Query privacy in wireless sensor networks

Bogdan Carbunar, Yang Yu, Larry Shi, Michael Pearce, Venu Vasudevan
Applications Research Center, Motorola Labs
Schaumburg, IL, 60195
{carbunar,yang,larry.shi,michael.pearce@motorola.com,cvv012}@motorola.com

***Abstract*— 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 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 investigate two architectures and their corresponding trust models. For the first model, consisting of multiple, mutually distrusting servers governing the network, we devise an efficient protocol, SPYC, and show that it provides full query privacy. For the second model, where all queries are performed through a single server, we introduce two metrics for quantifying the privacy achieved by a client's query sequence. We propose a suite of practical algorithms, then analyze the privacy and efficiency levels they provide. Our TOSSIM simulations show that the proposed query mechanisms are communication efficient while significantly improving client query privacy levels.**

I. 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 NEPTUNE [1], GEOSS [2], ORION [3], Ocean.US and IOOS [4] and LOOKING [5] are building ocean 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 these sensor networks is shaped by two factors. First, multiple organizations (e.g., government agencies, universities and companies) are involved, acting both as funding sources and primary investigators. Even though network owners need to collaborate for administrative purposes, they may not fully trust each other, due to diverging interests (e.g. GEOSS [2] involves 61 countries, NOPP [6] involves DARPA, the Department of State and the Department of Homeland Security among others). Second, external organizations interested in the areas monitored by the sensor network may be willing to pay for access to sensor readings.

Providing support for third party (client) queries raises privacy and efficiency issues. Clients may not be willing to disclose their areas of interest or access patterns, whereas the owners of the network want to preserve the network's resources. Consider the case of a server providing access to an under-water sensor network and an oil company with an interest in the network. The client's program may consist of several constructs of the type illustrated below, using SQL/TinyDB syntax. In this example, the oil company is interested in temperature readings of sensors placed in the rectangle of coordinates [10..20,30..40], every 20 seconds, for 500 seconds.

```
SELECT nodeid, temperature
  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 regions and types of readings of interest may be used by servers against the client's interests. The client's access patterns may further disclose its execution program. For instance, the server may discover that a client is interested only in readings from sensors placed in a certain region (e.g., the rectangle in the above example) or accessing multiple regions in a certain order.

The straightforward solution of the client always querying the entire network but keeping only readings from areas of interest imposes a significant load on the network. This is especially relevant for wireless sensor networks whose functionality is defined by the limited battery available to sensors.

In this paper we propose several efficient mechanisms for privately querying wireless sensor networks. We investigate two network models, one where access to sensor readings is provided by a single organization and one where any of multiple, mutually distrusting organizations can perform this operation. In our model, each organization is represented by a server with direct access to the sensor network. For the latter case, we propose SPYC, a protocol that decomposes a client's interaction with the network into two tasks. The first task, of privately naming each sensor, requires two servers, but is executed only when the client joins the system. The second task, requiring only one server, takes advantage of the virtual name space in order to privately access sensors of interest and is performed for each client query. Using standard cryptographic techniques, the client hides from the servers both the virtual sensor names and the actual sensors queried. We show that if the servers do not cooperate, this solution achieves full privacy.

In the former, single server case, we start our work with the observation that all queries reach the server unencrypted, thus making privacy possible only through the use of 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. We propose two metrics for measuring privacy, present a suite of practical transforms and study their efficiency and privacy achieved.

The paper is organized as follows. In Section II we survey existing research. In Section III we describe our system model and introduce the privacy problem we consider. In Section IV, we propose an algorithm, SPYC, for the multiple server trust model and in Section V we study the single server case. Section VI contains our single and multi server simulation results. Finally, Section VII concludes the paper.

II. Related Work

Anonymizers: The first anonymous system, the Mix-Net design, was proposed by Chaum [7]. 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.* [8] 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.

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. 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.

