Query privacy in wireless sensor networks

BOGDAN CARBUNAR, YANG YU, WEIDONG SHI, MICHAEL PEARCE AND VENU VASUDEVAN
Motorola Labs

Existing mechanisms for querying wireless sensor networks leak client interests to the servers performing the queries. The leaks are not only in terms of specific regions of interest but also of client access patterns. In this paper we introduce the problem of preserving the privacy of clients querying a wireless sensor network owned by untrusted organizations. We first propose an efficient protocol, SPYC, that ensures full client privacy in settings where the servers providing access to the network are honest-but-curious and whose collaboration does not extend beyond well-defined administrative purposes. Furthermore, we study the same query privacy problem in a setting where servers exhibit malicious behavior or where powerful external attackers have access to sensor network traffic information. In this setting we propose two metrics for quantifying the privacy achieved by a client's query sequence. We then extend SPYC with a suite of practical algorithms, then analyze the privacy and efficiency levels they provide. Our TOSSIM simulations show that the proposed extensions are communication efficient while significantly improving client privacy levels.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Security

Additional Key Words and Phrases: Data Storage and Query Processing, Security, Privacy, Data Integrity

1. INTRODUCTION

The traditional trust model of wireless sensor networks assumes the owner and deployer of the network to be the collector and consumer of sensor readings. While this makes sense for small, experimental networks, this is likely not to be the case for large scale sensor networks. Programs and projects such as [Geoss], [Orion], [Ocean.US/IOOS], [Looking] and Neptune [Howe et al. 2003] are building observatories and systems (including sensor networks) for observing and reporting earth, ocean and atmosphere information needed to address complex environmental problems.

The trust model for such sensor networks is shaped by two factors. First, multiple organizations (e.g., government agencies, universities and companies) can be involved, acting both as funding sources and primary investigators. Even though

Authors' address: Applications Research Center, Motorola Labs, Schaumburg, IL, 60195 {carbunar,yang,larry.shi,michael.pearce,cvv012}@motorola.com

network owners need to collaborate for administrative purposes, they may not fully trust each other, due to diverging interests. For instance, the [Geoss] project involves 61 countries and the [Nopp] project involves DARPA, the Department of State and the Department of Homeland Security among others. Second, external interest in the network, coupled with the organizers' wish to recover part of the investment, may lead to opening up the network to queries from third parties.

However, providing support for third party (client) queries may raise privacy and efficiency issues, since clients may not be willing to disclose their interests. To illustrate this problem, assume that each network owner organization is a server with direct access to the sensor network. Consider then the case of two or more servers providing access to an under-water sensor network and an oil company interested in querying certain sensors. The client's program may consist of several constructs of the type illustrated below, using a SQL/TinyDB syntax. In this example, the oil company is interested in temperature readings from sensors placed in the rectangle of coordinates [10..20,30..40], every 20 seconds, for a duration of 500 seconds.

```
SELECT nodeid, temperature AS T
  FROM sensors AS s
  WHERE s.x BETWEEN 10 AND 20
    AND s.y BETWEEN 30 AND 40
  SAMPLE PERIOD 20 s FOR 500 s
```

Knowledge of the client's profile, along with its access patterns and regions and readings of interest may be used by the servers against the client's interests. The access patterns may further disclose the execution program of the client. For instance, the servers may discover that a client is interested only in readings from sensors placed in a specific region (e.g., the rectangle in the above example) or accessing several regions in a certain order.

This latter case is illustrated in the following lines of code, extending the previous example. We assume this code is also executed once every 20 seconds, immediately after performing the temperature readings shown above. That is, if the temperature readings in the rectangle [10..20,30..40] exceed 100 degrees, find out the humidity reported by sensors in the rectangle [10..20,10..20], else query the sensors in the rectangle [30..40,30..40]. It is straightforward to see that the query traces generated by these lines of code allow the servers to reverse-engineer the original client code.

```
IF (T > 100) THEN                    ELSE
  SELECT humidity                      SELECT humidity
    FROM sensors AS s                    FROM sensors AS s
    WHERE s.x BETWEEN 10 AND 20          WHERE s.x BETWEEN 30 AND 40
      AND s.y BETWEEN 10 AND 20            AND s.y BETWEEN 30 AND 40
ENDIF
```

In this paper, we address this problem. Specifically, given a client query sequence $q_1, q_2, \ldots, q_l$, we need a solution that hides from the servers not only the exact loca-

tions and identities of the queried sensors, but also relationships between individual queries. Since sensor lifetimes are severely constrained by their limited batteries, we are particularly interested in solutions where both the communication and computation overheads imposed on sensors are reasonable. Thus, the straightforward solution, where for each query q_i the client queries the whole network and keeps the readings only from the sensors of interest, will not be considered. We will also prefer to take advantage of the local communication capabilities offered by wireless interfaces, such as broadcast, which is not only more efficient than repeated unicast transmissions but may also be used to hide from the sender the identity of the message receiver. Moreover, we will give preference to solutions where the computation required from sensors involved in queries consists only of efficient message authentication codes (keyed-hashes) and symmetric key operations, instead of the more computation intensive operations based on modular exponentiation (e.g., public key operations).

State of the art work in wireless sensor networks considers mainly the cases where (i) queries for events of interest originate from the network or (ii) sensors report events to the base station and the attacker has various degrees of control on network traffic. Specifically, [Shao et al. 2007] provide privacy for queries occurring in a data centric sensor network, where events detected are stored in the network, based on event attributes. Since attackers are assumed to be able to corrupt sensors, the confidentiality of events can be easily compromised. The solution of [Shao et al. 2007] is then to use a keyed mapping function for events and to store the (encrypted) events on storage sensors using an event-associated key. The work of [Shao et al. 2008; Yang et al. 2008; Ozturk et al. 2004; Kamat et al. 2005; Deng et al. 2004] studies the problem of preserving the privacy of event sources reporting data to the base station, in scenarios where attackers are able to capture up to all in-network data transmissions. The solution proposed in [Shao et al. 2008; Yang et al. 2008] consists of sending periodic, dummy traffic to obfuscate the real event reports. The work of [Ozturk et al. 2004; Kamat et al. 2005] consists of sending event reports on random walks in the network before forwarding them to the base station. [Deng et al. 2004] proposed several anti-traffic analysis techniques, including hiding the packet destination address using encryption, decorrelating packet sending times and controlling packet sending rates between parent and children sensor nodes.

Instead, we propose efficient mechanisms for privately querying sensor networks, in an environment where access to sensor readings is provided by servers governed by multiple, mutually distrusting organizations. That is, the system consists of (at least) two servers, where each server is administered by a different organization. This model can be particularly appealing to competing corporations, government or military departments, with strict information compartmentalization rules. It allows organizations to share the risk and investment in building a large scale sensor network, while preserving the privacy of their own queries. From a client perspective, our solutions enable clients to privately access a large-scale wireless sensor network, for fees that may be well under the cost of its deployment and management.

Contributions: We first investigate the case where the servers are honest but curious. Specifically, even though each server correctly follows a specified protocol, it can use information revealed by clients during correct protocol runs in order to

violate client privacy. In this setting we propose SPYC, a protocol that decomposes a client's interaction with the sensor network into two tasks. The first task, of privately naming each sensor, requires two servers, but is executed only once for each client, when the client registers with the system. The output of this task is a fresh, key space for the sensor network, to be known and used only by the client on whose behalf it was created. The second task takes advantage of the key space established during the first task in order to privately access readings from sensors of interest. This task is performed for each client query but requires interaction with only one server. Using standard cryptographic techniques, the client hides from the servers the sensor keys, the identities and locations of sensors queried and relationships between any two queries. We show that if the servers are honest and do not cooperate, this solution achieves full client privacy.

We further study the query privacy problem in an environment where servers or external attackers can exhibit Byzantine behavior and have sufficient resources to obtain traffic information from the sensor network. In this setting, we start by introducing two metrics for measuring client privacy, called "spatial" and "temporal" privacy. We then propose an extension to SPYC where each client query for a region of interest is transformed into a set of queries, each for a different region, individually contacted using SPYC. We introduce a suite of practical transforms, that is, functions that take as input a client query and output a set of regions (including the one in the input query) that need to be queried. Finally, for each proposed transform we study the tradeoff between its efficiency and the privacy it provides to clients.

While several existing solutions (e.g. the work of [Shao et al. 2008; Yang et al. 2008; Ozturk et al. 2004; Kamat et al. 2005; Deng et al. 2004]) can be used in conjunction with our solutions to further improve query privacy guarantees, we note that our work is different in several aspects. First, our system model is different since (i) clients external to the system push queries in the network and (ii) we are interested in preserving not only the query location privacy but also relationships (e.g., location proximity) between sensor queries occurring at different times, issued by the same or different clients. Second, our trust and attacker models are different from most of the related work, since all the queries issued by clients go through adversary controlled access points (the servers).

Our simulations show that the proposed SPYC protocol is implementable using off-the-shelf hardware platforms, with reasonable communication and time overhead compared to a straightforward, non-private query scheme. Regarding the query transform technique, our simulation results indicate that by tuning the transform parameters, we are able to achieve various tradeoffs between privacy and efficiency levels according to the application requirements and resource constraints.

Paper Layout: The paper is organized as follows. In Section 2 we describe our system model and introduce the studied privacy problem. In Section 3, we describe the initial SPYC protocol and in Section 4 we extend it to handle the case of malicious servers or powerful external attackers. Section 5 provides our TOSSIM simulation results. In Section 6 we survey existing research and Section 7 concludes the paper.

2. SYSTEM ASSUMPTIONS AND NOTATIONS

We consider a sensor network that is operated by dedicated server(s), and shared among multiple clients. Clients can access the network by sending queries to the server(s), which in turn issue the queries into the network. We assume sensors are bootstrapped with symmetric encryption keys shared with the servers. Moreover, well-behaved sensors install only code that is correctly signed by one of the servers. The purpose of code signatures is to prevent one of the servers from repudiating installed malicious code (e.g., monitoring sensor activity and reporting it later to the server). Servers are allowed to upload the code installed on any sensor, verify the signature of the server that installed it and analyze it. Thus, a single signature should be sufficient to prove if a particular server has installed malicious code.

Let the entire network be partitioned into a set of n regions, identified by $R = \{r_1, r_2, \ldots, r_n\}$. Each sensor belongs to exactly one region and a region may contain multiple sensors. The size of a region is application specific and sufficiently small to support the query resolution. We consider on-demand sensor network queries, consisting of acquiring sensor readings from specific regions of interest. Examples of such queries include, "how many cars have passed region A from time T?", or "what is the current temperature in region B?". While the solutions we present support both on-demand queries and registration of interest for specific regions, in this work we focus only on on-demand queries.

For ease of analysis, we assume that each query targets a single region (our work naturally expands to the cases where one query can target a subset of regions). Let $\mathbb{Q}$ denote the space of all possible query sequences and let $Q \in \mathbb{Q}$ denote a query sequence of length l, $Q = [q_1, q_2, \ldots, q_l]$. Let $Q[i]$ denote the ith query in the sequence Q. The region of interest for a query $q \in Q$ is given by the function $g : \{1, \ldots, l\} \mapsto R$, which takes as input the index of the query and returns the region identifier. For ease of presentation, we also use $Q = [g(q_1), g(q_2), \ldots, g(q_l)]$ to denote the query sequence. Moreover, we say $r_i \in Q$ to denote the fact that region r_i is queried during the sequence Q.

