Practical and Secure Multidimensional Query Framework in Tiered Sensor Networks

Chia-Mu Yu, *Student Member, IEEE*, Yao-Tung Tsou, Chun-Shien Lu, *Member, IEEE*, and Sy-Yen Kuo, *Fellow, IEEE*

***Abstract*—The two-tier architecture consisting of a small number of resource-abundant storage nodes in the upper tier and a large number of sensors in the lower tier could be promising for large-scale sensor networks in terms of resource efficiency, network capacity, network management complexity, etc. In this architecture, each sensor having multiple sensing capabilities periodically forwards the multidimensional sensed data to the storage node, which responds to the queries, such as range query, top-k query, and skyline query. Unfortunately, node compromises pose the great challenge of securing the data collection; the sensed data could be leaked to or could be manipulated by the compromised nodes. Furthermore, chunks of the sensed data could be dropped maliciously, resulting in an incomplete query result, which is the most difficult security breach. Here, we propose a simple yet effective hash tree-based framework, under which data confidentiality, query result authenticity, and query result completeness can be guaranteed simultaneously. In addition, the *subtree sampling* technique, which could be of independent interest to the other applications, is proposed to efficiently identify the compromised nodes. Last, analytical and extensive simulation studies are conducted to evaluate the performance and security of our methods. Prototype implementation on TelosB mote demonstrates the practicality of our proposed methods.**

***Index Terms*—Multidimensional query, secure query, sensor network.**

I. Introduction

Tiered Sensor Networks

SENSOR networks are expected to be deployed in some harsh or hostile regions for data collection or environment monitoring. Since there is a possibility of unstable connection between the authority and the network, in-network storage is necessary for caching or storing the data sensed by sensors. A straightforward method is to attach external storage to each node, but it is economically infeasible to keep all the sensed data in a sensor. Therefore, various data storage models for sensor networks have been studied in the literature. In [6] and [20], a notion of two-tier sensor networks was discussed by introducing an intermediate tier between the authority and sensors. The purpose of this tier is to cache the sensed data so that the authority can retrieve the cached data efficiently, reducing unnecessary communication with sensors.

Manuscript received July 06, 2010; revised November 12, 2010; accepted January 19, 2011. Date of publication January 31, 2011; date of current version May 18, 2011. The work of C.-S. Lu was supported by the National Science Council, Taiwan, under Grant NSC 98-2221-E-001-004-MY3. The work of S.-Y. Kuo was supported by the National Science Council, Taiwan, under Grant NSC 97-2221-E-002-216-MY3, by the Excellent Research Projects of National Taiwan University under Center for Quantum Science and Engineering (97R0066-65), by Quantum Computing, Quantum Information/Communication (97R0066-67), by the National Natural Science Foundation of China (NSFC) under Grant 60833002, and by "111" Project under Grant B08002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Wade Trappe.

C.-M. Yu and Y.-T. Tsou are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, and also with the Institute of Information Science, Academia Sinica, Taipei, Taiwan.

C.-S. Lu is with the Institute of Information Science, Academia Sinica, Taipei 11529, Taiwan (e-mail: lcs@iis.sinica.edu.tw).

S.-Y. Kuo is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2011.2109384

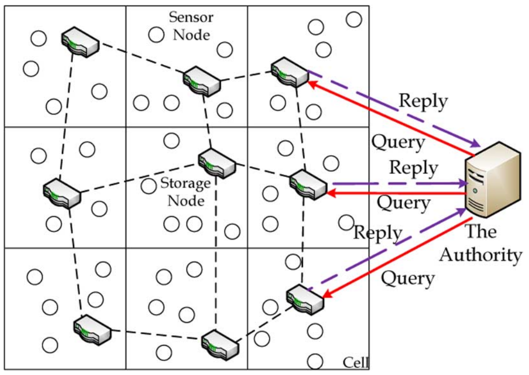


Fig. 1. Conceptual illustration of the tiered sensor network is shown. Each cell itself forms a multihop network.

The network model considered in this paper is illustrated in Fig. 1. More specifically, a small number of storage-abundant nodes, called *storage nodes*, which are equipped with several gigabytes of NAND flash storage [22], are deployed as the intermediate tier for data archival and query response. In practice, some currently available sensors such as RISE [19] and StarGate [26] can work as the storage nodes. The performance of such sensor networks, wherein external flash memory is attached to the sensors, was also studied in [14]. In addition, some theoretical issues concerning the tiered sensor networks, such as the optimal storage node placement, were also studied in [22]. In fact, such a two-tier network architecture has been demonstrated to be useful in increasing network capacity and scalability, in reducing network management complexity, and in prolonging network lifetime. More characteristics of the two-tier network model will be explained in detail in Section II.

Security Issues in Multidimensional Queries

Though a large amount of sensed data can be stored in storage nodes, the authority might be interested in only some portions of them. Thus, the authority issues proper queries to retrieve the desired portion of sensed data. As the sensed data have multiple attributes, the query could be multidimensional. Though many kinds of queries are useful in retrieving the sensed data meaningful to the authority, we restrict ourselves in this paper to discussing range query, top-k query, and skyline query, which are the most commonly used queries. Range query [11], [16], which is useful for correlating events in the network, is used to retrieve sensed data whose attributes are individually within a specified range. Top-k query [29], which is used to retrieve the sensed data whose ranking values are among the first k priority, can be used to extract the extreme sensor readings. Skyline query [5], [10], due to its promising application in multicriteria decision making, is also useful and important in environment monitoring, industry control, etc. These queries will be discussed in more detail in Section II.

Nonetheless, in the tiered network model, the storage nodes easily become the targets to be compromised because of their significant role in responding to queries. For example, the adversary can eavesdrop on the communications among nodes or compromise the storage nodes to obtain the sensed data, resulting in the breach of *data confidentiality* (DC). After the compromise of storage nodes, the adversary can also return the falsified query results to the authority, leading to the breach of *query result authenticity* (QRA). As confidentiality and authenticity can be fulfilled via off-the-shelf cryptographic algorithms, the most challenging is that the compromised storage nodes can violate *query result completeness* (QRC), creating an incomplete query result for the authority by dropping some portions of the query result.

Existing Works on Secure Queries

Secure range queries in tiered sensor networks have been recently studied. While DC and QRA of the range query in [21], [28], [33] can be preserved very well, owing to the use of the bucket scheme [7], [8], their QRC is achieved in different manners. In fact, as mentioned above, the main point in this line of research lies in how to guarantee QRC. In particular, in the tiered networks without the security concern, each sensor forwards the original sensed data to the storage node per epoch when it has sensor readings. In [21], QRC is achieved by using the encoding approach, which forces a state where, even when sensors do not generate the bucketed data, they are still required to send an unforgeable encoding number to the storage node instead of sending nothing. Thus, if the compromised storage node wants to report an incomplete query result, it has to forge the corresponding encoding numbers, which has been proven to be difficult. In [28] and [33], several crosscheck approaches were proposed. Spatial crosscheck requires that each node broadcasts its bucketed data to all the other sensors, and then forwards its own sensed data and the received sensed data to the storage node. Temporal crosscheck requires that each sensor forwards the data sensed not only at the current epoch but also at the previous several epochs. Hybrid crosscheck combines spatial and temporal crosscheck approaches. The idea behind these crosscheck approaches is that the information about the existence of a specific sensed data is distributed over all the sensed data. Consequently, to report an incomplete query result without being detected, the compromised storage node has to drop a large amount of other data, resulting in vacant epochs, within which no sensor is deemed to generate sensor readings. This will attract the attention of the authority, who will inspect the legitimacy of the storage node.

The preliminary version of this paper [31] addresses the issues of securing range, top-k, and skyline queries. In addition, very recently, by exploiting ranking and crosscheck approaches, top-k query can be secured in [32]. Nonetheless, DC is not taken into consideration in [32]. Last, to the best of our knowledge, no research effort was conducted on securing skyline query in tiered sensor networks.

Security and Efficiency Gap

As stated in [28], the encoding approach [21] is only suitable in environment monitoring scenarios, in which sensors constantly generate and report sensor readings, because the transmission of encoding numbers incurs tremendous communication overhead. As stated in [33], the encoding approach is only suitable for the one-dimensional query scenario because its communication overhead grows exponentially with the dimensionality of data. On the other hand, crosscheck approaches [28], [33] would be efficient only in the event detection scenarios, in which sensors generate sensed data only when triggered by specific events, because data broadcast will imply considerable communication overhead. In addition, as compared with the encoding approach, crosscheck approaches sacrifice the probability of detecting query result incompleteness for its applicability on multidimensional query. More importantly, in crosscheck approaches, there are many parameters that must be manually and heuristically set, restricting its practical use.

Due to their design nature, the methods in [21], [28], [33], and [32] are especially designed to secure range query and top-k query, respectively. Nevertheless, the authority could issue different kinds of queries to retrieve the desired portions of the sensed data. Therefore, their applicability is rather restricted.

Most importantly, the methods in [21], [28], and [32] only address the security issues incurred by the compromise of storage nodes with the unrealistic assumption that sensors cannot be compromised. In fact, the security impact resulting from the compromise of sensors can be considerable. Here, two kinds of specific attacks resulting from sensor compromises are considered. They are *collusion attack* and *false-incrimination attack* (Section III-B). The impact of collusion attacks, defined as the collusion among compromised sensors and compromised storage nodes, however, was only discussed in [33], wherein only a naive method, in which the authority occasionally checks if a random subset of sensors has been compromised, was proposed as a countermeasure. As indicated in [33], spatial crosscheck is subject to this attack.

A false-incrimination attack, as a kind of denial-of-service attack, can be extremely harmful but is never addressed in the literature. Unfortunately, in such an attack, the compromised sensors subvert the functionality of the secure query schemes by simply claiming that their sensed data have been dropped by the storage nodes. After that, the innocent storage nodes will be considered compromised and be revoked by the authority. It

TABLE I
COMPARISONS BETWEEN DIFFERENT SCHEMES FOR SUPPORTING SMQs (**CC**: COMMUNICATION COST; **DP**: DETECTION PROBABILITY; **DC**: DATA CONFIDENTIALITY; **RSC**: RESILIENCE AGAINST SENSOR COMPROMISES; **SQ**: SUPPORTED QUERIES)

	CC	DP	DC	RSC	SQ
Encoding approach [21]	$O(Nc^d d)$ (c: constant)	always ≈ 1	✓	×	Range Query
Spatial Crosscheck [28], [33]	$O(N^2 + Nd)$	vary significantly	✓	×	Range Query
Temporal Crosscheck [28], [33]	$O(N^{1.5}d)$	vary significantly	✓	×	Range Query
(Range) Hybrid Crosscheck [28], [33]	$O(N^2 + Nd)$	always ≈ 1	✓	×	Range Query
(Top-k) Hybrid Crosscheck [32]	$O(N^2)$	always ≈ 1	×	×	Top-k Query
SMQ-1 (this paper)	$O(N + d\sqrt{N})$	always ≈ 1	✓	×	Range+Top-k+Skyline Queries
SMQ-2 (this paper)	$O(N + d\sqrt{N})$	always ≈ 1	✓	✓	Range+Top-k+Skyline Queries

should be noted that all the methods in [21], [28], [32], and [33] suffer from false-incrimination attacks.

Contributions

Our major contributions are summarized as follows.

1) The secure multidimensional query (SMQ) schemes on the basis of the proposed hash tree-based framework are proposed to secure the range, top-k, and skyline queries in tiered networks (Section III). As shown in Table I, the parameter-free SMQ-1 scheme has the lowest communication cost among prior works in all the scenarios mentioned in prior works (environment monitoring and event detection purposes), while preserving the probability for detecting incomplete query results close to 1.
2) The security impact in the query answering scenarios incurred by sensor compromises is studied for the first time in the literature (Section III-B); *collusion attack* is formally addressed, and a new denial-of-service attack, *false-incrimination attack*, which can thwart the security purpose in all of the prior works, is first identified in our paper. Specifically, a novel technique called *subtree sampling* is introduced in SMQ-2 as a countermeasure to these two attacks.
3) In spite of several existing solutions to secure queries on tiered sensor networks, none of them have been implemented. In this paper, prototype implementation on TelosB mote demonstrates the practicality of our proposed methods.

II. SYSTEM MODEL

Network Model

As shown in Fig. 1, the sensor network considered in this paper is composed of a large number of resource-constrained sensors and a few *storage nodes*. Storage nodes are storage-abundant sensors. Similar to ordinary sensors, storage nodes also perform the sensing task and could be compromised. Storage nodes can communicate with the authority via direct or multihop communications. A *cell* (i.e., a block in Fig. 1) is a connected multihop network composed of a storage node and a number of sensors. The nodes in the network have synchronized clocks [24], and time is divided into epochs. The method to achieve the association between the storage node and sensors [20], [22] is not discussed here because the effectiveness of our proposed schemes does not rely on specific clustering algorithms.

For each cell, sensed data are forwarded along the aggregation tree [15] rooted at the storage node. The topology of the aggregation tree is known by the authority [3], [4]. Each sensor senses the data and temporarily stores the sensed data in its local memory within an epoch. If needed, at the end of an epoch, the sensors in a cell report the sensed data stored in local memory to the associated storage node responsible for answering the queries. Throughout this paper, our discussions focus on a single cell $\mathcal{C}$, composed of $N-1$ sensors, $\{s_i\}_{i=1}^{N-1}$, and a storage node $s_{\mathcal{M}}$.

