# Privacy- and Integrity-Preserving Range Query in Wireless Sensor Networks

Yao-Tung Tsou[†][*], Chun-Shien Lu[†] and Sy-Yen Kuo[*]
[†]*Institute of Information Science, Academia Sinica, Taipei 115, Taiwan*
[*]*Department of Electrical Eng., National Taiwan University, Taipei 106, Taiwan*

***Abstract*—A large-scale wireless sensor network constructed in terms of two-tiered architecture, where cloud nodes take charge of storing sensed data and processing queries with respect to the sensing nodes and querists, incurs security breach. This is because the importance of cloud nodes makes them attractive to adversaries and raises concerns about data privacy and query result correctness. To address these problems, we propose an efficient approach, namely EQ (efficient query), which mainly prevents adversaries from gaining the information processed by or stored in cloud nodes, and detects the compromised cloud nodes when they misbehave. EQ can not only achieve the goals of data privacy and integrity preserving but also ensure the secure range query without incurring false positive. For data privacy preserving, EQ presents an order encryption mechanism by adopting stream cipher to encrypt/decrypt all sensed data such that a cloud node can only process issued queries over stored data in the encryption domain. For data integrity/completeness, we manipulate a data structure of XOR linked list (X2L), which allows a querist to verify the integrity of retrieved data via the so-called verification information, *i.e.*, neighborhood difference in a storage-efficient manner. We demonstrate the feasibility and efficiency of EQ via experiments conducted on TelosB prototype sensor platform running TinyOS 1.1.15 and comparisons with state-of-the-arts.**

***Keywords*-Encryption; Integrity; Privacy; Query; Sensor network; Security.**

## I. Introduction

### A. Background and Related Work

Cloud based online storage service is more and more popular not only in traditional networks but also in wireless sensor networks. Because cloud storage would involve storing data on multiple cloud nodes (CNs) ($e.g.$, cloud servers or devices with abundant storage), data privacy and integrity preserving become important issues and thereby how to deal with compromised CNs is a major concern in this paper.

Recently, there are some researches [1] [2] [3] [4] [9] that have been proposed to prevent the adversaries from obtaining information stored or processed in CNs. In other words, each CN is necessary to process the data sensed by sensing nodes (SNs) and the queries issued from querists (QTs) in the encryption domain. However, some drawbacks still exist in these state-of-the-arts. While data privacy and query result integrity of the range query in [1] [2] [3] can be preserved exquisitely, owing to the use of bucket partitioning proposed by [8], their query result completeness is achieved in different manners. Unfortunately, they share some common weaknesses: (1) The bucketing scheme used in their approaches may cause false positive. More precisely, querists would receive some useless data values. (2) If a CN is compromised, it may leak some information to adversaries such as bucket tags. It allows adversaries to obtain a reasonable estimation on how many data values fall into a range or the sensed data of SNs and the queries of QTs. (3) Power and space consumption grows exponentially with the data size.

In [4], SafeQ-Basic and its optimized version, SafeQ-Bloom, are proposed. Because SafeQ-Basic applies prefix membership verification which needs to produce many hash values by using MD5 or SHA-1, it would cause large communication, computation, and storage overheads. Although SafeQ-Bloom employs Bloom filter to represent hash values in order to reduce communication and storage overheads, it leads to more computational overhead.

To address the above problems, Jiajun $et\ al.$ [9] have proposed a secure, efficient, and flexible range query scheme, namely SEF. They employ the Order-Preserving Symmetric Encryption (OPSE), which is a deterministic encryption scheme whose encryption function preserves the numerical order of the plaintexts [10], to preserve data privacy. For preserving authenticity and integrity of query results, an Authenticity & Integrity tree (AI tree) inspired by $Merkle$ hash tree (MHT) is constructed. The construction of AI trees, however, will lead to extra computational overhead, in particular when the number of operations of hashing increases. Moreover, since sensed data that are stored in NAND flash memory need to be read on a pagewise basis, it must cost extra loading for reading the redundant data values. Additionally, they also need to reserve some pages for storing digests, so the utilization of storage is not efficient.

### B. Overview and Contributions of This Paper

In this paper, we propose an efficient range query (EQ) scheme in WSNs with privacy and integrity preserved. EQ presents an order encryption mechanism (OEM), which possesses the characteristic of bucket scheme, to provide data privacy. The OEM technique allows a CN to retrieve the desired data according to the stored data and queries in the encryption domain. In order to verify the query result integrity/completeness, EQ applies the data structure X2L, which simply processes the operation of Exclusive-

OR (XOR), to construct neighborhood differences for query result verification of a querist. The comparisons among some well-known methods and our proposed method are shown in Table. I. We find that EQ is more efficient than the prior works in many aspects such as communication overhead, computation overhead, and storage overhead, while achieving privacy preserving, integrity verification, and accurate query result (QR) that are comparable with [1] [2] [3] [4] [9]. The main contributions of this paper include:

- We provide a novel mechanism for secure data storage and query processing in the encryption domain.
- We present an efficient data structure to guarantee the integrity/completeness of query results.
- We introduce a concept of cloud storage in two-tiered network architecture and our approach can be easily integrated into current mobility control protocols without raising much additional overhead.
- In comparison with the existing schemes, our approach performs better in requiring less power and storage.

