Hybrid Online Protocols for Source Location Privacy in Wireless Sensor Networks[☆]

Matthew Bradbury[a,*], Arshad Jhumka[a], Matthew Leeke[a]

[a]*Department of Computer Science,*
University of Warwick,
Coventry, CV4 7AL, United Kingdom

Abstract

Wireless sensor networks (WSNs) will form the building blocks of many novel applications such as asset monitoring. These applications will have to guarantee that the location of the occurrence of specific events is kept private from attackers, in what is called the *source location privacy* (SLP) problem. Fake sources have been used in numerous techniques, however, the solution's efficiency is typically achieved by fine-tuning parameters at compile time. This is undesirable as WSN conditions may change. In this paper, we first present an SLP algorithm — *Dynamic* — that estimates the relevant parameters at runtime and show that it provides a high level of SLP, albeit at the expense of a high number of messages. To address this, we provide a *hybrid online algorithm* — *DynamicSPR* — that uses directed random walks for the fake sources allocation strategy to reduce energy usage. We perform simulations of the various protocols we present and our results show that *DynamicSPR* provides a similar level of SLP as when parameters are optimised at compile-time, with a lower number of messages sent.

Keywords: Wireless sensor networks, Source location privacy, Fake sources, Random walks, Online algorithm

[☆]This is an extended version of a paper [1] that was published in the Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2015.

[*]Corresponding author

Email addresses: `M.Bradbury@warwick.ac.uk` (Matthew Bradbury), `H.A.Jhumka@warwick.ac.uk` (Arshad Jhumka), `M.Leeke@warwick.ac.uk` (Matthew Leeke)

DOI: `https://doi.org/10.1016/j.jpdc.2018.01.006`

1. Introduction

Wireless sensor networks (WSNs) are expected to form the foundation of smart infrastructures of the future. As novel WSN-based applications are developed, some of these will be intended for safety-critical or security-critical domains. One such class of applications is *monitoring* applications where one or more *assets* are observed. Such critical domains may include medical services [2] while non-critical domains may include habitat monitoring [3]. For security-critical applications, these WSN-based monitoring applications have to provide several security and privacy guarantees.

This paper focuses on one such property, namely *source location privacy* (SLP), which can be described as the problem of guaranteeing that the location information of a source node (or asset) can only be observed or inferred by those intended to observe it [4]. Such assets can be, for example, military personnel, endangered species [5], etc. As WSNs operate in a broadcast medium, attackers can intercept messages and use the knowledge gained to attack the network or the asset. In SLP, the attacker may use directional antennas to be aware of the direction (a type of *context*) a message was sent from and then use that direction information repeatedly to follow through the network to find the source of the messages and thus the location of the valuable asset the source had detected.

An actual deployment was undertaken to monitor badger locations in [6] with a WSN deployed to route information and specific nodes in the network designated as badger detection nodes. A larger deployment was undertaken by the WWF as part of the Wildlife Crime Technology Report [7], where wireless mesh networks, sensors attached to animals and UAVs were deployed to monitor and protect wildlife from poachers. Wireless messages were encrypted [8] to ensure that the *content* of the messages was protected. However, these two networks did not protect against *context* attacks, meaning that they could be abused by an attacker to find the location of the animals.

Fake source techniques have been previously used to provide SLP [1, 9, 10]. In [10], a heuristic was developed — which we refer to as *Static* — that provided very high SLP levels under specific parameterisation. However, *Static* is not suitable for real-world deployment as the parameters need to be fixed at *compile-time*, making the application unable to quickly adapt to changing network conditions. In this paper, we propose a novel heuristic — which we refer to as *Dynamic* — that determines the parameter values (of *Static*) at *runtime*. Through simulations, *Dynamic* was shown to provide comparable levels of SLP as *Static*, albeit at the expense of a higher message overhead [1]. To reduce the overhead, we propose a novel hybrid SLP protocol — called *DynamicSPR* — that uses a directed random walk to select fake sources. Our results show that *DynamicSPR* maintains similar SLP levels compared to *Dynamic*, and also provides a significant reduction in the energy cost of the protection scheme.

The remainder of this paper is as follows: In Section 2 we provide a survey of related work. Section 3 details our system models and Section 4 summarises the *Static* algorithm. In Section 5 we develop the *Dynamic* algorithm for SLP and then describe using a directed random walk in Section 6. In Section 7, we outline the simulation approach employed to generate the results in Section 8 that are discussed in Section 9. Section 10 concludes with a summary of contributions.

2. Related Work

The SLP problem first appeared around 2004 in the seminal work of [11], shortly followed by [4] and there has been a large amount of contributions since then [12–14]. The authors of [4] proposed a formalisation of the SLP problem, and subsequently investigated several algorithms to enhance SLP. One technique was to allocate *fake sources*, but the authors indicated that it had poor performance despite being an expensive strategy. It has since been shown that, for certain attacker models, fake sources can provide high levels of SLP [9]. The authors subsequently went on to propose an algorithm called *phantom routing*, where messages are first sent on a directed random walk of a given length to a *phantom node*. After reaching the phantom node, the message is routed to the sink either via flooding (PRS) or by single-path routing (PSRS).

2.1. Phantom-Routing Based Techniques

There have been several extensions to the phantom routing scheme originally proposed in [4]. Most focus on improving how the directed random walk is performed. GROW (Greedy Random Walk) [15] was one of the first extensions that proposed using a bloom filter to record previously visited nodes to make better decisions about the path the directed random walk should take. Another alternative approach is angle-based techniques such as PRLA [16] or ADRS [17], which use angles between key nodes such as the sink, source and current location to determine which node should be chosen next in the path. Some techniques aim to improve privacy by simply having multiple phantom routes [18], or an increased randomness and diversity of the location of phantom nodes [19], or by sacrificing nodes along the route by turning off their radio to entrap the attacker [20]. Other algorithms (such as [15]) investigated optimisations to reduce energy usage. This class of technique is temporal [21] in nature as the attacker is typically delayed whilst on route to the source.

Similar techniques to phantom routing exist, where messages are routed in a ring around the sink rather than in directed walks [22, 23]. Rings are first formed around the sink and then generated messages are forwarded through the rings before being routed to the source. Another technique uses communications to lure an attacker to an area of the network which is then disconnected from the rest of the network to trap the attacker [20].

However, there have been many works highlighting the deficiencies of phantom routing. An issue solved by the self-adjusting random walk [24] was that there could potentially be no suitable neighbour to continue the random walk. The solution thus proposed adjusted the direction of the walk to ensure that there would be sufficient suitable neighbours. Another problem is that the performance of phantom routing degrades when multiple sources are present [25]. This problem is still an open issue. In terms of new types of attack, the random walk technique has also been shown to reveal information about the location of the source [26], where a traffic-analysis attack developed based on random walks hitting the network boundary allows prediction of the source's location. The work in [27, 28] identified three attacks based on quantitative leakage of information (i) correlation-based source identification, (ii) routing traceback, and (iii) reducing source space. It is shown that phantom routing as well as algorithms that use the same source id for each message sent are vulnerable. Due to these weaknesses, it is worth considering alternative protection schemes.

2.2. Fake-Sources Based Techniques

Instead of relying on a time delay introduced by a random walk, the fake source technique uses on a subset of network nodes which act as decoys for the real source by becoming fake sources. Fake sources will periodically broadcast fake messages that are indistinguishable from the normal messages sent by the (real) source, with the aim to convince an attacker that the fake source is actually the real source node. When the set of nodes is the whole network, maximum SLP is achieved [29]. However, this configuration uses a large amount of energy, which reduces the network's lifetime. Thus, an intelligent fake sources selection strategy is required. [30] reinforces this point by showing that there is a trade-off to be made between SLP and energy used due to message retransmissions. This class of technique is spacial [21] in nature as the attacker is lured by the fake sources to a different area of the network.

The main criticism of fake sources is that they use a large amount of energy in comparison to other techniques, so it is important to fine-tune fake source selection strategies to reduce their energy cost. Another issue is that fake source techniques can perform poorly with multiple sources due to collisions between fake messages [31]. A criticism of both fake sources and routing-based techniques is that many existing solutions focusing on just providing SLP, whereas algorithms such as [32] provide location privacy and additionally identity and route privacy.

2.3. Other Techniques

Many strategies have combined fake sources and routing protocols to provide improved SLP. The work in [33] and [34] contributed the notion of CEM and PEM respectively. CEM aims to trap the attacker in a cycle instead of letting them find the source node whereas PEM draws the attacker away using extended paths that broadcast fake messages. In [35] the authors proposed imposing a tree structure on the network using fake sources at the leaf nodes, with a focus on using the minimal energy possible at nodes one-hop from the sink node to lengthen the network's lifetime. In [36], the RFL scheme uses the idea of fogs to provide privacy using a combination of fake messages inside the fogs and routing between multiple fogs to provide SLP. Other schemes have combined phantom routing with fake sources by creating them along the phantom route [37, 38]. Whereas, other routing techniques have used delay strategically to group messages together to prevent the attacker from making as much forward progress towards the source [39]. Finally, some solutions are starting to apply techniques that were previously discounted due to the large energy and computational cost involved. For example, [40] proposed a technique based on onion routing and distributed hash tables, but did not experimentally evaluate the energy cost of their technique.

In short, phantom routing provides a high level of SLP at relatively low energy usage while fake sources can potentially offer higher SLP levels but at the cost of higher energy usage. Further, all of these techniques use network and configuration information that needs to be available at compile-time.

2.4. Other Kinds of Context Privacy

There are several other research directions relating to the provision of privacy in WSNs. Some have investigated the problem of base station-location privacy [41–

43], combining sink and source location privacy [44, 45] and providing location privacy to multiple nodes simultaneously [44] by using fake routes branching off the main route. Others have focused on more powerful attackers, such as coordinated multiple attackers [9], or on a different kind of context privacy — temporal privacy [46] — where the time events occur need to be protected.

2.5. Anonymous Communications Outside of Wireless Sensor Networks

Much work has investigated traffic analysis attacks against general networks [47]. A common way to achieve sender or target privacy in internet communications is to use a proxy which acts as a middleman. A downside is that the sender and target are both leaked to the proxy. Onion routing [48, 49] solves this problem by sending a message between multiple hops before it reaches the destination. At each hop a layer of encryption is removed from the message revealing the next hop it should target. This ensures the target does not know the source and the intermediate hops do not know the target.

There tends to be less need for onion routing in WSNs as the infrastructure is typically owned and used by a single organisation. Though this may be less true with the introduction of WSN IP stacks [50]. But, because a WSN operates in a broadcast medium, the communication between intermediate hops is not kept hidden from the attacker. This inter-hop communication will leak context information if the messages are not routed in an SLP-aware manner.