Security Model

We consider the adversary who can compromise an arbitrary number of storage nodes. After node compromises, all the information stored in the compromised storage nodes will be exposed to the adversary. The goal of the adversary is to breach at least one of the DC, QRA, or QRC. We temporarily do not consider the compromise of sensors in describing SMQ-1 in Section III-A. The impact of sensor compromise on the security breach, however, will be explored in Section III-B. In particular, by compromising sensors, the adversary may attempt to launch a collusion attack and false incrimination attack, which will be defined in Section III-B. Since this paper focuses on the design for securing multidimensional query, we simply assume that the other security primitives such as broadcast authentication [13] and key establishment [30] are achievable.

Query Model

The sensed data can be represented as a d-dimensional tuple, $(A_1, A_2, \ldots, A_d)$, where $A_g, \forall g \in [1, d]$ denotes the gth attribute. The authority may issue an intended d-dimensional query to retrieve the desired portion of data stored in $s_{\mathcal{M}}$. Three types of queries, including range query, top-k query, and skyline query, are considered in this paper. For range query, its form, issued by the authority, is expressed as $\langle \mathcal{C}, t, l_1, h_1, \ldots, l_d, h_d \rangle$, which means that the sensed data to be reported to the authority should be generated by the nodes in cell $\mathcal{C}$ at epoch t, and their gth attributes, A_g's, should be within the range of $[l_g, h_g]$, $g \in [1, d]$.

Top-k query is usually associated with a scalar *ranking function*. With ranking function R, the sensed data, even if it is multidimensional, can be mapped to a one-dimensional *ranking value* individually. The top-k query issued by the authority is in the form of $\langle \mathcal{C}, t, R, k \rangle$. The goal of top-$k$ query in this paper is assumed to obtain the sensed data generated by the nodes in cell $\mathcal{C}$ at epoch t with the first k smallest ranking values.

TABLE II
NOTATION TABLE

	Description
s_i and $s_{\mathcal{M}}$	i-th sensor, and storage node
D	sensed data
$\mathcal{C}$	a cell
$A_i(A_i(c))$	i-th attribute (i-th attribute of a specific data, c)
d and w	number of attributes, and number of intervals of an attribute
$K_{i,t}$	unique key of s_i within epoch t
$(v_1, \ldots, v_d)$	bucket
$\mathcal{V}$	the set of all possible buckets
$\mathcal{A}$	the set of buckets satisfying the issued query
Y	average number of sensed data in $\mathcal{C}$
$p_i^V, p_i^\emptyset$	bucket prime of s_i
$k_{i,t}^V, k_{i,t}^\emptyset$	bucket prime of s_i within epoch t
$E_{i,t}$ and $B_{i,t}$	the set of encrypted data of s_i at epoch t, and the set of buckets of s_i at epoch t
$H_{i,t}, P_{i,t}$	the proof generated by s_i at epoch t
$\mathcal{E}_{i,t}$ and $\mathcal{B}_{i,t}$	the set of aggregated encrypted data of s_i at epoch t, and the set of aggregated buckets of s_i at epoch t
$\mathcal{H}_{i,t}, \mathcal{P}_{i,t}$	the proof collaboratively generated by s_i and its all descendants at epoch t
$\mathfrak{s}$	the set of compromised sensors
$\vartheta_{i,t}^m$ and $T_{i,t}^m$	the subtree proof generated by s_i and its descendants within m hops at epoch t, and the set of nodes contributing the formation of $\vartheta_{i,t}^m$
W_t	the set of witness nodes at epoch t
$\Upsilon_{[t_1,t_2]}$ and $\bar{\Upsilon}_{[t_1,t_2]}$	the set of nodes covered by sampled subtrees at the epochs in $[t_1, t_2]$ and the set of nodes to be inspected

For skyline query, the desired *skyline data* are defined as those not *dominated* by any other data. Assuming that smaller values are preferable to larger ones for all attributes, for a set of d-dimensional data, a datum c_i *dominates* another datum c_j if both the conditions, $A_g(c_i) \leq A_g(c_j)$, $\forall g \in [1, d]$, and $A_g(c_i) < A_g(c_j)$, $\exists g \in [1, d]$, where $A_g(c_i)$ is the gth attribute value of the datum c_i, hold. Hence, the form of the skyline query issued by the authority is given as $\langle \mathcal{C}, t \rangle$, which is used to retrieve the skyline data generated in cell $\mathcal{C}$ at epoch t.

III. SECURING MULTIDIMENSIONAL QUERIES

In this section, the proposed SMQ schemes are described. Our study is conducted evolutionarily through two steps. First, we present the SMQ-1 scheme, which constitutes the foundation of our proposed hash tree-based framework, in Section III-A. Nevertheless, SMQ-1 is secure only against the compromise of storage nodes. Thus, in Section III-B, SMQ-1 is enhanced to be SMQ-2, which is resilient against sensor compromises. Table II summarizes the notations frequently used in this paper.

A. SMQ-1

Our proposed SMQ-1 scheme consists of two phases: confidentiality-preserving data reporting (Section III-A2) and QRC verification (Section III-A3). The former is used to achieve DC and QRA properties, while the latter guarantees the QRC property. Since *bucket scheme* [7], [8] is used in the confidentiality-preserving data reporting phase, to make the paper self-contained, we will briefly review the bucket scheme in Section III-A1.

It should be noted that the assumption that only the storage nodes could be compromised is made when SMQ-1 is described. This assumption is made to ease the explanation of our hash tree-based framework and will be removed in Section III-B.

1) Bucket Scheme: Data encryption is a straightforward and common method of ensuring the DC property. Moreover, we hope that even when the adversary compromises the storage node $s_{\mathcal{M}}$, the previously stored information should not be exposed to the adversary. To this end, the keys used in encryption should be selected from a one-way hash chain. In particular, assume that a key $K_{i,0}$ is initially stored in sensor s_i. At the beginning of epoch t, the key $K_{i,t}$, which is used only within epoch t, is calculated as $\mathrm{hash}(K_{i,t-1})$, where $\mathrm{hash}(\cdot)$ is a cryptographic hash function, and $K_{i,t-1}$ is dropped. Suppose that sensor s_i has sensed data D at epoch t. One method for storing D in $s_{\mathcal{M}}$ while preserving privacy is to send the encryption $\{D\}_{K_{i,t}}$ of D with the key $K_{i,t}$. With this method, when an OCB-like authenticated encryption primitive [18] is exploited, the DC and QRA properties can be guaranteed simultaneously. Nevertheless, the query can be answered by $s_{\mathcal{M}}$ only if operations on encrypted data are possible. Hence, the *bucket scheme* proposed in [7] and [8] is used in the SMQ-1 scheme. There is a tradeoff between the communication cost and confidentiality in terms of bucket sizes because a larger bucket size implies higher DC and higher communication cost due to more superfluous data being returned to the authority. The design of optimal bucketing strategies is beyond the scope of this paper, and we refer the reader to [7] and [8] for more details.

In the bucket scheme, the domain of each attribute $A_g, \forall g \in [1, d]$, is assumed to be known in advance, and is divided into $w_g \geq 1$ consecutive nonoverlapping intervals sequentially indexed from 1 to w_g, under a publicly known partitioning rule. For ease of representation, in the following, we assume that $w_g = w, \forall g \in [1, d]$. A d-dimensional bucket is defined as a tuple, $(v_1, v_2, \ldots, v_d)$, where $v_g \in [1, w]$, $g \in [1, d]$. The sensor s_i, when it has sensed data at epoch t, sends to $s_{\mathcal{M}}$ the corresponding buckets, which are constructed by mapping each attribute of the sensed data to the proper interval index, and the sensed data encrypted by the key $K_{i,t}$.

Let $\mathcal{V}$ be the set of all possible buckets. Assume that there are on average Y and Y/N data generated in a cell and in a node, respectively, at epoch t. Assume that $D_{i,t,V}$ is a set containing all the data within the bucket $V \in \mathcal{V}$ sensed by s_i at epoch t. The messages sent from s_i to $s_{\mathcal{M}}$ at the end of epoch t can be abstracted as $\langle i, t, J_\sigma, \{D_{i,t,J_\sigma}\}_{K_{i,t}} \rangle$, where $J_\sigma \in \mathcal{V}, J_\sigma \neq J_{\sigma'}, 1 \leq \sigma, \sigma' \leq Y/N$ if there are some data sensed by s_i within epoch t. Note that s_i sends nothing to $s_{\mathcal{M}}$ at epoch t if $D_{i,t,J_\sigma} = \emptyset$, $\forall J_\sigma \in \mathcal{V}$. After that, $s_{\mathcal{M}}$, for example, can answer the range query according to the information revealed by the buckets. Assume that l_g and h_g are located within the α_gth and β_gth intervals, respectively, where $\alpha_g \leq \beta_g$, α_g, $\beta_g \in [1, w]$, and $g \in [1, d]$. The encrypted data falling into the buckets in the set $\mathcal{A} = \{(\rho_1, \ldots, \rho_d) \,|\, \alpha_g \leq \rho_g \leq \beta_g, g \in [1, d], (\rho_1, \ldots, \rho_d) \in \mathcal{V}\}$ are reported to the authority. In other words, once receiving the range query, $s_{\mathcal{M}}$ first translates the information $(l_1, h_1, \ldots, l_d, h_d)$ into the proper buckets and then replies all the encrypted data falling into the buckets in $\mathcal{A}$.

As an illustrative example, shown in Fig. 2, when s_i has sensed data 3, 4, and 11 of the gth attribute at epoch t, the message transmitted to the storage node at the end of epoch

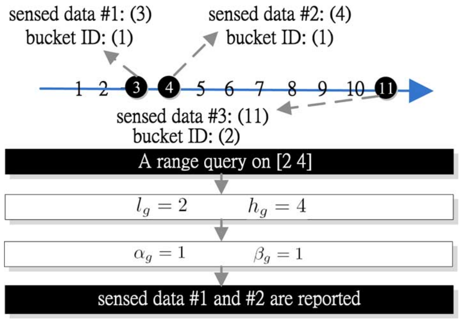


Fig. 2. Illustration of the bucket scheme. In this example, we emphasize a specific attribute A_g.

t is $\langle i, t, 1, \{\{3,4\}\}_{K_{i,t}}, 2, \{\{11\}\}_{K_{i,t}}\rangle$, assuming that $A_g \in [1,20]$, $w = 2$, and each interval length, set at 10, is the same.

Unfortunately, in tiered sensor networks, even when the bucket scheme is used, $s_{\mathcal{M}}$ could still drop some encrypted data and only report part of the results to the authority, resulting in an incomplete query result. In the following, we will describe an extended bucket scheme, which is a combination of the proposed hash tree-based framework and the bucket scheme, to detect the incomplete reply in a communication-efficient manner.

2) Confidentiality-Preserving Data Reporting: In the *hash tree* framework, SMQ-1 detects an incomplete reply by taking advantage of the aggregated hash collaboratively generated by the sensors. The storage node $s_{\mathcal{M}}$ is required to provide the *proof*, which is composed of the aggregated hash and a compact summary of the sensed data, to the authority at the epoch specified in the query so that the authority can use the proof to verify the completeness of the received query results. Since, in our design, all the hash generated by the nodes can be aggregated to yield the final proof, the communication cost can be reduced significantly. The confidentiality-preserving data reporting phase is further divided into an offline stage and an online stage, which will be expounded upon as follows.

Offline Stage: Assume that an aggregation tree [15] has been constructed after sensor deployment. Recall that the domain of attribute A_g is divided into w intervals. The sets $\{p_i^V, p_i^\emptyset \mid V \in \mathcal{V}\}$ of $(w^d+1)N$ prime numbers for all $i \in \{1,\ldots,N-1,\mathcal{M}\}$, called the sets of *bucket primes* of s_i, are all stored in $s_{\mathcal{M}}$. Note that each bucket prime should be selected to be different from each other. Since the association between $s_{\mathcal{M}}$ and ordinary sensors could be achieved after the sensor deployment, the above can be implemented by first storing the bucket primes of all nodes in all storage nodes of the network. After the association is achieved, $s_{\mathcal{M}}$ may keep the bucket primes of the nodes in its cell and erase the bucket primes in the other cells. In addition, a set $\{k_{i,0}^V, k_{i,0}^\emptyset \mid i \in \{1,\ldots,N-1,\mathcal{M}\}, V \in \mathcal{V}\}$ of w^d+1 keys is selected by the authority and is stored in each sensor s_i initially. For fixed i and t, $\{k_{i,t}^V, k_{i,t}^\emptyset \mid V \in \mathcal{V}\}$, $i \in \{1,\ldots,N-1,\mathcal{M}\}$, is called the set of *bucket keys* of s_i at epoch t. Bucket primes could be publicly-known, while bucket keys of s_i can be known only by s_i and the authority.

