

# Secure Range Queries in Tiered Sensor Networks

Jing Shi, Rui Zhang, and Yanchao Zhang  
 Department of Electrical and Computer Engineering  
 New Jersey Institute of Technology  
 Email: {js39,rz23,yczhang}@njit.edu

**Abstract**—We envision a two-tier sensor network which consists of 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 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 crosscheck 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.

## I. INTRODUCTION

Most future large-scale wireless sensor networks (WSNs) will follow a two-tier architecture [1]. As shown in Fig. 1, the lower tier comprises a large number of resource-constrained sensor nodes, while the upper tier contains fewer relatively resource-rich *master nodes*. 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. This two-tier architecture is indispensable for increasing network capacity and scalability, reducing system complexity, and prolonging network lifetime [1]–[3].

There are two ways for the network owner to access the data generated by sensor nodes in the two-tier WSN. First, sensor nodes send their data to nearby master nodes which in turn forward the data immediately along an upper-tier multi-hop path to the network owner. This approach requires a stable always-on communication connection from some master node(s) to the network owner, which is impossible or prohibitive to maintain in extreme and hazardous environments such as oceans, animal habitats, and battlefields. This approach may also consume unnecessary energy of master nodes if the network owner is only interested in a small portion of the potentially huge amount of data produced over time. The second approach takes advantage of the rapid progress in storage technology and views the WSN as a storage system [1]. In particular, it is now possible to equip each of the relatively fewer master nodes with several gigabytes of NAND flash storage for a few tens

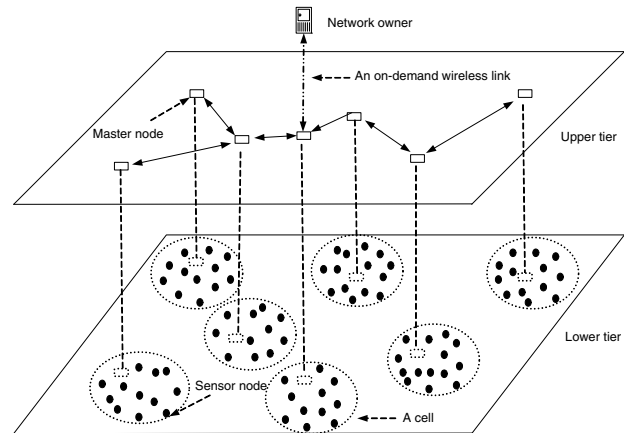


Fig. 1. An abstract two-tier sensor network architecture.

of dollars, though it may remain economically prohibitive to furnish each sensor node with large flash storage. Master nodes then collect data from nearby 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). As [2]–[5], this paper is concerned with the second approach. A WSN may need to support many types of data queries [3]. In this paper, we focus on supporting range queries which are an important and common type of queries in WSNs and ask for data within a certain range [6], [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 military and homeland security scenarios 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 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 at the same time. In addition, the adversary may instruct compromised master nodes to return juggled and/or incomplete data in response to range 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 such as military actions. 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.

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 [8], [9] to store encrypted data at master nodes and also ensure range-query efficiency. If the encryption algorithm is an OCB-like authenticated encryption primitive [10], 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. More details of their scheme can be found in Section VI. 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 nearby 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.

This paper, for the first time in literature, investigates techniques to secure range queries in event-driven two-tier WSNs. We also use the bucketing technique [8], [9] to strike a balance between data confidentiality and query efficiency. Our major contribution is a novel spatiotemporal crosscheck 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 network and adversary models. Section III illustrates how to perform range queries over encrypted data based on the bucketing technique [8], [9]. The spatial and temporal crosscheck techniques are then presented in Sections IV and V, respectively. Subsequently, we thoroughly evaluate the proposed techniques in Section VI and conclude in Section VII.

## II. NETWORK AND ADVERSARY MODELS

### A. Network Model

We assume a large-scale two-tier WSN as shown in Fig. 1 and introduced in Section I. The network region is partitioned into physical *cells*, each containing a master node in charge of sensor nodes in that cell. We follow the conventional assumption that master and sensor nodes know their geographic locations. In addition, they are assumed to know which cell they are in. These localization requirements are fundamental for WSN functionalities and can be fulfilled by many existing techniques. Depending on concrete applications, the cells of two neighboring master nodes may overlap, in which case sensor nodes in the overlapping region are affiliated with both master nodes.

Master nodes are assumed to have abundant resources in storage, energy (a solar panel and/or heavy-duty rechargeable batteries), and computation; they also form a multi-hop wireless mesh network via relatively long-range, high-rate 802.11-like radios. In contrast, sensor nodes are constrained in storage, energy, and computation; they also communicate with neighbor nodes via low-power, low-rate, short-distance radios like CC1100 or 802.15.4. Unlike conventional WSNs, we do not assume a stable communication connection to the external network owner. Instead, the network owner can query data by an on-demand wireless link (e.g., a satellite link) connected to some master node(s). To prevent storage overflow of master nodes, mobile sinks [11] can also be periodically (e.g., quarterly) dispatched to collect data and empty the storage of master nodes.

### B. Adversary Model

To disturb WSN operations, the adversary may launch arbitrary attacks such as passive eavesdropping, bogus-message injection, and physical-layer jamming, for which we resort to existing elegant defenses such as [12]–[20]. This paper focuses on thwarting attacks on secure range queries in tiered WSNs.

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 which have more important roles, the adversary will be more 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 investigation on the impact of compromised sensor nodes is left as future work.

## 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 approach.

We assume that time is divided into *epochs* and that sensor and master nodes are loosely synchronized. Each epoch consists of three phases of different length. In the longest *detection* phase, each sensor node performs sensing tasks. 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 events generated in that phase. 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  $c$  whose weight attribute is between 100 and 120 pounds.” Extending the proposed techniques to multi-dimensional range queries will be our future work.