3. Models

3.1. Network Model

A wireless sensor node is a device with a unique identifier that has limited computational capabilities and is equipped with a radio transmitter for communication. A WSN is a set of wireless sensor nodes with communication links between pairs of nodes. We assume that all nodes in the network have the same communication range. The nodes that are in direct communication range with a node n are called the neighbours of n.

There exists a distinguished node in the network called a *sink*, which is responsible for collecting data and which acts as a link between the WSN and the external world. Other nodes sense the presence of an asset and then route the data via $\langle$normal$\rangle$ messages along a computed route to the sink for collection. We assume that any node can be a data source and we assume a single node to be a data source at any time. We assume that the network is event-triggered, i.e., when a node senses an object, it starts sending messages periodically to the sink for a certain amount of time.

We assume ⟨normal⟩ messages to be encrypted and that the source node includes its ID in the encrypted messages. Using the ID, the sink can infer an asset's location as we assume the network administrators will record where they put nodes. We do not assume that WSN nodes have access to GPS due to the resulting increase in energy cost.

3.2. Safety Period Model

The objective of any WSN-based SLP solution is to ensure that the asset (at a specific location) is *never* captured through the WSN. However, two issues arise: (i) if the asset is not mobile, then a trivial solution is that the attacker can take as long as it requires to perform an exhaustive search of the network, until it catches the source, and (ii) if the asset is mobile, an exhaustive search is unsuitable as the source may have moved when the attacker reaches the location. Thus, with a mobile asset, the SLP problem against a local attacker present in the network can only be considered when it is *time-bounded*, i.e, the asset has to be captured within a given time window. Such a situation may occur during wild-life poaching and in military situations.

This notion of time window has been termed as *safety period* in the literature. There are two competing definitions of safety period: (i) The one used primarily by routing-based techniques, e.g. [4] is where the safety period is defined as the time required to capture the asset. The aim of these techniques is to maximise the safety period, i.e., the higher the time to capture, the higher the SLP level provided. (ii) The second notion is the dual of the first time, in that it is specified as the *maximum* time the attacker can take to capture the source. We use the second notion here.

3.3. Routing Protocol

In WSNs, a routing protocol is required so data can be transferred from a source node to the sink node. The routing protocol is considered to be a set of paths (a path is a sequence of communication links between pairs of nodes) and a message will travel along one of the paths to the sink. Each message may follow the same path or messages may follow different paths to the sink. In this paper the technique we propose in this paper is independent of the type of routing protocol used. In the SLP problem, an attacker will make use of the routing protocol to locate the asset.

3.4. Attacker Model

It was proposed in [51] that the strength of an attacker for WSNs could be factored along two dimensions, namely *presence* and *actions*. Presence captures the network coverage of the attacker, while actions capture the attacks the attacker can launch. For example, presence could be local, distributed or global, while actions could be eavesdropping or reprogramming among others. In this sense, the attacker we assume is a *local distributed eavesdropper* (distributed due to mobility) based on the patient adversary, introduced in [4]. Such an attacker is reactive in nature and proceeds as follows:

1. The attacker initially starts at the sink.
2. When the attacker is co-located at a node n and eavesdrops a message that has not been received before, from a neighbour node m, the attacker will move to m. Thus, in a normal setting, the attacker is geared to moving closer to the source as he only follows unique messages.
3. Once the source has been found, the attacker will no longer move.

Previous work [4] has assumed that the attacker has the ability to identify whether a message has been previously responded to. We also make this assumption and implement it by having the attacker record and compare the message type and sequence number. This is similar to the attacker comparing the encrypted message body against previously received messages. If this assumption is not made, then a routing protocol such as flooding will lead the attacker away from the source. The attacker will respond to both ⟨Normal⟩ and ⟨Fake⟩ messages, as ⟨Fake⟩ messages are encrypted and padded to be indistinguishable from ⟨Normals⟩ messages.

We assume that the attacker has the capability to perfectly detect which direction a message arrived from, that it has the same radio range as the nodes in the network, and also has a large amount of memory to keep track of information such as messages that have been heard. This is commensurate with the attacker models used in [1, 10, 30].

In this work, the attacker starts at the sink because the sink is the one location in the network where the attacker is *guaranteed* to eavesdrop a message from the source node, irrespective of the routing protocol used. The attacker could potentially start at any location in the network, however, the attacker may not receive messages due to their location not being on the route from the source to the sink. We assume that the sink is located at a base known to the attacker such as a military base or a field station used by scientists monitoring wildlife.

3.4.1. Capabilities

This attacker model lies on the bottom of total order of attacker capabilities shown in Figure 1. This is because some stronger attacks would weaken the the attacker's ability to capture the source by leaking information to the WSN regarding the attacker's position. For example, if the attacker attempted to disrupt the functioning of the network (e.g., by a DoS attack), it would reduce the amount of useful information the attacker could gather. If an attacker attempted to broadcast messages to influence the SLP or routing protocol, then the WSN could potentially detect an intrusion attack and respond by ceasing to broadcast around the attacker (similar to [20]).

$$\textit{eavesdrop} \rightarrow \textit{crash} \rightarrow \textit{disturbing} \rightarrow \textit{limited passive} \rightarrow \textit{passive} \rightarrow \textit{reprogramming}$$

Figure 1: Attacker capability hierarchy proposed in [51].

Performing certain attacks such as breaking into a sensor node to obtain encryption keys (i.e., passive attacks) *are* good strategies for an attacker trying to defeat SLP. The problem with such an attack is that it is also time consuming. For example, [51, p. 11] predicts that a key stealing attack will take around 30 minutes to perform in the field (not counting preparation time elsewhere or the time it takes to find, obtain and open a sensor). As our solution to SLP aims to provide a high level of SLP within a specific safety period, if the time taken to obtain encryption keys is larger than the safety period then the attacker will have failed to capture the source within this safety period. Then the attacker would have achieved better results by simply eavesdropping.

Thus, in the context of SLP in WSNs, one of the most powerful type of local attackers that can be had is the distributed eavesdropper which we assume here.

4. Problem Statement and Overview of the Static Heuristic

After having introduced the attacker model and the concept of safety period, we now present the abstract SLP problem addressed in this paper. Given a network $G = (V, E)$, a distributed eavesdropper $\hat{A}$ that is initially located at the sink, a source $src \in V$, a safety period P_{safety}, a routing algorithm $\mathcal{R}$, the problem is to select a set $F \subseteq V$ such that $\forall v \in F$, assign a tuple (n, p, d) to v, where n is either a temporary or permanent fake source, p is the fake message period and d is the duration over v sends fake messages such that $\hat{A}$ does not reach src within P_{safety} when $\hat{A}$ is following the movement rules defined in Subsection 3.4.

Table 1: Glossary

Name	**Description**
TFS	Temporary Fake Source
PFS	Permanent Fake Source
TailFS	Tail Fake Source — A type of fake source only used in *DynamicSPR*
n, j, k	Refer to nodes in the network. Node $\hat{A}$ refers to the attacker
P_{src}	Source Period — the time between the source sending messages
D_{TFS}	TFS Duration — the time a TFS exists for
P_{TFS}	TFS Period — the time between the TFS sending ⟨fake⟩ messages
P_{PFS}	PFS Period — the time between the PFS sending ⟨fake⟩ messages
$\#_{\mathcal{F}}$	Number of ⟨fake⟩ messages to be sent during the TFS duration
α	The average time it takes for one message to be received at a neighbour
$1\textsc{HopN}(j)$	The set of nodes in the 1-hop neighbourhood of j
$\Delta_{sink}(j)$	The distance in hops between node j and the sink
$\Delta_{src}(j)$	The distance in hops between node j and the source
Δ_{ss}	The distance in hops between the sink and the source
$\mathcal{N}_i$	The i^{th} ⟨normal⟩ message sent by the source
$\mathcal{A}$	The ⟨away⟩ message sent by the sink
$\mathcal{C}$	The ⟨choose⟩ message sent by fake sources at their end-of-life
$\mathcal{F}_i$	The i^{th} ⟨fake⟩ message sent by fake sources
$S_n(\mathcal{M})$	The time one of the four message types is sent by node n
$R_n(\mathcal{M})$	The time one of the four message types is received at node n
$\Delta_{as}(\mathcal{N}_i)$	The attacker's source distance after receiving i ⟨normal⟩ messages

The fake source allocation problem (assigning the tuple (n, p, d)) is an NP-complete problem [10], so there is a need for heuristics to calculate good values that provide high levels of SLP whilst using as little energy as possible. A heuristic called *Static* was initially proposed in [10] that the novel on-line heuristics we will present improve upon. We preview *Static* first, before presenting a version of *Static*, which we call *Dynamic* that can determine parameters on-line.

Static is based on three main parameters, which capture the tradeoffs involved between SLP and energy usage.

- the temporary fake source (TFS) duration (D_{TFS})
- the temporary fake source period (P_{TFS})
- the permanent fake source (PFS) period (P_{PFS})[1]

[1]We do not require permanent fake source *duration* as a permanent fake source is considered

The *Static* algorithm works as follows:

1. The source node repeatedly sends a $\langle$normal$\rangle$ message[2] $\mathcal{N}_i$ with a time period between messages of P_{src}, beginning with $\mathcal{N}_1$.
2. When the sink receives $\mathcal{N}_1$ it waits for a short period of time (ω) then broadcasts an $\langle$away$\rangle$ message $\mathcal{A}$ that floods the network.
3. When a one-hop neighbour of the sink receives $\mathcal{A}$ it becomes a TFS.
4. A TFS broadcasts a $\langle$fake$\rangle$ message $\mathcal{F}_i$ with period P_{TFS} for a duration of D_{TFS}, before becoming a normal node and broadcasting a $\langle$choose$\rangle$ message $\mathcal{C}$.
5. When a normal node receives $\mathcal{C}$ it becomes a PFS if the node believes itself to be the furthest node in the network from the sink, otherwise it will become a TFS. A PFS broadcasts a $\langle$fake$\rangle$ message $\mathcal{F}_i$ with period P_{PFS}.

- When a node receives a previously unencountered $\mathcal{A}$, $\mathcal{N}_i$ or $\mathcal{F}_i$ it updates its last seen sequence number for that message and rebroadcasts the message.

