

Intrusion Detection of a Wireless Sensor Network Over the Air Update Protocol

A S M Ashraful Alam* and David Eyers*

*Department of Computer Science, University of Otago, New Zealand, Email: {aalam,dme}@cs.otago.ac.nz

Abstract—Wireless Sensor Networks (WSN) form interconnections autonomously. They can be used to work together to sense and monitor some phenomena in a unified fashion. However, the software that runs on the sensors might require updating because of changes in requirements, routines, protocols and so on. It is not easy to round up all the deployed sensors in order to reprogramme them. Hence, the capability to reprogramme over the air (OTA) was developed and several specific protocols have been implemented that achieve this purpose. Each protocol has its own suitability depending on the context and scenario. Deluge is one such protocol. The ease of reprogramming opens a backdoor to a potential intruder who can abuse the protocol by injecting a malicious program. The design of Deluge does not cater for the intruder's activity; as a result, it is not possible to detect if the update was a legitimate one or not. The method we propose analyses the timing of update of each of the sensors in the network and computes a 'Intrusion Warning Score' through an algorithmic function to indicate a possible intrusion i.e. a probable illegitimate update.

I. INTRODUCTION

Wireless Sensor Networks (WSN) have become practical within the last two decades. The technology has attracted great attention in recent years due to significant advances in wireless and mobile communication technologies. One of the main driving forces behind the quick evolution of WSN is **its cost-worthiness and flexibility in deployment**. The sensors, called the nodes or motes, are generally self-powered, wireless, networked sensing devices with inbuilt limited processing hardware. The manager node, which is also called a sink or base station generally has long-lasting power and probably, remain connected to a workstation with greater processing capability. Motes and sinks together dynamically form an autonomous networked environment where all sensor nodes within range and sharing the same radio channel and protocols take part. In such a conventional WSN, data exchange takes place only locally using an autonomous relay mechanism that does not require any routing table to be able to do so. A WSN is ultimately employed for sensing and/or detecting certain physical phenomena, on which basic processing is performed at the individual mote, the processed information is relayed back to a computing device through the sink for further processing. WSNs are generally deployed for long-term monitoring at scales and resolutions that are difficult to manage. They may necessitate reprogramming after deployment to accommodate alteration in the environment, sensing applications, or scientific requirements [1].

Advances in semiconductors and embedded processors initiated the development of sensors with computational capability. Conventionally, sensors were built using **micro-electro mechanical systems (MEMS)** or CMOS or LED technology,

based on the purpose and property to sense. Today, nanoparticles are also used for tiny sized sensors. It is now possible to combine a wireless radio transceiver with a microprocessor, memory, and other micro-electro-mechanical sensors into a tiny device. These tiny sensors have advantage of being energy friendly and are deployed in environments with limited power and space. The WSN evolved from the distributed sensor network programme of the DARPA project and is still evolving with newer and smarter ideas such as integrated RFID-enabled WSN with inkjet-printed electronics on paper substrates, distributed WSN to tackle reliability, in-network processing to reduce communication cost [2]. There are other evolving areas like contention and schedule-based energy efficient medium access schemes, cognitive routing methodologies, cross-layer protocols, combining medium access control with routing functionality, security-conscious WSN, deployment of WSN within aircraft to replace wired infrastructure and so on. Today, WSN is developed around an IEEE standard that specifies lower network layers (i.e., MAC layer and physical layer) of a wireless network which focuses on low-cost, low-speed ubiquitous communication between devices [3].

WSNs operate unattended over long period of time that may range from months to years. There is often very little control over the sensors once they are deployed. Changes in the environment and purpose over time affects the operations of the sensors. This necessitates reprogramming of the sensors to suit the evolving requirements and also to enable bug fixes after deployment. The potential scale and types of embedded types of deployments of sensors may make it impossible to collect each sensor, reprogramme it and then redeploy it again. Network code propagation using data dissemination protocols promises a suitable solution to this problem.

The technology has huge potential for future commercial applications. The technology is still evolving with greater promises of being integrated into even larger scale deployment in the form of Internet of Things (IoT) or dusts. It is expected to play a critical role in the computing revolution of cooperating objects and smart embedded devices. However, industry adoption is hampered owing to several factors like difficulty in programming, security issues, integration with other systems and business processes and so on. Historically, security remains as a neglected part in any implementation process, until there is a major breach. One of the main reason why sensor networks has not been widely deployed in industrial systems and safety systems reliably is concerns over security. Reliability and safety critics and academics have continually expressed that the security issues in WSN have not been sufficiently addressed and it is not yet secure enough for mission critical employment. Security is a very important part of system reliability. Before WSN matures to a point that can

be reliably integrated in everyday life, security issues must be adequately addressed.

WSNs have special vulnerabilities that do not exist in wired networks. Security mechanism designed for conventional computing systems often are not appropriate for WSNs. Even the security techniques like the WEP standards for wireless networks cannot be employed in WSN because of memory and computing resource constraints. WSNs are more vulnerable to intrusion attempts and security threats from malicious sources. The limited resources in terms of memory, computational power and energy, make them easier target. The attack targets may range from being on physical sensors, wireless channels, visibility of the network, even to coordination and self-configuration of the network.

Several approaches to address security issues in WSN have been proposed in recent years. They approaches converge on security to ensure reliable data transfer using simple and flexible protocols with high level of confidentiality. Because of the resource constrained nature of the motes, security is a vulnerable area to explore. Lack of suitable computation capability hinders deploying heavy cryptographic measures, wireless nature exposes itself to the adversaries who can easily overcome the barriers to listen to all the traffic and even inject their own packets into the network. When deployed in a hostile environment, the attackers may have physical or logical access or proximity access capability. Threats to security can be directed to individual motes, towards the network traffic or a cluster of motes. A potential adversary may be able to interfere with goals related to security services like authentication, confidentiality, integrity, non-repudiation or even extend to clandestine takeover. In the event of failure from the expected security services i.e. breach of cryptographic measures, an intrusion detection system (IDS) becomes invaluable.

Intrusion is the ability to disrupt or degrade the intended functionality of the network or a part of it; it could be non-intentional or not. An IDS is able to identify malicious break in attempts to a trusted environment. It is directed towards a security mechanism that can detect abnormal behaviour in the network. A suitable IDS can detect attempt of intentional or unintentional break in events from an insider or outsider party. However, an IDS does not prevent the break in, but alerts following some predefined procedures in case of a security breach. Research in WSN security with respect to IDS has lots of potentials for new discovery and implementation. An IDS architectures for static WSNs has been suggested that watches over the communications in the neighbourhood [4]. Such an IDS is not capable to identify a malicious update that has been effected by an adversary in the WSN. We propose a novel intrusion detection algorithm that just requires a minor modification of the OTA update protocol by enabling an Active Message (AM). AM is a single-hop, unreliable packet used for network abstraction in TinyOS. It has a destination address, a type field, provides synchronous acknowledgements [5]. AM enables a centralised view of a network-wide knowledge that is processed and analysed to work out a score function according to an algorithmic formula to raise a red flag which could indicate a possible intrusion.

This work's contributions are two fold. 1) The IDS identifies the anomaly in software update pattern in terms of quantitative score. The higher score indicates a higher probability

of intrusion at some point in the WSN. The score can be deterministically classified when some additional information is available. 2) The scheme can provide useful insight for designing secure WSN i.e. the IDS expectations may be utilised to

We have studied how the different algorithm parameters

The rest of this paper is structured in the following way. In Section II, we discuss the security and intrusion primitives that are talked about in the WSN arena. We try to find out what security aspects have not been addressed by the present security researchers and the reasons behind any security deficiencies. In Section III, we explain the main concepts of Deluge and its special security measures that have been implemented on Deluge. In Section IV, we have explained the experiment methodology and its implementation details. We then present our evaluation of the experiment in Section VI. A few difficulties that we ran into throughout the development of this experiment are also commented in this section. Finally, we conclude our arguments in Section VII.

II. RELATED WORK / LITERATURE REVIEW

Security is the prime concern in the path to socialising the WSN technology. Due to its applicability in wide variety of purposes, keeping WSNs secure is a challenging task. WSNs do not have any session or presentation layer. Security techniques designed for MANETs are not readily applicable in WSN for its constraints like small processing capability, low battery life, small memory. various possible attacks on WSNs has been explained by [6], [7]. On the other hand, WSN security design and analysis suggests many techniques to address them. Many security schemes for WSN have been proposed targeting two main approaches—using cryptography and using IDS. Cryptographic approach actively aims to protect privacy, ensure authenticity, data integrity and confidentiality through use of cryptographic keys and algorithms, policies and procedures. While symmetric key cryptographic techniques are more efficient in required resources, they suffer from a shortcoming that the key must be shared for being useful. However, asymmetric key cryptography is free from this limitation with a greater consumption of resources. Anomaly detection, on the other hand, is a passive security measure that monitors the behaviour from a observer's point of view and aims to alert a suitable entity for further actions.