Online Stage: Each sensor s_i, at the beginning of epoch t, calculates $k_{i,t}^V = \text{hash}(k_{i,t-1}^V)$ and then drops $k_{i,t-1}^V$, $\forall V \in \mathcal{V}$. In addition, s_i also calculates $k_{i,t}^\emptyset = \text{hash}(k_{i,t-1}^\emptyset)$ and then drops $k_{i,t-1}^\emptyset$. Recall that each node s_i on average has Y/N sensed data at epoch t, and assume that the set of Y/N buckets associated with these Y/N sensed data is $B_{i,t} = \{v^{i,t,\sigma} \mid \sigma = 1,\ldots,Y/N\}$. Then, according to its sensed data, s_i calculates

$$H_{i,t} = \text{hash}_{K_{i,t}}\left(\sum_{\sigma=1}^{Y/N} k_{i,t}^{v^{i,t,\sigma}}\right) \tag{1}$$

where $\text{hash}_K(\cdot)$ denotes the keyed cryptographic hash function with key K, if it has sensed data, and $H_{i,t} = \text{hash}_{K_{i,t}}(k_{i,t}^\emptyset)$, otherwise. $H_{i,t}$ can be regarded as a *subproof* demonstrating the existence of the data sensed by s_i. It is assumed that s_i receives $\langle j_\rho, t, \mathcal{E}_{j_\rho,t}, \mathcal{B}_{j_\rho,t}, \mathcal{H}_{j_\rho,t}, \mathcal{P}_{j_\rho,t}\rangle$, $j_\rho \in [1, N-1]$, $\forall \rho \in [1,\chi]$ from its χ children, $s_{j_1},\ldots,s_{j_\chi}$. Afterward, the sensor s_i calculates $\mathcal{E}_{i,t} = (\bigcup_{\rho=1}^{\chi} \mathcal{E}_{j_\rho,t}) \bigcup E_{i,t}$, where $E_{i,t}$ denotes the set of encrypted data sensed by s_i at epoch t and $\mathcal{E}_{i,t}$ denotes the set of aggregated encrypted data generated by the sensors in the subtree rooted at s_i, and $\mathcal{B}_{i,t} = (\bigcup_{\rho=1}^{\chi} \mathcal{B}_{j_\rho,t}) \bigcup B_{i,t}$, where $B_{i,t}$ denotes the set of buckets of $E_{i,t}$ and $\mathcal{B}_{i,t}$ denotes the set of aggregated buckets generated by the sensors in the subtree rooted at s_i. We note that, due to the fact that the different sensor readings could be the same or in the same bucket, $E_{i,t}$, $\mathcal{E}_{i,t}$, $B_{i,t}$, and $\mathcal{B}_{i,t}$ will be multisets in certain cases. In addition, s_i also calculates

$$\mathcal{H}_{i,t} = \text{hash}_{K_{i,t}}\left(\sum_{\rho=1}^{\chi} \mathcal{H}_{j_\rho,t} + H_{i,t}\right) \tag{2}$$

where $\mathcal{H}_{i,t}$ denotes an aggregated hash. Finally, s_i reports $\langle i, t, \mathcal{E}_{i,t}, \mathcal{B}_{i,t}, \mathcal{H}_{i,t}\rangle$ to its parent node on the aggregation tree. Note that the leaf node on the aggregation tree is assumed to receive nothing.

Similarly, at epoch t, $s_{\mathcal{M}}$ computes $k_{\mathcal{M},t}^V = \text{hash}(k_{\mathcal{M},t-1}^V)$ and $k_{\mathcal{M},t}^\emptyset = \text{hash}(k_{\mathcal{M},t-1}^\emptyset)$, drops $k_{\mathcal{M},t-1}^V$ and $k_{\mathcal{M},t-1}^\emptyset$, and calculates $E_{\mathcal{M},t}$, $B_{\mathcal{M},t}$, and $H_{\mathcal{M},t}$ according to its own sensed data. In fact, the procedures $s_{\mathcal{M}}$ needed to perform after messages are received from the child nodes are the same as the ones performed by ordinary sensors. Acting as the root of the aggregation tree, however, $s_{\mathcal{M}}$ keeps the aggregated results, which are denoted as $\mathcal{E}_{\mathcal{M},t}$, $\mathcal{B}_{\mathcal{M},t}$, and $\mathcal{H}_{\mathcal{M},t}$, respectively, in its local storage and waits for the query issued by the authority. Furthermore, $s_{\mathcal{M}}$ calculates and keeps

$$\mathcal{P}_{\mathcal{M},t} = \prod_{V \in \mathcal{B}_{\mathcal{M},t}} p_i^V \cdot \prod_{i \in \{1,\ldots,N-1,\mathcal{M}\} \setminus R_t} p_i^\emptyset \tag{3}$$

where R_t is the set of sensors having sensed data at epoch t. Here, $\mathcal{P}_{\mathcal{M},t}$ can be thought of as a compact summary of the sensed data of the whole network and can be very useful for the authority in checking the completeness of the query result, while $\mathcal{H}_{\mathcal{M},t}$ can be used by the authority to verify the authenticity of $\mathcal{P}_{\mathcal{M},t}$. As a whole, $\mathcal{H}_{\mathcal{M},t}$ and $\mathcal{P}_{\mathcal{M},t}$ constitute a *proof* demonstrating the existence of the data sensed by the sensors.

An example showing how SMQ-1 works is illustrated in Fig. 3, where there are three sensors and one storage node. In this example, we assume only one attribute $(d = 1)$, whose possible values are within $[1, 20]$ and are divided into two intervals of equal length $(w = 2)$. In Fig. 3(a),

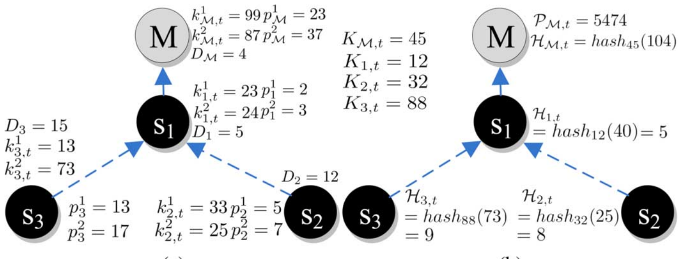

M
$k^1_{\mathcal{M},t} = 99$ $p^1_{\mathcal{M}} = 23$
$k^2_{\mathcal{M},t} = 87$ $p^2_{\mathcal{M}} = 37$
$D_{\mathcal{M}} = 4$
$K_{\mathcal{M},t} = 45$
$K_{1,t} = 12$
$K_{2,t} = 32$
$K_{3,t} = 88$
S1
$k^1_{1,t} = 23$ $p^1_1 = 2$
$k^2_{1,t} = 24$ $p^2_1 = 3$
$D_1 = 5$
$D_3 = 15$
$k^1_{3,t} = 13$
$k^2_{3,t} = 73$
S3
$p^1_3 = 13$
$p^2_3 = 17$
$k^1_{2,t} = 33$ $p^1_2 = 5$
$k^2_{2,t} = 25$ $p^2_2 = 7$
$D_2 = 12$
S2
(a)
M
$\mathcal{P}_{\mathcal{M},t} = 5474$
$\mathcal{H}_{\mathcal{M},t} = hash_{45}(104)$
S1
$\mathcal{H}_{1,t} = hash_{12}(40) = 5$
S3
$\mathcal{H}_{3,t} = hash_{88}(73) = 9$
$\mathcal{H}_{2,t} = hash_{32}(25) = 8$
S2
(b)

Fig. 3. Illustrative example of SMQ-1. (a) The network setting. (b) The forwarding of $\mathcal{H}_{i,t}$.

D_i denotes the data sensed by s_i. In Fig. 3(b), we illustrate the forwarding of $\mathcal{H}_{i,t}$. It is shown in Fig. 3(b) that, once receiving $\mathcal{H}_{2,t}$ and $\mathcal{H}_{3,t}$ from its descendants, s_1 computes $\mathcal{H}_{1,t} = \text{hash}_{K_{1,t}}(23+8+9) = \text{hash}_{12}(40) = 5$. Note that the output of hash function in Fig. 3 is assumed artificially. In reality, this depends on the hash function $hash_K(\cdot)$ used.

3) Query Result Completeness Verification: In the following, we discuss how $s_{\mathcal{M}}$ can answer different queries and how the authority can verify the QRC in SMQ-1.

Answering Range Query: Assume that a range query $\langle \mathcal{C}, t, l_1, h_1, l_2, h_2, \ldots, l_d, h_d \rangle$ is issued by the authority. The encrypted data falling into the buckets in the set $\mathcal{A}$ (the set of buckets which the encrypted data belong to, mentioned in Section III-A1), along with the proof composed of $\mathcal{H}_{\mathcal{M},t}$ and $\mathcal{P}_{\mathcal{M},t}$, are sent to the authority.

Once $\mathcal{P}_{\mathcal{M},t}$ is received, the authority immediately performs the prime factor decomposition of $\mathcal{P}_{\mathcal{M},t}$. Due to the construction of $\{p_i^V, p_{\mathcal{M}}^V, p_i^{\emptyset}, p_{\mathcal{M}}^{\emptyset} \mid i \in [1, N-1], V \in \mathcal{V}\}$, which guarantees that the bucket primes are all distinct, after the prime factor decomposition of $\mathcal{P}_{\mathcal{M},t}$, the authority can be aware of which node contributes which data within specified buckets. As a result, the authority can know which keys should be used to verify the authenticity and integrity of $\mathcal{H}_{\mathcal{M},t}$.

More specifically, we assume that $\mathcal{P}_{\mathcal{M},t} = (\mathfrak{p}_1)^{a_1} \cdots (\mathfrak{p}_\gamma)^{a_\gamma}, a_1, \ldots, a_\gamma > 0$, $\gamma \geq 0$, and that $\mathfrak{p}_1, \ldots, \mathfrak{p}_\gamma$ are distinct prime numbers. From the construction of $\mathcal{P}_{\mathcal{M},t}$, we know that $(\mathfrak{p}_{\hat{k}})^{a_{\hat{k}}}$, for $\hat{k} \in [1, \gamma]$, is equal to $(p_{k'}^{k''})^{a_{\hat{k}}}$, for $k' \in [1, N-1]$ and $k'' \in \mathcal{V}$. From the procedure performed by each node, it can also be known that the appearance of $(\mathfrak{p}_{\hat{k}})^{a_{\hat{k}}} = (p_{k'}^{k''})^{a_{\hat{k}}}$ in $\mathcal{P}_{\mathcal{M},t}$ means that at epoch t the sensor $s_{k'}$ produces $a_{\hat{k}}$ data falling into bucket $k'' \in \mathcal{V}$, contributing the bucket key $k_{k',t}^{k''}$ in total $a_{\hat{k}}$ times in $\mathcal{H}_{\mathcal{M},t}$. Here, the sensor $s_{k'}$ producing the data falling into the bucket $\emptyset$ means that $s_{k'}$ senses nothing. Thus, we can infer the total amount of data falling into specified buckets at epoch t. Recall that the authority is aware of the topology of the aggregation tree. After the prime factor decomposition of $\mathcal{P}_{\mathcal{M},t}$, the authority can reconstruct $\mathcal{H}_{\mathcal{M},t}$ according to the derived $a_{\hat{k}}$ and $\mathfrak{p}_{\hat{k}}$ by its own effort, because it knows $K_{i,t}$ and $k_{i,t}^V$, $\forall i \in \{1, \ldots, N-1, \mathcal{M}\}$, $\forall t \geq 0$, $\forall V \in \mathcal{V}$. Therefore, we know that $\mathcal{H}_{\mathcal{M},t}$ reconstructed by the authority is equal to the received $\mathcal{H}_{\mathcal{M},t}$ if and only if the received $\mathcal{P}_{\mathcal{M},t}$ are considered authentic. When the verification of $\mathcal{P}_{\mathcal{M},t}$ fails, $\mathcal{M}$ is considered compromised. When the verification of $\mathcal{P}_{\mathcal{M},t}$ is successful, the authority decrypts all the received encrypted data, and checks whether the number of query results falling into the buckets in $\mathcal{A}$ matches those indicated by $\mathcal{P}_{\mathcal{M},t}$. If and only if there are matches in all the buckets in $\mathcal{A}$, the received query results are considered complete.

Answering Top-k Query: Based on the proof, $\mathcal{H}_{\mathcal{M},t}$ and $\mathcal{P}_{\mathcal{M},t}$, generated in SMQ-1, the authority can know which buckets contain data, which allows it to also utilize such information to examine the completeness of query results of top-k query. In other words, top-k query can also be secured by means of SMQ-1. Because of the similarity between answering the range query and answering the top-k query, some details will be omitted in the following description.

For each bucket $(v_1, \ldots, v_d)$, $v_g \in [1, w]$, $g \in [1, d]$, we define a d-dimensional tuple, $(v'_1, \ldots, v'_d)$, where v'_i is the central value of the v_ith interval of the ith attribute. As $(v'_1, \ldots, v'_d)$ acts as the representative data of the bucket $(v_1, \ldots, v_d)$, the value $R((v'_1, \ldots, v'_d))$ can be used to be the representative ranking value of all the data in the bucket $(v_1, \ldots, v_d)$. Recall that we simply assume that the data with the first k smallest ranking values are desired. Assume that $\zeta_1, \ldots, \zeta_k \in \mathcal{V}$ are k buckets whose representative ranking values are among the first k smallest ones. $\zeta_1, \ldots, \zeta_k$ can be known by $s_{\mathcal{M}}$ according to the proper computation on $\mathcal{B}_{\mathcal{M},t}$. To answer a top-$k$ query, $\langle \mathcal{C}, t, k \rangle$, issued by the authority, the storage node $s_{\mathcal{M}}$ reports the buckets, $\zeta_1, \ldots, \zeta_k$, and their corresponding encrypted data, along with $\mathcal{H}_{\mathcal{M},t}$ and $\mathcal{P}_{\mathcal{M},t}$, to the authority because it can be known that the data with the first k smallest ranking values must be within $\zeta_1, \ldots, \zeta_k$. After receiving the query result, the authority can first verify the authenticity of $\mathcal{P}_{i,t}$ by using $\mathcal{H}_{i,t}$, and verify the QRC by using $\mathcal{P}_{i,t}$. Note that both of the above verifications can be performed in a way similar to the one described in verifying range query. In fact, after receiving $\mathcal{P}_{i,t}$, the authority knows which buckets contain data and the amount of data. Hence, knowing $\mathcal{B}_{\mathcal{M},t}$ and the representative ranking values of the buckets in $\mathcal{B}_{\mathcal{M},t}$, the authority can also obtain $\zeta_1, \ldots, \zeta_k$. Afterwards, what the authority should do is to check if it receives the buckets, $\zeta_1, \ldots, \zeta_k$, and if the number of data in each received bucket is consistent with the number indicated by $\mathcal{P}_{i,t}$. If and only if these two verifications pass, the authority considers the received query result to be complete and extracts the top-k result from the encrypted data sent from $s_{\mathcal{M}}$.