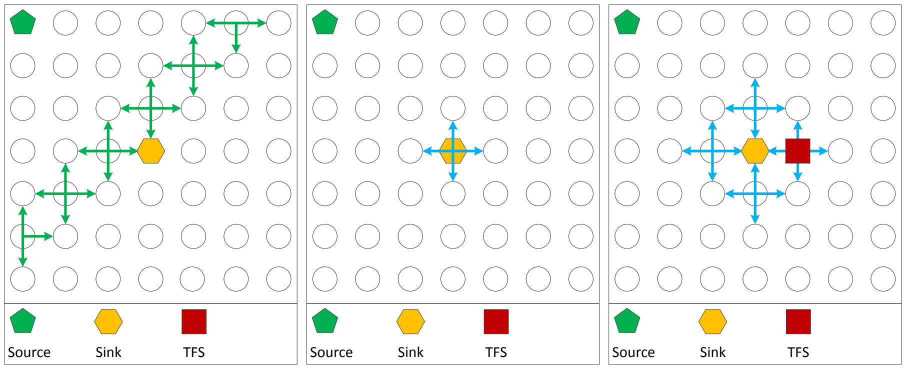
- When a node receives a previously unencountered $\mathcal{C}$ it updates its last seen sequence number for that message.

The authors of [10] performed a large-scale simulation to show the high levels of SLP achievable by *Static*, when varying the values of the main variables, i.e., D_{TFS}, P_{TFS} and P_{PFS}. The *Static* scheme has fake sources initially selected close to the sink, that slowly *move* away to positions further from the sink and the source. This allows the attacker to be slowly pulled away from the source and was been informed by one of the results of [10] which showed that a higher TFS duration improved SLP.

The problem is that if *Static* is deployed, D_{TFS}, P_{TFS}, and P_{PFS} must be fixed at compile time, making *Static* susceptible to poor performance under changing network conditions or incorrect parameters chosen at compile-time. Making these decisions correctly requires precise understanding of the environment where the network is being deployed. Thus, we now propose, in the next section, a novel heuristic, which we call *Dynamic*, that determines these parameter values at *runtime*, on a *per node* basis, obviating the need to incorporate operational knowledge at compile time.

a temporary fake source with ∞ duration.

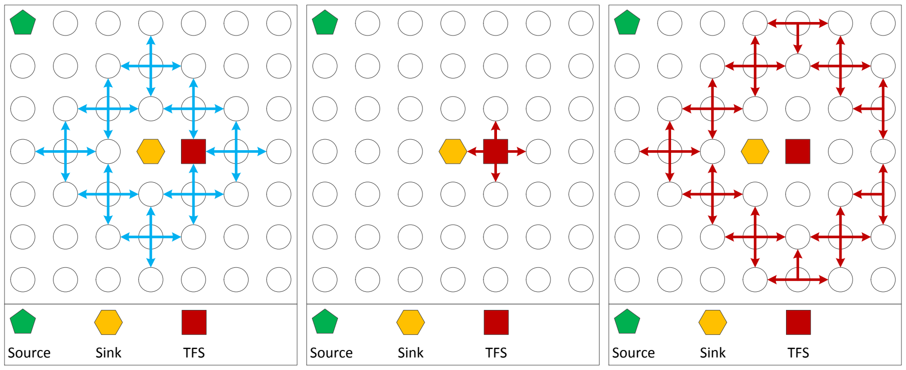
[2] In [10], the base routing protocol used was flooding.



(a) The source floods the network with ⟨normal⟩ messages, repeating every P_{src} seconds. Nodes record their Δ_{src}.

(b) After the sink receives the first ⟨normal⟩ message it waits ω seconds then floods an ⟨away⟩ message to start the fake source allocation.

(c) All nodes 1-hop from the source that receive the ⟨choose⟩ become TFSs. This diagram only shows 1 for simplicity.



(d) The ⟨away⟩ flood continues, allowing nodes to record Δ_{ss} and $\Delta_{sink}(j)$.

(e) The TFS starts sending ⟨fake⟩ messages, for the duration of the fake source.

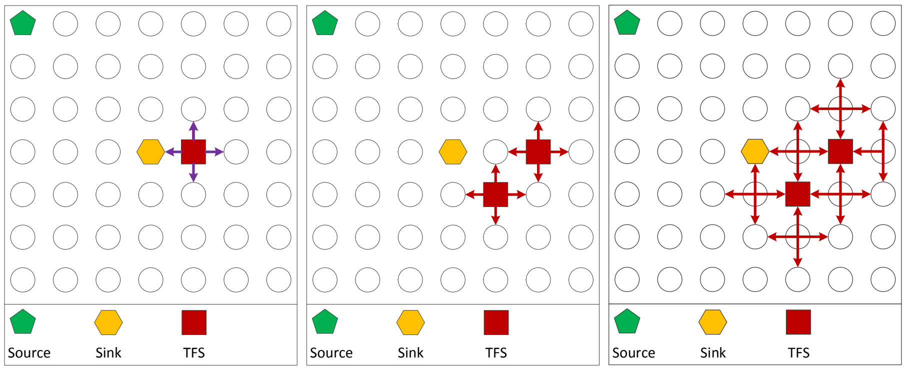
(f) The ⟨fake⟩ message flood should lure the attacker away from the source.

Figure 2: The common actions for the *Static* and *Dynamic* algorithms.

5. Dynamic: Estimating Parameters Online

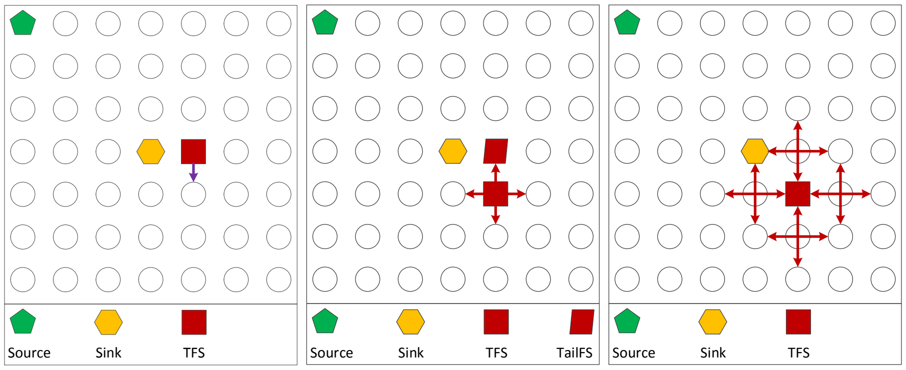
Three types of information are required for the online evaluation of D_{TFS}, P_{TFS}, and P_{PFS}: (i) parameters fixed at compile time of the firmware that are known to all nodes, (ii) information needed to derive the parameters that is not required during network execution and (iii) information that must be calculated during network execution and passed on to other nodes in the network.

The first piece of information required is the *source period* (P_{src}), which is fixed at compile time. The second required piece of information is the *delivery delay* (α), which is the time taken for a message sent by one node to be received at a neighbour. This delay has been the subject of research as it is an important



(a) Choose next fake source by broadcasting a ⟨choose⟩ message. (b) Nodes further from the source than the original TFS become fake sources. (c) The fake sources continue sending ⟨fake⟩ messages.

Figure 3: Spread of fake sources under *Static* or *Dynamic* after a TFS duration expires.



(a) Choose next fake source by unicasting a ⟨choose⟩ message. (b) Original TFS becomes a TailFS until it detects a further TFS. (c) TailFS detects further TFS and becomes normal again.

Figure 4: Spread of fake sources under *DynamicSPR* after a TFS duration expires.

value to take into account during clock synchronisation [52, 53]. Typically, α will be very small compared to the source period P_{src}, which means that its impact may be negligible on the final values. Finally, three pieces of network information are computed during execution: (i) the sink-source distance (Δ_{ss}), (ii) the sink distance for node n ($\Delta_{sink}(n)$) and (iii) the source distance for node n ($\Delta_{src}(n)$). All distances are calculated in hops. With this information, we will now explain how the three important parameters of *Static* are calculated.

To develop the framework for calculating the parameters, we view the fake source strategy of luring an attacker away from a source through an analogy with tug–of–war: In tug–of–war, two teams are pulling on either end of a rope, the team that pulls a marker on the rope over a certain point wins. In SLP, we can think of sending messages as pulling on the rope, the source is on one side and the fake sources are on the other. The attacker is the marker that will cause one team to lose, i.e., $\hat{A}$ captures the source because the *pull* from the source is greater than the pull from the fake sources. We will often refer to the *pull* of the source or the fake sources during our explanation. We will use this analogy in determining the values of parameters of *Dynamic* at runtime. To provide SLP, the fake sources need to exert a bigger pull on the attacker than the source.

For this analysis we use the following notation. The current time will be denoted by t, at $t = 0$ the source node sends the first ⟨normal⟩ message. The 1-hop neighbourhood of a node j is denoted by $1\text{HOPN}(j)$. We also define a function $\max_\perp$ which finds the maximum of all arguments that are not $\perp$.

5.1. Message Timings

We assume that the source routes messages to the sink such that the message will first reach the sink via a shortest path (e.g., by flooding or CTP [54]). Thus,

- The i^{th} ⟨normal⟩ message is sent by the source at time:

$$S_{src}(\mathcal{N}_i) = (i-1)P_{src} \tag{1}$$

- In the worst case when no SLP is provided, an attacker will receive $\mathcal{N}_1$ at $t = \alpha\Delta_{ss}$ and will have moved to be at $(\Delta_{ss} - 1)$ hops from the source. For the i^{th} ⟨normal⟩ message the attacker receives, the distance between the attacker and the source (Δ_{as}) will be:

$$\Delta_{as}(\mathcal{N}_i) = \max(0, \Delta_{ss} - i) \tag{2}$$

- An attacker will receive the i^{th} ⟨normal⟩ message at time:

$$R_{\hat{A}}(\mathcal{N}_i) = S_{src}(\mathcal{N}_i) + \alpha\Delta_{as}(\mathcal{N}_{i-1}) \tag{3}$$

⟨Away⟩ messages will be sent and received at these earliest times:

- The ⟨away⟩ message $\mathcal{A}$ is sent by the sink at time $S_{sink}(\mathcal{A})$. The wait between receiving a ⟨normal⟩ message and sending the ⟨away⟩ message is denoted by ω. This short wait was included to reduce collisions between the ⟨normal⟩ and ⟨away⟩ messages. ω was set to $\frac{P_{src}}{2}$ in this work.

$$S_{sink}(\mathcal{A}) = S_{src}(\mathcal{N}_1) + \alpha\Delta_{ss} + \omega \tag{4}$$

- A node j will receive $\mathcal{A}$ at the earliest at the time $R_j(\mathcal{A})$.

$$R_j(\mathcal{A}) = S_{sink}(\mathcal{A}) + \alpha\Delta_{sink}(j) \tag{5}$$

When node j receives an $\langle$away$\rangle$ message $\mathcal{A}$ (and its hop count is 0) or when it receives a $\langle$choose$\rangle$ message, the node becomes a TFS and starts broadcasting $\langle$fake$\rangle$ messages. The time at which j becomes a TFS is $\tau_{TFS}(j)$, where:

$$\tau_{TFS}(j) = \begin{cases} R_j(\mathcal{A}) & \text{if } j \in \text{1HopN}(sink) \\ \tau_{TFS}(k) + D_{TFS}(k) + \alpha & \text{if } k \in \text{1HopN}(j) \wedge \Delta_{sink}(k) < \Delta_{sink}(j) \end{cases} \tag{6}$$

The number of $\mathcal{N}$ messages sent between $t = 0$ and $t = \tau_{TFS}(j)$ is $\Sigma_j(\mathcal{N})$:

$$\Sigma_j(\mathcal{N}) = \left\lceil \frac{\tau_{TFS}(j)}{P_{src}} \right\rceil \tag{7}$$

5.2. Calculating TFS Duration

5.2.1. Intuition

When a node is selected as a TFS, the period at which it generates $\langle$fake$\rangle$ messages defines the pull it exerts on the attacker. Since it knows the pull exerted by the source (defined by P_{src}), we can estimate the duration over which the fake source needs to apply its greater pull to drag the attacker away from the source. Our estimation will make use of network delays encountered.

