Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 697098, 15 pages
http://dx.doi.org/10.1155/2015/697098

Research Article

An Enhanced Communication Protocol for Location Privacy in WSN

Abdel-Shakour Abuzneid, Tarek Sobh, and Miad Faezipour

University of Bridgeport, Bridgeport, CT, USA

Correspondence should be addressed to Abdel-Shakour Abuzneid; abuzneid@bridgeport.edu

Received 28 November 2014; Revised 27 February 2015; Accepted 8 March 2015

Academic Editor: Xinyi Huang

Wireless sensor network (WSN) is built of many sensor nodes. The sensors can sense a phenomenon, which will be represented in a form of data and sent to an aggregator for further processing. WSN is used in many applications, such as object tracking and security monitoring. The objects in many situations need physical and location protection. In addition to the source location privacy, sink location privacy should be provided. Providing an efficient location privacy solution would be challenging due to the open nature of the WSN. Anonymity is a key solution for location privacy. We present a network model that is protected against local, multilocal, and global adversaries that can launch sophisticated passive and active attacks against the WSN.

1. Introduction

A wireless sensor node (SN) is a simple autonomous host device. It can sense a phenomenon, convert the sensed information into data, process the data, and then transmit the data to a base station (BS) for further analysis. The SN host is very limited in terms of storage space, memory, processing power, communication bandwidth, and battery energy [1]. This work focuses on monitoring and tracking applications, such as tracking animal in the wildlife or a fellow soldier in the battlefield. When the SN senses an object, it reports data by sending it hop-by-hop to the BS. One of the most common applications discussed in the literature is the *panda monitoring game* [1]. When a sensor node detects a Panda in a certain area, it should report the data via a message to the BS. The fact that the data is sent out of the SN in open environment will be a reason for the adversary (ADV) to know the location of the SN and consequently the location of the object. In order to protect the Panda from the ADV, we need to implement in place an efficient source location privacy scheme (SLP). SLP is even more important in military, homeland security, and law enforcement, in addition to many civilian applications [2]. We need, as well, to provide location privacy for the BS (BSLP), which is the data aggregator and the controller of the WSN. The solution needs to provide anonymity where the ADV cannot know the identity of the SNs. We have to reduce the capture likelihood and increase safety period. Safety period is the time before the first SN is captured. All transmissions in the WSN happen in the open air where the SNs are unattended. One of the most important issues that any solution needs to account for is the energy conservation to maintain a longer life for the WSN. Thus, applying regular privacy and security algorithms might not be suitable for such networks.

2. Problem Statement

Privacy in WSN is typically categorized into two categories: data privacy and context privacy [3]. Data privacy focuses on data aggregation and data query. Contextual privacy involves location privacy, identity privacy, routing privacy, and temporal privacy. Providing SLP and BSLP could be achieved using many schemes such as *random walk, geographic routing, delay, dummy data sources, cyclic entrapment, anonymization, cross-layer routing, separate path routing, network coding, and other schemes* [1, 4].

In this work, we will use *anonymization* to provide SLP and BSLP. One of the first works to classify context privacy was done by Kamat et al. [5, 6], where they addressed the Panda hunter game. They claim that the routing scheme is responsible for hiding source location of an object. They have used two metrics to measure SLP: first, the *safety period*, which is the number of messages a source sends before it is captured, and second, the *capture likelihood*, which is the probability that an adversary can capture the source within a certain period. There are generally two ways to locate a source using passive attacks: *traffic analysis* [3, 7] and *packet tracing* [3, 8, 9]. We provide a framework that can be tested against other solutions using five metrics: *security*: the probability that the adversary successfully identifies the source, the intermediary SNs, or the BS; *energy cost*, where the solution will provide privacy with reasonable energy conservation; *storage memory cost* and *delivery time*, where the solution should consume a reasonable storage space and provide reasonable delays; *safety period*, where the solution will guarantee transmission of reasonable amount of messages before the first SN is captured. The rest of this paper is organized as follows. In Section 3, we give some background and literature survey. In Section 4, we will explain the suggested system model, network model, threat model, and the traffic model. In Section 5, we will introduce our anonymity protocol. In Section 6, we will have a thorough security analysis. In Section 7, we will have performance analysis and evaluation. In Section 8, we will summarize our work and suggest some additional models to the framework in the future work.

3. Background and Literature Survey

There are many solutions, which have been presented to solve the problems of SLP and BSLP. Conti et al. [1] categorized the solutions into groups. They have discussed many solutions and compared them in terms of the threat model, view of the network, power consumption, exposed information, and efficiency in providing "location privacy." Recently, anonymity has become a concern for WSNs. We have identified important literature discussing solutions for anonymity in WSN, such as HIR: Hashing-Based ID Randomization [10]; RHIR: Reverse HIR [10]; SAS: Simple Anonymity Scheme [11]; CAS: Cryptographic Anonymity Scheme [11]; MAQ: Max Query Aggregation [12]; APR: Anonymous Path Routing [13]; ACS: Anonymous Communications Scheme [14]; DCARPS: Destination Controlled Anonymous Routing Protocol for Sensor Nets [2]; PhID: Phantom ID [15]. None of these solutions provides location privacy against global ADVs and active attacks. A reasonable solution against global adversary introduced by Chen et al. [16] called efficient anonymous communication (EAC), which provides sender, link, and sink anonymity. The solution consists of three phases. Every SN creates multiple pseudonym IDs, which are global anonymous identity, anonymous broadcast identity, and anonymous one-hop identity, and anonymous acknowledgement identity. Each transmission should have a new anonymous pseudonym. The solution is relatively light; however, it easily breaks pseudonym synchronization. In addition, it fails to provide BSLP and it is not secure against traffic rate analysis attacks. We know that *anonymization* is not enough to achieve end-to-end privacy. There are some solutions based on dummy data sources where SNs send out fake packets to other nodes within the network. A fake packet does not contain any information about any real event but it helps to obfuscate the real traffic and to divert the adversary. The literature shows reasonable solutions using dummy sources. Some of them are designed to handle local ADV. However, few are suitable for global ADV. Some of the literatures presume a certain routing scheme, topology, network, and threat models. Ouyang et al. [17] introduced three different solutions to handle the global ADVs. In this work, we will enhance EAC, the efficient anonymous communicating protocol [16, 18]. EAC does not handle the pseudonyms synchronization very well. There are many situations where the system will get unsynchronized. It also could not handle multi-colluding adversaries and lacks a mechanism for time correlation attack. There is an extension for EAC presented in [4] which has reasonably addressed some of the issues of EAC. Most of the other solutions do not handle global or multi-colluding adversaries. Each solution focuses on certain scenarios of attacks. Our work is aimed to be comprehensive.

4. Preliminaries

Our contribution in this work provides sender anonymity, receiver anonymity, link anonymity, SLP, BSLP, and data privacy. The system is fed with inputs, such as the nature of the ADVs in the network and the residual energy in the SNs. We assume bidirectional links where two nodes are considered neighbors if they can hear each other [2]. The network considers one BS which aggregates sensed data from all the SNs. The BS works as an interface to the wired network [11]. Data packets generated by SNs are addressed to the BS; however, they go through multihop paths. Control packets are sent from the BS to the SNs by unicast or broadcast messages. To enhance BSLP, the BS acts like a normal SN in the network when it communicates with other SNs, to make it indistinguishable. The WSN runs in two phases: startup phase and communication phase. We assume that the SNs have the ability to obfuscate the addresses at the MAC level header [11]. All sensors are time synchronized [11]. There are many algorithms for SNs *time synchronization*, which is out of the scope of this work. The WSN will need a protocol for *network topology discovery* that allows the BS to view the global topology of the network without revealing the BS location [2]. The adversary nodes have very strong capabilities compared to the SNs. They are resource-rich; they have sufficient energy supply, computation capabilities, and unlimited storage memory. An adversary could run both *passive* and *active* attacks. We presume that only few compromised nodes could exist at one time due to the implementation of intrusion detection system (IDS) which is able to detect compromised SNs. We assume a global adversary, which can monitor the traffic of the entire network and can determine the node responsible for the initial transmission.

Assuming a global adversary means the following: (a) the worst-case scenario for area coverage where colluding sensors can cooperate to cover the whole network area [21]; (b) the worst-case scenario for timing where the coverage area of the adversary is not known at any time [21]. We also assume that the adversary is capable of observing SNs transmissions over extended periods. It is, however, not able to break the encryption algorithms or the hash functions used for securing data during transmission. We presume *abundant* traffic where sensors detect and transmit many packets, such as in the applications of environment monitoring.

5. Proposed Anonymous Model

The communication process is divided into *two* phases, namely: startup phase and communication phase.

