A Dynamic Fake Source Algorithm for Source Location
Privacy in Wireless Sensor Networks

Matthew Bradbury, Matthew Leeke, and Arshad Jhumka

Department of Computer Science,
University of Warwick, Coventry
United Kingdom, CV4 7AL
{bradbury, matt, arshad}@dcs.warwick.ac.uk

Friday 21st August 2015
Contents

1 Definitions
2 Related Work
3 Problem Statement
4 Derivations
5 Results
6 Conclusions
What is a Wireless Sensor Network?

A wireless sensor network (WSN) is a collection of computing devices called nodes, they have:

- a short range wireless radio
- an array of sensors such as light, heat and humidity
- a simple low powered CPU
- a battery with limited power supply

Applications include:

- Tracking
- Monitoring
What is Context Privacy?

Privacy threats can be classified as either content-based or **context-based**

- Content-based threats have been widely addressed (using cryptography) (Perrig et al. [8])
- Context-based threats are varied
- We focus on protecting the location context of broadcasting nodes
Important Considerations

- Wireless Sensor Nodes are energy constrained
- Sending messages is the most expensive task
- Receiving messages is the next most expensive task (Shnayder et al. [9])
The Problem of Source Location Privacy

Given:

- A WSN that detects valuable assets
- A node broadcasting information about an asset

Found:

- An attacker can find the source node by backtracking the messages sent through the network.
- So by deploying a network to monitor a valuable asset, a way has been provided for it to be captured.

The Problem:

- Panda-Hunter Game
- Difficult
Related Work

- Attacker Models (Benenson et al. [1])
- Phandom Routing (Kamat et al. [4])
- Fake Sources: TFS/PFS (Jhumka et al. [3]), CEM (Ouyang et al. [7])
- Combination: Tree-based (Long et al. [5])
- Global Attacker: (Mehta et al. [6])
Attacker Model

• We consider a single mobile *distributed eavesdropping* attacker

**Relevant System Assumptions:**

• Messages sent by a source are encrypted and include node ID
• Only the sink can tell a node’s location from the ID

**Assumptions:**

• The attacker can tell the direction a message came from
• The attacker can move at any speed and has no power limitations
• The attacker knows of (i) sink location, (ii) network topology, and (iii) routing algorithm
• Attacker can tell if a message has been seen before by using the sequence number and channel in the message header
A set of nodes will act as fake sources to lure the attacker away from the source’s location.

What parameters to use?

- **Rate** - What should it be? It will need to be faster than the source rate.
- **Duration** - How long should a node be a fake source?
- **Probability** - Should a node always become a fake source?

How does a node decide to become a fake source?
Three main parameters:

1. Temporary Fake Source Duration \((D_{TFS})\)
2. Temporary Fake Source Period \((P_{TFS})\)
3. Permanent Fake Source Period \((P_{PFS})\)

- Previously they have been fixed at compile time (Jhumka et al. [3]).
- A simulation of a deployment would thus be needed to find good settings.
- What if we could determine these parameters on-line.

The following is just a summary of the techniques, full derivation is available in the paper.
Dynamic Fake Source Allocation Heuristic

1. The source node sends a \( \langle \text{normal} \rangle \) message \( N_1 \) with period \( P_{src} \), beginning with \( N_1 \).
2. When the sink receives \( N_1 \) it waits \( \frac{P_{src}}{2} \) then broadcasts an \( \langle \text{away} \rangle \) message \( A \) that floods the network.
3. When a one-hop neighbour of the sink receives \( A \) it becomes a TFS.
4. A TFS broadcasts a \( \langle \text{fake} \rangle \) message \( F_i \) with period \( P_{TFS} \) for a duration of \( D_{TFS} \), before becoming a normal node and broadcasting a \( \langle \text{choose} \rangle \) message \( C \).
5. When a normal node receives \( C \) 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 \( \langle \text{fake} \rangle \) message \( F_i \) with period \( P_{PFS} \).
When a node receives a previously unencountered $N_i$ or $A$ or $F_i$ it updates its last seen sequence number for that message and rebroadcasts the message.

