Source Location Privacy-Aware Data Aggregation Scheduling for Wireless Sensor Networks

Jack Kirton, Matthew Bradbury and Arshad Jhumka
Department of Computer Science
University of Warwick, Coventry, UK
{J.D.Kirton, M.Bradbury, H.A.Jhumka}@warwick.ac.uk

***Abstract*—Source location privacy (SLP) is an important property for the class of asset monitoring problems in wireless sensor networks (WSNs). SLP aims to prevent an attacker from finding a valuable asset when a WSN node is broadcasting information due to the detection of the asset. Most SLP techniques focus at the routing level, with typically high message overhead. The objective of this paper is to investigate the novel problem of developing a TDMA MAC schedule that can provide SLP. We make a number of important contributions: (i) we develop a novel formalisation of a class of eavesdropping attackers and provide novel formalisations of SLP-aware data aggregation schedules (DAS), (ii) we present a decision procedure to verify whether a DAS schedule is SLP-aware, that returns a counterexample if the schedule is not, similar to model checking, and (iii) we develop a 3-stage distributed algorithm that transforms an initial DAS algorithm into a corresponding SLP-aware schedule against a specific class of eavesdroppers. Our simulation results show that the resulting SLP-aware DAS protocol reduces the capture ratio by 50% at the expense of negligible message overhead.**

***Keywords*-TDMA; Data Aggregation Scheduling; Source Location Privacy; Wireless Sensor Networks;**

I. Introduction

Wireless sensor networks (WSNs) are collections of sensor nodes that communicate through a wireless medium. They have been used in a wide variety of contexts, from safety critical purposes such as military [1] to non-critical applications such as habitat monitoring [2]. For both applications privacy is often a key component. Privacy can be described as the guarantee that information can only be observed (or deciphered) by those that it is intended for.

Motivating Scenario Animal poaching is becoming an increasingly lucrative business. To catch the criminals, WSNs have been deployed to monitor large areas where these animals roam [3]. A recent implementation of an asset monitoring network is the Wildlife Crime Technology Report[1].

Whilst the content of this data is protected via encryption, it is also important to protect context such as a location when routing packets. Otherwise, it is possible that a poacher could backtrack through messages in the network and locate the animal-detecting node [4]. Context is a concept derived from a number of attributes relating to situational information understood from the broadcast, which can include both environmental and temporal data. This renders typical cryptographic techniques inapplicable for this type of problem. The class of context-privacy focused on in this work is called *source location privacy* (SLP).

[1] worldwildlife.org/projects/wildlife-crime-technology-project

Source Location Privacy SLP is the process of keeping the location of the source hidden from a potential attacker in an asset monitoring network. When a sensor node, called the *source*, detects the presence of an asset, it will begin to broadcast messages to be collected at a node called the *sink*. Due to the fact that the source node is broadcasting from the location of the asset, it is possible for an attacker to trace messages to this node [4].

Much of the existing research on SLP has focused on the routing layer where the various techniques alter the traffic pattern by modifying the routing protocol used to mask the source node. In this paper, we conjecture that SLP can be provided at the MAC level by achieving a similar traffic alteration through a *TDMA-based* MAC protocol. TDMA-based MAC protocols work by assigning a time slot to a node which can then transmit data in that time slot only. Thus, one given slot assignment will give rise to one traffic pattern. Then, the idea is to develop a slot assignment such that it gives rise to convergecast traffic while also providing SLP. The class of convergecast protocol we focus on in this paper is called *data aggregation scheduling* (DAS), where parents send messages after collecting data from their respective children. The principle behind such a slot assignment is to cause nodes to transmit in such a way that a path is created while also creating another path that diverts an attacker away from a source. We make the following novel contributions: (i) We formalise the novel class of distributed eavesdropping attackers which we assume in this paper. (ii) We provide a decision procedure to verify whether a given slot assignment provides SLP. Finally, (iii) we present a 3-stage distributed algorithm that generates an SLP-aware DAS. To the best of our knowledge, this is the first work that focuses on SLP at the MAC level only.

The remainder of this paper is organised as follows: In Section II we review related work. In Section III we define necessary terminology. Section IV outlines the problem specification of the paper and formalises both the DAS problem and the SLP-aware DAS problem. In Section V, we propose a distributed algorithm to generate an SLP-aware DAS. Section VI describes our experimental setup and results. Finally, Section VII presents a summary of our contributions.

