A Spatiotemporal Approach for Secure Range Queries in Tiered Sensor Networks

Jing Shi, *Student Member, IEEE,* Rui Zhang, *Student Member, IEEE,* and Yanchao Zhang, *Member, IEEE*

***Abstract*—We target a two-tier sensor network with resource-rich master nodes at the upper tier and resource-poor sensor nodes at the lower tier. Master nodes collect data from sensor nodes and answer the queries from the network owner. The reliance on master nodes for data storage and query processing raises serious concerns about both data confidentiality and query-result correctness in hostile environments. In particular, a compromised master node may leak hosted sensitive data to the adversary; it may also return juggled or incomplete data in response to a query. This paper presents a novel spatiotemporal approach to ensure secure range queries in event-driven two-tier sensor networks. It offers data confidentiality by preventing master nodes from reading hosted data and also enables efficient range-query processing. More importantly, it allows the network owner to verify with very high probability whether a query result is authentic and complete by examining the spatial and temporal relationships among the returned data. The high efficacy and efficiency of our approach are confirmed by detailed performance evaluations.**

***Index Terms*—Wireless sensor networks, range query, security.**

I. Introduction

DATA access in wireless sensor networks (WSNs) normally follows a push or pull model. In the push model, the data continuously generated by sensor nodes are sent in real time to an external data sink, where various data queries can be resolved in a centralized fashion. The push model is well suitable for monitoring and surveillance applications that desire live data. In contrast, in the pull model, sensor data are stored inside the network and await on-demand queries. The pull model is very suitable for many civilian, commercial, scientific, and military applications without need for live data [2]–[6]. In addition, the pull model can greatly save the scarce energy of sensor nodes if only a small portion of the potentially huge amount of data produced over time may be needed [2]–[6]. The pull model is also the only feasible approach in remote and extreme environments where it is impossible or prohibitive to maintain an always-on high-speed communication connection from the WSN to the external sink whereby sensor data can be sent in real time.

Manuscript received April 6, 2010; revised June 25, 2010; accepted September 26, 2010. The associate editor coordinating the review of this paper and approving it for publication was T. Hou.

J. Shi was a Ph.D. student of Y. Zhang at the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07103 when this work was done (e-mail: js39@njit.edu).

R. Zhang and Y. Zhang are with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 (e-mail: {ruizhang, yczhang}@asu.edu).

This work was supported in part by the US National Science Foundation under grants CNS-0716302 and CNS-0844972 (CAREER). The preliminary version of this paper appeared in IEEE INFOCOM '09 [1].

Digital Object Identifier 10.1109/TWC.2010.102210.100548

In this paper, we target a large-scale two-tier WSN with in-network storage and query processing as in [2], i.e., following the pull model. The lower tier comprises a large number of resource-constrained sensor nodes, while the upper tier contains fewer relatively resource-rich *master node*s. Sensor nodes are mainly responsible for sensing tasks, while master nodes perform more resource-demanding computation and communication tasks. Master nodes also form a multi-hop wireless mesh network via long-range high-bandwidth radios. As in [2], every master node is equipped with several gigabytes of NAND flash storage, which costs only a few tens of dollars. The network field is partitioned into physical *cell*s, each containing a master node in charge of sensor nodes in that cell. Master nodes collect data from affiliated sensor nodes and store them locally for extended periods of time. The network owner can query data through an on-demand communication link (e.g., a satellite link) to some master node(s). This two-tier architecture is also indispensable for increasing network capacity and scalability, reducing system complexity, and prolonging network lifetime [2], [7].

The reliance on master nodes for data storage and query processing raises serious security concerns. In particular, many target application environments of WSNs such as forests and oceans are unattended and hostile in nature. Master nodes are attractive targets of attack and might be compromised by the adversary. Compromised master nodes will leak sensitive data such as intrusion events or rare animals' behavior patterns to the adversary. A sound scheme is thus required to let master nodes store encrypted data for which they do not hold the decryption keys, while enabling efficient query processing. In addition, the adversary may instruct compromised master nodes to return juggled and/or incomplete data in response to data queries from the network owner. Such attacks are obviously more subtle and harmful than blind DoS attacks on the WSN, especially when the query results are used as the basis for making critical decisions. The network owner thus cannot accept the query results at their face value. Instead, it should be able to verify that a query result is both *authentic* and *complete*. The term *authentic* means that all the data in the result originated from the purported sources and have not been tampered with, and *complete* means that the result includes all the data satisfying the query. A WSN may need to support many types of data queries. In this paper, we focus on supporting range queries which are an important and common type of queries in sensor networks and ask for data within a certain range [8].

Despite significant progress in WSN security, secure range queries have drawn attention only very recently. In their pioneering work [5], Sheng and Li apply the bucketing technique

[9], [10] to store encrypted data at master nodes and also ensure rang-query efficiency. If the encryption algorithm is an OCB-like authenticated encryption primitive [11], their scheme can ensure data confidentiality and query-result authenticity. To permit query-result completeness verification, they propose that every sensor node that has no data satisfying the range query return some verifiable information through the involved master node to the network owner. If the sink receives neither data nor the verifiable information from any sensor node, it knows that the master node must have maliciously omitted the data from that sensor node. Their approach is very suitable for monitoring-type WSN applications (e.g., temperature monitoring) in which each sensor node needs to periodically report data to its affiliated master node. It, however, may incur unnecessarily high communication overhead in event-driven WSN applications (e.g., target tracking) where events occur infrequently and most sensor nodes have no data to report. A detailed analysis of the communication overhead of this scheme can be found in Section V-A4.

This paper, for the first time in literature, investigates techniques to secure range queries in event-driven two-tier WSNs with in-network storage and query processing. We adopt the bucketing technique [9], [10] to strike a balance between data confidentiality and query efficiency. Our major contribution is a novel spatiotemporal approach for the network owner to verify query-result completeness. In particular, our approach consists of a spatial crosscheck technique and a temporal crosscheck technique. The former aims to create some relationships among data generated by sensor nodes in charge by the same master node, while the latter aims to embed some relationships among data produced in different time periods. These two techniques collectively allow the network owner to verify with high probability whether a query result is authentic and complete by examining the spatial and temporal relationships among the returned data. Since our approach involves only nodes that have data to report and uses only symmetric cryptographic primitives, it is very efficient in both communication and computation. The efficacy and efficiency of the proposed approach are confirmed by detailed performance evaluations.

The rest of this paper is structured as follows. Section II presents the adversary and event models. Section III illustrates how to perform range queries over encrypted data based on the bucketing technique [9], [10]. Section IV presents our spatiotemporal approach. Section V thoroughly analyzes and evaluates the proposed techniques. Section VI finally concludes this paper.

II. Adversary and Event Models

A. Adversary Model

This paper focuses on thwarting attacks on secure range queries in tiered WSNs, and we refer to the existing rich literature for effective defenses against other attacks. We assume that the adversary can compromise an arbitrary number of master nodes. Once compromising a master node, the adversary can access data stored there and also instruct it to return juggled and/or incomplete data in response to range queries from the network owner. The adversary may also compromise sensor nodes and access any information stored on them. Since a compromised sensor node only has very limited information and there are many more sensor nodes than master nodes with more important roles, the adversary will be motivated to compromise master nodes in order to do more damage. For lack of space, we follow the adversary model in [5] and focus on dealing with compromised master nodes in this paper. The impact of compromised sensor nodes on multidimensional range queries has been addressed in [12], in which the proposed countermeasures can apply to our work in this paper with minimal modifications.

B. Event-Driven Application Model

In this subsection, we clarify the event-driven application model used throughout the paper. Similar to [5], we assume that time is divided into *epoch*s and that sensor and master nodes are loosely synchronized. We assume there are totally F events observed in a target cell during epoch t, where $F \in [0, F_m]$ is a random number, and F_m is the maximum number of events that can be observed in a cell during each epoch. Let n_f denote the number of distinct sensor nodes detecting event $f, 1 \leq f \leq F$. It is worth noting that one sensor node may detect multiple events in the same epoch. F and n_f may vary in different cells and epochs and follow some distributions which depend on concrete applications.