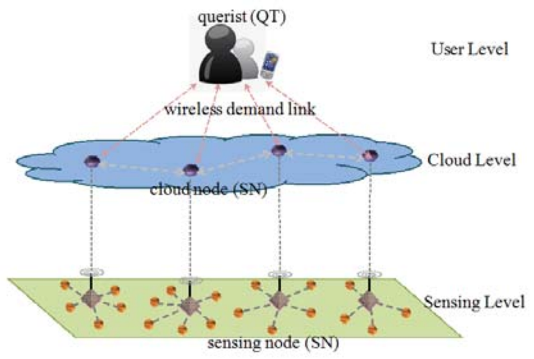

Figure 1. Wireless sensor network with two-tiered architecture.

## II. System and Attack Models

### A. System Model

As shown in Fig. 1, our system consists of three kinds of nodes: querist (QT), cloud node (CN) and sensing node (SN), which are set up in the user level, cloud level and sensing level, respectively. The network is partitioned into physical cells, according to the geographic location, wherein each SN belongs to one cell, and each cell contains a CN which is in charge of other SNs in the cell. Depending on concrete applications, cells may overlap such that a SN in the overlapping region is affiliated with more than one CN. In the user level, a querist/user can use a sensor (*e.g.*, TelosB) or mobile device (*e.g.*, smart phone) to start up a range query via the wireless demand link to get the desired data. The cloud level can be viewed as a large-scale cloud data center. Each CN in the cloud level collects the data sensed from the SNs in the ground. However, in order to achieve the goal of privacy-preserving, CNs are only able to manipulate encrypted sensed data. Moreover, SNs are limited in storage, energy, and computation; they are responsible for sensing data such as temperature, humidity, the number of ground shaking, the volume of a specific time interval, etc.

In our system, we formulate the query demand as: $query = (S_i, t, [min, Max])$, where $S_i$ denotes a SN's ID from which the data retrieval is desired, $t$ represents the time interval in which the data are monitored, and the query range is denoted by the $[min, Max]$.

### B. Attack Model

Many attacks, including passive eavesdropping, bogus-message injection, replay, and jamming attacks, may be launched by an adversary. However, we do not focus on these attacks because they can be efficiently defensed in [5] [6] [7]. In this paper, we aim at the security issues on the problem of compromised CNs. We assume that the adversary can arbitrarily compromise CNs. Once compromising a CN, the stored sensed data may be exposed to the adversary. The adversary is also able to instruct the compromised CN to return bogus or incomplete result in response to range queries issued from the QT. On the other hand, if a SN is compromised by an adversary, it may leak the sensed data to the adversary and/or be controlled by the adversary to launch attacks. Although the amount of sensed data stored in a SN is limited, the security breach problem still threaten to WSNs. We already addressed this problem in [7]. Here, we are interested in the security breach issue, wherein compromising CNs brings more damages to WSNs than compromising SNs.

## III. The Proposed Scheme

We propose an order encryption mechanism (OEM) that is simple and provably secure to allow efficient query of encrypted data. Let $\{d_i\}_{i=1}^{m}$ denote the plaintexts sensed by a SN and let their corresponding ciphertexts be $Enc_{k_i}(d_i)$'s, $Enc(.)$ denotes a symmetric encryption function. OEM supports the property of decomposing operation as:

$$\begin{aligned} & Enc_K(d_1|d_2|...|d_m) \\ & = Enc_{k_1}(d_1)|Enc_{k_2}(d_2)|...|Enc_{k_m}(d_m), \qquad (1) \end{aligned}$$

where "|" represents a concatenation operation.

OEM consists of two phases: order mapping and data encryption. In the order mapping phase, all $m$ data values (plaintexts) with each of $n$ bits are mapped to a region selected from an interval $[l_b, u_b]$. The length of the interval is chosen by the network owner to be proportional to the range of sensed data. Then, the network owner partitions the interval for enquiry to $\gamma$ ($\gamma > 0$) equivalent and consecutive non-overlapping regions with order preserved, i.e., the regions $[l_b, x_b]$ and $[y_b, u_b]$ satisfy $x_b < y_b$ and $x_b - l_b = u_b - y_b$. Each region $r_\rho$ is associated with a sequence number $\rho$ and a key $\{k_\rho, 1 \leq \rho \leq \gamma\}$, and $K$ is composed of $\{k_\rho\}_{\rho=1}^{\gamma}$. In

