Received 21 August 2022; revised 14 September 2022; accepted 17 September 2022. Date of publication 22 September 2022; date of current version 6 October 2022.

Digital Object Identifier 10.1109/OJCOMS.2022.3208088

False Event Message Detection Robust to Burst Attacks in Wireless Sensor Networks

YUICHI SEI [1,2] (Member, IEEE), AND AKIHIKO OHSUGA [1] (Member, IEEE)

[1] Department of Informatics, Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo 182-8585, Japan

[2] JST, PRESTO, Kawaguchi, Saitama 332-0012, Japan

CORRESPONDING AUTHOR: Y. SEI (e-mail: seiuny@uec.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science KAKENHI under Grant JP21H03496 and Grant JP22K12157, and in part by the Japan Science and Technology Agency (JST), Precursory Research for Embryonic Science and Technology, Japan, under Grant JPMJPR1934.

ABSTRACT A sensor device can be used to detect target events at low cost. Moreover, there is a significant risk of sensor nodes being compromised or captured within large wireless sensor networks (WSNs). The transmission of valid event messages to users by a WSN can be hindered by network congestion due to false event messages in a compromised node. Several methods are available for detecting false event messages. However, they rely on long bits of authentication codes and hence do not provide a fundamental solution to prevent network congestion. In the proposed method, various hashing vectors, which are space-efficient data structures that can determine whether the given data are an element of a set, are created based on the correct combination of authentication codes and placed in each node in advance. An event message contains an XOR of the authentication codes, and each node verifies it based on its hashing vector. If a node is acquired illegally, the information of the hashing vector and the XOR information of the authentication codes assigned to the correct event message is compromised, so we propose an algorithm to update the information securely. Compared to existing research, the number of hops required to detect a false event message increases by only about one hop, but the amount of traffic that a malicious node can generate can be reduced by about 60% or more. In other words, the proposed method effectively reduces the amount of traffic an attacker can generate with false event messages, which also reduces the overall network congestion.

INDEX TERMS Energy-efficient protocol, false event detection, security, wireless sensor networks.

I. INTRODUCTION

AS A CORE functionality, sensor devices identify and report events, e.g., trespassing or forest fires, and a wireless sensor network (WSN) is formed by connecting such sensor devices in a wireless network [1], [2], [3], [4]. Multihop wireless paths serve as the medium for the message in a WSN to reach its last recipient, the sink, following the detection of events of interest and transmission of the message by the sensor nodes (Fig. 1). Nonetheless, when sensor nodes are used in hostile environments, they become susceptible to attacks. An attacker can compromise and capture multiple sensors. Network congestion can occur, as such compromised nodes can generate numerous false event messages (hereinafter, *false messages*) following the obtention of the secret keys stored in these nodes (Fig. 2). This network congestion can prevent the transmission of a valid message to the sink, thereby obscuring the hacker's crime.

The rapid identification of false messages in-network is a concept explored in related studies. Network congestion can be partially addressed via the detection at an early stage. Many works tackle this challenging issue [5], [6], [7], [8]. Permitting each node to possess symmetric keys encapsulates the fundamental idea. Multiple message authentication codes (MACs) are contained in a report collectively produced by T proximal sensor nodes upon an event occurring. A node's agreement with the report is reflected by a MAC that is produced by a node using one of its symmetric keys. Further authentication information is incorporated into every message using multiple methods. The probabilistic validation of MACs contained in the report is enabled

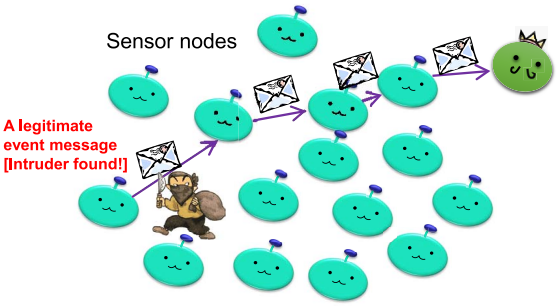

FIGURE 1. An example of detecting an event in a WSN.

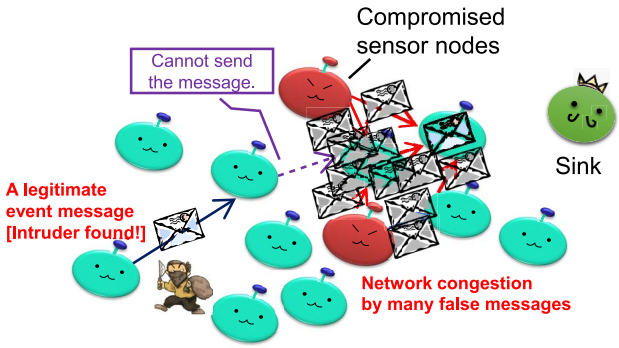

FIGURE 2. An example of false message attacks.

by every forwarding node within the process of a report being forwarded to the sink over several hops. Delivery does not occur when insufficient MACs are contained in a given report. As every message comprises significant quantities of authentication information, the quantity of traffic in every message is substantial despite their ability to identify false messages in-network. Therefore, the production of multiple false messages caused by a compromised node causes network congestion. Further, a significant proportion of the battery of sensor nodes is consumed due to legitimate messages. Consequently, the lifespan of a WSN diminishes as the sensor nodes function with a restricted battery. In addition, when more than T nodes are compromised, certain methods cause a total breakdown of security.

When more than T nodes are compromised, a streamlined method proposed in this study demonstrates a high likelihood of success in the detection of false messages in-network. Although this method significantly decreases the probability of network congestion, as the message length can be substantially reduced, this contemporary, cutting-edge method generates almost the same total message hops before detecting a false message.

Before the deployment, the sensor nodes are preloaded with developed information regarding an exclusive OR code (EOC), which enhances false event detection in typical cases and cases in which the sensor nodes leak more than T nodes. The proposed method's ability to reduce the total traffic when compromised nodes generate numerous false messages is evidenced by our mathematical analysis.

The main contributions of the paper are as follows.

1) First, we propose an algorithm to detect false event messages in WSNs with high probability. In our algorithm, each message contains event information and a one-time authentication code (EOC; exclusive OR code). The bit length of the proposed one-time authentication code is shorter than the authentication information used in existing studies. Thus, the required amount of traffic required for event messages can be reduced compared to existing methods. As a result, the proposed algorithm can prevent network congestion caused by malicious nodes. As mentioned previously, the goal of this study is to reduce traffic volume and network congestion in WSNs.
2) Second, each forwarding node can determine whether the one-time authentication code in the forwarding message is legitimate based on its hashing vector. A hash vector is a data structure that can determine whether the given data is an element of a set, thereby allowing a few false positives and achieving high spatial efficiency. The network manager can control the detection ability of hashing vectors. Here, when the target detection probability is set to 0.5, a false event message can be detected in two hops on average.
3) Third, an update mechanism of the one-time authentication code is also proposed. This update mechanism disables one-time authentication codes that the attacker may have obtained and updates them to new codes that cannot be predicted by the attacker. The update mechanism is a novel feature required only for the proposed method, and the ability of the proposed method to detect false messages using one-time authentication codes is almost even with existing methods. Note that the update mechanism does not require much traffic, and the overall traffic volume of the proposed method, including the update mechanism, is very low compared to existing methods.
4) Fourth, we conducted experiments and compared the proposed algorithm with four existing algorithms. The experimental results demonstrate that the proposed method can reduce the maximum amount of traffic an attacker can produce.

We present the subsequent sections of this study in the following manner: we portray a model of sensor networks and false events in Section II, explore related methods and their issues in Section III, illustrate the design of our method in Section IV, explore the efficacy of our method alongside the parameter settings in Section V (the findings from the previous section serve as the basis for assessments), and summarize this study in Section VII. The proposed method reduces the amount of traffic required to detect false event messages compared to existing studies while maintaining the ability to detect false event messages.

II. RELATED WORK

Secured data aggregation in WSNs has been a subject of inquiry in various studies [9], [10]. Although in-network

false events cannot be detected via the methodology in these studies, false events at the sink can be detected. Therefore, a WSN cannot detect trespassing or other such events due to network congestion compromised nodes cause.

The distribution of secret keys to sensors is enabled via a method proposed in studies focusing on in-network false event detection. Each node possessing symmetric keys encapsulates this method's fundamental concept. Each node's location serves as the basis for producing keys following deployment, or each node is preloaded with the keys. In addition, certain nodes share the same key. The MACs, key IDs, and IDs of sensors proximal to an event are carried in a report these sensors collectively produce. A node's agreement with the report is reflected by a MAC that a node produces using one of its symmetric keys. The probabilistic validation of the MACs carried in the report is enabled by every forwarding node within the process of a report being forwarded to the sink over several hops. Delivery does not occur when reports contain false MACs. The related literature endeavors to enhance the likelihood that a forwarding node possesses a key used to generate MACs in the event message.