Without loss of generality, we subsequently focus on a cell  $C$  consisting of  $N$  sensor nodes, denoted by  $\{S_i\}_{i=1}^N$ , and a master node, denoted by  $\mathcal{M}$ . Each node  $S_i$  is assumed to be 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.

The simplest solution to data confidentiality is letting each  $S_i$  use  $K_{i,t}$  to encrypt all the data it produced during each epoch  $t$  as a whole. Since  $\mathcal{M}$  does not know  $K_{i,t}$ , it cannot read the encrypted data. Although providing strong data confidentiality, this method does not support efficient range queries. In particular,  $\mathcal{M}$  has to return all the ciphertexts it collected for epoch  $t$  in response to a query from the network owner. Since the network owner can derive all the epoch keys, it can decrypt the ciphertexts to locate the data of interest if any. This is obviously very inefficient because the network owner may only have interest in the data falling in a small range.

We adopt the bucketing technique [8], [9] to strike a balance between data confidentiality and query efficiency. The bucketing technique partitions the queriable attribute domain into  $\varpi \geq 1$  consecutive non-overlapping regions (buckets), sequentially numbered from 1 to  $\varpi$ , and the partitioning rule is 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  $\mathcal{M}$  at

the end 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 [10] 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 cell ID  $c$  and epoch number  $t$  to  $\mathcal{M}$ . Upon receiving the query  $\langle c, t, \mathcal{Q}_t \rangle$ ,  $\mathcal{M}$  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  $\varpi$ . 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 to [9] 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 the issue is dealt with in the following two sections.

#### IV. SPATIAL CROSSCHECK

In this section, we present a novel spatial crosscheck technique to allow the network owner to verify the completeness of query results. We still consider cell  $C$  which consists of master node  $\mathcal{M}$  and sensor nodes  $\{S_i\}_{i=1}^N$ . We assume that  $S_i$  generates data of bucket  $j$  during epoch  $t$  with probability  $p_{i,j,t}, 1 \leq j \leq \varpi$ .

To enable quantitative analysis, we consider the following game between the network owner and  $\mathcal{M}$ . The network owner issues a sequence of range queries in the query phase of epoch  $t$  which together compose a set  $\mathcal{Q}_t \subseteq \{1, \dots, \varpi\}$  of bucket IDs. Note that the bucket IDs in  $\mathcal{Q}_t$  may not be continuous. We assume that  $\mathcal{M}$  omits data from different sensor nodes with equal probability  $p_d$ . If the network owner can eventually detect the misbehavior of  $\mathcal{M}$ , it wins the game and loses it otherwise.

The key idea of spatial crosscheck is to embed some relationships among data generated by nodes  $\{S_i\}_{i=1}^N$  during each epoch. 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. In the following, we detail two versions of spatial crosscheck: a broadcast-based one and a neighbor-based one. We will also analyze their performance 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.
- **$\mathcal{C}_T$ -communication cost:**  $\mathcal{C}_T$  is defined as the total communication energy consumption in bits for completeness verification of query results from cell  $C$ . Here we assume the same energy consumption to transmit and receive each bit across each hop.

#### A. Broadcast-based Spatial Crosscheck

The main idea is that before submitting its data to  $\mathcal{M}$ , every sensor node broadcasts information about its data (if any) within cell  $C$  during the reporting phase of each epoch. The broadcasted information is a vector with  $\varpi$  bits and called a *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  $V_i^t$ . For example, if  $\varpi = 8$  and  $V_{i,t} = 10101000$ , then  $S_i$  only detected data for buckets 1|3|5 during epoch  $t$ . Every node with data for submission should set a timer to the estimated longest end-to-end message transmission time in cell  $C$  to allow enough time for receiving other nodes' data indexes. Then it embeds every received data index along with the corresponding node ID into every data bucket it wants to submit. Finally, it sends the encrypted buckets to  $\mathcal{M}$ . For example, assume that  $S_i$  has 3|2|1 data items in buckets 1|3|5, respectively, and received data indexes from nodes  $\{S_{j_l}\}_{l=1}^k$ .  $S_i$  finally sends the following message to  $\mathcal{M}$  during the reporting phase.

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

Since the data of different sensor nodes can crosscheck each other, we name this technique broadcast-based spatial crosscheck.

The effectiveness of this technique can be easily seen through the following example. Assume that the network owner sends a query only for bucket 5; nodes  $\mathcal{W} \subset \{S_i\}_{i=1}^N$  have data in bucket 5, where  $S_i \in \mathcal{W}$ ; and  $\mathcal{M}$  drops the data bucket of  $S_i$ . Since  $\langle S_i, V_{i,t} \rangle$  is embedded in bucket 5 of every other node in  $\mathcal{W} \setminus \{S_i\}$ , 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  $\langle S_i, V_{i,t} \rangle$  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  $\langle S_i, V_{i,t} \rangle$  may reach the network owner, but one copy is enough. Therefore, each node does not store each received data index into all its buckets so as to save the energy consumption

incurred by transmitting these data indexes to  $\mathcal{M}$ . 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 store each received data index in each of its buckets (if any) with equal probability  $p_e$ . In what follows, we analyze the impact of such probabilistic embedding on the misbehavior-detection probability and the communication cost.

1) **Detection probability:** Let  $\Delta_t \subseteq \{S_i\}_{i=1}^N$  be the set of nodes which generated data buckets with IDs in the query  $\mathcal{Q}_t$ . The cardinality of  $\Delta_t$  can be approximated by

$$|\Delta_t| = \lceil \sum_{i=1}^N (1 - \prod_{j \in \mathcal{Q}_t} (1 - p_{i,j,t})) \rceil. \quad (1)$$

Assume that  $S_i \in \Delta_t$ . If  $\mathcal{M}$  drops the data from  $S_i$ , its misbehavior can be detected as long as at least one copy of  $S_i$ 's data index  $V_{i,t}$  can reach the network owner. There are, however, two cases in which the network owner cannot receive  $V_{i,t}$ . First,  $\Delta_t$  does not contain any node other than  $S_i$ . Second, no node in  $\Delta_t \setminus \{S_i\}$  stored  $V_{i,t}$  in any of its buckets satisfying  $\mathcal{Q}_t$ .