Answering Skyline Query: Assume that a skyline query $\langle \mathcal{C}, t \rangle$ is issued by the authority. Each bucket $(v_1, \ldots, v_d)$ can be thought of as a data point if the data point central in the bucket is considered as the representative of the bucket. With this notion, we define the *skyline bucket data set* $\mathcal{K}_t$ at epoch t as a subset of $\mathcal{B}_{\mathcal{M},t}$, in which each bucket is not dominated by the other buckets. $s_{\mathcal{M}}$ can construct $\mathcal{K}_t$ for each t easily because of its knowledge of $\mathcal{B}_{\mathcal{M},t}$. Recall that the domination here has been defined in the query model of Section II. Therefore, when receiving the skyline query $\langle \mathcal{C}, t \rangle$ issued by the authority, $s_{\mathcal{M}}$ can report $\mathcal{K}_t$, together with the proof and the encrypted data falling into $\mathcal{K}_t$, to the authority. The authority verifies the correctness of the proof as in range query and top-k query cases. With $\mathcal{P}_{\mathcal{M},t}$, the authority can also calculate $\mathcal{K}_t$. Finally, the authority can check if it receives the complete skyline query result.

Storage Overhead Reduction: The storage overhead due to the bucket primes and bucket keys can be reduced without compromising security at the expense of a slightly increased computation overhead. The details are as follows.

Assume that the bucket primes associated with a specific sensor s_i correspond to all the prime numbers in a fixed interval $[L_i, U_i]$, $L_i \leq U_i$. Note that, since the bucket primes of s_i is independent of the cell where it is located, L_i and U_i can be determined by the authority in advance. Here, we assume that there is a natural ordering $\mathbb{N}$ of all possible buckets, and a mapping between the prime numbers in $[L_i, U_i]$ and all possible buckets. For example, in the case of $d = w = 2$, $\mathbb{N}((1,1)) \leq \mathbb{N}((1,2)) \leq \mathbb{N}((2,1)) \leq \mathbb{N}((2,2))$. Assume also that *Bloom filters* [12], which are memory-efficient probabilistic data structures with the purpose of membership query, are available. Note that the design and optimization of the Bloom filter is beyond the scope of this paper. Now, all the prime numbers in $[L_i, U_i]$ are embedded into the Bloom filter associated with s_i. The bucket primes $\{p_i^V, p_i^\emptyset \,|\, V \in \mathcal{V}\}$ of s_i can be replaced with $p_i^\emptyset$, L_i, U_i, and a single Bloom filter to reduce space required for storage. To retrieve the bucket prime p_i^V, s_i can search for $\mathbb{N}(V)$th element in the Bloom filter of s_i and set it as p_i^V.

On the other hand, all the bucket keys $\{k_{i,0}^V, k_{i,0}^\emptyset \,|\, V \in \mathcal{V}\}$ of s_i can be replaced with a single *key-generation key*, K_{kg}. Specifically, recall that there is a natural ordering $\mathbb{N}$ on buckets. If only K_{kg} is stored, the bucket key $k_{i,t}^V$ can be generated by computing $k_{i,t}^V = \text{hash}_{K_{\text{kg}}}^{\mathbb{N}(V)}(K_{i,t})$, where $\text{hash}_{K_{\text{kg}}}^{\mathbb{N}(V)}(K_{i,t})$ means that the keyed cryptographic hash function $\text{hash}_{K_{\text{kg}}}^{\mathbb{N}(V)}(\cdot)$ is applied on $K_{i,t}$ $\mathbb{N}(V)$ times. Apparently, after this change, the security of SMQ-1 will not be downgraded because there is no difference between the scheme with multiple keys $\{k_{i,0}^V, k_{i,0}^\emptyset \,|\, V \in \mathcal{V}\}$ stored in a sensor and that a with single key K_{kg} stored in a sensor.

With the above storage overhead reduction method in mind, in the following, we still use the same notations, $\{p_i^V, p_i^\emptyset \,|\, V \in \mathcal{V}\}$ and $\{k_{i,0}^V, k_{i,0}^\emptyset \,|\, V \in \mathcal{V}\}$, to describe the bucket primes and keys for convenience in explaining our method. Nevertheless, from the implementation perspective, one can choose to plainly store all the bucket primes and bucket keys if such a storage requirement is affordable, or choose to use the above storage overhead reduction method otherwise.

B. SMQ-2

In the previous discussions, the impact of the compromise of ordinary sensors is excluded. In practice, the adversary could take the control of sensors to enhance the ability of performing malicious operations. Let $\mathfrak{s} \subseteq \{s_1, \ldots, s_{N-1}, \mathcal{M}\}$ be the set of compromised nodes. Here, we investigate the security impact of *collusion attack* and *false-incrimination attack*, and provide our countermeasure.

In a *collusion attack*, $s_\mathcal{M}$ and a subset of sensors are compromised. The compromised $s_\mathcal{M}$ in $\mathfrak{s}$ colludes with the other sensors in $\mathfrak{s}$ in the hope that more portions of the query results generated by innocent sensors can be dropped. Due to their nature, where the evidence of sensed data is embedded into the sensed data of some other nodes, all the broadcast-based crosscheck approaches [28], [32], [33] suffer from collusion attack. A naive method, called *random probing*, by which the authority occasionally checks if those nodes that report nothing really sense nothing by directly communicating with some of them, is proposed in [33]. Random probing, however, can discover the incomplete query results only by chance and cannot identify even one node in $\mathfrak{s}$.

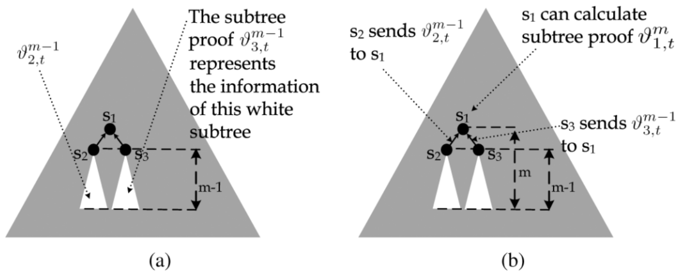
On the other hand, in this paper, we identify a new denial-of-service attack never addressed in the literature, called *false-incrimination attack*. In this attack, only a subset of sensors are compromised; i.e., $s_\mathcal{M} \notin \mathfrak{s}$. The nodes in $\mathfrak{s}$ intentionally provide the wrong subproofs to frame the innocent $s_\mathcal{M}$ as the compromised one. Unfortunately, all of the prior works [21], [28], [32], [33] suffer from this attack that is easily launched. For example, one node in $\mathfrak{s}$ can send the wrong encoding numbers to $s_\mathcal{M}$ and can broadcast the wrong sensing information in the encoding approach [21] and in the crosscheck approaches [28], [32], [33], respectively. All of these can frame $s_\mathcal{M}$ successfully.

Here, we note that false-incrimination attack can be thought of as a simplified version of Sybil attack [17]. Compared to Sybil attack, which can attack different network functionalities such as distributed storage, routing, and voting in different network settings, false incrimination attack is more specific to the scenario considered in our paper. Nonetheless, the adversary has to consume more resources (e.g., the efforts to fabricate legitimate IDs or compromise sensors) to launch a Sybil attack, but can consume much less resources to launch a false-incrimination attack.

As to the detection capability, recall that, with the unrealistic assumption made in Section III-A that only $s_\mathcal{M}$ will be compromised, we can ensure that $s_\mathcal{M}$ has been compromised if the proof fails to be verified in SMQ-1. Nevertheless, in view of the fact that both $s_\mathcal{M}$ and the sensors could be compromised, via our proposed method (described below), we can know that at least one node in $\mathfrak{s}$ exists in the set of nodes identified by SMQ-2. Afterwards, we can check the legitimacy of only a few nodes, identify at least one node in $\mathfrak{s}$, and then sweep [23], [25] the identified nodes in $\mathfrak{s}$.

1) Subtree Sampling in SMQ-2: SMQ-1 could be vulnerable to the false-incrimination attack since false subproofs injected by $\mathfrak{s}$ will be integrated with the other correct subproofs to construct a false proof. Here, we present a novel technique, called *subtree sampling*, enabling SMQ-1, with a slight modification, to efficiently but significantly mitigate the threat of false-incrimination attacks. In what follows, SMQ-1 with subtree sampling is called SMQ-2.

The idea of *subtree sampling* is to check if the proof constructed by the nodes in a random subtree with fixed height is authentic. After sampling, each node will be labeled differently. Finally, the authority can perform the inspection like software attestation [23], [25] only on a set of nodes with a specific label. With the guarantee that at least one compromised node exists in the set of nodes to be inspected, the authority can identify and remove the compromised nodes. SMQ-2 also consists of the confidentiality-preserving data reporting and QRC verification phases, and the former also has offline and online stages. Due to the similarity of SMQ-1 and SMQ-2, we emphasize only the different parts in the confidentiality-preserving data reporting phase.



$\vartheta_{2,t}^{m-1}$
The subtree proof $\vartheta_{3,t}^{m-1}$ represents the information of this white subtree
s1
s2
s3
m-1
(a)
s2 sends $\vartheta_{2,t}^{m-1}$ to s1
s1 can calculate subtree proof $\vartheta_{1,t}^{m}$
s3 sends $\vartheta_{3,t}^{m-1}$ to s1
m
(b)

Fig. 4. Conceptual diagrams of the use of $\vartheta_{i,t}^m$. In this illustration, we only emphasize on $\vartheta_{i,t}^{m-1}$ and $\vartheta_{i,t}^m$ without considering $\vartheta_{i,t}^0 \ldots \vartheta_{i,t}^{m-2}$. (a) s_2 and s_3 know $\vartheta_{2,t}^{m-1}$ and $\vartheta_{3,t}^{m-1}$, respectively. (b) After receiving messages from the children, s_1 can obtain $\vartheta_{1,t}^m$.

Offline Stage: Like SMQ-1, a set $\{k_{i,0}^V, k_{i,0}^{\emptyset} \,|\, i \in \{1,\ldots,N-1,\mathcal{M}\}, V \in \mathcal{V}\}$ of bucket keys and a set $\{p_i^V, p_i^{\emptyset} \,|\, i \in \{1,\ldots,N-1,\mathcal{M}\}, V \in \mathcal{V}\}$ of bucket primes are selected by the authority. Nonetheless, in SMQ-2, the set $\{k_{i,0}^V, k_{i,0}^{\emptyset} \,|\, V \in \mathcal{V}\}$ of bucket keys of s_i and the set $\{p_i^V, p_i^{\emptyset} \,|\, V \in \mathcal{V}\}$ of bucket primes of s_i are assigned to the node s_i.

Online Stage: Let m be a user-selected constant indicating the subtree height. Let $\vartheta_{i,t}^m$ generated by s_i at epoch t be the *subtree proof* of the data sensed by the nodes on a subtree of the underlying aggregation tree rooted at s_i with height m. In the SMQ-2 scheme, each s_i is assumed to receive $\langle j_\rho, t, \mathcal{E}_{j_\rho,t}, \mathcal{B}_{j_\rho,t}, \mathcal{H}_{j_\rho,t}, \mathcal{P}_{j_\rho,t}, \vartheta_{j_\rho,t}^0, \ldots, \vartheta_{j_\rho,t}^{m-1}\rangle$, $j_\rho \in [1, N-1]$, $\forall \rho \in [1,\chi]$, from its χ children, $s_{j_1},\ldots,s_{j_\chi}$. Each s_i calculates $\mathcal{E}_{i,t}$, $\mathcal{B}_{i,t}$, and $\mathcal{H}_{i,t}$ as in the SMQ-1 scheme. In addition, s_i calculates

$$\mathcal{P}_{i,t} = \prod_{\rho=1}^{\chi} \mathcal{P}_{j_\rho,t} \cdot P_{i,t} \tag{4}$$

where

$$P_{i,t} = \prod_{\sigma=1}^{Y/N} p_i^{v^{i,t,\sigma}} \tag{5}$$

if it has sensed data, and $P_{i,t} = p_i^{\emptyset}$ otherwise. Note that the leaf node on the aggregation tree is assumed to receive nothing. Furthermore, s_i performs the following operations. Assume that $\vartheta_{j_\rho,t}^v = \{\bar{H}_{j_\rho,t}^v, \bar{P}_{j_\rho,t}^v\}$ are available, where $v \in [0, m-1]$, $j_\rho \in [1, N-1]$, and $\rho \in [1,\chi]$. The sensor s_i calculates $\bar{H}_{i,t}^{v+1} = \text{hash}_{K_{i,t}}(\sum_{\rho=1}^{\chi} \bar{H}_{j_\rho,t}^v + H_{i,t})$ and $\bar{P}_{i,t}^{v+1} = \prod_{\rho=1}^{\chi} \bar{P}_{j_\rho,t}^v \cdot P_{i,t}$. The sensor s_i also calculates $\bar{H}_{i,t}^0 = H_{i,t}$ and $\bar{P}_{i,t}^0 = P_{i,t}$, where $H_{i,t}$ and $P_{i,t}$ are computed in the method stated above. Then, $\bar{H}_{i,t}^v$ and $\bar{P}_{i,t}^v$ are assigned to the set $\vartheta_{i,t}^v$. The sensor s_i keeps $\vartheta_{i,t}^m$ in its local storage if $\text{hash}_{K_{i,t}}(i) \le \xi$, where ξ is a predetermined threshold known by each node that will be analyzed later, and drops $\vartheta_{i,t}^m$ otherwise. At last, s_i sends $\langle i, t, \mathcal{E}_{i,t}, \mathcal{B}_{i,t}, \mathcal{H}_{i,t}, \mathcal{P}_{i,t}, \vartheta_{i,t}^0, \ldots, \vartheta_{i,t}^{m-1}\rangle$ to its parent node. $s_{\mathcal{M}}$ performs the same operations as s_i except that it keeps $\mathcal{E}_{\mathcal{M},t}$, $\mathcal{B}_{\mathcal{M},t}$, $\mathcal{H}_{\mathcal{M},t}$, and $\mathcal{P}_{\mathcal{M},t}$ in its local storage. The conceptual diagrams of the above procedures are depicted in Fig. 4.