Symmetric key cryptographic techniques are useful when both the secret key and the algorithm remains unknown, which can be difficult due to the fact that the WSN is expected to be deployed in exposed environment. Yet, most security schemes in WSN use the technique due to ease of hardware implementation, small computation, power and memory requirements. [8] recommends Skipjack, MISTY1, and Rijndael to be the most suitable block ciphers for WSNs, based on the combination of required memory, energy efficiency and intended security strength as the benchmark parameters. In terms of operation mode, pairwise links should use Output Feedback and broadcasts should prefer Coded Block Chain (CBC) mode. [9] suggested use of available transceiver chip as resource for hardware based AES encryption in the sensor as it outperforms software implementations of some of the most popular cipher algorithms suitable for WSN. [10] concludes

that the lightweight block cipher proposed by [11] achieves good performance, providing energy efficiency as well as suitable security for sensor nodes in a WSN in comparison to AES, Skipjack, Puffin, Salsa, Sosemanuk, RC4. However, cryptanalysis of the technique by [12] shows that it requires large number of rounds to make it significantly resistant against attacks. [13] found AES to be the most energy-efficient and RC5 to be better than RC6. He also commented that very few security research have been directed towards cryptographic modes of operation in WSN, which might offer interesting insight for certain modes that allow decryption using a decryption block, hence may significantly speed up the execution.

Asymmetric key cryptography tends to be resource intensive, but has the advantage of private key operation. It is very suitable in key exchange, digital signature and likewise. There are various public key cryptographic algorithms like Rabin's Scheme, RSA, Ntru-encrypt, Elliptic curve cryptography (ECC), Pairing Based Cryptography (PBC) and so on; few of them has been implemented in software and hardware for WSN [14], [15], [16]. Suitably customised implementations can result in a reduced protocol overhead, less packet transmission and power savings. ECC and PBC based techniques provide better trade-off between key size and security strength than RSA; thus is suggested to be suitable for WSN with advantages like lower computation time, storage space, transmitted data, required energy [17], [18]. However, asymmetric key cryptography cannot be used as general purpose scheme, rather to be used for authenticated session establishment, authentication, signature generation and verification, secret key exchange.

IEEE 802.15.4 specification does not require any security implementation. Yet, it describes the usage of the AES for either encryption (CTR mode) or authentication (CBC mode) or both (CCM-mode). Transport Layer Security (TLS) relies on the connection-oriented TCP protocol as transport mechanism. TLS implementations have been provided in TinyOS; however, Contiki OS still lacks the support. **NEED TO CHECK Contiki ASSERTION** TinyOS implemented Skipjack algorithm with the CBC mode in TinySec, which has been discarded later on, and support for hardware based AES-128 encryption has been provided using CC2420 chip. [19] discovered a (slightly) faster than brute force way to break the AES.

There are security issues with use of cryptography in WSN due to the nature of the hardware [20]. Encryption significantly increases energy and decreases packet throughput. In many cases, authentication is more desirable than encryption; but encryption is typically the standard method for authentication. Absence of session and presentation layers complicates the design security protocols. Moreover, WSNs do not have gateways or switches that can enable monitoring of the information flow. Hence, in case of failure from cryptographic measures, intrusions need to be detected before or after attackers have succeeded in breaching the WSN.

Usually an IDS detects suspicious behaviour in the WSN. IDS may also provide some of the information like identification and location of the intruder, extent of intrusion and so on. Butun, Morgera, and Sankar authoritatively discussed the constraints and challenges in WSN IDS design [21]. IDSs in WSN have been suggested using distributed and cooperative architecture. Some other IDS also exist that uses a hierarchical and centralised approach. Anomaly based approach works in

a distributed manner using support vector machine that minimizes communication overhead while performing in-network anomaly detection proposed by [22]. A standalone rule based approach to detect anomalies at all layers of a network stack in WSN has been suggested [23]. Rule based techniques rely on monitoring of group of nodes and routing tables for detection [24]–[27]. Spontaneous watchdog uses the sensor broadcast, density of deployed sensors for its detection [28]. Specification based IDS has been proposed that relies on data clustering and uses standard deviation from the average inter-cluster distance [25]–[27], [29]. Statistical based anomaly detection looks for anomalies in routing pattern [30]. Statistical methods that use hidden Markov model focus on the accuracy of the gathered data, rather than the security of the nodes or the links. [31]. Such a model monitors data arrival process in real time traffic on the nodes, analyses short term dynamic statistics using a multi-level sliding window event storage scheme that works on each node. The detection algorithms performs locally and individual nodes are not aware of the network-wide attacks [32]. Some other statistical based procedure exploits the stability of the neighbourhood information of the WSN nodes depending on average receive power and average packet arrival rate [33]. Game theory based IDS utilise Markov chain for its decision making process can monitor only one of the clusters of the network at a time, which leaves the rest of the network unprotected [34], [35], [36]. Reputation based IDS relies on heuristic ranking algorithms to identify most likely bad nodes in the network. Such IDSs use high scalable cluster-based hierarchical trust management protocol to effectively identifying the selfish and malicious nodes [37].

Software update in a WSN is a vulnerable process from security point of view. While cryptography is an algorithmic tool designed to verify authenticity and ensure security, IDSs monitor anomaly in network behaviour and raises alarm. There are other IDSs that initiate countermeasure process also, in addition to monitoring and raising alarm. Cryptography and monitoring both work in collaboration with each other and are intended to ensuring security and protect against intrusion. Despite these efforts, WSN passes through a vulnerable phase during the software update process due to the fact that the running operating system and application needs to switch from old version to the new ones and there is a significant increase in traffic owing to software update process. There are high number of transmitted packets in the network that uses huge network bandwidth, thereby depletes sensor energy. When a software update in WSN takes place, it replaces the executing embedded operating system and running application as a package. Cryptographic measures and IDS running in the local sensors are deactivated or unavailable, which open up a vulnerable window in time when it is easy for an intruder due to the transition. Therefore, this is a special case from security point of view and must be dealt with specific measures like an additional layer of security measures that must observe the update behaviour.

Several network data dissemination protocols have been implemented in different sensor/embedded systems operating systems. All the OTA software update protocols are targeted at attaining efficiency, throughput and conserve energy. The OTA reprogramming systems, in general, do not provide cryptographic protection for source authentication, integrity verification and likewise. Limitations of various techniques that

can be used in WSN reprogramming has been elaborated in [38]. The program binary authentication technique using a hash is not very useful in resource-constrained sensors. Deluge has been enhanced to handle security that we have elaborately discussed in section III. Deluge uses digital signatures for codes signing and authenticated broadcast scheme. It also incorporates digitally signed initial advertisement, which in turn authenticates the following messages in a chained fashion much like the CBC methods used in public key cryptography. However, we contend that it is possible for a malicious intruder to replicate an authentic image source and advertise new programme version which the nodes cannot repudiate. Even though, Deluge incorporates on-chip hardware based AES-128 encryption, Cryptography cannot completely prevent attacks. Constrained computational capacity and resources make it possible to compromise the cryptographic measures incorporated in the implemented WSN operating systems. The other vulnerability of such epidemic protocol is breach at any one point does the job of weakening the whole network for the attacker. It is possible for him to appear somewhere in the network, pretend to be an authentic source and disappear after successful transfer of the complete image to only one of the nodes in the network. The security mechanisms provided by Deluge would be unable to detect the initial breach in such cases and would allow the network to be flooded by a totally new version of the application that performs completely different function [39]. While the reprogramming protocols are essential for the WSNs, they could act as carriers of rapid spread of malicious code in the network. Hence, we propose an analytical tool that can have valuable insight into the propagation characteristics of OTA reprogramming protocol. The tool should be flexible for comparative studies of different protocols.

There is no previous works available that targets to secure update process using an IDS. The security countermeasures available in WSN systems is predominately cryptographic. None of the well-known WSN operating systems have ever incorporated an IDS due to the reality that such an employment would cut down on the resources, especially power. An IDS residing on the sensors is ideally need to be avoided by design.

Hence, we propose an IDS that can primarily be applied on OTA protocol. The technique has been tested successfully on Deluge. We have exploited services from other existing standard protocols to gather timing information outside the WSN and then process the timing information for intrusion detection. Use of timing analysis approach is a new technique in IDS for WSN. The technique may further be extended for use with broad range of intrusion detection motives specially in WSN and also in MANET. The technique is centrally controlled, resides outside the WSN, gathers processed information from the sensors in WSN. However, an IDS with that broad spectrum is out of scope of this paper. We shall limit ourselves within its application on WSN, specifically on Deluge only. Before one understands how the IDS works and is able to focus on its mechanism, one needs some prior knowledge about Deluge and its security techniques.

ADDITIONAL NOTES TO BE INCORPORATED or DISCARDED

Most of the solutions are context specific and are vulnerable depending on its application. The assumptions made on

the corresponding threats may be inconsistent with the problem domain and attempts to address unrealistic problems.

WSN have limitations in computing resources, memory and physical security. Even after stringent security implementation, it cannot overcome security threats and possible security weakness. Sensor nodes use wireless communication, which is particularly easy to eavesdrop on. Similarly, an attacker can easily inject malicious messages into the wireless network.