Now we analyze the probability  $P_{det,i}$  that the network owner can detect  $\mathcal{M}$ 's dropping of  $S_i$ 's data satisfying  $\mathcal{Q}_t$ . Consider node  $S_w \in \Delta_t \setminus \{S_i\}$ . The probability that the network owner can receive  $V_{i,t}$  from  $S_w$  is

$$P_w = (1 - p_d)(1 - \prod_{j \in \mathcal{Q}_t} (1 - p_{w,j,t} p_e)). \quad (2)$$

We thus have

$$P_{det,i} = 1 - \prod_{S_w \in \Delta_t \setminus \{S_i\}} (1 - P_w). \quad (3)$$

To enable the theoretical analysis, we assume that  $\{P_{det,i}\}_{S_i \in \Delta_t}$  are the same. Let  $D_x$  denote the event that  $\mathcal{M}$  drops the data of  $x$  nodes and  $U_x$  be the event that this behavior is not detected by the network owner. We can derive the detection probability as

$$\begin{aligned} P_{det}^{bro} &= \frac{1 - \Pr(D_0) - \sum_{i=1}^{|\Delta_t|} \Pr(U_i | D_i) \Pr(D_i)}{1 - \Pr(D_0)} \\ &= \frac{1}{1 - \Pr(D_0)} \cdot \left\{ 1 - (1 - p_d)^{|\Delta_t|} \right. \\ &\quad - \binom{|\Delta_t|}{1} p_d (1 - P_{det,i}) \cdot (1 - p_d)^{|\Delta_t| - 1} \\ &\quad - \binom{|\Delta_t|}{2} p_d^2 (1 - P_{det,i})^2 \cdot (1 - p_d)^{|\Delta_t| - 2} \\ &\quad \dots \\ &\quad \left. - \binom{|\Delta_t|}{|\Delta_t|} p_d^{|\Delta_t|} (1 - P_{det,i})^{|\Delta_t|} \right\} \\ &= \frac{1 - (1 - p_d P_{det,i})^{|\Delta_t|}}{1 - (1 - p_d)^{|\Delta_t|}}. \end{aligned} \quad (4)$$

2) *Communication Cost*: Now we derive the communication cost  $\mathcal{C}_t^{bro}$  incurred by broadcast-based spatial crosscheck in epoch  $t$ . In other words, 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.  $\mathcal{C}_t^{bro}$  thus consists of two parts:  $\mathcal{C}_{t,B}^{bro}$ , the energy consumption for each node broadcasting its own data index, and  $\mathcal{C}_{t,U}^{bro}$ , the energy consumption for each node sending its stored data indexes to the master node.

We first derive  $\mathcal{C}_{t,B}^{bro}$ . Recall that each data index is of  $\varpi$  bits. Assuming that each node ID is of  $l$  bits, i.e.,  $N \leq 2^l$ , each hop-wise transmission and reception of a data index and the corresponding node ID involve  $l + \varpi$  bits. We also assume the simplest broadcast technique in which each node receives and transmits a broadcast message once. Then we have

$$\mathcal{C}_{t,B}^{bro} = N(l + \varpi) \sum_{i=1}^N \left[ 1 - \prod_{j=1}^{\varpi} (1 - p_{i,j,t}) \right], \quad (5)$$

where  $1 - \prod_{j=1}^{\varpi} (1 - p_{i,j,t})$  is the probability that  $S_i$  detected data in at least one bucket in epoch  $t$  and thus has a data index to broadcast. Here we assume that compromised sensor nodes do not broadcast their data indexes if any.

Now we estimate  $\mathcal{C}_{t,U}^{bro}$ . Assume that the average number of hops from any node in cell  $C$  to  $\mathcal{M}$  is  $\bar{L}$ . Each node on the average receives  $\bar{n}_t = \sum_{i=1}^N [1 - \prod_{j=1}^{\varpi} (1 - p_{i,j,t})]$  data indexes, each embedded with equal probability  $p_e$  in any bucket it has. Assuming that a compromised node simply ignores all the received data indexes, we thus have

$$\mathcal{C}_{t,U}^{bro} = \bar{L}(l + \varpi) \sum_{i=1}^N \sum_{j=1}^{\varpi} p_{i,j,t} p_e \bar{n}_t. \quad (6)$$

Therefore, we have

$$\begin{aligned} \mathcal{C}_t^{bro} &= \mathcal{C}_{t,B}^{bro} + \mathcal{C}_{t,U}^{bro} \\ &= (l + \varpi) \sum_{i=1}^N \left[ 1 - \prod_{j=1}^{\varpi} (1 - p_{i,j,t}) \right] \cdot \\ &\quad [N + \bar{L} \sum_{i=1}^N \sum_{j=1}^{\varpi} p_{i,j,t} p_e]. \end{aligned} \quad (7)$$

### B. Neighbor-based Spatial Crosscheck

Broadcast-based spatial crosscheck requires each node to broadcast its data index to all the other nodes in cell  $C$  and store each received data index with probability  $p_e$  in each bucket it has. It may unfortunately cause unnecessary communication overhead. For example, assume that  $\mathcal{M}$  drops the data of node  $S_i$  satisfying the query from the network owner. Among the nodes that received and stored  $V_{i,t}$ , only those which also generated data satisfying the query are helpful for the network owner to catch  $\mathcal{M}$ .