II. Related Work

The SLP problem was first introduced in [4, 5] as the panda-hunter game, in which a WSN had been deployed to monitor valuable assets where the messages are routed from the node that detects the asset (*source node*) to a base station (*sink node*). In the game, there is an attacker who is attempting to

locate the asset by tracing back the wireless messages being broadcast by the source node.

Many techniques have been proposed to provide SLP against eavesdroppers [6, 7], as assumed in the paper. *Phantom routing* is a two stage protocol where the first stage sends messages on a directed random walk through the network and the second stage routes them to the sink [4, 5]. Further work has been performed to improve the random walk stage, including utilising a bloom filter [8] or location angles [9] to avoid backtracking. Another core principle is utilising *fake sources*, whereby they send padded and encrypted messages that are indistinguishable from those sent by the real source to lure the attacker in the wrong direction. Initially discounted due to suspected poor performance [4], it has since been shown that utilising fake sources can be a viable alternative to phantom routing [10, 11, 12]. Other works have also combined both principles into single protocols [13, 14].

There have been fewer techniques that use a cross-layer approach to provide SLP, e.g., [15] where beacon frames used by the MAC layer are modified to propagate messages away from the source before being convergecast. Differently from this work, the authors did not propose any new protocol.

III. Preliminaries

We provide the necessary formal background including the models we use, the program syntax and semantics and the computation and communication models.

A. System Model

Topology and processes A WSN node is a computing device associated to a unique identifier. Communication in WSNs is typically modelled with a circular communication range centered on a node, and assuming all nodes have the same communication range. With this model, a node is thought as able to exchange data with all devices within its communication range. In graph-theoretic terms, we represent a WSN as a *undirected* graph $G = (V, E)$ with a set V of vertices representing the nodes, and a set E of edges representing the communication links between pairs of nodes.

Program syntax We write programs in the guarded command notation (GCN) [16]. Hence, an action has the form $\langle name \rangle :: \langle guard \rangle \rightarrow \langle command \rangle$. In general, a *guard* is a predicate defined over the set of a process' variables. When a *guard* evaluates to true, the *command* can be executed, which takes the program from one state to another. When the state transition is complete, we say that event $\langle name \rangle$ has occurred. A *command* is a sequence of assignments and branching statements. A guard or command can contain universal or existential quantifiers of the form: $\langle quantifier \rangle \langle boundvariables \rangle : \langle range \rangle : \langle term \rangle$, where *range* and *term* are Boolean constructs. When a guard evaluates to true in a state, the corresponding action is *enabled* in that state. A special **timeout**($timer$) guard evaluates to true when a *timer* variable reaches zero. A **set**($timer, value$) command sets the timer variable to a specified value.

Communication Each process has a special *channel* variable, denoted by *ch*, modelling a FIFO queue of incoming messages sent by other processes. This variable is defined over the set of (possibly infinite) message sequences. An action with a **rcv**($sender$, msg) guard is enabled at process j when there is a message at the head of the channel variable $sender.j.ch$ of a process. Executing the corresponding action causes the message at the head of the channel to be dequeued, while msg and $sender$ are bound to the content of the message and the sender identifier.

B. Attacker Model

An attacker $\mathcal{A}$ is a distinct process that is characterised by its presence and the attacks (or actions) it can carry out [17]. In this paper, we assume the attacker to be a *distributed eavesdropper*, i.e., the attacker will be located at various positions in the network (at possibly different times, hence distributed) and will only listen to messages being transmitted (eavesdropper). Based on both attributes, the attacker process $\mathcal{A}$ will update its state.

In this paper, we propose a novel and generic model of a distributed eavesdropper, as shown in Figure 1. The attacker, called a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, is parameterised as follows: (i) R is the number of messages the attacker can receive before it makes a move, (ii) H is the number of the most recently visited locations it can record, (iii) M is the number of moves the attacker can make in a given period, (iv) s_0 is the starting point of the attacker, and (v) D is a function for the attacker that returns the set of possible new locations, based on the current data (e.g., messages heard) and historical data (e.g., previously visited locations). This parameterised attacker allow the development and understanding of attackers of various strengths.

Most work on SLP have focused on the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, where D is defined as follows: since $R = 1$, $M = 1$ and $H = 0$, it means that, when the attacker hears the first message coming from a location j, it will move to j.

IV. Problem Specifications

A. Problem Specification: Data Aggregation Scheduling

In this section, we formalise two different variants of the SLP-aware DAS problem. We will first formalise the DAS problem and subsequently define an SLP-aware DAS.