III. Range Queries over Encrypted Data

In this section, we illustrate how to realize data confidentiality, efficient range queries, and query-result authentication, which is the basis for our spatiotemporal crosscheck scheme.

We assume that each epoch consists of three phases. In the longest *detection* phase, each sensor node performs sensing. In the subsequent *reporting* phase, each sensor node submits to its master node all the data (if any) it produced during that epoch. Depending on concrete applications, each data item may comprise multiple attributes such as the weight of an observed object, its location, its speed and moving trajectory, and its appearance and lingering times. In the final *query* phase, the network owner may issue queries for data generated in that epoch. During the relatively much shorter reporting and query phases, some nodes may generate some data which will be carried over to the next epoch. In this paper, we aim to support only epoch-based and cell-based single-attribute range queries, e.g., "Return all the data items generated during epoch t in cell id_{cell} whose weight attribute is between 100 and 120 pounds."

Without loss of generality, we subsequently focus on a cell id_{cell} consisting of N sensor nodes, denoted by $\{S_i\}_{i=1}^{N}$, and a master node, denoted by $\mathcal{M}$. It should be noted that our techniques apply to every cell with or without a compromised master node which is hard to predict. Each node S_i is preloaded with a distinct key $K_{i,0}$ known only to itself and the network owner. At the end of epoch $t \geq 1$, S_i generates an epoch key $K_{i,t} = H(K_{i,t-1})$ and erases $K_{i,t-1}$ from its memory, where $H(\cdot)$ denotes a good hash function. Such epoch keys are used to realize forward-secure encryption of data produced in each epoch. For example, suppose that the adversary compromises $\mathcal{M}$ and S_i during epoch $t+1$. He will

not be able to read the encrypted data S_i submitted to $\mathcal{M}$ in the past t epochs, as all the corresponding encryption keys $\{K_{i,j}\}_{j=1}^{t}$ no longer exist.

As in [5], we adopt the bucketing technique [9], [10] to strike a balance between data confidentiality and query efficiency. The bucketing technique partitions the queriable attribute domain into $g \geq 1$ consecutive non-overlapping regions (buckets), sequentially numbered from 1 to g, and the value of g and the partitioning rule are assumed to be public knowledge. At the end of each epoch t, S_i, $\forall i \in [1, N]$, encrypts the data items falling into the same bucket as a whole and then sends these encrypted blocks to $\mathcal{M}$. For instance, assume that S_i has $3|2|1$ data items in buckets $1|3|5$, respectively. S_i sends the following message to $\mathcal{M}$ during the reporting phase of epoch t:

$$S_i \rightarrow \mathcal{M} : i, t, \langle 1, (data1, data2, data3)_{K_{i,t}} \rangle, \\ \langle 3, (data4, data5)_{K_{i,t}} \rangle, \langle 5, (data6)_{K_{i,t}} \rangle,$$

where $(\cdot)_*$ denotes an OCB-like authenticated encryption primitive [11] using the key on the subscript.

The query process is fairly simple. The network owner first converts its desired data range into bucket IDs (denoted by $\mathcal{Q}_t$) and then sends them together with id_{cell} and epoch number t to $\mathcal{M}$. Upon receiving the query $\langle id_{cell}, t, \mathcal{Q}_t \rangle$, $\mathcal{M}$ returns all the encrypted data blocks with bucket IDs in $\mathcal{Q}_t$ along with the corresponding sensor node IDs. The network owner can then derive all the corresponding epoch keys whereby to decrypt the received information. Since the data range of interest may not exactly span consecutive full buckets, the encrypted data of the lowest and highest buckets in the query reply may contain superfluous data items (false positives) the network owner does not want. One way to reduce such false positives and thus the related unnecessary communication overhead is to use finer bucketing, i.e., increasing g. This measure, however, helps $\mathcal{M}$ (if compromised) more accurately estimate the distribution of sensed data and thus may jeopardize the data confidentiality. We refer readers to [10] for optimal bucketing strategies which can achieve a good balance between false positives and data confidentiality.

In addition to ensuring data confidentiality against $\mathcal{M}$, the authenticated encryption primitive allows the network owner to detect forged or juggled data in the query result, as $\mathcal{M}$ does not know the correct epoch keys. Unfortunately, $\mathcal{M}$ may still omit data of some nodes which satisfy the query, leading to query-result incompleteness. This behavior is difficult to detect and will be addressed in the following section.

IV. A Spatiotemporal Approach to Secure Range Queries

In this section, we present a novel spatiotemporal approach allowing the network owner to verify the completeness of range queries, which includes a spatial crosscheck technique, a temporal crosscheck technique, and their combination.

A. Spatial Crosscheck (SC)

We first introduce a novel spatial crosscheck (SC) technique. We consider cell id_{cell} which consists of master node $\mathcal{M}$ and sensor nodes $\{S_i\}_{i=1}^{N}$. The key idea of spatial crosscheck is to embed some relationships among data generated by different nodes during each epoch, e.g., embedding the information about node S_j's data into node S_i's data buckets. Under this technique, if $\mathcal{M}$ omits data from some sensor nodes, the network owner can decide with high probability that the query result is incomplete by inspecting the spatial relationships among the returned data. $\mathcal{M}$ is thus forced to either return all the data satisfying the query or none of them in order to escape the detection.

To realize the SC technique, each node need disseminate the information about its own data which can then be embedded into other nodes' data. How to disseminate such information is critical in designing SC, which determines the efficacy and efficiency of SC. In the following, we detail three versions of SC, including a broadcast-based spatial crosscheck method (BSC), a neighbor-based one (NSC), and a hybrid one (HSC) which is the combination of BSC and NSC.

In BSC, before submitting its data to $\mathcal{M}$, every sensor node broadcasts the information about its data (if any) within cell id_{cell}. The broadcasted information consists of its ID and a vector of g bits, called *data index*, where each bit indicates whether the node has detected data in the corresponding bucket or not. We denote the data index of S_i at epoch t by $\mathsf{V}_{i,t}$. For example, if $g = 8$ and $\mathsf{V}_{i,t} = 10101000$, then S_i only detected data for buckets $1|3|5$ during epoch t. Every node with data for submission sets a timer to the estimated longest end-to-end message transmission time in cell id_{cell} to allow enough time for receiving other nodes' data indexes. Then it embeds its own data index and all the received ones along with the corresponding node IDs into each of its own data buckets. Finally, it sends the encrypted buckets to $\mathcal{M}$ as in Section III. For example, assume that S_i has $3|2|1$ data items in buckets $1|3|5$, respectively, and has data indexes $\{\mathsf{V}_{l,t}\}_{l=1}^{k}$ before submission, including the received ones and its own, i.e., $S_i \in \{S_l\}_{l=1}^{k}$. It finally sends the following message to $\mathcal{M}$ during the reporting phase.

$$S_i \rightarrow \mathcal{M} : i, t, \langle 1, (data1, data2, data3, \{j_l, \mathsf{V}_{l,t}\}_{l=1}^{k})_{K_{i,t}} \rangle, \\ \langle 3, (data4, data5, \{j_l, \mathsf{V}_{l,t}\}_{l=1}^{k})_{K_{i,t}} \rangle, \\ \langle 5, (data6, \{j_l, \mathsf{V}_{l,t}\}_{l=1}^{k})_{K_{i,t}} \rangle \ .$$

The effectiveness of BSC can be easily seen through the following example. Assume that the network owner sends a query only for bucket 5; nodes $\Theta \subseteq \{S_i\}_{i=1}^{N}$ have data in bucket 5; and $\mathcal{M}$ drops the data bucket of $S_i \in \Theta$. Since $\langle i, \mathsf{V}_{i,t} \rangle$ is embedded in bucket 5 of every other node in $\Theta \setminus \{S_i\}$, it can reach the network owner as long as $\mathcal{M}$ does not drop all the data buckets satisfying the query. If the network owner finds $\langle i, \mathsf{V}_{i,t} \rangle$ in any received bucket and does not receive the bucket from S_i, it can determine that $\mathcal{M}$ must be malicious. Therefore, $\mathcal{M}$ has to drop all the data buckets to escape detection.