We propose the neighbor-based spatial crosscheck technique to reduce the communication overhead. This technique

is motivated by the essential nature of event-driven sensor networks: if one node detects some 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 more likely that its nearby nodes also produced qualified data. It is thus sufficient to let the data of neighboring sensor nodes crosscheck each other.

Let us see some more details. We assume that each node has  $\lambda$  neighboring nodes within its transmission range. In addition, when any node detects some event, each of its neighbors is assumed to detect the same event with equal probability  $p_o$ . Take node  $S_i$  as an example which detected data in epoch  $t$  and locally broadcasts  $\langle \mu, i, V_{i,t} \rangle$  during the reporting phase, where  $\mu \geq 1$  is a system parameter specifying the maximum number of hops for which the message can be transmitted. Each of its  $\lambda$  neighbors ignores the message if it has no bucket specified in  $V_{i,t}$ . Otherwise, it inserts  $\langle i, V_{i,t} \rangle$  into any such bucket and then broadcasts  $\langle \mu-1, i, V_{i,t} \rangle$  to its own neighbors if  $\mu-1 > 0$ . To summarize, each node should locally broadcast not only its own data index but also other received data indexes which intersect its own index and have not been transmitted beyond  $\mu$  hops. A node may receive the same data index multiple times from different neighbors, in which case it should only process the first version.

1) *Detection probability*: Now we analyze the detection probability  $P_{det}^{nei}$  of neighbor-based spatial crosscheck. Due to space constraints, we consider a simple scenario in which an event was first detected at node  $S_i$  in epoch  $t$  which generated data in bucket  $j \in [1, \varpi]$ . According to our event-propagation model above, the  $x$ -hop ( $\forall x \geq 1$ ) neighbors of  $S_i$  each detected the same event and generated data in bucket  $j$  with  $p_o^x$ . We call  $S_i$  and all its  $x$ -hop neighbors *correlated nodes* of the corresponding event. Assume that the network owner issues a query  $\mathcal{Q}_t$  containing just bucket  $j$ , for which only  $S_i$  and its correlated nodes generated data. It is worth noting that this single-event assumption is actually in favor of the malicious  $\mathcal{M}$ . In particular, if  $S_i$  detected multiple events and  $\mathcal{Q}_t$  also contains multiple bucket IDs,  $\mathcal{M}$  can be considered malicious if the network owner can detect its dropping of data related to any of the queried bucket IDs. Therefore,  $P_{det}^{nei}$  to be analyzed may be an underestimation of the detection probability.

We first estimate the number of nodes with data satisfying  $\mathcal{Q}_t$ . Assuming that sensor nodes are uniformly distributed, the average number of  $x$ -hop neighbors of  $S_i$  is then  $x\lambda$ . Let  $\chi$  denote the number such that  $p_o^\chi \ll 1$ . We thus have

$$\mathcal{N}_t = 1 + \sum_{x=1}^{\chi} x\lambda p_o^x \approx 1 + \sum_{x=1}^{\infty} x\lambda p_o^x = 1 + \frac{\lambda p_o}{(1 - p_o)^2}, \quad (8)$$

which holds if  $\chi$  is sufficiently large.

Assuming that  $\mathcal{M}$  drops the data of any of the  $\mathcal{N}_t$  nodes, we first estimate the probability  $P'_{det}$  that the network owner can detect it.  $P'_{det}$  is obviously the probability that at least one copy of the data index of that node can reach the network owner. The average number of nodes which stored a given

data index is  $\lfloor \sum_{x=1}^{\mu} x \lambda p_o^x \rfloor$ . Since  $\mathcal{M}$  drops the data of any node with equal probability  $p_d$ , we can easily derive that

$$P'_{det} = 1 - p_d^{\lfloor \sum_{x=1}^{\mu} x \lambda p_o^x \rfloor}. \quad (9)$$

Similar to the derivation of Eq. (4), we can finally have

$$\begin{aligned} P_{det}^{nei} &= \frac{1 - \Pr(D_0) - \sum_{i=1}^{|\mathcal{N}_t|} \Pr(U_i|D_i)\Pr(D_i)}{1 - \Pr(D_0)} \\ &= \frac{1}{1 - \Pr(D_0)} \cdot \left\{ 1 - (1 - p_d)^{|\mathcal{N}_t|} \right. \\ &\quad - \binom{|\mathcal{N}_t|}{1} p_d (1 - P'_{det}) \cdot (1 - p_d)^{|\mathcal{N}_t|-1} \\ &\quad - \binom{|\mathcal{N}_t|}{2} p_d^2 (1 - P'_{det})^2 \cdot (1 - p_d)^{|\mathcal{N}_t|-2} \\ &\quad \dots \\ &\quad \left. - \binom{|\mathcal{N}_t|}{|\mathcal{N}_t|} p_d^{|\mathcal{N}_t|} (1 - P'_{det})^{|\mathcal{N}_t|} \right\} \\ &= \frac{1 - (1 - p_d P'_{det})^{|\mathcal{N}_t|}}{1 - (1 - p_d)^{|\mathcal{N}_t|}}. \end{aligned} \quad (10)$$

2) *Communication cost*: Now we derive the communication cost  $C_t^{nei}$  incurred by neighbor-based spatial crosscheck in epoch  $t$ . As in Section IV-B2, we only consider the cost associated with transmitting data indexes among sensor nodes and from sensor nodes to the master node.  $C_t^{nei}$  also consists of two parts:  $C_{t,B}^{nei}$ , the energy consumption for the  $\mu$ -hop broadcasting of each data index, and  $C_{t,U}^{nei}$ , the energy consumption for each node sending its stored data indexes to the master node.