5.1. Startup Phase. Prior to actual distribution of the sensors in the field of application, the SNs need to be tested, fully charged, and preloaded with some parameters which are needed during the startup phase and then during the communication phase. All parameters and terms used for the proposed solution are mentioned in Notations [4, 16, 19].

It is typical to presume that the WSN is considered secure for some short period after SNs' deployment in the field and before the steady communication phase. Zhu et al. [22] presented that WSN has a lower bound on the time interval that the adversary needs to be able to compromise a SN. During this time, the SNs can communicate and exchange all preloaded and calculated parameters safely. The SNs need to know their relative locations to the BS and to the neighboring SNs. Likewise, the BS needs to know the location of all the SNs participating in the WSN. There are Nemours *localization schemes* which are proposed in the literature [11, 16, 23, 24]. After the localization process is completed, each SN will know its smallest *hop-count* to the BS ($\mathrm{hc}_{i\leftrightarrow \mathrm{bs}}$).

5.1.1. Issuing the Pseudonyms. A SN uses any issued pseudonym only once. That means we need to use a one disposable pseudonym per a transmission. There are five kinds of transmissions that could happen in the WSN [4, 16]: (i) multihop transmission between a SN and BS, (ii) transmission between two neighbor SNs, (iii) broadcast sent by a SN or the BS, (iv) acknowledgement, and (v) fake broadcast. The process starts by creating a pseudonym ID for each SN_i, and we call it for short (PID_i) which is issued using the following expression:

$$\mathrm{PID}_i = H\left(\mathrm{ID}_i \oplus a_i\right). \tag{1}$$

The SN_i can calculate the broadcast pseudonym ID (BPID_i) according to the following expression:

$$\mathrm{BPID}_i = H\left(\mathrm{ID}_i \oplus b_i\right). \tag{2}$$

The SN_i can calculate the fake broadcast pseudonym ID (FBPID_i) according to the following expression:

$$\mathrm{FBPID}_i = H\left(\mathrm{ID}_i \oplus c_i\right). \tag{3}$$

SN_i should know its entire neighbor set N_i. A neighbor SN_j is a sensor that could receive signal from the SN_i and vice versa through one-hop transmission. SN_i will send $M_{\mathrm{discovery}}$, a broadcast discovery message, to exchange parameters with all one-hop neighbors. The format of the message is as follows:

$$\begin{aligned} &M_{\mathrm{discovery}} \\ &= k_{\mathrm{dis}} \\ &\quad \cdot \left(\mathrm{TTL} \parallel \mathrm{ID}_i \parallel k_{i\leftrightarrow \mathrm{bs}} \parallel \mathrm{kb}_i \parallel \mathrm{fkb}_i \parallel a_i \parallel b_i \parallel c_i \parallel \mathrm{HC}_{i\leftrightarrow \mathrm{bs}}\right). \end{aligned} \tag{4}$$

The value of TTL should be *one* for this transmission. SN_i will receive also a similar broadcast message from SN_j and from all other neighbors. Both SN_i and SN_j will calculate a new random value ($a_{i\leftrightarrow j}$) according to the following expression:

$$a_{i\leftrightarrow j} = H\left(\mathrm{ID}_i \oplus \mathrm{ID}_j\right). \tag{5}$$

Both SN_i and SN_j will calculate also a new pairwise key $k_{i\leftrightarrow j}$ according to the following expression:

$$k_{i\leftrightarrow j} = H\left(k_{i\leftrightarrow \mathrm{bs}} \oplus k_{j\leftrightarrow \mathrm{bs}}\right). \tag{6}$$

SN_i also calculates broadcast pseudonym ID for SN_j (BPID_j) according to expression (2). SN_i has already received the values of ID_j and b_j through $M_{\mathrm{discovery}}$. It also calculates the one-hop pseudonym ID ($\mathrm{OHPID}_{i\leftrightarrow j}$) shared between SN_i and SN_j as expressed in the following expression:

$$\mathrm{OHPID}_{i\leftrightarrow j} = H\left(a_i \oplus a_j\right). \tag{7}$$

Finally, acknowledgement pseudonym ID for SN_i (APID_i) will be calculated according to the expression below:

$$\mathrm{APID}_i = H\left(\mathrm{ID}_i\right). \tag{8}$$

SN_i will have a table TBL_i which contains the shared values with the neighbors as listed in Table 1. Thus, we have replaced the ID with *quintuple pseudonyms* [4] to reference the SN during the communication. Figure 1 shows two sensors after they have exchanged the required pseudonyms.

5.1.2. Deleting Security Information. After storing all required pseudonyms, parameters, and keys in TBL_i for SN_i, it would be the time to delete all unnecessary information for the purpose of security [4, 16, 25]. In addition, it will release some memory storage space [16, 18]. Most importantly, SN_i will delete ID_i and $\mathrm{HC}_{i\leftrightarrow \mathrm{bs}}$, which could be critical information for the adversary. In addition, SN_i will delete discovery messages received earlier from the neighbors.

5.2. Communication Phase. During the steady communication phase, when sensing and sending data by nodes to the BS takes place, *seven* operations continue until the network lifetime ends. These operations are [4] (i) sensing and sending a message to a neighbor, (ii) forwarding a message to a neighbor, (iii) broadcasting a message, (iv) acknowledgement, (v) sending a fake message, (vi) SN addition, and (vii)

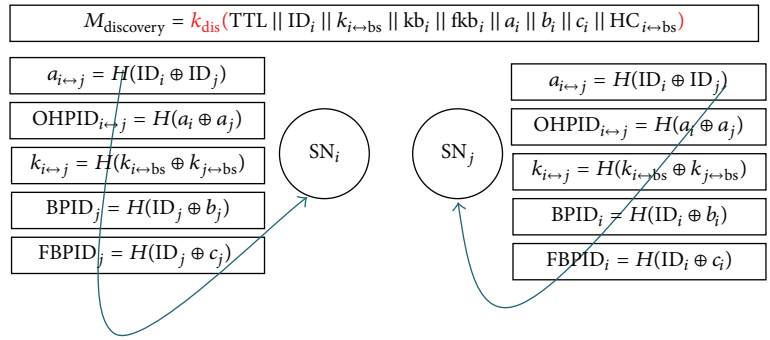

Figure 1: Two SN neighbors share the required pseudonyms.

Table 1: Shared values among sensor neighbors. If SN_i has N_i neighbors, then TBL_i will have N tuples, one tuple per a neighbor [4, 16].

Information in T_i per each neighbor	Tuple for SN_j
Shared random number	$a_{i\leftrightarrow j}$
Shared broadcast random number	b_j
Shared fake broadcast random number	c_j
Shared broadcast key	BPID_j
Shared fake broadcast key	FPID_j
Shared one-hop key	$k_{i\leftrightarrow j}$
Current one-hop pseudonym ID	$\text{OHPID}_{i\leftrightarrow j}$
Link direction	$\text{link}_{i\to j}$
Residual energy level	Δ_j

SN removal. A SN will have three roles, in terms of data transmission, during the communication phase [4, 16, 25]: (i) role as a sensor, (ii) as a forwarder or intermediary node, and (iii) as a broadcaster. In the following sections, we will use SN_i as a source node and SN_j as a neighbor to the source.

5.2.1. Transmission as a Sensor. When SN_i senses data, it needs to send a message hop-by-hop to the BS. The SN_i only recognizes itself by its newly calculated pseudonym (PID_i), and the BS will recognize source of the message through PID_i identity as well. Thus, the PID_i of the source needs to be included in the message until it reaches the BS. Consequently, the PID of a sensor will be updated after every transmission. When data is sensed and ready to be sent out, the SN_i needs to select one neighbor from N_i to forward the message to it. The selection process goes through a probabilistic protocol which guarantees that SN_i does not depend all the time on one neighbor when forwarding its data; first, for routing privacy and second for increasing the lifetime of the WSN. SN_i will form the message in the following format:

$$\begin{aligned} M_{i\to j} = {} & \text{OHPID}_{i\leftrightarrow j} \| E_{k_{i\to j}} \\ & \cdot \left(\text{PID}_i \| E_{k_{i\leftrightarrow\text{bs}}}(D_i)\right) \| \text{HMAC}_{k_{i\leftrightarrow\text{bs}}} \\ & \cdot \left(\text{PID}_i \| D_i\right). \end{aligned} \tag{9}$$

Once SN_i knows that the message ($M_{i\to j}$) is delivered to the BS, it needs to dispose of the current pseudonym PID_i and issue a new one for the next transmission as indicated in the following expression:

$$\text{PID}_i = H\left(\text{PID}_i \oplus a_i\right). \tag{10}$$

In addition, both SN_i and SN_j will dispose of the current $\text{OHPID}_{i\to j}$ and issue a new one for the next communication between the two sensors according to following expression:

$$\text{OHPID}_{i\to j} = H\left(\text{OHPID}_{i\to j} \oplus a_{i\to j}\right). \tag{11}$$

The message M will then be reformatted by the recipient SN_j. Then, it will be forwarded to the next node, in the path, SN_r, and so on, until it gets to the BS. If SN_j was the BS, then the BS uses the shared one-hop key between the sensor and the BS to decrypt the encrypted data and gets the PID_i, which the BS can use to recognize the SN_i. Thus, the BS will be able to recognize the originator of this message. Only at this point of time, BS can update the value of PID_i of SN_i. It also gets the data (D_i) which BS can read after decrypting the data by $k_{i\leftrightarrow\text{bs}}$.