Keys are randomly distributed to sensor nodes within false message detection methods with randomized key distribution mechanisms [5], [11]. A legitimate event can only be generated when the sensor nodes collect more than T discrete MACs. This mechanism can be used in a scenario in which the sink moves. Nonetheless, threshold behavior is displayed by the mentioned randomized key distribution mechanism. When the hacker or attacker has more than T keys, the detection of false events cannot occur.

Each sensor node's location serves as the basis for distributing keys to every node within false message detection methods using a location-based key distribution mechanism [6], [7], [8], [12]. The locations of the sink and nodes enable the static forwarding node identification for every source node. A burst of large quantities of false messages is generated by the above-mentioned methods despite their ability to detect false messages in-network, attributed to the long messages that compromised nodes can generate in such contexts. The default of $T = 5$ has been established in various existing studies. In WSNs, 8 bytes is typically thought to be the length of a MAC [13].

By contrast, 40 bytes is considered the standard packet size without security mechanisms in WSNs in some studies [14], [15]. Therefore, compromised nodes can generate a size that doubles the original packet in a false message, reaching up to 80 bytes in length. A method for detecting false messages effectively handles scenarios that involve the location of the sink altering and the presence of multiple compromised nodes in a certain study [7]. Nevertheless, as the method uses T MACs, attacks in the form of bursts of large quantities of false messages occur, similar to other relevant studies.

Babu *et al.* proposed Secure Data Aggregation based on the Principle Component Analysis (SDA-PCA) algorithm [9]. A cluster head is selected in their algorithm among neighboring sensor nodes according to node quality and energy level. The cluster head gathers sensing information of the surrounding sensor nodes, compresses the information, and then sends a message to the sink. Sensor nodes with high mobility are considered malicious nodes in SDA-PCA. Such nodes are not allowed to become cluster heads. This mechanism prevents false event messages from reaching the sink. Since the criteria for determining whether a node is malicious are clear, it is easy to place a malicious node so that it does not meet these criteria. Therefore, a malicious node may become a cluster head. It is also possible for a cluster head to provide much incorrect information. Since the algorithm assumes that the messages generated by the cluster head are correct, it cannot detect when a cluster head creates a false event message.

Wang *et al.* proposed a secure aggregation protocol in WSNs [10]. The protocol selects a cluster head among neighboring nodes based on link quality and remaining energy. Data confidentiality is protected because communications between nodes and cluster heads and between cluster heads and sinks are encrypted. Moreover, each node and each cluster head have a unique key, preventing spoofing. However, since only the sink can detect this spoofing, the spoofing cannot be detected while transferring the message to the sink. Therefore, if an attacker compromises even one node, he can generate many false event messages that the in-network cannot detect.

Furthermore, Pedroso and Santos [16] suggested a clustering algorithm (CONsensus Based Data FIlteriNg for IIoT; CONFINIT) that prevents false message attacks. In that algorithm, each node compares its sensing values with those of other nodes. If a node's sensing values differ significantly from those of other nodes, the node is likely malicious. In this case, other honest nodes ignore the potentially malicious node. As a result, CONFINIT tries to prevent malicious nodes from generating false event messages. However, if a malicious node succeeds in generating a false event message, it is not detected as a false event message because there is no mechanism to detect false event messages at the forwarding node or sink.

The methods introduced so far can prevent the generation of false event messages to some extent or detect them in the sink. However, they cannot detect false event messages in-network, i.e., at the forwarding node, which is the goal of this paper.

Ye *et al.* proposed a Statistical En-route Filtering (SEF) mechanism [5]. They proposed an algorithm that pioneered detecting false event messages in the network. Sensor nodes that detect the target event generate MACs based on the event content and node ID, and the cluster head, at minimum, collects T MACs. Keys for MAC generation are randomly deployed to each node and information about which node holds which key is not maintained. Thus, the problem is that with only T keys compromised, an attacker can generate any number of false event messages from any location.

By distributing "legitimate key ID combinations" to each node in advance, an in-network detection method was proposed for messages generated with illegitimate keys [7]. In their approach named MobiSink, when a message is forwarded to the sink, each forwarding node checks to see if the message has at least T MACs with the correct key IDs appended. The detection rate of false messages is high, but the disadvantage is that the message length becomes very long because T MACs and T key IDs must be appended to every message.

A combinatorial design-based partial en-route filtering scheme (CD-PEFS) was proposed by Kumar *et al.* [8]. In CD-PEFS, three cluster heads are selected in each cluster. One cluster head gathers sensing information from the surrounding sensor nodes and generates a message with T secret shares, T node IDs, and $(k'+1)$ MACs where k' represents the number of keys stored in cluster heads. The value of k' is 10 on average when the number of clusters is 150. As the number of clusters increases, the value of k' also increases. Although the detection rate of false event messages of CD-PEFS is very high, there is a great deal of data added to the message.

Yi proposed an en-route message authentication scheme (EMAS) [12]. In EMAS, each sensing node sends the information with event data, key ID, location ID, and MAC to a cluster head. The cluster head maps T MACs into a compressed filter using hash functions. The cluster head generates a message with the compressed filter, T node IDs, T location IDs, and T key IDs. Each forwarding node can detect false event messages based on the compressed filter, node IDs, location IDs, and key IDs. Although EMAS compresses T MACs into one filter, the amount of messages is still significant because T node IDs, T location IDs, and T key IDs are necessary for the authentication.

As mentioned above, all existing research aimed at detecting false event messages in-network adds many data to the event messages for authentication. False event messages generated by compromised nodes are detected in the in-network early. However, the amount of data in each message is so large that the composed nodes can easily cause network congestion. These four algorithms [5], [7], [8], [12] are compared with the proposed algorithm in Section V.

The suggested method uses one-time authentication codes instead of a large amount of authentication, such as T key IDs and T MACs, which significantly reduces the amount of data for authentication added to messages. This one-time authentication is a novel idea not found in existing research. Of course, since one-time authentication is used, an update mechanism is required. We propose an algorithm to perform this mechanism efficiently and securely.

III. SYSTEM MODEL

We outline the model of false event attacks and the sensor network model used in this section.

A. SENSOR NETWORK MODEL

Multiple small sensor nodes deployed at high density comprise the sensor network examined in this study. Adaptation to node failures and enhanced detection accuracy are enabled using multiple nodes to detect an event. A node is selected to be the center-of-stimulus node (CoS) while all detecting nodes report the signals they detect. The CoS collects and summates all received detection findings. A multihop path is used by the event message, which is a term used to describe the message the CoS delivers to the sink. All nodes use a localization scheme to identify their geographic location once they are deployed [17].

Given that the sensor nodes' design followed cost-effectiveness principles, we presume that they do not possess tamper-resistant hardware. Following deployment, we also presume that no movement occurs in the nodes. The literature reflects these suppositions [18], [19], [20], [21].

Symmetric-key cryptography serves as the basis for most WSNs due to the computationally expensive nature of cryptographic asymmetry based on public-key schemes [22].

In addition, we presume that every node can validate the sink's message, and the sink can validate its messages to the sensor nodes via a secure mechanism, such as μTESLA, thereby ensuring that the sink cannot be compromised [23].

We assume that the number of legitimate events is relatively small, e.g., a few times per month. The target events are assumed to include forest fires and intrusions by attackers, and such events do not occur frequently.

Possible applications for the proposed method include intruders, arson, theft, and other application scenarios that detect human crime. These applications are one of the example scenarios in WSNs. In addition, human crime is not expected to occur frequently. Thus, the proposed method works effectively for such applications. In other words, the amount of traffic required by the proposed method required is very low compared to existing methods. If the frequency of events is high, the number of updates of the one-time authentication code increases, and the amount of traffic required by the proposed method increases. This reduces the advantage of the proposed method in that it requires less traffic than the existing methods.

B. ATTACK MODEL

A network can have several sensor nodes compromised by a hacker within it. An attacker can load secret keys attained from distinct nodes onto a compromised node following the exposure of all codes, data, and secret keys resulting from a sensor node being compromised. Compromised nodes can seemingly detect a proximal event when no such event exists. These false reports, which can cause a user to make harmful decisions, can induce network congestion. We assume that WSNs are equipped with replicated node detection methods [24]. In other words, if an attacker takes more than one node, he/she has complete control over that node but cannot replicate many malicious nodes.

TABLE 1. Notations.

