Secure Continuous Aggregation in Wireless Sensor Networks

Lei Yu, *Member, IEEE,* Jianzhong Li, *Member, IEEE,* Siyao Cheng, Shuguang Xiong, Haiying Shen, *Member, IEEE*

Abstract—Continuous aggregation is usually required in many sensor applications to obtain the temporal variation information of aggregates. However, in a hostile environment, the adversary could fabricate false temporal variation patterns of the aggregates by manipulating a series of aggregation results through compromised nodes. Existing secure aggregation schemes conduct one individual verification for each aggregation result, which could incur great accumulative communication cost and negative impact on transmission scheduling for continuous aggregation. In this paper, we identify distinct design issues for protecting continuous in-network aggregation and propose a novel scheme to detect false temporal variation patterns. Compared with the existing schemes, our scheme greatly reduces the verification cost by checking only a small part of aggregation results to verify the correctness of the temporal variation patterns in a time window. A sampling-based approach is used to check the aggregation results, which enables our scheme independent of any particular in-network aggregation protocols as opposed to existing schemes. We also propose a series of security mechanisms to protect the sampling process. Both theoretical analysis and simulations show the effectiveness and efficiency of our scheme.

Index Terms—wireless sensor network, network security, secure continuous aggregation, sampling.

1 Introduction

In applications of wireless sensor networks(WSNs), the aggregations of sensed data, such as sum, average and predicate count, are very important for the users to get summarization information about the monitored area. Instead of collecting all sensor data [1]–[3] and computing aggregation results at the base station, in-network aggregation allows sensor readings to be aggregated by intermediate nodes, which efficiently reduces the communication overhead. Many in-network aggregation schemes have been proposed [4]–[7]. However, since WSNs are often deployed in an open and unattended environment, an adversary could undetectably take control of one or more sensor nodes and subvert correct in-network aggregations by manipulating the partial aggregation results or reporting arbitrary readings through compromised nodes.

In this paper we consider the security of continuous in-network aggregation in WSNs. In many WSN applications for environment monitoring, the users often need the temporal variation information in a series of aggregation results rather than an individual aggregation result. Thus, continuous aggregation of sensed data is usually desired. For a continuous aggregation query, a time interval, called *epoch*, is specified and the aggregation is evaluated in every epoch. The duration of every epoch specifies the amount of time sensor nodes wait before acquiring and transmitting each successive sample. Continuous aggregation is not merely for one-shot responses to sporadic queries. It helps the users to understand how the environment changes over time and track real-time measurements for trend analysis.

Because of the importance of temporal variation information of aggregation results, we focus on the attack against continuous in-network aggregation that the adversaries attempt to distort the real temporal variation pattern of the aggregate by disrupting a series of successive aggregation results. Figure 1 shows an example. The user is interested in a special variation pattern of average temperature shown in the shadowed box and could make critical decisions when the pattern is observed. The adversary can modify aggregation results

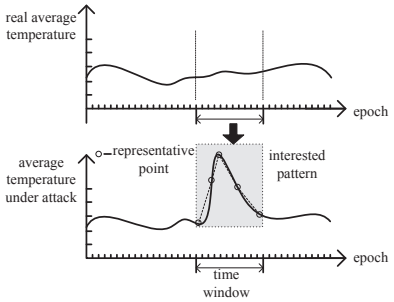


Fig. 1. The fabrication of the temporal variation pattern in a continuous aggregation

in the time window to fabricate the variation pattern that actually does not appear, which can lead to wrong decisions.

A number of secure aggregation schemes have been proposed [8]–[11]. SIA [8] addresses secure aggregation within the single aggregator network topology. A number of hierarchical secure aggregation schemes [9]–[11] are proposed for aggregation in tree network topology in which each node computes an intermediate aggregation result accounting for all sensing data of nodes in the sub-tree rooted at it. All these schemes aim to protect a single aggregation computation. Directly using these schemes in a continuous aggregation results in individual verification for every aggregation result in every epoch, which will incur a great communication cost especially for continuous aggregation having a long period or high frequency (i.e., small epoch). The additional communication caused by interactive procedures between the base station and sensor nodes for verification in every epoch also has a negative impact on the efficiency of transmission scheduling for a continuous data aggregation [12]. Besides, these schemes [8]–[10] also are tightly coupled with the tree topology and thus unable to work with various other in-network aggregation protocols [6], [7].

In this paper, we present an efficient scheme to detect false temporal variation patterns in a continuous aggregation. Our scheme verifies the correctness of the observed temporal variation pattern in a time window by checking only a small part of aggregation results termed *representative points*. The representative points are selected to capture the temporal variation pattern of the aggregate, as shown in Figure 1. Compared with the existing secure aggregation schemes, our scheme can considerably reduce the communication cost through selective verifications of aggregation results.

In our scheme, the correctness of representative points is checked by hypothesis testing techniques with samples from the WSN. While

- *A preliminary version of this paper appears in the proceedings of IEEE INFOCOM 2011.*
- *Lei Yu and Haiying Shen are with the Department of Electrical and Computer Engineering at Clemson University, SC, United States.*
- *Jianzhong Li and Siyao Cheng are with the Department of Computer Science and Technology, Harbin Institute of Technology, Heilongjiang, China.*
- *Shuguang Xiong is with Baidu Inc, Beijing, China.*
Email: {leiy, shenh}@clemson.edu, lijzh@hit.edu.cn, csyhit@126.com, n2xiong@gmail.com

Digital Object Indentifier 10.1109/TPDS.2013.63

providing nice security properties, the sampling-based approach only requires a part of nodes to be involved in the verification, and enables verification not to rely on any particular in-network aggregation protocol. To protect the sampling procedure, verifiable random sampling is proposed to protect the legitimacy of sampled nodes, and local authentication based on spatial correlation among sensor readings is proposed to protect the validity of sample readings. As a result, our scheme can effectively verify the temporal variation patterns for continuous aggregation, while being able to achieve low additional energy cost and work with various in-network aggregation protocols [6], [7]. We evaluate our scheme based on extensive experiments using a real trace of sensor readings. The experiment results show the efficiency and effectiveness of our scheme.

The rest of the paper is organized as follows. We present system models and design goals of our scheme in Section 3. We propose the details of our scheme in Section 4. We evaluate the performance and security of our scheme in Section 5. We present simulation results in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

Due to the importance of aggregation computation for WSN, secure aggregation has received great attention in recent years. A lot of secure aggregation schemes have been proposed [8]–[11], [13]–[17].

Wagner [13] evaluates the resilience of several aggregation functions against malicious nodes' contribution to the final computation results, and proposes to improve the resilience by truncation and trimming on the set of sensor readings as well as using robust estimators to compute aggregation. Under one single-aggregator network model, Przydatek et al. [8] propose an aggregate-commit-prove framework SIA to detect false aggregation results. In SIA, the base station generates a commitment to the collection of sensor readings by Merkle hash tree, and the home server verifies the results through reliable random sampling achieved by data commitment and interactive proofs with the base station. Yu et al. propose to directly use sampling to compute approximate aggregation results with provable guarantees that can always correctly answer aggregation queries [15].

A number of hierarchical secure schemes [9]–[11], [14], [16] have been proposed for in-network aggregation on tree topology, where each node computes an intermediate aggregation result accounting for the sensor readings of nodes in the sub-tree rooted at it. Hu and Evans [14] propose a secure aggregation scheme against one single malicious node in the network, in which each node checks the inconsistency of MACs from their children and grandchildren. Garofalakis et al. propose to combine cryptographic signatures and Flajolet-Martin sketch [18] to achieve verifiable count aggregation [16].

Several secure hierarchical aggregation schemes [9]–[11] follow an aggregation-commitment-attest framework. During the in-network aggregation, each node computes the hash as commitment over the input of its aggregation computation, intermediate results and data commitments from its children, and then sends the hash to its parent. Based on the commitments, interactive attest is performed between the base station and sensor nodes when aggregation completes. Yang et al. propose a secure hop-by-hop data aggregation protocol SDAP [9]. The tree topology is partitioned into multiple logical subtree groups, and sensor data are aggregated in every subtree separately to reduce the trust on high-level nodes. The groups returning outlier results are attested by checking the aggregation correctness along a random path. Chan et al. propose a provably secure hierarchical aggregation scheme SHIA [10]. In the attest phase of SHIA, the final commitment at the base station is broadcasted and each node checks that its own contribution was added into the aggregation by recomputing the final commitment with necessary information disseminated from its ancestor nodes. Frikken et al introduce modifications of SHIA which reduce original $O(d_{max} \log^2 n)$ communication per node to $O(d_{max} \log n)$ where d_{max} is the maximum degree of the aggregation tree and n is the number of nodes. Based on SHIA, Roy et al. propose a scheme to verify the histogram computation in order to securely estimate the median [17].

All these previous works address secure in-network aggregation within a snapshot query, so their approaches conduct verification for each single aggregation result. Unlike them, our work focuses on continuous in-network aggregation and aims to protect the temporal variation patterns of aggregation results. To protect continuous aggregation, previous approaches would conduct individual verification in every epoch and thus can incur a significant communication cost. In contrast, our approach only selectively verifies a small part of aggregation results in a time window.

