the sensor node registers itself with the selected peers. The sensor node will be notified by these peers whenever these peers detect any changes in their detected objects. If the sensor node does not have enough number of objects, the sensor node will enlist its peers for help to find more objects.

4. AGGREGATE QUERY PROCESSING OVER ANONYMIZED LOCATIONS

The *TinyCasper aggregate query processing* module supports *aggregate* and *alarm* queries over anonymized locations without knowing users' exact locations. For an *aggregate* query, given a query region, the query processor returns the number of objects within the query region. For an *alarm* query, given a query region and a threshold, the query processor keeps track of the density of the query region, i.e., the number of objects within the query region divided by the query region area. The query processor notifies the querying user whenever the density is larger than the threshold.

The main idea of the *TinyCasper aggregate query processing* module is to continuously maintain a *spatio-temporal histogram* to estimate the number of objects within each monitored area. The *spatio-temporal histogram* divides the system space into N disjointed equal-sized cells. We assume that the query processor is able to know the total number of objects M in the system. This can be achieved by installing wireless sensors at all entrances of the monitored building to count the number of objects entering and leaving the building. Initially, the query processor assumes that existing objects are evenly distributed in the system. Thus, the estimated number of objects located within each cell is $\widehat{N} = M/N$. Whenever the query processor receives a cloaked

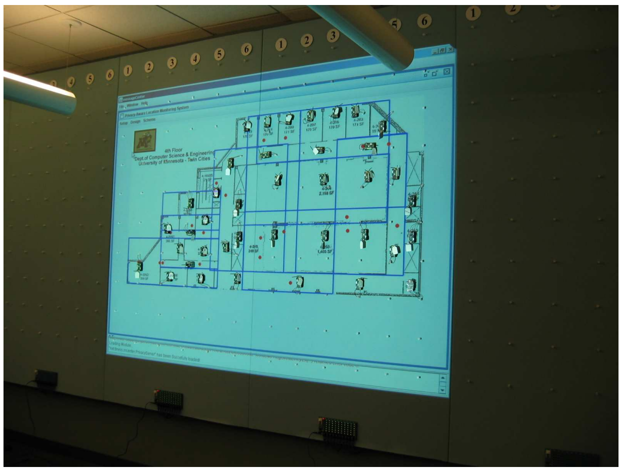

Figure 4: The prototype of *TinyCasper* on a physical test-bed with 39 MICAz motes

spatial region R along with the information about the number of objects R_N within this region, the query processor refines the histogram by performing two key steps. (1) The query processor determines the difference *diff* between R_N and $\widehat{N}$ of the cells overlapping with R. (2) The query processor updates the histogram by uniformly distributing R_N (or *diff*) among the cells within (or outside) R.

5. DEMO SETTINGS

Figure 4 depicts the prototype of *TinyCasper* on the TinyOS/Mote platform [10] in nesC [3] with 39 MICAz motes [2]. A floor plan is projected on two 4-foot by 8-foot boards using a projector. Also, we use an MIB510 serial gateway [1] connected to a computer as a base station. This prototype consists of the following three main modules.

1. Floor plan. We use the floor plan of the fourth floor of the Computer Science and Engineering Building at University of Minnesota - Twin Cities. The floor plan is divided into 39 monitoring areas that correspond to faculty offices, student labs, and hallways. A moving object generator is used to generate people movement within the floor plan. The exact locations of the moving objects are represented as red circles for illustration purposes only. However, these actual locations are not revealed to the server.

2. Wireless sensor nodes. In each of the 39 monitoring areas, we install a wireless sensor that is responsible for detecting the exact location of the users within its monitoring area. In this demo, the wireless sensor nodes get the identity and exact location of the moving objects within their monitoring areas from the *floor plan* module. Also, the sensor nodes get the privacy requirement, i.e., the degree of k-anonymity, from the server. Then, the sensor nodes with a non-zero object count collaborate with each other to find their cloaked spatial regions (represented as blue rectangular regions). Finally, the sensor nodes report their cloaked spatial regions along with the number of objects within the region to the server.

3. Server. Figure 5 depicts a GUI screen shot for the *TinyCasper* server. The server GUI is used by the system security or administration people where they can change the privacy requirements of the system at anytime. Furthermore, they can get the number of moving objects within a specified area or get an alarm if the number of people in a certain area exceeds a specified threshold. Using the server GUI, users are able to plot any area within the floor plan as a monitored area. However, they cannot see the exact locations of people, but they can see only the cloaked spatial regions reported by the sensor nodes (depicted as blue rectangles in Figure 5).

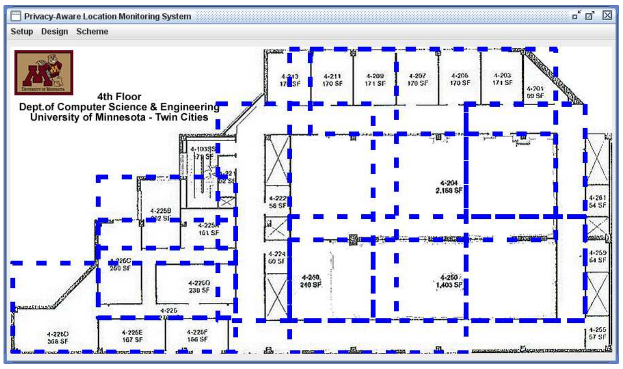

Figure 5: Server GUI

6. DEMO EQUIPMENTS

For the actual demo in the conference, we will use a small scale physical board from the one shown in Figure 4 that has the floor plan of Figures 5 plotted on it. We will use about ten MICAz motes as the sensor nodes and one MIB510 serial gateway as the base station for the wireless sensor network. Also, we will have a laptop that acts as the server, in which the audience can issue aggregate monitoring queries over anonymized locations depicted as rectangles. For illustration and comparison purposes, we will show a screen GUI in which actual people locations and actual sensor nodes' readings are reported. By contrasting this screen with the server screen, audience will be able to understand the privacy threat and how it is properly handled in *TinyCasper*.

7. REFERENCES

[1] Crossbow Technology Inc. MIB510 Serial Gateway. `http://www.xbow.com/Products/productdetails.aspx?sid=226`.

[2] Crossbow Technology Inc. MICAz 2.4GHz. `http://www.xbow.com/Products/productdetails.aspx?sid=164`.

[3] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *PLDI'03*, 2003.

[4] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *MobiCom*, 1999.

[5] G. James. Can't Hide Your Prying Eyes. *Computerworld*, 38:35–36, 2004. `http://www.computerworld.com/securitytopics/security/privacy/story/0,10%801,90518,00.html`.

[6] G. Kaupins and R. Minch. Legal and Ethical Implications of Employee Location Monitoring. In *Proceedings of the Hawaii International Conference on System Sciences*, 2005.

[7] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Procesing for Location Services without Compromising Privacy. In *VLDB*, 2006.

[8] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: A Privacy-Aware Location-Based Database Server (Demo). In *ICDE*, 2007.

[9] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *MobiCom*, 2000.

[10] UC Berkeley WEBS Project. TinyOS: Operating System Support for Sensor Tiny Networked Sensors. `http://www.tinyos.net`.

[11] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM TOIS*, 10(1):91–102, 1992.