The above example also manifests some possibly unnecessary communication overhead. In particular, multiple copies of $\langle i, \mathsf{V}_{i,t} \rangle$ may reach the network owner, while one copy is enough. Therefore, it is unnecessary for each node to store each received data index into all its buckets. One may think of letting each node only embed the data indexes into the buckets which will be queried by the network owner. It is, however,

difficult to predict the network owner's interests. Therefore, we propose that each node embed each received data index in each produced bucket with equal probability p_e.

Now we introduce the NSC technique which can further reduce the communication overhead. This technique is motivated by the essential nature of event-driven sensor networks: if one node detects any event, it is more likely for its neighboring nodes and less likely for those far away from it to detect the same event and thus generate data in the same bucket. Therefore, if one node generated data satisfying the query, it is most likely that its nearby nodes also generated qualified data. It is thus more cost-effective to let the data of neighboring sensor nodes crosscheck each other.

To illustrate NSC, we take node S_i as an example. Assume that S_i has generated event data in epoch t. Before submitting its data to $\mathcal{M}$, S_i locally broadcasts $\langle \mu, i, \mathsf{V}_{i,t} \rangle$, where μ is a system parameter specifying the maximum number of hops for which the message should be forwarded. Each of S_i's neighbors further locally broadcasts $\langle \mu-1, i, \mathsf{V}_{i,t} \rangle$ if $\mu-1 > 0$ and also inserts $\langle i, \mathsf{V}_{i,t} \rangle$ into each of its own data buckets (if any) with probability p_e. To summarize, each node locally broadcasts not only its own data index but also other received data indexes which have not been transmitted beyond μ hops. A node may receive the same data index multiple times from different neighbors, in which case it only processes the first version.

Intuitively, NSC significantly reduces the communication overhead in contrast to BSC, as it only requires local broadcasting of the data indexes within μ hops instead of the whole cell. As long as μ is small, the communication cost of NSC will be smaller than that of BSC. However, NSC has one flaw. Consider the circumstance that there are only a few events observed in the whole cell, and the number of nodes detecting each event is so small that the physical subregions associated with those events do not overlap. Then $\mathcal{M}$ can escape detection by dropping all the data buckets originating from one subregion but returning all those belong to other subregions.

We combine NSC and BSC to remedy each method's flaw and refer to this new method as HSC. In HSC, a node chooses to broadcast its data index within the cell with probability p_b and locally broadcast its data index following the NSC scheme with probability $1-p_b$. In this way, some nodes' data indexes are distributed in the whole cell. Since $\mathcal{M}$ cannot differentiate those nodes from others, it can no longer drop all the data generated in one subregion without being detected. The relationships among p_b, the detection capability, and the communication cost of NSC will be thoroughly analyzed and evaluated in Section V.

B. Temporal Crosscheck (TC)

Our HSC technique enables the network owner to detect $\mathcal{M}$'s misbehavior with very high probability as long as it can receive at least one data bucket. $\mathcal{M}$, however, can still escape the detection by dropping all the qualified data in the cell. To this end, we further propose a temporal crosscheck technique as a complement to HSC.

The basic idea of temporal crosscheck (TC) is to let each node embed some relationships among its own data produced in different epochs. To enable TC, we require each node to maintain a fixed-length FIFO buffer of $\mathcal{L}(g+\gamma)$ bits, where γ denotes the length of an epoch ID which can roll over. The buffer can thus hold at most $\mathcal{L}$ pairs of data indexes and epoch IDs. At the end of each epoch t, each node, say S_i, inserts its own data index $\mathsf{V}_{i,t}$ into the buffer with equal probability p_T. The buffer of S_i thus contains the data indexes of past epochs which are not necessarily consecutive. If S_i has any data for submission, it embeds the buffer into every bucket it wants to submit and then encrypts the data and the buffer in each bucket together using an OCB-like authenticated encryption primitive [11] with its epoch key. If $\mathcal{M}$ returns the data of S_i in response to a query, the buffer of S_i is also sent along with the data to the network owner and thus can be decrypted and authenticated by the network owner.

$\mathcal{M}$ can become very suspicious if the network owner does not receive any data for a few consecutive queries. Let $\alpha \geq 1$ be a system parameter, specifying the maximum number of consecutive droppings the network owner can tolerate. If the network owner does not receive any data for α consecutive queries which span no less than α epochs, it will suspect that $\mathcal{M}$ might have been compromised and thus would diagnose $\mathcal{M}$ using some accurate yet expensive technique such as software-based memory attestation [13]. To avoid raising the suspicion, $\mathcal{M}$ thus will not drop data for more than α consecutive queries.

To be more clear, assume that S_i generates some data in epoch t which will satisfy the query made in epoch t and inserts $\mathsf{V}_{i,t}$ in its buffer. We also assume that $\mathcal{M}$ consecutively drops α' query results, starting from epoch t. Because $\mathcal{M}$ tries to avoid raising suspicion, α' is not larger than α. If the network owner's next query arrives before $\mathsf{V}_{i,t}$ is moved out of the buffer and S_i has data satisfying the new query, $\mathcal{M}$ will have to return S_i's buffer as part of the query result. The network owner can immediately catch $\mathcal{M}$'s misbehavior after knowing that S_i indeed had qualified data in epoch t.

C. Combination of SC and TC

HSC and TC are complementary to each other. In particular, HSC enables the crosscheck of different nodes' data generated in the same epoch and thus forces the master node to either return or drop all the data, while TC enables the crosscheck of a node's own data generated in different epochs and targets detecting the drop-all misbehavior. On the one hand, without HSC, the malicious master node can escape detection by always dropping all the data from a fixed set of nodes and returning all the data of other nodes. On the other hand, without TC, the malicious master node can escape detection by dropping all the data in one epoch and returning all the data in the next epoch. Therefore, we require each node to simply follow both HSC and TC and refer to the combined technique as spatiotemporal crosscheck (STC).

V. Performance Evaluation

In this section, we evaluate the performance of our spatiotemporal approach. As said before, the encoding technique in [5] is the first and only related work on secure one-dimensional range queries besides our scheme, and we are the

first to notice that their approach is ill-suited for event-driven sensor networks. To validate this observation, we also compare our approach with the encoding technique. In addition, the encoding technique and our approach are both application-layer security mechanisms independent of the underlying network protocol stack and only require that unicast and broadcast communications be feasible as needed. To better demonstrate the tradeoff between security and overhead incurred by our scheme itself, we adopt the same approach as in [5] and just consider the additional communication overhead added by our approach in what follows. In this way, the impact of using different underlying network protocol stacks can be eliminated.

A. Performance Analysis

We analyze the performance of our HSC, TC, and STC techniques using the following metrics.

- P_{det}**–detection probability**: P_{det} is defined as the probability that the network owner can detect the misbehavior of $\mathcal{M}$ in returning incomplete query results.
- $\bar{\mathrm{T}}$**–communication cost**: $\bar{\mathrm{T}}$ is defined as the total communication energy consumption in bits for completeness verification of query results from cell id_{cell}. We only consider the cost associated with transmitting data indexes, which is the extra overhead incurred by our scheme. Since master nodes have much more energy resources than sensor nodes, we ignore the energy consumption for transmitting data indexes from master nodes to the network owner for simplicity. Here we assume the same energy consumption to transmit and receive each bit across each hop.

1) Analysis of HSC: **Detection Probability**

We first derive the detection probabilities of BSC and NSC, respectively. To enable quantitative analysis, we assume on average there are $\bar{F}$ events observed per epoch per cell, and each event is detected by $\bar{n}$ sensor nodes. The network owner issues a sequence of range queries in the query phase of an epoch t which together compose a set $\mathcal{Q}_t \subseteq \{1, \ldots, g\}$ of bucket IDs. Note that the bucket IDs in $\mathcal{Q}_t$ may not be continuous. We assume that $\mathcal{M}$ omits each data bucket from every sensor node with equal probability p_d.