For simplicity of presentation but also for reducing communication costs we choose to elect region (cluster) heads. That is, each region has a head, i.e., a sensor that assumes responsibility over the region. The region head may be periodically elected and the battery levels of sensors may be used as a metric in the election, in order to uniformize the wearing off of sensors. We assume an existing routing tree built upon the region heads, over which queries are disseminated from the root and sensing results are gathered back to the root. Routing between region heads that are in a parent/child relationship but not within transmission range of each other may naturally be relayed by non-head sensors. Besides routing, region heads are also responsible for broadcasting client queries to the sensors in the region on whose charge they are, collecting (using unicast packets) the individual answers and even, but not necessarily, aggregating the answers before transmitting them to the server through the routing tree. Several existing techniques for cluster-leader election and routing in sensor networks [Heinzelman et al. 2000] can be applied in this context.

We define a query transform as a function $T : Q \mapsto \mathcal{P}(R)$, where $\mathcal{P}(R)$ denotes the power set of R. Thus, a transform of $q_i \in Q$ is represented as $T(q_i) = m$, where

$m \subseteq R$. We only consider meaningful transforms, where $g(q_i) \in m$. The size of a transform is defined to be the size of m, denoted by $|m|$. Applying the transform T to each query of a query sequence Q, produces the sequence $M = [m_1, m_2, \ldots, m_l]$, where each set $m_i \subseteq R$, $i = 1, \ldots, l$. In this paper we only study transforms with a constant $|m_i|$ value. An example transform is T_h, where $T_h(q_i) = R$, $i = 1, \ldots, l$. That is, T_h requires the server to always query all the regions.

We define the cost of a transform T, denoted as $C(T)$, to be the network traffic incurred when querying the regions contained in M. To analytically evaluate $C(T)$, we consider the regions R to be organized in a grid shape. We assume that the servers are placed at one corner of the grid and the network traffic of querying a region scales linearly with the Manhattan distance between the server and the region. While in this model we abstract intra-region data aggregation, we do consider inter-region aggregation (this fits into clustering protocols such as LEACH [Heinzelman et al. 2000]). The expected distance between a region and the server can be computed as $\int_1^{\sqrt{n}} \int_1^{\sqrt{n}} (x + y) \mathrm{d}x\mathrm{d}y/n = \sqrt{n}$. Then, under this model, the cost of the previously presented T_h transform is $C(T_h) = c \cdot n \cdot l\sqrt{n}$, where c is a constant determined by the size of the query data and by the radio of the sensor nodes. $C(T_h)$ represents the upper bound on the cost of any transform.

The solution space is bounded by two transforms. The first transform is the aforementioned T_h, which provides the maximum privacy but also incurs the maximal cost. It is a well known fact that maximum privacy can only be achieved by T_h [Chor et al. 1995]. The second transform keeps the query sequence unchanged, providing no privacy, but imposing a minimal cost. Let T_l denote this transform, i.e., $T_l(r_i) = \{r_i\}$. We can estimate $C(T_l) = cl\sqrt{n}$.

2.1 Cryptographic Tools

In the following, we use $X \hookrightarrow_R D$ to denote the random choice of X from the domain D.

Pseudo Random Functions. A function family $H_K : \{0,1\}^n \rightarrow \{0,1\}^n$ where $K \in \{0,1\}^t$ is pseudorandom if it satisfies the following properties

—Given $x \in \{0,1\}^n$ and $k \in \{0,1\}^t$, there is a PPT algorithm that computes $H_k(x)$.

—For any PPT algorithm Alg, every polynomial $p(.)$ and all sufficiently large values of t

$$|\mathtt{Pr}_{\mathtt{k} \hookrightarrow_R \{0,1\}^t}[\mathtt{Alg}(\mathtt{H_k}, 1^t) = 1] - \mathtt{Pr}_{\mathtt{f} \hookrightarrow_R U_t}[\mathtt{Alg}(\mathtt{f}, 1^t) = 1]| \leq 1/\mathtt{p}(\mathtt{t}),$$

where U_f is the set of all functions mapping $\{0,1\}^n$ to $\{0,1\}^n$.

For implementation purposes, in the following H_K is a message authentication code using key K as argument.

Let len_H denote the bit size of the output of the HMAC function.

Pseudo Random Permutations. A permutation $E_K : \{0,1\}^n \rightarrow \{0,1\}^n$, where $K \in \{0,1\}^t$, is called pseudo random if for any PPT algorithm Alg, every polynomial $p(.)$ and all sufficiently large values of t

$$|\mathtt{Pr}_{\mathtt{k} \hookrightarrow_R \{0,1\}^t}[\mathtt{Alg}(\mathtt{E_k}, \mathtt{E_k^{-1}}, 1^t)] = 1] - \mathtt{Pr}_{\pi \hookrightarrow_R \mathtt{P_n}}[\mathtt{Alg}(\pi, \pi^{-1}, 1^t)] = 1]| \leq 1/\mathtt{p}(\mathtt{t})$$

where P_n is the set of all random permutations from $\{0,1\}^n$ to $\{0,1\}^n$. In the following, E_K is a block cypher and we use $D_K = E_K^{-1}$ to denote decryption.

2.2 Security Property

We now define the following property that needs to be satisfied by any private query transform.

Semantically Secure Transform. Given a query sequence $Q \in \mathbb{Q}$, a query transform $T_K : Q \rightarrow \{0,1\}^n$, where $K \in \{0,1\}^t$, is semantically secure if for any PPT algorithm Alg, every polynomial $p(.)$ and all sufficiently large values of t

$$|\mathtt{Pr_{q \in Q}[Alg(T_K(q), 1^t) = 1] - Pr_{rnd \leftarrow_R \{0,1\}^n}[Alg(rnd, 1^t) = 1]}| \leq \mathtt{1/p(t)}$$

That is, a query transform T_K is semantically secure if any PPT algorithm cannot distinguish the output of T_K from a random string of length n.

2.3 Attacker Model

In this paper we consider two models for the server behavior. The first model assumes that the servers are honest but curious. The second model considers also the case of malicious insiders (servers) or external, powerful attackers, performing traffic analysis attacks. In the following we describe each model in detail.

2.3.1 *Honest-but-Curious Server Model.* In this model, servers are assumed neither to collaborate, nor to attempt to actively interfere with the specified behavior of the protocol. Moreover, we do not consider the existence of external attackers, or if they exist, they are not powerful enough to subvert defense mechanisms deployed by the organizations that deployed the sensors (see Section 4 for a discussion of such defense mechanisms). However, any server can passively collect information during honest runs of the protocol and use it to infer client interests. This model makes sense especially in scenarios where each server also queries the network on behalf of its organization and intends to keep the other servers unaware of its interests. In this scenario organizations running the network may play an active part in keeping the network safe against more intrusive attacks, such as the ones presented later in this section. We present now several attacks that honest-but-curious servers can mount.

Query Packet Investigation: A server performing client queries can try to infer the identity of the destination sensor or of intermediate routers from the query packets provided by clients or from the results received from the network. For instance, the server could use its knowledge of the sensor network and of query results in order to infer the queried areas (e.g., the server knows that only sensors in region r_x can report temperatures in the range $[t_1, t_2]$. Finally, a server can try to build statistics of fields occurring in packets and use them to infer packet trajectories.

Tampering of Duplicate Query Packets: A server can mutate a given query packet and generate multiple, slightly different packets and inject them in the network, along with the original packet. The purpose of this attack is for the server to understand the original packet's structure. Such mutated packets could be obtained by removing, adding or changing the order of fields occurring in the original packet. The server could also try to add fields captured from previous query packets from

the same client or even from other clients. We make the observation that even though the server is injecting modified packets into the network, we still consider this attack to be under the umbrella of the honest-but-curious server scenario. This is because the original client query packet is injected in the network by the server.
Timing Attack: A server can measure the interval between the moment when it injects a client query into the network and the moment when it receives a result from the network. Statistics of these measurements could be used in order to infer the distance (hop count) to the queried regions.
Fake Client Registration: A server can register an arbitrary number of fake clients in order to learn information provided by other servers during client registration.
Battery Level Monitoring: If a server is able to query the remaining battery levels of sensors, it can use this information to infer the trajectory of query packets. This can be achieved by the server querying the battery levels in the network before and after a query and use the decreased values to infer the destination and the intermediate routers traversed by the query packet.

2.3.2 *Traffic Analysis Attacks.* In the second model, we consider the existence of malicious insiders (servers) or of external attackers powerful enough to thwart sensor network defenses deployed by the network's deployers and obtain traffic information. In the following we describe several types of such attacks.

Source RF Attacks: A resource-rich attacker (including one of the servers) may place multi-directional antennas inside the sensor network, or jammers in the vicinity of sensors, in order to identify packet relayers and even the destination sensor. Multi-directional antennas can be used to detect the transmission source of a packet and jammers can be used in order to trigger re-transmissions. The indoor location sensing systems of [Tao et al. 2003] and [Ladd et al. 2002], where "snooper" devices observe the signal strength of packets received from a target machine in order to infer its location, can be used by an attacker not only to detect sensor locations but also to eavesdrop on network transmissions.
Intrusion Attacks: An attacker can physically attack individual sensors, extract any code or data (e.g, keys) stored on the sensors and even install malicious code. The corrupted sensors can then be used in order to report local traffic, including participation in client queries.
Server Collaboration: Servers can collaborate and exchange information collected during interactions with clients.

In this paper we do not consider denial of service attacks, where attackers drop packets in order to prevent transmissions. We are only concerned with maintaining the privacy of successful queries.

3. SPYC

We propose an initial solution assuming the existence of two honest-but-curious mutually non-trusting servers (S_1 and S_2) providing access to the sensor network. Our protocol, called SPYC, consists of four procedures, (i) initialization, (ii) key space generation, (iii) query routing and (iv) result routing. The initialization procedure is executed only once, for each server. The key space generation is

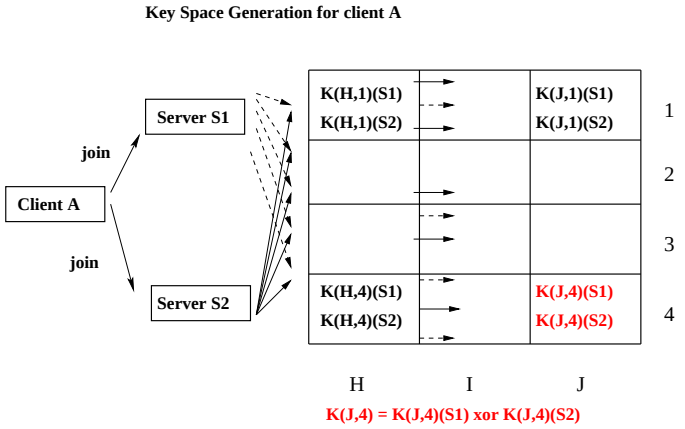


Fig. 1. Client registration. Client registers with the service, by contacting two servers S_1 and S_2 and requesting the generation of a new key space for network regions.

performed each time a new client registers and the query routing is executed each time a registered client has a query to perform. The concept behind SPYC is the following. The key space generation allows a client to generate fresh keys shared with the sensors in the network, such that none of the servers can retrieve the keys. During the query and result routing steps, the client uses these names and simple cryptographic primitives to construct packets that can be routed in the network, without leaking information to any of the servers that see them. The packets constructed by the client can be interpreted only by the intended sensors (the intermediate routers or destination). We now provide details for each procedure, starting with the initialization.