Notation	Description
n_i	Sensor node whose ID is i
k_i	n_i's key shared with the sink
$k_{i,j}$	Key shared with nodes n_i and n_j
r_i	n_i's code
T	Required number of nodes to generate a legitimate event message
q	Number of codes assigned to nodes in a grid
g	Number of grids
N	Number of nodes
v	The average number of occurrences of events per month
D	Maximum number of event messages in a grid per second
m	The bit length of a hashing vector
b	The bit length of a code
$\|E\|$	The bit length of an event description
FPR_t	The target false-positive rate of a hashing vector

An attacker can generate arbitrary event messages. However, the forwarding node may detect an attacker-generated event message as illegitimate since each forwarding node checks whether the event message is legitimate. The attacker is free to use the information from the compromised node to mislead the forwarding node into assuming that the attacker-generated event message is legitimate. Although an attacker can compromise multiple nodes, we assume that the number of compromised nodes does not exceed 100 since it is difficult to realistically expect an attacker to compromise more than 100 nodes without the system administrator being aware. An attacker can also intercept communications between normal nodes.

IV. METHOD

A. SUMMARY OF THE PROPOSED METHOD

Table 1 lists the notations used in this study. We used a regular geographic grid to separate the terrain. T nodes in the same grid can generate an event message.

The variable i represents the n_i's node ID, and n_i represents the sensor node. The sink generates code r_i available only once for each node n_i in advance. When creating an event message, nodes cooperate to create exclusive OR code (EOC) R from T codes in the same grid. Please note that the sink can know all possible legitimate combinations of T codes in advance. In other words, the sink knows all possible legitimate EOCs in advance. Even if an attacker compromises T nodes, the attacker cannot create a legitimate EOC when all compromised nodes are not on the same grid because a legitimate EOC needs to be created from T nodes in the same grid.

Each node is preloaded with a hashing vector created from all legitimate EOCs. Since the hashing vector can check whether an element is included in the set from which the hashing vector was generated, each forwarding node can detect a false event message (i.e., an event message with nonlegitimate EOC) with a certain probability.

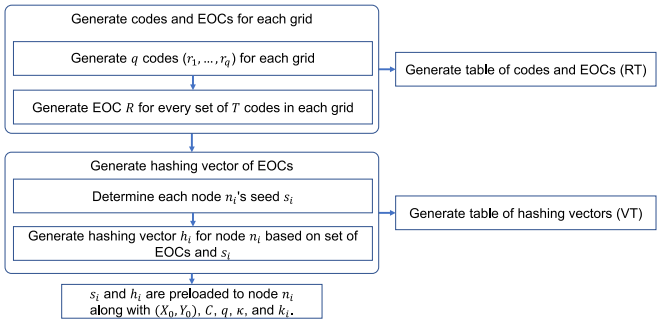

FIGURE 3. Overall process of the proposed algorithm at the sink in advance.

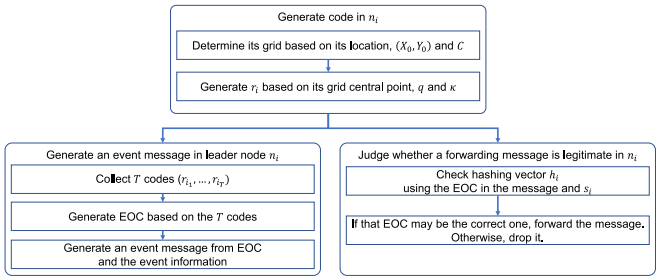

FIGURE 4. Overall process of the proposed algorithm in each node.

TABLE 2. Example of table of codes and EOCs (RT).

Grid ID	EOC number per grid	Set of T code	EOC
1	1	{001100..., 010110..., ...}	R_1 = 011110...
...	...	...	...
1	${}_qC_T$	{110110..., 110011..., ...}	$R_{{}_qC_T}$ = 111001...
...	...	...	...
g	1	{011101..., 110110..., ...}	$R_{(g-1)*{}_qC_T+1}$ = 001000...
...	...	...	...
g	${}_qC_T$	{011101..., 110110..., ...}	$R_{g*{}_qC_T}$ = 001000...

Here, there are two main challenges. First, if all nodes have the same hashing vectors, the detecting ability is the same among all hashing vectors. Hence, if a false event message is not detected at the first hop, it is not detected until the end. The second challenge is that the EOC information of a legitimate event message could be leaked to the attacker. Therefore, the update mechanism is necessary.

This process and the relationship between the process, table of codes and EOCs (RT), and table of hashing vectors (VT) are shown in Figs. 4 and 5, respectively. RT and VT are used to generate hashing vectors in advance and update hashing vectors, respectively.

Examples of RT and VT are shown in Tables 2 and 3. The following section describes the detailed process of the proposed algorithm. It also describes an approach to solving two issues.

The number of codes assigned to nodes in a grid is represented by $q(q \geq T)$, and g represents the number of grids. A node is given to each of the prepared qg codes—$r_1, r_2, \ldots, r_{qg}$. The variable b represents the codes' bit length. Methods involving local leader election [25], such as an event's occurrence, identify the leader node, n_w, within the nodes in a grid, G_w. The leader node, n_w, receives

TABLE 3. Example of table of hashing vectors (VT).

Node ID	Seed	Hashing vector
1	s_1 = 83a1f0cc	v_1 = 00101001...
2	s_2 = ba1422de	v_2 = 01000000...
...		
N	s_N = 940ffa8c	v_3 = 00110000...

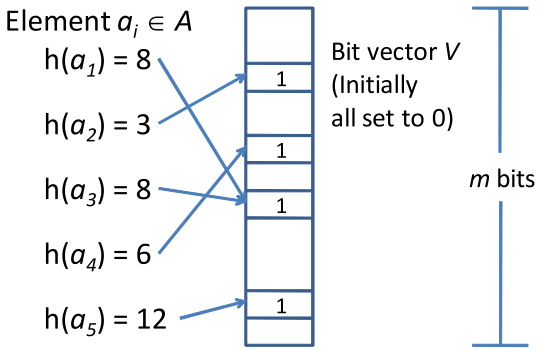

FIGURE 5. A hashing vector's execution process.

a report from its subordinate nodes when they detect an event. By adopting the XOR operation of the T codes selected randomly from the received keys, the leader node generates EOCs.

Determining whether an EOC is valid is facilitated by the information each node receives. In the scenario in which an attacker compromises more than T nodes, the false event in-network can be detected. Nonetheless, as aforementioned, the method proposed in this study and all existing related studies cannot detect false messages when a hacker compromises a T node in a grid. Nonetheless, location-based key distribution or the mechanisms used in this study prevent the establishment in other grids of the message's occurrence point.

The codes are used to generate an EOC. In other words, the EOC is generated by the XOR operation of T codes assigned to T nodes. The same code cannot be used again, as the forwarding of the message containing the EOC could be a compromised node. Therefore, the codes used to generate the EOC must be updated via a suitable method.

B. PROCESSES

First, each node creates its code (Section IV-B1). This code is used to generate a one-time authentication code (EOC) when nodes detect events. Each node stores a hashing vector (Section IV-B2) created from legitimate set of EOCs in advance at the sink node. Here, when an event is detected, several nodes collaboratively generate an event message containing an EOC (Section IV-B3). A forwarding node then determines whether the event message is legitimate using the hashing vector stored in the node and the EOC attached to the event message (Section IV-B4). Finally, the sink determines if the received message is correct (Section IV-B5). The details of this process are described in the following.

1) CREATION OF CODES

The variable G_w represents a grid with an ID of w. Each node n_i has a symmetric key k_i, which is shared with the sink. After deployment, node n_i generates cord r_i based on the ID of the grid where node n_i exists. The variable g represents the number of grids. Each grid has q codes. One of these codes is given to each node in the grid.

The grid that nodes are in determines the codes the nodes can create based on the proposed method. C and (X_0, Y_0) denote the grid size and grid's reference point, respectively. A reference point can encapsulate an arbitrary point. Grid G_i can be expressed as follows when (X_{x_i}, Y_{y_i}) is its central point:

$$\begin{aligned} X_{x_i} &= X_0 + x_i \cdot C \\ Y_{y_i} &= Y_0 + y_i \cdot C \\ (i, j &= 0, \pm 1, \cdots .). \end{aligned} \tag{1}$$

The nodes are preloaded with the reference point (X_0, Y_0), a hash function, h, a shared master secret key, κ, and grid size, C. A method also used in related studies allows the location of each node to be identified following deployment [26]. A noncomplex calculation allows each node to identify its grid's central point (X_{x_i}, Y_{y_i}).

The equation below enables the calculation of the code r_i of node n_i:

$$r_i = h\big(\kappa || X_{x_i} || Y_{y_i} || \Phi(i) \bmod q\big). \tag{2}$$

The rank of ID i among all node IDs in ascending order in the same grid is conveyed via the variable $\Phi(i)$, and concatenation is represented by $||$.