We first estimate  $C_{t,B}^{nei}$ . Each data index will be transmitted up to  $\mu$  hops in a probabilistic fashion, and each hop-wise transmission and reception involve  $l + \varpi + |\mu|$  bits. The total cost incurred by one data index is thus  $(l + \varpi + |\mu|)(1 + \sum_{x=1}^{\mu-1} x \lambda p_o^x)$  bits. Since we totally have  $\mathcal{N}_t$  data indexes as given in Eq. (8), we thus have

$$C_{t,B}^{nei} = \mathcal{N}_t (l + \varpi + |\mu|) \left( 1 + \sum_{x=1}^{\mu-1} x \lambda p_o^x \right). \quad (11)$$

We now estimate  $C_{t,U}^{nei}$ . Since there are approximately  $\sum_{x=1}^{\mu} x \lambda p_o^x$  copies of each of the  $\mathcal{N}_t$  data indexes, each copy will be sent along a  $\bar{L}$ -hop path to  $\mathcal{M}$ . Therefore, we can compute

$$C_{t,U}^{nei} = \mathcal{N}_t \bar{L} (l + \varpi + |\mu|) \sum_{x=1}^{\mu} x \lambda p_o^x. \quad (12)$$

It follows that

$$\begin{aligned} C_t^{nei} &= C_{t,B}^{nei} + C_{t,U}^{nei} \\ &= \mathcal{N}_t (l + \varpi + |\mu|) \left[ 1 - \lambda \mu p_o^\mu + (\bar{L} + 1) \sum_{x=1}^{\mu} x \lambda p_o^x \right]. \end{aligned} \quad (13)$$

## V. TEMPORAL CROSSCHECK

Our spatial crosscheck technique allows the network owner to detect  $\mathcal{M}$ 's misbehavior with very high probability as long as it can receive one data index of any node whose data were dropped.  $\mathcal{M}$ , however, can escape the detection by dropping all the qualified data. In this section, we propose a temporal crosscheck technique to deal with such attacks as a complement to the spatial crosscheck technique.

We assume that the network owner makes queries in each epoch with equal probability  $p_q$ . To be consistent with our analysis of spatial crosscheck, 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  $\alpha \geq 1$  be a system parameter, specifying the maximum number of consecutive droppings the network owner can tolerate. In particular, if the network owner does not receive any data for  $\alpha$  consecutive queries which spans no less than  $\alpha$  epochs, it will doubt 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 [21]. To avoid raising the suspicion,  $\mathcal{M}$  will not consecutively drop data for more than  $\alpha$  queries.

### A. Scheme Description

To enable temporal crosscheck, we require each node to maintain a fixed-length FIFO buffer of  $\mathcal{L}(\varpi + \eta)$  bits, where  $\eta$  denote the length of an epoch ID which can roll over. The buffer can thus hold at most  $\mathcal{L}$  pairs of data index and epoch ID. At the end of each epoch, each node, say  $S_i$ , inserts its own index 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. We denote by  $\mathcal{I}$  the time difference in epochs between the oldest data index and the data index to be inserted. Given the FIFO buffer management strategy,  $\mathcal{I}$  is also the longest time in 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 less than  $\mathcal{L}$ . If  $S_i$  has any data for submission to  $\mathcal{M}$ , it should also send its buffer which should be secured using OCB-like authenticated encryption primitive [10] with its epoch key. If  $\mathcal{M}$  sends the data of  $S_i$  in response to a query, it should as well send the buffer of  $S_i$  which can be decrypted and authenticated by the network owner.

We now explain the intuition behind temporal crosscheck. Assume that  $S_i$  detects some data in epoch  $t$  which satisfies the query made in epoch  $t$  and inserted  $V_{i,t}$  in its buffer. We also assume that  $\mathcal{M}$  consecutively drops  $\alpha' \leq \alpha$  query results from epoch  $t$ . If the network owner's next query arrives before  $V_{i,t}$  is moved out of the buffer and  $S_i$  has data satisfying the 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  $t$ .

### B. Detection Probability

Now we analyze the probability  $P_{det}^{tem}$  that the network owner can detect  $\mathcal{M}$ 's misbehavior with temporal crosscheck. Without loss of generality, we assume that  $\mathcal{M}$  consecutively drops  $\alpha' \leq \alpha$  query results, starting from epoch  $t$ , and will

faithfully answer the  $(\alpha' + 1)$ th query. Let  $\mathcal{T}_{\alpha'+1}$  denote the number of epochs covered by these  $\alpha' + 1$  queries. We have  $\mathcal{T}_{\alpha'+1} \geq \alpha' + 1$ , where the equation holds when the queries are made in consecutive epochs. Continue our example above. Since  $V_{i,t}$  will stay in  $S_i$ 's buffer for  $\mathcal{I}$  epochs, the network owner can detect  $\mathcal{M}$ 's misbehavior as long as  $\mathcal{T}_{\alpha'+1} \leq \mathcal{I}$ . It is also obvious that the smaller  $\alpha'$ , the shorter  $\mathcal{T}_{\alpha'+1}$ , the faster  $\mathcal{M}$  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 consider the effect of node  $S_i$ 's buffer on  $P_{det}$ . Let  $\mathcal{Q}_t$  and  $\mathcal{Q}_{t'}$  denote the set of bucket IDs queried in epochs  $t$  and  $t'$ , respectively, where  $t' = t + \mathcal{T}_{\alpha+1} - 1$ . Then 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.