3.1 Initialization

Let S be one of the servers that provides access to the sensor network. Let t be a system-wide security parameter. Server S generates a symmetric encryption key $k_i(S) \hookrightarrow_R \{0,1\}^t$ for each region r_i and stores it on each sensor in region r_i. S also generates a seed uniformly at random from the space $\{0,1\}^t$ and uses it to initialize a pseudo-random number generator G_S.

The value of t should be chosen according to the RSA Labs's recommendations [Kaliski 2003]. That is, until 2010 a value of $t = 80$ is sufficient, however later, it needs to be increased to 112 (until 2030) and 128 (starting 2031). These key length values are sufficient to ensure security against currently known attacks. The recommendations are based on the assumption that Moore's Law will also hold in the near future.

3.2 Client Registration

When client A registers with the service, it first creates a pseudo-random number generator G_A. It then contacts two of the servers, S_1 and S_2 (see Figure 1 for an illustration). Both servers perform the same operation, exemplified in the following for a generic server S. For each region r_i, $i = 1..n$, S generates a pseudo-random key $K_A^S(r_i) \hookrightarrow_R \{0,1\}^t$, using the PRG G_S. Let $M = A; K_A^S(r_i)$ denote the con-

Client	Key
A	K(J,4)(A)
B	K(J,4)(B)
...	...
X	K(J,4)(X)

(a)

Region Name	Key
(H,1)	$K(H,1) = K(H,1)(S_1) \oplus K(H,1)(S_2)$
...	...
(H,4)	$K(H,4) = K(H,4)(S_1) \oplus K(H,4)(S_2)$
...	...
(J,4)	$K(J,4) = K(J,4)(S_1) \oplus K(J,4)(S_2)$

(b)

Fig. 2. (a) Example Key Table stored on the head of region (J,4). (b) Example Key Table of client A.

catenation of A's name with this value. S sends to the head sensor of region r_i the message

$$[\mathtt{ADD}, \mathtt{A}, \mathtt{S}, \mathtt{E}_{\mathtt{k_i(S)}}(\mathtt{M}), \mathtt{H}_{\mathtt{k_i(S)}}(\mathtt{M})].$$

When receiving an ADD message, the head of region r_i first verifies S's encryption and HMAC fields. That is, if $F_3 = E_{k_i(S)}(M)$ denotes the third field of the ADD message and $F_4 = H_{k_i(S)}(M)$ is the fourth field, the sensor verifies that $\mathtt{H}_{\mathtt{k_i(S)}}(\mathtt{D}_{\mathtt{k_i(S)}}(\mathtt{F_3})) = \mathtt{F_4}$. If the condition holds, the region head stores $K_A^S(r_i)$ along with the client name A. When the head of region r_i has received keys from both S_1 and S_2 for client A, it computes the value $K_A^{S_1}(r_i) \oplus K_A^{S_2}(r_i)$ and stores it as its secret key shared with client A (see Figure 2(b) for an illustration).

After disseminating the new keys to all the sensors, each server S provides the client A with the region-based topology of the sensor network (including a routing tree) and a key table containing mappings $(r_i, K_A^S(r_i))$, between actual region names and keys. The client can then build the key of each region r_i as the XOR of the keys received from S_1 and S_2, that is, $K_A(r_i) = K_A^{S_1}(r_i) \oplus K_A^{S_2}(r_i)$ (see Figure 2(b)).

3.3 Query Routing

We describe the query routing procedure, which defines the T_K transform of SPYC. The procedure is illustrated using a source routing algorithm. We chose source routing due to its simplicity to describe and model its cost, however, the mechanisms we propose could be used in conjunction with other routing protocols.

Let q be a query of client A, referring to region r. Let $r_1, .., r_p$ be the ancestor regions of r in the routing tree of the sensor network. That is, $r_1, .., r_p$ are the regions that need to forward packets from a server (S_1 or S_2) to r. Let h be the height of the routing tree ,i.e, the maximum number of regions a packet needs to traverse from the server to any region. Figure 3(a) illustrates the following query routing process.

Client A picks a server $S \in \{S_1, S_2\}$ to perform query q. A also generates two fresh nonces $N, N' \hookrightarrow_R \{0,1\}^t$ and $h-p-1$ pseudo-random values $R_1, .., R_{l-p-1} \hookrightarrow_R \{0,1\}^{len_H}$. The R values have the same bit-size as the output of the function H. The client produces the following transform

$$\mathtt{T_K(q)} = \mathtt{N}, \mathtt{N'}, \mathtt{H}_{\mathtt{K_A(r)}}(\mathtt{N'}), \mathtt{H}_{\mathtt{K_A(r_j)}}(\mathtt{N})(\mathtt{j} = \mathtt{1..p}), \mathtt{R}_{\mathtt{1:1..h-p-1}}, \text{“}\mathtt{F\_PTR}''$$

In this message, the value $H_{K_A(r)}(N')$ is called the *stopping condition* since it en-

codes the identity of the region of interest, r. The values $H_{K_A(r_j)}(N)$ with $j = 1..p$ encode the identities of the routers, that is, the ancestors of r in the routing tree and are called *router conditions*. The values R_l, with $l = 1..h-p-1$ are pseudo-random values used as padding in order for all the ROUTE packets to have the same length. That is, all packets have h routing condition fields. Due to the properties of pseudo random functions (see Section 2.1), the server cannot distinguish between actual router conditions and the random padding values. Finally, the field "F_PTR" is used to point to the end of the packet and its purpose will be discussed later. A then uses its identity and the transform $T_K(q)$ to generate the following ROUTE packet

$$[\mathtt{ROUTE}, \mathtt{A}, \mathtt{T_K(q)}, \mathtt{H_{K_A(r_j)}(A, T_K(q))(j = 1..p)}, \mathtt{R'_{1:1..h-p-1}}]$$

where the values $R'_1, .., R'_{l-p-1} \hookrightarrow_R \{0,1\}^{len_H}$ are also used as padding. As mentioned in Section 3.2, each $K_A(r_j)$ value denotes a key shared by the client with an intermediate router and with the destination region. The $H_{K_A(r_j)}(A, T_K(q))$ values denote HMACs of string $A, T_K(q)$ computed with those keys and serve the purpose of authenticating the ROUTE packet. This prevents adversaries from modifying the content of the query as it travels through the network. That is, any sensor drops query packets which it cannot authenticate.

A more space efficient solution for authenticating queries is to require the client to sign a single hash of the query packet $A, T_K(q)$. A single authentication value is then used instead of h keyed hashes. This solution however requires the client to distribute its public key to all the sensors during the client registration step. In order to prevent a server from controlling the client's public key (e.g., replacing it with its own) the client "splits" the public key in two shares and sends one share to each of the two servers performing the registration. The shares are then combined at each sensor in order to reconstruct the client's public key. If any server tampers with the client's public key share, it will prevent the client from querying any sensor. However, no single server is able to replace the client's public key with its own. For simplicity of exposition, in the following we ignore the query authentication fields from the ROUTE packets.

While not explicitly included in the above ROUTE packet, the packet also contains the query for region r, encrypted with key $key = H(K_A(r))$. Only the client and the head sensor of r can derive the key value.

When a server S receives a ROUTE packet, it broadcasts the packet using its wireless interface. All the region heads that are within the server's transmission range receive this packet. Each region head s (e.g., of region r_i) that receives such a packet uses the key shared with client A and the nonce N' to verify the stopping condition. If the condition is not satisfied, s verifies the router condition, using the nonce N and the first field after the stopping condition field. If the router condition is not satisfied, it drops the packet. If the condition is satisfied, however, the sensor is a router. It then removes the first field after the stopping condition and appends it at the end of the packet. The sensor then re-broadcasts the resulting packet

$$[\mathtt{ROUTE}, \mathtt{A}, \mathtt{N}, \mathtt{N'}, \mathtt{H_{K_A(r)}(N')}, \mathtt{H_{K_A(r_j)}(N)(j = 2..p)}, \mathtt{R_{1:1..h-p-1}}, \text{“}\mathtt{F\_PTR}\text{''}, \mathtt{H_{K_A(r_1)}(N)}].$$

This process is repeated by region heads receiving this new ROUTE packet.

If the stopping condition is satisfied by the head s of region r_i, then s belongs

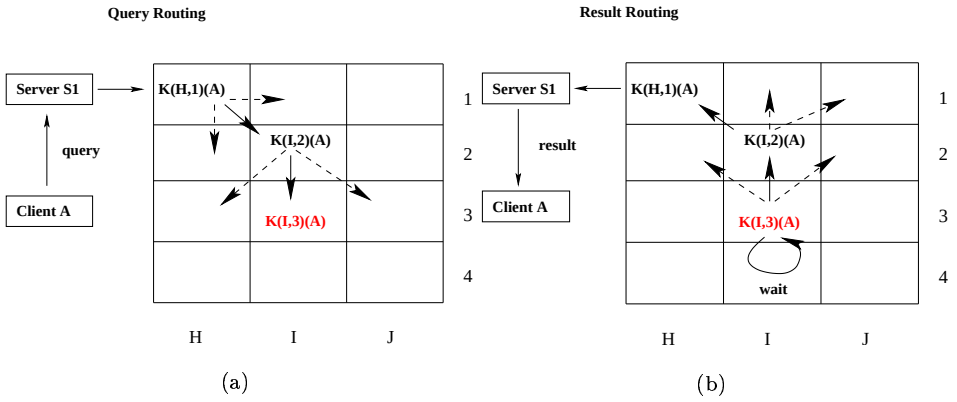


Fig. 3. (a) Client performs query using any server S (any of S_1 or S_2). The query is obfuscated using region keys, unknown to S and a fresh, public key K. A head of a region whose keyed hashed name matches the first routing condition field in the ROUTE packet, needs to further broadcast the packet. (b) The target region (I,3) satisfies the stopping condition, that is, $H_{K(I,3)}(N')$ equals the packet's stopping condition field. The head of this region no longer needs to broadcast the packet, but delays its answer for a time interval required for a packet to traverse two regions plus the time required by the processing of a packet in a region. In effect, this delay is equivalent to the existence of one more hop in the query packet. Then, it sends the results of the query to its parent region head.

#	Name	Value
1	Client	A
2	nonce	N
3	nonce	N'
4	StopCond	$H_{K(I,3)(A)}(N')$
5	F_PTR	F_PTR
6	Router	$H_{K(H,1)(A)}(N)$
7	Router	$H_{K(I,2)(A)}(N)$
8	Router	$H_{K(I,3)(A)}(N)$
9	Router	R_1

(a)

#	Name	Value
1	Client	A
2	nonce	K
3	Result	$result_s$
4	Router	$H_{K(I,2)(A)}(N)$
5	Router	$H_{K(H,1)(A)}(N)$

(b)

Fig. 4. (a) Example query routing packet used in Figure 3(a). (b) Example result routing packet generated by a sensor s in region (I,3) in Figure 3(b).

to the region of interest, that is, $r_i = r$. Only the client A and the region head of r can derive this information, being the only participants knowing the key $K_A(r)$. The packet received by s has the following format

$$[\mathtt{ROUTE}, \mathtt{A}, \mathtt{N}, \mathtt{N}', \mathtt{H}_{\mathtt{K_A(r)}}(\mathtt{N}'), \mathtt{R}_{\mathtt{l:1..h-p-1}}, \text{“}\mathtt{F\_PTR}\text{”}, \mathtt{H}_{\mathtt{K_A(r_j}}(\mathtt{N})(\mathtt{j} = \mathtt{1..p})].$$

The purpose of the “F_PTR” field is then to allow s to identify the padding values $R_{l:1..h-p-1}$. The padding values are stored between the stopping condition and the “F_PTR” field. The sensor s counts the padding values ($h-p-1$ in total) in order

to determine the delay it should apply before answering the client's query.