The node ID is broadcasted by each node n_i to one another in the grid G_w following deployment. The rank of the node n_i's ID i is ascertained by node n_i, which assembles IDs in ascending order.

Every pair of neighboring nodes shares a unique pairwise key. The variable $k_{i,j}$ represents the pairwise key of nodes n_j and n_i.

2) HASHING VECTOR

Every node uses a hashing vector to generate information about the EOCs. A Bloom filter is an equivalent concept to that of a hashing vector [27]. $H[A]$ denotes the denomination used to represent the hashing vector generated from set A. Using α and $H[A]$, where the former represents an element, α is a member of set A that can be ascertained. Nonetheless, false positives can occur, as α is not a member of A, yet $H[A]$ states that α is a member of A.

Fig. 5 portrays a hashing vector's execution process. The hash function h with a range of $1, \ldots, m$ is selected following the assignment of a vector, V, of m bits, all initially set to 0.

The bit at position $h(a)$ in V is set to 1 for every element $a \in A$ where a bit could be set to 1 many times. The bit position $h(\alpha)$ is checked when a query for α is made. It is determined that α is not in set A when the value is 0. Although the following inference could be incorrect, when

the value is not 0, α is considered to be in the set. If incorrect, the result is a false positive. An acceptable probability of producing a false positive can be facilitated by selecting a suitable parameter, m.

After hashing all n elements of $a \in A$ into a hashing vector, the probability that a particular bit is still 0 is $(1 - 1/m)^n$. Based on the definition of the natural logarithm, we have $\lim_{m\to\infty}(1 + 1/m)^m = e$. That is, for large m, we have $(1 + 1/m)^m \approx e$. Therefore, $(1 - 1/m)^n = ((1 + 1/(-m))^{-m})^{-n/m} \approx e^{-n/m}$. That is, the following equation demonstrates a hashing vector's false-positive rate (FPR):

$$\Re(n, m) \approx e^{-n/m}. \tag{3}$$

The length of the hashing vector's bits and the number of elements in a target set are represented by m and n, respectively. The target FPR is set to FPR_t. By solving the equation of $\mathrm{FPR_t} = e^{-n/m}$ with regard to m, we have

$$m = \left\lfloor -\frac{n}{\ln(FPR_t)} \right\rfloor. \tag{4}$$

Before deployment, hashing vectors are created at the sink from all possible EOCs that may be generated in the WSN. The specific EOC preparation is described in the next subsection. Provide each hashing vector to each node before deployment. The collective filtering power across the forwarding can be substantial, even if the number of false positives is significant. However, indeed, it does not make sense to have the same hash vector at all nodes. Therefore, each hashing vector v_i at node n_i is generated by a unique seed, s_i. Further, the hashing vector, v_i, is generated and checked using $h(s_i||\alpha)$ instead of $h(\alpha)$ as a hash value of α. EOCs are equivalent to elements in the proposed method. Pseudorandom number generators are used to generate s_i [28].

3) REPORT GENERATION

When an event occurs in grid G_w, $\{r_i, E\}$ embodies the form an event report takes after every node n_i that identifies the signal and prepares such a report. The description of the event is denominated by the variable E. A node n_i creates a message described as follows:

$$M_i = Enc\big(k_{i,u}, r_i\big)||E. \tag{5}$$

The encryption of the message, r_i, using key $k_{i,u}$ is represented by the variable $Enc(k_{i,u}, r_i)$. The ID of the leader node of grid G_w and the stream concatenation are represented by u and $||$, respectively. Subsequently, node n_i delivers M_i to n_u, the leader node. The leader node extracts and gathers all $Enc(k_{i,u}, r_i)||E$ located in the detecting nodes. T reports are selected randomly when the leader node collects more than T reports. The term $M_{i_1}, \ldots, M_{i_T}$ is representative of the T reports. In addition, the term $r_{i_1}, \ldots, r_{i_T}$ is attained after the reports are decrypted by node n_u, which then creates the following:

$$R_w = r_{i_1} \oplus r_{i_2} \ldots \oplus r_{i_T}. \tag{6}$$

The XOR operation is denoted by $\oplus$. The resulting R_w is the grid's EOC.

The leader node's neighbor is denoted by node n_v in the final report to the sink, represented by $M = E||Enc(K_{u,v}, R_w)$ and delivered by the leader node.

4) EN-ROUTE FILTERING

When node n_j sends the event message $M = E||Enc(K_{i,j}, R_w)$ to node n_i, the decryption of $Enc(K_{i,j}, R_w)$ enables node $n_i$to attain R_w. Subsequently, the node uses the hashing vector to determine whether the EOC R_w is valid. The message is dropped if the node deems the EOC R_w false. The message is delivered to the next node, n_s, when the node determines that the EOC R_w might be legitimate.

5) SINK VERIFICATION

Given that all information about EOCs is contained in the event message delivered to the sink, the sink can validate the EOCs. A message is considered generated by a hacker when it fails this validity assessment.

C. CODE UPDATE

The codes used to generate an EOC in a message must be updated if a legitimate message is delivered to the sink. This must occur, as there is a chance that the EOC was learned by a compromised node among one of the forwarding nodes of the message. Given that false messages can solely be generated from the grid in which an event occurs by an attacker cognizant of the EOC, a slight delay in updating the keys is not considered a significant issue.

To update EOCs' codes, the sink regulates multiple types of data.

1) DATA REGULATED AT THE SINK

The sink regulates the two tables presented below.

RT (Table of codes and EOCs): Let RS be the set of all EOCs that can be generated in the WSN. The variable R_i represents an element of RS. The codes $r_{i_1}, \ldots, r_{i_T}$ generate R_i. The ID of the grid apportioned R_i and R_i is linked to the above-mentioned T codes via RT.

VT (Table of hashing vectors): Before deployment, every node n_i has a hashing vector, v_i, assigned to it, where the sink generates the latter. A random seed, s_i, is created and used upon the sink generating the hashing vector, v_i. Node n_i is linked to each seed s_i and hashing vector v_i via VT.

2) UPDATING HASHING VECTORS AND CODE PROCEDURE

When grid G_w delivers a legitimate message to the sink with the R_w EOC, the sink generates the ID w of the grid apportioned R_w and R_w as the sink learns the T codes $(r_{i_1}, \ldots, r_{i_T})$ via RT.

The sink generates novel EOC $R'_w = r'_{i_1} \oplus r'_{i_2} \oplus \ldots r'_{i_T}$ and T codes via the use of pseudorandom number generators. Messages $M_j = Enc(k_j, r'_{i_j})(j = 1, \ldots, T)$ are delivered to

the nodes of grid G_w from the sink. Then, the sink sets R'_w and r'_{i_j} while removing R_w and r_{i_j} to update RT.

The old hashing vector v should be replaced with a new hashing vector v'. However, if the sink sends v' as is, even if it is encrypted, compromised nodes may decrypt the information and obtain v'. Therefore, in the proposed algorithm, only different bits of v and v' are sent from the sink. Subsequently, the sink delivers $v_i \oplus v'_i$, comprising distinct bits of v' and v to each node. For example, assume that $v = 0100100$ and $v' = 0100001$. In this case, the data {4, 6} is sent from the sink to the corresponding node. Equation (7) represents this process. The subscript "(2)" denotes the value shown in a binary number, and m signifies the bit length of a hashing vector.

$$\left\{d \in [0, m-1] | \left(\left(v_i \oplus v'_i\right) \ll d\right) 10...0_{(2)} = 10...0_{(2)}\right\}. \quad (7)$$

The symbol & represents a bitwise AND, and $\ll$ represents a left arithmetic shift.

Each node that receives the message reverses the specified bits from 1 to 0 or from 0 to 1.

V. ANALYSIS

A. NUMBER OF HOPS UNTIL THE NODES DETECT A FALSE MESSAGE

When detecting a false message, the required mean hop count was evaluated. For example, if the hop count is three, the third forwarding node has detected the false message. The probability, p_1, of identifying a false message in a node comprises the first step in the calculation. When no node is compromised, $(1 - FPR_t)$ represents the target probability of detecting a false message in a node. By suitably altering a hashing vector's bit length, an arbitrary value of FPR_t can be attained.