$$P_{det,i}^{tem} = p_T \cdot \left(1 - \prod_{j \in \mathcal{Q}_t} (1 - p_{i,j,t})\right) \cdot \left(1 - \prod_{j \in \mathcal{Q}_{t'}} (1 - p_{i,j,t'})\right) \cdot \Pr(\mathcal{T}_{\alpha+1} \leq \mathcal{I}), \quad (14)$$

where

$$\begin{aligned} \Pr(\mathcal{T}_{\alpha+1} \leq \mathcal{I}) &= \sum_{Y=\mathcal{L}}^{\infty} \Pr(\mathcal{T}_{\alpha+1} \leq \mathcal{I} | \mathcal{I} = Y) \cdot \Pr(\mathcal{I} = Y) \\ &= \sum_{Y=\mathcal{L}}^{\infty} \sum_{X=\alpha+1}^Y \Pr(\mathcal{T}_{\alpha+1} = X) \cdot \Pr(\mathcal{I} = Y), \\ \Pr(\mathcal{I} = Y)_{Y \geq \mathcal{L}} &= p_T \cdot \binom{Y-1}{\mathcal{L}-1} p_T^{\mathcal{L}-1} (1-p_T)^{(Y-\mathcal{L})}, \end{aligned}$$

and

$$\Pr(\mathcal{T}_{\alpha+1} = X) = p_q \cdot \binom{X-1}{\alpha} p_q^{\alpha} (1-p_q)^{(X-\alpha-1)}.$$

Finally, we can derive the detection probability under temporal crosscheck as

$$P_{det}^{tem} = 1 - \prod_{i=1}^N (1 - P_{det,i}^{tem}). \quad (15)$$

### C. Communication Cost

We define the communication cost  $\mathcal{C}_{T,t}^{tem}$  of temporal crosscheck as the energy consumed for relaying the temporal buffers of non-compromised nodes to  $\mathcal{M}$  in each epoch  $t$ . Recall that  $\bar{L}$  denote the average number of hops from any node to  $\mathcal{M}$ . We thus have

$$\mathcal{C}_{T,t}^{tem} = \bar{L} \mathcal{L} (\varpi + \eta) \sum_{i=1}^N \left(1 - \prod_{j=1}^{\varpi} (1 - p_{i,j,t})\right). \quad (16)$$

## VI. PERFORMANCE EVALUATION

In this section, we thoroughly evaluate the performance of our spatiotemporal crosscheck approach and also compare it with the encoding technique proposed in [5]. For clarity, we first brief the encoding technique and then analyze its detection probability and communication cost using our models. Then we present detailed performance evaluation results obtained using MATLAB.

### A. Review of the Encoding Technique

The basic idea of the encoding technique has been discussed in Section I. Here we present some more details. In this technique, for each bucket  $j \in [1, \varpi]$ , node  $S_i$  generates a so-called 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

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

where  $H_{L_c}(\cdot)$  denotes a good hash function of  $L_c$  bits and  $\mathcal{U}_t \subseteq \{S_i\}_{i=1}^N$  denotes the set of nodes having no data satisfying  $\mathcal{Q}_t$ .  $\mathcal{M}$  need return the query result and  $CERT_t$  to the network owner. Since the network owner knows all the epoch keys, it then recomputes  $CERT_t$  based on the query result and the query  $\mathcal{Q}_t$ . If the result matches what it received, 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 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 for  $\mathcal{M}$  is thus to guess correct encoding numbers. The average number of buckets satisfying  $\mathcal{Q}_t$  is  $\sum_{i=1}^N \sum_{j \in \mathcal{Q}_t} p_{i,j,t}$ , and each is dropped with equal probability  $p_d$ . Therefore,  $\mathcal{M}$  need guess  $N_{guess} = p_d \sum_{i=1}^N \sum_{j \in \mathcal{Q}_t} p_{i,j,t}$  encoding numbers, which succeeds with probability  $\frac{1}{2^{L_e N_{guess}}}$ . The network owner can detect  $\mathcal{M}$  with probability  $P_{det}^{enc} = 1 - \frac{1}{2^{L_e N_{guess}}}$ .

The communication cost  $\mathcal{C}_t^{enc}$  of the encoding technique comes from transmitting the encoding numbers of sensor nodes to  $\mathcal{M}$ . Since each node  $S_i$  has  $\sum_{j=1}^{\varpi} (1 - p_{i,j,t})$  empty buckets on the average, the same number of encoding numbers need be generated and sent to  $\mathcal{M}$ . We thus have

$$\mathcal{C}_t^{enc} = \bar{L} L_e \sum_{i=1}^N \sum_{j=1}^{\varpi} (1 - p_{i,j,t}). \quad (17)$$

Sheng and Li [5] also give an elegant solution to the optimal length of  $L_e$ . For fair comparisons, we thus just analyze  $P_{det}^{enc}$  and  $\mathcal{C}_t^{enc}$  of their encoding technique under our network and adversary models.

### B. Performance of Spatial and Temporal Crosscheck Techniques

We assume that there are  $N$  sensor nodes uniformly distributed in each cell of regular shape. Each master node is assumed to be at the center of its cell. According to [22], the average number of hops from any node to its master node is  $\mathcal{L} = O(\sqrt{N})$ . To enable quantitative evaluations, we assume that  $\bar{L} \approx \lceil \sqrt{N}/2 + 1 \rceil$ . In addition, the queryable attribute domain is assumed to be partitioned into  $\varpi = 10$  buckets. We also assume that the data resulting from one event fall into the same single bucket. For ease of presentation, we denote broadcast-based spatial crosscheck by BSC, neighbor-based

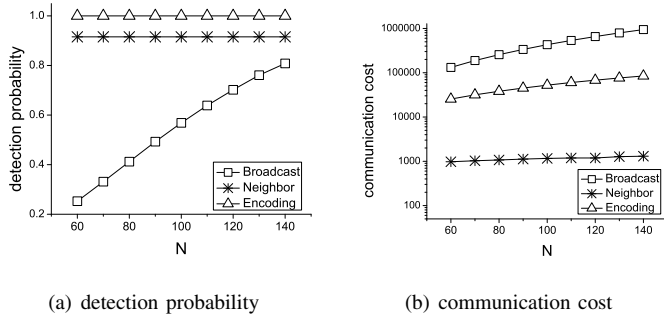


Fig. 2. Impact of  $N$ , the number of nodes in one cell.