The delay is used as a defense against timing attacks launched by server S or external attackers. To prevent an attacker from inferring the depth in the routing tree of the queried region, s waits for an interval equal to $(h - p - 1)(2T_t + T_p)$ before broadcasting the RSLT packet. T_t is an approximation of the time required to send a packet across two neighboring regions and T_p is an approximation of the time required to process a ROUTE packet in a region (two hash computations and two comparisons for all the nodes in the region).

Instead of using random paddings, the client could include routing conditions corresponding to actual, randomly chosen successors (in the routing tree) of the queried region r. Then, the destination head s would also broadcast the received ROUTE packet, instead of waiting for an interval dictated by the padding values. This strategy provides increased client privacy: an attacker able to detect the source of packet transmissions (using for instance techniques described in Section 6) would be unable to infer the destinations of ROUTE packets as the sensors last to forward them. This advantage comes however at the expense of higher network traffic.

3.4 Building the Query Answer

After or during the wait interval (of $(h - p - 1)(2T_t + T_p)$ seconds) the sensor s performs the reading requested by the client. That is, it broadcast the query to all the sensors in its region, collects their results and aggregates them. Let $reading(s)$ denote the result of the operation and as before $key = H(K_A(r))$. s generates a random number rnd and produces the value

$$\mathtt{result_s} = \mathtt{E_{key}(reading(s), rnd, H_{K_A(r)}(reading(s), rnd))},$$

s uses the fields of the received ROUTE packet and the value $result_s$ to generate the following result (RSLT) packet

$$[\mathtt{RSLT, A, N, result_s, H_{K_A(r_j)}(N))(j = 1..p)}]$$

which it sends over the broadcast channel.

3.5 Result Routing

The routing of the RSLT packet is very similar to that of the ROUTE packet. That is, any sensor $s \in r_i$ receiving such a packet uses the second field, A and the third field, the nonce N, to generate the value $H_{K_A(r_i)}(N)$. If this value equals the last field of the packet, the sensor removes the last field from the packet and re-broadcasts the resulting packet. Otherwise, the sensor drops the packet. When the last router receives this packet, it sends it to the server S, which in turn sends it to the client A.

The client uses the key $key = H(K_A(r))$ to decrypt the fourth field of the packet. It then compares the second field of the decryption's result, $K_A(r)$ with the key of region r stored in its key table (see Figure 2(b)). If they match, the client accepts the $reading(s)$ value.

We make the observation that if the sensor network maintains a routing tree, that is, each sensor maintains a link to its parent sensor, the propagation of the RSLT packets can be done using unicast transmissions instead of broadcast.

3.6 Extending SPYC with Crowds

[Reiter and Rubin 1998] introduced Crowds, a system providing anonymity for web users. It groups users into a large, geographically distributed group, that issues requests to web servers on behalf of group members. Whenever a group member has a request for a web server, instead of sending it directly to the server it sends it to another, randomly selected group member. When a group member receives such a request, it either sends it to another randomly chosen group member (with probability p) or to the server (with probability $1 - p$). This technique could be used to address the problem that we study in our work, or used in conjunction with it.

We propose now propose two solutions for combining SPYC with Crowds. In the first solution, DIST-CROWDS, the sensors are assumed to be aware of the network topology. Then, the client uses SPYC to forward the query to a randomly chosen sensor s_r, who then locally follows the Crowds procedure explained above. The client query also contains the region of interest, encrypted with the key shared with s_r. Before forwarding the query, s_r also re-encrypts the query with the key of the new, locally chosen group member. Note that this solution imposes a significant routing storage and management overhead on individual sensors.

In our second solution, CENTRAL-CROWDS, the sensors are topology agnostic. The client decides then all the sensors that will forward the query, that is, the Crowds "group" members that will route the query around before sending it to its destination region. Let those "group" sensors be $s_1, .., s_q$. Then, the client uses SPYC to generate the query path between s_{q-1} and s_q ($Q_{q-1,q}$) and encrypt the result with the key shared with s_{q-1} ($E_{s-1}(Q_{q-1,q})$). It then generates the query path between s_{q-2} and s_{q-1}, $Q_{q-2,q-1}$ and encrypts the concatenation $Q_{q-2,q-1}; E_{s-1}(Q_{q-1,q})$ with the key shared with s_{q-2} and so on. This process is similar to onion routing, also used in Tor. Each "group" sensor is able to remove one layer of the encryption, obtain the query path to the next "group" sensor, obfuscated using the SPYC approach and send it using broadcast toward that sensor.

3.7 Defenses

We now show the defenses provided by SPYC against the attacks described in Section 2.3 for a honest-but-curious attacker.

Theorem 1. *In the honest-but-curious server model, SPYC provides a semantically secure transform. That is, the transform of any query is indistinguishable from a random string.*

PROOF. Let us assume that there exists a PPT algorithm Alg such that for a query sequence $Q \in \mathbb{Q}$ and the associated transform T_K described in Section 3.3, the following inequality holds

$$|\mathtt{Pr}_{\mathtt{q}\in\mathtt{Q}}[\mathtt{Alg}(\mathtt{T}_\mathtt{K}(\mathtt{q}), 1^\mathtt{t}) = 1] - \mathtt{Pr}_{\mathtt{rnd}\hookrightarrow_\mathtt{R}\{0,1\}^{2\mathtt{t}+\mathtt{hlen}_\mathtt{H}}}[\mathtt{Alg}(\mathtt{T}_\mathtt{K}(\mathtt{q}_2), 1^\mathtt{t}) = 1]| > 1/\mathtt{p}(\mathtt{t}),$$

for a polynomial $p(.)$ and sufficiently large values t. Then, we build an algorithm Alg' that breaks the pseudo randomness of the H_K function. Specifically, algorithm Alg' consists of the following steps

—Generate nonces $N_1, N_1', N_2, N_2' \hookrightarrow \{0,1\}^t$.

—Select region $r \hookrightarrow_R R$ and let $r_1, ..r_p \in R$ its ancestors.
—Generate random values $R_{p+2}, .., R_h, R'_1, .., R'_h \hookrightarrow_R \{0,1\}^{len_H}$.
—Generate

$$\mathtt{T_K(q)} = \mathtt{N_1}, \mathtt{N'_1}, \mathtt{H_{K(r)}(N'_1)}, \mathtt{H_{K(r_j)}(N_1)}, \mathtt{j = 1..p}, \mathtt{R_{p+2}}, .., \mathtt{R_h}, \text{"F\_PTR"},$$

$$\mathtt{rnd} = \mathtt{N_2}, \mathtt{N'_2}, \mathtt{R'_1}, \mathtt{R'_2}, , ., \mathtt{R'_h}, \text{"F\_PTR"}.$$

—Pass $T_K(q)$ and rnd to algorithm Alg.

That is, the transform $T_K(q)$ contains the stopping condition for the region of interest r, routing conditions for its ancestors and random paddings. The random string rnd is selected from the space $\{0,1\}^{2t+hlen_H}$.

If algorithm Alg can distinguish between $T_K(q)$ and rnd with a chance better than $1/p(t)$, then algorithm Alg' can also distinguish between $H_{K(r)}(N'_1)$ and a random value, R'_1, or between $H_{K(r_j)}(N_1)$ and a random value R'_{j+1}, j=1..p, with more than a $1/(hp(t))$ chance, where h is a constant. This contradicts the pseudo randomness of the function H_K (see Section 2.1). □

In the following we further investigate the honest-but-curious attacks introduced in Section 2.3.

Query Packet Investigation: The servers involved in the key generation process can derive the keys of sensors for a particular client only if they cooperate. This is because the key of a sensor is the result of an xor between two different values, each provided separately by one of the servers. If the two servers do not collaborate and generate those values independently, the result of the xor will be a random value, which none of the servers can predict. Theorem 3.7 then proves the fact that the server performing queries cannot distinguish a query string from a random string.

Furthermore, the server performing a query cannot use the result of a query to infer the identities of queried regions. This is because the result of a query, $readings(s)$, is packaged as $E_{key_x}(reading(x), rnd, H_{K(r_x)}(reading(x), rnd))$ by the queried head of region r_x. A similar approach to the one used in Theorem 3.7 can be used to show that the use of a freshly generated random value rnd for each reading performed, makes result packets indistinguishable from random strings. We also make the observation that the use of the keyed hash inside the encrypted $readings(s)$ value authenticates the sensor readings, since only the client and the head of region r_x know $K(r_x)$.

Finally, all queries have the same length, due to the artificial padding of query packets with routing condition fields up to the maximum path length (h, the height of the routing tree). This prevents the server from inferring the distance (hop count) to the regions queried by clients.

Tampering of Duplicate Query Packets: The server or corrupted sensors cannot change the fields of query packets they route. This is because each query packet is authenticated by the client.

Timing Attack: The time delay introduced by the head s of the destination region after receiving the ROUTE packet, hides the hop count distance of s from the server. The time delay essentially makes the duration of queries from different

regions indistinguishable. The T_t and T_p values used in the time delay equation (see Section 3.3) could be either predefined or dynamically computed by s based on timestamps added to the ROUTE packet by the intermediate routing sensors.
Fake Client Registration: The values sent encrypted by servers to sensors during the key space generation step (see Section 3.2) are randomly generated for each client on whose behalf the operation is performed. Even if an attacker registers multiple fake clients, the sensor keys assigned for those clients will be with high probability different from the keys assigned to the same sensors on behalf of other clients. Such an attacker is then unable to use the collected information to derive the keys of sensors as viewed by any other client.
Battery Level Monitoring: In order to collect battery readings from all the sensors, a flood followed by a converge-cast is required. This makes this operation more expensive than the ROUTE/RSLT routing procedure, significantly reducing the number of useful queries that can be performed on the network. Moreover, battery readings on state-of-the-art wireless sensor nodes are not particularly accurate.

However, this attack can be prevented by placing various restrictions on queries concerning the battery levels of sensors. One type of restriction could be to allow servers to query battery levels of sensors only periodically. Conversely, sensors could periodically notify servers of their remaining battery levels. For instance, if servers can access such information only 3 times a day and data is assumed to be accurate, associations between clients and queried regions will be obfuscated as a function of the number of client queries performed in 8 hours. Another restriction could be to allow sensors to report battery readings only within a certain interval before running out of power.
Traffic Analysis Attacks: If servers also query the network on behalf of their organizations, hey have all the interest to keep the privacy of their queries intact. To achieve this, the organizations could apply several techniques, besides making sure that the servers do not collaborate. For instance, both the sensors and the code installed on them can be periodically inspected. Also, the sensor network can be constantly monitored to detect abnormal behavior, e.g., searching for jamming hardware. Note also that intrusion and source RF attacks can be expensive. A stronger attacker, able to deploy more multi-directional antennas and jammers or corrupt more sensors, can benefit more from such attacks. However, such a behavior also increases the attacker's chance of being detected.

3.8 Storage Considerations

Considering infeasible an exhaustive search in the space $\{0, .., 2^{80} - 1\}$ [Kaliski 2003] we restrict the size of keys of regions to 80 bits. If client names are also at most 80 bits long, the key table of a sensor supporting 10000 concurrent clients is around 0.2MB [1]. The space required by the key table can be reduced in scenarios where the servers can group clients into clusters based on mutual client trust (for instance, all individual clients from a major corporation). Furthermore, the system could support the ability of removing keys when clients leave. This operation can be performed by any server S, through the dissemination of the message

[1] While the 80 bit limit is valid until 2010, when it increases to 112 bits, by 2010 the storage space on sensor nodes will also increase.