5.2.2. Transmission as a Forwarder. When SN_i sends the message one-hop uplink to the neighbor SN_j, SN_j needs to forward the message to another intermediary node or in the best case to the BS if SN_j is a neighbor of the BS. Upon receiving $M_{i\to j}$, SN_j will match $\text{OHPID}_{i\to j}$ in its table, T_j, under the column related to SN_i. If there is no match, then the message definitely is not addressed for SN_j and it will be dropped immediately. If it matches, then the message is addressed to SN_j and it will be decrypt using $k_{i\to j}$. The message will be forwarded to SN_r after M is reformatted as in

$$\begin{aligned} M_{j\to r} = {} & \text{OHPID}_{j\leftrightarrow r} \| E_{k_{j\to r}} \\ & \cdot \left(\text{PID}_i \| E_{k_{i\leftrightarrow\text{bs}}}(D_i)\right) \| \text{HMAC}_{k_{i\leftrightarrow\text{bs}}}\left(\text{PID}_i \| D_i\right). \end{aligned} \tag{12}$$

Right after the data is *received* by SN_j and *forwarded* to the next one-hop SN_r, the SN_j updates both pseudonyms, $\text{OHPID}_{i\leftrightarrow j}$ and $\text{OHPID}_{j\leftrightarrow r}$. SN_j now is ready to exchange

another message with SN_i using the new pseudonym $OHPID_{i\leftrightarrow j}$. However, SN_j is not yet ready to send data to SN_r since SN_r does not update the $OHPID_{j\leftrightarrow r}$ until D_i is forwarded to the next hop, say NS_v. What would happen if SN_j senses new data while it is busy forwarding a previous message? It has to wait until $OHPID_{j\leftrightarrow r}$ is updated or it can use a different neighbor to forward the new message.

5.2.3. Acknowledgment. As expected in data networks, message could be lost or become corrupted. In either case, retransmission is required. Because SNs change PIDs after each transmission, synchronizing PIDs is crucial. Updating the pseudonyms depends on successful message transmission and reception. Technically, a sender and a receiver should alter the pseudonym value only after making sure that the data are sent correctly and received perfectly by the BS. The lack of direct connection between the sender and the receiver makes it a complicated process. The BS cannot send direct acknowledgement to the source if it is multiple hops away. We have to depend on multiple acknowledgements along the path between the source and the destination. Transmitting data through a route of y hops needs y acknowledgements. SN_i needs to calculate acknowledgement pseudonym ID ($APID_i$) according to the following expression:

$$\mathrm{APID}_i = H\left(\mathrm{APID}_i \oplus b_i\right). \tag{13}$$

The message will be sent out now with the current value for $APID_i$. Thus, we will rewrite $M_{i\to j}$ as it appears in

$$\begin{aligned} M_{i\to j} = {} & \mathrm{Padding} \parallel \mathrm{OHPID}_{i\leftrightarrow j} \parallel E_{k_{i\to j}} \\ & \cdot \left(\mathrm{APID}_i \parallel \mathrm{PID}_i \parallel E_{k_{i\leftrightarrow \mathrm{bs}}}\left(D_i\right)\right) \parallel \mathrm{HMAC}_{k_{i\leftrightarrow \mathrm{bs}}} \\ & \cdot \left(\mathrm{PID}_i \parallel D_i\right). \end{aligned} \tag{14}$$

Padding is added to make sure all the one-hop packets have the same size to prevent ADV from identifying the source by *traffic analysis* and *size correlation* attacks. When SN_j receives the packet, it will reformat the packet as in expression (15) and then will send it to SN_r:

$$\begin{aligned} M_{j\to r} = {} & \mathrm{APID}_i \parallel \mathrm{OHPID}_{j\leftrightarrow r} \parallel E_{j\to r} \\ & \cdot \left(\mathrm{APID}_j \parallel \mathrm{PID}_i \parallel E_{k_{i\leftrightarrow \mathrm{bs}}}\left(D_i\right)\right) \parallel \mathrm{HMAC}_{k_{i\leftrightarrow \mathrm{bs}}} \\ & \cdot \left(\mathrm{PID}_i \parallel D_i\right). \end{aligned} \tag{15}$$

The transmission of $M_{j\to r}$ should be heard by all the neighbors including both SN_i and SN_r. If SN_i hears the message and reads $APID_i$, the SN_i knows that $M_{i\to j}$ was received correctly by SN_j. Only at this time, SN_i updates the value of $OHPID_{i\leftrightarrow j}$. PID_i will be updated, as well, since SN_i is the source of the message. Here are two scenarios.

Scenario 1. The packet sent by SN_i is lost or got corrupted. In this scenario, SN_j considers nothing happened, so it will not forward any message onward. Meanwhile, SN_i will wait for ζ time to expire. It will send the message again with updated $APID_i$. Once the message is acknowledged according to the procedure explained earlier and if the SN is the source, then PID_i, $OHPID_i$, and $APID_i$ will be updated and be ready for a new packet to be sent out. If it is intermediary SN, only $OHPID_i$ and $APID_i$ are updated.

Scenario 2. The packet is received correctly by SN_j, the new packet $M_{j\to r}$ is sent out including the acknowledgement $APID_i$, and SN_j updated the value of $OHPID_{i\leftrightarrow j}$. However, SN_i does not hear the forwarded message $M_{j\to r}$ within time ζ. At this moment, SN_i does not know for sure if the packet was delivered or the acknowledgement is lost. It has to account for the worst-case scenario. A copy of the message will be retransmitted to SN_j with the current $OHPID_i$ and updated $APID_i$. SN_j can recognize the message because of the value of old $OHPID_i$. After receiving the *retransmitted* packet, it now sends direct acknowledgement to SN_i as in the following expression:

$$\mathrm{ACK}_{i\leftarrow j} = \mathrm{APID}_i \parallel \mathrm{Padding}. \tag{16}$$

BS is treated similar to normal SN, so it has to acknowledge every packet it receives. After the packet is delivered to the BS, and after the message is acknowledged, the PID_i (of the source) will be updated. Both the SN_i and the BS will be ready to exchange a new message. As long as the new message does not reach the BS before the old PID_i is updated, the system will continue to be synchronized. This way, we have a possible window of one message. However, there is no guarantee to have all messages arrive in order. The only way to do it is if the system accounts for minimum interval time $\omega_{\min}$, which is the minimum time span between two messages. However, in some cases, the packet cannot be delivered at all through a certain route. One obvious scenario is when an intermediate node is depleted out of energy before it forwards the message onward. The message needs to be rerouted through a different intermediate node which could cause a sever delay. We suggest implementing a sliding window mechanism as exhibited in Figure 2. For each sensor, we can have a window of W size. If we have k bits for the sequence number of the PID, then we can have a sliding window of $2^k - 1$ slots. Once the BS finds out that the sequence of PIDs is not synchronized, it will send a message to resynchronize the sensor with the BS.