Reiter and Rubin [9] introduced Crowds, which provides anonymity for web users. It groups users into a large, geographically distributed group, that issues requests to web servers on behalf of group members. It is difficult to adapt the random walk of Crowds to sensor networks. Since Crowds members may be geographically distributed, a simple forward operation may traverse multiple routers. While feasible in the internet, this operation could generate heavy traffic in a sensor 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 [10]. Chor *et al.* [10] 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 [10]–[13]. For a comprehensive analysis of database PIR we refer the reader to the survey of Gasarch [14].

The solution proposed in Section IV 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 client/servers communication. Since sensor readings are not locally available on the servers and due to sensor battery constraints, the main goal of our query privacy solution is to reduce both the communication overhead between servers and sensors and the computation required from sensors.

Location Privacy in Sensor Networks: Gruteser and Grunwald [15] propose a middleware architecture that can be implemented by a centralized location broker service to support anonymous usage of location-based services. It prevents the location information transmitted by a user from being used to re-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.

Kamat *et al.* [16] propose a technique called phantom routing to protect the source location of sensor network communication, particularly, the location of the reporting sensors. The technique uses localized flooding instead of full scale flooding, to prevent attackers from discovering the source sensor by eavesdropping on hop-by-hop wireless communications. Our work proposes a definition of privacy and protection that is much broader, since source privacy is only one aspect of our privacy solution. The phantom routing technique is orthogonal to the solutions proposed in our work. Thus, it may be combined with our solution to prevent physical side-channel exploits that try to defeat the protection by physically tracing the source of sensor messages.

III. 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 use the curious-but-honest server model, where servers may record client data and use it to deduce client behavior. Thus, clients do not trust the server(s) with their queries. We assume sensors are bootstrapped with the public keys of the servers and install only code that is correctly signed by one of the servers. In the multi server case, we assume that the servers may communicate in order to manage the network, but do not trust each other and do not share client information. Moreover, each server may also query the network on behalf of its organization and intends to keep the other servers unaware of its interests.

Each server installs its execution code on each sensor and inspects the code installed by the other organizations. Furthermore, each server constantly monitors the sensor network in order to detect abnormal behavior. This includes looking for corrupted sensors or communication eavesdropping attacks, performed by other servers or third-party intruders. This model is natural not only for governmental and military departments with well defined privacy and secrecy rules, but also for cooperating commercial companies with rigid privacy contracts.

We assume an existing routing tree in the underlying network, over which queries are disseminated from the root and sensing results are gathered back to the root of the tree. In this work we consider on-demand sensor network queries, consisting of acquiring sensor readings from specific regions of interest. While the solutions we present support both on-demand queries and registration of interest in regions of interest, in the following we focus only on client queries.

Without loss of generality, we assume that the entire area covered by the network is divided into a set of n regions, identified by $R = \{r_1, r_2, \ldots, r_n\}$. Each query targets a subset of regions specified through their identifiers. For ease of analysis, we assume that each query targets a single region. Let $Q = [q_1, q_2, \ldots, q_l]$ denote a query sequence of length l. The region of interest for a query q_i, $i = 1, \ldots l$, 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 r_i is queried in Q.

We define a query transform as a function $T : Q \mapsto 2^R$, where 2^R is the power set of R. Thus, a transform of $q_i \in Q$ is represented as $T(q_i) = m$, where $m \subseteq R$. The transform is said to be *correct* if $g(q_i) \in m$. The size of a transform is defined as the size of m, denoted as $|m|$. 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 for which the value $|m_i|$ is constant. An example transform is T_h, where $T_h(q_i) = R$, $i = 1, \ldots, l$. T_h requires the server to always query all the regions.

Transform Cost: 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 evaluate $C(T)$, we consider the regions R to be organized in a grid shape. The servers are placed at one corner of the grid. We assume that 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 [17]).

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. Moreover, $C(T_h)$ represents the upper bound on the cost of any transform.