An attacker can acquire the N_c hashing vectors when the N_c nodes are compromised. Resultantly, elements of RS can be searched. The variable m_l represents the number of elements in RS. Since q codes are assigned to each grid and T codes are selected from the q codes for generating an event message, the number of possible combinations of T codes per grid is ${}_qC_T$. Moreover, because there are g grids, the number of possible combinations of T codes of all grids is represented by

$$m_l = g \cdot {}_qC_T. \quad (8)$$

The bit length of a code or an EOC is b. The number of possible representations of b bits is 2^b. Since the number of legitimate EOCs is m_l (Equation (8)), the number of false EOCs is $2^b - m_l$. When an attacker compromises N_c nodes, the attacker can have N_c hashing vectors. The probability that N_c hashing vectors cannot detect each false EOC as false is $(FPR_t)^{N_c}$ because the false positive rate of each hashing vector is set to (FPR_t). Let m_f characterize the number of false EOCs that the attacker thinks may be legitimate. The value is represented by

$$m_f = FPR_t^{N_c}\left(2^b - m_l\right). \quad (9)$$

Since there are m_l legitimate EOCs and m_f false EOCs that the attacker considers possibly legitimate, the probability that the attacker generates a false EOC is represented by $m_f/(m_l + m_f)$. Each forwarding node can detect a false EOC with probability $(1 - FPR_t)$ if the node receives a false EOC. Therefore, the expected probability that each forwarding node can judge the received EOC is generated by the attacker is represented by

$$p_1 = \frac{m_f}{m_l + m_f} \times (1 - FPR_t). \quad (10)$$

Let $p_h{}'$ signify the expected number of hops until the forwarding nodes identify a false event message and let H represent the maximum number of hops to the sink. The probability that the ith node can identify a false event message is denoted by $(1 - p_1)^{i-1} \cdot p_1$ because $(i - 1)$ nodes do not identify it with probability $(1 - p_1)^{i-1}$ and the ith node detects it with probability p_1. Therefore, the expected number of hops is given by

$$p_h{}'(H) = \sum_{i=1}^{H} i \cdot (1 - p_1)^{i-1} \cdot p_1 = \frac{1 - (1 - p_1)^H}{p_1}. \quad (11)$$

The variable H represents the maximum number of hops to the sink.

B. THE MAXIMUM QUANTITY OF TRAFFIC AN ATTACKER CAN CREATE PER GRID

The variable D represents an upper limit of the occurrence frequency of events within the same grid. The bit length of the event information is $|E|$ and the bit length of EOC is b. When there are D event messages, the amount of the messages is represented by

$$Q = (|E| + b) \cdot D. \quad (12)$$

C. THE QUANTITY OF COMMUNICATION TRAFFIC RESULTING FROM LEGITIMATE EVENT MESSAGES

A message related to updating codes from the sink must also be delivered by the nodes to the sink alongside an event message based on the proposed method. When nodes in a grid detect an event, up to q nodes send the event information E and its code to the leader node. Then the leader node generates an event message, which is forwarded to the sink. When the number of hops is h, the total amount for generating and forwarding the event message to the sink is given by

$$LT = (|E| + b)(q + h). \quad (13)$$

The variable Λ represents the expected number of bits in a hashing vector that must be altered upon the sink altering one element of RS. The variables m and n denote the hashing vectors' bit length and the number of elements, respectively.

Let S signify a set with $(n-1)$ elements, let S_α characterize $S \cup \{\alpha\}$, let S_β represent $S \cup \{\beta\}$, and let $S_{\alpha\beta}$ denote $S \cup \{\alpha, \beta\}$. Here, $\alpha \notin S$ and $\beta \notin S$. Let $H[S]$ symbolize a hashing vector generated from set S.

Assume that α represents the deleting element, and β embodies the new element. We should consider the following five cases.

1. $H[S] = H[S_\alpha] = H[S_\beta]$
2. $H[S] = H[S_\alpha]$ and $H[S] \neq H[S_\beta]$
3. H[S] $\neq H[S_\alpha]$ and $H[S] = H[S_\beta]$
4. $H[S] \neq H[S_\alpha]$, $H[S] \neq H[S_\beta]$, and $H[S_\alpha] = H[S_\beta]$
5. $H[S] \neq H[S_\alpha]$, $H[S] \neq H[S_\beta]$, and $H[S_\alpha] \neq H[S_\beta]$

The number of bits in the hashing vector that must be changed when the sink changes one element of the RS is different in each case. The number of bits that must be changed is 0 in case 1 because deleting α and adding β do not affect the hashing vector. Similarly, the number of bits that must be varied is 1, 1, 0, and 2, in the cases 2, 3, 4, and 5, respectively.

Please recall that $\Re(n, m)$ signifies the false positive rate of a hashing vector with a bit length m and number of elements n (Equation (3)). Hence, the probability that $H[S] = H[S_\alpha]$ is represented by $\Re(n-1, m)$ because this situation characterizes the false positive of $H[S]$ against $\alpha \notin S$. Consequently, the probability of each case occurring is expressed as follows.

1. $\Re(n-1, m)\ \Re(n-1, m)$
2. $\Re(n-1, m)\ (1 - \Re(n-1, m))$
3. $(1 - \Re(n-1, m))\ \Re(n-1, m)$
4. $(1 - \Re(n-1, m))\ (1/m)$
5. $(1 - \Re(n-1, m))\ (1 - \Re(n-1, m) - \frac{1}{m})$

The item (1/m) in case 4 represents that the probability of $h(\alpha) = h(\beta)$. Recall that the bit length of the output of the hash function h is m.

Consequently, the following expression arises:

$$\begin{aligned}\Lambda(n, m) &= \Re(n-1, m)\ (1 - \Re(n-1, m)) \\ &\quad + (1 - \Re(n-1, m))\Big(\Re(n-1, m) \\ &\quad + 2 * \Big(1 - \Re(n-1, m) - \frac{1}{m}\Big)\Big) \\ &= e^{\frac{1-n}{m}}\Big(1 - e^{\frac{1-n}{m}}\Big) + \frac{e^{\frac{1-2n}{m}}\Big(e^{\frac{n}{m}}(-2+m) + e^{\frac{1}{m}}m\Big)}{m} \\ &= \frac{2\ e^{\frac{1-n}{m}}(m-1)}{m}. \end{aligned} \tag{14}$$

The variable H represents the mean number of hops a node makes to the sink. Equation (4) is used to calculate a hashing vector's bit length, m. The term $\log_2 m$ signifies the number of bits required to embody an arbitrary bit position of the hashing vector. The terms N and $\log_2 N$ represent the numbers of sensor nodes and bits required to embody a node, respectively. Consequently, the following expression demonstrates communication traffic quantity for updating hashing vectors and codes:

$$ET = \big(N \cdot \Lambda(m_l, m)\log_2 m\ + bT\big) \cdot (log_2\ N) \cdot H. \tag{15}$$

D. AMOUNT OF CONSUMED ENERGY

During the reception, transmission, and power down, the leaked current was 16, 18, and 0.01 mA, respectively, when we used mica2 Berkeley motes as sensor nodes [29]. Although these sensor nodes are not the newest ones, they are still the subject of research [30], [31], [32] because of their superior performance. We presume a voltage of 3 V and a bit rate of 19.2 Kbps are standard figures for the two units of measurement.

$LT + ET$ represents the number of bits for sending an event message and updating EOCs. Since the bit rate is 19.2 kbps, the required time of treating (LT + ET) bits is (LT + ET)/(19.2 · 1000).

Additionally, because sending and receiving a message requires (16+18) mA and the voltage is 3V, the required energy for sending an event message and updating EOCs is given by $3(LT + ET)/(19.2 \cdot 1000)$. When there are v event messages, the consumed energy is denoted by $3(LT + ET)/(19.2 \cdot 1000)v$.

Each sensor node consumes energy even if there are no events. There are N sensor nodes and each sensor node requires 0.01 mA. Therefore, the consumed energy in the WSN per month is represented by $N \cdot 3 \cdot 0.01 \cdot 3600 \cdot 24 \cdot 30$, where 3600·24·30 signifies the number of seconds per month.

In total, the following expression describes the monthly energy consumption of all sensor nodes when there are N sensor nodes and events occur v times per month:

$$\begin{aligned}E_I(v) = &\frac{3(LT + ET)(18 + 16)}{19.2 \cdot 1000}v \\ &+\ N \cdot 3 \cdot 0.01 \cdot 3600 \cdot 24 \cdot 30\ [mJ].\end{aligned}$$

VI. EVALUATION

A. PARAMETER SELECTION

In our experiments, the default parameter values were set as follows: $D = 1$, $H = 50$, $T = 5$, $|E| = 64$, $g = 1,000$, $q = 10$, and $N = 10,000$. These parameter settings were determined as follows.

We assume that the target events do not occur frequently; thus, the default value for the number of events per month was $D = 1$, and the value of D varied from 1 to 10 in our experiments.

Here, H is the average number of hops from a node to the sink. This value varies depending on the configuration of the WSN. In the literature [5], the maximum and minimum numbers of hops were set to 100 and 10, respectively, i.e., the average number of hops was set to approximately 50. In addition, the number of hops was set to 50 in a previous study [7]. Thus, in reference to these studies, H was set to 50 in our experiments. The number of hops to the sink does not have strong effect on the evaluation because all methods (other than SEF), including the proposed method, detect the false message within one or two hops.