5.2.4. Transmission as a Broadcaster. Typically, the BS is required to broadcast messages for control and management purposes. Likewise, a sensor might need to broadcast a message to the BS for network setup, maintenance, and other management issues. A sensor, as well, could broadcast a message for emergency or urgency reasons depending on the application at hand. The framework requires keeping all the packets, transmitted throughout the network, indistinguishable by the adversary. Thus, all the messages need to have the same *size*. Broadcast is *one-to-many* transmission; a SN_i can broadcast a message to all the neighbors in the set N_i. Each SN is preloaded with a broadcast key (kb_i) and assigned broadcast pseudonym ID ($BPID_i$). The broadcast message sent by SN_i is formatted as in the following expression:

$$M_b = \mathrm{Padding} \parallel \mathrm{BPID}_i \parallel E_{\mathrm{kb}_i}\left(D_b\right). \tag{17}$$

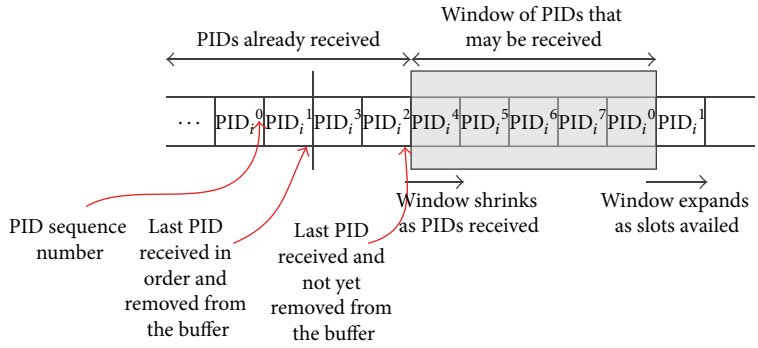

FIGURE 2: Sliding window for received PIDs [19].

The broadcast message from a source SN_i will be received by all the neighbors $\in N_i$. SN_i and the recipients will update $BPID_i$ according to the following expression:

$$BPID_i = H_1\left(BPID_i \oplus b_i\right). \tag{18}$$

Upon receiving the broadcast message (M_b), SN_j decrypts the message using the stored key (kb_i) in the table TBL_j. It then encrypts it again using kb_j and broadcasts M_b to its one-hop neighbors set (N_j) as in the following expression:

$$M_b = BPID_i \parallel BPID_j \parallel E_{kb_j}\left(D_b\right). \tag{19}$$

When the BS receives a broadcast message, it is ultimately the destination, so intuitively it does not need to broadcast the message again. Our proposed framework assumes that the BS behaves like a normal sensor. To maintain this precourse, we require the BS to broadcast the message again. Thus, we introduce the limited broadcast where the BS will broadcast only to one hop with TTL = 1. Likewise, to reduce the unnecessary traffic, we have required the intermediary nodes to rebroadcast only to the neighbors with smaller HC in case of uplink messages and to neighbors with bigger HC in case of downlink messages. What if the neighbors, with less HC, are all dead? We could end up with a section completely disconnected from the BS. The protocol needs to consider broadcast messages to all the neighbors in this case.

5.2.5. Limited Broadcast Messages. A sensor inside network maze can only recognize the neighbor sensors and the BS. When SN broadcasts a message uplink, then all sensor neighbors should hear it. When intermediary sensor gets broadcast message, it should rebroadcast the message if and only if the message is received from SN with bigger HC. This way the message is directed to the BS through multiple routes while saving unnecessary traffic. The TTL = HC_{max} where HC_{max} preloaded at the startup phase. The value will keep reduced by one until it gets to the BS. In case of the downlink messages from the BS to all the sensors, the intermediary sensor would rebroadcast the message if and only if it is coming from a SN with smaller HC. In this case the message will hold TTL = 0 without change as an indication of BS as the source until it reaches all the sensors. The special case of broadcast is when TTL = 1 where the message will be broadcasted to one hop only.

5.2.6. Fake Broadcast Message. The sensors need to send fake messages to prevent *time correlation* attack and statistical analysis. The rate of the fake messages follows a certain protocol. Fake message is a one-hop broadcast message. However, to prevent correlation, the message needs to behave similar to real messages. Therefore, the message needs to be encrypted and has similar size as real messages. This will make the fake messages indistinguishable from the real messages. Since it has to carry a dummy data, we chose to make it carry *residual energy* (Δ) of the source sensor. This information will be extracted by the recipient neighbors and saved in the related tuple in the table TBL. The fake broadcast message sent by SN_i is explained in the following expression:

$$M_f = \text{Padding} \parallel FPID_i \parallel E_{kf_i}\left(\Delta i\right). \tag{20}$$

The fake broadcast message from SN_i will be received by all the neighbors, for example, SN_j, SN_u, $SN_v \in N_i$. SN_i and the recipients will then update $FPID_i$ according to expression

$$FPID_i = H_1\left(FPID_i \oplus c_i\right). \tag{21}$$

There is no need to worry about the pseudonym synchronization since the main purpose of fake message is to show activity in idle sensors to obfuscate real messages.

5.3. SN Removal. There are many reasons why we need to remove a sensor from the network. For instance, when the battery of the sensor is about to deplete, it should refrain from participating in transmission in preparation of removing it from the network. This would protect against lost data and keep the pseudonyms synchronized.

In some cases, WSN uses intrusion detection system (IDS) [26], to protect against active attacks. Once SN is captured by the ADV, SN must be removed from the network and banned from participation in data transmission. Procedurally, if SN_i opts to be removed, it sends two messages requesting the removal. The first message is to the BS as in expression

$$M_{i\to j} = OHPID_{i\leftrightarrow j} \parallel E_{k_{i\to j}}\left(PID_i \parallel E_{k_{i\leftrightarrow bs}}\left(D_{\text{remove}}\right)\right). \tag{22}$$

The tuple of the SN_i in the BS tables will be disabled permanently. The SN_i will send a broadcast message to the neighbors according to

$$M_b = \text{Padding} \parallel \text{BPID}_i \parallel E_{\text{kb}_i}(D_{\text{remove}}). \quad (23)$$

Once the neighbors get the message D_{remove}, they will delete the tuple related to SN_i from the tables. Neighbors will refrain from exchanging messages with SN_i. The same process could be established by the BS to remove a captured SN from the network.

5.4. SN Addition. To add SN_i to the network, it will be preloaded with all setup parameters: ID_u, a_u, b_u, c_u, H_1, $k_{u\leftrightarrow \text{bs}}$, and kb_u, and fkb_u. Right after deployment, SN_i calculates the shared parameters with its neighbors. The BS should be trusted to run the process. Before sharing the calculated parameters with the SN_i, the neighbors authenticate the new sensor. The BS will send special key k_{add} to all the nodes by a broadcast message. SN_i will be preloaded with the same key as well. SN_i and the neighbors will use this special key to authenticate with each other. Initially, the BS sends the following broadcast message to all of its one-hop neighbors as in the following expression

$$M_b = \text{Padding} \parallel \text{BPID}_{\text{bs}} \parallel E_{\text{kb-bs}}(D_{\text{add}}), \quad (24)$$

where D_{add} is expressed in expression

$$D_{\text{add}} = \text{hc} \parallel k_{\text{add}}. \quad (25)$$

The initial value for hc is 0 where it represents the hop count. It will be incremented every time the message is forwarded.

6. Security Analysis

We need to analyze our solution for both passive and active ADV attacks. The ADV has global view of the WSN. Usually, ADV starts by monitoring transmission somewhere in the network and then attempts to capture the location of source SNs or the BS. Once ADV determines the identity and location of a source SN, it consequently launches active attacks against certain SNs and tries to disrupt the operation of the entire WSN. The main strength of passive ADV is the fact that neither SNs nor the BS will know about their existence since they act passively. However, active attacks can be detected if the WSN implements a good IDS.

6.1. Security against Passive Attacks. SNs use disposable pseudonyms to recognize each other instead of using the real IDs. There is absolutely no real ID stored in the SN and absolutely no one pseudonym is used more than once. In addition, data is encrypted all the way from the source to the destination using pairwise shared keys between sources and the BS.

Eavesdropping and Content Analysis. ADV can intercept messages without being able to decrypt them because they are encrypted. The only information ADV can get of a captured message is the pseudonyms which are all temporary IDs and have no further use except to calculate new pseudonyms. However, the ADV cannot get, from the captured messages, important parameters $a_{i\leftrightarrow j}$, b_i, or c_i which are required to calculated new pseudonyms. Thus, the ADV will not be able to create new pseudonyms.

Hop-by-Hop Trace. ADV can track hop-by-hop messages from one node to another by overhearing the transmitted messages to capture either the source SN or the BS. The ADV will be faced with many real and fake transmissions, which are identical, during the lifetime of the WSN. Furthermore, each node uses different routes and the message content changes completely after each transmission.

Size Correlation. This attack would not work for our framework since all the messages have commensurate size whether it is real or fake.

Identity Correlation. ADV cannot relate overheard IDs to SNs. It is not possible since SNs use different pseudonyms every time a message is transmitted.

Rate Monitoring. ADV tries to find different transmission rates in the network, such as having higher transmission rate nearby the BS or busy SNs. This is handled by issuing fake messages all over the network to maintain similar transmission rate over the lifetime of the WSN.