Table I
COMPARISONS OF OUR METHOD WITH STATE-OF-THE-ART ($N$: NUMBER OF NODES; $m$: NUMBER OF SENSED DATA)

| | Security | Accurate QR | Detection Probability | Communication Overhead | Computation Overhead | Storage Overhead |
|---|---|---|---|---|---|---|
| **Encoding approach** (bucket scheme) [1] | Reasonable estimation | No | always $\approx 1$ | $O(N)$ | $O(1)$ | Exponential |
| **Spatial Crosscheck** (bucket scheme) [2] [3] | Reasonable estimation | No | vary significantly | $O(N^2)$ | $O(1)$ | Exponential |
| **Temporal Crosscheck** (bucket scheme) [2] [3] | Reasonable estimation | No | vary significantly | $O(N^{1.5})$ | $O(1)$ | Exponential |
| **Hybrid Crosscheck** (bucket scheme) [2] [3] | Reasonable estimation | No | always $\approx 1$ | $O(N^2)$ | $O(1)$ | Exponential |
| **SafeQ** [4] | Hard estimation | Yes | always $\approx 1$ | $O(N)$ | $O(m^2)$ (HMAC operations and/or Bloom filter generation) | Exponential |
| **SEF** [9] | Hard estimation | Yes | always $\approx 1$ | $O(N)$ | $O(m \log m)$ (hashing operations and AI tree generation) | Linear |
| **EQ** (our method) | Hard estimation | Yes | always $\approx 1$ | $O(N)$ | $O(m)$ | Linear |

particular, the key $k_\rho$ is a parameter decided by the network owner in the initialization phase and used for generating a keystream. The key $k_\rho$ also works like an index so that a CN can search data values in response to the queries of QTs. Recall that the traditional bucket scheme will expose the indices to CNs, but the indices are concealed from CNs in OEM. More precisely, CNs cannot conjecture what are the stored data from interpreting the indices. In the encryption phase shown in Fig. 2, it consists of a finite state machine, which is initialized with a $n$-bit initialization value (IV). In order to generate the keystream with the size as same as the plaintext, an IV and $k_\rho$ are used as the parameters of block cipher encryption. In particular, for the same plaintext that needs to produce different ciphertexts, a non-repeating counter is used as an IV, which is involved as an input in the encryption operation. Therefore, the semantic security can be satisfied. Each data value $\{d_i, 1 \leq i \leq m\}$ is first XORed with a keystream, which is a pseudorandom binary sequence. Subsequently, the resulting bits are then randomly permuted to obtain the encrypted data value ($E_{k_\rho}(d_i)$). For the decryption process, it just works inversely with respect to the encryption process.

In the following, we describe the behaviors of SN, CN, and QT in detail. Without loss of generality, we focus on one cluster consisting of $N$ SNs denoted as $\{SN_i\}_{i=1}^N$, and a CN. The OEM function is setup for the QT and $\{SN_i\}_{i=1}^N$ at system initialization. We use the notation $Enc_K(D)$ to denote the encryption of plaintexts $D = \{d_i\}_{i=1}^m$, where the OEM function is $Enc(\cdot)$ and $K = \{k_\rho\}_{\rho=1}^\gamma$.

***Sensing Node Behavior***. To prevent a CN from knowing the actual sensed data, the SN applies OEM to encrypt sensed data for confidentiality. To perform OEM, we map each $n$-bit data value $\{d_i, 1 \leq i \leq m\}$ sensed by a SN to a respective region. Each region will be assigned a key $\{k_\rho, 1 \leq \rho \leq \gamma\}$ selected from $K$ and $k_\rho$ is then used to generate a $n$-bit keystream to encrypt the data fallen into that region. For example, if the interval [1,10] is partitioned into 5 equivalent and consecutive non-overlapping regions, then the key set $\{k_\rho\}_{\rho=1}^5$ generated by the key generator

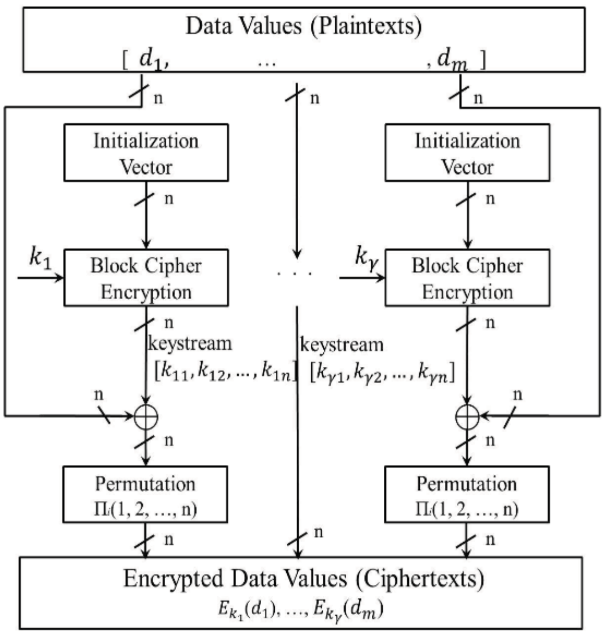
Data Values (Plaintexts)
$[\ d_1, \ \ldots \ , d_m\ ]$
n
Initialization Vector
Initialization Vector
$k_1$
$k_\gamma$
Block Cipher Encryption
Block Cipher Encryption
keystream $[k_{11}, k_{12}, \ldots, k_{1n}]$
keystream $[k_{\gamma1}, k_{\gamma2}, \ldots, k_{\gamma n}]$
Permutation $\Pi_i(1, 2, \ldots, n)$
Permutation $\Pi_i(1, 2, \ldots, n)$
Encrypted Data Values (Ciphertexts)
$E_{k_1}(d_1), \ldots, E_{k_\gamma}(d_m)$