[40] suggests a distributed approach in which nodes do not have a global view, monitor their neighbourhood and collaborate with their nearest neighbours for their normal operational condition, yet can detect an intrusion against blackhole and selective forwarding attacks. [41] proposes a mechanism based on trust that uses node localisation, density of the WSN, collaborative sensing, cryptographic techniques and focuses on misbehaviour characterisation and behaviour monitoring to build a 'Reputation Table' for each node. However, the mechanism have assumptions about motives of the adversary, symmetric links, nodes are not compromised or faulty at the time of bootstrapping.

[42] proposes a guaranteed interleaved hop-by-hop authentication schemes for detection of injected false data immediately when no more than t nodes are compromised, where t is a system design parameter.

III. DELUGE

A. Popular OTA Protocols

Reprogramming of the sensors in a WSN is one of the most important management task. Performing the task manually may work out for a small no of sensors, but this is not a scalable idea. Hence automatic OTA reprogramming of sensors becomes an important issue. While some of the protocols designed for OTA reprogramming deals with a subset of nodes, most others necessarily update the whole network at one go resulting into all nodes in the WSN running same version of the software. The subset reprogramming allows the same WSN to preferentially perform different tasks, which is a convenience and again increases management overhead. In Trickle, which is a well-known OTA update protocol, the nodes periodically advertise their own running code version to the neighbours. When a neighbour listens to an older version, it disseminates the updated code. Trickle addresses single packet dissemination and uses suppression and dynamic adjustment of the broadcast rate to limit transmissions among neighboring nodes. The Trickle algorithm controls the rate of transfer by suppressing a nodes own broadcast instead of flooding the network [43]. Another OTA update protocol, Multihop Network Programming (MNP) is an enhancement over Trickle or Deluge in a way that is more efficient in image transfer using 'pipelining' to enable faster data propagation and also in terms of energy. MNP handles the problems that occur from message collision and hidden terminal using a greedy approach of sender selection algorithm to guarantee that at a time, there is at most one programme transmitting source in a neighbourhood. MNP [44]. While the above mentioned reprogramming protocols are either unsuitable or inefficient in a mobile environment, ReMo does not make any assumptions on the location of nodes and suggests an energy efficient, multihop reprogramming protocol for mobile

WSNs. ReMo is based on a periodic metadata broadcast paradigm whose probability is dynamically adjusted based on neighbour density and pages are transferred out of order based on availability which in turn depends on high potential of code availability, not link quality alone [45]. Comparison of code update completion time of different protocols indicates that while Deluge performs moderately, MNP is the fastest among them. Typhoon also reliably delivers large objects to all nodes using a combination of spatially-tuned timers, prompt retransmissions, and frequency diversity and has the benefits of reduced dissemination time and energy consumption by up to three times [46]. MOAP-SW allows a subset of the network to be programmed efficiently [47].

B. Deluge Protocol

Deluge is an efficient and reliable protocol for disseminating large data objects, such as program binaries that are larger than sensor memory [48]. Combining the mechanism with bootloader allows programming the wireless nodes over the air. Deluge has been implemented in TinyOS and later ported on Contiki, both are prominent WSN open source operating systems. The original protocol was implemented at Barkley by Jonathan Hui. The later version Deluge 2.x was implemented at Johns Hopkins notably by Razvan, who is also credited with secure design of the protocol and author of another OTA protocol Typhoon. Deluge, (which is) an improvement over Trickle, divides the program image into pages that are further split into packets while in transition. Instead of nodes autonomously advertising its own version like Trickle, the nodes periodically advertise the newest pages (only the ones changed from prior version) available to send. Other nodes then request the required pages following a sequential order [48].

Deluge solves several problems in network programming. It deals with larger program image or data objects, mostly much larger than the memory (RAM) available in a sensor. It works on an environment where node density is variable to the extent that includes asymmetric links as well. It displays reliable behaviour by correcting packets in environment that is characterised by high packet loss rate. Deluge ensures that all the nodes remain updated with the latest software version. In cases when a node missed several updates and needs to update its state to the very newest version, it can do so without requiring to go through the missed updates. The protocol ensures quick propagation of program codes and requires minimum service interruption.

Deluge displays few prominent characteristics that make it prominent. It transfers complete binary image when it is necessary to do so. It disseminates a program, image from one or more sources to many nodes over multi-hop WSN. The density aware epidemic property achieves reliability in the unpredictable wireless medium. The epidemic protocol works on a state machine principle and follows few local rules that ultimately achieves global behaviour. Deluge allows the sensor to store multiple program images in its flash memory, one of them is identified as golden image which is able to run with minimal support in case of a failure. Deluge can work in conjunction with a isolated bootloader to switch from running image to a newer one. The protocol achieves high performance due to its three-phase handshaking technique, robust support

for asymmetric links and density aware mechanism that suppresses redundant advertise messages and requests.

To represent large data objects, Deluge splits in into fixed size pages. Pages are basic data transfer objects which are further split into fixed no of packets for wireless transmission. Both pages and packets use 16-bit CRC for integrity and data correctness.

The deluge mechanism is simple **yet interesting**. Each node periodically advertises its version **no** and summary of software/data object. Version no is used to differentiate between updates and is chronologically increased for each subsequent updates. Nodes in the neighbourhood can determine from this advertisement if any of its neighbours have an old profile. Nodes listening to such advertisements suppresses their own advertisement if it has the same version of software; however, if it has a newer version then responds with its object profile. This is done in a fashion that can be termed as controlled flood. The nodes with older data object then determines which portion of data it needs updating and requests it from any neighbour it can reach. It works through a three-phase handshaking (advertise-request-updates) protocol which is helpful in a lossy environment to achieve reliability. Node with an old profile requests for the new page and switches to new *receive* mode. However, the neighbouring nodes which require the update can take advantage by receiving the same page without entering into a contention for requesting. The nodes utilise pipelining to increase throughput by working on completed pages before all pages in the image are complete.

A node in *receive* actively requests remaining packets required to complete a page. After making a request, the node waits for response. However, a node need not be in *receive* mode in order to receive data packets. Packets needed to complete a certain page are saved without any regard to its state. A node in *transmission* broadcasts all requested packets from a given page in round-robin order with an aim of achieving fairness.

C. Security of Deluge

The original Deluge protocol did not consider security of operations. Security of Deluge was mainly suggested by [38] in the form of solution to following problems:

- Authentication of program binary ensures that the transferred program image is from its author and has not been forged. This is usually done by signing and verifying the data stream.
- Verification of data integrity ensures that the binary was not changed during transit and if there was a change then it is detected by the recipient.
- System freshness ensures that older programs cannot overwrite the newer ones.

Since size of the entire program, page and packets are known before a transmission takes place, the program can be split into messages, each message contains a cryptographic hash of the next message and digitally sign the head of hash message. It suggested to use a particular implementation of RSA signature and SHA-1 hashes. However, the RSA signature technique has not been implemented, instead AES-128 based symmetric key has been chosen, as we have mention beforehand. Use of hash

function that binds later packets is a kind of digital signature by the base-station, and facilitates authentication [38].

D. Security Vulnerability of Deluge

Deluge security measures are not conclusive to this assertion that it can completely resent security threats. An adversary may employ the suppression to prevent the update propagation, waste network resources, disrupt the normal operation of code dissemination. He may even be able to inject malicious codes to pollute the network and deny intended function.

Advertisements are not authenticated, any node in the WSNs may advertise any message. So an adversary can launch several attacks on Deluge by injecting bogus or modified advertisement packets

IV. METHODS

The proposed IDS and its techniques were evaluated in comprehensive simulations. While the system has been elaborately designed and aimed at testbed evaluation, it has not yet been performed at this stage of the research. The testbed implementation will be evaluated at some point in the future. The simulation environment possessed the ability to observe the software update pattern in a fully functional WSN where each node generates and reports the time at which the new software update completes and starts executing. The raw data generated from simulation logs were processed using a semi-automated scheme that processed the raw data using a bash script. This semi-processed data were further processed with a programme written in cpp to make them suitable for simple mathematical calculations. Then these data were related and analysed using a spreadsheet programme to produce the IWS.

Simulation Environment

The IDS requires deluge protocol to be able to report the time of executing a new version of its software. The related techniques and design issues have been elaborately described in Section-IV. Deluge protocol was originally implemented in TinyOS. The protocol has been ported to other embedded operating systems including Contiki. We used deluge protocol implemented in 'Contiki: The Open Source OS for the Internet of Things' for our experiments. We installed Contiki v2.7 on a system running on Ubuntu 12.04. 'Cooja', a network simulator for Contiki, was used to simulate the software update pattern associated with Deluge protocol. We also used a plugin for Contiki called Mobility to aid our experiential setup. Although the plugin was designed to deal with positions over time for the nodes in the simulation, we used it to load the exact location of the sensor motes into the 'Cooja' simulator.

Design of Experiments

The simulations were carried out as information gathering exercises. They were carefully designed and planned with a view to evaluating the proposed IDS in different settings. Experiments on various topologies described in the following subsection were carried out for completeness of the exercise.