The following lemma 1 and lemma 2 are about the detection probabilities of BSC and NSC, respectively.

LEMMA *1: With* BSC*, the network owner can detect* $\mathcal{M}$ *dropping any particular data bucket satisfying* $\mathcal{Q}_t$ *with probability*

$$P_{det}^{\mathsf{B}} = 1 - p_d^{(|\mathcal{Q}_t|Np-1)p_e} \ , \tag{1}$$

where

$$p = 1 - (1 - \frac{1}{g}\frac{\bar{n}}{N})^{\bar{F}}, \tag{2}$$

and $|\mathcal{Q}_t|$ *is the cardinality of* $\mathcal{Q}_t$.

Proof: We first estimate the number of buckets generated in the whole cell which satisfy the query of network owner, denoted by Δ_t. We assume that the data resulting from one event fall into a single bucket. Let E_f $(1 \le f \le F)$ be the set of nodes which have detected event f, where $|\mathsf{E}_f| = n_f$. Then the probability of any node S_i detecting event f and belonging to E_f is given by $\Pr(S_i \in \mathsf{E}_f) = \frac{n_f}{N}$, where $\Pr(*)$ denotes the probability of one event $*$ occurring. Let B_j $(1 \le j \le g)$ be the set of nodes that have data in the bucket j. Each event falls into any bucket with equal probability $1/g$, and the probability that S_i has data in bucket j is given by $\Pr(S_i \in \mathsf{B}_j) = 1 - \prod_{f=1}^{F}(1 - \frac{1}{g}\frac{n_f}{N})$.

The probability p that each node has data in each bucket can be approximated by $1-(1-\frac{1}{g}\frac{\bar{n}}{N})^{\bar{F}}$. The total number of nodes with at least one non-empty bucket can be approximated by Np, and thus we can derive Δ_t as $\Delta_t = |\mathcal{Q}_t|Np = |\mathcal{Q}_t|N(1-(1-\frac{1}{g}\frac{\bar{n}}{N})^{\bar{F}})$, where $|\mathcal{Q}_t|$ denotes the cardinality of $\mathcal{Q}_t$.

Now we consider the detection probability P_{det}^{B} that the network owner can detect $\mathcal{M}$ dropping any particular data bucket satisfying $\mathcal{Q}_t$. Assume that node S_i generated some data buckets satisfying $\mathcal{Q}_t$, and one of them has been dropped by $\mathcal{M}$. The network owner can detect any particular dropped bucket if it receives any data bucket that contains $\langle i, \mathsf{V}_{i,t}\rangle$. Denote by n_w the number of data buckets containing $\langle i, \mathsf{V}_{i,t}\rangle$ and satisfying $\mathcal{Q}_t$. It is easy to see that n_w can be approximated by $(|\mathcal{Q}_t|Np-1)p_e$, where p_e is the probability that a node stores a received index in each bucket. Assume that $\mathcal{M}$ drops each data bucket with equal probability p_d, the probability that all the n_w copies of the data index are dropped by $\mathcal{M}$ is $p_d^{n_w}$. Therefore, P_{det}^{B} can be derived as $P_{det}^{\mathsf{B}} = 1 - p_d^{n_w} = 1 - p_d^{(|\mathcal{Q}_t|Np-1)p_e}$. ∎

LEMMA *2: With* NSC*, the network owner can detect* $\mathcal{M}$ *dropping any particular data bucket satisfying* $\mathcal{Q}_t$ *with probability*

$$P_{det}^{\mathsf{N}} = 1 - p_d^{n_w} \ , \tag{3}$$

where

$$n_w = \Big(\sum_{x=1}^{\mu} x\lambda p_o^x + (|\mathcal{Q}_t|(N_\mu + 1) - \sum_{x=1}^{\mu} x\lambda p_o^x - 1)p\Big)p_e \ , \tag{4}$$

$p_o = 1 - \frac{\sqrt{4\lambda(\bar{n}-1)+\lambda^2}-\lambda}{2(\bar{n}-1)}$, *and* $N_\mu = \frac{\mu(\mu+1)}{2}\lambda$.

Proof: To derive P_{det}^{N}, we need first estimate the number of copies of each node's data index. For analytical tractability, we assume that when any node detects one event, each of its neighbors detects the same event with equal conditional probability p_o, and that when $\mu \to \infty$, the total number of nodes that detected the same event in NSC approaches that in BSC.

Assuming that sensor nodes are uniformly distributed and that each node has λ neighboring nodes, the average number of x-hop neighbors of S_i is approximately $x\lambda$. Let χ denote the number such that $p_o^\chi \ll 1$. We thus have the average number of nodes detecting the same event as

$$\bar{n} \approx 1 + \sum_{x=1}^{\chi} x\lambda p_o^x \approx 1 + \sum_{x=1}^{\infty} x\lambda p_o^x = 1 + \frac{\lambda p_o}{(1-p_o)^2} \ , \tag{5}$$

which holds if χ is sufficiently large. By solving Eq. (5), we have $p_o = 1 - \frac{\sqrt{4\lambda(\bar{n}-1)+\lambda^2}-\lambda}{2(\bar{n}-1)}$.

Now we derive P_{det}^{N}. Similar to P_{det}^{B}, P_{det}^{N} can be calculated as $1-p_d^{n_w}$, where n_w is the number of data buckets containing $\langle i, \mathsf{V}_{i,t}\rangle$ and satisfying the query $\mathcal{Q}_t$. The number of nodes within μ hops from S_i, denoted by N_μ, is given by $N_\mu = \lambda + 2\lambda + \ldots + \mu\lambda = \frac{\mu(\mu+1)}{2}\lambda$.

There are totally $|\mathcal{Q}_t|(N_\mu+1)$ possible buckets satisfying the query, among which $\sum_{x=1}^{\mu} x\lambda p_o^x + 1$ resulting from the same event are non-empty. For the other $|\mathcal{Q}_t|(N_\mu+1) - \sum_{x=1}^{\mu} x\lambda p_o^x - 1$ buckets, each is non-empty with probability p. The total number of buckets satisfying the query within μ-hop neighborhood is thus $\sum_{x=1}^{\mu} x\lambda p_o^x + 1 + (|\mathcal{Q}_t|(N_\mu+1) - \sum_{x=1}^{\mu} x\lambda p_o^x - 1)p$. So n_w can be estimated as $n_w = \big(\sum_{x=1}^{\mu} x\lambda p_o^x + (|\mathcal{Q}_t|(N_\mu+1) - \sum_{x=1}^{\mu} x\lambda p_o^x - 1)p\big)p_e$. ∎

It is easy to see $P_{det}^{\mathsf{N}} \le P_{det}^{\mathsf{B}}$ from Eqs. (3) and (1). We simply present the derivation for $|\mathcal{Q}_t| = 1$ here for space constraints. Since $Np \approx \bar{n}$ and $(\sum_{x=1}^{\mu} x\lambda p_o^x + 1 + (N_\mu - \sum_{x=1}^{\mu} x\lambda p_o^x)p)$ approaches $\bar{n}$ when μ increases, we have $\sum_{x=1}^{\mu} x\lambda p_o^x + (N_\mu - \sum_{x=1}^{\mu} x\lambda p_o^x)p \le (Np-1)$ and thus $P_{det}^{\mathsf{N}} \le P_{det}^{\mathsf{B}}$.

Since each data bucket is broadcasted following BSC with probability p_b and NSC with probability $1-p_b$, we have the following theorem regarding the detection probability of HSC without giving a straightforward proof.