In this context, T or more nodes must detect the same phenomenon. Thus, if the value of T is too large, the ability to detect legitimate events is reduced. In contrast, if the value of T is too small, the success rate of attacks will increase. This tradeoff must be considered. Many previous papers [5], [7], [12] assumed a default value of $T = 5$, and we followed this parameter setting in reference to these

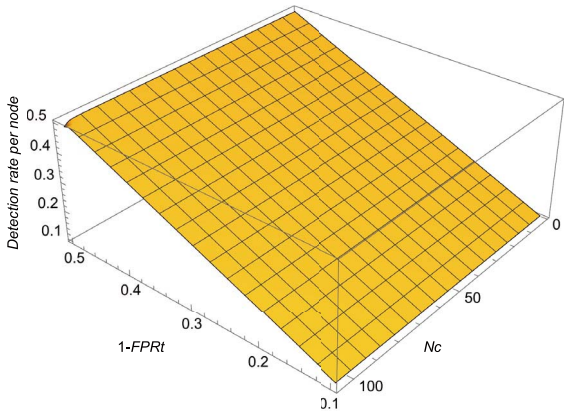

FIGURE 6. Detection rate per node of the proposed method with varying *Nc* and FPR_t.

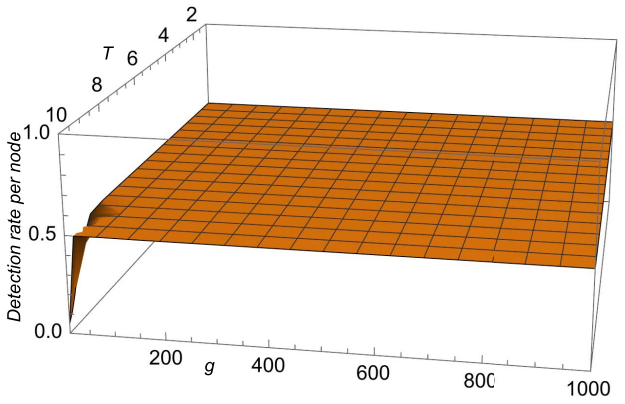

FIGURE 7. Detection rate per node of the proposed method with varying *T* and *g*.

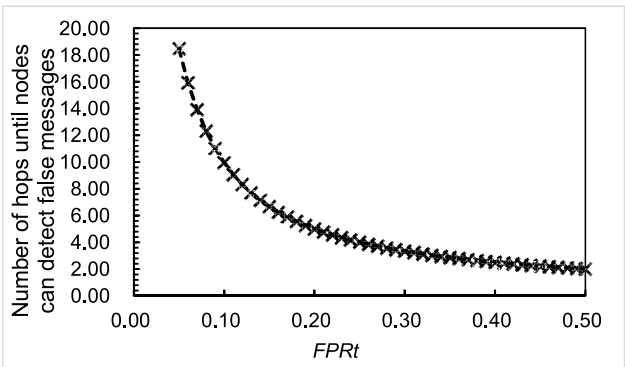

FIGURE 8. Number of hops until false messages are detected by the nodes with varying *FPRt*.

previous studies. Note that the value of T varied from 1–10 in our experiments.

The number of bits required to represent the contents of an event is highly dependent on the application. In a WSN without security mechanisms, approximately 40 bytes are required to represent a detected event [14], [15]. Based on this value and assuming that additional information, e.g., detailed sensing data, may be added, the value of $|E|$ was set to 64 bytes in our experiments.

The number of grids g can be determined by the system administrator to control the detection rate of false messages and the amount of traffic. We performed preliminary experiments and set g to values that indicate good performance (e.g., $g = 1{,}000$). In our experiments, the value of g was varied from 10–1,000. Note that the value of q varies automatically with the value of g ($q = N/g$).

Here, N is the number of nodes in a WSN. In the literature [7], the default N value was set to 10,000. We followed this setting and set N=10,000 as the default value, and the value of N varied from 1,000–10,000 in our experiments.

B. EVALUATION RESULTS

Initially, the FPR_t and the number of compromised nodes, N_c, determine how we perform the experiments on the detection rate per node. Fig. 6 illustrates the findings. We chose compromised nodes randomly from the entire network. This evaluation setting is common for evaluating detection methods' performance [5], [7], [8], [12].

The detection rate is almost the same as the *FPRt* value. However, when N_c is around 100 and *FPRt* is set to 0.5, the detection rate is slightly lower than the *FPRt* value. However, because it is difficult to imagine a situation in which more than 100 nodes are compromised without the WSN administrator being aware of it, *FPRt* can be set to 0.5.

The detection rate is lower around $Nc = 100$ and $FRP_t = 0.5$ because the attacker can estimate legitimate EOCs by compromising many sensor nodes. Consequently, the attacker can check the obtained hashing vectors whether or not randomly generated codes are legitimate. When *Nc* is large, the attacker can check a greater number of hashing vectors. When FRP_t is small, the ability to judge each hashing vector is large. Therefore, when *Nc* is large and FRP_t is small, the probability that an attacker can generate legitimate EOCs becomes large.

Fig. 7 shows the detection rate per node when changing the values of T and g. We set $N = 10,000$. The detection rate is very low when g and T are around 10. The symbol g represents the number of grids, so each grid has 1,000 nodes in this case. If each grid has many sensor nodes and T is large, the number of elements in RS is huge according to Equation (8). As a result, the detection rate becomes low, according to Equation (10). Therefore, this parameter setting should not be recommended. We recommend that the number of sensor nodes of each grid is from 10 to 20. That is, if N is 10,000, g is recommended to be from 500 to 1,000.

Fig. 8 demonstrates the projected number of hops until the detection of a false message. The higher the value of *FPRt*, the higher the detection rate and smaller the number of hops required before detection; when *FPRt* is 0.5, the average number of hops is 2, indicating that a sufficiently high detection rate is achieved.

We compared SEF [5], MobiSink [7], CD-PEFS [8], and EMAS [12] to the proposed method. Fig. 9 shows the results. When $Nc > T$, we found the security mechanism of SEF was inoperative, as the number of hops was 50. Conversely, owing to the considerable authentication

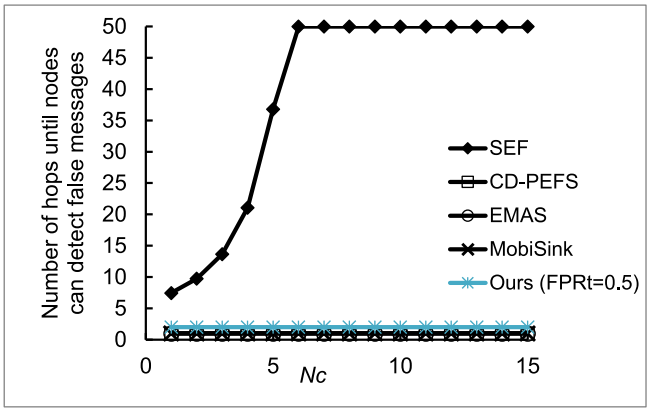

FIGURE 9. Number of hops until false messages are detected by the nodes with varying *Nc*.

information provided to each message, CD-PEFS and EMAS needed a maximum of one hop to detect false messages. The number of hops until nodes can detect false messages of SEF is high because the security mechanism of SEF breaks down when more than T nodes are compromised. In contrast, other methods, including the proposed method, can detect false messages even when more than T nodes are compromised.

Since the security mechanism of CD-PEFS, EMAS, and MobiSink are robust, the number of hops until nodes can detect false messages is only one hop. Furthermore, because the detection rate of the proposed rate is 0.5 (this value is equal to 1-FPR_t), the number of hops of the proposed method is two on average. Although the proposed method increases the number of hops required to detect a false message by one hop, the experiments described later show that it reduces the amount of traffic an attacker can generate.

Then, when no node is compromised, we calculated the mean energy consumption per node. Fig. 10 illustrates the findings. The methods used in related studies demonstrate that the proposed method requires more energy. Nonetheless, even when 10 events occurred monthly, we only found a marginal difference in the increase rate, from 77.8 to 78.3. In this experiment, we assume there are no compromises. The result represents the amount of traffic required to run the WSN during normal operation. Although the amount of traffic per event message is the smallest in the proposed method, the proposed method requires additional traffic for updating EOCs and hashing vectors. Despite these differences in the proposed methods, there is little difference in the amount of energy consumed by the WSN as a whole for all methods due to the relatively large amount of energy consumed by the nodes during normal operation.