$[\mathtt{GC}, \mathtt{A}, \mathtt{S_S}(\mathtt{H}(\mathtt{GC}, \mathtt{A}, \mathtt{T}))]$, where T denotes the current time and is used to prevent replay attacks (e.g., when client A registers, de-registers and then registers again at a later time). When a sensor receives a validly signed GC message with a fresh timestamp, it deallocates A's space before further broadcasting the message.

By truncating the output of the crypto-hash function H to 2 bytes and considering TinyOS packets with a payload of 40 bytes, a ROUTE type packet can be used to query a sensor network with a 16 hop depth (h=16). The truncation generates a chance of region name collisions of 2^{-16}. However, the chance of collisions decreases exponentially with the distance between the server and the queried region, becoming 2^{-160} for a region 10 hops away from the root. While collisions may create additional network traffic, they do not decrease the privacy offered by SPYC. The client may only receive more replies than expected, replies that can be easily discarded.

4. QUERY OBFUSCATION

It becomes obvious upon close inspection of SPYC that unfortunately it is vulnerable to traffic analysis attacks. In particular, SPYC is vulnerable to source RF attacks, intrusion attacks and server collaboration. In this section we study the query privacy problem in a scenario where an attacker is able to gather information about the trajectory of each ROUTE and RSLT packet through the network. In this case query privacy can be achieved using obfuscation. When queries have a strong temporal dimension (i.e., are relevant only if performed in a certain order and at specific time intervals), obfuscation implies querying regions beyond the ones of interest. We model this by using a transform T, that, given a region of interest in a client's query, produces a set of regions to be queried by the server. For each region in this set, the client produces a ROUTE packet as described in Section 3.3. The server performing the queries takes all resulting ROUTE packets, injects them into the network, waits to receive all the results and forwards them to the client. In the following we abstract the SPYC layer processing of queries and instead focus on transforming the region of interest in a client query to a set of regions to be queried by the servers.

4.1 Practical Privacy Metrics

Using the definitions from Section 2, in the following we are interested in various practical solutions within the space defined by T_l and T_h. We want our solutions to achieve graceful tradeoffs between the privacy level and the associated costs. We quantify these tradeoffs using two metrics for evaluating a transform's ability to conceal the spatial and temporal patterns of the original query.

4.1.1 *Spatial Privacy Level.* Using the notations of Section 2, let $\tilde{Q}$ denote the set of unique regions in Q, and $\tilde{M}$ denote the set of unique regions in M. Let s denote the size of $\tilde{Q}$, i.e., $s = |\tilde{Q}|$. Intuitively, a transform conserves spatial privacy if, given the transformed sequence M, it is difficult for the server to guess the regions in Q. Let $S(T)$ denote the *spatial privacy level* of a transform T. We define $S(T)$ as the inverse of the server's probability to guess a region in $\tilde{Q}$, given $\tilde{M}$.A larger $S(T)$ indicates a better spatial privacy level. For example, $S(T_h) = \frac{n}{s}$, since $\tilde{M} = n$. This is the best spatial privacy level achievable by any transform.

Also, $S(T_l) = 1$, since $\tilde{M} = \tilde{Q}$.

4.1.2 *Temporal Privacy Level.* From a temporal privacy perspective, we consider the query frequency of regions in the sequence Q. Let X denote a random variable with discrete vocabulary R. Consider the distribution of X in Q with probability function $p_i = Pr[X = r_i \in R]$. Ideally, the corresponding distribution of X in M should differ from $\{p_i\}$ to conceal the frequency pattern in Q.

We use the notion of Kullback-Leibler divergence (D_{KL}) to measure the difference between two distributions. Consider the distribution of X in M with probability function $\bar{p}_i = Pr[X = r_i \in R]$. Since a transform appends regions that are of no interest to queried regions, it may be that $\sum_{r_i \in Q} \bar{p}_i < 1$. In such cases, we proportionally scale the $\bar{p}_i$ values, such that $\sum_{r_i \in Q} \bar{p}_i = 1$. Specifically, when computing D_{KL} we consider the scaled distribution function $p'_i = \frac{\bar{p}_i}{\sum_{r_i \in M} \bar{p}_i}$. Let $R(T)$ denote the *temporal privacy level* of a transform T. We have

$$R(T) = D_{KL}(\{p_i\}||\{p'_i\}) = \sum_{r_i \in R} p_i \log \frac{p_i}{p'_i} \tag{1}$$

Based on information theory, $R(T)$ is always non-negative, i.e., $R(T) \geq 0$, with $R(T) = 0$ if and only if $p_i = p'_i$, for all $r_i \in R$. A greater $R(T)$ indicates a larger difference between the two distributions. Thus, we intend to maximize the cross entropy to conceal the frequency patterns of the initial query sequence.

Consider the example of T_h. We have $\bar{p}_i = \frac{1}{n}$ for each $r_i \in R$, and after scaling, $p'_i = \frac{1}{s}$. Then,

$$R(T_h) = \sum_{r_i \in R} p_i \log p_i s = \log s + \sum_{r_i \in R} p_i \log p_i \ .$$

Let $H(Q)$ denote the information entropy of Q, i.e., $H(Q) = -\sum_{r_i \in R} p_i \log p_i$. We have $R(T_h) = \log s - H(Q)$. It is easy to see that $R(T_l) = 0$.

Since $R(T)$ is defined based on the distribution of the query frequency of regions, $R(T)$ does not capture information such as correlation among queried regions, for instance, when region r_2 is queried every time after r_1 is queried. We plan to investigate higher order Markov source models in our future work to address this issue.

4.2 Privacy *vs.* Cost Tradeoffs

We study practical solutions in the space bounded by T_l and T_h, that is, satisfying the following constraints,

$$1 \leq S(T) \leq \frac{n}{s} \ , \tag{2}$$

$$0 \leq R(T) \leq \log s - H(Q) \ , \tag{3}$$

$$cl\sqrt{n} \leq C(T) \leq cnl\sqrt{n} \ . \tag{4}$$

In the following we consider either off-line or on-line query sequences. Off-line queries can be used for instance for surveillance purposes. For example, the requirement "report the temperature in region r_1 every 5 minutes, and in r_2 every 10 minutes" generates a query sequence $[r_1, r_1, r_2, r_1, r_1, r_2, \ldots]$. On-line queries can occur in instances where the client follows moving objects or diffusing phenomena.

4.2.1 *Off-Line Queries.* In the case of off-line queries, either the entire initial query sequence or the query sequence distribution is known *a priori*. This scenario can occur if the client can tolerate long query delays, allowing buffering of queries before execution. In this context, we propose and study three off-line transforms, the union transform, randomized transform and hybrid transform.

Union Transform (UT): Given Q, UT performs the transform $UT(q_i) = \cup_{q_j \in Q}\{g(q_j)\}$, i.e., each query is transformed to the set of all regions that appear in Q. For example, for a query sequence $Q = [r_1, r_3, r_1, r_3, r_5]$, we have $UT(r_1) = UT(r_3) = UT(r_5) = \{r_1, r_3, r_5\}$. Since all regions in M appear in Q, we have $S(UT) = 1$. Also, the distribution of all regions in M is uniform. Thus, $\bar{p}_i = p'_i = \frac{1}{s}$, for $r_i \in M$ (recall that s is the number of unique regions in Q). Thus, $R(UT) = \log s - H(Q)$. Moreover, we have $C(UT) = cls\sqrt{n}$.

Randomized Transform (RT): Given Q, RT transforms each query into a randomized set of regions, which includes the original region to be queried. That is, for every $q_i \in Q$, RT randomly chooses $z-1$ regions, denoted as R^z, from $R - \{g(q_i)\}$, where z is a pre-specified parameter. Then, we have $RT(q_i) = R^z \cup \{g(q_i)\}$. For example, for a query sequence $Q = [r_1, r_3, r_5, r_1, r_3, r_5, \ldots]$, one possible transformed sequence is $M = [\{r_1, r_8, r_9\}, \{r_3, r_4, r_7\}, \{r_2, r_5, r_6\}, \{r_1, r_5, r_{10}\} \ldots]$.

Theorem 2. *The expected value of the spatial privacy of RT, $E[S(RT)]$, is equal to $n(1-\alpha^{l+1})/s$, where $\alpha = 1 - \frac{z}{n}$ and l is the length of the query sequence.*

Proof. Given a query sequence Q (whereby s and n are fixed), the number of unique regions in $RT(Q)$ is a function of z, denoted as $f(z) = |\tilde{M}|$. We first show the estimation of $E(f(z))$. In the trivial case where $l = 1$, let f_1 denote $E(f(z))$. We have $f_1 = z$.

Let f_2 denote $E(f(z))$ in the case of $l = 2$, i.e., f_2 is the expected number of unique regions in the transformed query of a sequence of length 2, e.g., $[r_i, r_j]$. Consider the following equivalent experiment. Initially, we assume a bag of n balls, out of which z balls are red and the rest are black (these z red balls correspond to f_1). We then randomly take z balls from the bag, mark them red and put them back (this step corresponds to transforming r_j so that another set of z regions are appended to M). Then, f_2 is the expected number of red balls in the bag after the experiment. The expected number of red balls taken out of the bag in the experiment is the mean of a hypergeometric distribution, given by $\frac{z^2}{n}$. Thus, the expected number of red balls in the bag after the experiment is $f_2 = 2z - \frac{z^2}{n}$.

When $l = 3$, repeat the above experiment with f_2 red balls in the bag initially. The expected number of red balls in the bag after the experiment is $f_3 = f_2 + z - \frac{z \cdot f_2}{n}$. And so on. We have

$$f_i = f_{i-1} + z - \frac{z \cdot f_{i-1}}{n} = \alpha f_{i-1} + z \ , \tag{5}$$

where $i = 1, \ldots, l$, $f_1 = z$, and $\alpha = 1 - \frac{z}{n}$. Through algebraic manipulation, we have

$$E(f(z)) = f_l = n(1 - \alpha^{l+1}) \ . \tag{6}$$

Since $E[S(RT)] = \frac{E(f(z))}{s}$, the theorem results.

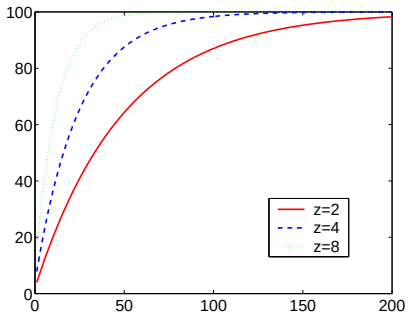


Fig. 5. Illustration of $E(f(z))$, $n = 100$

□

In Figure 5 we plot $E(f(z))$ with parameters $z = 2, 4, 8$, $n = 100$, and $l = 1, \ldots, 200$. It can be observed that $E(f(z))$ approaches n faster for larger z values. Even for $z = 2$, $E(f(z))$ exceeds 64 when $l > 50$.

The following result provides the temporal privacy of RT.