THEOREM *1: By* HSC*, the network owner can detect* $\mathcal{M}$ *dropping any particular data bucket satisfying* $\mathcal{Q}_t$ *with probability*

$$P_{det}^{\mathsf{H}} = 1 - p_b(1 - P_{det}^{\mathsf{B}}) - (1-p_b)(1 - P_{det}^{\mathsf{N}}) \ . \tag{6}$$

In practice, multiple buckets may be dropped by $\mathcal{M}$. $\mathcal{M}$'s misbehavior can be detected if the network owner can detect at least one dropped bucket. The following theorem is about the overall detection probability of HSC.

THEOREM *2: The network owner can detect* $\mathcal{M}$ *if* $\mathcal{M}$ *drops at least one data buckets with probability*

$$P_{det}^{\mathsf{HSC}} = \frac{1 - (1 - p_d P_{det}^{\mathsf{H}})^{\Delta_t}}{1 - (1-p_d)^{\Delta_t}} \ , \tag{7}$$

where P_{det}^{H} *is given in Eq. (6).*

Proof: P_{det}^{HSC} can be interpreted as probability that given that $\mathcal{M}$ has dropped at least one bucket, at least one of them can be detected by the network owner. Denote by X and Y the number of buckets dropped by $\mathcal{M}$ and the number of dropped packets detected by the network owner, respectively. We have

$$\begin{aligned} P_{det}^{\mathsf{HSC}} &= \Pr(Y>0|X>0) \\ &= \frac{\Pr(Y>0, X>0)}{1-\Pr(X=0)} \\ &= \frac{\sum_{x=1}^{\Delta_t}(1-\Pr(Y=0|X=x))\Pr(X=x)}{1-(1-p_d)^{\Delta_t}} \\ &= \frac{\sum_{x=1}^{\Delta_t}(1-P_{det}^{\mathsf{H}})^x \binom{\Delta_t}{x} p_d^x (1-p_d)^{\Delta_t - x}}{1-(1-p_d)^{\Delta_t}} \\ &= \frac{1-(1-p_d P_{det}^{\mathsf{H}})^{\Delta_t}}{1-(1-p_d)^{\Delta_t}} \ . \end{aligned}$$

∎

From Theorem 2, we can see that the detection probability of HSC increases as P_{det}^{H} increases, which coincides with the intuition. In addition, the larger Δ_t is, the more buckets are dropped by $\mathcal{M}$ for given p_d, and thus the higher the detection probability.

Communication Cost

Lemma 3 and lemma 4 are about the communication costs of BSC and NSC, respectively.

LEMMA *3: Assuming that the average number of hops from any node in the cell to* $\mathcal{M}$ *is* $\bar{L}$*, the communication cost incurred by* BSC *in epoch* t *of one cell is given by*

$$\bar{\mathrm{T}}^{\mathsf{BSC}} = N(l+g)\bar{m}_t(1+\bar{L}gpp_e) \ , \tag{8}$$

where $\bar{m}_t = \lceil N\big(1-(1-p)^g\big)\rceil$ *is the number of nodes sending out the data index.*

Proof: $\bar{\mathrm{T}}^{\mathsf{BSC}}$ consists of $\bar{\mathrm{T}}_B^{\mathsf{BSC}}$, the energy consumption for each node broadcasting its own data index, and $\bar{\mathrm{T}}_U^{\mathsf{BSC}}$, the energy consumption for each node sending its stored data indexes to the master node.

We first derive $\bar{\mathrm{T}}_B^{\mathsf{BSC}}$. Recall that each data index is of g bits. Assuming that each node ID is of l bits, i.e., $N \le 2^l$, each hop-wise transmission and reception of a data index and the corresponding node ID involve $l+g$ bits. Then the number of nodes sending out the data indexes can be approximated as

$$\bar{m}_t = \lceil N\big(1-(1-p)^g\big)\rceil \ , \tag{9}$$

where $1-(1-p)^g$ is the probability that one node generates at least one data buckets in epoch t and thus need broadcast data index. We assume the simplest broadcast technique in which each node receives and transmits a broadcast message once. Then we have

$$\bar{\mathrm{T}}_B^{\mathsf{BSC}} = \bar{m}_t N(l+g) \ , \tag{10}$$

Now we estimate $\bar{\mathrm{T}}_U^{\mathsf{BSC}}$. Assume that the average number of hops from any node in cell id_{cell} to $\mathcal{M}$ is $\bar{L}$. Each node on the average generates gp data buckets and receives $\bar{m}_t$ data indexes, each of which is embedded with equal probability p_e into each of its non-empty buckets. We thus have

$$\bar{\mathrm{T}}_U^{\mathsf{BSC}} = Ngpp_e\bar{m}_t\bar{L}(l+g) \ . \tag{11}$$

Combining Eq. (10) and Eq. (11), we have $\bar{\mathrm{T}}^{\mathsf{BSC}} = N(l+g)\bar{m}_t(1+\bar{L}gpp_e)$. ∎

From Lemma 3, we can see that the communication cost of BSC is proportional to the number of sensor nodes per cell.

LEMMA *4: Assuming that the average number of hops from any node in the cell to* $\mathcal{M}$ *is* $\bar{L}$*, the communication cost incurred by* NSC *in epoch* t *of one cell is given by*

$$\bar{\mathrm{T}}^{\mathsf{NSC}} = \bar{m}_t(l+g+|\mu|)(1+\sum_{x=1}^{\mu-1} x\lambda) + \bar{m}_t\bar{L}(l+g)N_\mu gpp_e \ , \tag{12}$$

where $\bar{m}_t$ *is given in Eq. (9).*

Proof: Similar to $\bar{\mathrm{T}}^{\mathsf{BSC}}$, $\bar{\mathrm{T}}^{\mathsf{NSC}}$ consists of two parts: $\bar{\mathrm{T}}_B^{\mathsf{NSC}}$, the energy consumption of μ-hop broadcasting each data index, and $\bar{\mathrm{T}}_U^{\mathsf{NSC}}$, the energy consumption of each node sending its stored data indexes to the master node.

We first estimate $\bar{\mathrm{T}}_B^{\mathsf{NSC}}$. Each data index (in the form of $\langle \mu, i, \mathsf{V}_{i,t}\rangle$) is transmitted up to μ hops. So each hop-wise transmission and reception involve $l+g+|\mu|$ bits, where $|\mu|$ is the length of μ. The cost incurred by broadcasting one data index is thus $(l+g+|\mu|)(1+\sum_{x=1}^{\mu-1} x\lambda)$ bits. Similar to BSC, there are $\bar{m}_t$ (given in Eq. (9)) nodes which have generated data indexes. We thus have $\bar{\mathrm{T}}_B^{\mathsf{NSC}} = \bar{m}_t(l+g+|\mu|)(1+\sum_{x=1}^{\mu-1} x\lambda)$.

We now estimate $\bar{\mathrm{T}}_U^{\mathsf{NSC}}$. Each of the $\bar{m}_t$ data indexes (in the form of $\langle i, \mathsf{V}_{i,t}\rangle$) has approximately $N_\mu gpp_e$ copies, each of which is sent along a $\bar{L}$-hop path to $\mathcal{M}$. Therefore, we can

compute $\bar{\mathrm{T}}_U^{\mathsf{NSC}} = \bar{m}_t \bar{L}(l+g) N_\mu g p p_e$. Combining Eq. (10) and Eq. (11), we have $\bar{\mathrm{T}}^{\mathsf{NSC}} = \bar{m}_t(l+g+|\mu|)(1+\sum_{x=1}^{\mu-1} x\lambda) + \bar{m}_t \bar{L}(l+g) N_\mu g p p_e$. ■

Comparing Eq. (12) with Eq. (8), we can see that although each data index broadcasted in NSC is longer than that in BSC due to the addition of hop information μ, the communication cost of NSC is lower than that of BSC as long as $1+\sum_{x=1}^{\mu-1} x\lambda \ll N$, which is the number of nodes that need rebroadcast each received data index.

In addition, we have the following theorem regarding the communication cost incurred by HSC without giving the straightforward proof.