5.2.2. Derivation

To calculate D_{TFS}, we set the duration to be the difference in time between the TFS sending the first $\langle$fake$\rangle$ message and the attacker receiving the next $\langle$normal$\rangle$ message, less the time it takes to send the next $\langle$choose$\rangle$ message:

$$D_{TFS}(j) = R_{\hat{A}}(\mathcal{N}_{\Sigma_j(\mathcal{N})+1}) - \tau_{TFS}(j) - \alpha \tag{8}$$

For the case $\Delta_{sink}(j) = 1$, the attacker will have already received $\mathcal{N}_1$ and the next $\langle$normal$\rangle$ message it will receive is $\mathcal{N}_2$. The following timing information is known about the nodes 1-hop away from the sink:

$$\tau_{TFS}(j) = \alpha\Delta_{ss} + \omega + \alpha \tag{9}$$
$$R_{\hat{A}}(\mathcal{N}_1) = \alpha\Delta_{ss} \tag{10}$$
$$R_{\hat{A}}(\mathcal{N}_2) = P_{src} + \alpha(\Delta_{ss} - 1) \tag{11}$$

Using this information the duration for a node $j \in 1\textsc{HopN}(sink)$ is:

$$D_{TFS}(j) = R_{\hat{A}}(\mathcal{N}_2) - \tau_{TFS}(j) - \alpha = P_{src} - \omega - 3\alpha \tag{12}$$

The next step is to calculate the duration for nodes that are n-hops away from the sink, where $n > 1$. In this case the attacker has now received $\mathcal{N}_n$ and the duration of this TFS is to last until $\mathcal{N}_{n+1}$ is received. The knowledge about a node k that is $(n-1)$-hops from the sink, can be used to calculate when the node j that is n-hops from the sink becomes a TFS at $\tau_{TFS}(j)$.

$$\begin{aligned} \tau_{TFS}(j) &= \tau_{TFS}(k) + D_{TFS}(k) + \alpha \\ &= \tau_{TFS}(k) + (R_{\hat{A}}(\mathcal{N}_n) - \tau_{TFS}(k) - \alpha) + \alpha \\ &= R_{\hat{A}}(\mathcal{N}_n) \\ &= (n-1)P_{src} + \alpha(\Delta_{ss} - (n-1)) \end{aligned} \tag{13}$$

$$R_{\hat{A}}(\mathcal{N}_{n+1}) = nP_{src} + \alpha(\Delta_{ss} - n) \tag{14}$$

Therefore the duration is given by:

$$D_{TFS}(j) = R_{\hat{A}}(\mathcal{N}_{n+1}) - \tau_{TFS}(j) - \alpha = P_{src} - 2\alpha \tag{15}$$

For nodes where $\Delta_{sink}(j) > 1$, the duration is equal to the time that the attacker would receive the next $\langle$normal$\rangle$ message less the time it received the current $\langle$normal$\rangle$ message and α.

$$D_{TFS}(j) = R_{\hat{A}}(\mathcal{N}_{i+1}) - R_{\hat{A}}(\mathcal{N}_i) - \alpha = P_{src} - 2\alpha \tag{16}$$

So for any node j the $D_{TFS}(j)$ will be:

$$D_{TFS}(j) = \begin{cases} P_{src} - \omega - 3\alpha & \text{if } \Delta_{sink}(j) \in \{1, \perp\} \\ P_{src} - 2\alpha & \text{otherwise} \end{cases} \tag{17}$$

As α is not available to the nodes during runtime and because α is expected to be very small relative to P_{src}, α is ignored in the final result.

$$D_{TFS}(j) = \begin{cases} P_{src} - \omega & \text{if } \Delta_{sink}(j) \in \{1, \perp\} \\ P_{src} & \text{otherwise} \end{cases} \tag{18}$$

To aid in handling unreliable information dissemination, when the node's distance to the sink $\Delta_{sink}(j)$ is unknown (i.e., set to $\perp$) the smaller duration is used.

5.3. Calculating Number of Fake Messages to Send

5.3.1. Intuition

There exists a relation between the TFS duration, the TFS period and the number of message that are sent in that period. Either the number of messages to be sent can be defined in terms of the duration and period, or the period can be defined in terms of the duration and number of messages. We choose the latter, as shown in Equation (20), as this allows the algorithm to make its decisions based on the number of ⟨normal⟩ messages sent during the TFS duration.

$$\#_{\mathcal{F}}(j) = \frac{D_{TFS}(j)}{P_{TFS}(j)} \tag{19}$$

$$P_{TFS}(j) = \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)} \tag{20}$$

5.3.2. Derivation

Two approaches are provided to calculate the number of ⟨fake⟩ messages to send.

Pull From Attacker:

$$\#_{\mathcal{F}}(j) = \max_{\perp}(1, 2\Delta_{sink}(j)) \tag{21}$$

This approach aims to pull the attacker back from its estimated position assuming no SLP protection. In this case we assume that TFS nodes propagate away from the source at the same rate that an attacker moves towards the source. This can be a reasonable assumption when the duration of the TFS is equal to the source period. This means that a TFS will need to send twice the $\Delta_{sink}(j)$ to dissuade the attacker back from its position. $\Delta_{sink}(j)$ message are needed to pull back from the sink to the TFS, another $\Delta_{sink}(j)$ messages are needed to pull the attacker from its position back to the sink.

Pull From Sink:

$$\#_{\mathcal{F}}(j) = \max_{\perp}\left(1, \begin{cases} \Delta_{sink}(j) & \text{if } \perp \in \{\Delta_{src}(j), \Delta_{ss}\} \\ \Delta_{src}(j) - \Delta_{ss} & \text{otherwise} \end{cases}\right) \tag{22}$$

This approach aims to pull the attacker back from the sink's location. This approach is less aggressive compared to the previous approach and is not as focused on trying to pull the attacker all the way back, but instead keeping it in a location between the TFS and the source. An important benefit of this approach is that $\Delta_{src}(j) - \Delta_{ss}$ is used to calculate the sink distance. This means that a TFS closer to the source will send fewer messages than TFS further away.

5.4. Calculating TFS Period

5.4.1. Intuition

The TFS period of a fake source defines how fast ⟨fake⟩ messages are sent by the node, i.e., it captures the strength of the pull of the fake source. A TFS must send at least 1 ⟨fake⟩ message to keep parity with the number of ⟨normal⟩ messages sent. $c > 1$ messages need to be sent to ensure that at least one ⟨fake⟩ message reaches the attacker when collisions occur. To pull an attacker back h hops, $h \times c$ different ⟨fake⟩ messages need to be sent.

5.4.2. Derivation

Using the TFS duration $D_{TFS}(j)$ and the number of ⟨fake⟩ messages to send $\#_{\mathcal{F}}(j)$ the TFS period is obtained by dividing them. The P_{TFS} can not be allowed to go below 3α as collisions would then occur between the current and the previously broadcast ⟨fake⟩ message. Again as α is unavailable, $P_{TFS}(j)$ is finally defined without it. This requires $\#_{\mathcal{F}}(j)$ to be defined in such a way that it does not lead to $P_{TFS}(j)$ being set to 3α or less.

$$P_{TFS}(j) = \max\left(3\alpha, \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)}\right) \qquad (23)$$

$$P_{TFS}(j) = \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)} \qquad (24)$$

5.5. Calculating PFS Period

5.5.1. Intuition

By the time a PFS has been created, many TFSs should have been pulling the attacker away from the source. This means the PFS should not need to send as many ⟨fake⟩ messages as a TFS. It will need to send at least 1 ⟨fake⟩ message for each ⟨normal⟩ message sent by the source, plus some extra to consider collisions.

5.5.2. Derivation

If an attacker can be guaranteed to have been moved far enough from the source by TFSs, then having $P_{PFS}(j) = P_{src}$ would be preferable. However, the attacker's position should not be relied upon to be far enough away, meaning the algorithm requires $P_{PFS}(j) < P_{src}$, such that any PFSs retain the ability to pull back the attacker and cope with collisions of $\langle$fake$\rangle$ messages.

A lower bound on the period $P_{PFS}(j) \geq \alpha$ exists, as the PFS cannot physically send messages more often than that. There also exists an upper bound of $P_{PFS}(j) < P_{src}$ as the PFS should not broadcast slower than the source.

The technique used here is to set the PFS period to the source period multiplied by the receive ratio of $\langle$fake$\rangle$ messages at the source ($\psi_{src}(\mathcal{F})$). This is justified because it means for every $\langle$normal$\rangle$ message sent the PFS should send enough $\langle$fake$\rangle$ messages for the attacker to receive at least one $\langle$fake$\rangle$ message.

$$P_{PFS}(j) = \max(P_{src} \times \psi_{src}(\mathcal{F}), 3\alpha) \quad (25) \qquad P_{PFS}(j) = P_{src} \times \psi_{src}(\mathcal{F}) \quad (26)$$

In order to calculate this receive ratio, the source node needs to keep a record of the number of $\langle$fake$\rangle$ messages sent and received. The sequence number records the overall number sent and an additional counter records the number of times the sequence number was updated as the number received. This information must be transmitted back to the PFS, where it is needed, using $\langle$normal$\rangle$ messages. This is best effort as a PFS may not receive every $\langle$normal$\rangle$ message sent.

$$\psi_{src}(\mathcal{F}) = \frac{\text{fake\_messages\_received} + 1}{\text{fake\_sequence\_number} + 1} \tag{27}$$

5.6. Summary

These derivations allow the *Dynamic* algorithm to determine the three main parameters online and adapt to changes in the network. The diagram shown in Figure 5a depicts how node types can change during the execution of *Dynamic* and an ideal spread of fake sources for *Dynamic* is shown in Figure 6a. However, as the energy usage is high for this technique there is a desire to optimise it.