Evaluating the IDS on each of the topologies consisted of several test runs. The test runs were of two kind: 1) Test runs aimed as establishing baseline data consisted of 20

individual simulations which were initiated from the sink; and 2) Simulations that replicated an intrusion at different points of the WSN were initiated once from each of the nodes except the sink. Results obtained from all these simulations were utilised to model the expected pattern for a particular topology.

Topologies

The effects of the proposed IDS were tested in different topologies, which ranges from being very simple to complex ones, were tested for thoroughness. The topologies were designed basing on different design considerations that were established using number of trial and error test runs. We used several python scripts to place the motes according to specific requirements like shape, distance, range and power level.

The shape of the experiential topologies were designed as linear, circular, elliptical, double rings, wide line, grid, tree and other random topologies. The names of the topologies here indicate that in each case the motes were deployed along the circumference of the geometrical shape that enabled packet propagation to take place according to a designed fashion. In each topology, minimum 20 nodes were deployed for a test to take place. More motes were necessary to replicate certain topologies. Table-I shows the number of motes that were used in experiments in corresponding topological deployments. The motes were carefully placed at a distance of 30—35 meter along the circumference of the intended topological shape. This distance was found necessary to allow multiple transmission paths to all the motes along the desired directions. This was further augmented with other constraints based on topological specification. For most experiments, the radio transmission power level was set at 100% which has an approximate maximum effective transmission range of 50 meter and is able to interfere other motes upto 100 meter away. The Owheo WSN topology was additionally tested at 50% power level to identify the effects of power level on the IDS

Procedure

Experiment procedure was a combination of several well defined activities that included running simulation, analysing system log and processing data to arrive at a conclusion.

Processing of Individual Test Run: The first step in the experiment was to run the simulation in the environmental set up described above according to the specific design requirement. At the very beginning of the simulation, all the motes in the WSN were booted up at same time running one version of a diagnostic application. The simulated motes performs network discovery process immediately on boot up and forms the autonomous WSN. The diagnostic application would necessarily report its time, version number and mote ID on the serial terminal every second at its serial port. Cooja simulator allows to print this outputs from the serial ports of all motes in the 'Mote output' window.

The simulation allowed the diagnostic application in the base station to dynamically begin the update process using deluge protocol and propagated another version of the operating system software, which is also called software image. The new version of the image was transmitted to all motes in a fashion as determined by the deluge and other associated protocols. Whenever transfer of the new image was complete

Topologies	Linear	Circular	Elliptical	Double Rings	Wide Line	Grid	Tree	Owheo WSN
Number of Motes	20	20	20	46	45	72	25	37

TABLE I: Number of motes used in each topological deployment

for a mote, the new image would replace its old version of the application and would reboot to begin executing the new one. The new application would also necessarily allow the motes to print out their time, version number and mote ID on the serial terminal every second. First time a mote declared its running version number to be of the new image was the update time for that mote. We continued running the simulator until all the motes in the WSN were updated to the new version software image. All information in the ‘Mote output’ window, which were in reality outputs from all the mote serial terminals, were logged and preserved in a file.

The necessary information was filtered out from the log file using a bash script. The details of the bash script are out of the scope of this paper but can be made available on request. Here necessary information meant, the update time which was the first time each mote reported that it was running new version of the software image, its ID that were preserved in a file in following fashion:

```
-----
| Time (in ms) | Mote ID |
-----
```

A cpp application was used to take input from this semi-processed information and another file that preserved the position information of the motes in the deployment. The application processed the data to find out the euclidean distance of the motes from the sink. It additionally sorted the rows based on update time in ascending order. At this instance, we preserved the processed information in a file in following fashion:

```
-----
| Time (in ms) | Mote ID | Distance |
-----
```

Building the Baseline Data: Processed data from multiple simulation runs initiated from the sink were considered to build the baseline data for each topology. Any change in the topology or variables in the environment setting would necessitate recalibrating the baseline. We considered 20 individual simulation runs initiated from the sink. The processed data from all these runs were statistically treated to find out the ‘mean’ and ‘standard deviation’ of update time recorded at each of the nodes. Both ‘mean’ and ‘standard deviation’ were used as input to the IDS to calculate the IWS according to the utility function shown in Equation-1.

Calculating IWS: Simulations were carried out to replicate intrusion attack at different nodes. The individual simulations runs were treated in a fashion similar to building baseline data. During an update replicating an intrusion, the update time recorded at each of the motes were processed and mathematically treated according to the utility function in Equation-1. The calculation performed the following computations:

- 1) Subtract the new update time from the corresponding ‘mean’ of reference update time for each of the motes to calculate the absolute difference. For example, new update time of updated mote ‘ n ’(new) is subtracted from ‘mean’ update time found from the baseline data for mote ‘ n ’ and its absolute value was calculated.
- 2) The absolute difference is divided by a function of ‘standard deviation’ to avoid any ‘division by zero’ scenario. This also smooths the data to avoid unrelated variations.
- 3) Sum up the resulting scores at all individual nodes in the WSN to find out the total IWS for the update.

The resulting numbers indicated the nature of the update as has been described in Subsection-V-D. This presented an approximation of abnormality of the new update pattern. The higher the score, the more dissimilar was the pattern.

However, the score itself does not preserve any topology awareness, as such a low score must be checked against topological orientation. This is required for additional contribution.

V. OVERVIEW OF THE SYSTEM

The proposed IDS resides on a computing system that is physically connected to the sink. It relies on existing protocols for its operations. Related information is collected utilising standard transport layer protocols. Here ‘related information’ indicates identification number of any mote i and the time at which the new image updated the mote t_i , is then processed. The IDS analyses the update time of each of the sensors in the WSN and computes a ‘Surprise Score’ or ‘Intrusion Warning Score’ (IWS) using utility function in Equation-1 and indicates the cases of a possible intrusion, i.e., a probable illegitimate update.

A. Threat Model

The threat model is directed towards injection of unauthorised program image to infect the entire WSN. The main threat scenario that the IDS is expected to be able to handle is compromise of a particular node in the WSN and then attacker being able to utilise the same node or replicate the node to inject a new software image using deluge protocol in the WSN. However, once the new image transfer has been injected, the node may remain alive or disappear. In case the compromised mote disappears, WSN would consider it as a node failure instead of node compromise. Actually, the node was hijacked to infect the WSN and create more confusion in determining how the incident took place. In addition to this, the IDS also considers a threat model where a new node pops up and injects the new software image by compromising existing WSN security mechanism.

The IDS cannot identify an intrusion in certain scenarios. For example, when an intrusion takes place in close vicinity of the sink, this IWS may not always be significantly large to be identified as an intrusion. On the other hand, due to node

compromise in one of the motes and intrusion initiated from another mote may result into unsurprising score due to fact that the effects may cancel each others contribution to the score. The deployment should consider additional security measures to handle such scenarios.

B. Assumptions

The system is expected to achieve a time synchronisation. That's entirely reasonable, but it needs to be stated as an assumption about the WSN under study. I AM NOT SURE IF THIS PART IS AN ASSUMPTION, THIS IS A FACT.

The thesis assumes several settings and capabilities that are applicable to the WSN. One of the most important assumption is: there is only one sink in the network. While WSN with multiple sinks are also possible, the IDS needs to be designed differently and are excluded from the study. It further assumes that the sink is protected from all kind of physical or logical attacks and cannot be compromised. The system do not consider mote mobility scenario and assumes all motes in the WSN are static.

The system operates in a fashion that whenever a new image is updated in a mote, it sends a conformation AM to all base stations in the WSN, thus the software update behaviour in the WSN is transparent to the IDS. The IDS is aware of the topological information that has been setup in the WSN. It knows the location of all nodes deployed in the network.

C. Algorithm

NOT STATED / DISCUSSED IN THIS PAPER
This is a part in the original thesis though.

D. System Design

The intrusion detection algorithm suggests some modification of the OTA update protocol by incorporating an Active Message (AM). An AM is a single-hop, unreliable packet used for network abstraction. It has a destination address, a type field, provides synchronous acknowledgements [5]. An AM can be used to carry necessary timing information to the sink once an object has been transferred completely to any mote and the mote reboots with the new image running. All motes in a WSN has a shared sense of time synchronisation which is accurate to $1 \mu s$ and typically is achieved using TSMP protocol [49]. The timing information carried by the AM is therefore synchronised with other nodes in the WSN and can be used readily for processing by the IDS.

1) *IDS Framework*: The IDS framework components has been shown in 1. The IDS has three kinds of components — 1) the motes in the WSN that are subject to legitimate or illegitimate update; 2) an IDS transport client (ITC) application that runs on the sink and works in coordination with the IDS to assists in intrusion detection; and 3) an IDS application that runs on a system with higher computing resources which is physically attached to the sink, and houses a network database (NDB). These components works in a unified fashion to make the IDS work.