Definition 1 (Non-colliding slot): Given a network $G = (V, E)$, a node $n \in G$, and a slot i in which n can transmit its payload, we say that i is non-colliding for n iff $\forall m \in CG(n) \cdot m.slot \neq i$, with $CG(n)$ being the set of nodes in the 2-hop neighbourhood of n.

Definition 2 (Strong DAS): Given a network $G = (V, E)$, a strong TDMA-based collision-free data aggregation schedule is a sequence of sets of senders $\langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$, satisfying the following constraints:

1) $\sigma_i \cap \sigma_j = \emptyset, \forall i \neq j$,
2) $\bigcup_{i=1}^{l} \sigma_i = V \setminus \{S\}$,
3) $\forall n \in \sigma_i, 1 \leq i \leq l-1 : \forall m \in N, n \cdot m \ldots S$ is a shortest path : $(m \in \sigma_j, j > i) \vee (m = S)$,
4) $\forall m, n \in \sigma_j, 1 \leq j \leq l-1 : m \neq n \Rightarrow n \notin CG(m) \wedge m \notin CG(n)$.

The four conditions respectively captures: (i) nodes are allocated at most one time slot in a given period, (ii) every node (apart from the sink) gets at least one transmission slot, (iii) All the neighbours of a node will transmit in later slots and (iv) the schedule is collision-free.

Definition 3 (Weak DAS): Given a network $G = (V, E)$, a weak TDMA-based collision-free data aggregation schedule is

Attacker Process (R, H, M, s_0, D)-$\mathcal{A}$ where R is the number of messages an attacker can capture before making a move, H is the number of the most recently visited locations, M is the number of moves the attacker can make in a single period, s_0 is the initial location of the attacker $\mathcal{A}$, and D is a function that $\mathcal{A}$ uses to select its next location, based on its history and set of messages captured.

variables
% history of recent locations, current location
history[], curLoc : array of node ids of size H, int **init** $\langle\rangle, s_0$;

% set of messages received, number of moves made, history index
msgs, moves, num: set of msgs, int, int **init** $\emptyset, 0, 0$;

constants
period: timer; % obtained from DAS schedule denoted by α

actions
NextP:: **timeout**(*period*) $\rightarrow$ *msgs, moves* := $\emptyset, 0$;
set (*period*, α)

ARcv:: **rcv** $\langle l, m\rangle \rightarrow$
if $|msgs| < R$ **then**
msgs $\cup \{l\}$;
fi;

Decide:: *msgs* $\neq \emptyset \rightarrow$
if (*moves* $<$ M) **then**
if ($H > 0$) **then**
history[num] := *curloc*;
num := (*num* + 1) mod H;
fi;
curloc := *D*(*msgs*, *history*);
moves, msgs := *moves* + 1, $\emptyset$;
fi

Figure 1: A (R, H, M, s_0, D)-attacker

a sequence of sets of senders $\langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$, satisfying the following constraints:

1) $\sigma_i \cap \sigma_j = \emptyset, \forall i \neq j$,
2) $\bigcup_{i=1}^{l} \sigma_i = V \setminus \{S\}$,
3) $\forall n \in \sigma_i, 1 \leq i \leq l-1 : \exists m \in N, n \cdot m \ldots S$ is a path : $(m \in \sigma_j, j > i) \vee (m = S)$,
4) $\forall m, n \in \sigma_j, 1 \leq j \leq l-1 : m \neq n \Rightarrow n \notin CG(m) \wedge m \notin CG(n)$.

The main difference between the strong and weak versions of DAS is that the strong version requires a node to have all of its neighbours closer to the sink with a slot higher than its own. On the other hand, the weak version requires at least *one* such neighbour to transmit later.

B. Problem Specification: Safety Period

To capture the notion of source location privacy (SLP) awareness in TDMA-based DAS scheduling, a concept called *safety period* is required [4]. It was initially defined to estimate the number of messages needed to capture a source. The problem with this definition of safety period is that validation time is unbounded or potentially very large. As such, we use an alternative, but analogous, definition for safety period, which we estimate using the notion of capture time.

Definition 4 (Capture Time of $\mathcal{P}$): Given network $G = (V, E)$, an algorithm $\mathcal{P}$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a source $\mathcal{S}$, the capture time $\delta^G_{\mathcal{P},\mathcal{A}}$ of $\mathcal{P}$ in the presence of $\mathcal{A}$ in G is the *minimum* time taken for $\mathcal{A}$ to capture $\mathcal{S}$ in G, i.e., we say that, starting from s_0, $\mathcal{P}$ provides $\delta^G_{\mathcal{P},\mathcal{A}}$-SLP for $\mathcal{S}$ in G in the presence of $\mathcal{A}$.