When a node receives a previously unencountered $C$ it updates its last seen sequence number for that message.
Calculating the Temporary Fake Source Duration ($D_{TFS}$)

Aim to pull an attacker back before it next gets a chance to move again.

- Set the duration to be the difference in time been 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.
- $D_{TFS}$ is usually the source period.
- This scheme should see the attacker pulled to the same source distance as the TFS.
Calculating the Temporary Fake Source Period ($P_{TFS}$)

- The $P_{TFS}$ period is the time between two sequential ⟨fake⟩ messages being sent.
- Set the period to be the ratio between the TFS duration and the number of messages that needs to be sent.
Calculating the Number of Fake Messages To Send ($P_{TFS}$)

Need to consider:
- Each message sent will not always pull the attacker away from the source.
- More messages need to be sent than the distance an attacker needs to be pulled back.
- The WSN does not know where the attacker is.
- It has to estimate its position.

When calculating, consider two approaches:
- The TFS sends messages equal to twice the sink distance (the estimated attacker distance if no protection was in use)
- The TFS sends messages equal to the source distance minus the sink-source distance
Calculating the Number of Fake Messages To Send ($P_{TFS}$)

Distances:
- Sink-Source: 6 hops
- TFS-Sink: 3 hops
- TFS-Attacker: 6 hops
- TFS-Source: 9 hops

TFS Messages Sent:
- $2 \times$ TFS-Sink: $2 \times 3 = 6$
- TFS-Source − Sink-Source: $9 - 6 = 3$
Calculating the Number of Fake Messages To Send ($P_{TFS}$)

Distances:
- Sink-Source: 6 hops
- TFS-Sink: 3 hops (same)
- TFS-Attacker: 4 hops (closer)
- TFS-Source: 7 hops (closer)

TFS Messages Sent:
- $2 \times$ TFS-Sink: $2 \times 3 = 6$
- TFS-Source − Sink-Source: $7 - 6 = 1$
Calculating the Number of Fake Messages To Send ($P_{TFS}$)

**IV**

Distances:
- Sink-Source: 6 hops
- TFS-Sink: 3 hops \textit{(same)}
- TFS-Attacker: 2 hops \textit{(closer)}
- TFS-Source: 3 hops \textit{(closer)}

TFS Messages Sent:
- $2 \times$ TFS-Sink: $2 \times 3 = 6$
- TFS-Source $-$ Sink-Source: $3 - 6 = -3$
Calculating the Permanent Fake Source Period ($P_{PFS}$)

- PFS has infinite duration, so techniques used for TFS will not work.
- We set the PFS period to be equal to the source period times the delivery ratio of fake messages at the source.
- This delivery ratio is reported from the source to the PFSs via normal messages.
- *Intuition*: At least one fake message should reach the sink during the source period.
Experimental Setup: Simulation Environment

- TOSSIM
- meyer-heavy noise model
- Links have low probability of asymmetry
Experimental Setup: Network Configuration I

- Square grid network of $11^2$, $15^2$, $21^2$ and $25^2$ nodes
- Nodes 4.5 meters apart – chosen experimentally
- Sink node positioned at the center of the grid
- Source node position in one of the four corners or at a random location along the network edge
- Attacker starts at the location of the sink
- 300 Repeats
Experimental Setup: Network Configuration II

- Source Node
- Sink Node
- Furthest
Experimental Setup: Safety Period

- Calculated for each network size and source rate
- Defined as twice the amount of time it took an attacker to capture the source when no protection was in place
- Twice the time allows an attacker to go to the opposite end of the network and back
- A bounded safety period allows bounded simulation time
Results

**Figure: Source Period 1.0 seconds**

**Figure: Source Period 0.125 seconds**
Summary

- Previous work had parameters unchangeable during runtime.
- This prevented the algorithms from being easily deployable.
- Dynamically determining settings on-line had a minor decrease of performance in terms of capture ratio, but offers the benefit of being able to respond to changing network conditions.
Future Work

Handle further changes in the network:
- Changes to the source’s locations
- Node crashes

Also:
- Energy optimisations by minimising number of TFSs and PFSs
The End

Any Questions?