IV. Multiple Servers

In this section we propose SPYC, an efficient solution providing full privacy guarantees when multiple servers, owned by different, mutually distrusting organizations, provide access to the sensor network. SPYC decomposes the query process into two orthogonal tasks and assigns each task to the servers providing access to the network (see Figure 1). The first task, executed once for each client, consists of constructing a virtual naming space

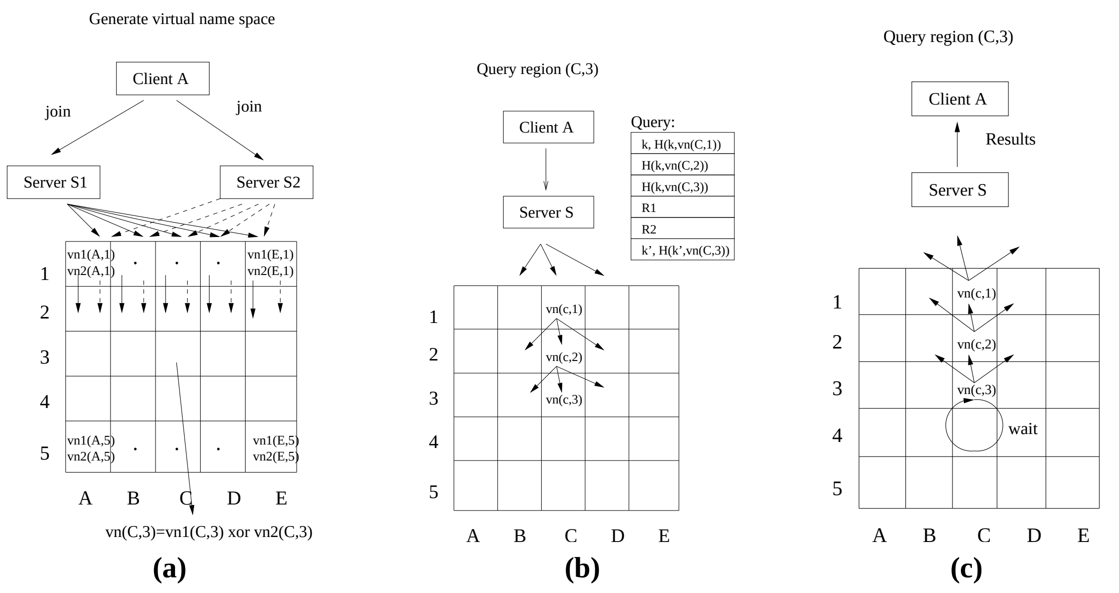


Fig. 1. Operation of the two server solution. (a). Client registers with the service, by contacting two servers S_1 and S_2 and requesting the generation of a new virtual name space for network regions. (b). Client performs query using any server S. The query is obfuscated using virtual region names, unknown to S and a fresh, public key K. Sensors in a region whose keyed hashed name matches one of the hashes in the query packet, further broadcast the packet. (c). The target region (C,3) satisfies the condition included in the last field of the packet, that is, $H(k', vn_C^3)$ equals the value in the last field. Sensors in that region do not need to further broadcast the packet, but delay their answer for a time interval equivalent to traveling over two more hops. Then, they send the results of the query to their parent region.

mapping to actual region identifiers. The second task, executed for each client query, consists of routing in the sensor network based on virtual region names. Initially, after completion of the network deployment step and before clients can join the system, the following procedure is executed by each server S.

Initialization: Let sp be a system-wide security parameter. Server S generates an asymmetric encryption key pair (k_S^{priv}, k_S^{pub}) and stores k_S^{pub} on each sensor. Moreover, it stores on each sensor in a region r_i a symmetric encryption key $k_i(S)$. S also generates a seed uniformly at random from the space $\{0,1\}^{sp}$ and uses it to initialize a pseudo-random number generator G_S. The following operation is then executed every time a client joins the system.