Let $T_{i,t}^m$ be a subtree with height m, rooted at s_i at epoch t. Note that, when s_i and m are fixed, the corresponding subtree is invariant with t. Nonetheless, to ease the explanation of our methods, the superscript t remains. The generation of $\vartheta_{i,t}^m$ depends only on the nodes in $T_{i,t}^m$. In other words, the compromised nodes outside the $T_{i,t}^m$ cannot affect $\vartheta_{i,t}^m$. Let W_t be the *witness set* of sensors satisfying $\text{hash}_{K_{i,t}}(i) \le \xi$ at epoch t. The nodes in W_t are called *witness nodes* at epoch t. With such a definition of W_t, $T_{i,t}^m$ can also be regarded as a subtree of the underlying aggregation tree, whose nodes contribute to the construction of $\vartheta_{i,t}^m$, $s_i \in W_t$. Assume that the query result is found to be incomplete at epoch t. The authority requests $\vartheta_{i,t}^m$ for some t's for the corresponding witness nodes in W_t's via authenticated broadcast [13]. The method whereby the nodes securely forward their subtree proofs to the authority is described in Section III-B2. Without loss of generality, we assume in the subsequent discussion that the authority requests $\vartheta_{i,t}^m$ for $t \in [t_1, t_2]$ and $s_i \in W_t$.

Counteracting False-Incrimination Attack: To identify the nodes in $\mathfrak{s}$, all the nodes at first are considered (labelled) *neutral* at epoch t by the authority. The nodes in $T_{i,t}^m$ become *innocent* at epoch t if the verification of $\vartheta_{i,t}^m$ passes, and the nodes in $T_{i,t}^m$ become *suspicious* at epoch t otherwise. The above verification goes as follows.

According to the $\bar{P}_{i,t}^m$ in the subtree proof $\vartheta_{i,t}^m$, the authority knows which nodes in the subtree $T_{i,t}^m$ contribute data and their amount. The authority can, based on this information, calculate $\bar{H}_{i,t}^m$, $\forall i$ and $\forall t \in [t_1, t_2]$ by itself. As a consequence, the verification passes if and only if the $\bar{H}_{i,t}$ calculated by the authority itself is equal to the $\bar{H}_{i,t}$ extracted from the received $\vartheta_{i,t}^m$. As a whole, each time the above checking procedures are performed according to $\vartheta_{i,t}^m$ for each $s_i \in W_t$ at epoch t, the nodes in $T_{i,t}^m$ will be assigned to one of three sets, which are $\mathfrak{S}_t^i$, $\mathfrak{I}_t^i$, and $\mathfrak{N}_t^i$, containing suspicious nodes, innocent nodes, and neutral nodes, respectively. Note that, for a fixed epoch t, those nodes that do not belong to $\bigcup_{s_i \in W_t} T_{i,t}^m$ are neutral, and, at this point, $\mathfrak{S}_t^i$ and $\mathfrak{I}_t^i$ could overlap. We sequentially define the innocent set $\mathfrak{I}_t = \bigcup_{s_i \in W_t} \mathfrak{I}_t^i$, suspicious set $\mathfrak{S}_t = (\bigcup_{s_i \in W_t} \mathfrak{S}_t^i) \setminus \mathfrak{I}_t$, and neutral set $\mathfrak{N}_t = \{s_1,\ldots,s_{N-1}\} \setminus (\mathfrak{S}_t \bigcup \mathfrak{I}_t)$ at epoch t. Define $\Upsilon_{[t_1,t_2]}$ and $\bar{\Upsilon}_{[t_1,t_2]} \subseteq \Upsilon_{[t_1,t_2]}$ as the set of nodes covered by sampled subtrees at the epochs in $[t_1,t_2]$ and the set of nodes to be inspected, respectively. Note that we only need to check the legitimacy of the nodes in $\bar{\Upsilon}_{[t_1,t_2]}$ to find at least one node in $\mathfrak{s}$. Finally, assuming that at epochs within $[t_1,t_2]$, $\mathfrak{s}$ always sends the wrong subtree proofs, no compromised node acts as a witness node, and $|\mathfrak{s}| = 1$, we can have a *weak filtering criteria*

$$\bar{\Upsilon}_{[t_1,t_2]} = \bigcap_{t \in [t_1,t_2], \mathfrak{S}_t \neq \emptyset} \mathfrak{S}_t \tag{6}$$

that contains the compromised node in $\mathfrak{s}$ injecting the false subproof if at least one $\mathfrak{S}_t^i$, $t \in [t_1,t_2]$, is nonempty, and

$$\bar{\Upsilon}_{[t_1,t_2]} = \bigcap_{t \in [t_1,t_2]} \mathfrak{N}_t \tag{7}$$

contains the compromised node in $\mathfrak{s}$ injecting the false subproof otherwise (Theorem 1).

The intuition behind the checking procedure is illustrated in Fig. 5. It can be observed that, as $(t_2 - t_1)$ grows, the size of

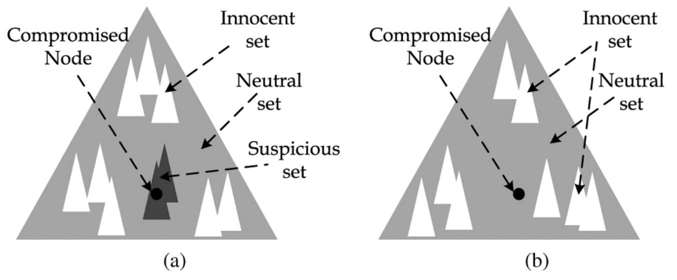


Fig. 5. Conceptual diagrams of identifying the compromised nodes. (a) Only the nodes in dark-gray area need to be attested. (b) Only the nodes in light-gray area need to be attested.

the set of the nodes the authority needs to perform the inspection will be drastically shrunk so that the computation and communication cost required in the inspection will be substantially reduced accordingly.

Unfortunately, there are two cases where the capability of the weak filtering criteria would be reduced. First, when $W_t \cap \mathfrak{s} \neq \emptyset$, the compromised sensor happening to be the witness node sends the legitimate subtree proof and will be considered innocent. Second, while $|\mathfrak{s}| > 1$ and different nodes in $\mathfrak{s}$ are covered by different $T_{i,t}^m$'s at different epochs, the intersection rule in (6) could fail because it is unlikely that some nodes in $\mathfrak{s}$ are included in $\Upsilon_{[t_1,t_2]}$ for all $t \in [t_1, t_2]$. Nevertheless, these threats can be mitigated greatly by slightly modifying the node categorization rule as follows.

The definitions of innocent, neutral, and suspicious sets are the same as the ones described before, but this time we arrive at a *strong filtering criteria*

$$\bar{\Upsilon}_{[t_1,t_2]} = \bigcup_{t\in[t_1,t_2],\mathfrak{S}_t\neq\emptyset} \mathfrak{S}_t \tag{8}$$

that contains at least one compromised node injecting the false subproof if at least one $\mathfrak{S}_t^i$, $t \in [t_1, t_2]$, is nonempty, and $\bar{\Upsilon}_{[t_1,t_2]} = \bigcap_{t\in[t_1,t_2]} \mathfrak{N}_t$ contains at least one compromised node injecting the false subproof otherwise (Theorem 2). Here, compared with (6), (8) trades the accuracy of identifying the compromised nodes for the detection capability. In other words, although the authority probably needs to check the legitimacy of more nodes, the strong filtering criteria guarantees that at least one node in $\mathfrak{s}$ can be found in the presence of multiple compromised nodes that could act as witness nodes.

Discussion: In summary, though we do not have a specific design on SMQ-2, our proposed SMQ-2 scheme, which is extended from SMQ-1, has been perfectly resilient against collusion attack. This can be proven easily as follows. Recall that the objective of collusion attack is to remove a fraction of the query result (sensed data), and each sensor contributes to the construction of $\mathcal{H}_{\mathcal{M},t}$ in both SMQ-1 and SMQ-2 if it has sensed data. To remove the sensed data of a specified sensor s_i, the adversary needs to construct $\mathcal{H}'_{\mathcal{M},t}$ corresponding to the query result without the sensed data generated by s_i, which may involve the bucket key $k_{i,t}^{\emptyset}$ and the bucket keys of the other sensors contributing the construction of $\mathcal{H}_{\mathcal{M},t}$. As a result, since the adversary does not possess these keys, there is only negligible probability that the adversary successfully launches a collusion attack without being detected.

On the other hand, as for the resilience of SMQ-2 against false-incrimination attack, we have the following two theorems.

Theorem 1: The weak filtering criteria guarantees that at least one node in $\mathfrak{s}$ can be identified if the following three conditions: 1) $\mathfrak{s}$ always sends false subtree proofs, 2) the nodes in $\mathfrak{s}$ do not happen to be the witness node, and 3) $|\mathfrak{s}| = 1$, hold.

Proof: Assume that $\mathfrak{S}_t \neq \emptyset$ for some t. By the definition of $\mathfrak{S}_t$, $\mathfrak{S}_t \bigcap \mathfrak{s} \neq \emptyset$ for some t. Since $|\mathfrak{s}| = 1$ (Condition 3), it follows that $\mathfrak{s} \subseteq \bigcap_{t\in[t_1,t_2],\mathfrak{S}_t\neq\emptyset} \mathfrak{S}_t$ must hold.

Assume that $\mathfrak{S}_t = \emptyset$ for all t. Assume that $s \subseteq \mathfrak{I}_t$ for some t. This means that, at epoch t, $\mathfrak{s}$ either works as the witness node reporting the correct subtree proof or selectively sends the correct subtree proof, contradicting our assumptions that the node in $\mathfrak{s}$ will not be the witness node (Condition 2) and always sends the false subtree proof (Condition 1). Thus, with the definition of $\mathfrak{I}_t$, we can know that $\mathfrak{s} \nsubseteq \mathfrak{I}_t$. It follows that $\mathfrak{s} \subseteq \{s_1 \ldots, s_{N-1}, \mathcal{M}\} \setminus \mathfrak{I}_t$ for all t. As $|\mathfrak{s}| = 1$ (Condition 3), the case that $\mathfrak{s} \subseteq \bigcap_{t\in[t_1,t_2]}(\{s_1 \ldots, s_{N-1}, \mathcal{M}\} \setminus \mathfrak{I}_t) = \bigcap_{t\in[t_1,t_2]} \mathfrak{N}_t$ must hold. ■

Theorem 2: After the strong filtering criteria is applied, with probability $1-(1-(m|\mathfrak{s}|)/(N))^z$, where $z = (\xi)/(2^{\ell_h})N(t_2 - t_1)$ denotes the total number of witness nodes at epochs $t \in [t_1, t_2]$, at least one node in $\mathfrak{s}$ can be identified if $\mathfrak{s}$ always sends false subtree proofs.

Proof: Define B_1 as the event in which at least one node in $\mathfrak{s}$ is identified. Define B_2 as the event where $\mathfrak{S}_t \neq \emptyset$ for some t. With such definitions, the probability $\Pr[B_1]$ that at least one node in $\mathfrak{s}$ is identified can be formulated as $\Pr[B_1] = \Pr[B_1 \mid B_2]\Pr[B_2] + \Pr[B_1 \mid \bar{B}_2]\Pr[\bar{B}_2] \geq \Pr[B_1 \mid B_2]\Pr[B_2]$. Obviously, according to the definition of $\mathfrak{S}_t$, $\Pr[B_1 \mid B_2] = 1$. Define $A_{\mathfrak{s}}$ as the set of nodes that are the ascendant nodes of the nodes in $\mathfrak{s}$ and are within m hops away. One can know that $|A_{\mathfrak{s}}| \approx m|\mathfrak{s}|$. Since there are z witness nodes in total at epochs $t \in [t_1, t_2]$, the probability that these z sampled subtrees do not cover the nodes in $\mathfrak{s}$ is $(1 - (|A_{\mathfrak{s}}|)/(N))^z$. Therefore, the probability that at least one sampled subtree covers at least one node in $\mathfrak{s}$ is $1 - (1 - (|A_{\mathfrak{s}}|)/(N))^z \approx 1 - (1 - (m|\mathfrak{s}|)/(N))^z$, which in essence is equivalent to $\Pr[B_2]$. As a whole, $\Pr[B_1] \geq \Pr[B_1 \mid B_2]\Pr[B_2] = 1 \cdot (1 - (1 - (|A_{\mathfrak{s}}|)/(N))^z) \approx 1 - (1 - (m|\mathfrak{s}|)/(N))^z$. ■

Furthermore, there would still be a case where the subtree sampling result obtained under both the weak and strong filtering criteria may be affected. In particular, as $\mathfrak{s}$ only reports the wrong subtree proofs at certain epochs, some nodes in $\mathfrak{s}$ will be considered innocent at those epochs and escape detection. Note that selectively sending the wrong subtree proofs also downgrades the false-incrimination attack's security impact because the authority may not issue the query at the epochs at which the wrong subtree proof is sent, and, thus, the innocent $s_{\mathcal{M}}$ will not be framed. Through the simulation, we demonstrate in Section IV that both weak and strong filtering criteria are resilient to and well-behaved in such cases.