Figure 2. Encryption of sensed data.

consists of 5 keys, $k_1$, $k_2$, $k_3$, $k_4$, and $k_5$. Consider the sensed data set $D = \{d_i\}_{i=1}^5 = \{4, 1, 7, 8, 9\}$ collected by a $SN_1$ with ID 1 in time slot $t = 2$, we illustrate the behavior of the sensing node on the left-hand side of Fig. 3(b). The SN directly maps the sensed data set $D$ to the ordered intervals (step 1) and encrypts them using the stream cipher proposed in Fig. 2 (step 3). As for the integrity of query result, we apply a data structure X2L to preserve and verify the integrity of a query result (step 3). Each data value $\{d_i, 1 \leq i \leq m\}$ will be concatenated with its neighborhood difference ($d_{i-1} \oplus d_{i+1}$) [13], expressed as $(d_i | d_{i-1} \oplus d_{i+1})$. In other words, $d_i | d_{i-1} \oplus d_{i+1}$ instead of single $d_i$ will be encrypted for confidentiality. Note that both $d_0$ and $d_{m+1}$ are dummy in that the neighborhood difference for $d_1$ and $d_m$, respectively, can be built. In addition, in order to prevent a CN from responding empty or false query results when

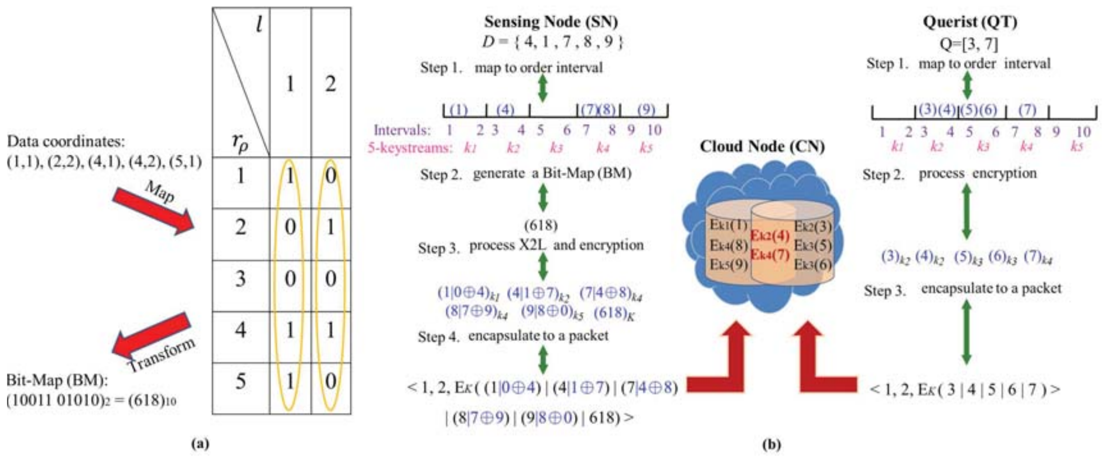
Data coordinates:
(1,1), (2,2), (4,1), (4,2), (5,1)
Map
Transform
Bit-Map (BM):
(10011 01010)2 = (618)10
(a)
Sensing Node (SN)
D = { 4, 1, 7, 8, 9 }
Step 1. map to order interval
Intervals:
5-keystreams:
Step 2. generate a Bit-Map (BM)
(618)
Step 3. process X2L and encryption
Step 4. encapsulate to a packet
< 1, 2, EK( (1|0⊕4) | (4|1⊕7) | (7|4⊕8) | (8|7⊕9) | (9|8⊕0) | 618) >
Cloud Node (CN)
Querist (QT)
Q=[3, 7]
Step 1. map to order interval
Step 2. process encryption
Step 3. encapsulate to a packet
< 1, 2, EK( 3 | 4 | 5 | 6 | 7 ) >
(b)

Figure 3. System model for data retrieval in the encrypted domain: (a) A processes for constructing a BM and (b) a flow for data retrieval.