Task 1. Virtual Name Space Generation: 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. Both servers perform the same operation, exemplified for S_j, $j = 1, 2$. Server S_j generates n pseudo-random numbers $vn_A^{S_j}(r_i)$, $i = 1..n$, using G_{S_j}. $r_1, .., r_n$ denote the regions in the sensor network. S_j sends to each sensor in region r_i the following message

$$\mathtt{ADD}(\mathtt{A}, \mathtt{S_j}, \mathtt{E}_{\mathtt{k_i(S_j)}}(\mathtt{A}, \mathtt{vn}_\mathtt{A}^{\mathtt{S_j}}(\mathtt{r_i}))),$$

When receiving an ADD message, a sensor in region r_i first verifies S_j's encryption. If it verifies, it stores $vn_A^{S_j}(r_i)$ under client A. This step could be performed using specialized mechanisms such as the ones included in the SPINS [18] suite. When a sensor in region r_i has received virtual names from both S_1 and S_2 for client A, it stores $vn_A^{S_1}(r_i) \oplus vn_A^{S_2}(r_i)$ as its virtual name under client A. When this process completes, S_j sends client A the region-based topology of the sensor network (including a routing tree) and a table $(r_i, vn_A^{S_j}(r_i))$, $i = 1..n$, of mappings between actual region names and virtual names. The client can then build the virtual name of each region r_i as the XOR of the names received from S_1 and S_2, $vn_A(r_i) = vn_A^{S_1}(r_i) \oplus vn_A^{S_2}(r_i)$.

The following operation is executed each time a registered client needs to perform a query. Even though we use source routing to forward queries, our solution could be used in conjunction with other routing protocols.

Task 2. Region-Based Source Routing: Let A's query refer to region r_q and let $r_q^1, .., r_q^p$ be the p ancestor regions of r_q in the routing tree of the sensor network, that is, the regions that need to forward packets from the root sensor to r_q. Let l be the height of the routing tree ,i.e, the maximum number of regions a packet needs to traverse from the root to any region. A picks a server S, that could be either S_1 or S_2 or a different server, if available, to perform the query. A also chooses two new keys K and K' uniformly at random from the space $\{0,1\}^{sp}$ and sends the following ROUTE packet to S

$$\mathtt{ROUTE}(\mathtt{A}, \mathtt{K}, \mathtt{H}(\mathtt{K}, \mathtt{vn_A}(\mathtt{r_q^1})), .., \mathtt{H}(\mathtt{K}, \mathtt{vn_A}(\mathtt{r_q^p})), \mathtt{H}(\mathtt{K}, \mathtt{vn_A}(\mathtt{r_q})),$$

$$\mathtt{R_1}, .., \mathtt{R_{l-p-1}}, \mathtt{K'}, \mathtt{H}(\mathtt{K'}, \mathtt{vn_A}(\mathtt{r_q})))$$

H is used to denote a one-way function, *e*.g., a cryptographic hash function. $R_1, .., R_{l-p-1}$ are pseudo-random numbers generated using G_A. Their size is equal to the size of the output of the function H, thus concealing the distance between QE and the region of interest, r_q. While not explicitly included in the above ROUTE packet, the packet also contains the query for r_q, encrypted with key $key_q = H(vn_A(r_q))$. Only the client and sensors in r_q can derive key_q. A chooses fresh K and K' values for each of its queries.

Server S sends the ROUTE packet to the root of the network that on its turn forwards it using a broadcast primitive. A sensor s from region r_i that receives this packet compares $H(K, vn_A(r_i))$, where K is the second field of the packet, with the third field of the packet, $H(K, vn_A(r_q^1))$. If there is no match, it drops the packet. Otherwise, it verifies whether $H(K', vn_A(r_i))$, where K' is the one to last field of the packet, equals the last field of the packet, $H(K', vn_A(r_q))$. If they differ, s strips the third field, $H(K, vn_A(r_q^1))$, from the packet, and uses its local knowledge to forward the resulting packet toward sensors in neighboring regions. Sensors receiving the forwarded ROUTE packets record the source region as their parent region, and repeat this "compare and forward" procedure to route the packet to r_q.