3 Problem Statement

3.1 Network and Query Model

We assume a large-scale multi-hop WSN with a set of sensor nodes $S = \{s_1, \ldots, s_N\}$ and a trusted base station. The base station knows the total number of nodes $N = |S|$. All the nodes and the base station are loosely time synchronized with a secure time synchronization service [19], [20]. Each node has the same communication radius R_c.

We assume a continuous querying environment for WSNs. For a continuous aggregation query, the base station initially disseminates a query into the network, consisting of the epoch duration, the period of the aggregation query and a nonce number *nonce*. The nonce number is a cryptographically secure random number generated by the base station and used only once to to uniquely identify current query and prevent replay of old messages. The aggregation query period is divided into epochs. In each epoch, each node calculates a partial aggregate with its current sensor readings, and the base station obtains a final aggregation result.

Since most physical quantities in practical environments, such as temperature and humidity, usually change continuously, we assume that the duration of an epoch is small enough to reflect the continuous variation of the measured physical quantities with respect to time. As a result, the aggregation results would exhibit continuous variations. Such continuity actually can be characterized as that the difference between aggregation values at two successive epochs is bounded, i.e.,

$$|A(t) - A(t+1)| \leq \Lambda \tag{1}$$

where $A(t)$ is the true aggregation result in epoch t. Here Λ is determined by the characteristics of the observed physical quantities and the length of epoch duration.

The user is interested in some temporal variation pattern of the aggregation results which last for more than one epoch. As opposed to previous works [8]–[10], we assume the aggregation is performed over the network without specifying any particular in-network structure such as tree [5].

3.2 Security Assumptions

We assume that each sensor node has a unique identifier and shares a unique secret symmetric key with the base station. By pairwise key establishment schemes [21], [22], each node shares a pairwise key with each of its direct neighbors and two-hop neighbors. A broadcast authentication protocol such as μTESLA [23] exists such that any node can authenticate a message from the base station. We also assume that WSNs have a short safe bootstrapping phase right after network deployment [21], during which adversaries cannot successfully compromise any nodes.

3.3 Attack Model and Security Goal

We assume that the adversary can compromise multiple nodes and obtain the security information embedded in these nodes, but cannot

compromise the base station which is well-secured. We assume the Byzantine fault model where a compromised node is under the full control of the adversary and can misbehave in an arbitrary way. Multiple compromised nodes can collude to attack.

We focus on the attacks against in-network continuous aggregation, which aim to make the base station to accept a series of false aggregate results of which the temporal variation pattern deviates from the real one in a noticeable scale. The ways to carry out these attacks include providing false sensor readings and manipulating partial results via compromised nodes. However, no matter which way the attacks use, we can generally model the attacks as

$$Ag(t) = A(t) + D(t) \qquad t_s \le t \le t_e \tag{2}$$

where $Ag(t)$ is the aggregation result received by the base station in epoch t and $D(t)$ is the deviation of the aggregation result in epoch t caused by the attack. $Ag(*)$, $A(*)$ and $D(*)$ are regarded as time-variant functions. $[t_s, t_e]$ is the duration where the temporal variation pattern of aggregation results is manipulated. We assume that the attack preserves the continuity in the fabricated aggregation results, since if not it can be easily detected by the users through checking Formula (1). In other words, we have $|Ag(t) - Ag(t+1)| \le \Lambda$.

Our security goal is to protect the authenticity of the temporal variation pattern observed by the users. Specifically, for a series of aggregation results $Ag = (Ag(t), Ag(t+1), \ldots, Ag(t+l))$ in a continuous aggregation, we want to guarantee that if the base station accepts Ag, the temporal variation pattern of Ag is close to the true pattern with a high probability.

Notations We list below notations in this paper.

- u,v,w (in lower case) are sensor nodes.
- N is the total number of sensor nodes.
- N_u is the set of u's neighbors in u's communication range including itself.
- N_u^2 is the set of u's two-hop neighbors outside its communication range.
- R_c is the communication radius of sensor nodes.
- $K_u$is u's individual key shared between u and the BS.
- $MAC(K, m)$ is the message authentication code of message m generated with a symmetric key K.
- $r_{u,t}$ is the sensor reading of u in epoch t.

4 SECURE CONTINUOUS AGGREGATION

4.1 Overview

During the period of a continuous aggregation query, each sensor node caches l_{max} number of sensor readings that contribute to the aggregations in the latest l_{max} epochs. l_{max} determines the maximum length of the time window in which the temporal variation pattern of the aggregation results can be verified.

Once the users observe an interesting temporal variation pattern of the aggregate, they can verify its authenticity *on-demand*. However, in the circumstance that the adversary is interested in suppressing the real appearance of an interesting temporal variation pattern, the users cannot decide when to conduct verification since they do not know when the interesting pattern really appears. Thus, *periodic* verification is required. To this end, the period of the aggregation query is divided into successive time windows. Each time window consists of several successive epochs. At the end of each time window, the temporal variation pattern in this time window is verified.

Either in the on-demand verification or in the periodic verification, the Base Station (BS) selects some points from the series of aggregation results in the time window to be verified, and checks their correctness to detect any fabrication of temporal variation patterns. Considering that the adversary can manipulate only a small number of aggregation results such as extreme points to tamper with the temporal variation pattern, it may be ineffective to check a set of randomly selected points to detect forged patterns, because the selected points may not cover these manipulated points, which causes that the attack is not detected. Thus, to guarantee effective attack detection, the selected points should be able to capture the temporal variation pattern in the time window like extreme points. We refer to these points as *representative points* and the epoch of a representative point as *representative epoch* hereinafter. After the selection of representative points, the BS broadcasts a verification request, which includes the representative epochs, the sampling ratio ϱ and a nonce number $nonce_v$, to the WSN. Once receiving the verification request, each node decides whether to act as a sampled node. Before the sampled nodes send to the BS their sensor readings of every representative epoch, their neighboring nodes verify the correctness of sample data and authenticate the sample messages.

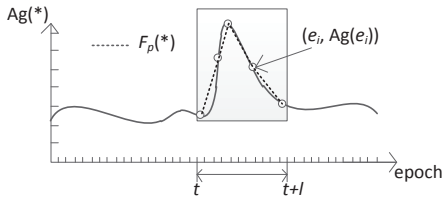


Fig. 2. Definition of representative points

With the sensor reading samples, the BS checks the correctness of the aggregation results of each representative epoch by hypothesis testing. The general form of the hypothesis tests is

$$H_0 \ : \ A(t) = Ag(t) \quad \text{vs} \quad H_a \ : \ A(t) \neq Ag(t) \tag{3}$$

If the aggregation results in all representative epochs are verified as correct, the temporal variation pattern in the time window is assumed to be authentic.

In the rest of this paper, we suppose that the time window to be verified is from epoch t to epoch $t + l$, denoted by $[t, t + l]$, and we always take the points at the boundary epochs t and $t + l$ as two representative points. Obviously, $l + 1 < l_{max}$.

4.2 Representative Point Selection

We first give the definition of representative point to formally characterize the requirement that is to capture the temporal pattern of the whole aggregation result series. Figure 2 shows an example.

Definition 1 (representative points): Let $P = \{(e_i, Ag(e_i)) \mid 1 \le i \le p, e_1 = t < e_2 < \ldots < e_{p-1} < e_p = t + l\}$ be a set of points in the time window $[t, t + l]$ where $Ag(e_i)$ is the aggregation result in epoch e_i. Let $F_P(*)$ be the piece-wise linear function consisting of connected line segments, each of which is between point $(e_i, Ag(e_i))$ and $(e_{i+1}, Ag(e_{i+1}))$ for $1 \le i \le p - 1$. If F_P is a best approximation of the series of aggregation results $Ag(*)$ within $[t, t + l]$ among all possible $F_{P'}$ where $P' = \{(e'_i, Ag(e'_i)) \mid 1 \le i \le p, e'_1 = t < e'_2 < \ldots < e'_{p-1} < e'_p = t + l\}$, we say P captures the temporal pattern of aggregation results and the points in P are representative points in the time window $[t, t + l]$. Here the goodness of approximation is assessed by the approximation error between $F_P(*)$ and $Ag(*)$, which is measured by their Euclidean distance $E(t, t + l) = \sqrt{\sum_{k=t}^{t+l} \{Ag(k) - F_P(k)\}^2}$.

4.2.1 Representative Point Selection

Given Definition 1, the representative point selection (RPS) problem can be described as follows. Given an integer p ($p \ge 2$), find a set of points $P = \{(e_i, Ag(e_i)) \mid 1 \le i \le p, e_1 = t < e_2 < \ldots < e_{p-1} < e_p = t + l\}$ such that the error of approximation of $Ag(*)$ by $F_P(*)$ in the time window $[t, t + l]$ is minimized and $|P| = p$.