THEOREM 3: *The communication cost incurred by* HSC *in epoch t of one cell is given by*

$$\bar{\mathrm{T}}^{\mathsf{HSC}} = p_b \bar{\mathrm{T}}^{\mathsf{BSC}} + (1-p_b)\bar{\mathrm{T}}^{\mathsf{NSC}}, \tag{13}$$

where p_b is the probability that one data bucket is processed following BSC*, and $\bar{\mathrm{T}}^{\mathsf{BSC}}$ and $\bar{\mathrm{T}}^{\mathsf{NSC}}$ are given in Eq. (8) and Eq. (12), respectively.*

2) Analysis of TC: **Detection Probability**

Now we analyze P_{det}^{TC}, the probability that the network owner can detect $\mathcal{M}'s$ misbehavior with temporal crosscheck. To enable the theoretical analysis, we assume that the network owner issues queries in each epoch with equal probability p_q. We consider multiple queries in each epoch as a single query and assume that $\mathcal{M}$ drops either none or all of data in the query result.

Let $\mathcal{I}$ be the time difference in epochs between the oldest data index and the data index to be inserted. Given the FIFO strategy, $\mathcal{I}$ is also the longest time in unit of epochs that a data index can stay in a buffer. $\mathcal{I}$ is apparently a random variable depending on p_T and $\mathcal{L}$ and no smaller than $\mathcal{L}$.

Without loss of generality, we assume that $\mathcal{M}$ consecutively drops $\alpha' \le \alpha$ query results, starting from epoch t, and will faithfully answer the $(\alpha'+1)$-*th* query. Assume that $\mathsf{V}_{i,t}$, i.e., the data index of S_i at epoch t, has been embedded into the buffer, which happens with probability p_T. Let $\mathcal{T}_{\alpha'+1}$ denote the number of epochs covered by these $\alpha'+1$ queries. We have $\mathcal{T}_{\alpha'+1} \ge \alpha'+1$, where the equation holds when the queries are made in consecutive epochs. Since $\mathsf{V}_{i,t}$ will stay in S_i's buffer for $\mathcal{I}$ epochs, the network owner can detect $\mathcal{M}$'s misbehavior as long as $\mathcal{T}_{\alpha'+1} \le \mathcal{I}$. It is also obvious that the larger α', the higher probability that $\mathsf{V}_{i,t}$ is removed from the buffer, the smaller probability that $\mathcal{M}$ will be detected, and vice versa. Therefore, hereafter we shall consider the worst-case scenario, i.e., $\alpha' = \alpha$ in favor of $\mathcal{M}$.

We first analyze $P_{det,i}^{\mathsf{TC}}$, i.e., the probability that the network owner can detect $\mathcal{M}$ has dropped S_i's data. Let $\mathcal{Q}_t$ and $\mathcal{Q}_{t'}$ denote the sets of bucket IDs queried in epochs t and t', respectively, where $t' = t+\mathcal{T}_{\alpha+1}-1$. Then the probability that S_i generates data buckets in $\mathcal{Q}_t$ and also embeds $\mathsf{V}_{i,t}$ into its buffer is $p_T \cdot \left(1-(1-p)^{|\mathcal{Q}_t|}\right)$. In epoch t', $\mathsf{V}_{i,t}$ is still stored in the buffer with the probability $\Pr(\mathcal{T}_{\alpha+1} \le \mathcal{I})$ and S_i generates data buckets in $\mathcal{Q}_{t'}$ with probability $1-(1-p)^{|\mathcal{Q}_{t'}|}$. So the probability that the network owner finds that S_i did have data in epoch t satisfying $\mathcal{Q}_t$ can be derived as follows.

$$P_{det,i}^{\mathsf{TC}} = p_T \cdot \left(1-(1-p)^{|\mathcal{Q}_t|}\right) \cdot \left(1-(1-p)^{|\mathcal{Q}_{t'}|}\right) \cdot \Pr(\mathcal{T}_{\alpha+1} \le \mathcal{I}) \ , \tag{14}$$

where

$$\begin{aligned} &\Pr(\mathcal{T}_{\alpha+1} \le \mathcal{I}) \\ &= \sum_{Y=\mathcal{L}}^{\infty} \Pr(\mathcal{T}_{\alpha+1} \le \mathcal{I} | \mathcal{I} = Y) \cdot \Pr(\mathcal{I} = Y) \\ &= \sum_{Y=\mathcal{L}}^{\infty} \sum_{X=\alpha+1}^{Y} \Pr(\mathcal{T}_{\alpha+1} = X) \cdot \Pr(\mathcal{I} = Y) \ . \end{aligned} \tag{15}$$

Since $\mathcal{I} \ge \mathcal{L}$ and thus $Y \ge \mathcal{L}$, we can get $\Pr(\mathcal{I}=Y) = p_T \binom{Y-1}{\mathcal{L}-1} p_T^{\mathcal{L}-1} (1-p_T)^{(Y-\mathcal{L})}$. We also have $\Pr(\mathcal{T}_{\alpha+1}=X) = p_q \binom{X-1}{\alpha} p_q^{\alpha} (1-p_q)^{(X-\alpha-1)}$.

Finally we have the following theorem regarding the detection probability of TC without giving a straightforward proof.

THEOREM 4: *With* TC*, assuming that $\mathcal{M}$ consecutively drops all the data for no more than α epochs, the network owner can detect $\mathcal{M}$ with probability*

$$P_{det}^{\mathsf{TC}} = 1 - (1 - P_{det,i}^{\mathsf{TC}})^N \ , \tag{16}$$

where $P_{det,i}^{\mathsf{TC}}$ is given in Eq. (14).

Communication Cost

The following theorem is about the communication cost incurred by TC.

THEOREM 5: *The communication cost incurred by* TC *in epoch t of one cell is given by*

$$\bar{\mathrm{T}}^{\mathsf{TC}} = \bar{L}\mathcal{L}(g+\gamma)\bar{m}_t g p \ , \tag{17}$$

where $\bar{m}_t$ and p are given in Eq. (9) and Eq. (2), respectively.

Proof: Recall that $\bar{L}$ denotes the average number of hops from any node to $\mathcal{M}$, and each buffer is of $\mathcal{L}(g+\gamma)$ bits. Similar to spatial crosscheck technique, there are $\bar{m}_t$ nodes generating data at each epoch, each of which on average generates gp data buckets, where $\bar{m}_t$ and p are given in Eq. (9) and Eq. (2), respectively. We thus have $\bar{\mathrm{T}}^{\mathsf{TC}} = \bar{L}\mathcal{L}(g+\gamma)\bar{m}_t g p$. ■

3) Analysis of STC*:* Since STC is a simple combination of HSC and TC, we have the following theorem regarding the detection probability of STC without giving the straightforward proof.

THEOREM 6: *Under* STC*, the network owner can detect $\mathcal{M}$'s misbehavior with probability*

$$P_{det}^{\mathsf{STC}} = 1 - (1 - P_{det}^{\mathsf{HSC}})(1 - P_{det}^{\mathsf{TC}}) \ , \tag{18}$$

where P_{det}^{HSC} and P_{det}^{TC} are given in Eq. (7) and Eq. (16), respectively.

Likewise, we have the following theorem regarding the total communication cost of STC.

THEOREM 7: The communication cost incurred by STC in each epoch of one cell is given by

$$\bar{\mathrm{T}}^{\mathsf{STC}} = \bar{\mathrm{T}}^{\mathsf{HSC}} + \bar{\mathrm{T}}^{\mathsf{TC}} \ , \tag{19}$$

where $\bar{\mathrm{T}}^{\mathsf{HSC}}$ and $\bar{\mathrm{T}}^{\mathsf{TC}}$ is given in Eq. (13) and Eq. (17), respectively.