6. DynamicSPR: Fake Source Allocation Strategy

The *Dynamic* heuristic (and thus *Static*) works in such a way that fake sources spread out across part of the network. Nodes close to the source are prevented from becoming fake sources and nodes that are believed to be the furthest from the source become a PFS (see Figure 6a). The downside is that the many fake sources cause a high message overhead (i.e., use a large amount of energy) and may often undo the work of other fake sources or induce a high proportion of collisions. To circumvent this problem, we consider an alternative technique for fake source selection and use the *directed random walk* technique from *phantom routing* to achieve this[3]. The benefit of the directed random walk is that it allocates a node far from the source, which is a good location for a PFS to be.

DynamicSPR (dynamic single path routing algorithm), is a modification of the *Dynamic* heuristic that uses a directed walk away from the source node to allocate fake sources. Intuitively, the use of a directed random walk should prevent multiple competing fake sources from being created, allowing high levels of SLP to be provided whilst using fewer messages. This approach to allocating fake sources matches the optimal solution presented in [55], where the routing of messages causes the attacker to move away from the source. A difference between *Dynamic* and *DynamicSPR* is in the way the sink is notified that SLP should start being provided. Instead of starting fake source allocation when the first ⟨normal⟩ is received, instead the source node first sends a ⟨notify⟩ message to the sink to inform it that an object has been detected.

6.1. DynamicSPR *Overview*

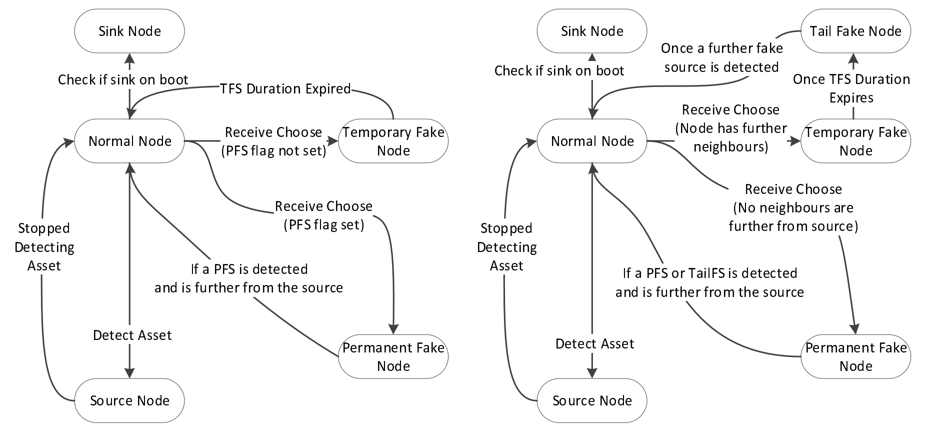
Initially a wave of ⟨away⟩ messages are sent to inform nodes of their sink distance. When the sink receives a ⟨notify⟩ from the source, a ⟨choose⟩ message will be sent to start the fake source allocation. At the end of a fake node's duration the node then chooses a neighbour that has the furthest source distance and sends them a ⟨choose⟩ message. If there are multiple candidates, the next fake node will be chosen randomly from them. Along the walk TFSs will be created to pull the attacker away hop–by–hop. The walk will end when no neighbours are further from the source than the current node, this node will be a PFS.

The state machine for *DynamicSPR* is depicted in Figure 5b. A typical execution of *DynamicSPR* will result in a fake source selection shown in Figure 6b.

6.2. Ensuring Reliability

It is unrealistic to assume that links between sensor nodes are reliable or that the links remain bi-directional [56]. Unreliable links become problematic when considering delivery of messages along single path routes, because if one message is lost it can prevent the single-path route from reaching its target. If a ⟨choose⟩ message fails to be delivered then the directed walk could terminate prematurely

[3]The routing protocol for ⟨normal⟩ messages remains unchanged for *DynamicSPR*.


Sink Node
Check if sink on boot
TFS Duration Expired
Normal Node
Receive Choose (PFS flag not set)
Temporary Fake Node
Receive Choose (PFS flag set)
Stopped Detecting Asset
If a PFS is detected and is further from the source
Detect Asset
Permanent Fake Node
Source Node
Sink Node
Tail Fake Node
Once a further fake source is detected
Check if sink on boot
Once TFS Duration Expires
Receive Choose (Node has further neighbours)
Normal Node
Temporary Fake Node
Receive Choose (No neighbours are further from source)
Stopped Detecting Asset
If a PFS or TailFS is detected and is further from the source
Detect Asset
Permanent Fake Node
Source Node

(a) *Static* and *Dynamic* (b) *DynamicSPR*

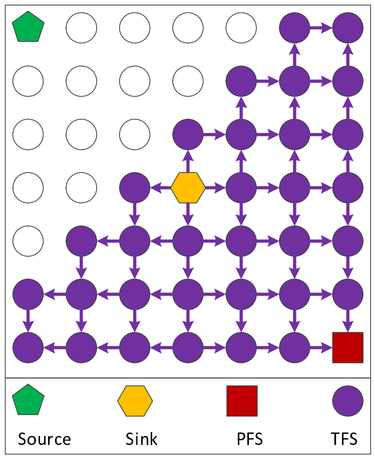
Figure 5: The conditions under which nodes transition from one type to another.

and no more SLP protection would be allocated. To ensure the walk is continued, a ⟨choose⟩ message needs to be retransmitted until it is acknowledged. However, having the fake source only sending ⟨choose⟩ messages until they are acknowledged may stop providing SLP for a time as no ⟨fake⟩ messages are flooded through the network. To resolve this problem a new type of fake source — *tail* fake sources (TailFS) — will be created along the directed walk to continue providing SLP.

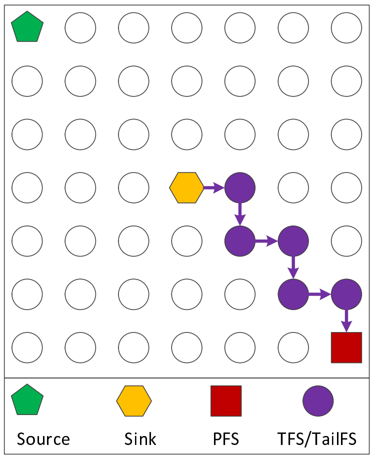
A TFS becomes a TailFS after its duration expires. While it is a TailFS, both ⟨choose⟩ and ⟨fake⟩ messages are sent periodically. As soon as a ⟨fake⟩ message is received from a fake source that is further from the real source than the TailFS is, the TailFS reverts to being a normal node and ceases sending ⟨choose⟩ and ⟨fake⟩ messages. In this case the ⟨fake⟩ message acts as an acknowledgement packet. Using this technique pairs of fake sources will be created along a walk, a TFS and a TailFS that is one-hop close to the source than the TFS is. As a TailFS has a potentially unbounded duration, the fake broadcast period is set to be those used for PFSs ($P_{PFS}(j)$). The ⟨choose⟩ message is repeated every TFS duration ($D_{TFS}(j)$) until the TailFS becomes a Normal node.

6.3. Choosing The Next Fake Source

In order to choose the next fake node from the 1-hop neighbourhood, nodes need to keep their neighbour's updated. To do this every node periodically broadcasts a message informing neighbours of their id as well as important distance information that is used to make a decision on the next fake source. The next fake node in the directed random walk is chosen randomly from the



(a) *Static* and *Dynamic*



(b) *DynamicSPR*

Figure 6: Best-case spread of fake sources for three different fake source algorithms.

set defined in Equation 28 and is allowed to have the same source distance as the current fake source. By reducing the strictness of the distance decision, the directed walk is allowed to go along a path of same-distance nodes which could potentially help the walk reach further from the source.

$$\textsc{CanBeFake}(j) = \{n \mid n \in 1\textsc{HopN}(j) \wedge \Delta_{src}(n) \geq \Delta_{src}(j)\} \tag{28}$$

6.4. Unchanged Settings and Their Impact

The way D_{TFS}, P_{TFS}, and P_{PFS} are calculated in *DynamicSPR* has not been changed from the *Dynamic* algorithm. In one sense, *DynamicSPR* can be considered a special instance of *Dynamic* where all but one link at each hop becomes unidirectional.

6.5. Number of Messages to Send

Whilst the approach to calculate the TFS period has not changed, a more conservative number of ⟨fake⟩ messages can be sent. This is because there is no longer a wave of fake sources that are potentially competing against one another.

At minimum for the first ⟨normal⟩ messages, two ⟨fake⟩ messages will need to be sent. This is because the first normal message will have at best pulled the attacker one hop from the source towards the sink, and the TFS will be allocated 1 hop from the sink in a direction away from the source. This means the TFS must first pull the attacker to the sink and then again towards the

TFS. For future ⟨normal⟩ messages at least one ⟨fake⟩ message will be required, as the ⟨normal⟩ message will pull the attacker one hop towards the source. In both of these situations noise or unreliability may cause messages to be lost, so depending on how reliable these links are extra messages may need to be sent. However, we assume a node will not know how reliable all links between itself and the attacker are, so do not have the nodes consider it.

The number of messages each TFS sends over its duration is listed below. Randomly choosing 1 or 2 messages is included to consider the case where messages are lost due to collisions.

- 1 message over the duration (called `Fixed1`)
- 2 messages over the duration (called `Fixed2`)
- Randomly chosen between 1 or 2 messages over the duration (called `Rnd`)

7. Experimental Setup

We explain the simulation tool and protocol configurations used for validation.

7.1. Simulation Environment and Network Configuration

The TOSSIM (version 2.1.2) simulation environment was was used in all experiments [57]. TOSSIM is a discrete event simulator capable of accurately modelling sensor nodes and the modes of communications between them.[4]

A square grid network layout of size $n \times n$ was used in all experiments, where $n \in \{11, 15, 21, 25\}$, i.e., networks with 121, 225, 441 and 625 nodes respectively. None of the algorithms described in this work require the network topology to be in a grid structure, as our communication model does not guarantee it. We find grids a good way to test the algorithms, hence why we have tried to structure them as such. These sizes were chosen as we believe that they are practical sizes for deployments and also because the attacker tends to capture the source more often on smaller networks [10]. Hence, we believe that testing with small or medium sized networks provide better opportunities to uncover poor SLP performance in these protocols.