If the test $H(K', vn_A(r_i)) = H(K', vn_A(r_q))$ holds, s belongs to the region of interest, r_q. Only the client and sensors from r_q can derive this information. s performs the specified query, appends $vn_A(r_i)$ to the result and encrypts the resulting string with key $key_i = H(vn_A(r_i))$, producing the following RSLT packet,

$$\mathtt{RSLT}, \mathtt{E}_{\mathtt{key_i}}(\mathtt{reading(s)}, \mathtt{vn_A(r_i)}, \mathtt{n(r_q)})$$

where $n(r_q)$ denotes the number of sensors in region r_q. In order to hide from server S the distance to the client's region of interest, s fakes the sending of ROUTE broadcast packets to bogus regions $R_1, .., R_{l-p-1}$, by waiting $(l - p - 1)(2T_t + T_p)$ seconds before sending the RSLT packet to its parent region. In this equation, 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). Sensors in region r_q send their RSLT packets to the parent region of r_q using a broadcast primitive. The sensors in the parent region forward RSLT packets on their turn to their own parent region and so on, until the packets reach server S and then client A. A uses key $H(vn_A(r_q))$ to decrypt the received RSLT packets. A valid decrypted message consists of a query result followed by $vn_A(r_q)$ and $n(r_q)$.

To see why SPYC provides full client privacy, when the servers do not collaborate, consider the following argument. Server S learns nothing from the queries, even if $S = S_1$ or $S = S_2$. This is because S_1 and S_2 can compute the virtual names of sensor only if they cooperate. Moreover, a client changes the keys K and K' during each query. Even if the client queries the same region, r_i, multiple times, the value $H(K, vn(r_i))$ will change for different K values. If S can find K_1 and K_2 such that $H(K_1, vn(r_i)) = H(K_2, vn(r_i))$ for any virtual region name $vn(r_i)$, it can also reverse the cryptographic hash function, which is a assumed to be computationally infeasible.

All queries have the same length, l. This, along with timed delay used in the protocol conceal the distance between the root sensor and the queried regions. Encryption of sensor readings hides possibly known associations between sensor readings and existing regions. Encryption also authenticates the readings, since only the client and sensors in region r_q know $vn_A(r_q)$. Moreover, a malicious server S cannot drop arbitrary reports since any RSLT packet contains the number of RSLT packets the client should receive, $n(r_q)$.

A. Discussion

We analyze several properties of SPYC, propose optimizations, then describe several attacks and defenses.

Storage Considerations: Considering infeasible an exhaustive search in the space $\{0, .., 2^{80} - 1\}$ [19] we restrict the size of virtual names of regions to 80 bits. This enables a sensor with 0.1MB available memory to simultaneously support more than 10000 clients [1].

By truncating the output of 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 diameter. The truncation generates a chance of region name collisions of 2^{-16}. However, the chance of collisions decreases exponentially with the distance between the root node 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.

Region Heads: A possible mechanism for further reducing communication costs is to elect a region head for each region. The ROUTE packets are transmitted via region heads only, until they reach the head of the intended region. The head in the intended region

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

broadcasts the ROUTE packet to all sensor nodes in its region. The sensor readings of these nodes are then transmitted in unicast packets back to the head. The head may choose to aggregate these readings before transmitting them back to the server via region heads. This has the additional benefit of hiding from querying server S the number of sensors in the queried region, which, in specific circumstances may leak information about the region. Several existing techniques for cluster-leader election and routing in sensor networks [17] can be applied in this context.

Virtual Name Leaks: Client privacy can be compromised if sensors can report to servers their virtual names or details of queries forwarded or executed. However, similar to clients, servers can also query the network on behalf of their organizations, thus, needing the same privacy guarantees. This makes the servers willing to allow sensors to run only server signed code and provide mechanisms for sensors to report the code they are running, along with the corresponding server signature, when requested. These mechanisms can allow any participant to discover corrupted sensors and trace insiders benefiting from covert channels inserted in the code.