6.2. Security against Active Attacks. ADV could *physically* compromise SN which gives the ADV the advantage of monitoring messages sent and received through the captured SN. We assume ADV knows encryption protocols used by the system while the system hides *encryption keys* and IDs. In such attacks, ADV tries to compromise SNs to get information related to security of the system. Consequently, it will monitor all packets going through the compromised SN to capture the identities and important parameters. Once ADV captures some privacy information, it reports it to external executers to do further damages (such as killing the Panda). Yet the IDS cannot detect that the SN is physically captured due to the passive behavior of ADV. In some other attacks, ADV captures SNs and actively forge messages, send replay messages, and do other forms of invasive attacks. Moreover, ADV could load powerful computers with captured credentials to launch more catastrophic attacks.

If ADV physically compromises a SN, then it captures two sets of information:

(i) information related to the node itself: the current pseudonym PID, the parameters used to calculate the pseudonyms, the hash functions, the keys, and other information as listed in Notations;

(ii) information related to the neighbors as listed in Table 1.

In this case, the ADV will have all it needs to issue new pseudonyms and send messages out to the neighbors.

Scenario 1. If ADV physically compromises SN_i and if SN_j and $SN_r \in N_i$, so SN_i knows some information about both SN_j and SN_r. However, it cannot calculate important information such as $a_{j\leftrightarrow r}$ which is required for one-hop communication between SN_j and SN_r [16]. The reason is SN_i would need ID_j and ID_r, which are both deleted at the end of the startup phase. If SN_i hears a message transmitted among the neighbors, then it cannot determine the source SN.

Scenario 2. If ADV physically compromises multiple SNs, let us call it set N_{cs}, and collects number of messages, let us call it set N_{cm}. Then, the number of compromised PIDs is equal to N_{cm} since each message has one unique PID. If the source $SN_i \notin N_{cs}$, then ADV cannot know the source node [16].

Scenario 3. If the message sent by source SN_i as in Scenario 2 passes through $SN_j \in N_{cs}$ or even through multiple compromised SNs, it will not be able to correlate the captured PID with SN_i.

Scenario 4. If a message sent by source SN_i and $\forall SN \in N_i$ is also $\in N_{cs}$ (all neighbors are *physically* compromised), then ADV will be able to know that SN_i is the source.

It is unrealistic situation to have many compromised SNs. However, this proves that one or very few *physically compromised* SNs do not for sure leak the ID and location of the SN. None of the literature we have seen discussed this worse-case situation where you have physically compromised SNs. We have proved that even if we have some physically compromised nodes, our system still can limit the leak of the IDs and locations.

6.3. BS Security. ADV can learn about a SN receiving a message in two ways: (i) when SN retransmits the message, which is traced by ADV and (ii) the ADV being able to correlate the ID with the recipient SN. ADV cannot locate the BS by one compromised SN because every message uses a different pseudonym. It will need to compromise multiple SNs along the path to the BS, providing that compromised SNs can collude together. The goal of this model is to delay the capturing of the BS in the presence of multiple captured and colluding SNs.

Let us presume $SN_r \in N_{cs}$. It issues a message with D_b such that $APID_r \parallel OHPID_{r\leftrightarrow u} \parallel E_{r\to u}(APID_r \parallel PID_r \parallel E_{k_{r\leftrightarrow bs}}(D_b))$ [4]. If ADV compromises multiple nodes along the path to the BS where it can decrypt the data at each SN to always read this signature ($PID_r \parallel E_{k_{r\leftrightarrow bs}}(D_b)$). Reading the similar signature, while ADV knows that every message should be directed uplink to the BS tells the ADV that it is in the right path to get the BS. Compromised SNs can even collude to force the message to route through certain area in effort to focus the search for the BS. This way, it will be able to locate the BS sooner which is a function of (i) the size of the network, (ii) the traffic density in the network, and (iii) the number of compromised SNs. To solve this issue, the message signature ($PID_r \parallel E_{k_{r\leftrightarrow bs}}(D_b)$) needs to be scrambled at every hop. So, every message will be forwarded to the next hop as follows:

$$M_{u\to x} = APID_u \parallel OHPID_{u\leftrightarrow x} \parallel E_{u\to x} \cdot \left(APID_u \parallel PID_r \parallel PID_u \parallel E_{k_{u\leftrightarrow bs}}\left(E_{k_{r\leftrightarrow bs}}(D_i)\right)\right). \quad (26)$$

We have added multiple levels of encryption, which will be done at every hop using the shared key between the hop and the BS. In addition, PID of the hop will be added in sequence, so the BS can do the decryption in sequence upon message arrival. This solution increases the size of the message proportionally to the number of hops the message goes through. It will also add multiple levels of decryption at the BS. We suggest having the onion encryption done for a distance of few hops, O_h [4]. So, if $O_h = 2$, then we have only two extra encryptions. In addition, we need to account for two PID's added to the message. This size of O_h can be adjusted by the closed-loop control system according to the security needs and the WSN environment.

6.4. SLP and BSLP. SLP and BSLP are achieved at first by having successful source, link, and BS anonymity, which was thoroughly argued earlier. ADV cannot infer any information from the intercepted messages. Passive attacks will not endanger the location privacy. Having IDS will handle active attacks. However, having active attacks by physically compromised SNs could hinder the location privacy if many SNs are captured in one area.

6.5. Timing Privacy. By using fake messages at variable interval times, it becomes super hard for the ADV to correlate messages being transmitted over the network as exhibited in Figure 3.

6.6. Routing Privacy. Although shortest path routing is used in this framework, choosing the next hop is done according to certain *probabilistic* algorithm, which accounts for the residual energy levels and usage frequency to increase the route privacy, as exhibited in Algorithm 1 and Figure 4. ADV cannot relate routes to nodes [20]. Even if two messages follow the exact same route, ADV will see them as if they are two different routes since each node along the route has different PIDs.

6.7. Data Privacy. Messages are encrypted before transmission and reencrypted at every hop along the route to the BS. Data will be authenticated by a message digest created by a secure hash function. The physically compromised nodes are able to inject data in the network but good IDS can detect the falsified data. The system adopts a secure procedure to remove compromised SNs from the WSN and adopts another procedure to add sensors as needed.