2) Cascade Traceback: In SMQ-2, since witness nodes are required to send their subtree proofs to the authority, the set $\mathfrak{s}$ of compromised nodes, lying on the path between the authority and witness nodes, can manipulate the subtree proofs sent from the witness nodes so that the compromised nodes can escape from being detected and the innocent $s_{\mathcal{M}}$ can still be framed. We, motivated by the IP traceback techniques used in internet security literature [27], develop a new traceback scheme especially suitable for WSNs to resist against such malicious manipulation.

Particularly, we need a mechanism, by which the receiver not only can know whether the received message is manipulated by the intermediate nodes, but also can identify the last node altering the message if that node does exist. Although IP traceback techniques in the internet security literature [27] seem to be a solution to this issue, their usual assumption that the intermediate nodes are innocent in most cases makes them vulnerable to the compromised nodes in WSNs. In view of this, we develop a *cascade traceback* scheme. The idea is simple; each node constructs a cryptographic hash according to the whole message received from the descendant nodes and the self-generated data. The hash is appended to the message to be sent. Afterward, the final messages received by the authority will be associated with several hashes. Using these hashes, the authority can verify the integrity of the received message.

To be more specific, in the cascade traceback scheme, each node s_i is assumed to receive χ messages from χ children nodes. Afterwards, s_i sends $M_{(i)} \,\|\, h_{K_{i,t}}(M_{(i)})$, where $M_{(i)}$ denotes the concatenation of the messages from χ children nodes and the data generated by s_i itself, to the parent node. Note that the leaf node on the aggregation tree directly sends the data it generates. With this procedure, the packet the authority receives is of the form $M_{(\mathcal{M})} \,\|\, h_{K_{i,t}}(M_{(\mathcal{M})})$. To verify the integrity of the data generated by the nodes, the authority checks whether the $h_{K_{i,t}}(M_{(\mathcal{M})})$ calculated by itself according to the received $M_{(\mathcal{M})}$ is equal to the received $h_{K_{i,t}}(M_{(\mathcal{M})})$. With the assumption that $s_{\mathcal{M}}$ has children sensors $s_{j_1}, s_{j_2}, \ldots,$ and s_{j_χ}, $j_\rho \in [1, N-1]$, $\rho \in [1, \chi]$, the authority proceeds with the verification of $M_{(\mathcal{M})}$ that is of the form $M_{(j_1)} \| h_{K_{i,t}}(M_{(j_1)}) \| \cdots \| M_{(j_\chi)} \| h_{K_{i,t}}(M_{(j_\chi)})$ if the verification of $s_{\mathcal{M}}$ is successful and is aware of the message manipulation otherwise. Note that the verification of $M_{(\mathcal{M})}$ means that the verification is conducted for each part $M_{(j_\rho)} \,\|\, h_{K_{i,t}}(M_{(j_\rho)})$, $\rho \in [1, \chi]$, in $M_{(\mathcal{M})}$. Such verification is conducted for each node until we examine all nodes in the cell. If and only if all of the verifications pass, the messages from the nodes are considered intact.

When the cascade traceback scheme is used to forward the subtree proofs in a specific time interval $[t_1, t_2]$, s_i prepares the corresponding subtree proofs as the data generated by itself if $s_i \in W_t$, for some $t \in [t_1, t_2]$, and s_i prepares nothing otherwise. After receiving the notification from the authority, the subtree proofs, along with a number of MACs, will be forwarded to the authority.

The security of the cascade traceback scheme is two-fold. First, cascade traceback guarantees that the adversary cannot manipulate the messages generated by nodes. Second, cascade traceback guarantees that if the intermediate nodes manipulate the transmitted message, the last node performing the manipulation can be identified by the authority.

To show that the first security goal of cascade traceback can be achieved, we assume that the sensor s_i has χ children sensors, $s_{j_1}, s_{j_2}, \ldots,$ and s_{j_χ}, $j_\rho \in [1, N-\lambda]$, $\rho \in [1, \chi]$. Suppose that the adversary does not possess the key $K_{i,t}$. In this case, the adversary cannot generate a valid hash $h_{K_{i,t}}(M_i')$ corresponding to the manipulated message M_i'. Suppose that the adversary possesses $K_{i,t}$. In this case, $h_{K_{i,t}}(M_i')$ can be generated and the verification on s_i passes. Nonetheless, as M_i' is of the form $M_{(j_1)} \| h_{K_{i,t}}(M_{(j_1)}) \| \cdots \| M_{(j_\chi)} \| h_{K_{i,t}}(M_{(j_\chi)})$, if the adversary does not possess all the corresponding keys, then at least the verification on one of nodes $s_{j_1}, s_{j_2}, \ldots,$ and s_{j_χ} will fail.

The second security goal can be restated formally as follows. As long as the transmitted messages are manipulated by an intermediate node x, with the cascade traceback scheme, the authority can identify a set $N(x)$ of nodes, where $N(x)$ is defined as a set of nodes including x and its one-hop neighbors. In other words, the authority can ensure that one of nodes in $N(x)$ manipulates the forwarded message. Note that, if multiple nodes alter the message, the last node altering the message will be x. This property can be proven as follows. Suppose that the verification stalls on $s_{\mathcal{M}}$. Obviously, this means that $x = s_{\mathcal{M}}$. In addition, according to the rule of the cascade traceback scheme, $N(s_{\mathcal{M}})$ must contain x. Since $s_{\mathcal{M}} \in N(s_{\mathcal{M}})$, the security goal is satisfied. Without loss of generality, consider again the sensor s_i and its χ children sensors, $s_{j_1}, s_{j_2}, \ldots,$ and s_{j_χ}. Suppose that the verification stalls on the sensor s_{j_ρ}, $\rho \in [1, \chi]$. This means that $N(s_{j_\rho})$ must contain x. In the case where s_i replaces $M_{(i)}$ with $M'_{(i)}$ making $M_{(j_\rho)}$ unverifiable, the second security goal is satisfied because $s_i \in N(s_{j_\rho})$. On the other hand, the case that $x \neq s_i$ can be further divided into two cases, which are the cases where $x = s_{j_\rho}$ and $x \neq s_{j_\rho}$, respectively. In the case where $x = s_{j_\rho}$ and the verification on s_{j_ρ} fails, the second security goal is satisfied because $s_{j_\rho} \in N(s_{j_\rho})$. In the case where $x \neq s_{j_\rho}$, a contradiction occurs because, as s_i and s_{j_ρ} are innocent, the verification should not be stalled on s_{j_ρ}. Hence, according to the argument, the second security goal of the cascade traceback scheme can be achieved.

IV. Performance Evaluation

We provide numerical (Section IV-A), simulation (Section IV-B), and prototype implementation (Section IV-C) results to demonstrate the efficiency and effectiveness of our SMQ schemes.

A. Numerical Result

We will focus on analyzing the critical issue of detecting an incomplete query result in tiered networks. Here, the following two metrics will be analyzed.

1) Detection probability: the probability that query result incompleteness is detected.
2) Communication cost: the number of bits in the communications required in the data report of each sensor and query response of $s_{\mathcal{M}}$ for the proposed schemes.

It is assumed that the number of hops between $s_{\mathcal{M}}$ and each sensor is $\sqrt{n}$ for a collection of n uniformly deployed nodes [2]. In this section, both detection probability and communication cost are discussed at a fixed epoch t.

The communication cost of the encoding approach [21] grows exponentially with the number of attributes. Some crosscheck approaches [28], [33] have relatively low detection probability. The schemes in [32] do not fulfill the DC property. Due to the above reasons, in the following, we only compare SMQ-1 with *(range) hybrid crosscheck* [28], [33], which achieves the best balance between the detection probability and communication cost in the literature. Note that the parameter setting required in hybrid crosscheck is the same as that in [33].

1) Detection Probability: With the fact that the greater the portion of query result $s_{\mathcal{M}}$ drops, the higher the probability that the authority detects it, we consider the worst case where only one bucket and its corresponding data in the query result are dropped by $s_{\mathcal{M}}$ and the number of data sensed by a node is either 0 or 1 as the lower bound of detection probability. Since it is probable that multiple sensed data fall into the same bucket, we further assume that there are on average $p_{dtob}Y$ buckets, where $0 \le p_{dtob} \le 1$, generated in cell $\mathcal{C}$.

To return an incomplete query result in SMQ-1 without being detected by the authority, $s_{\mathcal{M}}$ should create a proof, $\langle \hat{\mathcal{H}}_{\mathcal{M},t}, \hat{\mathcal{P}}_{\mathcal{M},t} \rangle$, corresponding to the incomplete query result. Since bucket primes can be known by the adversary, $\hat{\mathcal{P}}_{\mathcal{M},t}$ can be constructed easily. Nevertheless, $\hat{\mathcal{H}}_{\mathcal{M},t}$ cannot be constructed, since the bucket keys of sensors generating the bucket dropped by $s_{\mathcal{M}}$ are not known by the adversary. Therefore, only two options can be chosen by the adversary. First, the adversary can guess directly to obtain $\hat{\mathcal{H}}_{\mathcal{M},t}$, with probability being $2^{-\ell_h}$, where ℓ_h is the number of bits output by a keyed hash function. This implies that the detection probability $\mathbb{P}_{\text{det},1}^{\text{SMQ}-1}$ of SMQ-1 is $1 - 2^{-\ell_h}$ for the first case. Second, knowing the aggregation tree topology, the adversary may also follow the rule of SMQ-1 to construct the $\hat{\mathcal{H}}_{\mathcal{M},t}$ without considering the bucket keys of the dropped buckets. To construct $\hat{\mathcal{H}}_{\mathcal{M},t}$ in such a way, the adversary has to know at least $\max\{p_{dtob}Y - 1, N - 1\}$ bucket keys. Thus, the probability for the adversary to guess successfully is $2^{-\ell_k \max\{p_{dtob}Y-1,N-1\}}$, where ℓ_k is the number of bits of a key, leading to the detection probability $\mathbb{P}_{\text{det},2}^{\text{SMQ}-1}$ being $1 - 2^{-\ell_k \max\{p_{dtob}Y-1,N-1\}}$ for the second case. Overall, the final detection probability, $\mathbb{P}_{\text{det}}^{\text{SMQ}-1}$, is $\min\{\mathbb{P}_{\text{det},1}^{\text{SMQ}-1}, \mathbb{P}_{\text{det},2}^{\text{SMQ}-1}\}$. As Fig. 6 depicts, the detection probability of SMQ-1 is close to 1 in any case. However, hybrid crosscheck is effective only when a few sensed data are generated in the network. Such a performance difference can be attributed to the fact that the sensed data in the network are summarized securely and deterministically in the proof of SMQ-1 but are summarized merely probabilistically in hybrid crosscheck. As the detection probability of hybrid crosscheck depends on d, in Fig. 6 we vary d and N but should note that $\mathbb{P}_{\text{det}}^{\text{SMQ}-1}$ is independent of d. Note that as SMQ-2 is extended from SMQ-1, the detection probability $\mathbb{P}_{\text{det}}^{\text{SMQ}-2}$ is the same as $\mathbb{P}_{\text{det}}^{\text{SMQ}-1}$ according the similar arguments. Thus, the corresponding curve is omitted in Fig. 6.

2) Communication Cost: For fair comparison, as in prior works [28], [33], we do not count the number of bits in representing data, $\mathcal{E}_{i,t}$, since the delivery of $\mathcal{E}_{i,t}$ is necessary in any data collection scheme.

Each sensor s_i in SMQ-1 is required to send $\mathcal{B}_{i,t}$ and $\mathcal{H}_{i,t}$ to its parent node. Nevertheless, $\mathcal{B}_{i,t}$ and $\mathcal{H}_{i,t}$ can be aggregated along the path in the aggregation tree. As a consequence, s_i actually has only one-hop broadcast containing $\mathcal{B}_{i,t}$ and $\mathcal{H}_{i,t}$

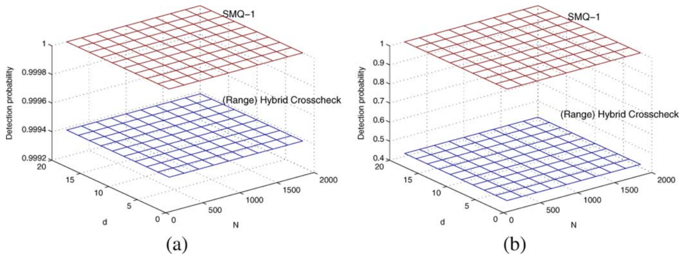


Fig. 6. Detection probability of SMQ-1 and hybrid crosscheck in the cases that (a) $Y = 100$ and (b) $Y = 5000$.