Let $F_{(a,b)}(x)$ ($a < b$) be the linear function through point $(a, Ag(a))$ and $(b, Ag(b))$. $I(a, b)$ is the approximation error of the aggregation

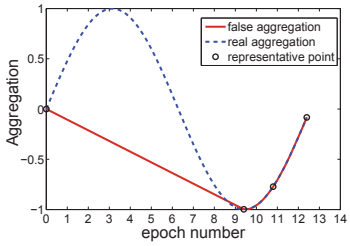


Fig. 3. An example of attack against RPS algorithm.

results from epoch a to b by $F_{(a,b)}(x)$. Then, we have

$$F_{(a,b)}(x) = \frac{Ag(b) - Ag(a)}{b-a}(x-a) + Ag(a) \tag{4}$$

$$I(a,b) = \sum_{k=a}^{b} \{Ag(k) - F_{(a,b)}(k)\}^2. \tag{5}$$

Assuming $t' > t$, we let $E(t,t',p_{t'})$ be the minimum approximation error when $p_{t'}$ ($p_{t'} \geq 2$) number of points between epoch t and t' ($> t$) (including t and t') are selected as representative points for the time window $[t,t']$. Suppose $e'_1 = t, e'_2, \ldots, e'_{p_{t'}-1}, e'_{p_{t'}} = t'$ are representative epochs, the points of epochs $e'_1, e'_2, \ldots, e'_{p_{t'}-1}$ must be an optimal selection of $p_{t'}-1$ points for approximating $Ag(*)$ in the time window $[t, e'_{p_{t'}-1}]$. By the optimal substructure of the problem, the following recursive formula is given to compute $E(t,t',p_{t'})$,

$$E(t,t',p_{t'}) = \begin{cases} \min\limits_{t+p_{t'}-2 \leq k < t'} & \{E(t,k,p_{t'}-1) + I(k,t')\}, \\ & \text{if } \ 2 < p_{t'} < t'-t+1, \\ 0, & \text{if } \ p_{t'} \geq t'-t+1, \\ I(t,t'), & \text{if } \ p_{t'} = 2. \end{cases} \tag{6}$$

Based on (6), we propose a dynamic programming algorithm called *RPS algorithm* to solve the RPS problem. The pseudocode of RPS algorithm is shown in Appendix A (All appendices are in the supplementary file). The algorithm takes $O(lp)$ space and $O(l^3p)$ time, considering the $O(l)$ time of the computation of formula (5).

4.2.2 RPS with Pre-specified Points

With the knowledge of RPS algorithm and the ability of predicting the real temporal variation pattern of the aggregate, the adversary may try to forge a series of aggregation results of which the selected representative points have aggregation values equal or close to the real ones. If such attempt is successful, the check of representative points will not detect the fabrication of the temporal variation. Figure 3 shows an example of fabricated series of aggregation results and the representative points selected by RPS algorithm over the fabricated series. The aggregation values of representative points are the same as the real aggregation results, which causes that the false pattern between epoch 0 and 9 cannot be detected. Considering such possibility, the randomness is introduced to make the output of the selection algorithm unpredictable. To this end, each data point in $(t, t+l)$ (not including epoch t and $t+l$) is pre-specified as a representative point with a probability of q in our scheme. Then, the remaining number of representative points including the ones at two boundary epochs t and $t+l$ are selected to minimize the approximation error. On the other hand, some points such as the maximum and minimum aggregation results, which describe the significant characteristics of the temporal variation pattern, should be always pre-specified as representative points.

Therefore, we consider the problem of RPS with pre-specified points (RPS-P): given a set of pre-specified points $\widetilde{P}$ ($\widetilde{P} \supseteq \{(t, Ag(t)), (t+l, Ag(t+l))\}$) and an integer p ($p \geq |\widetilde{P}|$), find a set of representative points P such that the approximation error of $Ag(*)$ by $F_P(*)$ in $[t, t+l]$ is minimized while $\widetilde{P} \subset P$ and $|P| = p$. We can see that RPS-P problem becomes RPS when $\widetilde{P} = \{(t, Ag(t)), (t+l, Ag(t+l))\}$.

Let $\widetilde{P} = \{(t_i, Ag(t_i)) | 1 \leq i \leq \widetilde{p}\ (\widetilde{p} \geq 2), t_1 = t < t_2 < \ldots < t_{\widetilde{p}-1} < t_{\widetilde{p}} = t+l\}$. Assuming $i \geq 2$, we let $\widetilde{E}(t_1, t_i, p_i)$ be the minimum approximation error when p_i ($p_i \geq i$) number of representative points, which includes pre-specified ones $\{(t_j, Ag(t_j)) | 1 \leq j \leq i\}$, are selected for the approximation in the time window $[t_1, t_i]$. Similarly, the following recursive formula is given to compute $\widetilde{E}(t_1, t_i, p_i)$,

$$\widetilde{E}(t_1,t_i,p_i) = \begin{cases} \min\limits_{i-1 \leq k \leq c_i} \{\widetilde{E}(t_1, t_{i-1}, k) & + E(t_{i-1}, t_i, p_i - k + 1)\}, \\ & \text{if } \ i < p_i < t_i - t_1 + 1, i > 2 \\ 0, & \text{if } \ p_i \geq t_i - t_1 + 1, i > 2 \\ \sum\limits_{j=1}^{i-1} I(t_j, t_{j+1}), & \text{if } \ p_i = i > 2. \\ E(t_1, t_i, p_i), & \text{if } \ i = 2. \end{cases} \tag{7}$$

where $c_i = \min\{t_{i-1} - t_1 + 1, p_i - 1\}$. $E(t_{i-1}, t_i, p_i - k + 1)$ and $E(t_1, t_i, p_i)$ are computed by (6).

Based on (7), a dynamic programming algorithm is also proposed to solve RPS-P problem, referred to as RPS-P algorithm. The pseudocode is shown in Appendix B. Considering the function call RPS() to RPS algorithm takes $O(l^3p)$ time and $O(lp)$ space, RPS-P algorithm takes $O(l^3p^3\widetilde{p})$ time and $O(\widetilde{p}p + lp)$ space.

4.2.3 The number of representative points

Selecting more representative points can further enhance the capability of our scheme to detect forged temporal variation pattern, because a larger number of representative points can better capture the variation pattern of aggregation results and have a higher probability to cover the manipulated period. However, since each representative point needs to be verified by collecting sensor reading samples in the corresponding representative epoch from the WSN, more representative points mean higher communication cost. Therefore, there is a tradeoff between detecting capability and communication cost. Section 4.2.1 and 4.2.2 actually address the optimal representative point selection to minimize the approximation error with a given budget on communication cost, i.e., a given number of representative points. On the other hand, the users would need to decide at least how many representative points are required to achieve the desired detecting capability of the scheme. Thus, here we consider the problem of minimizing the number of representative points given a certain degree of the approximation error that the users can tolerate.

The problem can be formally described as follows. Given a set of pre-specified points $\widetilde{P}$ and the maximum approximation error that the users can tolerate, denoted by $\mathbb{E}$, find a set of representative points P such that the approximation error of $Ag(*)$ by $F_P(*)$ in $[t, t+l]$ is not greater than $\mathbb{E}$ and $|P|$ is minimized. Based on RPS-P algorithm, the solution for this problem is simple. Let $p = l+1$. According to Formula 7, RPS-P algorithm will generate the result of $\widetilde{E}(t, t+l, k)$ for all $\widetilde{p} \leq k \leq l+1$. Because it is obvious that $\widetilde{E}(t, t+l, k)$ decreases as k increases, we can simply conduct a linear or binary search to find the first entry with $\widetilde{E}(t, t+l, k) \leq \mathbb{E}$, where k is the minimum number of representative points for the problem. By auxiliary matrix $s[i, j]$, we can obtain the corresponding representative points in the same way as RPS-P algorithm except p_l is initialized with the obtained minimum k in line 22. The computation takes $O(l^6\widetilde{p})$ time and $O(\widetilde{p}l + l^2)$ space.

4.3 Secure Sampling

After selecting the representative points, the BS checks the correctness of each representative point by sampling and hypothesis testing shown in Formula (3). In our scheme, the WSN is uniformly sampled. Sampling provides nice security properties that the integrity of each sample can be authenticated by a single node with its individual key, and malicious nodes cannot fabricate or change the reported samples of honest nodes. However, the adversary can provide forged samples through the compromise nodes to make the BS accept the null

hypothesis in (3). Besides, if without any protection, the malicious nodes can easily and always pretend to be sampled to provide false samples while not being detected. Therefore, we consider the following problems in the sampling process: first, how the BS verifies the legitimacy of sampled nodes; second, how to detect false samples provided by the malicious nodes.

4.3.1 Verifiable Random Sampling

In the verifiable random sampling, each node decides whether it is sampled by computing a cryptographically secure pseudo-random function h. h uniformly maps the input values into the range of $[0, 1)$. Specifically, being informed of a sampling ratio ϱ broadcasted from the BS, each node, say v, checks the inequality

$$h_{K_v}(nonce|nonce_v) \leq \varrho \tag{8}$$

where $nonce$ and $nonce_v$ are nonce numbers that are disseminated within each aggregation query and verification request respectively. If Inequality (8) holds, v sends its sample $R_v = (r_{v,e_1}, r_{v,e_2}, \ldots, r_{v,e_p})$, i.e., its sensor readings in the representative epochs, to the BS. In this way, whether a node is sampled is decided by the node individual key and two nonce numbers which are known by the BS. Since each time of verification different $nonce|nonce_v$ is used, the nodes are randomly selected to be sampled for every verification. Also, a malicious node cannot arbitrarily claim to be sampled, since the BS can verify the legitimacy of v as sampled node by checking whether (8) holds.