To be SLP aware, an SLP-aware protocol will have a lower probability of capturing a source than a protectionless protocol over the same time period. Then, the *safety period* of $\mathcal{S}$ in G in the presence of $\mathcal{A}$ is defined as follows:

$$\delta^G_{\mathcal{S},\mathcal{A}} = C_s \times \delta^G_{\mathcal{P},\mathcal{A}} \text{ where typically } 1 < C_s < 2 \quad (1)$$

Now, if $\delta^G_{\mathcal{P}^s,\mathcal{A}} \leq \delta^G_{\mathcal{S},\mathcal{A}}$ for a given $G, \mathcal{A}, \mathcal{S}$, then the probability is 1.

C. Problem Specification: SLP-Aware DAS

In this paper, the algorithm $\mathcal{P}$ that we develop is a strong/weak DAS that is SLP aware for some specified safety period δ. To this end, we first present a decision procedure, called *VerifySchedule*, that verifies whether a given DAS schedule can lead to an attacker capturing the source, i.e., reaching the source within the safety period. The decision procedure, shown in Algorithm 1, works in a way analogous to model checking, where it returns a violating trace to show how an attacker can capture a source.

The decision procedure *VerifySchedule* works as follows: the function GENERATEALLATTACKERTRACES generates all possible traces (i.e., sequence of locations) that the attacker can take. Any such sequence $\langle s_0 s_1 \ldots s_j \rangle$ has the property that $\forall\ 0 \leq i < j, (s_i, s_{i+1}) \in E$, i.e., the attacker moves one hop at a time. The trace is then analysed by checking the validity of each step. Each step is checked to see whether this step of $\mathcal{A}$ is allowed by $\mathcal{F}$ (by computing the B variable - Line 7) and by $\mathcal{A}$'s parameters. If a step is not valid, then the trace is discarded. Else, the next step is considered until the source is reached. When the source is reached, the time taken is noted. If it is less than the safety period, there is a capture else subsequent steps are considered. The function returns a boolean value *False* if a trace of $\mathcal{A}$ can be constructed that satisfies both $\mathcal{F}$ and its parameters but ends up capturing $\mathcal{S}$.

Definition 5 (Strong (resp. weak) SLP-aware DAS): Given a network $G = (V, E)$, a DAS assignment $\mathcal{F}^s$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a given source node $\mathcal{S}$, $\mathcal{F}^s$ is a strong (resp. weak) SLP aware DAS protocol for $\mathcal{S}$ in G in the presence of $\mathcal{A}$ if and only if:

1) $\mathcal{F}^s$ is a strong (resp. weak) DAS,
2) $\delta^G_{\mathcal{F}^s,\mathcal{A}} > \delta^G_{\mathcal{F},\mathcal{A}}$, for another DAS $\mathcal{F}$.

The two conditions for a schedule to be a strongly (resp. weakly) SLP-aware DAS are:

1) The schedule has to be a strong (resp. weak) DAS,
2) The capture time of $\mathcal{F}^s$ is greater than that of $\mathcal{F}$.

Definition 6 (δ-SLP-aware for $\mathcal{S}$): Given a network $G = (V, E)$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a safety period δ, a source $\mathcal{S}$ and a DAS schedule $\mathcal{F}^s$, we say that $\mathcal{F}^s$ is *δ-SLP-aware*[2] for $\mathcal{S}$ in G in the presence of a (R, H, M, s_0, D)-$\mathcal{A}$ attacker if and only if *VerifySchedule*$(G, \mathcal{F}, \mathcal{A}, \delta, \mathcal{S})$ = $(True, \perp, \delta)$. Otherwise, if *VerifySchedule*$(G, \mathcal{F}, \mathcal{A}, \delta, \mathcal{S})$ = $(False, pc, p)$, then we say that $\mathcal{A}$ captures $\mathcal{S}$ in G under $\mathcal{F}$ within p rounds using pc.

[2]We will also say SLP-aware when δ is clear in the context.

Algorithm 1 An algorithm to verify if a TDMA slot assignment is SLP-aware.