6.8. Energy and Location Safety Period. In this work, we will assume a simple energy consumption model [20, 27, 28].

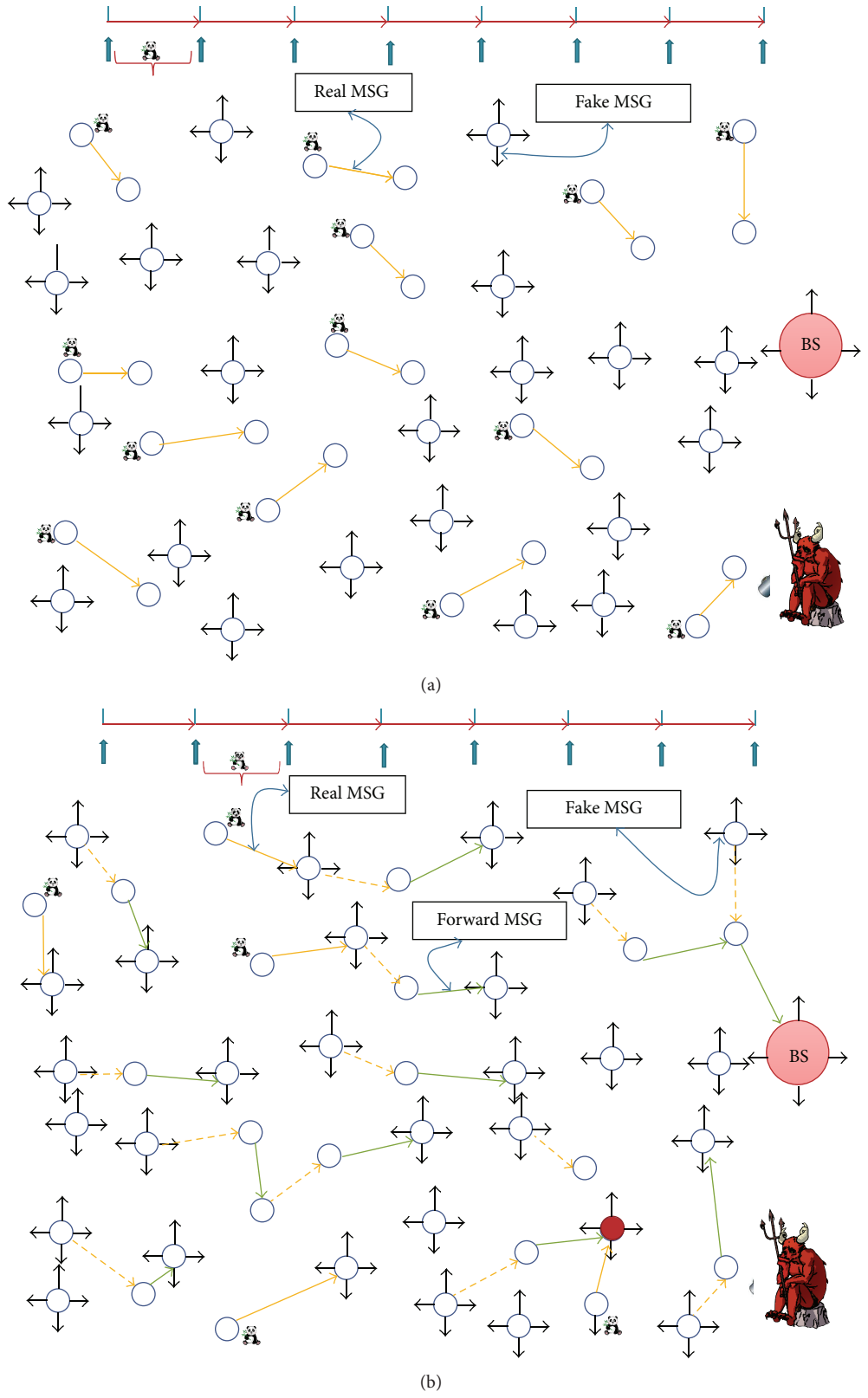

FIGURE 3: (a) Sensors sense events and send real messages while the rest send fake messages, with fake MSG's: $P_r = 1/N = 1/48 = 2\%$ and without fake MSG's: $P_r = 1/n = 1/11 = 9\%$. (b) Sensors sending new real or forward messages will not send fake messages. The more the network gets busy the less fake messages transmitted [19, 20].

```
Procedure: SELECT_FORWARD_NODE_PROC;
Input: N_i
Output: Messages out
Δ = 0;
for i in N_i                          // total number of neighbors
{Δ = Δ + SN[i, Δ];}
Average_Δ = Δ/N_i;
for i in N_i
{
        x = Average_Δ;
        if (SN[i, link] == uplink) && (SN[i, Δ] >= x)
        J = SN[i, OHPID]
        X = SN[i, Δ];
}
If j != null
Return;
for i in N_i
{
        x = Average_Δ;
        if (SN[i, link] == equal-link) && (SN[i, Δ] >= x)
        J = SN[i, OHPID]
        X = SN[i, Δ];
}
If j != null
Return;
for i in N_i
{
        x = Average_Δ;
        if (SN[i, link] == down-link) && (SN[i, Δ] >= x)
        J = SN[i, OHPID]
        X = SN[i, Δ];
}
If j != null
Return;
```

Algorithm 1: Selecting the forward SN according to the residual energy level.

The radio consumes $\mathcal{E}$ nJ/bit for both transmission and reception by the SN's circuitry. In addition, it consumes ε nJ/bit/m^2 for the transmitter amplifier to achieve an acceptable signal to noise ratio. Therefore, to transmit k bits for r distance, the total transmission energy dissipation will be

$$E_{\text{trans}} = k * \mathcal{E} + k * r^2 * \varepsilon. \tag{27}$$

Moreover, the receiver would consume for reception of k-bit message

$$E_{\text{receiv}} = k * \mathcal{E}. \tag{28}$$

The energy spent for transmission or reception is almost constant per a message since we have fixed-size messages to prevent size correlation attacks. If we have p percent of the nodes issue fake data at a specific period, then p percent of the energy and the bandwidth is wasted for the sake of WSN rate privacy. The consumption of transmitting fake messages is a double-fold since the transmitter will consume E_{trans} for every message and all the neighbors n will consume $(n * E_{\text{trans}})$. When the transmission range increases, n increases. The total energy consumed in the network to send real messages in one interval, where (q) is the average percentage of SNs sending or forwarding real messages and (n) is the average SN neighbors [28], is as follows:

$$E_R = (q * N)\left(k * \mathcal{E} + k * r^2 * \varepsilon\right) + (q * N * n)(k * \mathcal{E}). \tag{29}$$

The exact value of q cannot be known since it is an event driven value, which depends on the situation of the WSN at that time. The total energy consumed in the network to send fake messages in one interval where (p) is the specified percentage value of fake messages at one interval is as follows:

$$\begin{aligned} E_F = {} & (p) * N * \left(k * \mathcal{E} + k * r^2 * \varepsilon\right) \\ & + (p) * N * n * (k * \mathcal{E}). \end{aligned} \tag{30}$$

Energy transmission efficiency ETE [20] can be calculated as

$$\text{ETE} = \frac{E_R}{E_R + E_F}. \tag{31}$$

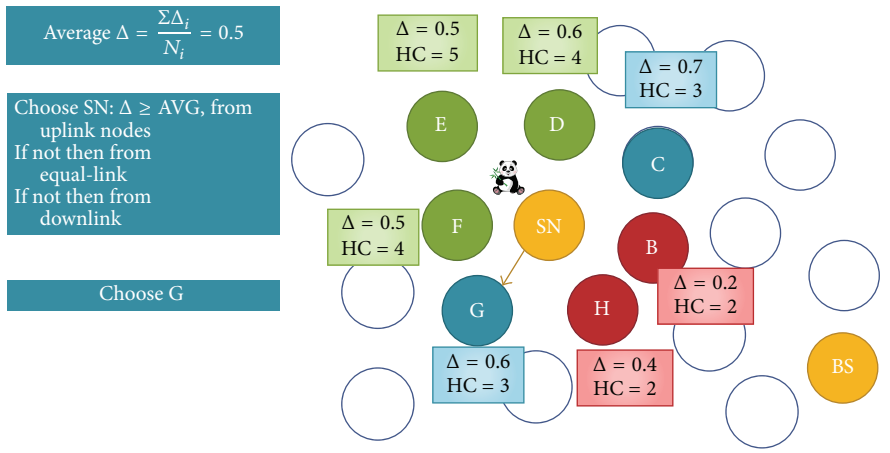

Figure 4: How to choose the forwarding node according to the energy levels of the neighboring sensors [19, 20].

The energy safety period is the time when the first sensor is depleted. However, transmission of data is distributed among all sensors to make sure the safety period lasts longer. The level of the energy in every sensor is measured and considered as a decision factor (among others) for selecting the next-hop for the message. This will definitely help to distribute the network accumulative energy consumption over the whole WSN. The location safety period is the period between capturing the first packet and finding out the location of the source or the BS. It is measured as the ratio between the period of discovering the location and the interval period [25]. We have implemented a WSN of 300 SNs uniformly distributed over 200 × 200 area. The average distance between SNs is 15. For the simulation, a SN sends only 10% fake message of the total messages. That means that, among ten real messages, it will send only one fake message since we presumed a busy network. This could be scaled up or down. Increasing the fake message will increase the power dissipation.

Figure 5 shows that EEAC provides improved source safety period compared to EAC [4]. Figure 6 shows that EEAC also provides a stronger BS safety period compared to EAC [4]. However, the source safety period is much more than the sink safety period in both schemes. It is always harder to achieve BSLP due to the volume of transmissions nearby the BS. The efficiency of energy dissipation is due to adopting Algorithm 1 for selecting the forward SN where it takes in consideration the residual energy levels of the SNs, which was not considered in EAC. Figure 7 shows the delay in EEAC is very comparable to the EAC due the extra processing and security implementation. Furthermore, Figure 8 shows that the average total delivery time source-to-BS is also very comparable to EAC.

7. Performance Evaluation

In this section, we evaluate the performance of our system, by evaluating the storage, processing, computational, and communication costs.

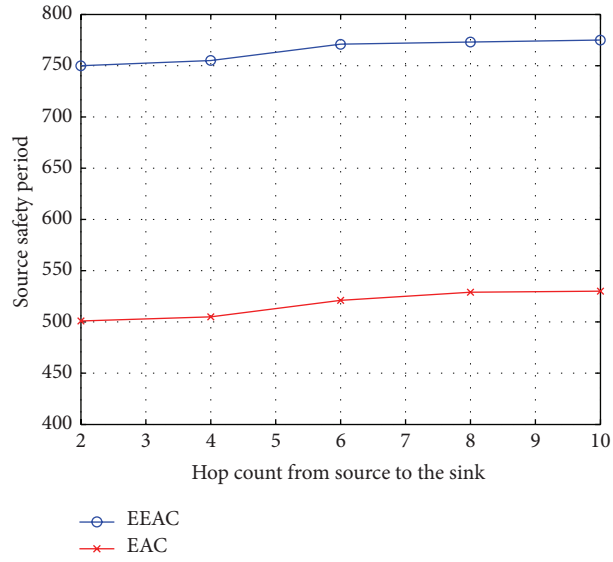

Figure 5: Source safety period.