The actual number of samples returned by this sampling approach is random. Thus, the determination of sampling ratio ϱ needs to provide a probabilistic guarantee to achieve the target sample size of at least m_t. Here Theorem 1 is given to decide ϱ. The proofs of all theorems in this paper are given in the supplementary file.

Theorem 1: Given a target sample size of at least m_t, to guarantee the final sample size $m \geq m_t$ with a probability of at least $1 - \delta_s (\delta_s < 0.5)$, the sampling ratio ϱ is at least ϱ^*

$$\varrho^* = \frac{0.5}{N * c^2 + N^2}(Nc^2 + 2(m_t - 0.5)N + (N^2c^4 + 4N^2c^2(m_t - 0.5) - 4(m_t - 0.5)^2Nc^2)^{0.5})$$

where $c = \Phi^{-1}(\delta_s)$.

4.3.2 Local Sample Authentication

In many applications like environment monitoring, the measurements from multiple sensors in the same space are often highly correlated and exhibit a high similarity on statistical distributions. This fact has been widely exploited in efficient information extraction [24], routing protocols [25], scheduling algorithms [26] and attack detection [27]. We also exploit such fact to propose a local sample authentication mechanism to prevent the malicious sampled node from arbitrarily providing false samples.

In the local sample authentication, each sampled node, say v, broadcasts its sample R_v to its neighbors in order to obtain authentication from its neighbors before sending R_v to the BS. Each neighbor u verifies the validity of R_v. If the verification is successful, u sends to v the message authentication code of R_v computed by u's key used for the authentication of v's sample.

Local Authentication Key Setup. To derive keys for the local sample authentication, each node u is loaded with a seed key K_u^s before deployment. The BS holds the seed keys of all nodes. During the safe bootstrapping phase, u discovers its one-hop neighbors and computes a key by $K_{u,v}^a = H(K_u^s|v)$ for each neighbor v, where H is a secure cryptographic hash function. Then, u erases K_u^s. The key $K_{u,v}^a$ is stored and used by u to authenticate samples from v. Since each authentication keys is bound with a neighbor pair, the adversary cannot use it to authenticate samples from arbitrary nodes except for the corresponding neighboring node. The erasure of seed keys prevents the adversaries from deriving all the authentication keys.

Local Verification and Authentication. Once receiving v's sample $R_v = (r_{v,e_1}, r_{v,e_2}, \ldots, r_{v,e_p})$, each neighbor u collects necessary information from its neighborhood and verifies the validity of R_v by checking the following two conditions:

$$\rho_{u,v} > \rho_T^u \tag{9a}$$

$$\mu_v \notin \text{OutlierDetect}(\{\mu_w | w \in \widetilde{N}_u, \widetilde{N}_u \subseteq N_u\}) \tag{9b}$$

where $\rho_{u,v}$ is the Pearson correlation coefficient between $R_u = (r_{u,e_1}, r_{u,e_2}, \ldots, r_{u,e_p})$ and R_v, ρ_T^u is a threshold determined by node u or pre-specified by the user. $\rho_{u,v}$ is computed by

$$\rho_{u,v} = \rho(R_u, R_v) = \frac{\sum_{i=1}^{p}(r_{u,e_i} - \mu_u)(r_{v,e_i} - \mu_v)}{\sqrt{\sum_{i=1}^{p}(r_{u,e_i} - \mu_u)^2}\sqrt{\sum_{i=1}^{p}(r_{v,e_i} - \mu_v)^2}}$$

where $\mu_x (x = u, v, w)$ is the average of R_x denoting sensor readings of node x in the representative epochs. OutlierDetect(S) is a function call to some outlier detection method to identify and return outliers in set S.

The condition (9a) is used to detect possible abnormal reductions of spatial correlation. Considering that real sensor readings from two proximate nodes are often highly correlated and hold similar temporal variation patterns, the fabricated data from a malicious node is very likely to have a low correlation coefficient with the real data from neighboring nodes.

Generally, $\rho(R_u, R_v)$ depends on the distance $d(u, v)$ between u and v. In Geostatistics, a covariance function $\gamma(\cdot)$ is used to model the relation between $\rho(R_u, R_v)$ and $d(u, v)$ [28]. The covariance function is assumed to be nonnegative and decrease monotonically with increasing distance d. With the knowledge of the covariance model for observed physical qualities, the users can decide the value of ρ_T^u. For example, given *Power Exponential model* $\gamma(d) = e^{-d/\theta_1}(\theta_1 > 0)$, since u is within the communication range of v, we have $d(u, v) \leq R_c < \tau R_c (1 < \tau \leq 2)$, then ρ_T^u can be set as $\gamma(\tau R_c)$ according to that $\gamma(d)$ decreases monotonically with increasing d. The users can determine τ according to the spatial correlation model of sensor readings in the monitored area, with following the rule that τ should be large enough such that the condition (9a) is true for real sensor readings with a high probability, and also should be as small as possible in order to efficiently filter false sensing data.

If the users cannot prior estimate and pre-specify ρ_T^u by the knowledge of the covariance model before network deployment, we need to determine it during operation. We propose to estimate ρ_T^u autonomously by a model-free method. The node u randomly selects a subset of nodes from its two-hop neighbors N_u^2, denoted by $\widetilde{N}_u^2$, and collets their sensor readings in the representative epochs. For each node $w' \in \widetilde{N}_u^2$, u computes the correlation coefficient $\rho_{u,w'}$, and sets $\rho_T^u = MEDIAN(\{\rho_{u,w'} | w' \in \widetilde{N}_u^2\})$. The median value is used to replace average to defend against the attack of deflating ρ_T^u by malicious samples, since median is more robust than average [13]. Because u usually has a longer distance to its any two-hop neighbors than to its one-hop neighbors, we have $\rho_{u,v} > \rho_T^u$.

The condition (9b) exploits the amplitude similarity of sensor reading series in the neighborhood. Beside sharing the similar temporal pattern, two series of sensor readings R_u and R_v from the proximate neighborhood do not have too much deviation on their amplitude scales μ_u and μ_v. To check the condition, u first collects the means of sensor readings in the representative epochs from a set of randomly selected nodes in N_u, denoted by $\widetilde{N}_u$. For a sparse network, u may also collect the means of its two-hop neighbors. Then, the outlier detection technique is used to check the abnormality of μ_v, which achieves statistically robustness and effectiveness. Considering that there may be multiple malicious neighbors to provide forged data, we use Rosner's test [29] for outlier detection. It is a generalization

of Grubbs' test [30] for multiple outliers. To use it, an upper limit U_r must be specified on the number of potential outliers present. In our scheme U_r depends on $|\widetilde{N}_u|$ and U_r can be set to at most $|\widetilde{N}_u|/2$. In this paper, a significance level $\alpha = 0.05$ is used for Rosner's test.

Once u finds both (9a) and (9b) hold for R_v, u computes $MAC(K^a_{u,v}, R_v)$ with its local authentication key $K^a_{u,v}$ and sends $MAC(K^a_{u,v}, R_v)$ to v. Otherwise, u ignores the authentication request. A pseudo-code of the entire procedure of local authentication is shown in Appendix D, including the procedure of Rosner's test.

4.3.3 Sample Message Transmission

After v collects c ($c \geq 1$) number of MACs from its neighbors $\{u_i | 1 \leq i \leq c\}$, v transmits to the BS its sample message

$$S_v = \{R_v, (v, u_1, \ldots, u_c), XMAC\}$$

, where $XMAC = MAC(K_v, R_v) \oplus MAC(K^a_{u_1,v}, R_v) \ldots \oplus MAC(K^a_{u_c,v}, R_v)$. The XOR of MACs reduces the communication cost, which has been proved to be secure [31]. Here c is a security threshold pre-specified by the users.

4.4 Aggregation Verification

Once broadcasting the verification request, the BS waits for some time t_w to ensure the arrivals of all samples. Considering the network delivery time of the verification requests and sample messages, t_w should be at least twice of the message delivery time from the network boundary to the BS plus the time for the local sample authentication. According to the procedure of local sample authentication, the time required to complete it consists of the time of one hop broadcast from a sample node and two hop broadcast from each of its neighbor nodes, and also the time for each neighbor to collect sensor readings in its two hop neighborhood and for the sampled node to collect MACs from its one-hop neighbors. These times can be easily estimated and accordingly the time for the local sample authentication can be estimated. When time expires, the BS first checks the validity of every arrived sample and the sample size, and then verifies the aggregation results in representative epochs.