4) Analysis of the Encoding Technique: We have briefly discussed the encoding technique, denoted by ENCODE, in Section I. Here we present more details about it to facilitate its comparison with our techniques. The main idea of ENCODE is to let each sensor node return some unforgeable proof for each empty bucket. Under this technique, for each bucket $j \in [1,g]$, node S_i generates an *encoding number* as $num(i,j,t) = H_{L_e}(i||j||t||K_{i,t})$, where $H_{L_e}(\cdot)$ denotes

a good hash function of L_e bits and $K_{i,t}$ is the epoch key of S_i introduced in Section III. Each $num(i,j,t)$ needs to be sent to the master node $\mathcal{M}$. Given a query $\mathcal{Q}_t$, $\mathcal{M}$ generates a condensed certificate as

$$CERT_t = H_{L_c}(||_{S_i \in \mathcal{U}_t, j \in \mathcal{V}_{i,t}} H(i||j||t||num(i,j,t))),$$

where $H_{L_c}(\cdot)$ denotes a good hash function of L_c bits, $\mathcal{U}_t \subseteq \{S_i\}_{i=1}^N$ denotes the set of nodes having at least one data buckets not satisfying $\mathcal{Q}_t$, and $\mathcal{V}_{i,t}$ denotes the empty bucket IDs within the range of $\mathcal{Q}_t$. $\mathcal{M}$ need return the query result and $CERT_t$ to the network owner. Since the network owner knows all the epoch keys, it can recompute $CERT_t'$ based on the query result and the query $\mathcal{Q}_t$, and then compares $CERT_t'$ with $CERT_t$. If the results match, it considers $\mathcal{M}$ legitimate and malicious otherwise.

Assume that $\mathcal{M}$ drops some data from some nodes. Since $\mathcal{M}$ does not know their epoch keys, it can only escape the detection by guessing encoding numbers for the dropped buckets or $CERT_t$. It is assumed in [5] that L_c is sufficiently large such that the probability of a successful guess, i.e., $1/2^{L_c}$, is negligible. The only option left for $\mathcal{M}$ is thus to guess correct encoding numbers. The average number of buckets satisfying $\mathcal{Q}_t$ is $|\mathcal{Q}_t|Np$, where p is given in Eq. (2), and each bucket is dropped with equal probability p_d. Therefore, $\mathcal{M}$ need guess $N_g = |\mathcal{Q}_t|p_d Np$ encoding numbers, which succeeds with probability $\frac{1}{2^{L_e N_g}}$. The network owner can thus detect $\mathcal{M}$ with probability $P_{det}^{\mathsf{ENCODE}} = 1 - \frac{1}{2^{L_e N_g}} \approx 1$, and ENCODE can be considered as a deterministic method.

The communication cost of ENCODE, denoted by $\bar{\mathrm{T}}^{\mathsf{ENCODE}}$, comes from transmitting the encoding numbers of sensor nodes to $\mathcal{M}$. Since each node S_i has $g(1-p)$ empty buckets on the average, the same number of encoding numbers need be generated and sent to $\mathcal{M}$. We thus have

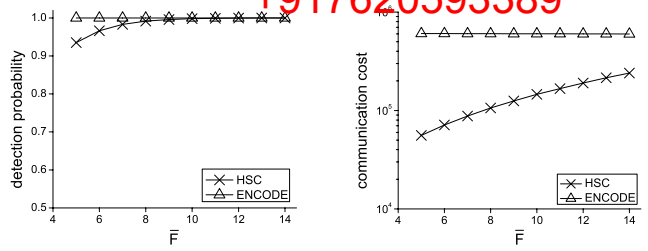
$$\bar{\mathrm{T}}^{\mathsf{ENCODE}} = \bar{L} L_e N g (1-p) \ . \tag{20}$$

We can see that the communication cost of ENCODE is proportional to the number of empty buckets, which makes it inefficient for event-driven sensor networks, where events occur infrequently and most nodes have no data to report.

B. Numeric Results

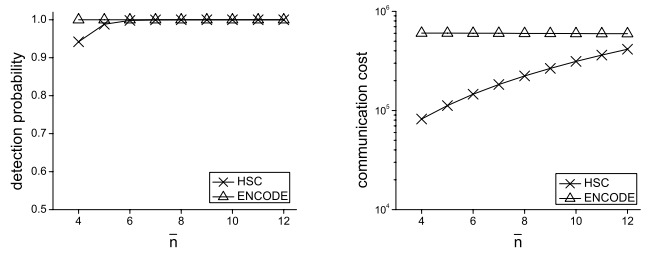
We assume that N sensor nodes are uniformly distributed in a cell of regular shape and that the master node is at the center of the cell. According to [14], the average number of hops from any node to its master node is $\bar{L} = O(\sqrt{N})$. To enable quantitative evaluations, we assume that $\bar{L} \approx \lceil \sqrt{N}/2 + 1 \rceil$.

Fig. 1 shows the impact of $\bar{F}$ on HSC and ENCODE, where the communication cost is in $\log_{10}$ scale. Here we assume that the network owner queries two buckets in epoch t, i.e., $|\mathcal{Q}_t| = 2$. The following parameters are used: $N = 500$; $g = 10$; $\bar{n} = 6$; $p_d = 0.8$; for HSC, $p_e = 0.6$, $\mu = 3$, $\lambda = 5$, and $p_b = 0.1$; for ENCODE, $L_e = 10$. Fig. 1(a) shows that ENCODE has a detection probability close to 1, while the detection probability of HSC is very close to that ENCODE, say higher than 0.95 when $\bar{F}$ is larger than 6. In addition, the detection probability of HSC increases as $\bar{F}$ increases. The reason is that the larger $\bar{F}$, the more non-empty buckets each node has, and the more nodes embed other nodes' data indexes. Fig. 1(b) shows that the communication cost of HSC is lower than that of ENCODE when $\bar{F}$ is small. This coincides with the intuition since the communication costs of HSC and ENCODE are determined by the numbers of non-empty buckets and empty ones, respectively, which makes HSC more suitable for event-driven WSN where events occur infrequently. In addition, the communication costs of HSC increases as $\bar{F}$ increases because more nodes will have data indexes to broadcast.



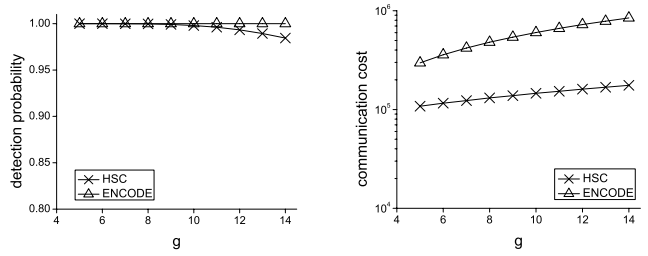
(a) detection probability (b) communication cost

Fig. 1. Impact of $\bar{F}$.



(a) detection probability (b) communication cost

Fig. 2. Impact of $\bar{n}$.

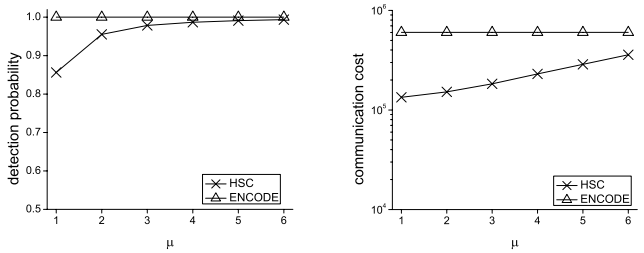


(a) detection probability (b) communication cost

Fig. 3. Impact of g.

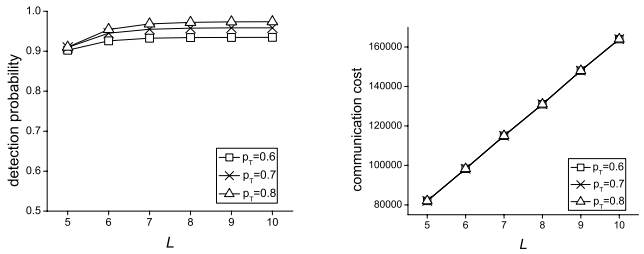
Fig. 2 shows the impact of $\bar{n}$ on HSC and ENCODE, where $\bar{F} = 10$ and other parameters are the same as those in Fig. 1. Similar to $\bar{F}$, the larger the $\bar{n}$ is, the higher the detection probability and communication cost of HSC, while ENCODE is almost unaffected. The reason is that as $\bar{n}$ increases, the number of non-empty buckets increases, so does the detection probability and communication cost of HSC. At the same time, when $\bar{F}$ is small, the number of empty buckets is much larger than that of non-empty ones, so the communication cost of ENCODE is relatively unaffected.