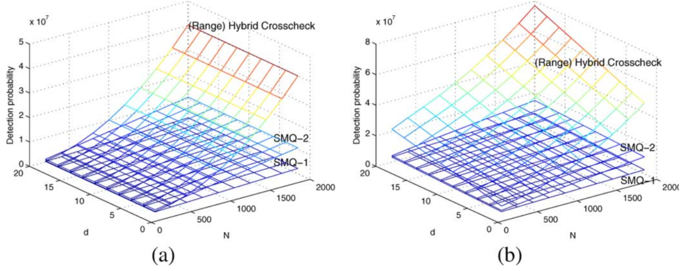


Fig. 7. Communication cost of SMQ-1 and hybrid crosscheck in the cases where (a) $Y = 100$ and (b) $Y = 5000$.

once at each epoch. In addition, to answer a range query, $s_{\mathcal{M}}$ is responsible for sending $\mathcal{H}_{\mathcal{M},t}$, $\mathcal{P}_{\mathcal{M},t}$, and the buckets in $\mathcal{A}$. In summary, the communication cost, $\mathbb{T}^{\text{SMQ}-1}$, can be calculated as $(N-1)\ell_h + p_{dtob}Y\lceil \log w \rceil d\sqrt{N} + \ell_h + \ell_P + |\mathcal{A}|\lceil \log w \rceil d = O(N + d\sqrt{N})$, where ℓ_P is the number of bits used to represent the bucket prime $\mathcal{P}_{\mathcal{M},t}$. In addition to the operations defined in SMQ-1, each node s_i in SMQ-2 additionally transmits $\mathcal{P}_{i,t}$, $\vartheta_{i,t}^0, \ldots, \vartheta_{i,t}^{m-1}$ to its parent node on the aggregation tree. As a result, $\mathbb{T}^{\text{SMQ}-2} = \mathbb{T}^{\text{SMQ}-1} + (N-1)\ell_P + m(N-1)(\ell_h + \ell_P) = O(N + d\sqrt{N})$. Note that in the regular case, where the query result incompleteness does not occur, cascade traceback will not be performed. Thus, the communication cost incurred by the use of cascade traceback is not counted in $\mathbb{T}^{\text{SMQ}-2}$.

As shown in Fig. 7 where the parameters $\ell_h = 80$ and $\ell_P = 1000$ are used, the communication cost of SMQ-1 and SMQ-2 is significantly lower than that of hybrid crosscheck. More specifically, as the communication cost of hybrid crosscheck can be represented asymptotically as $O(N^2 + Nd)$ and will be increased drastically with N and d, the proposed SMQ schemes exhibit very low communication cost regardless of the amount of sensed data in the network due to the fact that the size of the proof ($\mathcal{H}_{\mathcal{M},t}$ and $\mathcal{P}_{\mathcal{M},t}$) used in the SMQ schemes always remains a constant. Hence, the communication cost of the SMQ schemes will be dominated by the aggregation procedure, the average hop distance between $s_{\mathcal{M}}$ and each node, and the transmission of buckets, resulting in $O(N + d\sqrt{N})$ communication cost.

B. Simulation Result

Here, through the simulation, the efficiency of subtree sampling in identifying the compromised nodes is examined. Specifically, the following two metrics will be considered:

1) The number $|\bar{\Upsilon}_{[t_1,t_2]}|$ of nodes to be attested after the use of witness nodes within the time interval $[t_1, t_2]$.

2) The number $\delta_{[t_1,t_2]}$ of compromised sensors identified by the authority. Note that, in Figs. 9–12, $\delta_{[t_1,t_2]}$ is labeled *identifying number*.

1) Simulation Environment: We used MATLAB to conduct the simulation here, which is, thus, independent of mote selection. The reason that we choose MATLAB as our simulation platform is as follows. The primary purpose of our simulation is to evaluate the efficiency of our subtree sampling technique. Such evaluation is irrespective of radio chipset and microcontroller in the mote. In fact, it is hardware independent. In addition, a large number of sensor nodes are needed to fairly evaluate the efficiency. Note that if only a small-scale network is considered (e.g., ten nodes in total), straightforward methods like probing each of sensors may work well. Thus, we choose MATLAB to conduct the efficiency evaluation of our subtree sampling technique in a simulated large-scale network. (As can be seen later, the maximum number of sensor nodes in the simulation reaches 1000.)

In the simulation environment, different numbers of nodes were deployed to simulate a single cell. The communication range of each node is the same, and is set properly such that the nodes form a connected network [1]. In the cell, the aggregation tree [15] is constructed. We also assume error-free and collision-free packet transmission. Since our focus is on the resilience of the subtree sampling against false-incrimination attack, a sensor is picked randomly as the compromised one. As simultaneously launching false-incrimination attacks from multiple compromised nodes does not benefit the adversary, so we assume that only one compromised node ($|\mathfrak{s}| = 1$) launches the false-incrimination attack.

2) Impact of $[t_1, t_2]$*:* Recall that the authority checks the proofs of witness nodes in the time interval $[t_1, t_2]$. We argue that the efficiency of the subtree sampling is independent of $[t_1, t_2]$. Instead, it is only dependent on subtree height m and the number z of witness nodes within $(t_2 - t_1)$ epochs. Fig. 8 can be used to testify for our argument. Specifically, in Fig. 8(a), where the weak filtering criteria is used, two curves representing $\bar{\Upsilon}_{[t_1,t_2]}$ in the cases of $(t_2 - t_1) = 5$ and $(t_2 - t_1) = 20$ are plotted independently. One can observe from Fig. 8(a) that the two curves are nearly identical. The same observation can also be obtained in Fig. 8(b), where the strong filtering criteria is used. This means that our argument is correct even if the filtering criteria used are different. Therefore, in the following discussion of the impact of the other parameters, we simply ignore the impact of $[t_1, t_2]$. Finally, one may observe in Fig. 8(b) that the number of nodes to be attested does not monotonically decrease with m and z. We will explain this phenomenon in Section IV-B4.

3) Impact of N: Comparing Figs. 9(a) and (c) and comparing Figs. 10(a) and (c), we have an immediate observation that $|\bar{\Upsilon}_{[t_1,t_2]}|$ increases N. This stems from two facts. First, as N increases, it is less probable for the sampled subtrees to cover $\mathfrak{s}$. It follows that, when the weak filtering criteria is used, the number of subtrees covering $\mathfrak{s}$ will be decreased, leading to increased $|\bar{\Upsilon}_{[t_1,t_2]}|$ because of the increased cardinality of the intersection result of the subtrees covering $\mathfrak{s}$. Second, more importantly, as the sampled subtrees are more likely to be innocent, (7) will be utilized to determine $\bar{\Upsilon}_{[t_1,t_2]}$. Based on this observation, as N grows, we know that $|\bar{\Upsilon}_{[t_1,t_2]}|$ will increase because, for fixed z, as N is doubled, $|\bar{\Upsilon}_{[t_1,t_2]}|$ grows more than double in most cases. For example, when N is set to be 500 and 1000, as $z = 200$ and all the sampled subtrees are innocent, the corresponding $|\bar{\Upsilon}_{[t_1,t_2]}|$'s will be 300 and 700, respectively. This can be used to explain the differences of Figs. 9(a) and (b), and Figs. 10(a) and (c) with respect to their quantities.

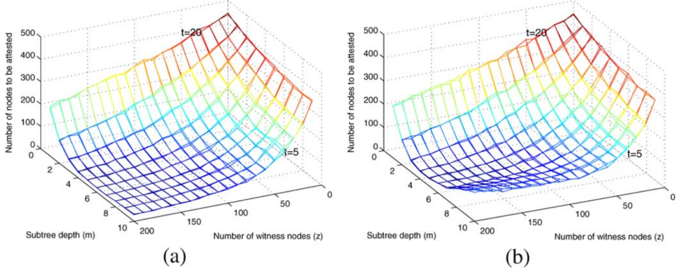


Fig. 8. Results of subtree sampling in the cases of $(t_2 - t_1) = 5$ and $(t_2 - t_1) = 20$. (a) Weak filtering criteria. (b) Strong filtering criteria (must be viewed on a color display.).

4) Impact of m, z*, and Filtering Criteria:* From Figs. 9(a) and (c), we can see that $|\bar{\Upsilon}_{[t_1,t_2]}|$ is monotonically decreased with increase of m and z as the weak filtering criteria is used. This is due to the nature of the weak filtering criteria that the suspicious sets at different epochs are intersected to determine $\bar{\Upsilon}_{[t_1,t_2]}$. Nevertheless, as the strong criteria is employed, since all the nodes in the suspicious sets at different epochs need to be checked, $|\bar{\Upsilon}_{[t_1,t_2]}|$ could turn to be increasing as m and z grow, as shown in Figs. 10(a) and (c). For example, instead of the intersection of two dark triangles in the case where the weak criteria is used [Fig. 5(a)], when strong filtering criteria is employed, the union of the two dark triangles constitutes $\bar{\Upsilon}_{[t_1,t_2]}$. As m and z are small, it is more probable that the subtrees do not cover $\mathfrak{s}$. Due to the fact that the weak and strong filtering criteria share the same rule (7), the trend of $|\bar{\Upsilon}_{[t_1,t_2]}|$ appears to be the same, no matter which filtering criteria is used. As m and z grow, however, it is more likely that multiple subtrees cover $\mathfrak{s}$, and, thus, the difference between the intersection and union operations would appear.

On the other hand, from Figs. 9(b), (d), 10(b), and (d), one can observe easily that as long as $\mathfrak{s}$ always sends the falsified subproofs, the detection probability is close to 1 regardless of m and z. This also revalidates Theorems 1 and 2.

5) Impact of Incriminating Probability: In Figs. 9 and 10, we assume that $\mathfrak{s}$ always sends falsified packets, attempting to frame the innocent $s_{\mathcal{M}}$. Nonetheless, $\mathfrak{s}$ may choose to launch a false incrimination attack occasionally in reality. Note that, as mentioned in Section III-B1, doing so also proportionally downgrades the security impact of false-incrimination attacks. Define p as the *incriminating probability* that $\mathfrak{s}$ launches false-incrimination attack at each epoch. The results in Fig. 11 (Fig. 12) are plotted according to the same same setting as in Fig. 9 (Fig. 10) except that p is set to 0.8. We may be aware of the fact that, as p decreases, $|\bar{\Upsilon}_{[t_1,t_2]}|$ does not vary substantially, when comparing Figs. 9(a) and 11(a), and comparing Figs. 9(c) and 11(c), but $\delta_{[t_1,t_2]}$ turns to decrease when comparing Figs. 9(b) and 11(b), and comparing Figs. 9(d) and 11(d). The above phenomena are due to the fact that some of

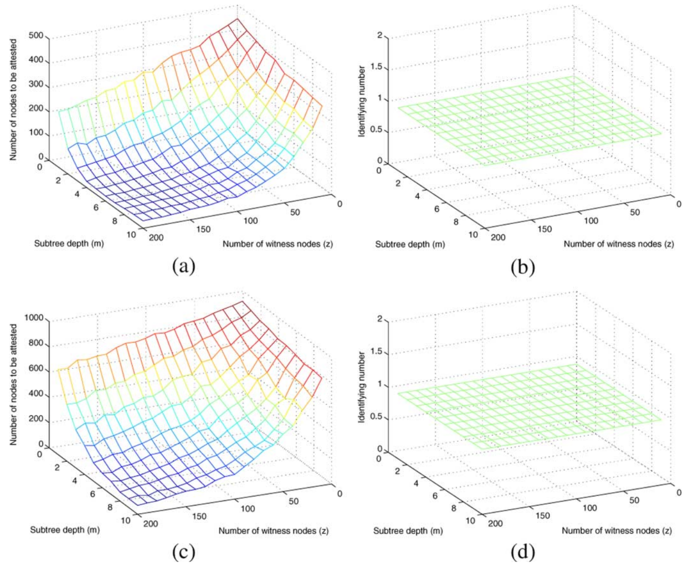


Fig. 9. Results of subtree sampling in the case of weak filtering criteria and $p = 1$ (p is defined in Section IV-B5). (a) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 500)$. (b) $\delta_{[t_1,t_2]}$ $(N = 500)$. (c) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 1000)$. (d) $\delta_{[t_1,t_2]}$ $(N = 1000)$ (must be viewed on a color display.).

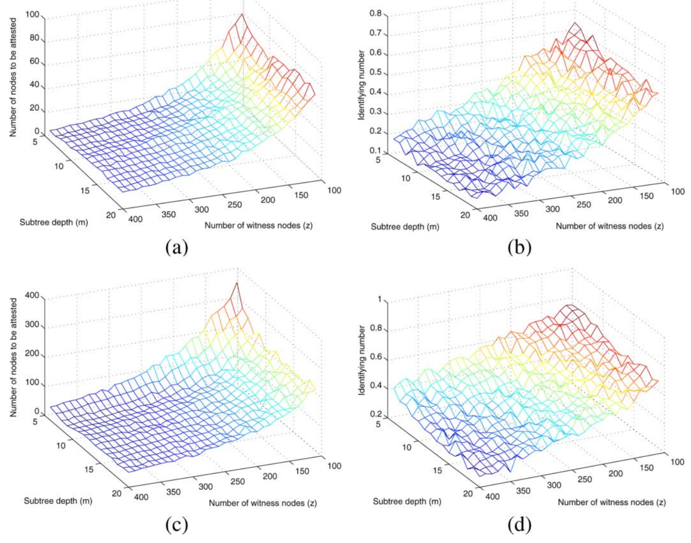


Fig. 11. Results of subtree sampling in the case of weak filtering criteria and $p = 0.8$. (a) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 500)$. (b) $\delta_{[t_1,t_2]}$ $(N = 500)$. (c) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 1000)$. (d) $\delta_{[t_1,t_2]}$ $(N = 1000)$ (must be viewed on a color display.).

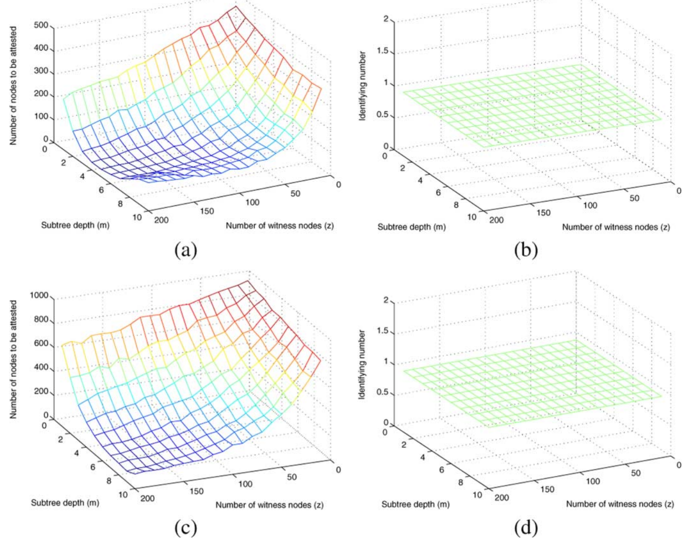


Fig. 10. Results of subtree sampling in the case of strong filtering criteria and $p = 1$ (p is defined in Section IV-B5). (a) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 500)$. (b) $\delta_{[t_1,t_2]}$ $(N = 500)$. (c) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 1000)$. (d) $\delta_{[t_1,t_2]}$ $(N = 1000)$ (must be viewed on a color display.).