Theorem 3. *The temporal privacy of RT, $R(RT)$, is equal to $\sum p_i \log(\frac{a}{p_i} + b)$, where $a = \frac{z-1}{s(z-1)+n-z}$ and $b = \frac{n-z}{s(z-1)+n-z}$.*

PROOF. Recall that p_i denotes the probability of r_i in Q, and p'_i denotes the probability of r_i in M. M contains lp_i transform from r_i, with r_i occurring exactly once in each of them. M also contains $l(1-p_i)$ transforms from regions other than r_i. The probability for r_i to be in any of these $l(1-p_i)$ transforms is $\frac{z-1}{n-1}$. Since in total zl regions are queried in M, we have

$$\bar{p}_i = \frac{lp_i + l(1-p_i)\frac{z-1}{n-1}}{zl} = \frac{z-1}{z(n-1)} + p_i \frac{n-z}{z(n-1)} \tag{7}$$

After scaling, we have

$$p'_i = \frac{\bar{p}_i}{\sum_{r_i \in Q} \bar{p}_i} = a + bp_i \ , \tag{8}$$

where $a = \frac{z-1}{s(z-1)+n-z}$ and $b = \frac{n-z}{s(z-1)+n-z}$. Thus,

$$R(RT) = \sum_{r_i \in Q} pi \log \frac{p_i}{p'_i} = -\sum_{r_i \in Q} p_i \log(\frac{a}{p_i} + b) \tag{9}$$

We can verify that $\frac{\partial \mathrm{R(RT)}}{\partial \mathrm{p_i}} = \frac{b^2}{p_i(a+bp_i)^2} \geq 0$ for every $r_i \in Q$.

□

Thus, $R(RT)$ is a convex function. Using Lagrangian relaxation, $R(RT)$ is lower bounded by zero when $p_i = \frac{1}{s}$, for every $r_i \in Q$. While $R(RT)$ increases with s and z, the lower bound is independent of s and z. Moreover, given a fixed s and p_i's, $R(RT)$ is upper bounded by $\log s - H(Q)$ when $z = n$.

Finally, it is straightforward to see that $C(RT) = clz\sqrt{n}$.

Hybrid Transform (HT): We observe that while UT achieves better temporal privacy over RT, RT enables more spatial privacy. A hybrid transform can be designed by combining the basic ideas of UT and RT. Given Q, we first randomly choose a set of z' regions from $R - \tilde{Q}$, denoted as $R^{z'}$. Then, HT performs the transform $HT(q_i) = \cup_{q_j \in Q} g(q_j) \cup R^{z'}$. It can be verified that $S(HT) = \frac{s+z'}{s}$, $R(HT) = \log s - H(Q)$, and $C(T_3) = cl(s+z')\sqrt{n}$.

We summarize these results by listing the privacy levels and costs of UT, RT, HT, T_l, and T_h in Table I. Since $E(f(z))$ approaches n as z increases, it can be seen that RT degenerates to T_h when z approaches n. When $z = 1$, RT degenerates to T_l. Also, HT degenerates to T_h when $z' + s$ approaches n, and degenerates to UT when $z' = 1$. Therefore, based on application requirements and cost budget, a suitable transform can be chosen with appropriate parameter settings.

Table I. Privacy levels and costs of three transforms

	$S(T)$	$R(T)$	$C(T)$
T_l	1	0	$cl\sqrt{n}$
UT	1	$\log s - H(Q)$	$cls\sqrt{n}$
RT	$\frac{E(f(z))}{s}$	$-\sum p_i \log(a + bp_i) - H(Q)$	$clz\sqrt{n}$
HT	$s + z's$	$\log s - H(Q)$	$cl(s+z')\sqrt{n}$
T_h	$\frac{n}{s}$	$\log(n) - H(Q)$	$cln\sqrt{n}$

Note that in general, the spatial and temporal privacy levels are orthogonal. That is, there are transforms that can improve on either the spatial or temporal privacy, without affecting the other. For example, the UT transform improves the temporal privacy without affecting the spatial privacy. On the other hand, the following transform improves the spatial privacy without affecting the temporal privacy. Specifically, let P denote a set of random regions that do not appear in the given query sequence Q. Then, each query in Q is transformed to the original query region plus P. While the queried regions are expanded to improve the spatial privacy, the normalized query frequency of regions in Q and the consequent temporal privacy are not changed. Note however, that there are transforms that can improve on both privacy metrics, including the RT and HT transforms.

4.2.2 *On-line Queries.* We now consider the case of on-line queries, where the client generates the query sequence on-the-fly. While RT can still be applied in this case, both UT and HT are not directly applicable, since the client lacks a complete $\tilde{Q}$ to perform the transform. However, information about the query sequence accumulated during the query process can be effectively used in the transform to improve achieved privacy levels. To handle this new setting, we generalize the definition of a transform function introduced in Section 2.

In Section 2, the transform function is defined as $T : R \mapsto \mathcal{P}(R)$, indicating the same transform to be applied for a specific region, regardless of its position in the query sequence. Here, we re-define the function as $T : R \times \mathbb{Z} \mapsto \mathcal{P}(R)$, where $\mathbb{Z}$ is the set of positive integers. Therefore, the transform of the same region may vary over time.

We propose a dynamic transform (DT) algorithm, described in Figure 6, where z denotes the predefined transform size and t is an input parameter. In DT, we use

RT for the first t queries. During the querying process, the client keeps track of the distribution of regions in M. Then, during each query q_j, $j > t$, the transform chooses $z - 1$ regions from M with the lowest distribution. That is, similar to HT, the transform attempts to "uniformize" the distribution of regions in M. However, the process is done adaptively. There is no guarantee on the uniform distribution of regions occurring in the output of the transform. The closeness to uniformity depends on the initial query sequence and the size of the transform, z.

Begin
1. Use RT for the first t queries, with parameter z
2. Calculate the probability distribution of regions in Q
3. **For** each new query q_i
4. Generate $T(q_i)$ using RT, with parameter z
5. Pick the least frequent region r in Q, $r \notin T(q_i)$
6. Pick a random region $r' \in T(q_i)$, $r' \neq g(q_i)$
7. Replace r' with r
8. Update the probability distribution of regions in Q

End

Fig. 6. Pseudo-code for DT

5. SIMULATION RESULTS

5.1 Simulation Setup

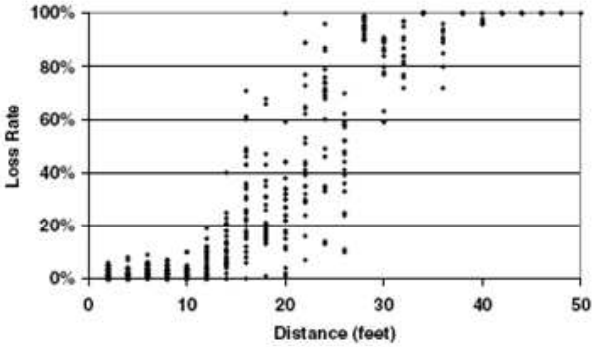
We implemented and evaluated the proposed techniques on a packet-level TOSSIM [Levis et al. 2003]. Using TOSSIM's empirical communication model [Woo et al. 2003] (Figure 7(a)), a 30×30 deployment grid with a 4-foot spacing was generated, with the radio transmission range of sensors set to 50 feet. The network was divided into 10×10 equally sized regions, with each region consisting of 9 nodes. We denoted these regions as $r_1, \ldots, r_{100}$, shown in Figure 7(b).

As previously stated, we implemented a region-based routing scheme to reduce network traffic and avoid congestion. Specifically, one node in every region was designated to be region head. Region members only communicated with their heads, which might communicate to other heads in neighboring regions (the distance between each pair of adjacent heads being 12 feet on average). Ignoring the boundary effect, each region head could communicate with 4 other heads in neighboring regions. Moreover, we allowed up to 5 retransmissions per lost packet.

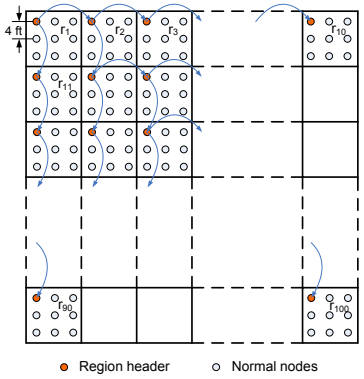
We also implemented a tree-based routing substrate to facilitate the source routing scheme used in SPYC. A binary routing tree, rooted at the head of r_1, was generated to connect all heads to the head of r_1 through shortest paths (Figure 7(b)).

For each simulation run, we generated a query sequence as follows. We first randomly chose s regions to query, with s varying from 10 to 90, in increments of 10. For each region, we chose an expected query rate uniformly distributed in $[1, 9]$. A query sequence of length 300 was then generated using a Poisson distribution with the expected querying rates.

For each query, we generated its corresponding source routing path according to the above routing substrate, which was then embedded into the query packet



(a) TOSSIM packet loss rate over distance



(b) Simulated network deployment, with a binary routing tree for the single-server case

Fig. 7. Simulated packet loss model and network deployment

using the SPYC solution described in Section 3.3. The query packets were always injected into the head of r_1 and were then routed toward the destination region using the SPYC scheme. After reaching the destination region, the query packets were broadcast by the region head to all region members. The members then transmitted their sensor readings to the head. After packing all sensor readings into one data packet, the head routed the data packet back to the server. In the case of obfuscation, a query was first transformed to a set of queries according to the chosen transformation. These transformed queries were then successively injected into the network.

During our simulation, we recorded both the number of inter-region packet trans-

missions and the overall time delay to execute each query. Note that we did not count the number of intra-region transmissions, because intra-region communication requires much less power than inter-region communication, due to a shorter transmission range.

5.2 Initial SPYC Protocol

We first implemented over TOSSIM the initial SPYC protocol that includes client registration, client query routing, and result routing. We studied the cost of computing a crypto hash function H, on a sensor node. We adopted the existing MD5 code [Malinen] as the hash function used in our simulation. Since TOSSIM itself did not provide time statistics for CPU operations, we implemented the MD5 code on a TelosB sensor node [Polastre et al. 2005]. We executed the MD5 code for 20 times on the TelosB platform, with an average time cost of around 7 msec.

The structure of a query packet was the following: 1 byte contained the client identifier, 2 bytes were used for every nonce (N and N') and hash value (indicating routing information and stopping condition). Since for the grid structure of regions used in this experiment the maximum path length is 20, the payload of a query packet is 47 bytes. Compared to the baseline which used a plain source routing packet (assuming a query path of 20 hops, with 1 byte region identifier for each hop), this incurred a cost of 27 bytes. However, we recorded almost the same number of packet transmissions for SPYC and the baseline (the T_l case shown in Figure 8(c)). Moreover, the observed time delay required to complete a query was approximately twice of the baseline delay shown in Figure 8(d). This was because on average, a baseline query required the ROUTE and RSLT packets to be transmitted over a total of 10 hops. However, due to the mandatory wait period in our query scheme (illustrated in Figure 3), SPYC always delayed the transmission of query and data packets as if the destination was 10 hops away from the server (a total of 20 hops).

The TelosB platform consists of a TI MSP430 processor and TI CC2420 radio component. The power consumption of the MSP430 in active state can be derived from the fact that it consumes 250 microAmp @ 3V[TIM], translating to 0.75 mW. During the ROUTE packet forwarding step, each intermediate sensor node along the path performs three MD5 operations (one for checking the packet's authenticity, one for checking the stopping condition and one for checking the router condition). Since each MD5 takes 7mSec, the computation energy cost for every node on the path is 15.75 μJ. The energy consumed by the CC2420 radio is approximately 1.8 microJ to receive a byte and 2.1 microJ to transmit a byte[Mathur et al. 2006]. The length of a packet (either a query or result packet) is around 57 bytes (including the 10 bytes MAC header and a 47 byte payload). Since every node involved in the query process needs to relay one query packet and one result packet, the total energy consumption by every node involved in the query is roughly (15.75 + (1.8+2.1) * 57) * 2 = 476 μJ.