▷ G: network topology, $\mathcal{F}$: schedule, $\mathcal{A}$: attacker
▷ δ: safety period, $\mathcal{S}$: source
▷ Function returns boolean, a (violating) sequence, and a (capture) period
function VERIFYSCHEDULE(G, $\mathcal{F}$, $\mathcal{A}$, δ, $\mathcal{S}$)
  period, num, history $\leftarrow$ 0, 0, $[0, \ldots, 0]_{1 \times \mathcal{A}.\mathrm{H}}$
  P $\leftarrow$ GENERATEALLATTACKERTRACES(G, $\mathcal{F}$, $\mathcal{A}$, $\mathcal{S}$)
  while P $\neq \emptyset$ **do**
    pc $\leftarrow$ CHOOSE(P)
    for $n_i \in$ pc **do** ▷ $pc = n_0 \cdot n_1 \ldots$
      B $\leftarrow$ 1HOPNSWITHRLOWESTSLOTS(n_i, $\mathcal{F}$, $\mathcal{A}$.R)
      if $n_{i+1} \notin \{m \mid (m, n_i) \in E\}$ ▷ Going to an unheard location or not according to $\mathcal{A}$.D
          $\wedge$ (S(n_{i+1}) $\notin$ B $\vee$ $n_{i+1} \notin \mathcal{A}$.D(B, history)) **then**
        break
      if S(n_i) > S(n_{i+1}) **then** period, moves $\leftarrow$ period + 1, 1
      else if moves = $\mathcal{A}$.M **then break**
      else moves $\leftarrow$ moves + 1
      if $n_{i+1} = \mathcal{S} \wedge$ period $\leq \delta$ **then**
        return (False, pc, period) ▷ Captured within safety period
      if $\mathcal{A}$.H > 0 **then** ▷ Only update history if the attacker is capable
        history[num], num $\leftarrow n_i$, (num + 1) mod $\mathcal{A}$.H
    P $\leftarrow$ P $\setminus$ {pc}
  return (True, $\perp$, δ)

Parameter	Symbol	Description	Value
		Protectionless DAS	
Source Period	P_{src}	The rate at which the source generates messages	5.5s
Slot Period	P_{slot}	The duration of a single slot	0.05s
Dissemination Period	P_{diss}	The duration of the dissemination period	0.5s
Number of Slots	$slots$	The number of slots that can be assigned	100
Minimum Setup Periods	$\mathcal{MSP}$	The number of periods before the source is activated	80
Neighbour Discovery Periods	$\mathcal{NDP}$	The number of periods for neighbour discovery	4
Dissemination Timeout	DT	The number of dissemination messages sent by a node	5
		SLP DAS	
Search Distance	SD	The maximum number of hops search messages make	3, 5
Change Length	CL	The length of the change path generated	$\Delta_{ss} - SD$

Table I: Parameters for protectionless and SLP DAS.

Here, the decision procedure works in a way analogous to model checking where a counterexample is returned. Similarly, pc represents the counterexample in terms of the locations the attacker can visit before capturing the source. Further, if $\mathcal{F}^s$ is δ-SLP aware for $\mathcal{S}$, then $\delta^G_{\mathcal{F}^s,\mathcal{A}} > \delta$.

V. THEORY

The 3-stage protocol works as follows: We first generate a *normal* DAS schedule. Then, we search for a suitable location in the network where the attacker can be *tricked* for some time. Finally, the *trick* is to reassign slots to some nodes to ensure that the attacker takes a longer route towards the source, thus delaying it (causing the safety period to expire). While slots are reallocated in stage 3, an important property of the distributed protocol is that the DAS property is preserved.

Figure 2 shows a protocol that generates a DAS schedule, the algorithm in Figure 3 searches for a suitable location in the network to start a redirection and the final phase is shown in Figure 4. We now explain the working of the protocols.

Phase 1: DAS schedule

The protocol for the DAS schedule (Figure 2) is started by the sink. A node keeps track of its potential parents, its distance from the sink and its allocated slot. When it receives a set of "non-empty" messages, it chooses one of the senders as its parent and also updates its slot to be less than the minimum of all slots seen. This novel aspect is to ensure that, when the slot assignment is subsequently refined, then the sender can very easily identify a new parent. This ensures the DAS property is satisfied.

On the other hand, when a node broadcasts its state, it includes the state of its neighbours. A receiving node will then potentially have information about its 2-hop neighbours. It then verifies whether it shares the same slot as one of its 2-hop neighbours. If it does, then one of the two colliding neighbours will update its slot value, guaranteeing collision-freedom.

Phase 2: Node Locator

The node locator protocol of Figure 3 searches for a node which is at a certain distance from the sink. The idea is that the distance of that node is far enough from the source.