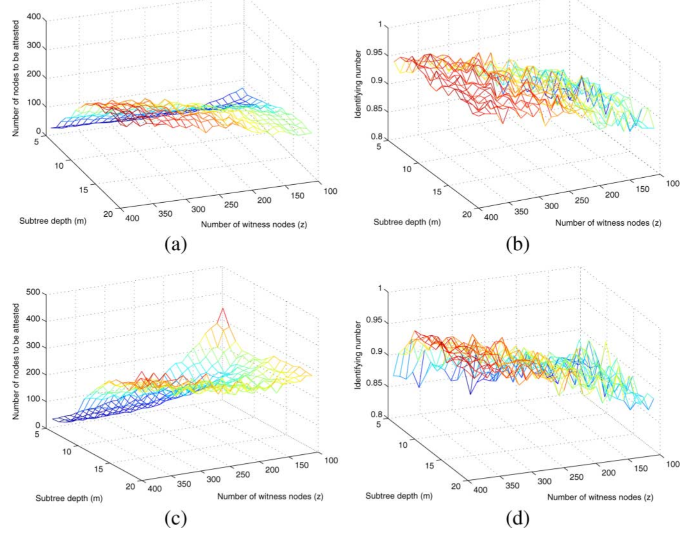


Fig. 12. Results of subtree sampling in the case of strong filtering criteria and $p = 0.8$. (a) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 500)$. (b) $\delta_{[t_1,t_2]}$ $(N = 500)$. (c) $|\bar{\Upsilon}_{[t_1,t_2]}|$ $(N = 1000)$. (d) $\delta_{[t_1,t_2]}$ $(N = 1000)$ (must be viewed on a color display.).

sampled subtrees deemed innocent actually contain $\mathfrak{s}$. This happens when such subtrees cover $\mathfrak{s}$ but at an epoch $\mathfrak{s}$ works honestly. In this situation, no matter whether the weak or strong filtering criteria is used, $\mathfrak{s}$ will be excluded from $\bar{\Upsilon}_{[t_1,t_2]}$, resulting in the decrease of $\delta_{[t_1,t_2]}$. When m and z increase, it is more likely that the compromised sensor happening to work honestly is contained in a sampled subtree. Thus, $\delta_{[t_1,t_2]}$ will become worse with increase of m and z in this situation. In addition, since the subtrees sampled at the epochs at which $\mathfrak{s}$ honestly works are useless in identifying $\mathfrak{s}$, as $|\bar{\Upsilon}_{[t_1,t_2]}|$ are not overwhelmingly large, more sensors need to be checked to find $\mathfrak{s}$.

As shown in Figs. 11(b) and (d), $\delta_{[t_1,t_2]}$, due to the use of weak filtering criteria, is somewhat frustrating. Here, the results in case of strong filtering criteria are plotted in Fig. 12. The comparison of Figs. 11(b) and 12(b) [or Figs. 11(d) and 12(d)] shows that when the strong filtering criteria is employed, it is more likely to discover $\mathfrak{s}$. This is because, as mentioned in Section III-B1, the strong filtering criteria helps eliminate the case where two subtrees at different epochs covering $\mathfrak{s}$ with different judgements (innocent or suspicous) lead to the exclusion of $\mathfrak{s}$ from $\bar{\Upsilon}_{[t_1,t_2]}$. Note that, as mentioned in Section IV-B4, when strong filtering criteria is employed, $|\bar{\Upsilon}_{[t_1,t_2]}|$ will turn to be increasing as m and z grow. This phenomenon also appears in the case of $p = 0.8$, and can be observed in Figs. 12(a) and (c).

TABLE III
SUMMARY OF PROTOTYPE IMPLEMENTATION OF THE PROPOSED SMQ SCHEMES

	ROM	RAM
SMQ-1	23715 Bytes	755 Bytes
SMQ-2	26806 Bytes	769 Bytes

TABLE IV
SUMMARY OF ENERGY CONSUMPTION OF THE PROPOSED SMQ SCHEMES (PERIOD: 60 s)

	CPU	Radio	Overall
SMQ-1	657.5842855 mJ	1124.9958465 mJ	1782.5801325 mJ
SMQ-2	659.8521885 mJ	1128.8540635 mJ	1788.7062520 mJ

C. Prototype Implementation

To study the practicality of our proposed SMQ schemes for the current generation of sensors, we have implemented a prototype of our schemes on TelosB motes on top of the TinyOS platform (Micro-Controller: TI MSP430F1611; ROM: 48 KB+ 256B; RAM: 10 KB; Radio Chipset: ChipCon CC2420).

In the implementation, we focused only on observing the memory overhead of ordinary sensors, because storage node is storage-abundant and is always capable of storing the required materials. The setting of $d = 3$, $w = 5$, and $m = 3$ was used. The bucket primes and keys, each of which are of length 128 bits, were plainly stored in the sensors, and do not utilize the methods stated in Section III-A3. Together with the AES encryption function in a CC2420 chipset, CBC-MAC mode is used to implement the hash function. Table III reports the results of the prototype implementation of the proposed SMQ schemes.

Our program code was also run on TOSSIM in TinyOS 1.1.15 to evaluate the energy consumption of the SMQ schemes. Note that TOSSIM is a discrete-event simulator especially for TinyOS WSNs, on which TinyOS code can be executed directly. Due to the above feature, though TOSSIM is, in essence, a simulator, its estimation of energy consumption is rather accurate. Note that, since the AES function in a CC2420 chipset cannot be acceptable in TOSSIM, we, instead, implemented a software-based AES function for the SMQ schemes in the TOSSIM environment. Thus, we should be aware of that the energy consumption can be further reduced if the hardware-based AES function in CC2420 is used. In our setting, the length of an epoch was 1 s; i.e., the SMQ scheme was triggered every second. The period for which we conducted the simulation was 60 s. The results of TOSSIM simulation are depicted in Table IV.

V. CONCLUSION

We proposed SMQ schemes for securing multidimensional queries, including range query, top-k query, and skyline query. Two critical performance metrics, detection probability and communication cost, were analyzed. In particular, the performance of SMQ is superior to all of the prior works. We also investigated the security impact of collusion attacks and newly identified false-information attacks, and we explored the resiliency of the proposed schemes against these two attacks.

REFERENCES

[1] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in *Proc. ACM MobiHoc*, Lausanne, Switzerland, 2002.

[2] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proc. IEEE INFOCOM*, Miami, FL, 2005.

[3] H. Chan and A. Perrig, "Efficient security primitives from a secure aggregation algorithm," in *Proc. ACM CCS*, Alexandria, VA, 2008.

[4] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proc. ACM CCS*, Alexandria, VA, 2006.

[5] H. Chen, S. Zhou, and J. Guan, "Towards energy-efficient skyline monitoring in wireless sensor networks," in *Proc. EWSN*, Delft, The Netherlands, 2007.

[6] P. Desnoyers, D. Ganesan, H. Li, M. Li, and P. Shenoy, "PRESTO: A predictive storage architecture for sensor networks," in *Proc. HotOS*, Santa Fe, NM, 2005.

[7] H. Hacigumus, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database service provider model," in *Proc. ACM SIGMOD*, Madison, WI, 2002.

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

[9] C. Krauß, M. Schneider, and C. Eckert, "Defending against false-endorsement-based DoS attacks in wireless sensor networks," in *Proc. ACM WiSec*, Alexandria, VA, 2008.

[10] W. Liang, B. Chen, and J. Yu, "Energy-efficient skyline query processing and maintenance in sensor networks," in *Proc. ACM CIKM*, Napa Valley, CA, 2008.

[11] X. Li, Y. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. ACM SenSys*, Los Angeles, CA, 2003.

[12] S. Lumetta and M. Mitzenmacher, "Using the power of two choics to improve bloom filters," *Internet Math.*, vol. 4, no. 1, pp. 17–33, 2009.

[13] D. Liu and P. Ning, "Multi-level μTESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. Embedded Comput. Syst. (TECS)*, vol. 3, no. 4, pp. 800–836, 2004.

[14] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," in *Proc. ACM/IEEE IPSN*, Nashville, TN, 2006.

[15] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny AGgregation service for ad-hoc sensor networks," in *Proc. USENIX OSDI*, Boston, MA, 2002.

[16] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[17] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: Analysis and defenses," in *Proc. IPSN*, Berkeley, CA, 2004.

[18] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," *ACM Trans. Inf. Syst. Security*, vol. 6, no. 3, pp. 365–403, 2003.

[19] RISE Project [Online]. Available: http://www.cs.ucr.edu/~rise

[20] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mobile Netw. Applicat.*, vol. 8, no. 4, pp. 427–442, 2003.

[21] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *Proc. IEEE INFOCOM*, Phoenix, AZ, 2008.

[22] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," *IEEE Trans. Mobile Comput.*, to be published.

[23] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based Attestation for wireless sensors," in *Proc. ESAS*, Visegrad, Hungary, 2005.

[24] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: Secure and resilient time synchronization in wireless sensor networks," in *Proc. ACM CCS*, Alexandria, VA, 2006.

[25] A. Seshadri, A. Perrig, L. V. Doorn, and P. Khosla, "SWATT: SoftWare-based attestation for embedded devices," in *Proc. IEEE S&P*, Berkeley, CA, 2004.

[26] Stargate Gateway (SPB400) [Online]. Available: http://www.xbow.com

[27] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, 2000.

[28] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009.

[29] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Top-k monitoring in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 7, pp. 962–976, Jul. 2007.

[30] C.-M. Yu, C.-S. Lu, and S.-Y. Kuo, "Non-interactive pairwise key establishment for sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 3, pp. 556–569, Sep. 2010.

[31] C.-M. Yu, C.-S. Lu, and S.-Y. Kuo, Secure Multidimensional Queries in Tiered Sensor Networks 2009 [Online]. Available: http://arxiv.org/abs/0911.4238v2

[32] R. Zhang, J. Shi, Y. Liu, and Y. Zhang, "Verifiable top-k queries in tiered sensor networks," in *Proc. IEEE INFOCOM*, San Diego, CA, 2010.

[33] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," in *Proc. ACM MobiHoc*, New Orleans, LA, 2009.

Chia-Mu Yu (S'09) is working toward the Ph.D. degree at the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

He was a research assistant in the Institute of Information Science, Academia Sinica, Taipei, Taiwan. He is currently a visiting scholar at Harvard University. His research interests include experimental and theoretical aspects of sensor network security.

Yao-Tung Tsou is currently working toward the Ph.D. degree at the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

He has been a research assistant in the Institute of Information Science, Academia Sinica, Taipei, Taiwan, since 2009. His research interests include sensor network security and applications.

Chun-Shien Lu (M'99) received the Ph.D. degree in electrical engineering from National Cheng-Kung University, Tainan, Taiwan, in 1998.

From October 1998 to July 2002, he joined Institute of Information Science, Academia Sinica, Taiwan, as a postdoctoral fellow for his military service. From August 2002 to July 2006, he was an assistant research fellow at the same institute. Since July 2006, he has been an associate research fellow. His current research interests mainly focus on various topics (including signal processing and security) of multimedia, sensor network security, and compressive sensing.

Dr. Lu organized a special session on Multimedia Security in the 2nd and 3rd IEEE Pacific-Rim Conference on Multimedia, respectively (2001–2002). He coorganized two special sessions (in the area of media identification and DRM) in the 5th IEEE International Conference on Multimedia and Expo (ICME), 2004. He is a guest coeditor of the *Special Issue on Visual Sensor Network, EURASIP Journal on Applied Signal Processing*, in 2005. He has owned two U.S. patents, two ROC patents, and one Canadian patent in digital watermarking. He won the Ta-You Wu Memorial Award, National Science Council in 2007 and was a corecipient of a National Invention and Creation Award in 2004. From July 2007, he has served as a member of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society. He is currently an associate editor of IEEE TRANSACTIONS ON IMAGE PROCESSING. He is a member of the ACM.

Sy-Yen Kuo (S'85–M'88–SM'98–F'01) received the B.S. degree in electrical engineering from National Taiwan University, the M.S. degree in electrical and computer engineering from the University of California at Santa Barbara, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, in 1979, 1982, and 1987.

He is a Distinguished Professor at the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan and was the Chairman at the same department from 2001 to 2004. He was a Chair Professor and Dean of the College of Electrical and Computer Engineering, National Taiwan University of Science and Technology from 2006 to 2009. He spent his sabbatical years as a Visiting Professor at the Computer Science and Engineering Department, the Chinese University of Hong Kong, from 2004 to 2005 and as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000, respectively. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty fellow at the Jet Propulsion Laboratory of California Institute of Technology. His current research interests include dependable systems and networks, mobile computing, and quantum computing and communications. He has published more than 300 papers in journals and conferences, and also holds more than ten U.S. and Taiwan patents.

Prof. Kuo received the distinguished research award between 1997 and 2005, consecutively, from the National Science Council in Taiwan and is now a Research Fellow there. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference (DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.