In terms of the energy cost of the entire network, every node involved in the query process needs to receive a packet (either a query or result packet), perform three MD5 operations, and then transmit the packet. We can easily multiply the number of packet transmissions captured by Figure 8 (c) with 476 μJ to obtain the total energy consumed by the entire network. Thus, the energy consumption of the entire network scales linearly with the number of transmitted packets.

5.3 Obfuscation

We have further implemented the query obfuscation algorithms of Section 4 on top of the previously described implementation of SPYC. In the following we present our findings.

5.3.1 *Off-Line Queries.* For the off-line case, the complete query sequence was generated and transformed before they were injected into the network. We focus on examining the performance of UT, RT with $z = 4$, and HT with $z' = 10$. For cases with different parameter settings, we observed similar performance trend that confirmed our analysis in Section 4.2.1. T_l was used as the baseline, while T_h was used to provide the upper bound of spatial and temporal privacy. We depict the results averaged over 200 random runs in Figure 8.

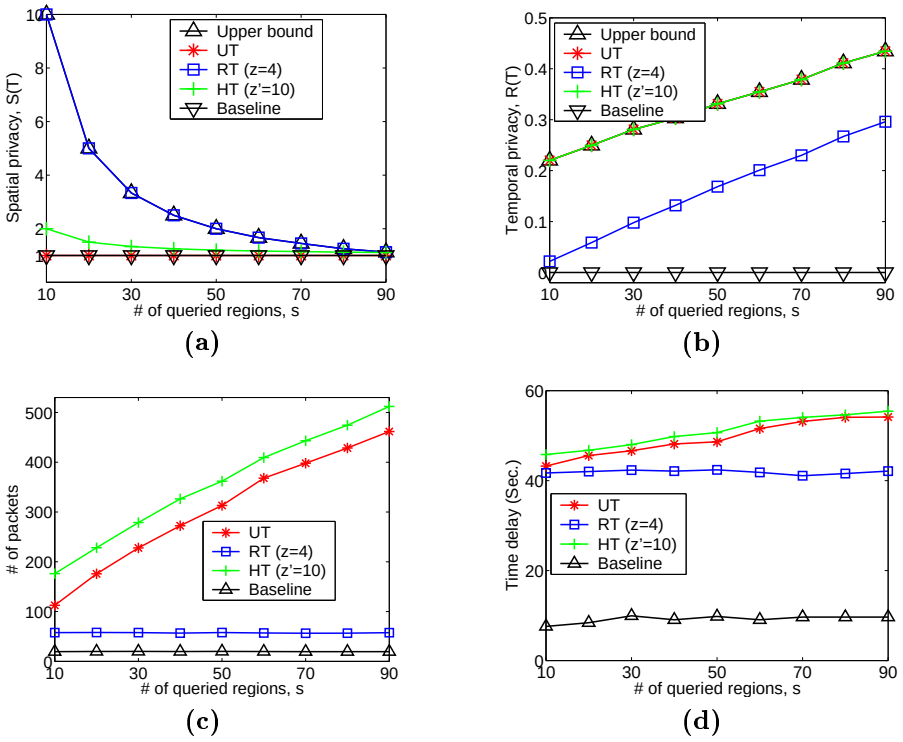
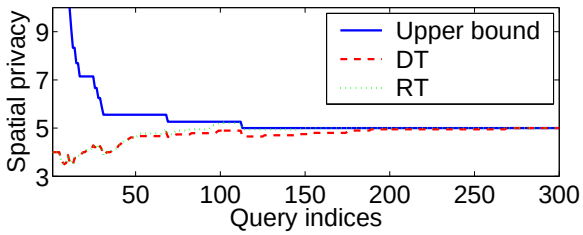
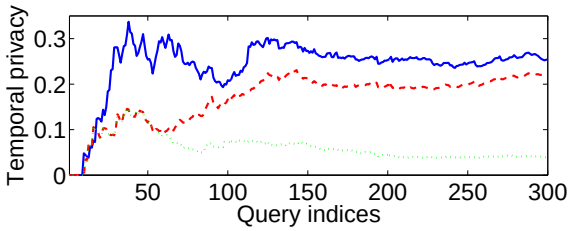


Fig. 8. Simulation results for Union, Random, and Hybrid Transforms (900 sensor nodes, $n = 100$, $l = 300$, $z = 4$, $z' = 10$

It can be observed that the spatial privacy of RT almost achieved the upper bound. This was because with $z = 4$ and $l = 300$, $E(f(z))$ was very close to n. Also, the spatial privacy of HT was between UT and RT. The temporal privacy of all transforms increased with s, confirming our analysis. However, the temporal privacy of RT was only 10% of the upper bound for $s = 10$, and 68% for $s = 90$. As expected, the cost of RT, in terms of number of packet transmissions, was almost



(a) Spatial privacy



(b) Temporal privacy

Fig. 9. Simulation results for the on-line transform ($s = 20$, $z = 4$, $t = 50$)

constant (approximately 2 times larger than the baseline). However, the cost of UT increased fast with s: approximately 6 times higher than the baseline for $s = 10$, and 24 times higher for $s = 90$. Similar trends could be observed for HT. RT also achieved an almost constant time delay, consistently lower than the delay of UT and HT. However, even RT incurred approximately 4 times the delay of the baseline. By looking into the detailed data traces, we discovered that the increased delay of UT and RT was mainly because of severe packet collisions resulting from the high number of communication requests, which were resolved by time consuming re-transmissions. This indicated (1) UT and RT were suitable for delay tolerant applications, and (2) techniques including multi-packet reception (e.g., CDMA, FDMA) can be applied to mitigate the increased delay. In conclusion, while RT achieved lower temporal privacy, it was more practical for larger values of s. This was due to RT's constant costs. However, for relatively small values of s, UT and HT should be preferred, since they provided higher temporal privacy levels at a small cost.

5.3.2 *On-Line Queries.* For the on-line DT algorithm, we generated a query sequence of length 300 using the method described above, with parameters $s = 20$, $z = 4$, and $t = 50$. We recorded the temporal and spatial privacy levels resulting from the on-line transform after the execution of each query. Figure 9 shows the results for DT along with the privacy levels of a pure RT method, and the upper bound of the privacy levels.

We observed that for the first 50 queries, both spatial and temporal privacy levels

of RT and DT were the same. For the remaining queries, the spatial privacy of DT was still very close to that of RT. However, DT achieved a level of temporal privacy much closer to the upper bound than RT did. This was because the DT algorithm attempted to balance the distribution of regions that had already been queried. We performed several simulations with different settings of s and t and observed similar trends.

6. RELATED WORK

Source Anonymity in Sensor Networks: [Shao et al. 2008] study the problem of preserving the anonymity of sensors reporting events to a base station, when the attacker is external, global and passive. The work proposes the idea of generating probabilistic dummy traffic. Sensor nodes send out data (real and dummy events), with the time between successive transmissions being determined by an (exponential) distribution.

While this technique could be used as an alternative to our solution, we note that (i) it makes the additional, strong assumption that the adversary cannot compromise sensors and (ii) it generates significant traffic even when no real events need to be reported. In order to address the latter issue, [Yang et al. 2008] have proposed the use of special sensors called "proxies", that proactively filter dummy traffic on its way to the base station. While this indeed reduces the message overhead, it requires additional hardware, which is particularly prone to physical attacks. This is because an attacker with global traffic knowledge can easily detect the proxies, as being the nodes that transmit less messages then they receive.

[Ozturk et al. 2004] and [Kamat et al. 2005] proposed several techniques to address the problem of preserving the location privacy of source sensors. The attacker is assumed to be mobile, able to snoop on transmissions and thus move toward the source of events. The first technique proposed uses probabilistic flooding in order to improve the privacy achieved by a basic flooding algorithm. The second technique, uses fake messages in the flooding process in order to mislead the attacker toward another source. The last technique, called phantom flooding, requires a source to send each event notification on a random walk in the graph. The last hop on the random walk is then required to flood the event. The phantom flood mechanism is similar to the Crowds solution, introduced in [Reiter and Rubin 1998] (see the Anonymizers paragraph for a discussion of Crowds). We note that our protocols can be in conjunction with these techniques. However, their weakness rests in the assumption that the attacker is external: if the attacker is able to corrupt nodes, its chance of thwarting these defense mechanisms increases significantly. This is because a corrupted node can report to the attacker its participation in a random walk for any source sensor.

[Shao et al. 2007] proposed pDCS, a privacy extension for data centric sensor networks (DCS). Instead of storing the events reported by sensors in a central database, in DCS the data is stored inside the network. Data is stored based on its attributes (e.g., data type, geographic location), on (storage) sensors decided based on a mapping function. An instance mapping function is the Geographic Hash Table [Ratnasamy et al. 2003]. Since in DCS systems the mapping function is public, anyone can infer the storage location for any events of interest. The solution

proposed by [Shao et al. 2007] is to use a keyed mapping function for events and to store the data on storage sensors using a key known only to detection sensors. They propose three different keyed mapping functions, (i) group-key-based mapping, (ii) time-based mapping and (iii) cell-based mapping.

There are two differences between pDCS and our work. First, the system model is different. We assume only external queries that go through central servers providing access to sensor events, whereas in pDCS queries originate from the network (either sensors or user carried mobile hosts able to connect to sensors in their vicinity). Second, the problem that we address is orthogonal to pDCS. Since in our model all client queries go through the central servers, our concern is with hiding the locations of sensors of interest from the servers and from third parties, including external attackers that may have access to traffic information. Note that our work can be used in conjunction with pDCS. For instance, we could use pDCS to obliviously (to an attacker) store sensor events in the network and then use our techniques to export network events to external clients.

[Deng et al. 2004] focus on two attacks on sensor networks. The first attack uses traffic analysis to detect the location of the base station. The second attack uses this knowledge to subject the base station to DoS attacks, by blocking communication to and from the base station. To address the first problem, the authors propose anti-traffic analysis techniques. These include (i) hiding the packet destination address using encryption, (ii) decorrelating packet sending times by requiring nodes to randomly permute the order of the packets they receive from their neighbors before transmitting them further and (iii) controlling packet sending rates between parent and children sensor nodes. We note that these mechanisms can be used in conjunction with our solutions, to further prevent in-network traffic analysis by external attackers. For the second attacks, [Deng et al. 2004] propose the use of multiple base stations and of multiple paths.

[Anand et al. 2005] study the problem of probabilistically quantifying the security of sensor network protocols, with respect to sensor data distributions and network topologies. Assuming a model where data from the sensors is continuous and time-varying, this work focuses on an attacker whose goal is to infer the values aggregated by the network. Moreover, the focus is on techniques to tolerate compromised nodes and even mislead the attacker. Proposed techniques include sending spurious data from sensors and filtering it at the aggregation point, data cloaking (perturbing the data by a predefined offset) and attribute value correlations (when the adversary does not know the correlation). The goal of our work is different, since we do not attempt to mislead an adversary that needs to infer sensor data aggregates. Instead, we focus on hiding the sensing locations of interest for external clients that access the network through central servers.

[Tao et al. 2003] and [Ladd et al. 2002] consider the problem of using wireless LAN location-sensing for security applications. They present a server-side indoor location sensing system, where "snooper" devices observe the signal strength of packets received from a rogue machine in order to infer its location. During a training, off-line phase, two algorithms are deployed in order to process signal strength readings collected by snoopers from a target device. In the context of our work, these algorithms can be used by attackers to derive the locations of queried sensors. However,

the need for a training phase and for snoopers moving through the network, makes these attacks vulnerable to discovery.