The chronology of the activities that take place in a WSN in a software update and are associated with the IDS shown in 1. 1) Updates are initiated using deluge protocol; 2) Deluge

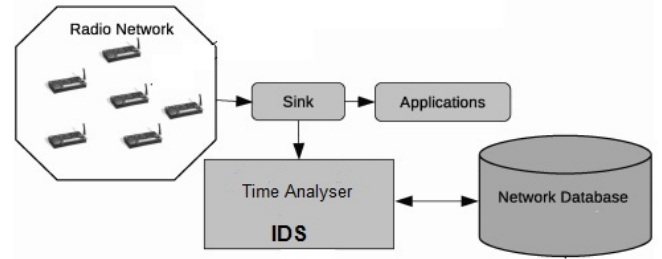


Fig. 1: IDS Framework

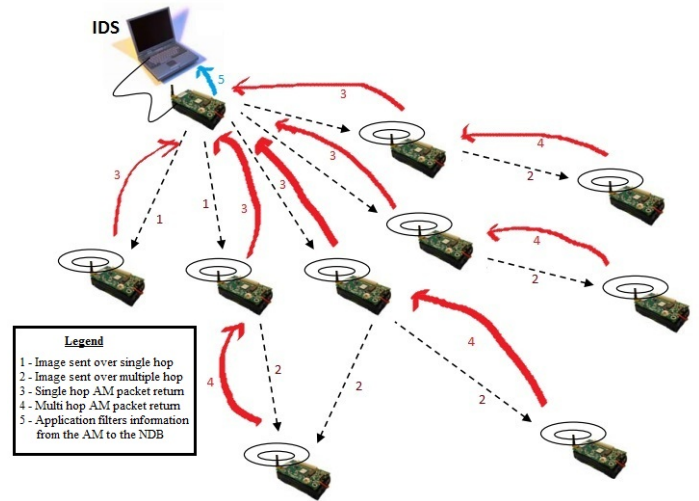


Fig. 2: IDS Modelling

disseminates the new software image using the motes enroute to reach all the motes using a relay mechanism; 3) The nodes deliver their timing information to the sink using an AM packet; 4) AM packets from the distant nodes rely on the other nodes enroute to reach the sink using a relay mechanism; and 5) The ITC application in the sink delivers relevant information to the IDS for processing. The IDS analyses the acquired timing information to arrive at decision.

2) *Determining IWS*: The intrusion detection activity begins with the logging of motes in the network by endorsing update time. Once updated with new software image, each node initiates an AM with the update time information to the sink. An AM packet initiated for the IDS has several fields: destination address, type identifier that indicates the update time information and the protocol for IDS, timing information and a checksum value. The packet travels through the network using existing transport layer protocols. When it arrives at the sink, the checksum value is examined by the ITC to ensure that the packet content has not been altered enroute. The ITC then inserts the update time, mote ID and related information to the NDB in the IDS. The IDS maintains an identifier to differentiate among different update times from different runs for the same mote.

The first software update run in the WSN is considered to be legitimate and the IDS takes this timing information as baseline. It is possible and desirable that the IDS is trained with

several other update runs. More training improved the ability of the IDS to detect intrusion more accurately. Utilising multiple runs are useful and beneficial as it increases accuracy and dependency. However, the IDS needs to be manually controlled to enable multiple training runs. Outcome from these runs are statistically treated to build the baseline.

Once the IDS baseline has been built, all subsequent updates are liable to scrutiny by the IDS. The timing analyser module in the IDS inspects the NDB once an update activity has been detected. It determines the changes in the statistics of the update time and calculates IWS based on following utility function:

$$IWS = \sum_{i=0}^n \frac{|\mu_i - t_i|}{\sigma_i + 1} \quad (1)$$

where,

- i — Identification of the Mote
- t_i — Image Update Time at Mote i
- μ_i — Mean Update Time at Mote i
- σ_i — Standard Deviation of Update Time at Mote i

3) *Building Intrusion Warning Zones*: Threshold for building of Intrusion Warning Zone (IWZ) depends on the assumed ability of the IDS having IWS knowledge of all possible intrusions in the network. Such knowledge can be built by replicating intrusion activity from all nodes in the network once. However, in case where it is not possible, replicating intrusion activity from the furthest few nodes may serve the purpose. IDS can build a scale of possible expected range of IWS for the deployed topology and can classify the IWS in different zones to imply certain warning level.

The IDS considers an IWS upto 10% from the scale to be safe and marks as 'GREEN' zone, an IWS above 30% to be possible intrusion and is marked as 'RED' zone. However, an IWS within the range of 10% — 30% can neither be considered safe, nor an intrusion, hence is marked as 'GREY' that indicates a situation where rules are not known. However, the thresholds are not conclusive and may require some variation based on the density, power level used in the WSN and scale of deployment.

4) *Database Structure*: The NDB stores the statistics of the update time of all the motes in the WSN for each update applied. It consists of three flat tables that performs as storage. The tables have been designed to improve the IDS performance. First, In-Network Activity Table (INT) consists of the following fields:

```
mote_id, timestamp, id_seq,
upd_seq, upd_type, zone_type
```

where, *mote_id* is the source node ID, *timestamp* is the time when the new software image starts executing, *id_seq* is the number of times an update arrived from same *mote_id*, *upd_seq* is the number of time an update has been initiated from the sink. The unique key for INT is *mote_id, upd_seq* pair. *upd_type* indicates if the record belonged to a baseline update. For a baseline update type, the *zone_type* is zero, while

for others it is updated to indicate if the update related to that particular mote from that particular run was processed as GREEN, GREY or RED.

Secondly, the Baseline Table (BLT) holds the processed information from the legitimate runs and has the following fields:

```
mote_id, mu, sigma
```

where, *mote_id* is the source node ID, *mu* is the mean time and *sigma* is the standard deviation for each *mote_id* from all legitimate runs.

Finally, the Intrusion Activity Table (IAT) filters out necessary information from INT about most recent software update and contributes to the performance of the IDS. The table has two fields:

```
mote_id, timestamp
```

where, *mote_id* and *timestamp* are same as the corresponding fields in INT. Every time a decision on an update has been made and a new update is detected, this table is flashed.

5) *Activities in the Network*: As part of enhancement into the deluge protocol, the motes are enriched with the prior knowledge of the sink or sinks in the WSN. Whenever an update takes place in a mote, it reboots with the new software running and stores the bootup time in its memory. I NEED TO CONFIRM IF THE TIME IS ALREADY SAVED WITH THE IMAGE AS PART OF DELUGE PROTOCOL IMAGE/GOLDEN IMAGE. The locally stored information is used later in case it requires to recover from packet delivery failure. The information *type, mote_id, time_stamp, upd_seq* are packed along with an additional checksum in an AM packet and forwarded to the sink/sinks. The AM has the following information in the packet:

```
|-----|
|Source|-----|
|  ID  |msgtype|timestamp|upd_seq|checksum|
|-----|
```

Figure: Need to draw a better picture

The AM packet is relayed through the network to the sink.

6) *Activities in the Sink*: As an AM arrives from the radio network to the sink, the ITC application running on the sink verifies the checksum. If the checksum fails, the packet is discarded and failure recovery is initiated once only in coordination with the IDS. However, if the AM passes checksum, the ITC application extracts the relevant information and populates the INT in the resident host system.

7) *Activities in the IDS*: An update is typically initiated from the host system where the IDS resided. The IDS constantly monitors the programming port and coordinates with the ITC. They automatically identify the first update as input for baseline training data. It is possible to put the IDS in training mood for extended number of updates. Result from all updates run during the training period are statistically treated to build the baseline. Though the IDS can work theoretically

with a single update, it is recommended to have minimum three updates for the training phase while running more training updates is desirable. For our experiments, we have considered 20 runs for each topology.

Once the training phase ends, the IDS calculates mean and standard deviations at each of the motes and stores the information on BLT. It also stores the number of update runs it considered for the BLT. This number contributes two folds: 1) it indicates the reliability of the training data / baseline; and 2) the number is used for statistical processing in case there is a need to increase the training data at a later time or data from an update is identified as acceptable is automatically included in baseline computation and the BLT is updated. Once the BLT has been built, the IDS runs on active mood and all updates are subject to intrusion checks. When an update is initiated in the WSN, data pushed into the INT are filtered and relevant information related to the ongoing update populates the IAT. The IDS waits upto a specific time, which is determined by the maximum time taken to complete a network-wide update during the training phase, before it reports the findings about intrusion possibility. While the procedure described above is quite straight forward, it gets complicated because of the unreliable packet transmission pattern experienced by WSN due to node failure, link failure and node reboot [50], [51]. IDS reports the findings with a provision of updating it later when an expected data from certain mote are not available within reasonable time. In such case, the IDS employs a packet recovery mechanism to obtain necessary information.

The packet recovery mechanism is initiated when a valid packet is not available from one or more motes within the determined maximum time from the updates during the training phase. Upon receiving signal from the IDS, the ITC initiates AM to the specific requesting a retransmission of the update time and waits further triple time for the required AM packet. If the packet is available, IWS is recomputed and decision is updated accordingly. If the packet is still not available within specified time, the node is marked as compromised or failed and a notification is generated.

VI. EVALUATION

The performance and effects of the proposed IDS were tested in different topologies, such as linear, circular, elliptical, double rings, wide lines/grid, tree, and other topologies. While discussing all of these topologies are useful, we shall concentrate our discussion on results and findings elliptical topology only. Discussing any of the topologies presents fair idea about the outcomes from the others. We shall mention the cases when the outcome was considered to be exception and contrary to the expectations. The space requirement also motivates us to limit the discussion to necessary details only.