4.4.1 Sample Message Verification

For every sample message, say S_v claimed from node v, the BS verifies its validity in two steps. First, the BS verifies the legitimacy of the claimed sampled node v by checking whether Inequality (8) holds because the BS knows h, $nonce$, $nonce_v$ and K_v. Then, the BS verifies $XMAC$ in the sample message. Since the BS holds the seed key K^s_u of any node u, it can generate u's authentication key $K^a_{u,v}$. The BS generates $K^a_{u_i,v}$ for each node u_i in the ID list $(u_1, \ldots, u_T)$ in S_v, recomputes $XMAC$ and compare it with the one in S_v for equality. If the verification in any step above fails, the BS drops S_v and raises an alarm. Otherwise, the BS accepts S_v. In this way, all invalid sample messages are dropped.

During the local sample authentication, a false sample may pass the local verification and be successfully authenticated by c neighbors due to sufficient number of compromised nodes in the same neighborhood. However, it is expensive for the adversary to provide a large portion of false samples because of the verifiable random sampling and local sample authentication. Thus, we assume the number of false samples is relatively small to the total sample size and we can use Rosner's test to detect outlying sensor readings in each representative epoch. The sampled nodes from which outlying sensor readings are detected are labeled as outlying nodes and the hypothesis testing is conducted over the samples excluding those from outlying nodes.

4.4.2 Hypothesis testing for aggregation verification

Let m be the final sample size after the sample message verification. The set of nodes where the final samples are from is denoted by $\{s_i | 1 \leq i \leq m\}$. Here we discuss the verification for the count, average and sum queries in a representative epoch k respectively.

Predicate Count Aggregate. The predicate count query is used to determine the total number of nodes whose sensor readings have some property in the network (e.g., number of sensors sensing temperature $> 30°C$). Let $A_{c(\Theta)}(k)$ and $Ag_{c(\Theta)}(k)$ be the true aggregation result and the in-network aggregation result respectively for counting the nodes whose sensor readings in epoch k satisfy some predicate Θ. The predicate count aggregation is verified by the following hypothesis testing with regard to probability distribution:

$$\begin{aligned} H_0 &: p_1 = \frac{Ag_{c(\Theta)}(k)}{N}, p_2 = 1 - \frac{Ag_{c(\Theta)}(k)}{N} \\ H_a &: p_1 \neq \frac{Ag_{c(\Theta)}(k)}{N}, p_2 \neq 1 - \frac{Ag_{c(\Theta)}(k)}{N} \end{aligned} \tag{10}$$

where p_1 is the probability of a sensor node satisfying Θ and $p_2 = 1 - p_1$.

According to the hypothesis testing theory, the χ^2 goodness-of-fit test can be used for (10). m_1 be the number of samples satisfying Θ in epoch k and $m_2 = m - m_1$. The test statistic is computed by

$$X^2 = \frac{(m_1 - m\frac{Ag_{c(\Theta)}(k)}{N})^2}{m\frac{Ag_{c(\Theta)}(k)}{N}} + \frac{(m_2 - m(1 - \frac{Ag_{c(\Theta)}(k)}{N}))^2}{m(1 - \frac{Ag_{c(\Theta)}(k)}{N})} \tag{11}$$

When H_0 is true, X^2 approximately follows χ^2-distribution with one degree of freedom. Let $\chi^2_\alpha(1)$ be the upper 100α percentage point of χ^2-distribution with one degree of freedom. Given a significance level α, if $X^2 \geq \chi^2_\alpha(1)$ or the p-value of X^2 is smaller than α, H_0 is rejected. Otherwise, H_0 is not rejected and the BS accepts $Ag_{c(\Theta)}(k)$ as true.

For the predicate count aggregation in epoch k, the use of χ^2 goodness-of-fit test assumes adequate expected numbers in each cell, i.e., $mp_1 \geq 10$ and $mp_2 \geq 10$. Then, the hypothesis testing would be inefficient in the case that $p_1 >> p_2$ (e.g., p_2 is small) or $p_1 << p_2$ (e.g., p_1 is small), because a large sample size is required to achieve $mp_2 \geq 10$ or $mp_1 \geq 10$ respectively. Suppose $p_2 = 0.01$, the sample size should be 1000 at least. To address this problem, we use an alternative verification approach which verifies whether the minority is true, i.e, either checking p_2 if $p_1 >> p_2$ or checking p_1 if $p_1 << p_2$. This is because that if the result is not true, it most likely matters only when the attacks cause the number in a category to decrease too much to a small degree. Therefore, if $p_1 << p_2$ ($p_1 >> p_2$) and p_1 (p_2) is not true, we assume that $A_{c(\theta)}$ ($N - A_{c(\theta)}$) is notably larger than $Ag_{c(\theta)}$ ($N - Ag_{c(\theta)}$).

Suppose $Mg = \min\{Ag_{c(\theta)}(k), N - Ag_{c(\theta)}(k)\}$ and M is the true number in the cell corresponding to Mg. We conduct sampling with ratio ϱ against the nodes within the smaller cell. Then, the number of received samples m follows a Binomial distribution $B(M, \varrho)$. Given m, if $m > Mg$, then the aggregation result is rejected. Otherwise, we can use one-tail binomial test to verify whether Mg is correct. Let X be a random variable following Binomial distribution $B(Mg, \varrho)$. Given a significant level α_B, we compute $m_U = \min\{k \mid \Pr(X \geq k) \leq \alpha_B\}$, If $m > m_U$, the aggregation result is rejected.

Average Aggregate. Let $A_a(k)$ and $Ag_a(k)$ be the true aggregation result and the in-network aggregation result of average query in epoch k respectively. According to the central limit theorem, the sample mean is approximately normally distributed for large sample sizes. Thus, t-test is used to test the hypothesis

$$H_0 : A_a(k) = Ag_a(k) \text{ vs } H_a : A_a(k) \neq Ag_a(k). \tag{12}$$

With the sample mean $\widehat{A}_a(k) = \frac{1}{m}\sum_{i=1}^{m} r_{s_i,k}$ and the sample variance $\sigma^{*2} = \frac{1}{m-1}\sum_{i=1}^{m}(r_{s_i,k} - \widehat{A}_a(k))^2$, the test statistic is computed by

$$T = \frac{\widehat{A}_a(k) - Ag_a(k)}{\sigma^* / \sqrt{m}} \tag{13}$$

When H_0 is true, T follows t-distribution with $m-1$ degrees of freedom. Let $t_{\frac{\alpha}{2}}(m-1)$ be the upper $100\alpha/2$ percentage point of t-distribution with $m-1$ degrees of freedom. Given a significance level α, if $|T| \geq t_{\frac{\alpha}{2}}(m-1)$ or the p-value of T is smaller than α, H_0 is rejected. Otherwise, the BS accepts $Ag_a(k)$ as true.

Sum Aggregate. For the sum query, the sum aggregation result $Ag_s(k)$ is checked by verifying the average aggregation $\frac{Ag_s(k)}{N}$.

5 Analysis and Parameter Determination

In this section we analyze the effectiveness of our scheme for detecting false temporal variation patterns of aggregates, and discuss how to determine the parameters including sampling ratio and probability of pre-specifying representative point used in our scheme. The analysis of our scheme's overhead is given in Appendix F.

5.1 Effectiveness Analysis

5.1.1 Verification Effectiveness of Representative Point

Our aggregation verification scheme provides a statistical security guarantee for the aggregation result in each representative epoch. This is because two kinds of errors could occur in the hypothesis testing: Type I (false positive) if H_0 is rejected when it is actually true; Type II (false negative) if H_0 is not rejected when its actually false. Type I error causes the BS to reject the true aggregation results and the Type II error causes the BS to accept the false aggregation results.

Theorem 2: Let m be the number of samples and α be the significance level used for the hypothesis testing, we have:

- If the in-network aggregation result $Ag(k)$ is true, then the BS accepts it with probability at least $1-\alpha$.
- Given a constant value Δ, if $|Ag(k) - A(k)| > \Delta$, then:
  - 1) With χ^2 goodness-of-fit test for the predicate count aggregation, the BS rejects $Ag_{c(\theta)}(k)$ with probability at least
$$1 - \left\{\Phi\left[\frac{1}{\sigma_c}\left(a_\chi + m\frac{\Delta}{N}\right)\right] - \Phi\left[\frac{1}{\sigma_c}\left(-a_\chi + m\frac{\Delta}{N}\right)\right]\right\} \quad (14)$$
where $p_r = \frac{A_{c(\Theta)}(k)}{N}$, $p_1 = \frac{Ag_{c(\Theta)}(k)}{N}$, $a_\chi = \sqrt{\chi^2_\alpha(1) m p_1(1-p_1)}$, $\sigma_c = \sqrt{m p_r(1-p_r)}$ and Φ is the standard normal cumulative distribution function. Here we assume $p_r(1-p_r) \neq 0$.
2) Through verifying the smaller number of two cells with one-tail binomial test and sampling ratio ϱ, the BS rejects $Ag_{c(\theta)}(k)$ with probability at least
$$\sum_{i=m_U+1}^{Mg+\Delta} \binom{Mg+\Delta}{i} \varrho^i (1-\varrho)^{Mg+\Delta-i} \quad (15)$$
  - For the average aggregation over the whole network, the BS rejects $Ag_a(k)$ with probability at least
$$1 - \left[\Phi\left(t_{\frac{\alpha}{2}}(m-1) + \frac{\Delta}{\sigma/\sqrt{m}}\right) - \Phi\left(-t_{\frac{\alpha}{2}}(m-1) + \frac{\Delta}{\sigma/\sqrt{m}}\right)\right] \quad (16)$$
where σ^2 is the population variance of sensor readings in epoch k.