Client Leaks: A malicious participant could collaborate with corrupt clients or could register fake clients with servers S_1 and S_2 in order to retrieve sensor names. However, since each client maintains its own virtual name space for the sensor network and can only share its own view of the network, the privacy of non-cooperating clients would not be affected. Moreover, the client registration process could be made more rigorous, for instance requiring a public key certificate, to prevent registration of bogus clients.

Battery Level Monitoring: Servers playing an administrative role may be able to query the remaining battery levels of sensors. If a server performs such an operation before and after a query, it can detect the exact sensors involved. This is however an expensive operation, requiring not only a flood but also a complete converge-cast operation. To prevent this attack, various restrictions may be placed on the battery level query types. For instance, if servers can query battery levels periodically, e.g. only 3 times a day, associations between clients and queried regions will be obfuscated as a function of the number of client queries performed in 8 hours.

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 the source or relayers of packets. Since the accuracy of predictions increases with the number of such devices deployed, these attacks are costly and prone to detection by other servers or clients.

V. Single Server

While the multi-server model considered above provides full privacy guarantees as long as the servers do not cooperate, a question worth answering is what can be achieved when a single server governs the sensor network. The solution space for the single server model is bounded by two transforms. The first transform, T_h, maps the query of any region to the entire space R, which provides maximum privacy and incurs maximal cost. Also, we can verify that maximum privacy can only be achieved by T_h. 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 $C(T_l) = cl\sqrt{n}$.

A. Practical Privacy Metrics

We are interested in various practical solutions within the space defined by T_l and T_h, achieving 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.

1) Spatial Privacy Level: Using the notations of Section III, 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 $\tilde{M} = \tilde{Q}$.

2) Temporal Privacy Level: From a temporal privacy perspective, we consider the query frequency of regions in 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 of 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 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 $\bar{p}_i$ 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

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

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 distortion between the two distributions. Thus, we intend to maximize the cross entropy to conceal frequency patterns of 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.

B. Privacy v.s. Cost Tradeoffs

We study practical solutions in the space bounded by T_l and T_h, i.e., for any transform T in this space,

$$\begin{aligned} 1 \leq \quad & S(T) && \leq \frac{n}{s} \ , && (2) \\ 0 \leq \quad & R(T) && \leq \log s - H(Q) \ , && (3) \\ cl\sqrt{n} \leq \quad & C(T) && \leq cnl\sqrt{n} \ . && (4) \end{aligned}$$

In the following we consider query sequences generated either off-line or on-line. Off-line queries are often used for routine monitoring or 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.

1) Off-Line Queries: In the case of off-line queries, the entire initial query sequence is known *a priori*. 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, FT transforms each query into a randomized set of regions, which includes the original region to be queried. That is, for every $q_i \in Q$, FT 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 $Q = [r_1, r_3, r_5, r_1, r_3, r_5, \ldots]$, the transformed sequence can be $M = [\{r_1, r_8, r_9\}, \{r_3, r_4, r_7\}, \{r_2, r_5, r_6\}, \{r_1, r_5, r_{10}\} \ldots]$.

Given a query sequence Q (whereby s and n are fixed), the number of unique regions in $FT(Q)$ is a function of z, denoted as $f(z) = |\tilde{M}|$. Using hyper-geometric distribution, we estimate the expected value of $f(z)$ as

$$E[f(z)] = n(1 - \alpha^{l+1}) \ , \quad (5)$$

where $\alpha = 1 - \frac{z}{n}$ and l is the length of the query sequence. Thus, $E[S(RT)] = \frac{E(f(z))}{s}$. The temporal privacy of FT is $R(RT) = -\sum p_i \log(\frac{a}{p_i} + b)$, where $a = \frac{z-1}{s(z-1)+n-z}$ and $b = \frac{n-z}{s(z-1)+n-z}$. It can be verified that while lower-bounded by zero, $R(RT)$ increases with s and z. Moreover, for a fixed s and p_i's, $R(RT)$ is maximized to $\log(s) - H(Q)$ 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.

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,

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	$\frac{n}{s}$	$\log(n) - H(Q)$	$cln\sqrt{n}$

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 III.