We have selected data from elliptical topology for several reasons. Ellipse has a shape which is almost midway among the range of topologies between a circular and a linear one. Extending the minor axis of the ellipse makes the topology equivalent to a circle, while contracting it is equivalent to a grid or even a linear topology. Thickening the perimeter of the circle turns it into a double ring. Again, the linear topology can be considered as a tree with one or two branches. Extending it with more branches presents a fairly complex tree structure.

Because of these similarities, discussing elliptical topology is an interesting choice.

The shape of the ellipse has been designed considering several requirements. The ellipse shape deployment has been shown in Figure-???. The motes on either side of minor axis remained within each others transmission range, whereas motes on the either side of the major axis did not. Total 20 nodes were deployed with the minor axis being 30 meter and the major axis being way larger than that. The transmission power level was set at 100% which allowed a transmission range of 50 meter and nodes were able to interfere other nodes at upto 100 meter apart. The motes were placed at a space of 35 meter along the circumference of the ellipse. The deployment enabled multiple transmission paths to all the motes. It contributed to achieve redundancy and reliable image transfer.

The simulations proceeded without interruption and produced expected results. To build the baseline data, we assigned mote ID 1 as the sink and ran 20 updates initiated from the sink. The output from these runs were processed as describes in Section-IV and were statistically treated to compute the 'mean' and 'standard deviation' of the time at which each of the nodes were updated with the new image. The computation results have been presented in Table-II.

Once the baseline data were built, intrusion scenarios were replicated at other nodes according to the experiment protocols. Output of these replications were processed to compute IWS. Results obtained from these simulated intrusions are also reported in Table-II.

IWS shown in the last row of Table-II were used to build a scale that ranged from 0 to maximum of *IWS*. Building this scale was crucial for the ability to classify the IWS to meaningful zones. However, it is not possible to build the scale without the knowledge of the expected IWS from the furthest nodes in the network. Nevertheless, we established two threshold levels at 10% and at 30% that were useful in zone classification using intuitive heuristics. Details of the zoning has been discussed in Section-V-D3.

Table-II shows the statistical measurements obtained at different motes and the IWS measured when each of the motes were considered as intrusion point. It can be clearly seen that IWS at the intrusion points differ considerably. The first row in Table-II contain the Node IDs that are headers for the data displayed underneath. The next two rows describe the 'mean' and 'standard deviation' that were computed from the 20 legitimate updates initiated from the sink. Cells in the last row contain the IWS calculated using the utility function shown in Equation-1 for the test runs initiated from the node indicated at top of the column. The data shown in the table can best be interpreted when related to Figure-???. Figure-?? shows the actual relative positions of the motes with mote ID written at the center of the tiny circle and IWS measured when an intrusion attempt has been initiated from the mote. The figure additionally divides the region in GREEN, GREY and RED zones according to the logic discussed in V-D7. The graph in Figure-?? shows the relative measures of the IWS from the motes.

From the data presented in Table-II, analysing the baseline data we find that the motes further away from the sink are

Node ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
μ	0	19	19	69	139	213	276	344	377	432	432	432	377	335	273	207	137	68	17	19
σ	0	5	5	6	22	36	37	43	44	44	44	44	44	45	44	33	24	5	3	5
IWS	0.01	16	19	73	140	203	246	336	344	396	395	535	406	332	269	237	113	72	16	16

TABLE II: Mean and Standard Deviation of motes and IWS from intrusion at motes. Further explained in Figure-??

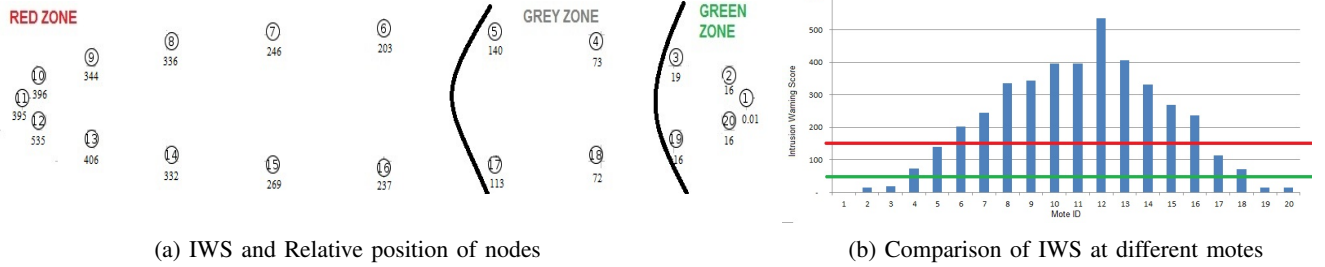


Fig. 3: Relative position and comparison of IWS for intrusion at different motes from data in Table-II

found to have higher ‘mean’ and ‘standard deviation’ in terms of update time. However, ‘standard deviation’ raises gradually, though it does not vary to a great extent for relatively small distance difference when multiple paths are present. This results from the availability of more redundant paths make the image transfer more efficient and reliable which contribute to the stability of ‘standard deviation’. Therefore, even remaining at considerably different distances from the sink, mote ID 8, 9, 10, 11, 12, 13, 14 and 15 have quite similar ‘standard deviation’ of update time. In contrast to this, ‘mean’ increases sharply as the motes are placed further from the sink. The quantities established from the baseline data shows that motes closer to the sink have lower ‘mean’ which increases sharply with the increase in distance.

The utility function presented in Equation-1 is designed to neutralise the undesired changes in ‘mean’ and ‘standard deviation’ which are expected to contribute to a near *zero* IWS. A near *zero* IWS is further classified as GREEN zone to indicate a legitimate software update. Table-II and Figure-?? shows defined zones and IWS measured when intrusion was initiated at different motes.

The IWS increases linearly with the increase in distance from the sink. The ‘distance’ term can be misleading in the context of our discussion and demands further clarification. Euclidean distance between a mote and the sink does not make useful sense as the packets in the network must travel using a relay mechanism through other motes in the network. Because of the availability of the multiple paths, it is not possible to quantify the exact distance that a packet would travel while transporting from a mote to the sink. However, an estimation of distance along the most probable link path based on the connectivity may be considered.

IWS is considerably higher for intrusion at distant motes than ones which are close to the sink. This phenomena is topology dependent, which also implies that it is connectivity dependent. It is possible that if IWS in a network are organised in a specific order, it would represent/resemble the original topology in some fashion. For example, in the elliptical topology, the link path between node 1 and node 11 through node 5 is represented by a curve with a positive slope or rise. On

the other hand, the link path through node 15 is represented by a fall or a negative sloped curve. There is a relationship between the curvature/eccentricity of the original topology and the represented curve or trend. However, establishing the relationship is beyond the scope of this paper.

The main contribution of this research project comes from the fact that the IDS is able to report an anomaly from a software update in quantitative term like the IWS. The claim is evident from the data presented in Table-II. For example: in this case, a regular legitimate update would be initiated from node ID 1. When such a scenario was simulated and the timing information was compared against the baseline, it resulted into a IWS of 0.01. The result matched the theoretical expectation, as a legitimate update initiated from the sink should produce a near *zero* IWS. On the other hand, when intrusion scenario were simulated from the nodes near the sink, the IWS reported was fairly low, e.g. 16 for mote ID 2, 19, and 20 and 19 for node 3 and likewise. Because of the fact that IWS of the motes close to the sink are likely to be very low, which is in line with the theoretical expectation, it is quite difficult to conclude if such a IWS indicate an intrusion. However, for the distant motes, the IWS is quite high and the IDS can easily conclude as intrusion, e.g. IWS of 246 for mote ID 7, 269 for mote ID 15, 395 for mote ID 11, 535 for mote ID 12 and likewise indicate intrusion.

While reporting severity or likelihood of an intrusion in quantitative term like the IWS is useful, it is more beneficial if the quantity is associated with some kind of interpretation. The technique of associating such interpretation has been discussed in Section-V-D3 and in this Section. In the elliptical topology intrusion scenario, maximum IWS reported in the presented data in Table-II was 535. Motes reporting an IWS upto 53.5 would be considered in the GREEN zone. For example, in figure-??, mote ID 2, 3, 19, and 20 are in this zone. On the other hand, IWS above 53.5 and below 160.5 are considered to be in the GREY zone. Motes in this zone are 4, 5, 17, and 18. They can neither be considered free from intrusion, nor it is possible to easily arrive at a decision that the scores indicate intrusion for sure. In contrast to both the scenario describes above, IWS above 160.5 is considered to be an intrusion. All

Node ID	1	2	3	4	5	6	7	8	9	10	11	...	31	32	33	34	35	36	37	38
Power 100%	512	529	512	512	512	512	476	478	476	459	458	...	49	47	53	48	49	51	47	29
Power 50%	31	59	254	31	31	283	32	32	254	31	253	...	31	31	31	30	31	31	30	0

TABLE III: IWS from intrusion at motes in ‘Owheo Sensor Network’. Further explained in Figure-??