7.1. Storage Evaluation. There are two sets of information stored in a SN_i: (i) information related to the sensor itself, such as random numbers: (a_i, b_i, c_i), pseudonyms: $(PID_i, BPID_i, FPID_i, APID_i)$, keys: $(k_{i\leftrightarrow bs}, kb_i, fkb_i)$ and (ii) information related to each neighbors which includes random numbers: $(a_{i\leftrightarrow j}, b_j, c_j)$, pseudonyms: $(OHPID_{i\leftrightarrow j}, BPID_j, FPID_j)$, and keys: $(k_{i\leftrightarrow j})$, Misc: $(link_{i\leftrightarrow j}, \Delta_j)$.

If we presume that the keys, the random numbers, the pseudonyms, and the hash functions are all n bits long in average and the required bits for miscellaneous data altogether are *two* bytes and the average number of neighbors N_{ave}, then the total storage memory required is

$$\text{Storage} = 10n + (7n + 16) * N_{ave}. \quad (32)$$

Chen et al. [16] indicated the storage for SAS, CAS, APR, DCARPS, and EAC. We also calculated the storage for PhID, ACS, HIR, and RHIR. All are listed in Table 2. The size of storage proportionally increases when the size of n increases.

Table 2: Performance comparison. N is the total number of sensors; N_{ave} is the average number of neighbors; k is number of stored hash values where the SN stores k hash values per one neighbor which are calculated in advance at startup phase [4, 16, 19, 25].

	Scheme	Storage cost (bits)	Computation cost
1	SAS	$2nN + 4nN_{ave} + 16$	No hashing operations
2	CAS	$6n + 7nN_{ave} + 16$	Two hashing operations and two encryptions
3	HIR	$2n + 2nN_{ave}$	One hashing function
4	RHIR	$2n + 2nN_{ave} + nkN_{ave}$	No hashing functions
5	APR	$9n + 7nN_{ave} + 2N - 2N_{ave} - 2$	Six hashing functions
6	DCARPS and Global DCARPS	$3n$	No hashing functions
7	ACS	$5nN_{ave}$	Two hashing functions
8	PhID	$(3n + 2) * N_{ave}$	Four hashing functions
9	EAC	$6n + 6nN_{ave} + 2$	Four hashing operations
10	E^2AC	$10n + (7n + 16) * N_{ave}$	Four hashing operations and O_h encryptions

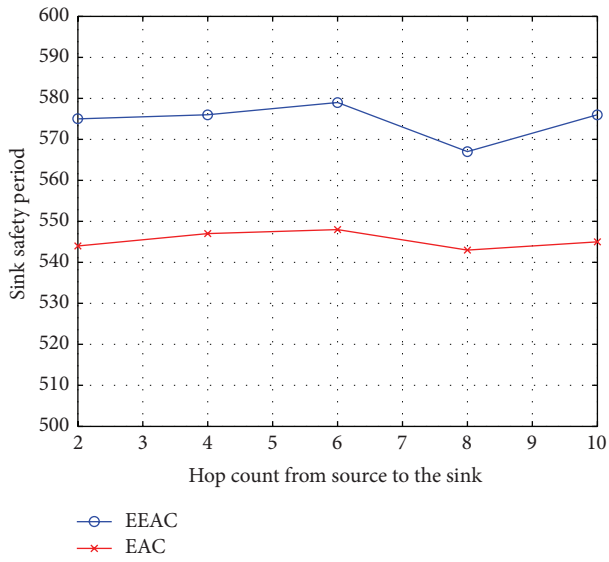

Figure 6: BS safety period.

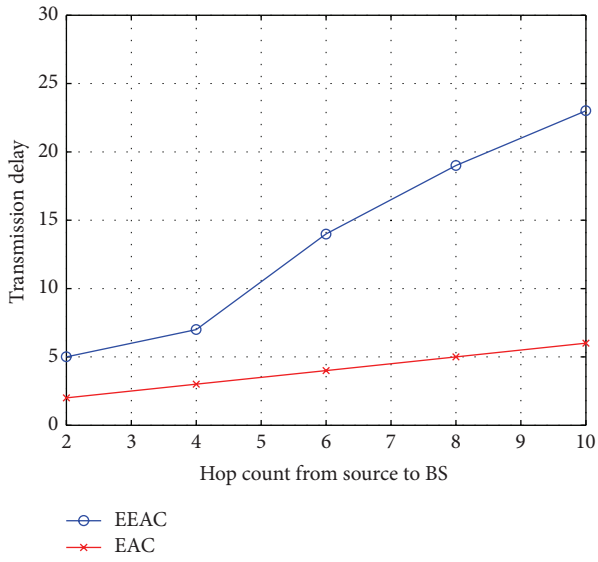

Figure 7: Transmission delay.

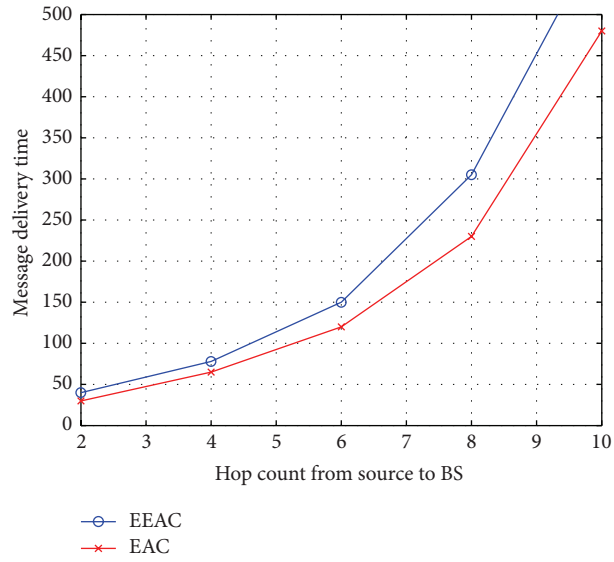

Figure 8: Message delivery time.

The most common hashing functions are MD4 [29] which uses 128-bit digest, SHA-1 [29] which uses 160-bit digest, and *Whirlpool* [29] which uses 512-bit digest. The size of the storage would increase when the number of neighbors increases. Each SN has limited flash memory size, which could confine the maximum number of neighbors that a sensor can fit. As an example, TelosB mote [11, 16] has 1 MB external flash memory. Thus, if one neighbor node requires 1.2 k bits of storage memory, then TelosB could have more than 800 neighbors, which is very much more than what is needed in practical networks. Although our model shows a bit of increase in the storage required to store the pseudonyms but it is the only one among discussed protocols in this work that provides a steady and functional anonymity and location privacy under sever global and active attack. In addition, the current technology provides enough storage, which makes it no issue at all.

7.2. Processing and Computational Evaluation. Hash functions are used to calculate the pseudonyms and symmetric cryptography is used to encrypt the messages. Because we need to calculate three pseudonyms and one acknowledgement after each transmission, thus using encryption to create pseudonyms was avoided since it requires more processing power compared to hash functions. When a SN senses data, it needs to have OWH calculations for PID, OHPID, and APID at the sender and OHPID at the receiver. If the system opts for data authentication, then another hash function is needed. The source node needs only one encryption for the data if $O_h = 0$; however, it needs O_h more encryptions if onion fashion is used [4]. Each intermediary node needs one decryption operation and then another encryption for the new data. Chen et al. [16] indicates that SAS does not use hashing or encryption to create pseudonyms because it uses already created pseudonyms from a space. The other scheme by Chen et al. [16], CAS, uses two hashing operations and two encryption operations. APR uses at least six hashing functions. DCARPS uses constant IDs, so there are no hashing functions or encryptions for creating IDs. EAC has four hashing operations. None of the other schemes can achieve privacy against global threats and active adversary attacks. The power consumption due to the additional encryption operations is marginal compared to the power consumption caused by data transmission.

7.3. Communication Cost Evaluation. The most expensive operation for power consumption is transmission of bits from one node to another. We use two stages for air communication in our framework: (i) the startup phase and (ii) the communication phase. The data transmission during startup phase is minimal since the BS beacon and the neighbors' information exchange happens once before the steady communication phase. During communication phase, data will be forwarded hop-by-hop to the BS. Every packet is equally sized to prevent time and size correlation. We have introduced a probabilistic fake packet transmission scheme which none of the other protocols adopted. When a SN does not have real message, it will send fake message according to the protocol discussed earlier.