actually there are data values that correspond to the query of QT, we construct a Bit-Map (BM) table for QT to check query result incompleteness (step 2). Actually, the BM table will be transferred to be a number that acts like a global hash of the locations of sensed data values. For example, in Fig. 3(a), we define the data coordinate as ($r_\rho$, $l$), which represents that a data value is fallen into the $l$-th location of the region $r_\rho$. Thus, this coordinate in the BM table will be denoted by 1. Following the above rule, the five data values in step Fig. 3(a) will be recorded in the Bit-Map table, and the binary information $(1001101010)_2$ of Bit-Map table will be transferred to a decimal digit $(618)_{10}$, which is defined to be a BM here. Finally, the $SN_1$ encapsulates its ID 1, time slot 2, encrypted data values, and a BM into a packet expressed as:

$$\langle 1, 2, E_K((1|0 \oplus 4)|(4|1 \oplus 7)|(7|4 \oplus 8)|(8|7 \oplus 9)|(9|8 \oplus 0)|618)\rangle, \tag{2}$$

and sends the packet to its closest CN (step 4). Note that for the encrypted data value in a packet (Eq. (2)), it is composed of $m$ ciphertexts via concatenation, each of which has $n$ bits. More specifically, the plaintext mentioned here is, in fact, composed of a sensed plaintext and its corresponding neighborhood difference.

***Cloud Node Behavior***. After receiving a packet from a SN, the CN will extract each data item, which is composed of a $n$-bit data value (say, $d_i$) and its neighborhood difference (say, $d_{i-1} \oplus d_{i+1}$), by decomposing the ciphertext in the packet via Eq. (1). Since each data item (say, $d_i|d_{i-1} \oplus d_{i+1}$) has $2n$ bits, it is easy for the CN to retrieve each data item by slicing the ciphertext every $2n$ bits, where the first $n$-bit is the data value ($d_i$) and the last $n$-bit is the neighborhood difference ($d_{i-1} \oplus d_{i+1}$). Here, the data value in each data item is denoted as $E_{k_\rho}(d_i)$ $(1 \leq i \leq m,\ 1 \leq \rho \leq \gamma)$. When the QT wants to query the CN for the data, for example, in the range $[3, 7]$ sensed by a $SN_i$ with ID $i = 1$ in time slot $t = 2$, it builds the query packet $< 1, 2, E_K(3|4|5|6|7) >$ and sends the query to the CN. Upon receiving the query packet, the CN searches for the requested data stored in its database in the encryption domain. Specifically, the encrypted data stored in the CN's database intersected to the query range is the QR for the QT, such as $E_{k_2}(4)$ and $E_{k_4}(7)$, as showed in CN of Fig. 3(b). More specifically, all the retrieved data associated with their neighborhood differences are packaged together in a packet as the QR, wherein each entry is denoted as $(d_i|d_{i-1} \oplus d_{i+1})$. On the other hand, if no matched data can be found, the CN needs to send an encrypted BM (*e.g.*, $\{E_K(618)\}$ in Fig. 3(b)) as QR for the sake of checking whether the CN cheats the QT by sending a fake QR. Finally, the CN responds the QR to the QT.

It should be noted that in [4], their neighborhood chaining technique only concatenates a data value with its left neighbor. However, our strategy directly concatenates a data value with its neighborhood difference (left neighbor XOR right neighbor) such that the neighborhood difference can verify the left and right neighbors of a data value simultaneously. Moreover, our method does not need to append verification object [4] to query results for integrity detection; thus, extra communication overhead can be saved. On the contrary, our method only needs to send a BM once no data are match during the data retrieval process conducted in a CN.

***Querist Behavior***. After a QT receives the QR obtained via Eq. (1), there are four cases that the QT has to consider for integrity detection. Let the region size used in order mapping of OEM be $\tau$. Let the set of actual query results be $\{(d_\ell|d_{\ell-1} \oplus d_{\ell+1})_{k_{\lceil \frac{d_\ell}{\tau} \rceil}}$, $(d_{\ell+1}|d_\ell \oplus d_{\ell+2})_{k_{\lceil \frac{d_{\ell+1}}{\tau} \rceil}}$, ..., $(d_\hbar|d_{\hbar-1} \oplus d_{\hbar+1})_{k_{\lceil \frac{d_\hbar}{\tau} \rceil}}\}$, where $d_\ell < d_{\ell+1} < \cdots < d_\hbar$. For any encrypted item $(d_j|d_{j-1} \oplus d_{j+1})_{k_{\lceil \frac{d_j}{\tau} \rceil}}$, its can be decrypted to attain $(d_j|d_{j-1} \oplus d_{j+1})$ using the key $k_{\lceil \frac{d_j}{\tau} \rceil}$. The rules of integrity detection for (decrypted) query data are described as follows.