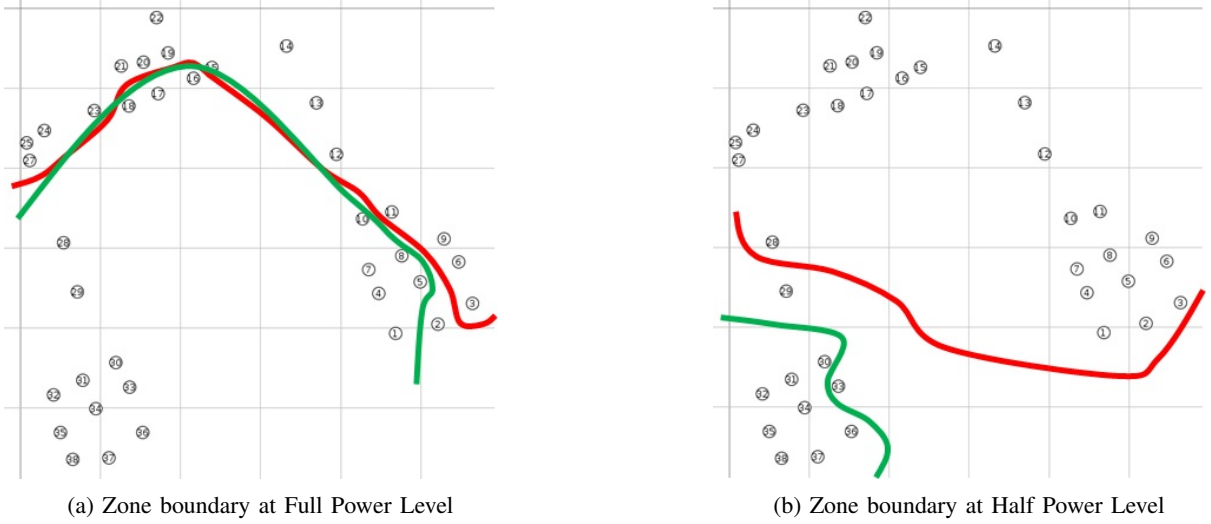


Fig. 4: Comparison of Zone boundary for power level variations in ‘Owheo Sensor Network’. IWS of the motes at the power levels are shown in Table-III

other motes in the topology have a IWS above this threshold level and hence are considered to be more susceptible to intrusion.

The idea of zoning associated with the IWS is very beneficial in securing a WSN. It is immensely useful in designing a secure network deployment, which is considered to be another major contribution of the research work. The motes which are found to be in the GREEN zone must be adequately secured by design with additional physical security as intrusion in these motes are not observed by the IDS. On the other hand, motes in the GREY zone need additional monitoring which would aid determining if the IWS indicated a real intrusion or a false positive. The additional monitoring could be in the form of an algorithmic or physical observation or listening post or even employing additional protocol. The rest of the region, identified by the RED zone, is under the effective surveillance of the IDS and is considered to be secure.

The concept of zoning using the IDS is a very useful tool in designing secure WSN deployment. A network is more secure when it has a smaller GREEN and GREY zone. A smaller GREEN zone implies the requirement of lesser resources to impenetrably secure less number of motes. Similarly, a smaller or non-existent GREY zone would lesser monitoring measures. Such a secure WSN can be designed by physically relocating some of the motes or even by varying the factors like power level. For example, we have simulated the IWS scenario of the ‘Owheo Sensor Network’ at two different power levels in our quest to design a secure WSN using the IDS. ‘Owheo Sensor Network’, which is the WSN deployed in the Department

of Computer Science building at the University of Otago, is intended to be used for research purposes.

Table-III shows the IWS at different motes as intrusion points at 100% and 50% power levels. In ‘Owheo Sensor Network’, mote ID 38 was marked as the sink and mote ID 26 does not exist. At 100% power level, 19 nodes (mote ID 1, 4, 5, 7, 8, 10, 16, 17, 18, 28, 29, 30, 31, 32, 33, 34, 35, 36, and 37) were in GREEN zone, only one node (mote ID 2) was in GREY zone and the rest 16 nodes were in RED zone. The deployment and the zoning boundaries at 100% power level has been shown in Figure-??. When the same deployment was simulated at 50% power level, only seven nodes (mote ID 30, 31, 32, 34, 35, 36, and 37) were in GREEN zone, two nodes (mote ID 29, and 33) were in GREY zone and rest 27 motes were in RED zone. The deployment and the zoning boundaries at 50% power level has been shown in Figure-??. Comparing the scenarios presented in Figure ?? and ??, it can clearly be established that Figure-??. i.e. deployment at 50% power level is more secure than the other because of two reasons: 1) it has a smaller sized GREEN and GREY zone; and 2) there are lesser number of nodes in these zones. The deployment can be made more secure by physically relocating some of the motes from GREEN zone to RED zone, provided the relocation does not hamper the original intended purpose of the WSN.

The IWS is also able to detect node failures in the WSN. This additional advantage comes as a by-product contribution. However, as this was not a design consideration, the contribution is robust enough, yet it is worth mentioning. The case of a node not reporting its update time within a specified

delay is handled by a error recovery procedure put in place by the IDS. If the recovery procedure described in section-V-D6 fails, there is a possibility that the node might have failed or have been compromised. However, such failure resulting into contributing towards an IWS which is quite different for the update pattern detected from the timing information reported by the other nodes. The IDS structure is such that it has an expected pattern and shape of IWS for an intrusion at some specific point. If the shape is distorted to a great extent, it indicates some unusualness which can be attributed to some other phenomena like node failure or compromise. The node failure or node compromise can take place concurrently with or without intrusion. It might also indicate a different type of attack associated with a physical intrusion. An interpretation from the IDS view of the topology would enable identification of some other types of attacks in WSN like node relocation, node repudiation, node compromise.

VII. CONCLUSION

Building secure WSN is of paramount importance, yet it is quite difficult.

Wrap the whole thing up;

A. write here

Signature _____

Contact Address: 128, Owheo Building
133, Union Street (West)
Dunedin - 9010, New Zealand

REFERENCES

- [1] A. Bari, A. Jaekel, and S. Bandyopadhyay, "Optimal placement of relay nodes in two-tiered, fault tolerant sensor networks," in *2007 IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, VOLS 1-3*, ser. IEEE Symposium on Computers and Communications ISCC. IEEE; IEEE Commun Soc; IEEE Comp Soc; Inst Telecommun; Univ Aveiro; Inovacao; AT&T, 2007, pp. 809–814, 12th IEEE Symposium on Computers and Communications, Aveiro, PORTUGAL, JUL 01-04, 2007.
- [2] V. Lakafosis, A. Rida, R. Vyas, L. Yang, S. Nikolaou, and M. Tentzeris, "Progress towards the first wireless sensor networks consisting of inkjet-printed, paper-based rfid-enabled sensor tags," *Proceedings of the IEEE*, vol. 98, no. 9, pp. 1601–1609, Sept 2010.
- [3] "IEEE Standard Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Std. 802.15.4-2011 (Revision of IEEE Std. 802.15.4-2006)*, September 2011.
- [4] R. Roman, J. Zhou, and J. Lopez, "Applying intrusion detection systems to wireless sensor networks," in *Consumer Communications and Networking Conference*, vol. 1, 2006, pp. 640–644.
- [5] P. Levis, "Tep 116: Packet protocols," TinyOS Core Working Group, Tech. Rep., 2012, tinyOS-Version: i 2.1.
- [6] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," in *The first IEEE international conference on system integration and reliability improvements*, vol. 25, 2006, p. 94.
- [7] T. G. Roosta, *Attacks and defenses of ubiquitous sensor networks*. ProQuest, 2008.
- [8] Y. W. Law, J. Doumen, and P. Hartel, "Survey and Benchmark of Block Ciphers for Wireless Sensor Networks," *ACM TRANSACTIONS ON SENSOR NETWORKS*, vol. 2, no. 1, FEB 2006.
- [9] M. Healy, T. Newe, and E. Lewis, *Smart Sensors and Sensing Technology*, ser. part 1. Springer Berlin Heidelberg, 2008, ch. Analysis of Hardware Encryption Versus Software Encryption on Wireless Sensor Network Motes, pp. pp 3–14, doi 10.1007/978-3-540-79590-2_1.
- [10] X. Zhang, H. Heys, and C. Li, "Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks," in *25th Biennial Symposium on Communications (QBSC), 2010*, May 2010, pp. 168–172.
- [11] A. Youssef, S. Tavares, and H. Heys, "A new class of substitution-permutation networks," in *Workshop on Selected Areas in Cryptography: SAC '96, Workshop Record 1996*. Queen's University, 1996, pp. 132–147.
- [12] H. M. Heys and S. E. Tavares, "Cryptanalysis of substitution-permutation networks using key-dependent degeneracy," *J-CRYPTOLOGIA*, vol. 20, no. 3, pp. 258–274, 1996.
- [13] S. Ben Othman, A. Trad, and H. Youssef, "Performance evaluation of encryption algorithm for wireless sensor networks," in *International Conference on Information Technology and e-Services (ICITeS), 2012*, March 2012, pp. 1–8.
- [14] H.-L. Yeh, T.-H. Chen, P.-C. Liu, T.-H. Kim, and H.-W. Wei, "A Secured Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography," *SENSORS*, vol. 11, no. 5, pp. 4767–4779, MAY 2011.
- [15] K. Xue, C. Ma, P. Hong, and R. Ding, "A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks," *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, vol. 36, no. 1, pp. 316–323, JAN 2013.
- [16] C.-T. Li, C.-Y. Weng, and C.-C. Lee, "An Advanced Temporal Credential-Based Security Scheme with Mutual Authentication and Key Agreement for Wireless Sensor Networks," *SENSORS*, vol. 13, no. 8, pp. 9589–9603, AUG 2013.
- [17] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, ser. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 245–256. [Online]. Available: <http://dx.doi.org/10.1109/IPSN.2008.47>
- [18] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Címará, J. López, and R. Dahab, "Tinybpc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *Comput. Commun.*, vol. 34, no. 3, pp. 485–493, Mar. 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.comcom.2010.05.013>
- [19] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique Cryptanalysis of the Full AES," in *ADVANCES IN CRYPTOLOGY - ASIACRYPT 2011*, ser. Lecture Notes in Computer Science, Lee, DH and Wang, XY, Ed., vol. 7073. Korea Univ, Ctr Informat Secur Technol; Korean Federat Sci & Technol Soc; Seoul Natl Univ; Elect & Telecommunicat Res Inst; Seoul Metropolitan Govt, 2011, pp. 344–371, 17th Annual International conference on the Theory and Application of Cryptology and Information Security, Seoul, SOUTH KOREA, DEC 04-08, 2011.
- [20] J. Lopez, "Unleashing public-key cryptography in wireless sensor networks," *Journal of Computer Security*, vol. 14, no. 5, pp. 469–482, 2006.
- [21] I. Butun, S. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 266–282, First 2014.
- [22] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in *2007 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, VOLS 1-14*, ser. IEEE International Conference on Communications. IEEE, 2007, pp. 3864–3869, IEEE International Conference on Communications (ICC 2007), Glasgow, SCOTLAND, JUN 24-28, 2007.
- [23] I. Onat and A. Miri, "A real-time node-based traffic anomaly detection algorithm for wireless sensor networks," in *2005 Systems Communications, Proceedings: ICW 2005, WIRELESS TECHNOLOGIES; ICHSN 2005, HIGH SPEED NETWORKS; ICMCS 2005, MULTIMEDIA COMMUNICATIONS SYSTEMS; SENET 2005, SENSOR NETWORKS*, Dini, P and Lorenz, P and Soulhi, S and Cherkaoui, S and Mynbaev, D and Rodrigues, JJ and Hafid, A and Zepernick, HJ and Zheng, J, Ed. IEEE Montreal Sect; Bell; IARIA, 2005, pp. 422–427, International Conference on Systems Communications, Montreal, CANADA, AUG 14-17, 2005.
- [24] F. Santoso, "A Decentralised Self-dispatch Algorithm for Square-grid Blanket Coverage Intrusion Detection Systems in Wireless Sensor