[4]The source code for the implementation of the algorithm and the scripts used to test it can be found at `https://bitbucket.org/MBradbury/slp-algorithms-tinyos`.

The node neighbourhoods were generating using LinkLayerModel[5] with the parameters shown in [1, Table I]. Noise models were created using the first 2500 lines of `meyer-heavy.txt`[6]. A single source node generated messages and a single sink node collected messages. The source and sink nodes were distinct and assigned positions in the SourceCorner configuration from [10] where the sink is in the centre and the source in the top left corner. The rate at which messages from the real source were generated was varied $P_{src} \in \{2, 1, 0.5, 0.25\}$. At least 5000 repeats were performed for each combination of parameters. Separate repeats were performed individually for the three *DynamicSPR* configurations: `Fixed1`, `Fixed2` and `Rnd`. Nodes were located 4.5 meters apart, which was the distance experimentally determined to give a grid network topology. This separation distance ensured that messages (i) pass through multiple nodes from source to sink, (ii) can move only one hop at a time, and (iii) will usually (but is not guaranteed) to be passed to horizontally or vertically adjacent nodes.

7.2. Simulation Experiments

An experiment constituted a single execution of the simulation environment using a specified protocol configuration, network size, source period and safety period. An experiment terminated when the source node had been captured by an attacker or the safety period had expired. An attacker was implemented based on the log output from TOSSIM. It maintains internal state about its location using node identifiers. When a node receives a message, if the attacker is at that location it will move based on the attacker model specified in Subsection 3.4.

7.3. Safety Period

As stated in Subsection 3.2, we use a definition for safety period such that it leads to bounded simulation times. For a given network size and source rate, using flooding as a base routing protocol, we calculate the average time it takes the attacker to detect the real source (i.e., capture the asset). We call this the *capture time*. As established in Subsection 3.2, the safety period needs to be larger than the capture time, to allow for the chance that the attacker moves in the wrong direction and to allow it the chance to rectify the mistake. In this paper, the safety period value used is double the average time taken from the

[5]LinkLayerModel is a tool provided with TOSSIM that generates link strengths between nodes using experimental results.

[6]meyer-heavy.txt is a noise sample file provided with TOSSIM.

Table 2: Safety period (seconds) for each network size and source period (seconds).

$\mathbf{P_{src}}$ / **Size**	**2**	**1**	**0.5**	**0.25**
$\mathbf{11 \times 11}$	83.65	41.99	21.35	11.00
$\mathbf{15 \times 15}$	124.06	61.82	31.24	15.98
$\mathbf{21 \times 21}$	183.12	91.78	46.23	23.40
$\mathbf{25 \times 25}$	222.92	111.67	56.07	28.37

first ⟨Normal⟩ message sent until the attacker to capture the source. The safety periods used in our simulations are shown in Table 2.

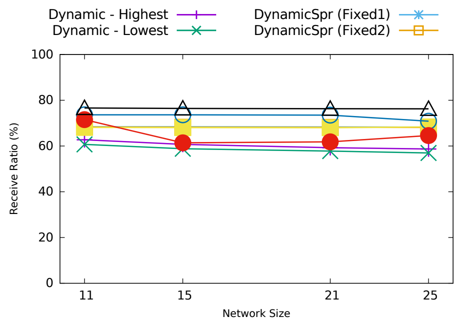
8. Results

In this section we describe the results of the *DynamicSPR* protocol via a comparison between them and the *Dynamic* SLP protocol. From our simulations we collected various metrics about the performance of the *Dynamic* and *Dynamic-SPR* protocols. The following metrics will be analysed in this section:

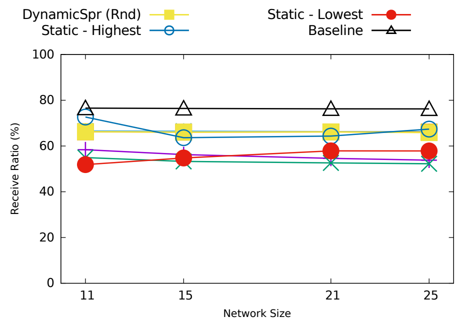
1. **Received Ratio** — This is the percentage of messages that were sent by the source and received by the sink.
2. **Capture Ratio** — This is the percentage of runs in which the attacker reaches the location of the source, i.e., captures the source.
3. **Average Number of Fake Messages Sent** — This is the average number of ⟨fake⟩ messages sent across all nodes.
4. **Attacker Distance** — This is the average attacker distance from the source recorded at the end of a run.

Several graphs include values for *Baseline* results which are the results for normal (or *protectionless*) flooding under the same conditions as the results when SLP is provided. These results are included to put the overhead of running the fake source protocols into perspective.

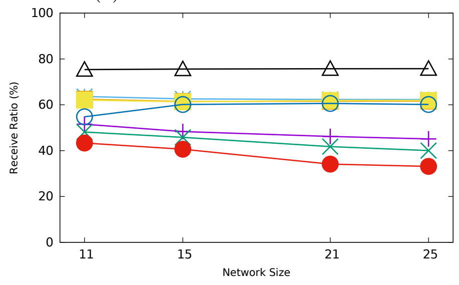
The results for *Static* are also included for some graphs to situate the performance of *Dynamic* and *DynamicSPR*. These simulation include all the parameter combinations that were used in [10]. These results are not an exhaustive search of the parameter space, but include values likely to be good. Conversely, these parameters also produced results that demonstrated poor performance.



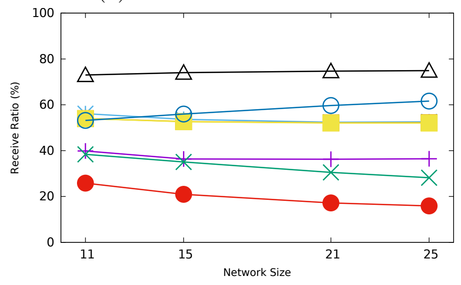
(a) Source Period 2.0 seconds



(b) Source Period 1.0 second



(c) Source Period 0.5 seconds



(d) Source Period 0.25 seconds

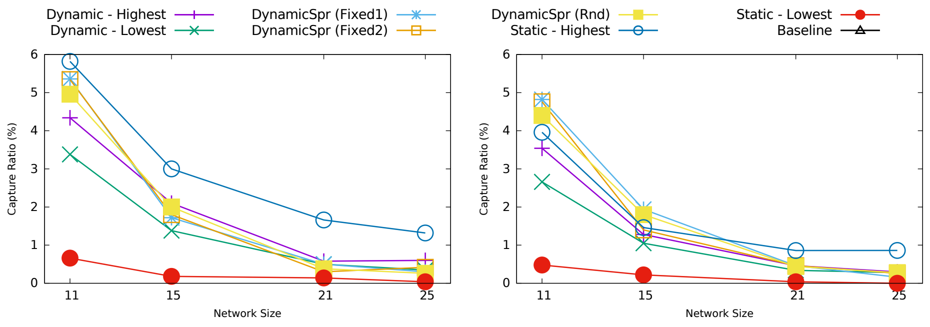
Figure 7: Results showing the percentage of ⟨normal⟩ messages received at sink.

8.1. Received Ratio

The graphs in Figure 7 show that *DynamicSPR* has a strictly better delivery ratio than *Dynamic* and has a similar delivery ratio to the best-case *Static*. This improvement over *Dynamic* is due to the lower number of ⟨fake⟩ messages being sent and thus a reduction in the number of collisions. Fewer collisions allow more ⟨normal⟩ messages to be successfully flooded from the source to the sink.

As previously experienced with the *Dynamic* protocol [1], the faster the source is broadcasting ⟨normal⟩ messages, the lower the delivery ratio is. This is, again, caused by the higher number of collisions. Also, larger network sizes tend to have lower the delivery ratios, due to the ⟨normal⟩ message requiring more successful transmissions over unreliable links leading to a lower final delivery probability. Both behaviours are also true for the *DynamicSPR* protocol.

In Figures 7c and 7d the delivery ratio becomes much lower than the baseline case, especially for the worst case *Static* and *Dynamic*. In these two cases, the low source period (leading to a high message rate) increases unreliability and causes a low delivery ratio. This then leads to a low capture ratio as the attacker has fewer opportunities to move. So for the remaining analysis, we will



(a) Source Period 2.0 second

(b) Source Period 1.0 seconds

Figure 8: Results showing the capture ratio.

focus on the two slower source periods (2 and 1 seconds between messages) as their delivery ratios remain comparable to the baseline case meaning that the SLP provided by the *DynamicSPR* and *Dynamic* protocols are due to the SLP protocols and not due to the lower delivery ratios.

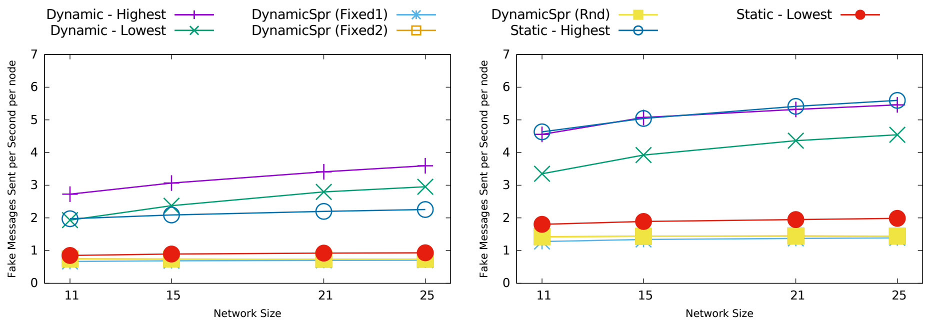
8.2. Capture Ratio

Figure 8 shows the capture ratio for the best and worst *Dynamic* and *Static* protocol results, as well as the three configurations of the *DynamicSPR* protocol. *DynamicSPR* provides a similar capture ratio compared to *Dynamic* across the four different source periods. The capture ratio for the baseline case is not visible because it is 100% for all network sizes, and has been hidden by the scale of the graph. The results for *Static* show that good parameterisation can lead to very low capture ratios. But this is traded off with very high energy usage (in terms of ⟨fake⟩ messages). *DynamicSPR* manages to achieve low capture ratios that approach *Static* for larger networks whilst incurring less energy usage as will now be shown.