Fig. 3 shows the impact of g, where $\bar{F} = 10$, $\bar{n} = 6$, and



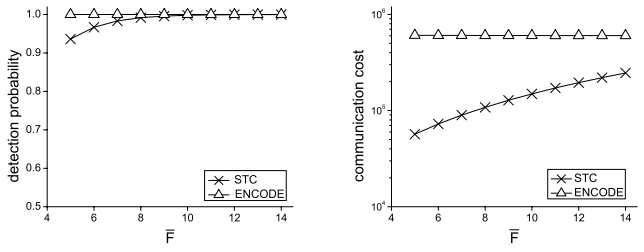
(a) detection probability (b) communication cost

Fig. 4. Impact of μ on HSC.



(a) detection probability (b) communication cost

Fig. 5. Impact of p_T and $\mathcal{L}$ on TC.



(a) detection probability (b) communication cost

Fig. 6. STC vs. ENCODE.

other parameters are the same as those in Fig. 1. We can see from Fig. 3(a) that as g increases, the detection probability of HSC slightly decreases. The reason is that given other parameters fixed, the number of nodes have data in the same buckets decreases as g increases, so does the number of nodes satisfying a given query. HSC relies on other nodes that have qualified data and have also embedded the data index about the missing data bucket, so the detection probability slightly decreases as g increases. In addition, Fig. 3(b) shows that the communication costs of HSC and ENCODE both increase as g increases, which can be easily understood.

Fig. 4 shows the impact of μ on HSC. ENCODE is not affected by μ, whose performance is plotted for reference only. We can see from Fig. 4(a) and Fig. 4(b) that the larger the μ is, the higher the detection probability and communication cost of HSC. This coincides with the intuition, since under NSC, the number of copies of a given data index embedded increases as μ increases, leading to the increase in the detection probability and communication cost of HSC.

Fig. 5 shows the impact of the buffer length $\mathcal{L}$ and p_T on TC. Here we assume that the network owner queries data in each epoch with probability $p_q = 0.8$, $\alpha = 4$, and $|\mathcal{Q}_t| = 2$. Other parameters are $\bar{F} = 25$, $\bar{n} = 10$, and $g = 10$. We can see from Fig. 5(a) that the detection probability of TC slightly increases as $\mathcal{L}$ increases for the same p_T. In addition, the larger the p_T is, the higher the detection probability. These coincide with the intuition, as the higher the p_T is, the more likely a data index is inserted into the buffer, and the larger $\mathcal{L}$ is, the longer the inserted data index will stay in the buffer. Both cases will lead to the increase in the detection probability of TC. In addition, Fig. 5(b) shows that the communication cost of TC is independent of p_T and only increases as $\mathcal{L}$ increases, which is also anticipated.

Fig. 6 compares the performance of STC and ENCODE, where the following parameters are used: $\bar{n} = 6$; for HSC, $\lambda = 5$, $\mu = 3$, and $p_b = 0.1$; for TC, $\mathcal{L} = 3$, $p_T = 0.6$, $\alpha = 4$. We can see from Fig. 6(a) and Fig. 6(b) that STC achieves comparable detection probability but incurs much lower communication overhead in comparison with ENCODE when $\bar{F}$ is small. These results highlight the suitability of STC for event-driven WSNs.

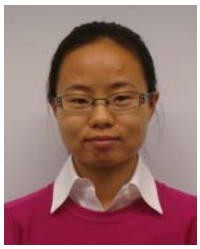
VI. Conclusion

In this paper, we presented a novel spatiotemporal technique to secure range queries in event-driven two-tier sensor networks. Our technique can prevent compromised master nodes from reading hosted data and also achieves high query efficiency. In addition, our technique allows the network owner to verify the authenticity and completeness of any query result. Compared with prior work, our technique can achieve a comparable detection probability with much lower communication overhead in even-driven WSNs. The efficacy and efficiency of our technique are confirmed by detailed evaluations.

References

[1] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," in *IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.
[2] P. Desnoyers, D. Ganesan, and P. Shenoy, "TSAR: a two tier sensor storage architecture using interval skip graphs," in *Proc. ACM SenSys'05*, San Diego, California, USA, Nov. 2005, pp. 39–50.
[3] M. Shao, S. Zhu, W. Zhang, and G. Cao, "pDCS: security and privacy support for data-centric sensor networks," in *Proc. IEEE INFOCOM'07*, Anchorage, Alaska, USA, May 2007, pp. 1298–1306.
[4] N. Subramanian, C. Yang, and W. Zhang, "Securing distributed data storage and retrieval in sensor networks," in *IEEE PerCom'07*, White Plains, NY, Mar. 2007.
[5] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in sensor networks," in *Proc. IEEE INFOCOM'08*, Phoenix, AZ, Apr. 2008, pp. 46–50.
[6] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," in *IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.
[7] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, "The tenet architecture for tiered sensor networks," in *Proc. ACM SenSys'06*, Boulder, Colorado, USA, Oct. 2006, pp. 153–166.
[8] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. ACM SenSys'03*, Los Angeles, California, USA, Nov. 2003, pp. 63–75.
[9] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD'02*, Madison, Wisconsin, June 2002, pp. 216–227.
[10] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proc. VLDB'04*, Toronto, Canada, Aug. 2004, pp. 720–731.

[11] P. Rogaway, M. Bellare, and J. Black, "OCB: a block-cipher mode of operation for efficient authenticated encryption," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 365–403, Aug. 2003.
[12] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," in *Proc. ACM MobiHoc'09*, New Orleans, LA, May 2009, pp. 197–206.
[13] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "SWATT: software-based attestation for embedded devices," in *Proc IEEE S&P'04*, Berkeley, CA, USA, May 2004.
[14] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.



Jing Shi received the B.E. in Communication Engineering and the M.E. in Communication and Information System from Huazhong University of Science and Technology, China, in 2003 and 2006, respectively, and the Ph.D. in Electrical and Computer Engineering from New Jersey Institute of Technology in 2010. Her research interests are network and distributed system security, wireless networking, and mobile computing.

Rui Zhang received the B.E. in Communication Engineering and the M.E. in Communication and Information System from Huazhong University of Science and Technology, China, in 2001 and 2005, respectively. He is currently a Ph.D. student in School of Electrical, Computer, and Energy Engineering at Arizona State University. His research interests are network and distributed system security, wireless networking, and mobile computing. He was a software engineer in UTStarcom Shenzhen R&D center from 2005 to 2007.

Yanchao Zhang received the B.E. in Computer Science and Technology from Nanjing University of Posts and Telecommunications, China, in 1999, the M.E. in Computer Science and Technology from Beijing University of Posts and Telecommunications, China, in 2002, and the Ph.D. in Electrical and Computer Engineering from the University of Florida in 2006. He is currently as an Associate Professor in School of Electrical, Computer, and Energy Engineering at Arizona State University. Before ASU, he was an Assistant Professor of Electrical and Computer Engineering at New Jersey Institute of Technology from 2006 to 2010. His primary research interests are network and distributed system security, wireless networking, and mobile computing. He is an Associate Editor of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, a Feature Editor of IEEE WIRELESS COMMUNICATIONS, and a Guest Editor of IEEE WIRELESS COMMUNICATIONS Special Issue on Security and Privacy in Emerging Wireless Networks. He was a TPC Co-Chair of Communication and Information System Security Symposium, IEEE GLOBECOM 2010, and a Workshop Co-Chair of ACM MSWiM'10. He received the NSF CAREER Award in 2009.