Finally, we analyzed the proposed method, CD-PEFS, and SEF to ascertain the maximum traffic quantity a hacker could create per grid. Fig. 11 presents the findings. Given that T MACS can be appended to the message in CD-PEFS and SEF, an attacker could generate long messages, producing substantially more traffic than with the proposed method. By contrast, a lightweight authentication mechanism was performed via the proposed method, yielding minimal traffic

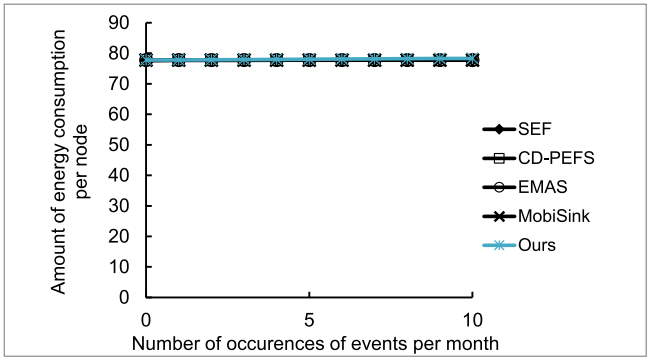

FIGURE 10. Energy consumption when no node is compromised.

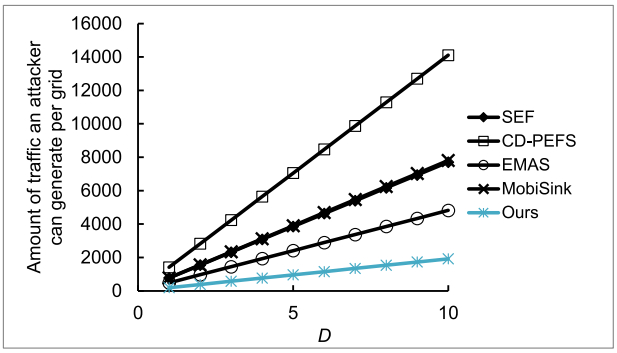

FIGURE 11. Maximum quantity of traffic an attacker can produce per second within a grid (varying *D*).

production by compromised nodes. In existing studies, each event message requires T MACs or T key IDs for the message authentication; hence, the amount of traffic becomes very large. On the other hand, our method requires only one EOC for authentication. This mechanism can greatly reduce false message traffic. As a result, the maximum amount of traffic an attacker receives for the proposed method is the lowest compared to the existing methods.

Fig. 9 reflects the detection rate, which suggests the performance of CD-PEFS is superior. Conversely, Fig. 11 reflects the amount of traffic; thus, CD-PEFS performs poorly.

We have additional experiments with varying T and N. The results are shown in Figs. 12 and 13, respectively. In all existing methods, the traffic increases as T increases. Conversely, the proposed method's traffic is not affected by its value because T codes are mapped to a single EOC. The bit length of this EOC is not affected by the value of T. On the other hand, the value of N has no significant effect on any of the methods. Some methods require node ID information, but the bit length of the node ID only increases on a log scale compared to an increase in the number of nodes.

From these results, in all parameter settings, the amount of traffic an attacker can generate of the proposed method is the lowest compared to other methods.

VII. DISCUSSION

A symmetric key encryption scheme encrypts all messages. This encryption process is common in all existing studies.

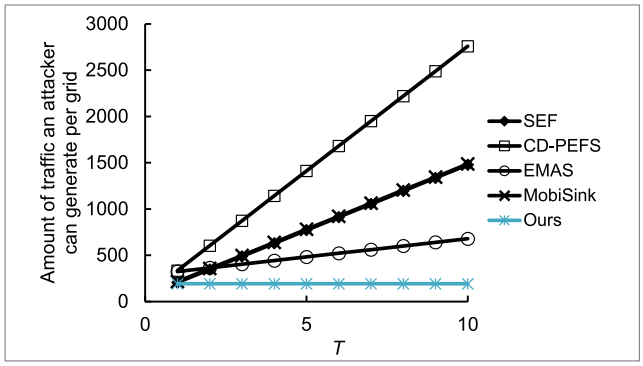

FIGURE 12. Maximum quantity of traffic an attacker can produce per second within a grid (varying T).

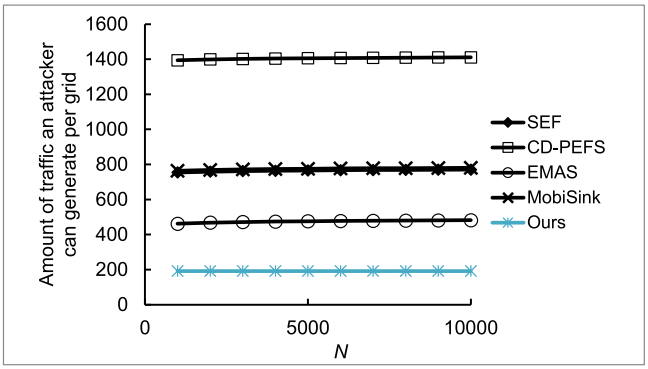

FIGURE 13. Maximum quantity of traffic an attacker can produce per second within a grid (varying N).

The computation cost of a symmetric key encryption scheme is much lower than a public key cryptography scheme [33]. Further, the calculation of a hash value can be performed with much less computation than that required for encryption using a symmetric key encryption scheme [34]. Regarding the operation of XOR and checking a specified bit of a hashing vector, the computation cost can be negligible because these operations are simple bit operations. Therefore, the proposed method's computation cost is minimal for each sensor node.

The sink needs to generate a seed and its code for each sensor node. Then, it generates EOCs by an XOR operation. Moreover, it needs to generate a hashing vector for each sensor node. The number of EOCs is $g\ {}_qC_T$; therefore, if the values of g or q are large, the sink needs to generate a lot of EOCs. However, because the generation of EOC requires only a simple XOR operation, this process does not take a long time.

The amount of traffic is evaluated in Figs. 10 and 11, where there are no malicious nodes and a malicious node generates a lot of false messages, respectively. From these figures, the proposed method is robust against false message attacks by malicious nodes, although the amount of traffic during no malicious nodes is almost identical to that of the existing method.

The proposed method assumes that the occurrence frequency of legitimate events is not high. The communication overhead becomes non-negligible if many legitimate events occur because the codes of several sensor nodes should be updated each time a legitimate message is forwarded to the sink. Notably, an attacker wants to generate network congestion to prevent a legitimate event message from reaching the sink. If there are many legitimate events, it is challenging for an attacker to attack all legitimate events in the first place. The assumption that the situation in which such an attack might occur is one in which the number of correct event messages is small is realistic.

The maximum number of messages an attacker can produce equals the maximum number of messages a legitimate node receives. Additionally, the maximum number of messages a node can produce can be determined by the WSN manager according to the application of the WSN. For example, if the application is crime detection, an event does not occur so frequently. Therefore, even if there is an upper limit to the number of messages that can be generated, normal node operation is not restricted.

VIII. CONCLUSION

In this paper, we proposed an algorithm that can detect false messages in a large WSN with a small number of hops. Unlike existing studies, only a single, one-time authentication code is assigned to each event message for authentication in the proposed method, significantly reducing the amount of message traffic compared to existing methods. Since existing research requires T MACs, T key IDs, or other information for authentication, an attacker can generate much traffic just by generating one false message. On the other hand, the existing methods can detect a false message in one hop, but it takes two hops to detect a false message with the default setting of the proposed method. However, the proposed method can prevent network congestion attacks by significantly reducing the amount of traffic. Experiments have shown that the proposed method can reduce traffic volume by more than 60%. In a future study, we will deploy several hundred sensor nodes and perform demonstration tests.

REFERENCES

[1] T. Vaiyapuri, V. S. Parvathy, V. Manikandan, N. Krishnaraj, D. Gupta, and K. Shankar, "A novel hybrid optimization for cluster-based routing protocol in information-centric wireless sensor networks for IoT based mobile edge computing," *Wireless Pers. Commun.*, pp. 1–24, Jan. 2021, doi: 10.1007/S11277-021-08088-W.

[2] K. Haseeb, Z. Jan, F. A. Alzahrani, and G. Jeon, "A secure mobile wireless sensor networks based protocol for smart data gathering with cloud," *Comput. Electr. Eng.*, vol. 97, Jan. 2022, Art. no. 107584, doi: 10.1016/J.COMPELECENG.2021.107584.

[3] M. S. Rajan *et al.*, "Diagnosis of fault node in wireless sensor networks using adaptive neuro-fuzzy inference system," *Appl. Nanosci.*, pp. 1–9, Jun. 2021, doi: 10.1007/S13204-021-01934-0.