8.3. Average Number of Fake Messages Sent

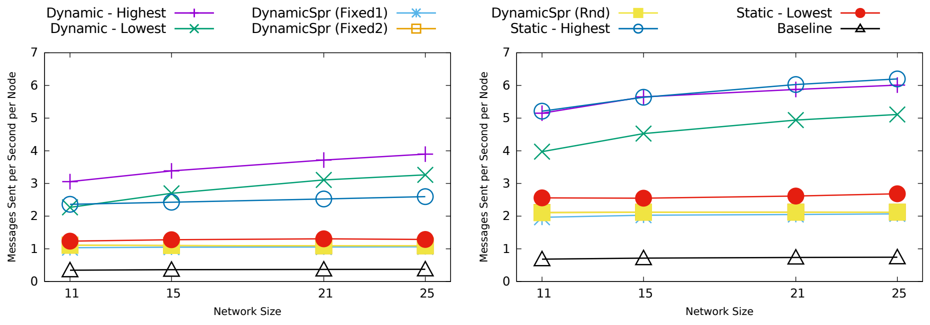
The purpose of *DynamicSPR* was to reduce the message overhead of *Dynamic*. As sending and receiving messages tends to be the most energy expensive activity performed [3], we will analyse the number of messages sent as a proxy for energy consumption.

The graphs in Figures 9 and 10 show that the *DynamicSPR* protocol performs strictly better than the *Dynamic* and *Static* protocols with respect to the number



(a) Source Period 2.0 second (b) Source Period 1.0 seconds

Figure 9: Results showing the average number of ⟨fake⟩ messages sent per node per second.



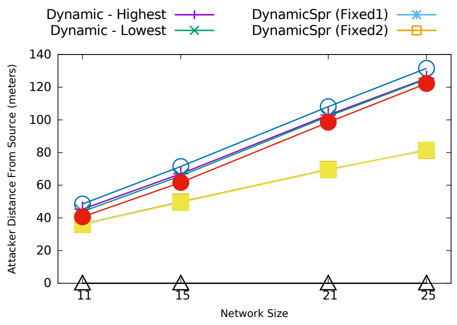
(a) Source Period 2.0 second (b) Source Period 1.0 seconds

Figure 10: Results showing the average number of messages sent per node per second.

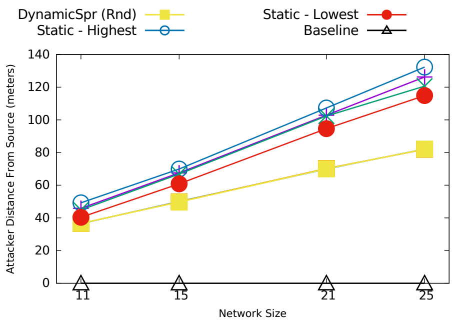
of messages sent per node per second[7]. This reduction is due to *DynamicSPR* using a reliable directed random walk compared to the *Dynamic* protocol that uses a controlled flooding approach to allocate fewer fake sources. Although, it is possible that a better parameterisation of *Static* could lead to a lower energy usage than *DynamicSPR*. Note that the lowest *Static* here does not correspond to the lowest capture ratio for *Static*, in fact low messages sent here correspond to the maximum *Static* capture ratio.

There is very little difference between the three *DynamicSPR* configurations, even though `Fixed2` had the opportunity to send twice as many messages as `Fixed1`. This behaviour is because these parameters only control the ⟨fake⟩ broadcast rate of a TFS and after a TFS's duration expires it becomes a TailFS. A TailFS will broadcast ⟨fake⟩ messages until another node receives the ⟨choose⟩

[7]We use messages sent per node per second because these techniques tend to cause each node to send a similar number of messages over the safety period.



(a) Source Period 2.0 second



(b) Source Period 1.0 seconds

Figure 11: Results showing the attacker's distance from the source.

message it sends. The rate at which these ⟨fake⟩ messages are broadcasted is decided using the PFS equations, as the duration of a TailFS is potentially infinite. A TailFS may exist for some time if the links between nodes are unreliable. As the simulated links are not perfectly reliable, the rate at which the TailFS broadcasts messages dominate the rate at which the TFS broadcast messages. Hence, preventing different TFS broadcast rates from greatly affecting the number of ⟨fake⟩ messages sent. In more reliable situations these settings may have an observable effect as the TailFS would not exist as long[8].

8.4. Attacker Distance

The attacker distance metric quantifies how good the SLP protocol is at pulling an attacker away from the source. Figure 11 shows that the *Dynamic* protocol is better able to pull the attacker further from the source compared to *DynamicSPR*. There is little difference between the different techniques used in the protocols. Both protocols perform better than baseline protectionless which failed to prevent the attacker from finding the source. The cause of the difference is that *Dynamic* is capable of allocating a PFS at a node further from the source than *DynamicSPR*. This is because *Dynamic* uses a controlled flooding of the network with TFSs which allows it to reach all nodes. In comparison, *DynamicSPR* uses a directed walk along the source distance gradient, which allows a local source distance maxima to be found, but not necessarily the global maxima.

[8]In our simulations we used `meyer-heavy.txt` which is a very noisy sample to setup the noise model which leads to higher link unreliability compared to other noise samples such as `casino-lab.txt`. We also use a LinkLayerModel that allows links to become asymmetric.

8.5. Results Summary

Overall, we conclude that *DynamicSPR* protocol provides similar SLP level to *Dynamic*, but is however more energy efficient in terms of messages sent.

9. Discussion

Improving Delivery Ratio: We observed that, at higher message rates, the delivery ratio decreases to around 30% for *DynamicSPR*. This is mainly due to the higher number of messages and the increased likelihood of occurrence of the hidden terminal problem [58], leading to more collisions. To improve significantly upon the delivery ratio, protocols proposed in [58] or other protocols such as the Collection Tree Protocol [54] can be used. Due to the different routing structure we plan on applying this framework and analysis to obtain fake broadcast rates for when CTP is used to route messages from the source to sink. Minor changes to the framework are expected, such as those made for *DynamicSPR*.
Dealing with Different Attackers: In this paper, we have assumed a distributed eavesdropping attacker that backtracks on the network traffic to capture the source within a specified time, termed as the safety period. In Subsection 3.4 we chose to assume attackers have the ability to perfectly determine whether a message is new or not. An alternative to this would be to have attackers not respond to messages for a certain time period after moving. Setting the time period of this model is a difficult decision, which is why we focused on the attacker model where the attacker has perfect knowledge regarding whether a message has been previously eavesdropped.

10. Conclusion

We present two online fake sources-based protocols, namely *Dynamic* and *DynamicSPR*, that provide SLP. These protocols are based on the *Static* heuristic, which was proposed to address the intractable nature of SLP. We have developed a framework for estimating SLP-relevant parameters. *Dynamic* was shown to provide high levels of SLP at the expense of a high message overhead. To circumvent this problem, we have proposed *DynamicSPR*, a hybrid SLP protocol, that uses a directed random walk to allocate fake sources. Our results show that, in general, the SLP levels provided by *DynamicSPR* are low like those of *Static* and *Dynamic*, whilst being more energy efficient. This makes *DynamicSPR* scalable and suitable for WSN deployments that require SLP.

Acknowledgements

This research was supported the Engineering and Physical Sciences Research Council (EPSRC) [DTG grant EP/M506679/1].

Bibliography

[1] M. Bradbury, M. Leeke, 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), 2015, pp. 531–538. doi:10.1109/Trustcom.2015.416.

[2] A. Milenković, C. Otto, E. Jovanov, Wireless sensor networks for personal health monitoring: Issues and an implementation, Comput. Commun. 29 (13-14) (2006) 2521–2533. doi:10.1016/j.comcom.2006.02.011.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA '02, ACM, New York, NY, USA, 2002, pp. 88–97. doi:10.1145/570738.570751.

[4] P. Kamat, Y. Zhang, W. Trappe, C. Ozturk, Enhancing source-location privacy in sensor network routing, in: 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), 2005, pp. 599–608. doi:10.1109/ICDCS.2005.31.

[5] A.-M. Badescu, L. Cotofana, A wireless sensor network to monitor and protect tigers in the wild, Ecological Indicators 57 (2015) 447–451. doi:10.1016/j.ecolind.2015.05.022.

[6] V. Dyo, S. A. Ellwood, D. W. Macdonald, A. Markham, N. Trigoni, R. Wohlers, C. Mascolo, B. Pásztor, S. Scellato, K. Yousef, Wildsensing: Design and deployment of a sustainable sensor network for wildlife monitoring, ACM Trans. Sen. Netw. 8 (4) (2012) 29:1–29:33. doi:10.1145/2240116.2240118.

[7] WWF, Wildlife crime technology project, Online, accessed: 2016-06-03 (2012–2016).

URL worldwildlife.org/projects/wildlife-crime-technology-project

[8] Intel, Anti-poaching technology: Wearables are helping save rhinos, Online, accessed: 2016-06-29 (2014).
URL web.archive.org/web/20160324001414/iq.intel.com/how-wearable-technology-is-helping-saving-rhinos-from-poachers

[9] A. Jhumka, M. Leeke, S. Shrestha, On the use of fake sources for source location privacy: Trade-offs between energy and privacy, The Computer Journal 54 (6) (2011) 860–874. doi:10.1093/comjnl/bxr010.

[10] A. Jhumka, M. Bradbury, M. Leeke, Fake source-based source location privacy in wireless sensor networks, Concurrency and Computation: Practice and Experience 27 (12) (2015) 2999–3020. doi:10.1002/cpe.3242.

[11] C. Ozturk, Y. Zhang, 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, SASN '04, ACM, New York, NY, USA, 2004, pp. 88–93. doi:10.1145/1029102.1029117.

[12] M. Conti, J. Willemsen, B. Crispo, Providing source location privacy in wireless sensor networks: A survey, IEEE Communications Surveys and Tutorials 15 (3) (2013) 1238–1280. doi:10.1109/SURV.2013.011413.00118.

[13] R. Rios, J. Lopez, J. Cuellar, Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures, Springer International Publishing, Cham, 2014, Ch. Location Privacy in WSNs: Solutions, Challenges, and Future Trends, pp. 244–282. doi:10.1007/978-3-319-10082-1_9.

[14] N. Li, N. Zhang, S. K. Das, B. Thuraisingham, Privacy preservation in wireless sensor networks: A state-of-the-art survey, Ad Hoc Networks 7 (8) (2009) 1501–1514, privacy and Security in Wireless Sensor and Ad Hoc Networks. doi:10.1016/j.adhoc.2009.04.009.

[15] Y. Xi, L. Schwiebert, W. Shi, Preserving source location privacy in monitoring-based wireless sensor networks, in: 20th International Parallel and Distributed Processing Symposium, 2006, pp. 1–8. doi:10.1109/IPDPS.2006.1639682.