The cost per message at one interval time is

$$\text{Average Message Cost} = \frac{R + (N - R)\, P_r + A}{R}, \quad (33)$$

where R is the total number of SNs with real messages at one interval time, P_r the probability of sending fake message by SNs, and A the average number of acknowledgements in one interval. None of the other schemes addressed the issue of rate analysis attacks, which is one of the easiest attacks any adversary can use. Using fake messages is an expensive solution. However, we have designed our model to be adaptive to the network traffic situation by using a closed-loop system. The BS can always increase or decrease the amount of fake messages used according to the reports it is getting about the system security.

8. Conclusions and Future Work

The scheme presented in this work provides source, link, and BS anonymity, SLP and BSLP. Most of the previous work assumed local adversary view and passive attack model. This work addressed local and global adversary network view. It handles both passive and active attack models. Anonymity cannot provide temporal privacy. To provide temporal privacy, the global adversary needs to see a maze of transmissions happening all over the network. Fake messages were introduced. However, using fake messages needs to be adjusted to manage the energy consumption. The presented model can handle both homogenous and heterogeneous sensor nodes in terms of initial energy levels. It uses the energy level as the main indicator in deciding how to route packets forward. We have demonstrated that our framework can withstand most of the known passive and active attacks. We have discussed many scenarios and provided solutions. The storage cost was mathematically analyzed for the framework. We also have discussed the complexity of computational operations performed in the system, which includes encryption and hash functions. To provide security against multi colluding active attackers, we have introduced onion encryptions. The future work would include enhancement on the probabilistic scheme for fake messages usage. We also will implement our model for clustered networks.

Notations

a_i:	Random number shared between SN_i and BS
b_i:	Random number shared between SN_i and neighbors
c_i:	Random number shared between SN_i and neighbors
H:	Hash function to create pseudonyms, the keys, and data digest
$k_{i\leftrightarrow bs}$:	Pairwise key shared between SN_i and BS
kb_i:	Broadcast key for SN_i
fkb_i:	Fake broadcast key for SN_i
N:	Number of SNs
N_i:	Number of neighboring
$HC_{i\leftrightarrow bs}$:	Hop-count between SN_i and BS
PID_i:	Pseudonym ID shared between SN_i and BS
$BPID_i$:	Broadcast pseudonym ID
$a_{i\leftrightarrow j}$:	Random value shared between SN_i and SN_j
$k_{i\leftrightarrow j}$:	Pairwise key shared between SN_i and SN_j
$OHPID_{i\leftrightarrow j}$:	Pseudonym ID shared between SN_i and SN_j
$APID_i$:	ACK pseudonym ID for SN_i
$BPID_i$:	Fake broadcast pseudonym ID
TBL_i:	Table in SN_i for shared parameters
TIME_STAMP:	Time stamp
D_i:	Sensed data transmitted inside the message

SEQ_NO:	Sequence number for a message
TTL:	Time to live
MCG_LGTH:	Message size
$\Delta_{residual}$:	Residual energy
$\oplus$:	XOR Operation
$\|\|$:	Concatenation operation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. Conti, J. Willemsen, and B. Crispo, "Providing source location privacy in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1238–1280, 2013.

[2] A. A. Nezhad, A. Miri, and D. Makrakis, "Location privacy and anonymity preserving routing for wireless sensor networks," *Computer Networks*, vol. 52, no. 18, pp. 3433–3452, 2008.

[3] L. Yao, L. Kang, P. Shang, and G. Wu, "Protecting the sink location privacy in wireless sensor networks," *Personal and Ubiquitous Computing*, vol. 17, no. 5, pp. 883–893, 2013.

[4] A. Abuzneid, T. Sobh, and M. Faezipour, "An enhanced communication protocol for anonymity and location privacy in WSN," in *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC '15)*, 2015.

[5] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pp. 599–608, Columbus, Ohio, USA, June 2005.

[6] C. Ozturk, Y. Zhang, W. Trappe, and M. Ott, "Source-location privacy for networks of energy-constrained sensors," in *Proceedings of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, pp. 68–72, May 2004.

[7] J. Deng, R. Han, and S. Mishra, "Countermeasures against traffic analysis attacks in wireless sensor networks," in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pp. 113–126, September 2005.

[8] Y. Jian, S. Chen, Z. Zhang, and L. Zhang, "A novel scheme for protecting receiver's location privacy in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 10, pp. 3769–3779, 2008.

[9] X. Li, X. Wang, N. Zheng, Z. Wan, and M. Gu, "Enhanced location privacy protection of base station in wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks (MSN '09)*, pp. 457–464, IEEE, Fujian, China, December 2009.

[10] Y. Ouyang, Z. Le, Y. Xu et al., "Providing anonymity in wireless sensor networks," in *Proceedings of the IEEE International Conference on Pervasive Services (ICPS '07)*, pp. 145–148, July 2007.

[11] S. Misra and G. Xue, "Efficient anonymity schemes for clustered wireless sensor networks," *International Journal of Sensor Networks*, vol. 1, no. 1-2, pp. 50–63, 2006.

[12] R. Di Pietro and A. Viejo, "Location privacy and resilience in wireless sensor networks querying," *Computer Communications*, vol. 34, no. 3, pp. 515–523, 2011.

[13] J.-P. Sheu, J.-R. Jiang, and C. Tu, "Anonymous path routing in wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pp. 2728–2734, May 2008.

[14] X. Luo, X. Ji, and M.-S. Park, "Location privacy against traffic analysis attacks in wireless sensor networks," in *Proceedings of the International Conference in Information Science and Applications (ICISA '10)*, pp. 1–6, Seoul, Republic of Korea, April 2010.

[15] J.-H. Park, Y.-H. Jung, H. Ko, J.-J. Kim, and M.-S. Jun, "A privacy technique for providing anonymity to sensor nodes in a sensor network," in *Ubiquitous Computing and Multimedia Applications*, T.-H. Kim, H. Adeli, R. J. Robles, and M. Balitanas, Eds., vol. 150, pp. 327–335, Springer, Berlin, Germany, 2011.

[16] J. Chen, X. Du, and B. Fang, "An efficient anonymous communication protocol for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 14, pp. 1302–1312, 2012.

[17] Y. Ouyang, Z. Le, D. Liu, J. Ford, and F. Makedon, "Source location privacy against laptop-class attacks in sensor networks," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks (SecureComm '08)*, pp. 1–10, ACM, Istanbul, Turkey, September 2008.

[18] J. Chen, H. Zhang, B. Fang, X. Du, L. Yin, and X. Yu, "Towards efficient anonymous communications in sensor networks," in *Proceedings of the 54th Annual IEEE Global Telecommunications Conference (GLOBECOM '11)*, pp. 1–5, December 2011.

[19] A.-S. Abuzneid, T. Sobh, M. Faezipour, A. Mahmood, and J. James, "Fortified anonymous communication protocol for location privacy in WSN: a modular approach," *Sensors*, vol. 15, no. 3, pp. 5820–5864, 2015.

[20] A. Abuzneid, T. Sobh, and M. Faezipour, "Temporal privacy scheme for end-to-end location privacy in wireless sensor networks," in *Proceedings of the International Conference on Electrical, Electronics, Signals, Communiction & optimization (EESCO '15)*, pp. 2476–2481, 2015.

[21] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran, "Toward a statistical framework for source anonymity in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 248–260, 2013.

[22] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 500–528, 2006.

[23] I. Mabrouki and A. Belghith, "E-SeRLoc: an enhanced serloc localization algorithm with reduced computational complexity," in *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC '13)*, pp. 153–158, July 2013.

[24] L. Lazos and R. Poovendran, "SeRLoc: secure range-independent localization for wireless sensor networks," in *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '04)*, pp. 21–30, ACM, Philadelphia, Pa, USA, October 2004.

[25] H. Chen and W. Lou, "From nowhere to somewhere: protecting end-to-end location privacy in wireless sensor networks," in *Proceedings of the IEEE 29th International Performance Computing and Communications Conference (IPCCC '10)*, pp. 1–8, IEEE, Albuquerque, NM, USA, December 2010.

[26] A. Abduvaliyev, A.-S. K. Pathan, J. Zhou, R. Roman, and W.-C. Wong, "On the vital areas of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1223–1237, 2013.

[27] M. A. Abuhelaleh, T. M. Mismar, and A. A. Abuzneid, "Armor-LEACH—energy efficient, secure wireless networks communication," in *Proceedings of the17th International Conference on Computer Communications and Networks (ICCCN '08)*, pp. 753–759, August 2008.

[28] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences*, p. 223, IEEE, January 2000.

[29] W. Stallings, *Network Security Essentials*, Prentice Hall, 3rd edition, 2007.