[4] R. Singh and S. Singh, "Smart border surveillance system using wireless sensor networks," *Int. J. Syst. Assur. Eng. Manag.*, vol. 13, pp. 1–15, Aug. 2021, doi: 10.1007/S13198-021-01208-6.

[5] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proc. IEEE INFOCOM* vol. 4, 2004, pp. 2446–2457, doi: 10.1109/INFCOM.2004.1354666.

[6] J. Wang, Z. Liu, S. Zhang, and X. Zhang, "Defending collaborative false data injection attacks in wireless sensor networks," *Inf. Sci.*, vol. 254, pp. 39–53, Jan. 2014.
[7] Y. Sei and A. Ohsuga, "False event detection for mobile sinks in wireless sensor networks," in *Proc. Eur. Intell. Security Inform. Conf. (EISIC)*, 2013, pp. 52–59, doi: 10.1109/EISIC.2013.15.
[8] A. Kumar, N. Bansal, and A. R. Pais, "A partial key pre-distribution based en-route filtering scheme for wireless sensor networks," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 1, pp. 1471–1486, Jan. 2021, doi: 10.1007/S12652-020-02216-3.
[9] M. V. Babu, J. A. Alzubi, R. Sekaran, R. Patan, M. Ramachandran, and D. Gupta, "An improved IDAF-FIT clustering based ASLPP-RR routing with secure data aggregation in wireless sensor network," *Mobile Netw. Appl.*, vol. 26, no. 3, pp. 1059–1067, Jun. 2021, doi: 10.1007/S11036-020-01664-7.
[10] Y. Wang, F. Li, P. Ren, S. Yu, and Y. Sun, "A secure aggregation routing protocol with authentication and energy conservation," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 1, Jan. 2022, Art. no. e4387, doi: 10.1002/ETT.4387.
[11] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12, doi: 10.1109/INFOCOM.2006.304.
[12] C. Yi, "En-route message authentication scheme for filtering false data in WSNs," *Security Commun. Netw.*, vol. 2021, Sep. 2021, Art. no. 4068507, doi: 10.1155/2021/4068507.
[13] S. Ruj, A. Nayak, and I. Stojmenovic, "Pairwise and triple key distribution in wireless sensor networks with applications," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2224–2237, Nov. 2013, doi: 10.1109/TC.2012.138.
[14] M. T. Hansen, "Asynchronous group key distribution on top of the cc2420 security mechanisms for sensor networks," in *Proc. ACM WiSec*, 2009, pp. 13–20.
[15] C. Gezer, M. Niccolini, and C. Buratti, "An IEEE 802.15.4/ZigBee based wireless sensor network for energy efficient buildings," in *Proc. IEEE WiMob*, 2010, pp. 486–491.
[16] C. Pedroso and A. Santos, "Dissemination control in dynamic data clustering for dense IIoT against false data injection attack," *Int. J. Netw. Manag.*, vol. 32, no. 5, May 2022, Art. no. e2201, doi: 10.1002/NEM.2201.
[17] M. Saska, T. Krajnik, and L. Pfeucil, "Cooperative μUAV-UGV autonomous indoor surveillance," in *Proc. Int. Multi-Conf. Syst. Sygnals Devices*, Mar. 2012, pp. 1–6, doi: 10.1109/SSD.2012.6198051.
[18] M. Tanveer, G. Abbas, Z. H. Abbas, M. Bilal, A. Mukherjee, and K. S. Kwak, "LAKE-6SH: Lightweight user authenticated key exchange for 6LoWPAN-based smart homes," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2578–2591, Feb. 2022, doi: 10.1109/JIOT.2021.3085595.
[19] M. Alizadeh, M. H. Tadayon, and A. Jolfaei, "Secure ticket-based authentication method for IoT applications," *Digit. Commun. Netw.*, to be published, doi: 10.1016/J.DCAN.2021.11.003.
[20] P. Wang, F. Xue, H. Li, Z. Cui, L. Xie, and J. Chen, "A multi-objective DV-Hop localization algorithm based on NSGA-II in Internet of Things," *Mathematics*, vol. 7, no. 2, p. 184, Feb. 2019, doi: 10.3390/MATH7020184.
[21] Y. Jin, L. Zhou, L. Zhang, Z. Hu, and J. Han, "A novel range-free node localization method for wireless sensor networks," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 688–692, Apr. 2022, doi: 10.1109/LWC.2021.3140063.
[22] N. Verma, A. Kaushik, and P. Nayak, "A lightweight secure authentication protocol for wireless sensor networks," in *Proc. Int. Conf. Innovat. Comput. Commun.*, 2021, pp. 291–299, doi: 10.1007/978-981-15-5113-0_21.
[23] D. Liu and P. Ning, "Multilevel μTESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 4, pp. 800–836, Nov. 2004, doi: 10.1145/1027794.1027800.
[24] Y. Sei and S. Honiden, "Distributed detection of node replication attacks resilient to many compromised nodes in wireless sensor networks," in *Proc. 4th Annu. Int. Conf. Wireless Internet (WICON)*, Nov. 2008, pp. 28:1-28:8.
[25] N. Elsakaan and K. Amroun, *Distributed and Reliable Leader Election Framework for Wireless Sensor Network (DRLEF)* (Lecture Notes in Networks and Systems), vol. 378. Cham, Switzerland: Springer, 2022, pp. 123–141, doi: 10.1007/978-3-030-95918-0_13.
[26] S. Wang, X. Jiang, and H. Wymeersch, "Cooperative localization in wireless sensor networks with AOA measurements," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 6760–6773, Aug. 2022, doi: 10.1109/TWC.2022.3152426.
[27] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[28] R. E. N. Editor and P. L'ecuyer, "Efficient and portable combined random number generators," *Commun. ACM*, vol. 31, no. 6, pp. 742–751, Jun. 1988, doi: 10.1145/62959.62969.
[29] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance measurements of motes sensor networks," in *Proc. 7th ACM Int. Symp. Model. Anal. Simulat. Wireless Mobile Syst. (MSWiM)*, Oct. 2004, p. 174, doi: 10.1145/1023663.1023695.
[30] S. R. Jondhale, R. Maheswar, and J. Lloret, "Fundamentals of wireless sensor networks," in *Received Signal Strength Based Target Localization and Tracking Using Wireless Sensor Networks*. Cham, Switzerland: Springer, 2022, pp. 1–19.
[31] J. Singh, R. Kaur, and D. Singh, "Energy harvesting in wireless sensor networks: A taxonomic survey," *Int. J. Energy Res.*, vol. 45, no. 1, pp. 118–140, Jan. 2021, doi: 10.1002/ER.5816.
[32] A. W. Bhat and A. Passi, "Wireless sensor network motes: A comparative study," in *Proc. 9th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2022, pp. 141–144, doi: 10.23919/INDIACOM54597.2022.9763269.
[33] G. De Meulenaer, F. Gosset, F. X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proc. IEEE WiMob*, 2008, pp. 580–585, doi: 10.1109/WIMOB.2008.16.
[34] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, pp. 1384–1393, 2015, doi: 10.1109/TIFS.2015.2410137.

YUICHI SEI (Member, IEEE) received the Ph.D. degree in information science and technology from the University of Tokyo in 2009. From 2009 to 2012, he was with Mitsubishi Research Institute. He joined the University of Electro-Communications in 2013, where he is currently an Associate Professor with the Graduate School of Informatics and Engineering. He is also a Researcher with Japan Science and Technology Agency (JST), PRESTO. His current research interests include pervasive computing, privacy-preserving data mining, and software engineering. He was a recipient of the IPSJ Best Paper Award and the JSCE Hydraulic Engineering Best Paper Award in 2017. He is a member of the IEEE Computer Society, IEEE Signal Processing Society, Information Processing Society of Japan, Institute of Electronics, Information and Communication Engineers, and Japan Society for Software Science and Technology.

AKIHIKO OHSUGA (Member, IEEE) received the Ph.D. degree in computer science from Waseda University in 1995. From 1981 to 2007, he was with Toshiba Corporation. He joined the University of Electro-Communications in 2007, where he is currently a Professor with the Graduate School of Informatics and Engineering. He is also a Visiting Professor with the National Institute of Informatics. His research interests include agent technologies, Web intelligence, and software engineering. He received the IPSJ Best Paper Awards in 1987 and 2017. He is a member of the IEEE Computer Society (IEEE CS), Information Processing Society of Japan (IPSJ), Institute of Electronics, Information and Communication Engineers, Japanese Society for Artificial Intelligence (JSAI), Japan Society for Software Science and Technology (JSSST), and Institute of Electrical Engineers of Japan. He has been a Fellow of IPSJ since 2017. He served as a Chair of IEEE CS Japan Chapter, and a member of JSAI Board of Directors, JSSST Board of Directors, and JSSST Councilor.