5.1.2 Verification Effectiveness of Aggregation Variation Pattern

Successful verification of the representative points means that the temporal variation pattern observed by the users shares these common points with the real one. This at least indicates that the variation pattern given by linear piece-wise function F_p is embedded in the real variation pattern of aggregate and provides users the pattern information in a certain extent. The verification of representative points forces the adversary to distort the variation pattern by manipulating a series of aggregation results between two representative points, i.e., in a time window where no epochs are selected as representative epochs. However, we can show that it is difficult for the adversary to generate such an undetected series including true representative points.

Except for pre-specified representative points, RPS-P algorithm selects the representative points at the positions where the approximation error by piece-wise linear function is minimized. Hence, for the variations of the aggregation results, our algorithm most likely captures the turning points in them, which is inevitable especially when the adversary wants to fabricate some change trends of interest to users. The way for the adversary to avoid turning points is to generate linear trends with two true end points, since the algorithm tends to select two end points on the line. But randomness introduced in the selection of optimal representative points can help to prevent such attempt, because every point in the series is pre-specified as representative point with probability q.

We assume that the adversary manipulates a series of aggregation results of length l_f, denoted by $Ag(1), Ag(2), \ldots, Ag(l_f)$, which deviate from the true values by at least $D(1), D(2), \ldots, D(l_f)$ respectively. Each data point of aggregation results is verified with a probability of q. If being verified, $Ag(i)$ is rejected with probability at least $Pr_{reject}(D(i))$, which is given by Theorem 2. Therefore, the probability of $Ag(i)$ being detected is at least $qPr_{reject}(D(i))$. Then, since there are l_f number of aggregation results, the fabricated series can be detected with probability at least
$$1 - \prod_{i=1}^{l_f}(1 - qPr_{reject}(D(i))) \quad (17)$$
We can see it increases with l_f, q and $D(i)$.

Suppose that D_{max} is the maximum deviation of the fabricated series from the true values, i.e., $D_{max} = \max_{t=t_s}^{t_e} D(t)$. Given successful verification of representative points, the undetected fabricated series should return to true values at the start and end positions, which means that the deviation decreases to zero. According to the continuity assumptions for the real and fabricated aggregation results in Section 3, the difference between aggregation values at two successive epochs is bounded by Λ, then the maximum decreasing speed of the difference between the real and fabricated aggregation results is 2Λ. Then, we have
$$l_f \geq 2\frac{D_{max}}{2\Lambda} = \frac{D_{max}}{\Lambda} \quad (18)$$
We note that D_{max} is infinity norm distance between two series and characterizes the difference between their variation patterns. Formula (18) indicates that the adversary needs to fabricate a longer series of aggregation results to achieve larger distortion of variation pattern, which then increases the detection probability due to larger l_f.

5.2 Parameter Determination

5.2.1 Sampling Ratio

A larger sample size can reduce the probabilities of Type I and Type II error in the hypothesis testing. Since the probability of Type I error is limited by the significance level α, the determination of the sample size aims at limiting the probability of Type II error. Formally, given the maximum deviation Δ which the user can tolerate, the problem is how to determine the sample size such that if $|Ag(k) - A(k)| > \Delta$ for some $k \in \{e_1, e_2, \ldots, e_p\}$, the probability of H_0 being rejected is at least $1-\beta$.

For the verification against the smaller number in cells when $p_1 << p_2$ or $p_1 >> p_2$, the desired sampling ratio ϱ should satisfy that the probability of Formula (15) is at least $1-\beta$. we can use Theorem 1 to compute ϱ, by letting $m_t = m_U + 1$, $N = Mg + \Delta$ and $\delta_s = \beta$.

For the verification using χ^2 goodness-of-fit test, according to Theorem 2, the desired sample size should satisfy
$$\Phi\left[\frac{1}{\sigma_c}\left(a_\chi + m\frac{\Delta}{N}\right)\right] - \Phi\left[\frac{1}{\sigma_c}\left(-a_\chi + m\frac{\Delta}{N}\right)\right] \leq \beta \quad (19)$$

for the predicate count aggregation in epoch k.

For the average aggregation in epoch k, the desired sample size should satisfy

$$\Phi\left(t_{\frac{\alpha}{2}}(m-1)+\frac{\Delta}{\sigma/\sqrt{m}}\right)-\Phi\left(-t_{\frac{\alpha}{2}}(m-1)+\frac{\Delta}{\sigma/\sqrt{m}}\right)\leq\beta \qquad (20)$$

If σ_c and σ in (19) and (20) are known, we can use a binary search in $[1, N]$ to find the least sample size in epoch k, denoted by $m(k)$, required to satisfy the inequalities (19) and (20). However, σ_c and σ are most likely unknown in practice. To address this problem, the BS could first broadcasts the verification request with an initial sampling ratio ϱ_1 to collect samples for estimating σ_c and σ in each representative epoch k. Then, the BS computes $m(k)$. The number of received samples m may be smaller than $m(k)$ because of insufficient initial sampling ratio ϱ_1, packet loss and local verification filtering. In these cases, a new sampling ratio $\varrho_2(k)$ is computed for epoch k according to Theorem 1 with $m_t = m(k)$. The BS broadcasts $\{\varrho_2(k) \mid m < m(k), k \in \{e_1, e_2, \ldots, e_p\}\}$ to the network. Any sensor node, which satisfies Inequality (8) where $\varrho = \varrho_2(k)$, transmits its reading in epoch k to the BS if the node was not sampled at the time of the previous round of sampling.

5.2.2 Probability of Pre-specifying Representative Point

Assuming that the adversary attempts to fabricate a series of aggregation results which deviate from the real ones by at least Δ (Δ is the maximum deviation that the user can tolerate). According to Formula (17), the detection probability is at least $1-(1-qPr_{reject}(\Delta))^{l_f}$. The adversary may reduce the detection probability by decreasing l_f which is the length of fabricated aggregation series. However, Formula (17) also indicates that we can ensure the detection probability by increasing q, the probability of a data point being specified as representative point, to offset the impact of decreasing l_f. Accordingly, we can determine the minimum value of q.

The sampling ratio given in Section 5.2.1 ensures $Pr_{reject}(\Delta) \geq 1-\beta$. Then, we have $1-(1-qPr_{reject}(\Delta))^{l_f} \geq 1-(1-q(1-\beta))^{l_f}$. According to Formula (18), the minimum detection probability that the adversary can achieve is

$$1-(1-q(1-\beta))^{\frac{\Delta}{\Lambda}} \qquad (21)$$

We choose q in order to ensure a desired lower bound γ for the detection probability by

$$1-(1-q(1-\beta))^{\frac{\Delta}{\Lambda}} \geq \gamma \qquad (22)$$

Then we have

$$q \geq \frac{1-(1-\gamma)^{\frac{\Delta}{\Lambda}}}{1-\beta} \qquad (23)$$

As we can see, q increases with Λ which is the bound of the difference between aggregation values at two successive epochs for characterizing continuity. Larger Λ enables the adversary to create the same degree of pattern distortion within a shorter length series of aggregation results. Therefore, a larger q is required in order to ensure representative point selection from the forged series and thus to preserve the detection probability. With prior knowledge about measured physical quantity, the users can estimate bound Λ and compute the corresponding probability q by Formula (23).

6 Evaluation

In this section, we evaluate the performance of local sample authentication and aggregation verification by simulations. We use Matlab to perform the simulations. To evaluate our local sample authentication approach, we simulate a WSN based on a real world deployment with 54 sensor nodes (ID from 1 -54) in the Intel Research lab, which includes a trace of sensor readings collected between February and April, 2004 [32] and node locations. We suppose the communication radius $R_c = 10m$. The sensors collected time-stamped humidity, temperature and voltage values in 31 second intervals. Excluding two nodes having incomplete data and one node having abnormal data, we use the first 2000 epochs of the data in the day 03/08 from the remaining 51 nodes. We assume a continuous aggregation query on the temperature attribute during the first 2000 epochs. During this period the temperature varies between 20 and 35. The periodic verification is conducted with a time window size $l + 1 = 200$, and 10 time windows are numbered in the order. We note that in the real trace nodes have missing readings in some epochs and we estimate these missing data by linear regression in a time window.

6.1 Performance of Local Sample Authentication

In this section, the representative epochs are uniformly chosen from a time window with an interval of 10 epochs. The performance of the local sample authentication is evaluated by the following two metrics:

- Approval rate of real samples: The ratio of the number of nodes whose data can be successfully authenticated by at least c neighbors to the total number of nodes in the benign environment. Even in benign environment, not all samples would be successfully authenticated in practice because Formula (9a) and (9b) are two statistical conditions and there may be not sufficient neighbors. The samples that cannot be authenticated will not be accepted by the BS. This metric indicates the degree of influence of the local sample authentication on the availability of real samples.
- Disapproval rate of false samples: The ratio of the number of false samples that cannot be successfully authenticated by up to c neighbors to the total number of compromised sampled nodes in the hostile environment. It indicates the degree of the prevention of the false samples by the local sample authentication.