Phase 3: Slot Refinement

The protocol of Figure 4 works as follows: when a node is selected from the node selector algorithm of Figure 3, the selected node becomes the first to trigger a change in slots of some nodes. The number of nodes to change slots is either determined by $lenD$ when nodes have at least two potential parents or until it encounters a node with only one potential parent. The selected node m chooses one of its potential parents (but not its parent) as the node to change its slot, i.e., an *upstream* node n will change its slot. For the attacker to move to n first, the slot value of n needs to be smaller than all the other nodes in m's neighbourhood. When n changes its slot, it has to inform its children to update their slots. This is achieved by setting *Normal* to 0, i.e., it is an update phase.

Now, in the DAS algorithm of Figure 2, whenever a node disseminates its state information, it specifies whether the dissemination is an update or not. If it is an update, a node checks if its slot value requires changing. If it does, then it changes its slot value to a value less than its parent (to maintain the DAS property) and also informs its children that an update is required. Up to $lenD$ nodes will start a slot update.

VI. EXPERIMENTAL SETUP AND RESULTS

A. Simulation Environment and Network Configuration

The algorithms described in Section V have been implemented using TinyOS (version 2.1.2) and are executed in TOSSIM. TOSSIM is a discrete event simulator capable of accurately modelling WSNs. Two algorithms are simulated: protectionless DAS (Figure 2) is the baseline to compare against SLP DAS.

The network layout used was a square grid with dimensions of 11×11, 15×15 and 21×21, with the top-left node being the source and the centre node the sink. The distance between each node pair was set to 4.5m, allowing only for vertical and horizontal messages transmission. An ideal communication model was used, to test the algorithms over a

$\mathcal{F}$: DAS protocol for process i

variables
myN: set of int; % set of neighbours
% set of potential parents, set of children
Npar, *children*: set of int, set of int **init** $\emptyset$, $\emptyset$;
Others[]: array of set of int **init** $\emptyset$; % Set of potential competitors

% 2-hop neighbourhood information (hop,slot)
Ninfo[]: array of (int,int) **init** $\perp$;

hop,par,slot: int,int,int **init** $\perp, \perp, \perp$; % DAS information
Normal: bool **init** 1;

% Control start of DAS protocol, timer for dissemination
start, *dissem*: bool, timer **init** 1, α;

constants
% Is the node the sink, length of redirection
sink, *length*: bool, int;
size: int **init** Δ;

actions
init:: *start* $\rightarrow$ % sink triggering the protocol.
  if (*sink*) **then**
    $\forall n \in$ *myN* **do** *Ninfo*[n] := $(\perp, \perp)$; **od**
    hop, par, slot, start := 0, $\perp$, Δ, 0;
    Ninfo[i].hop, Ninfo[i].slot := *hop*, Δ;

% Node state dissemination
dissem:: **timeout**(*dissem*) $\rightarrow$
  BCAST$\langle DISSEM, Normal, i, \{$*Ninfo[j]* $\mid j \in$ *myN*$\}$, *par*$\rangle$;
  set $\langle dissem, \alpha \rangle$;

% Normal dissem message from parent
receiveN:: **rcv**$\langle DISSEM, 1, j, N, p \rangle$ $\rightarrow$
  if (*slot* $= \perp \wedge$ *N[j].slot* $\neq \perp$) **then**
    Npar, *Others[j]* := $Npar \cup \{j\}$, *Others[j]* $\cup \{n \mid$ *N[n].slot* $= \perp\}$;
  $\forall n, N[n] \neq \perp$ **do** *Ninfo[n]* := $N[n]$; **od**

% update message from parent.
receiveU:: **rcv**$\langle DISSEM, 0, j, N, p \rangle$ $\rightarrow$
  if (*par* $= j$) **then**
    if (*slot* $\geq$ *N[j].slot*) **then**
      slot, *Normal* := *N[j].slot* $-$ 1, 0;
    $\forall n, N[n] \neq \perp$ **do** *Ninfo[n]* := $N[n]$; **od**

% finished receiving all messages
process:: **rcv**$\langle\rangle$ $\rightarrow$
  if (*slot* $= \perp$) **then**
    % Choosing parent on shortest path and its unassigned children.
    hop := min$\{h \mid (h, s) \in$ *Ninfo[k]*$, k \in$ *Npar*$\}$ + 1;
    par := min$\{k \mid (k \in$ *Npar*, *Ninfo[k]* $= (hop, s)\}$;
    slot := *Ninfo[par].slot* $-$ rank(*i, Others[par]*) $-$ 1;
    Ninfo[i] := (*hop*, *slot*);
    $\forall n \in$ *myN*, *Ninfo[n].slot* $= \perp$ **do** *children* := *children* $\cup \{n\}$; **od**

  % Detection of slot collision then resolve.
  if $(\exists j, j \neq i :$ *Ninfo[j].slot* $=$ *slot*$)$ **then**
    if (*hop* $>$ *Ninfo[j].hop* $\vee$ (*hop* $=$ *Ninfo[j].hop* $\wedge$ *i* $> j$)) **then**
      slot := *slot* - 1;

Figure 2: Phase 1 - DAS algorithm: A slot assignment protocol for data aggregation scheduling.