- If there exists $\ell < j < \hbar$ but $(d_{j-1}|d_j \oplus d_{j-2}) \notin QR$, then the QT can know the missing $d_{j-1}$ by examining from the available items $(d_j|d_{j-1} \oplus d_{j+1})$ and $(d_{j+1}|d_j \oplus d_{j+2})$ to attain $d_{j-1} = d_{j-1} \oplus d_{j+1} \oplus d_{j+1}$, where the neighborhood difference $d_{j-1} \oplus d_{j+1}$ comes from $(d_j|d_{j-1} \oplus d_{j+1})_{k_{\lceil \frac{d_j}{\tau} \rceil}}$ and the plaintext $d_{j+1}$ comes from $(d_{j+1}|d_j \oplus d_{j+2})_{k_{\lceil \frac{d_{j+1}}{\tau} \rceil}}$.
- If $(d_\ell|d_{\ell-1} \oplus d_{\ell+1}) \notin QR$, then the QT can detect the missing $d_\ell$ by calculating $d_\ell = d_\ell \oplus d_{\ell+2} \oplus d_{\ell+2}$ from the items $(d_{\ell+1}|d_\ell \oplus d_{\ell+2})$ and $(d_{\ell+2}|d_{\ell+1} \oplus d_{\ell+3})$.
- If $(d_\hbar|d_{\hbar-1} \oplus d_{\hbar+1}) \notin QR$, then the QT can detect the missing $d_\hbar$ by calculating $d_\hbar = d_{\hbar-2} \oplus d_\hbar \oplus d_{\hbar-2}$ from the items $(d_{\hbar-1}|d_{\hbar-2} \oplus d_\hbar)$ and $(d_{\hbar-2}|d_{\hbar-3} \oplus d_{\hbar-1})$.
- If $QR$ only contains a BM, then the QT has to recover the BM table (like the one in Fig. 3(a)) from decrypting the received encrypted BM to further verify whether there does not exist the data values corresponding to the query of QT.

## IV. System Analysis

We assume that the adversary can arbitrarily compromise CNs. Once compromising a CN, the data stored in it may be exposed to the adversary. The adversary is also able to instruct the compromised CN to return bogus or incomplete results in response to the range queries issued from the QT. In order to conceal from adversaries the actual data values collected by SNs and the actual queries issued by the querist/base station, we provide OEM for preserving privacy. In addition, to verify the return bogus or incomplete results, we apply the data structure of X2L. We have privacy analysis, integrity analysis, and communication overhead as follows.

*(1) Privacy Analysis:* We begin with a review of the concept of negligible function. It is used in security analysis to characterize the probability that a computationally-bounded adversary successfully breaks a computationally secure encryption scheme [11] [12] [14]. A function $f(\cdot)$ is negligible if for every polynomial $p(\cdot)$, there exists an $\aleph$ such that for all integers $n > \aleph$, $f(n) < 1/p(n)$. According to modern cryptography [12], events that occur with negligible probability are unlikely to occur and they can be ignored for all practical purposes. In security analysis, the parameter $n$ typically determines the length of the secret key and the security of the encryption scheme. While increasing the parameter $n$, the probability that an adversary successfully breaks the encryption scheme decays fast. Specifically, if the parameter $n$ is reasonably large, $n \geq 128$ bits, it is computationally difficult for an adversary to infer sensed data content by breaking the cryptographic ciphers in practice.

A proof of our scheme that satisfies the property of negligible function is as follows.

*Proof.* We show that for any positive polynomial $p$ $f(n) < 1/p(n)$ holds for sufficiently large $n$. Let $d$ be the degree of $p$ and assume that an adversary succeeds in violating a security condition with $f(n){:=}(\frac{1}{2})^n$. We have:

$$\lim_{n\to\infty} \frac{f(n)}{1/p(n)} = \lim_{n\to\infty} \frac{p(n)}{2^n} \triangleq \lim_{n\to\infty} \frac{\partial^d p(n)}{\partial^d 2^n} = 0.$$

Here $\triangle$ is a $d$-fold application of L'Hôpital's rule. Since $\frac{f(n)}{1/p(n)}$ converges to 0 and $p$ is positive, for sufficiently large $n$ ($n \geq 128$) [12] [14], we have $f(n) < 1/p(n)$. Therefore, $f$ is a negligible function.

*(2) Integrity Analysis:* The goal of integrity verification here is to detect any missing data, so a fragile structure like X2L satisfy our purpose. The integrity of QR can be verified firmly, as described in the Querist Behavior of Sec. III, because the structure of X2L is extremely fragile in that any missing data can destroy it. Moreover, since X2L uses XOR operation to calculate each data value, its computational complexity is $O(m)$.

*(3) Communication Overhead:* For fair comparison, as in prior works [2], [3], we do not count the number of bits in representing data (like packet header information, bucket tags, *etc*) since the delivery of the representing data is necessary in any data collection scheme. Each SN in EQ is required to send packets to its closest CN. As a consequence, each SN actually has only one-hop broadcast containing encrypted data items once at each epoch. Since the communication overhead of the EQ will be dominated by the average hop distance between a CN and each SN, and the transmission of packets, the communication cost can be calculated as $\bar{H}(N-1) = O(N)$, where $\bar{H}$ denotes the average hop distance, and $N-1$ denotes the number of packets in a cluster.