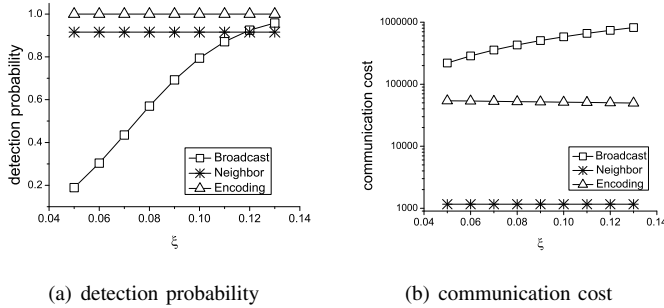


Fig. 3. Impact of event-detection probabilities.

spatial crosscheck by NSC, temporal crosscheck by TC, and the encoding technique by ENCODE.

For each bucket  $j \in [1, N]$ , we further assume that each node generates data in bucket  $j$  during epoch  $t$  with the same probability, i.e.,  $p_{i_1, j, t} = p_{i_2, j, t}, \forall i_1 \neq i_2 \in [1, N]$ . Assume that the events associated with bucket  $j$  occur according to a poisson distribution with parameter  $\xi(1 + 0.1r)$ , where  $\xi \geq 0$  is a constant and  $r \in [0, 1]$  is a random number. Then we have  $p_{i, j, t} = 1 - e^{-\xi(1+0.1r)}$ . Such assumptions are just used to facilitate the performance evaluation.

Fig. 2 shows the impact of  $N$  on the detection probability and the communication cost of BSC, NSC, and ENCODE, where the communication cost is in log10 scale. Here we assume that the network owner queries one bucket in epoch  $t$  for simplicity. The following parameters are used:  $p_d = 0.5$ ;  $\xi = 0.08$ ; for broadcast-based spatial crosscheck,  $p_e = 0.8$ ; for neighbor-based spatial crosscheck,  $p_o = 0.25$ ,  $\mu = 3$ , and  $\lambda = 6$ ; for the encoding technique,  $L_e = 10$ . As we can see, the detection probability of BSC is more affected by  $N$  than those of NSC and ENCODE. In addition, NSC consumes much less energy than ENCODE at the slight sacrifice in the detection probability.

Fig. 3 shows the impact of  $p_{i, j, t}$  or  $\xi$  on BSC, NSC, and ENCODE, where  $N = 100$  and other parameters are the same as in Fig. 2. With the increase of  $\xi$  and thus  $p_{i, j, t}$ , the communication cost of ENCODE decreases because the number of empty buckets decreases, while the communication cost of BSC will increase because more nodes have data indexes to broadcast. The communication cost of NSC is not

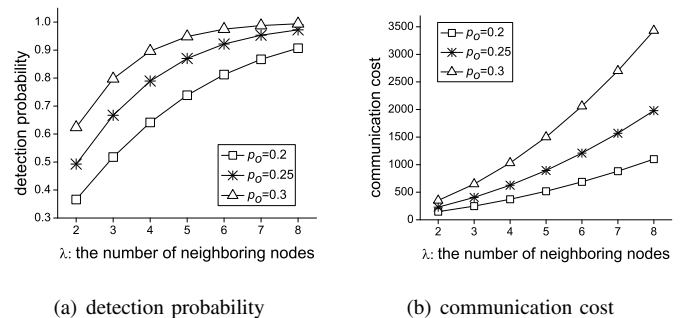


Fig. 4. Impact of  $p_o$  and  $\lambda$  on neighbor-based spatial crosscheck (NSC)

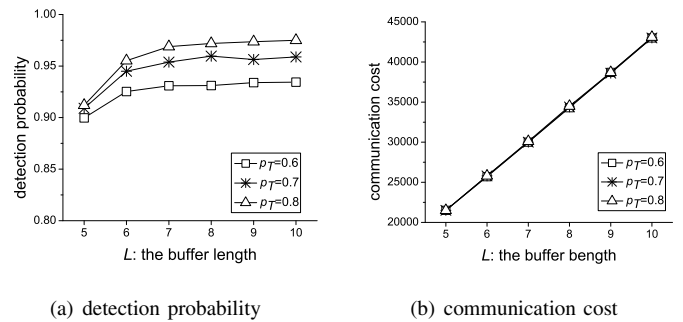


Fig. 5. Impact of  $p_T$  and  $L$  on temporal crosscheck (TC)

affected by  $\xi$ . We can see that as long as  $\xi$  is sufficiently small and thus events occur infrequently, NSC always consumes much less energy than ENCODE.

Fig. 4 shows the impact of  $p_o$  and  $\lambda$  on NSC, where  $\mu = 4$ . Recall that the average number of copies of a given data index is  $\sum_{x=1}^{\mu} \lambda x p_o^x$ , which will increase with  $\lambda$  and  $p_o$ . Therefore, the detection probability of NSC will increase with  $\lambda$  and  $p_o$ , as is evidenced in Fig. 4(a). This comes with the increase of the communication cost, as more copies of a given data index will be transmitted. The result in Fig. 4 is thus of no surprise.

Fig. 5 shows the impact of the buffer length  $L$  and  $p_T$ , the probability that a data index is inserted into the buffer. Here we assume that the network owner queries data in each epoch with probability  $p_q = 0.8$  and each query contains three buckets. To illustrate Fig. 5, consider, as an example, node  $S_i$  whose data satisfying the query was dropped by  $\mathcal{M}$  in epoch  $t$ . The larger  $p_T$ , the more probable  $S_i$  inserts its data index at epoch  $t$  into its buffer; the larger  $L$ , the longer time the data index stays in the buffer for opportunistic transmission to the network owner. Therefore, the detection probability  $P_{det}^{tem}$  of TC increases with both  $L$  and  $p_T$ , as can be seen in Fig. 5(a). The communication cost of TC, as defined in Eq. (16), is only related to and increases with  $L$ . It is thus of no surprise to observe the result in Fig. 5(b).