NSearch: Search protocol for process i

variables
% Start of search, first node of the redirection, length redirection
start, startNode, pr: bool, int, int **init** $1, 0, 0$;
from: set of int **init** $\emptyset$;

constants
% distance to travel (length of "phantom" route)
dist, *diam*: int, int **init** SD, δ;

actions
% Begin search process
startS:: *start* $\wedge$ *sink* $\rightarrow$
  $\forall c \in$ *children* **do**
    if (*Ninfo[c].slot* $=$ min$\{Ninfo[k].slot \mid k \in$ *children*$\}$) **then**
      start, *aNode* := 0, *c*;
  BCAST$\langle SEARCH, i,$ *aNode*, *dist*$\rangle$;

% Receive search message
receiveS:: **rcv**$\langle SEARCH, k, j, d \rangle$ $\rightarrow$
  from := *from* $\cup \{k\}$;
  if $(i = j)$ **then**
    if $((d = 0) \wedge (Npar \setminus \{par, k\} \neq \emptyset)$ **then**
      startNode, pr := $1, \lceil sp/3 \rceil$;
    if $((d = 0) \wedge (Npar \setminus \{par, k\} = \emptyset))$ **then**
      if (*children* $\neq \emptyset$) **then**
        aNode := choose(*children*);
      else
        aNode := choose(*myN* $\setminus \{par\}$);
      BCAST$\langle SEARCH, i,$ *aNode*, $d \rangle$;
    if $(d > 0)$ **then**
      $\forall c \in$ *children* **do**
        if (*Ninfo[c].slot* $=$ min$\{Ninfo[k].slot \mid k \in$ *children*$\}$) **then**
          aNode := *c*;
      od
      BCAST$\langle SEARCH, i,$ *aNode*, $d - 1 \rangle$;

Figure 3: Phase 2 - Node Locator: An algorithm that searches for a suitable location in the network for where redirection can occur. The algorithm inherits the variables of Algorithm in Figure 2

reliable network. The noise model being used is casino-lab. The routing algorithm used by both protectionless DAS and SLP DAS is flooding. Each node periodically broadcasts a message in its time slot.

B. Algorithm Parameters

Table I show the parameters used by the algorithms in these simulations. SLP DAS inherits all of the parameters from protectionless DAS and adds two additional parameters (search distance and change length). The search distance is the number of hops search messages travel away from the sink. The change length is the maximum length of a decoy path.

If the attacker is to capture the source, the capture time is $C =$ *period length* $\times (\Delta_{ss} + 1)$. The safety period for SLP DAS is $1.5C$, where 1.5 is the increase factor for the capture time between SLP DAS and protectionless DAS. To bound simulation time, an upper time bound of *number of nodes* $\times$ *source period* $\times$ 4 is used.

C. Attacker Model

We implemented a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker (see Section III), with the attacker knowing the period length.

D. Expected Outcomes

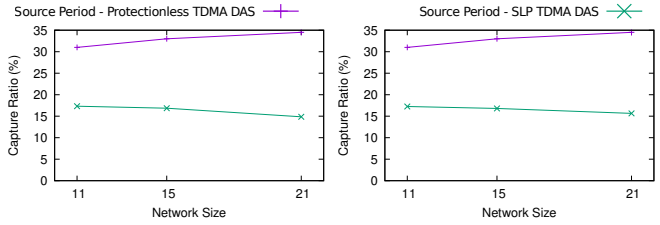
The metric we shall focus on is the capture ratio as this is the most important factor when considering an SLP algorithm. We compare the differences between protectionless DAS and SLP

SRefine: Slot refinement protocol for process i

actions
% Begin change process
startR:: *startNode* $\rightarrow$
  startNode, aNode $:= 0,$ choose$(Npar \setminus \{par\} \setminus from)$;
  nSlot $:=$ min$(\{Ninfo[j].slot | j \in myN\} \cup \{slot\})$;
  BCAST$\langle CHANGE, i, aNode, nSlot, pr - 1\rangle$;