- Networks,” in *2011 IEEE VEHICULAR TECHNOLOGY CONFERENCE (VTC FALL)*, ser. IEEE VTS Vehicular Technology Conference Proceedings. IEEE; IEEE Vehicular Technol Soc, 2011, IEEE 74th Vehicular Technology Conference (VTC), San Francisco, CA, SEP 05-08, 2011.
- [25] R.-C. Chen, C.-F. Hsieh, and Y.-F. Huang, “A new method for intrusion detection on hierarchical wireless sensor networks,” in *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '09. New York, NY, USA: ACM, 2009, pp. 238–245. [Online]. Available: <http://doi.acm.org/10.1145/1516241.1516282>
- [26] C.-C. Su, K.-M. Chang, Y.-H. Kuo, and M.-F. Horng, “The new intrusion prevention and detection approaches for clustering-based sensor networks [wireless sensor networks],” in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, March 2005, pp. 1927–1932 Vol. 4.
- [27] A. A. Strikos, “A full approach for intrusion detection in wireless sensor networks,” 2007.
- [28] R. Roman, J. Zhou, and J. Lopez, “Applying intrusion detection systems to wireless sensor networks,” in *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, vol. 1, Jan 2006, pp. 640–644.
- [29] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, “Distributed anomaly detection in wireless sensor networks,” in *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*, Oct 2006, pp. 1–5.
- [30] E.-H. Ngai, J. Liu, and M. Lyu, “On the intruder detection for sinkhole attack in wireless sensor networks,” in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 8, June 2006, pp. 3383–3389.
- [31] S. Doumit and D. Agrawal, “Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks,” in *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, vol. 1, Oct 2003, pp. 609–614 Vol.1.
- [32] I. Onat and A. Miri, “A real-time node-based traffic anomaly detection algorithm for wireless sensor networks,” in *Systems Communications, 2005. Proceedings*, Aug 2005, pp. 422–427.
- [33] —, “An intrusion detection system for wireless sensor networks,” in *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, vol. 3, Aug 2005, pp. 253–259 Vol. 3.
- [34] A. Agah, S. Das, K. Basu, and M. Asadi, “Intrusion detection in sensor networks: a non-cooperative game approach,” in *Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium on*, Aug 2004, pp. 343–346.
- [35] S. K. Das, “Preventing dos attacks in wireless sensor networks: A repeated game theory approach,” *International Journal of Network Security*, pp. 145–153, 2007.
- [36] Y. B. Reddy, “A game theory approach to detect malicious nodes in wireless sensor networks,” in *Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, ser. SENSORCOMM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 462–468. [Online]. Available: <http://dx.doi.org/10.1109/SENSORCOMM.2009.76>
- [37] F. Bao, I.-R. Chen, M. Chang, and J.-H. Cho, “Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection,” *Network and Service Management, IEEE Transactions on*, vol. 9, no. 2, pp. 169–183, June 2012.
- [38] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, “Securing the deluge network programming system,” in *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2006, pp. 326–333.
- [39] C. Karlof, N. Sastry, and D. Wagner, “Tinysec: A link layer security architecture for wireless sensor networks,” in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 162–175, <http://doi.acm.org/10.1145/1031495.1031515>.
- [40] K. Ioannis, T. Dimitriou, and F. C. Freiling, “Towards intrusion detection in wireless sensor networks,” in *Proc. of the 13th European Wireless Conference*, 2007.
- [41] J. Sen, “An efficient security mechanism for high-integrity wireless sensor networks,” *arXiv preprint arXiv:1012.2516*, 2010.
- [42] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “Interleaved hop-by-hop authentication against false data injection attacks in sensor networks,” *ACM Trans. Sen. Netw.*, vol. 3, no. 3, August 2007, issn 1550-4859. [Online]. Available: <http://doi.acm.org/10.1145/1267060.1267062>
- [43] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *USENIX ASSOCIATION PROCEEDINGS OF THE FIRST SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI'04)*. USENIX Assoc; ACM SIGOPS, 2004, pp. 15–28, 1st Symposium on Networked Systems Design and Implementation, San Francisco, CA, MAR 29-31, 2004.
- [44] S. S. Kulkarni and L. Wang, “Mnp: Multihop network reprogramming service for sensor networks,” in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, ser. ICDCS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 7–16. [Online]. Available: <http://dx.doi.org/10.1109/ICDCS.2005.50>
- [45] P. De, Y. Liu, and S. K. Das, “ReMo : An Energy Efficient Reprogramming Protocol for Mobile Sensor Networks,” in *2008 IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS*. IEEE Comp Soc; Hong Kong Univ Sci & Technol; Univ Texas; Hong Kong Polytechn Univ; Croucher Fdn; KC Wong Educ Fdn; IBM Res; Google; Elsevier Publicat, 2008, pp. 60–69, 6th IEEE International Conference on Pervasive Computing and Communications, Hong Kong, PEOPLES R CHINA, MAR 21-21, 2008.
- [46] C.-J. M. Liang, R. Musäloiu-e, and A. Terzis, “Typhoon: A reliable data dissemination protocol for wireless sensor networks,” in *Wireless Sensor Networks*. Springer Berlin Heidelberg, 2008, pp. 268–285.
- [47] G. Maia, A. L. Aquino, D. L. Guidoni, and A. A. Loureiro, “A multicast reprogramming protocol for wireless sensor networks based on small world concepts,” *Journal of Parallel and Distributed Computing*, vol. 73, no. 9, pp. 1277 – 1291, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0743731513001111>
- [48] J. W. Hui and D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 81–94.
- [49] K. S. J. Pister and L. Doherty, “Tsmp: Time synchronized mesh protocol,” in *In Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08)*, 2008.
- [50] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, “A line in the sand: A wireless sensor network for target detection, classification, and tracking,” *Comput. Netw.*, vol. 46, no. 5, pp. 605–634, Dec. 2004. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2004.06.007>
- [51] R. Beckwith, D. Teibel, and P. Bowen, “Unwired wine: sensor networks in vineyards,” in *Sensors, 2004. Proceedings of IEEE*, Oct 2004, pp. 561–564 vol.2.