In Section III, the transform function is defined as $T : R \mapsto 2^R$, indicating the same transform to be applied for a specific region, regardless of its position in the query sequence. Here, we define the function as $T : R \times \mathbb{Z} \mapsto 2^R$, 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 2, 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. 2. Pseudo-code for DT

VI. SIMULATION RESULTS

We implemented and evaluated the proposed techniques on a packet-level TOSSIM [20]. Using TOSSIM's empirical communication model [21], 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 $r_1, \ldots, r_{100}$.

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

In our simulations, queries were always injected by the server into the head of r_1. The queries were then routed towards the destination region through a multi-hop route consisting only of region heads. 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 following, details of the routing protocol are described for single-server and multi-server cases, respectively.

A. Single-Server

We implemented over TOSSIM a tree-based routing scheme to efficiently deliver query packets to region heads. 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. We assumed that every head knows the region heads in its subtree. Maintaining the subtree information is practical, since the number of heads is much smaller than the number of sensor nodes.

We implemented a query protocol for simultaneously querying multiple regions. A query packet consisting of all queried regions was generated at the server and injected to the head of r_1. Based on its subtree information, the head of r_1 divided the query into at most two subqueries and transmitted them to its children heads accordingly. The children further divided the queries, if necessary, and transmitted them towards the intended destinations. When data packets were generated at the queried regions, the routing process was reversed to transmit the data packets back to the server. This process allowed us to effectively reduce the number of transmitted packets.

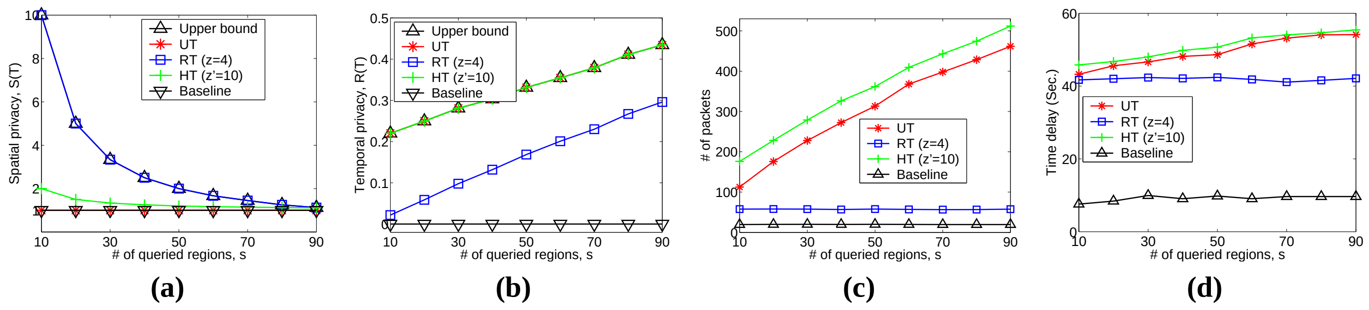


Fig. 3. Simulation results for Union, Random, and Hybrid Transforms (900 sensor nodes, $n = 100$, $l = 300$, $z = 4$, $z' = 10$

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. In the off-line case, the query sequences were fed to the transforms as a whole, while in the on-line case, the transform algorithm (Figure 2) processed the queries in a real-time fashion.

We recorded the number of packet transmissions required to execute the queries over the routing tree. Note that we did not count the number of intra-transmissions, because intra-region communication requires much less power than inter-region communication, due to a shorter transmission range.

1) Off-Line Queries: For the off-line experiments we chose $z = 4$ for RT and $z' = 10$ for HT. T_l was used as the baseline, and 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 3.

It can be observed that the spatial privacy of RT 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 number of packet transmissions, was almost 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 detailed data trace, we discovered that the increased delay of UT and RT was mainly because of severe packet collision resulted from the large amount of communication requests, which were resolved by time consuming re-transmissions. This indicated (1) UT and RT are 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 achieves lower temporal privacy, it is more practical for larger values of s. This is due to RT's constant costs. However, for relatively small values of s, UT and HT should be preferred, since they provide higher temporal privacy levels for a small cost.