% Receive change message
receiveC:: **rcv**$\langle CHANGE, p, j, s, d\rangle$ $\rightarrow$
  if $(d > 0 \wedge i = j)$ **then**
    if $(myN \setminus \{par\} \setminus from \neq \emptyset)$ **then**
      aNode $:=$ choose$(myN \setminus \{par\} \setminus from)$;
      slot, Ninfo[i] $:=$ $s - 1$, $(hop, slot)$;
      nSlot $:=$ min$(\{Ninfo[k].slot \mid k \in myN\} \cup \{slot\})$;
      BCAST$\langle CHANGE, i, aNode, nSlot, d - 1\rangle$;
  if $(d = 0 \wedge i = j \wedge (myN \setminus \{par\} \setminus from \neq \emptyset))$ **then**
    Normal, slot $:= 0, s - 1$;
    Ninfo[i] $:= (hop, slot)$;

Figure 4: Phase 3 - Slot refinement: An algorithm that refines the original slot assignment $\mathcal{F}$. The algorithm inherits the variables of Algorithms in Figures 2 and 3



(a) Search distance = 3 (b) Search distance = 5

Figure 5: Plots showing the capture ratio with different network sizes and search distances

DAS. Capture ratio is the ratio of runs in which the attacker manages to capture the source before the safety period ends.

E. Results

Figure 5 shows the results of capture ratio with varying values of network size. We observe that the SLP-aware DAS protocol reduces the capture ratio by 50%.

VII. Conclusion

Our main contribution in this paper is a 3-phase distributed protocol that returns an SLP-aware DAS schedule. Simulation results show that the capture ratio drops by 50% when SLP-aware DAS is used over a protectionless DAS with little overhead.

References

[1] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications : Remote large-scale environments," in *MILCOM 2009 - 2009 IEEE Military Communications Conference*, Oct 2009, pp. 1–7.

[2] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, ser. WSNA '02. New York, NY, USA: ACM, 2002, pp. 88–97.

[3] A.-M. Badescu and L. Cotofana, "A wireless sensor network to monitor and protect tigers in the wild," *Ecological Indicators*, vol. 57, pp. 447–451, 2015.

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

[5] C. Ozturk, Y. Zhang, and W. Trappe, "Source-location privacy in energy-constrained sensor network routing," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ser. SASN '04. New York, NY, USA: ACM, 2004, pp. 88–93.

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

[7] R. Rios, J. Lopez, and J. Cuellar, *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*. Cham: Springer International Publishing, 2014, ch. Location Privacy in WSNs: Solutions, Challenges, and Future Trends, pp. 244–282.

[8] Y. Xi, L. Schwiebert, and W. Shi, "Preserving source location privacy in monitoring-based wireless sensor networks," in *20th International Parallel and Distributed Processing Symposium*, Apr. 2006, pp. 1–8.

[9] W. Wei-Ping, C. Liang, and W. Jian-xin, "A source-location privacy protocol in WSN based on locational angle," in *IEEE International Conference on Communications (ICC)*, May 2008, pp. 1630–1634.

[10] A. Jhumka, M. Leeke, and S. Shrestha, "On the use of fake sources for source location privacy: Trade-offs between energy and privacy," *The Computer Journal*, vol. 54, no. 6, pp. 860–874, 2011.

[11] A. Jhumka, M. Bradbury, and M. Leeke, "Fake source-based source location privacy in wireless sensor networks," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 12, pp. 2999–3020, 2015.

[12] M. Bradbury, M. Leeke, and A. Jhumka, "A dynamic fake source algorithm for source location privacy in wireless sensor networks," in *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Aug. 2015, pp. 531–538.

[13] J. Long, M. Dong, K. Ota, and A. Liu, "Achieving source location privacy and network lifetime maximization through tree-based diversionary routing in wireless sensor networks," *IEEE Access*, vol. 2, pp. 633–651, 2014.

[14] M. Dong, K. Ota, and A. Liu, "Preserving source-location privacy through redundant fog loop for wireless sensor networks," in *13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, Liverpool, UK, Oct. 2015, pp. 1835–1842.

[15] M. Shao, W. Hu, S. Zhu, G. Cao, S. Krishnamurth, and T. La Porta, "Cross-layer enhanced source location privacy in sensor networks," in *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '09)*, 2009, pp. 1–9.

[16] E. W. Dijkstra, "Self stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, 1974.

[17] Z. Benenson, P. M. Cholewinski, and F. C. Freiling, *Wireless Sensors Networks Security*. IOS Press, 2008, ch. Vulnerabilities and Attacks in Wireless Sensor Networks, pp. 22–43.