[Gruteser and Grunwald 2003] study the related problem of supporting anonymous usage of location-based services. The authors proposed a middleware architecture that can be implemented by a centralized location broker service. The solution prevents the location information transmitted by a user from being used to identify the user. The solution adjusts the resolution of location information along spatial or temporal dimensions to meet certain anonymity requirements. It prevents precise location information from being used to identity the users. Our work uses a completely different definition of privacy. Our main concern consists in hiding the regions of interest and access patterns of clients.

[Hoh et al. 2007] study the problem of protecting the privacy of users periodically reporting their coordinates from probe vehicles equipped with GPS devices. The work proposes a time-to-confusion metric to quantify the privacy of a set of location traces and then propose an uncertainty-aware algorithm that can guarantee a specified maximum time-to-confusion. We note that these techniques cannot be readily applied to our setup, since the sensors we consider are fixed and do not report their location, assumed to be already known.

Privacy Preserving Data Aggregation The data mining field studies the problem of extracting useful information from large amounts of data, usually collected from many different sources. Often enough, the knowledge mining operations have to be performed while preserving the privacy of the data owners. In this domain, the work of [Wang et al. 2004] is similar to our approach, being centered on the concept of query sets. Specifically, a query set $\{q_1, ..q_n\}$ is said to be safe if a data mining server cannot infer private data from answers to queries $q_1, .., q_n$. Since a dynamic inference of query set safety is a complex operation, static variants have been proposed. However, note that the queries $q_1, .., q_n$ are performed on data stored on a single data warehouse server. This is different from our work where the data of interest needs to be retrieved from remote sensor nodes, reachable only through other sensors. For extensive surveys on privacy preserving approaches to data mining, we suggest the work of [Verykios et al. 2004; Zhang and Zhao 2007].

Of particular interest to our work is a variant of privacy preserving aggregation, of data collected in wireless sensor networks. The work of [He et al. 2007] studies the problem of computing an in-network private sum operation of values reported by sensors. The work proposes two ingenious methods. The first method, CPDA, forms clusters of neighboring nodes and takes advantage of commutativity properties of polynomials in order to allow cluster heads to compute the sum of values reported by cluster members. The cluster heads however are unable to infer the individual values reported by honest cluster members if at least two members are honest. The second method, SMART, reduces the computation complexity of CPDA at the expense of additional traffic overhead. In SMART, each reporting sensor splits its value in several "slices" and sends each slice (except one which it keeps to itself) to a different sensor. Each sensor buffers a certain number of slices before aggregating and forwarding the result toward the sink. Both methods are private and allow the sink node to correctly reconstruct the sum of all the values reported by the sensors, in a network without packet losses. Our work is different

in concept, since we provide sensor network clients with the ability to privately query remote sensors or regions. Each query targets a single region in the network. While multiple reports from sensors belonging to the region of interest may be aggregated, we do not attempt to provide the client with an aggregate of readings from the entire network.

Anonymizers: The first anonymous system, the Mix-Net design, was proposed by Chaum [1981]. The system hides the correspondence between senders and receivers of messages by repeatedly encrypting messages with the public keys of a predefined set of mixes. A mix is a server that decrypts received encrypted messages with its corresponding private key and reorders them before forwarding them to the next mix. Dingledine *et al.* [2004] introduced Tor, an Internet anonymizing system. Tor sequentially creates circuits of "onion routers", pseudo-randomly chosen from a predefined set of servers. The set and state of onion routers is maintained by multiple directory servers. Freedman and Morris [2002] proposed Tarzan, a system providing anonymity in peer-to-peer networks. Also based on Chaum's Mix-Net, Tarzan hides sender/receiver relationships through the use of dynamically and pseudo-randomly chosen sets of relaying peers. Moreover, nodes can join and leave at any time, without the use of entry points.

While the multi-server solution proposed in this paper resembles a mix-net, we note two major differences. First, our solution does not require dynamically establishing a cascade of routers for each region to be queried. This operation involves a high communication overhead, prohibitive for a wireless sensor network. Instead, our solution requires two servers to establish for each client a secret key shared with each sensor. This operation is performed only once, when the client registers. The cost of this operation is then amortized over long query sequences. Second, the simplicity of our solution stems from a different trust model. Specifically, access to the sensor network is only given through a small (two) set of servers. This is clearly not the case of Internet or peer-to-peer anonymizers.

In Section 3.6 we have described Crowds, a system providing anonymity for web users and proposed two methods for combining it with SPYC. In the case where the two servers are trusted not to collaborate, SPYC not only provides full client privacy, but it is also more traffic efficient than any Crowds implementation. This is because when using Crowds, with non-zero probability, a query needs to be forwarded around the network at least once.

Note that existing, general purpose Internet anonymizers could be used in conjunction with our solutions, by hiding the client identities from servers. However, the privacy provided is a function of the number of clients that simultaneously query the network.

Database PIR: Database private information retrieval from a single curious-but-honest server with unlimited computation power can only be achieved by transferring the entire database [Chor et al. 1995]. Chor *et al.* [1995] proposed database replication among multiple non-cooperating servers as a framework for reducing communication between the client and servers. Several solutions have been proposed for the replicated database environment [Chor et al. 1995; Ambainis 1997; Ishai and Kushilevitz 1999; Beimel et al. 2002]. For a comprehensive analysis of database PIR we refer the reader to the survey of Gasarch [2004].

Our SPYC protocol (see Section 3) also uses two non cooperating servers to provide fully private querying mechanisms. However, we address the problem of providing client privacy in wireless sensor networks, thus facing different communication and computation constraints. Since each server holds a local copy of the database, PIR strives to minimize the client/server communication overhead. Since sensor readings are not locally available on the servers and due to sensor battery constraints, an essential goal of our query privacy solution is to reduce both the communication overhead between servers and sensors and the computation required from sensors.

7. CONCLUSIONS

In this paper we have studied the problem of preserving the privacy of clients querying sensor networks, through untrusted servers. We have classified two attack models. For the honest-but-curious server model, we have presented the SPYC protocol that guarantees full query privacy as long as the servers do not collaborate with each other. Otherwise, or in light of malicious attackers, we have proposed several metrics for quantifying query privacy. Then, we have extended SPYC with the query transform technique. According to the metrics proposed, we have investigated a suite of practical transforms.

Our TinyOS simulations have shown the efficiency of the SPYC protocol, with reasonable overhead when compared to a straightforward, non-private query mechanism. When query transforms were employed to provide further obfuscation, our simulation results have indicated that various tradeoffs between privacy and efficiency levels can be achieved by selecting a suitable transform and tuning the transform parameters.

REFERENCES

Ti msp430 specification. `http://focus.ti.com/lit/ml/slab034m/slab034m.pdf`.

AMBAINIS, A. 1997. Upper bound on the communication complexity of private information retrieval. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP)*. 401–407.

ANAND, M., IVES, Z., AND LEE, I. 2005. Quantifying eavesdropping vulnerability in sensor networks. In *DMSN '05: Proceedings of the 2nd international workshop on Data management for sensor networks*.

BEIMEL, A., ISHAI, Y., KUSHILEVITZ, E., AND RAYMOND, J.-F. 2002. Breaking the o(n1/(2k-1)) barrier for information-theoretic private information retrieval. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*. 261–270.

CHAUM, D. L. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24,* 2, 84–90.

CHOR, B., GOLDREICH, O., KUSHILEVITZ, E., AND SUDAN, M. 1995. Private information retrieval. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*.

DENG, J., HAN, R., AND MISHRA, S. 2004. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*.

DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. 2004. Tor: The second-generation onion router. In *13th USENIX Security Symp*.

FREEDMAN, M. J. AND MORRIS, R. 2002. Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. 193–206.

GASARCH, W. 2004. A survey on private information retrieval. The Bulletin of the EATCS, 82:72–107.

GEOSS. Taking the Pulse of the Planet: EPA's Remote Sensing Information Gateway. `http://www.epa.gov/geoss/`.

GRUTESER, M. AND GRUNWALD, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of MobiSys*.

HE, W., LIU, X., NGUYEN, H., NAHRSTEDT, K., AND ABDELZAHER, T. 2007. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*.

HEINZELMAN, W. R., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*.

HOH, B., GRUTESER, M., XIONG, H., AND ALRABADY, A. 2007. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*.

HOWE, B. M., MCGINNIS, T., KIRKHAM, H., AND MASSION, G. 2003. Sensor networks for cabled ocean observatories. In *Proceedings of the 3rd International Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC)*.

ISHAI, Y. AND KUSHILEVITZ, E. 1999. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 79–88.

KALISKI, B. 2003. TWIRL and RSA key size. `http://www.rsasecurity.com/rsalabs/node.asp?id=2004`.

KAMAT, P., ZHANG, Y., TRAPPE, W., AND OZTURK, C. 2005. Enhancing source-location privacy in sensor network routing. In *Proceedings of ICDCS*. 599–608.

LADD, A. M., BEKRIS, K. E., RUDYS, A., MARCEAU, G., KAVRAKI, L. E., AND WALLACH, D. S. 2002. Robotics-based location sensing using wireless ethernet. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*.

LEVIS, P., LEE, N., WELSH, M., AND CULLER, D. 2003. Tossim: Accurate and scalable simulation of entire tinyos applications. In *ACM SenSys*. 126–137.

LOOKING. The Laboratory for the Ocean Observatory Knowledge INtegration Grid (LOOKING). `http://lookingtosea.ucsd.edu/`.

MALINEN, J. Md5 c code.

MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. 2006. Ultra-low power data storage for sensor networks. In *Proceedings of IPSN*.

NOPP. National Oceanographic Partnership Program. `http://www.nopp.org/`.

OCEAN.US/IOOS. The National Office for Integrated and Sustained Ocean Observations. `http://www.ocean.us/`.

ORION. Ocean observatories initiative. `http://www.joiscience.org/ocean_observing`.

OZTURK, C., ZHANG, Y., AND TRAPPE, W. 2004. Source-location privacy in energy-constrained sensor network routing. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*.

POLASTRE, J., SZEWCZYK, R., AND CULLER, D. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of IPSN*.

RATNASAMY, S., KARP, B., SHENKER, S., ESTRIN, D., GOVINDAN, R., YIN, L., AND YU, F. 2003. Data-centric storage in sensornets with ght, a geographic hash table. *Mob. Netw. Appl. 8,* 4.

REITER, M. K. AND RUBIN, A. D. 1998. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur. 1,* 1, 66–92.

SHAO, M., YANG, Y., ZHU, S., AND CAO, G. 2008. Towards statistically strong source anonymity for sensor networks. In *Proceedings of IEEE INFOCOM*.

SHAO, M., ZHU, S., ZHANG, W., AND CAO, G. 2007. pDCS: Security and privacy support for data-centric sensor networks. In *IEEE INFOCOM*.

Tao, P., Rudys, A., Ladd, A. M., and Wallach, D. S. 2003. Wireless lan location-sensing for security applications. In *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security.*

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. 2004. State-of-the-art in privacy preserving data mining. *SIGMOD Rec. 33,* 1.

Wang, L., Jajodia, S., and Wijesekera, D. 2004. Securing olap data cubes against privacy breaches. In *Proceedings of the IEEE Symposium on Security and Privacy.*

Woo, A., Tong, T., and Culler, D. 2003. Taming the underlying challenges of multihop routing in sensor networks. In *ACM SenSys.*

Yang, Y., Shao, M., Zhu, S., Urgaonkar, B., and Cao, G. 2008. Towards event source unobservability with minimum network traffic in sensor networks. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security.*

Zhang, N. and Zhao, W. 2007. Privacy-preserving data mining systems. *Computer 40,* 4.