Figure 4 illustrates the approval rate of real samples in each time window under different security threshold c. As we can see, the approval rate in each time window decreases as c increases. This is because the number of nodes having up to c neighbors decreases as c increases. When $c = 1$ and $c = 2$, the approval rate is higher than 90% and 85% respectively. However, the approval rate is lower than 80% when $c = 3$, which is because that the simulated network is sparse (the average degree of the nodes is 5). It indicates that with a reasonable value of c, here say 2, our local sample authentication approach have a small effect on the availability of real samples.

To measure the disapproval rate of false samples, we assume that N_c number of nodes are randomly compromised in the network. We also assume a collusion attack in which a compromised node always provides valid authentication for another one in the neighborhood. The security threshold $t = 2$. The compromised nodes generate false samples in three manners: first, a false sample is generated by adding a constant noise $e = 100$ to each sensor reading in the representative epochs as $r_{u,e_i} \leftarrow r_{u,e_i} + e$; second, $r_{u,e_i} \leftarrow e$ where e is drawn from a uniform random distribution $U(20, 35)$; third, $r_{u,e_i} \leftarrow r_{u,e_i} + e$ where e is drawn from a normal distribution $N(20, 100)$. The first manner preserves the correlation between the real sample and the other ones in the neighborhood. The second manner preserves the amplitude similarity. The third manner brings changes both in the correlation and in the amplitude.

Figure 5(a)~5(c) show the disapproval rate of false samples respectively generated by the above three manners under different N_c. The results are averaged over 50 runs. In each run, N_c nodes are randomly selected as compromised nodes. In each figure, we can see that the disapproval rate of false samples decreases as N_c increases in every time window. This is because that more compromised nodes would incur a higher probability for that a compromised node providing false samples has c compromised neighbor nodes to launch the collusion attack. When $N_c = 1$ and $N_c = 4$, the disapproval rate is higher than 80% in all three figures. Since the network size is small (51 nodes),

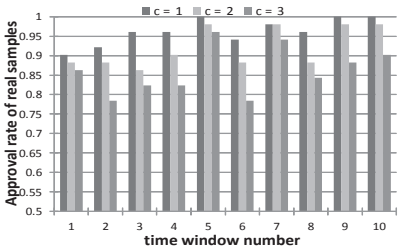


Fig. 4. Approval rate of real samples in every time window.

10 compromised nodes make up a significant fraction of the network and cause the worst results.

6.2 Performance of Aggregation Verification

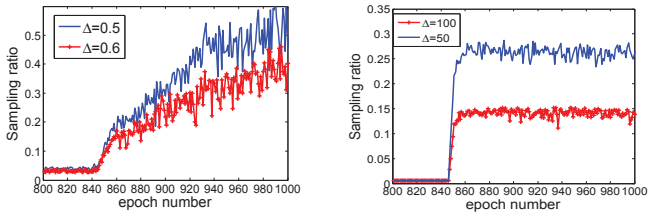
To evaluate our aggregation verification scheme, we simulate a large-scale WSN of 1000 nodes and the sensing readings of each node are synthesized by adding random noises drawn from $N(0, 0.25)$ to the sensor readings of a random node from the above real world deployment. We consider continuous average and predicate count aggregation which counts the number of nodes whose temperature readings are greater than 25 in the time window [800 1000].

We simulate two attacks against the continuous average aggregation and predicate count aggregation in the time window [800 1000] respectively. In the attack against average aggregation, the adversary aims to delay the true time when the temperature rapidly increases. In the attack against predicate count aggregation, the adversary aims to fabricate a false fluctuation of predicate count value.

Figure 6 and 7 show both the real aggregation results and forged ones. Real aggregation results have a rapid increase pattern between (800 900) for both average and count. Figure 8 shows the population variance of temperature in every epoch. For the representative point selection, the total number of representative points $p = 20$. To decide the probability q of pre-specifying random representative points, we investigate the continuity of real aggregation results, and we have that the value difference between two successive epochs falls in $[-0.15, 0.15]$ with probability 99.6% for average aggregation, and in $[-20, 20]$ with probability 98.9% for predicate count aggregation. Then, we let $\Lambda = 0.15$ and $\Lambda = 20$ for two types of aggregation respectively. By Formula (23) with $\gamma = 0.9$ and $\beta = 0.05$, we have $q \geq 0.033$ for average aggregation with $\Delta = 0.5$ and $q \geq 0.043$ for count aggregation with $\Delta = 50$. Hence, we pre-specify each data point in (800 1000) as a representative point with $q = 0.05$.

Figure 6 and 7 also show the representative points selected by RPS-P algorithm. We can see these points well capture the temporal variation in the continuous aggregation. In the figures, some of points are clustered together, which indicates it is not necessary to use as many as 20 number of representative points to capture the patterns. Figure 9 shows the approximation errors with different numbers of representative points. Initially, representative point selection for average aggregation has 12 pre-specified points including randomly selected points, and for count aggregation has 10 pre-specified points. As we can see, the approximation errors are maximum when using only pre-specified points, and more representative points gain little after 13 and 11 numbers respectively. Only one additional point can significantly reduce the approximation error. This is because of single change point in variation patterns. Given the maximum tolerant approximation error, we can determine the minimum number of representative points by the method in Section 4.2.3.

We use $\alpha = 0.02$ significant level in the hypothesis testing. In the simulation, we conduct two round samplings as described in Section 5.2.1. The initial sampling ratio is set to 0.05. The sample variance of first-round collected samples is computed. Based on that, new sampling ratio is computed with $\beta = 0.05$ in Formula (19) and (20), and additional samples are collected if necessary.



(a) Average aggregation. (b) Count aggregation.

Fig. 10. Sampling ratio for average and count aggregation in epochs of [800 1000]

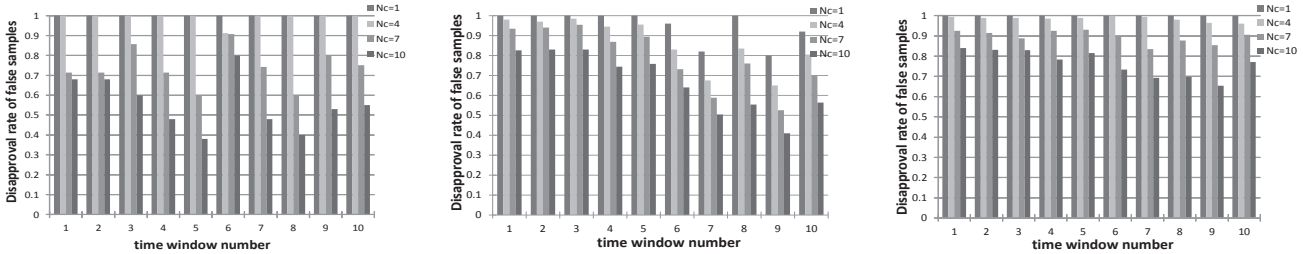
Given different maximum tolerable deviations Δ, Figure 10(a) and 10(b) show the new sampling ratios estimated by Formula (19) and (20) for average and predicate count aggregations in every epoch in [800 1000] respectively. The figures indicate that less samples are required for verification with a larger maximum tolerable deviation. The sampling ratio for average aggregation verification is strongly correlated with the variance of sensor readings, as we can see from Figure 10(a) and Figure 8. In contrast, Figure 10(b) shows that the sampling ratio for predicate count verification has a lower correlation with population variance than for average aggregation, because the sampling ratio for count aggregation verification is decided by the proportion of nodes satisfying the predicate and the aggregation result in every epoch, instead of population variance.

It is worth to note that the sampling ratio between epoch 800 and 850 in Figure 10(b), which is estimated by Formula (19), is not valid, because χ^2 goodness-of-fit test would be failed when the count aggregation is zero or close to zero. As described in Section 4.4.2, we use binomial test to verify the smaller number among two cells and compute the sampling ratio satisfying Formula (15). For $Ag_{c(\theta)}(k) = 0$, we have $m_U = 0$ and $Mg = 0$ in Formula (15), and ϱ should satisfy $\sum_{i=1}^{\Delta} \binom{\Delta}{i} \varrho^i (1-\varrho)^{\Delta - i} \geq 1 - \beta = 0.95$. Then, we derive $\varrho = 0.03$ for $\Delta = 50$, and $\varrho = 0.015$ for $\Delta = 100$. Here the aggregation results claim no nodes having readings larger than 25. Our scheme actually ensures at least one node returns its sample with a high probability if there are at least Δ nodes having readings greater than 25.