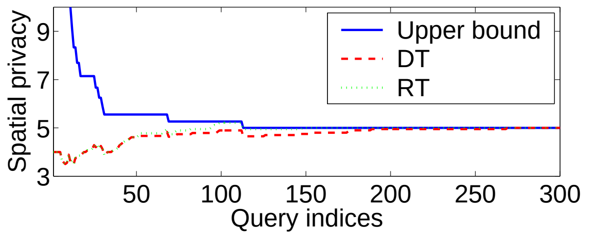
2) On-Line Queries: For the on-line algorithm, we generated a query sequence of length 300 using the method described above, with $s = 20$, $z = 4$, and $t = 50$. We recorded the temporal and spatial privacy levels resulting from the on-line transform after the execution of each query. Figure 4 shows 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. Similar trends were observed in the results.

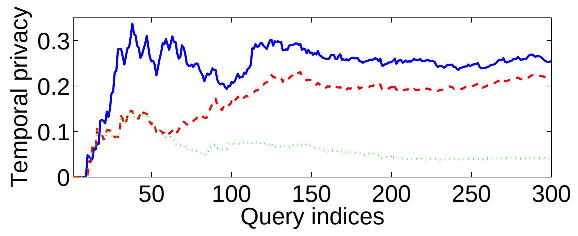
B. Multiple Server

We now document our experience with the TOSSIM implementation of SPYC (see Section IV). We simulated runs of SPYC for the network configuration described at the beginning of this section, consisting of 900 sensor nodes placed in 100 regions. We successfully injected queries, in the shape of hashed source routing packets, into the network and retrieved data from the network.

The structure of a query packet was the following: 1 byte contained the client identifier, 2 bytes were used for every key (K and K') and hash value (indicating



(a) Spatial privacy



(b) Temporal privacy

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

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 (shown in Figure 3(c)). Moreover, the observed time delay required to complete a query was approximately twice of the baseline delay shown in Figure 3(d). This was because on average, a baseline query required both query and data packets to be transmitted over 10 hops. However, due to the mandatory delay in our query scheme, SPYC always mimicked the transmission of query and data packets over 20 hops.

VII. Conclusions

In this paper we studied the problem of preserving the privacy of clients querying sensor networks, through untrusted servers. We proposed single and multi-server systems and trust models. Our TinyOS simulations show that our multi-server solution introduces reasonable overhead when compared to a straightforward, non-private query mechanism. For the single server model, the simulations show how the parameters of our algorithms can be used to provide clients with various tradeoffs between privacy and efficiency levels.

References

[1] Bruce M. Howe and Timothy McGinnis. Sensor networks for cabled ocean observatories. `http://www.neptune.washington.edu/pub/whats_neptune/whats_neptune.html`.

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

[3] Orion. `http://www.orionprogram.org/`.

[4] The National Offi ce for Integrated and Sustained Ocean Observations. `http://www.ocean.us/`.

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

[6] National Oceanographic Partnership Program. `http://www.nopp.org/`.

[7] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symp*, 2004.

[9] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[10] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

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

[12] Yuval Ishai and Eyal Kushilevitz. 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*, pages 79–88, 1999.

[13] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-Francois Raymond. 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*, pages 261–270, 2002.

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

[15] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of MobiSys*, 2003.

[16] Pandurang Kamat, Yanyong Zhang, Wade Trappe, and Celal Ozturk. Enhancing source-location privacy in sensor network routing. In *Proceedings of ICDCS*, pages 599–608, 2005.

[17] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-effi cient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, 2000.

[18] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[19] Burt Kaliski. TWIRL and RSA key size. `http://www.rsasecurity.com/rsalabs/node.asp?id=2004`, 2003.

[20] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *ACM SenSys*, pages 126–137, Nov. 2003.

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