### C. Spatiotemporal Crosscheck vs. Encoding

Now we compare the proposed spatiotemporal crosscheck technique with the encoding technique. Since previous results have demonstrated the significant advantages of NSC over

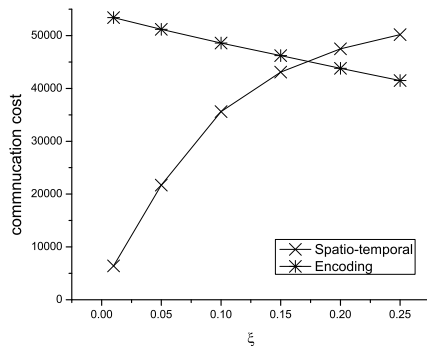


Fig. 6. Communication cost: spatiotemporal vs. encoding

BSC, we only consider the integration of NSC and TC. It follows that  $P_{det} = 1 - (1 - P_{det}^{nei})(1 - P_{det}^{tem})$ , where  $P_{det}^{nei}$  and  $P_{det}^{tem}$  are defined in Eq. (10) and Eq. (15), respectively. Likewise, the total communication cost of spatiotemporal crosscheck is  $C_t = C_t^{nei} + C_t^{tem}$ , where  $C_t^{nei}$  and  $C_t^{tem}$  are defined in Eq. (13) and Eq. (16), respectively.

The detection probabilities of both spatiotemporal crosscheck and the encoding technique are comparably close to 1 (both larger than 0.95). Fig. 6 compares their communication costs. As  $\xi$  grows, the communication cost of spatiotemporal increases because more information enabling crosscheck need be transmitted, while the communication cost of the encoding technique decreases because the number of empty buckets decreases, and vice versa. In general, as long as  $\xi$  is sufficiently small, spatiotemporal crosscheck incurs much less communication overhead than the encoding technique.

In summary, the above results prove that the proposed spatiotemporal crosscheck approach is well suited for sensor networks with sparse events, while the encoding technique is more suitable for sensor networks in which events occur either continuously or frequently.

## VII. CONCLUSION

In this paper, we presented a novel spatiotemporal crosscheck technique to ensure secure range queries in event-driven two-tier sensor networks. Our technique prevents compromised master nodes from reading hosted data and also ensures high query efficiency. In addition, our technique allows the network owner to ensure that any query result contains complete and authentic data that originated from purported sources and have not been tampered with. The efficacy and efficiency of our technique are confirmed by detailed evaluations. As the future work, we intend to extend spatiotemporal crosscheck to support multi-dimensional range queries and other types of data queries.

## ACKNOWLEDGE

This work was supported in part by the US National Science Foundation under grant CNS-0716302.

## REFERENCES

- [1] 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 *ACM SenSys'06*, Boulder, Colorado, USA, Oct. 2006, pp. 153–166.
- [2] P. Desnoyers, D. Ganesan, and P. Shenoy, "TSAR: A two tier sensor storage architecture using interval skip graphs," in *ACM SenSys'05*, San Diego, California, USA, Nov. 2005, pp. 39–50.
- [3] Y. Diao, D. Ganesan, G. Mathur, and P. J. Shenoy, "Rethinking data management for storage-centric sensor networks," in *Third Biennial Conference on Innovative Data Systems Research (CIDR'07)*, Asilomar, CA, USA, Jan. 2007, pp. 22–31.
- [4] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," in *ACM MobiHoc'06*, Florence, Italy, May 2006, pp. 344–355.
- [5] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *IEEE INFOCOM'08*, Phoenix, AZ, April 2008, pp. 46–50.
- [6] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *ACM SenSys'03*, Los Angeles, California, USA, Nov. 2003, pp. 63–75.
- [7] 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.
- [8] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *ACM SIGMOD'02*, Madison, Wisconsin, 6 2002, pp. 216–227.
- [9] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Thirtieth international conference on Very large data bases (VLDB'04)*, Toronto, Canada, Aug. 2004, pp. 720–731.
- [10] 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, 2003.
- [11] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least privilege and privilege deprivation: towards tolerating mobile sink compromises in wireless sensor networks," in *ACM MobiHoc'05*, Urbana-Champaign, IL, USA, May 2005, pp. 378–389.
- [12] K. Ren, W. Lou, and Y. Zhang, "LEDS: Providing location-aware end-to-end data security in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 5, pp. 585–598, May 2008.
- [13] Y. Zhang, W. Liu, Y. Fang, and D. Wu, "Secure localization and authentication in ultra-wideband sensor networks," *IEEE J. Select. Areas Commun., Special Issue on UWB Wireless Communications - Theory and Applications*, vol. 24, no. 4, pp. 829–835, Apr. 2006.
- [14] Y. Zhou, Y. Zhang, and Y. Fang, "Access control in wireless sensor networks," *Ad Hoc Networks, Special Issue on Security in Ad Hoc and Sensor Networks*, vol. 5, no. 1, pp. 3–13, Jan. 2007.
- [15] Y. Zhou and Y. Fang, "A two-layer key establishment scheme for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1009–1020, September 2007.
- [16] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *ACM CCS'02*, Washington, DC, Nov. 2002, pp. 41–47.
- [17] W. Du, J. Deng, Y. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *ACM CCS*, Washington, DC, Oct. 2003, pp. 42–51.
- [18] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *ACM CCS'03*, Washington, DC, Oct. 2003, pp. 52–61.
- [19] L. Lazos and R. Poovendran, "SeRLoc: Secure range-independent localization for wireless sensor networks," in *ACM WiSe'04*, Philadelphia, PA, Oct. 2004, pp. 21–30.
- [20] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE J. Select. Areas Commun., Special Issue on Security in Wireless Ad Hoc Networks*, vol. 24, no. 2, pp. 247–260, Feb. 2006.
- [21] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "SWATT: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2004, pp. 272–282.
- [22] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.