## V. Performance Evaluation

In this section, the performance of our method is evaluated and compared with the schemes of S&L [1], SafeQ-Basic/SafeQ-Bloom [4], and SEF [9]. The performance comparison is analyzed in four aspects: (1) energy consumption of processing sensed data in a SN; (2) energy consumption of transmitting encrypted sensed data from a SN to a CN; (3) storage overhead in a CN; and (4) energy consumption of a CN in sending query result and verification information[1] to a QT. In our experiments, we do not compare with the schemes in [2] [3] because they may incur a serious security problem in that a compromised SN can easily send false bit maps to breach the integrity verification of the network. In

[1]Here, verification information mean the least significant bit of neighborhood difference and the Bit-Map associated with each plaintext.

Table II
CODE SIZE OF EQ

| | ROM | RAM) |
|---|---|---|
| **SN** | 15.4(KB) | 2.8(KB) |
| **CN** | 11.4(KB) | 0.4(KB) |
| **QT** | 19.4(KB) | 2.2(KB) |

section V-A, we present a publicly available implementation of EQ for the TelosB platform and describe performance comparisons in section V-B.

### A. Hardware Module and Experimental Setup

TelosB is composed of an MSP430 micro controller, an 802.15.4 TI wireless transceiver, 10 KB RAM, 48 KB ROM, and 1024 KB flash memory. The platform fully supports TinyOS1.X/2.X and implements a network stack. To implement security functionalities of the state-of-the-arts, we employed SHA-1 as a hash function in SafeQ-Basic/SafeQ-Bloom and SEF, and applied the DES encryption algorithm in S&L and SafeQ-Basic/SafeQ-Bloom. The OPSE function proposed in [10] was used as the underlying OPSE function in SEF. For our method, we adopted stream cipher and 128-bit keystream to encrypt sensed data values. As for network setting, the network consists of 12 TelosB motes as SNs and three TelosB motes as CNs. Each cell is composed of four SNs and one CN. Note that each cell works independently, thus, we mainly focus on one cell in the following discussions. We implemented our method in nesC code, the programming language used for TinyOS. The code size of EQ is shown in Table II.

### B. Experimental Results

As part of the measurement process, we used Agilent 34411A power meter to measure the average current (under the assumption that the voltage supply is constant) drawn by the evaluation nodes (i.e., SNs or CNs). Subsequently, the measures were displayed and recorded on a PC using the NI LabVIEW SignalExpress graphic user interface (GUI) software to communicate with the power meter.

*1) Energy consumption of processing sensed data:* As the results shown in Fig. 4, we have the following observations: (1) Our method EQ only consumes 15% more energy than S&L since EQ has to perform X2L and generate a BM for incomplete verification. (2) Compared with EQ, SEF-basic/SEF-opt consumes about 2.5 times more energy, SafeQ-Basic consumes 4 times more energy, and SafeQ-Bloom consumes 10.8 times more energy. This is because SEF-Basic/SEF-Opt has to construct the AI tree that needs to perform many hash operations and SafeQ-Basic/SafeQ-Bloom adopts the prefix membership verification strategy, which still requires a large number of hash operations. Note also that, because SafeQ-Bloom has to perform a lot of additional hash operations to obtain the Bloom filter, it consumes more energies than SafeQ-Basic.

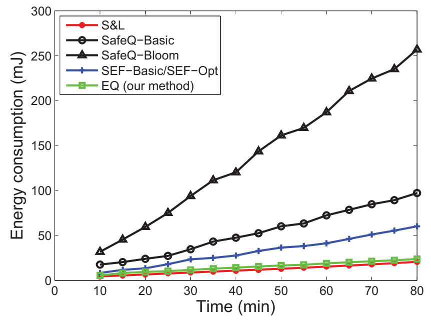

Figure 4. Average energy consumption of processing sensed data for a SN.

*2) Energy consumption of transmitting encrypted data:* Fig. 5 shows the energy consumption of transmitting encrypted data from a SN to a CN. We assumed that there were 10 data values sensed by a SN. As shown in Fig. 5, the energy consumption of EQ approximated to S&L and SEF, while there are 10 128-bit encrypted sensed data. However, since SafeQ needs to perform many operations of hashing and appends this hash value to the transmitted packet, it may cause more energy consumption when transmitting packet from a SN to a CN. Fig. 5 demonstrates that when compared with EQ, SafeQ-Basic consumes about 12 times more energy and SafeQ-Bloom consumes about 3 times more energy.

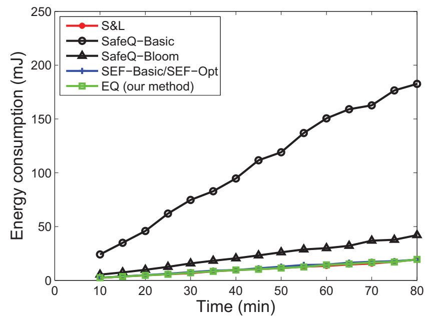

Figure 5. Average energy consumption of transmitting encrypted data from a SN to a CN.