We run simulations 100 times in two scenarios of without attacks and under attacks respectively, shown in Figure 6 and 7. We find that our scheme achieves 100% detection rate for the attacks against two types of aggregations. Without any attacks, our scheme incurs about 5% and 10% false alarm rate for predicate count aggregation with $\Delta = 50$ and $\Delta = 100$ respectively. For average aggregation, our scheme incurs about 4% and 7% false alarm rate with $\Delta = 0.5$ and $\Delta = 0.6$ respectively. We can see that a bigger Δ causes a higher false alarm rate. This is because the estimated sampling ratio decreases when Δ increases, and less samples cause higher probability of Type I error. Our results indicate that the false alarm rate can be decreased by choosing a smaller significant level $\alpha = 0.01$.

7 Conclusion

In this paper, we identify distinct design issues for secure continuous aggregation in WSNs. An efficient verification scheme is proposed to protect the authenticity of the temporal variation patterns in the aggregation results. Compared with the existing secure aggregation schemes, our scheme only need to check a small portion of aggregation results in a time window and thus greatly reduces the verification cost. We define representative points and propose corresponding algorithms for representative point selection. By exploiting the spatial correlation among the sensor readings in close proximity, a series of security mechanisms are also proposed to protect the sampling procedure. Our simulations validate our scheme design.



(a) Disapproval rate of false samples generated by the first manner.

(b) Disapproval rate of false samples generated by the second manner.

(c) Disapproval rate of false samples generated by the third manner.

Fig. 5. Disapproval rate under three manners of forging false samples.

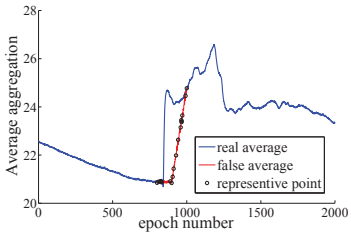


Fig. 6. Continuous average aggregation under attack.

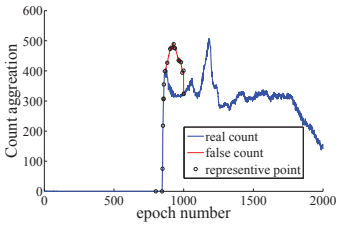


Fig. 7. Continuous count aggregation under attack.

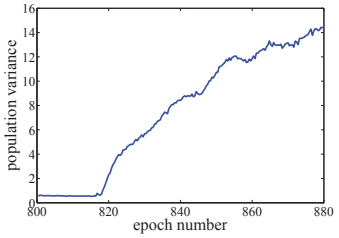


Fig. 8. Population variance in each epoch of [800 1000].

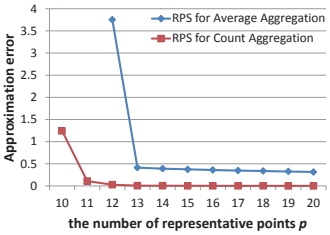


Fig. 9. The number of representative points V.S. approximation error

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-CSR 1025649,OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282, and in part by the National Grand Fundamental Research 973 Program of China under grant 2012CB316200, the Major Program of National Natural Science Foundation of China under grant 61190115, the Key Program of the National Natural Science Foundation of China under grant 61033015, and the National Natural Science Foundation of China under grant 60933001.

References

[1] Z. Cai, S. Ji, J. S. He, and A. G. Bourgeois, "Optimal distributed data collection for asynchronous cognitive radio networks," in *IEEE ICDCS*, 2012, pp. 245–254.

[2] S. Ji and Z. Cai, "Distributed data collection and its capacity in asynchronous wireless sensor networks," in *IEEE INFOCOM*, march 2012, pp. 2113 –2121.

[3] S. Ji, R. Beyah, and Z. Cai, "Snapshot/continuous data collection capacity for large-scale probabilistic wireless sensor networks," in *IEEE INFOCOM*, march 2012, pp. 1035 –1043.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *MobiCom*. New York, NY, USA: ACM, 2000, pp. 56–67.

[5] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.

[6] K.-W. Fan, S. Liu, and P. Sinha, "On the potential of structure-free data aggregation in sensor networks," in *IEEE InfoCom*, 2006, pp. 1–12.

[7] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and deltas: Efficient and robust aggregation in sensor network streams," in *ACM SIGMOD*, 2005, pp. 287–298.

[8] B. Przydatek, D. Song, and A. Perrig, "SIA: secure information aggregation in sensor networks," in *ACM SenSys*, 2003, pp. 255–265.

[9] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP:a secure hop-by-hop data aggregation protocol for sensor networks," in *ACM MobiHoc*, 2006, pp. 356–367.

[10] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *ACM CCS*, 2006, pp. 278–287.

[11] K. B. Frikken and J. A. Dougherty, IV, "An efficient integrity-preserving scheme for hierarchical sensor aggregation," in *ACM WiSec*. New York, NY, USA: ACM, 2008, pp. 68–76.

[12] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *IEEE INFOCOM*, 2009, pp. 2159 –2167.

[13] D. Wagner, "Resilient aggregation in sensor networks," in *Proc of the 2nd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2004, pp. 78–87.

[14] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *Workshop on Security and Assurance in Ad hoc Networks*. IEEE Computer Society, 2003, p. 384.

[15] H. Yu, "Secure and highly-available aggregation queries in large-scale sensor networks via set sampling," in *ACM/IEEE IPSN*, 2009.

[16] M. Garofalakis, J. Hellerstein, and P. Maniatis, "Proof sketches: Verifiable in-network aggregation," in *IEEE ICDE*, april 2007, pp. 996 –1005.

[17] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Securely computing an approximate median in wireless sensor networks," in *Proc of the 4th international conference on Security and privacy in communication netowrks*. ACM, 2008, pp. 6:1–6:10.

[18] P. Flajolet, G. N. Martin, and G. N. Martin, "Probabilistic counting algorithms for data base applications," 1985.

[19] S. Ganeriwal, S. Capkun, C. chieh Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc of the 4th ACM workshop on Wireless security*. ACM Press, 2005, pp. 97–106.

[20] K. Sun and P. Ning, "Tinysersync: secure and resilient time synchronization in wireless sensor networks," in *ACM CCS*, 2006, pp. 264–277.

[21] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *ACM CCS*, 2003, pp. 62–72.

[22] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *ACM CCS*, 2003, pp. 52–61.

[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Wireless Networks*, 2001, pp. 189–199.

[24] H. Gupta, V. Navda, S. R. Das, and V. Chowdhary, "Efficient gathering of correlated data in sensor networks," in *ACM MobiHoc*, 2005, pp. 402–413.

[25] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *ACM/IEEE IPSN*. New York, NY, USA: ACM, 2004, pp. 28–35.

[26] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE International Conference on Communications*, vol. 2, 2001, pp. 472–476.

[27] F. Liu, X. Cheng, and D. Chen, "Insider attacker detection in wireless sensor networks," in *IEEE InfoCom*, May 2007, pp. 1937–1945.

[28] M. C. Vuran and I. F. Akyildiz, "Spatial correlation-based collaborative medium access control in wireless sensor networks," *IEEE/ACM Trans on Networking*, vol. 14, no. 2, pp. 316–329, 2006.

[29] T. H. S. C. Yu, R.C. and J. Froines, "Quality control of semi-continuous mobility size-fractionated particle number concentration data," *Atmospheric Environment*, vol. 38(20), pp. 3341–3348, 2004.

[30] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, February 1969.

[31] R. G. M. Bellare and P. Rogaway, "Xor macs: New methods for message authentication using finite pseudo-random functions," in *Proc. of Crypto*, 1995.

[32] "Intel lab data," in *http://berkeley.intel-research.net/labdata*.

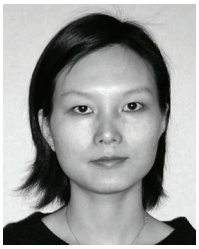
[33] P. S. Mann, *Introductory Statistics*. Wiley, 2006.

Lei Yu received the PhD degree in computer science from Harbin Institute of Technology, China, in 2011. He currently is a post doctoral research fellow in the Department of Electrical and Computer Engineering at Clemson University, SC, United States. His research interests include sensor networks, wireless networks, cloud computing and network security.

Jianzhong Li is a professor in the School of Computer Science and Technology at Harbin Institute of Technology, China. In the past, he worked as a visiting scholar at the University of California at Berkeley, USA, as a staff scientist in the Information Research Group at the Lawrence Berkeley National Laboratory, USA, and as a visiting professor at the University of Minnesota, USA. His research interests include data management systems, sensor networks and data intensive computing. He has published more than 150 papers in refereed journals and conferences, and has been involved in the program committees of major computer science and technology conferences, including SIGMOD, VLDB, ICDE, INFOCOM, ICDCS, and WWW. He has also served on the editorial boards for distinguished journals, including TKDE.

Siyao Cheng received the PhD degree in computer science from Harbin Institute of Technology, China, in 2012. Her research interests include data management and wireless communication in sensor networks.

Shuguang Xiong received the PhD degree in computer science from Harbin Institute of Technology, China, in 2011. He is currently a software engineer in Baidu inc, Beijing, China. His research interests include data management and networking in wireless ad hoc and sensor networks.



Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on peer-to-peer and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the Program Co-Chair for a number of international conferences and member of the Program Committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.