[16] W. Wei-Ping, C. Liang, W. Jian-xin, A source-location privacy protocol in WSN based on locational angle, in: IEEE International Conference on Communications (ICC), 2008, pp. 1630–1634. doi:10.1109/ICC.2008.315.

[17] P. Spachos, D. Toumpakaris, D. Hatzinakos, Angle-based dynamic routing scheme for source location privacy in wireless sensor networks, in: Vehicular Technology Conference (VTC Spring), 2014 IEEE 79$^{\text{th}}$, 2014, pp. 1–5. doi:10.1109/VTCSpring.2014.7022833.

[18] P. Kumar, J. Singh, P. Vishnoi, M. Singh, Source location privacy using multiple-phantom nodes in WSN, in: TENCON 2015 - 2015 IEEE Region 10 Conference, 2015, pp. 1–6. doi:10.1109/TENCON.2015.7372969.

[19] J. Huang, M. Sun, S. Zhu, Y. Sun, C.-c. Xing, Q. Duan, A source-location privacy protection strategy via pseudo normal distribution-based phantom routing in WSNs, in: Proceedings of the 30$^{\text{th}}$ Annual ACM Symposium on Applied Computing, SAC '15, ACM, New York, NY, USA, 2015, pp. 688–694. doi:10.1145/2695664.2695843.

[20] A. Nassiri, M. A. Razzaque, A. H. Abdullah, Isolated adversary zone for source location privacy in wireless sensor networks, in: 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), 2016, pp. 108–113. doi:10.1109/IWCMC.2016.7577042.

[21] A. Jhumka, M. Bradbury, Deconstructing source location privacy-aware routing protocols, in: Proceedings of the Symposium on Applied Computing, SAC'17, ACM, New York, NY, USA, 2017, pp. 431–436. doi:10.1145/3019612.3019655.

[22] L. Yao, L. Kang, F. Deng, J. Deng, G. Wu, Protecting source–location privacy based on multirings in wireless sensor networks, Concurrency and Computation: Practice and Experience 27 (15) (2015) 3863–3876. doi:10.1002/cpe.3075.

[23] X. Niu, Y. Yao, C. Wei, Y. Liu, J. Liu, X. Chen, A novel source-location anonymity protocol in surveillance systems, in: 2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015, pp. 100–104. doi:10.1109/IIKI.2015.30.

[24] L. Zhang, A self-adjusting directed random walk approach for enhancing source-location privacy in sensor network routing, in: Proceedings of the

2006 International Conference on Wireless Communications and Mobile Computing, IWCMC '06, ACM, New York, NY, USA, 2006, pp. 33–38. doi:10.1145/1143549.1143558.

[25] C. Gu, M. Bradbury, A. Jhumka, M. Leeke, Assessing the performance of phantom routing on source location privacy in wireless sensor networks, in: 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC), 2015, pp. 99–108. doi:10.1109/PRDC.2015.9.

[26] R. Shi, M. Goswami, J. Gao, X. Gu, Is random walk truly memoryless — traffic analysis and source location privacy under random walks, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 3021–3029. doi:10.1109/INFCOM.2013.6567114.

[27] Y. Li, J. Ren, J. Wu, Quantitative measurement and design of source-location privacy schemes for wireless sensor networks, Parallel and Distributed Systems, IEEE Transactions on 23 (7) (2012) 1302–1311. doi:10.1109/TPDS.2011.260.

[28] L. Lightfoot, Y. Li, J. Ren, Star: design and quantitative measurement of source-location privacy for wireless sensor networks, Security and Communication Networks 9 (3) (2016) 220–228. doi:10.1002/sec.527.

[29] K. Mehta, D. Liu, M. Wright, Protecting location privacy in sensor networks against a global eavesdropper, IEEE Trans. on Mobile Computing 11 (2) (2012) 320–336. doi:10.1109/TMC.2011.32.

[30] A. Thomason, M. Leeke, M. Bradbury, A. Jhumka, Evaluating the impact of broadcast rates and collisions on fake source protocols for source location privacy, in: 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2013, pp. 667–674. doi:10.1109/TrustCom.2013.81.

[31] J. F. Laikin, M. Bradbury, C. Gu, M. Leeke, Towards fake sources for source location privacy in wireless sensor networks with multiple sources, in: 15th IEEE International Conference on Communication Systems (ICCS'16), 2016, pp. 1–6. doi:10.1109/ICCS.2016.7833572.

[32] R. A. Shaikh, H. Jameel, B. J. D'Auriol, H. Lee, S. Lee, Y.-J. Song, Achieving network level privacy in wireless sensor networks, Sensors 10 (3) (2010) 1447–1472. doi:10.3390/s100301447.

[33] Y. Ouyang, Z. Le, G. Chen, J. Ford, F. Makedon, Entrapping adversaries for source protection in sensor networks, in: International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006, 2006, pp. 10–34. doi:10.1109/WOWMOM.2006.40.

[34] W. Tan, K. Xu, D. Wang, An anti-tracking source-location privacy protection protocol in WSNs based on path extension, Internet of Things Journal, IEEE 1 (5) (2014) 461–471. doi:10.1109/JIOT.2014.2346813.

[35] J. Long, M. Dong, K. Ota, A. Liu, Achieving source location privacy and network lifetime maximization through tree-based diversionary routing in wireless sensor networks, IEEE Access 2 (2014) 633–651. doi:10.1109/ACCESS.2014.2332817.

[36] M. Dong, K. Ota, 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, 2015, pp. 1835–1842. doi:10.1109/CIT/IUCC/DASC/PICOM.2015.274.

[37] P. K. Roy, J. P. Singh, P. Kumar, M. Singh, Source location privacy using fake source and phantom routing (fsapr) technique in wireless sensor networks, Procedia Computer Science 57 (2015) 936–941, 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015). doi:10.1016/j.procs.2015.07.486.

[38] L. Bai, L. Li, S. Qian, S. Zhang, Random selection false source-based algorithm for protecting source-location privacy in WSNs, in: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016, pp. 2064–2069. doi:10.1109/FSKD.2016.7603499.

[39] M. Bradbury, A. Jhumka, A near-optimal source location privacy scheme for wireless sensor networks, in: 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2017, pp. 409–416. doi:10.1109/Trustcom/BigDataSE/ICESS.2017.265.

[40] P. Palmieri, Preserving Context Privacy in Distributed Hash Table Wireless Sensor Networks, Springer International Publishing, Cham, 2016, pp. 436–444. doi:10.1007/978-3-319-29814-6_37.

[41] J. Deng, R. Han, S. Mishra, Countermeasures against traffic analysis attacks in wireless sensor networks, in: First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005, 2005, pp. 113–126. doi:10.1109/SECURECOMM.2005.16.

[42] N. Baroutis, M. Younis, Using fake sinks and deceptive relays to boost base-station anonymity in wireless sensor network, in: Local Computer Networks (LCN), 2015 IEEE 40th Conference on, 2015, pp. 109–116. doi:10.1109/LCN.2015.7366289.

[43] G. Chai, M. Xu, W. Xu, Z. Lin, Enhancing sink-location privacy in wireless sensor networks through k-anonymity, International Journal of Distributed Sensor Networks 8 (4) (2012) 1–16. doi:10.1155/2012/648058.

[44] H. Chen, W. Lou, On protecting end-to-end location privacy against local eavesdropper in wireless sensor networks, Pervasive and Mobile Computing 16, Part A (2015) 36–50. doi:10.1016/j.pmcj.2014.01.006.

[45] H. Park, S. Song, B. Y. Choi, C. T. Huang, Passages: Preserving anonymity of sources and sinks against global eavesdroppers, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 210–214. doi:10.1109/INFCOM.2013.6566765.

[46] P. Kamat, W. Xu, W. Trappe, Y. Zhang, Temporal privacy in wireless sensor networks: Theory and practice, ACM Trans. Sen. Netw. 5 (4) (2009) 28:1–28:24. doi:10.1145/1614379.1614380.

[47] X. Fu, B. Graham, R. Bettati, W. Zhao, On countermeasures to traffic analysis attacks, in: IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003., 2003, pp. 188–195. doi:10.1109/SMCSIA.2003.1232420.

[48] D. Goldschlag, M. Reed, P. Syverson, Onion routing, Communications of the ACM 42 (2) (1999) 39–41. doi:10.1145/293411.293443.

[49] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, W. Jia, A new cell-counting-based attack against tor, IEEE/ACM Transactions on Networking 20 (4) (2012) 1245–1261. doi:10.1109/TNET.2011.2178036.

[50] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, J.-P. Vasseur, M. Durvy, A. Terzis, A. Dunkels, D. Culler, Industry: Beyond interoperability: Pushing the performance of sensor network IP stacks, in: Proceedings of the 9th ACM

Conference on Embedded Networked Sensor Systems, SenSys ’11, ACM, New York, NY, USA, 2011, pp. 1–11. doi:10.1145/2070942.2070944.

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

[52] K. Liu, Q. Ma, H. Liu, Z. Cao, Y. Liu, End-to-end delay measurement in wireless sensor networks without synchronization, in: Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on, 2013, pp. 583–591. doi:10.1109/MASS.2013.71.

[53] Y.-C. Wu, Q. Chaudhari, E. Serpedin, Clock synchronization of wireless sensor networks, Signal Processing Magazine, IEEE 28 (1) (2011) 124–138. doi:10.1109/MSP.2010.938757.

[54] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, P. Levis, Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks, ACM Trans. Sen. Netw. 10 (1) (2013) 16:1–16:49. doi:10.1145/2529988.

[55] M. Bradbury, A. Jhumka, Understanding source location privacy protocols in sensor networks via perturbation of time series, in: INFOCOM, 2017 Proceedings IEEE, 2017, pp. 1611–1619. doi:10.1109/INFOCOM.2017.8057122.

[56] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, M. Alves, Radio link quality estimation in wireless sensor networks: A survey, ACM Trans. Sen. Netw. 8 (4) (2012) 34:1–34:33. doi:10.1145/2240116.2240123.

[57] P. Levis, N. Lee, M. Welsh, D. Culler, Tossim: accurate and scalable simulation of entire tinyos applications, in: Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys ’03, ACM, New York, NY, USA, 2003, pp. 126–137. doi:10.1145/958491.958506.

[58] H. Zhang, A. Arora, Y. ri Choi, M. G. Gouda, Reliable bursty convergecast in wireless sensor networks, Computer Communications 30 (13) (2007) 2560–2576. doi:10.1016/j.comcom.2007.05.046.