*3) Storage overhead in a CN:* The results regarding storage overhead, including encrypted data values and verification information, in a CN are shown in Fig. 6. Although our method EQ is approximate to S&L and SEF-Basic/SEF-Opt, it is much better than SafeQ-Basic/SafeQ-Bloom. More specifically, the encoding numbers and buckets used in S&L require slight more storage consumption than EQ; SEF-Basic/SEF-Opt uses HMAC of AI tree to provide verification information and slightly consumes more storage than EQ; and SafeQ-Basic consumes about 12.6 times more storage overhead and Safe-bloom still consumes 2.1 times more storage than EQ.

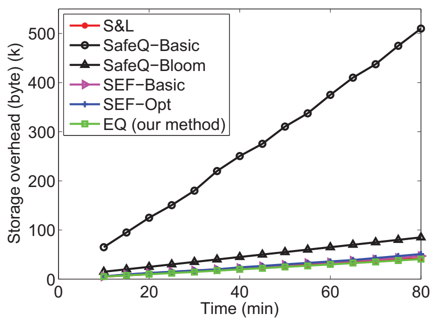

Figure 6. Average storage overhead in a CN.

*4) Energy consumption of a CN:* Fig. 7 depicts the energy consumption of a CN in returning query result and verification information to QT. We can observe that EQ consumes less energy than other methods. Specifically, SEF-Basic/SEF-Opt consumes about 1.7 times more energy and Safe-Basic/SafeQ-Bloom consumes about 3.4 times more energy than our method. Moreover, the false positive incurred by the bucking technique causes S&L to consume 1.8 times more energy than EQ.

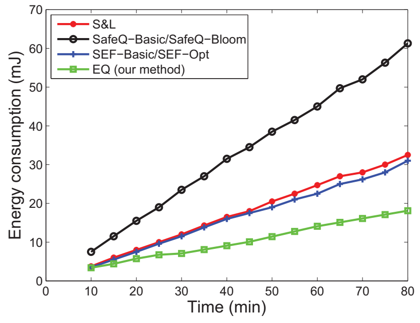

Figure 7. Average energy consumption for a CN in sending query result and verification information to a QT.

## VI. Conclusion

We propose an efficient range query scheme for two-tiered sensor networks in a privacy and integrity preserving fashion. In addition to security and privacy analysis, and communication cost analysis, we demonstrate via system implementation on TelosB prototype sensor platform that our method, when compared to the state-of-the art, consistently (1) consumes less energy consumption of processing sensed data in a SN; (2) consumes less energy consumption of transmitting encrypted sensed data from a SN to a CN; (3) requires less storage overhead in a CN; and (4) consumes less energy consumption of a CN in transmitting query result and verification information to a QT.

## Acknowledgment

C.-S. Lu was supported by National Science Council, Taiwan, under Grant NSC 98-2221-E-001-004-MY3. S-Y. Kuo is also with School of Electronic and Information Engineering, Beijing Jiaotong University. He was supported by National Science Council, Taiwan, under Grant NSC 99-2221-E-002-106-MY3, National Natural Science Foundation of China (NSFC) under Grant 60833002, and "111" Project under Grant B08002.

## References

[1] B. Sheng and Q. Li,"Verifiable privacy-preserving range query in two-tiered sensor networks," *IEEE INFOCOM*, pp. 457-465, 2008.

[2] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," *IEEE INFOCOM*, pp. 945-953, 2009.

[3] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," *Proc. 10th ACM MobiHoc*, pp. 197-206, 2009.

[4] F. Chen and A. X. Liu, "SafeQ: secure and efficient query processing in sensor networks," *IEEE INFOCOM*, pp. 1-9, 2010.

[5] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," *IEEE INFOCOM*, pp. 954-962, 2009.

[6] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: a secure sensor network communication architecture," *6th ACM IPSN*, pp. 479-488, 2007.

[7] C.-M. Yu, Y.-T. Tsou, C.-S. Lu, and S.-Y. Kuo, "Practical and secure multidimensional queries in tiered sensor networks," *IEEE Trans. on Information Forensics and Security*, vol. 6, no. 2, pp. 241-255, June 2011.

[8] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," *Proc. VLDB*, pp. 720-731, 2004.

[9] B. Jiajun, Y. Mingjian, H. Daojing, X. Feng, and C. Chun, "SEF: A secure, efficient, and flexible range query scheme in two-tiered sensor networks," *Int. Journal of Distributed Sensor Networks*, Article ID 126407, 12 pages, 2011.

[10] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," *Advances in Cryptology EUROCRYPT*, vol. 5479, pp. 224-241, 2009.

[11] W. Lu, A. Varna, and M. Wu, "Security analysis for privacy preserving search of multimedia," *IEEE ICIP*, pp. 2093-2096, 2010.

[12] J. Katz and Y. Lindell, "Introduction to modern cryptography: pprinciples and protocols," *Chapman & Hall/CRC*, 2007.

[13] P. Sinha, "A memory-efficient doubly linked list," *Linux Journal*, vol. 2005, issue 129, January 2005.

[14] T. Chen, L. Ge, X. Wang, and J. Cai, "TinyStream: A lightweight and novel stream cipher scheme for wireless sensor networks," *Proc. Int. Conf. on Computational Intelligence and Security*, 2010.