International Journal of Science and Engineering Research (IJ0SER),

Vol 3 Issue 11 November-2015
3221 5687, (P) 3221 568X

Enhancement and Detection of Unauthorized User in Wireless Sensor Network Using SDiDrip

[1]M.Senthil Kumar, [2] P.Partha Sarathi ME.,PhD
[1] Student, [2] Assistant Professor
Department of Computer Science & Engineering ,
Akshaya College of engineering &Technology
Coimbatore,Tamilnadu
india

Abstract- **A data discovery and dissemination protocol for wireless sensor networks (WSNs) is responsible for updating configuration parameters of, and distributing management commands to, the sensor nodes. All existing data discovery and dissemination protocols suffer from two drawbacks. First, they are based on the centralized approach; only the base station can distribute data items. Such an approach is not suitable for emergent multi-owner-multi-user WSNs. Second, those protocols were not designed with security in mind and hence adversaries can easily launch attacks to harm the network. This paper proposes the first secure and distributed data discovery and dissemination protocol named SDiDrip. It allows the network owners to authorize multiple network users with different privileges to simultaneously and directly disseminate data items to the sensor nodes. Moreover, as demonstrated by our theoretical analysis, it addresses a number of possible security vulnerabilities that we have identified. Extensive security analysis show SDiDrip is provably secure. We also implement SDiDrip in an experimental network of resource-limited sensor nodes to show its high efficiency in practice.**

Keywords- **wireless sensor networks (WSNs), SDiDrip , DIP and Drip, DHV**

INTRODUCTION

After a wireless sensor network (WSN) is deployed, there is usually a need to update buggy/old small programs or parameters stored in the sensor nodes. This can be achieved by the so-called data discovery and dissemination protocol, which facilitates a source to inject small programs, commands, queries, and configuration parameters to sensor nodes. Note that it is different from the code dissemination protocols (also referred to as data dissemination or reprogramming protocols) which distribute large binaries to reprogram the whole network of sensors. For example, efficiently disseminating a binary file of tens of kilobytes requires a code dissemination protocol while disseminating several 2-byte configuration parameters requires data discovery and dissemination protocol. Considering the sensor nodes could be distributed in a harsh environment, remotely disseminating such small data to the sensor nodes through the wireless channel is a more preferred and practical approach than manual intervention.

In the literature, several data discovery and dissemination protocols have been proposed for WSNs. Among them, DHV , DIP and Drip are regarded as the state-of-the-art protocols and have been included in the TinyOS distributions. All proposed protocols assume that the operating environment of the WSN is trustworthy and has no adversary. However, in reality, adversaries exist and impose threats to the normal operation of WSNs . This issue has only been addressed recently by which identifies the security vulnerabilities of Drip and proposes an effective solution.

EXISTING SYSTEM

The existing data discovery and dissemination protocols employ the centralized approach in which, data items can only be disseminated by the base station. Unfortunately, this approach suffers from the single point of failure as dissemination is impossible when the base station is not functioning or when the connection between the base station and a node is broken. In addition, the centralized approach is inefficient, non-scalable, and vulnerable to security attacks that can be launched anywhere along the communication path [2]. Even worse, some WSNs do not have any base station at all. For example, for a WSN monitoring human trafficking in a country's border or a WSN deployed in a remote area to monitor illicit crop cultivation, a base station becomes an attractive target to be attacked..

The underlying algorithm of both DIP and Drip is Trickle. Initially, Trickle requires each node to periodically broadcast a summary of its stored data. When a node has received an older summary, it sends an update to that source. Once all nodes have consistent data, the broadcast interval is increased exponentially to save energy. However, if a node receives a new summary, it will broadcast this more quickly. In other words, Trickle can disseminate newly injected data very quickly. Among the existing protocols, Drip is the simplest one and it runs an independent instance of Trickle for each data item.

In practice, each data item is identified by a unique key and its freshness is indicated by a version number. For example, for Drip, DIP and DHV, each data item is represented by a 3-tuple <key; version; data>, where key is used to uniquely

identify a data item, version indicates the freshness of the data item (the larger the version, the fresher the data), and data is the actual disseminated data (e.g., command, query or parameter)

- Centralized approach
- Only the base station can distribute data items

- When the connection between the base station and a node is broken the existing system will not works.
- Vulnerable to security attacks

The Dynamic Behavior Of A Data Dissemination Protocol For Network Programming At Scale

To support network programming, we present Deluge, a reliable data dissemination protocol for propagating large data objects from one or more source nodes to many other nodes over a multihop, wireless sensor network. Deluge builds from prior work in density-aware, epidemic maintenance protocols. Using both a real-world deployment and simulation, we show that Deluge can reliably disseminate data to all nodes and characterize its overall performance. On Mica2-dot nodes, Deluge can push nearly 90 bytes/second, one ninth the maximum transmission rate of the radio supported under TinyOS. Control messages are limited to 18% of all transmissions. At scale, the protocol exposes interesting propagation dynamics only hinted at by previous dissemination work. A simple model is also derived which describes the limits of data propagation in wireless networks. Finally, we argue that the rates obtained for dissemination are inherently Lower than that for single path propagation. It appears very hard to significantly improve upon the rate obtained by Deluge and we identify establishing a tight lower bound as an open problem.

Wireless sensor networks (WSNs) represent a new class of computing with large numbers of resource-constrained computing nodes cooperating on essentially a single application. WSNs must often operate for extended periods of time unattended, where evolving analysis and environments can change application requirements, creating the need alter the network's behavior by introducing new code. Collectively programming all nodes before deployment could put thousands Of nodes within communication range, making suppression of protocol messages essential. Third, complete reliability is required since every byte must be correctly received by all nodes that need reprogramming, even in the presence of high loss rates and evolving link qualities common to WSNs.

Deluge's density aware, epidemic properties help achieve reliability in unpredictable wireless environments and robustness when node densities can vary by factors of a thousand or more. Representing the data object as a set of fixed-size pages provides a manageable unit of transfer which allows for spatial multiplexing and supports efficient incremental upgrades. Second, we characterize the propagation dynamics of Deluge using a real-world deployment and simulation. In a real deployment, Deluge can disseminate data with 100% reliability at nearly 90 bytes/second, one-ninth the maximum transmission rate of the radio supported under Tiny OS. Under simulation, propagation of large data objects in large networks exposes interesting propagation behavior only hinted at in previous dissemination work. Third, we develop a simple model of Deluge's propagation behavior and use it to identify different factors which limit the overall bandwidth
of any multihop communication protocol.

Dhv: A Code Consistency Maintenance Protocol For Multi-Hop Wireless Sensor Networks

Ensuring that every sensor node has the same code version is challenging in dynamic, unreliable multi-hop sensor networks. When nodes have different code versions, the network may not behave as intended, wasting time and energy. We propose and evaluate DHV, an efficient code consistency maintenance protocol to ensure that every node in a network will eventually have the same code. DHV is based on the simple observation that if two code versions are different, their corresponding version numbers often differ in only a few least significant bits of their binary representation. DHV allows nodes to carefully select and transmit only necessary bit level information to detect a newer code version in the network. DHV can detect and identify version differences in O(1) messages and latency compared to the logarithmic scale of current protocols. Simulations and experiments on a real MicaZ testbed show that DHV reduces the number of messages by 50%, converges in half the time, and reduces the number of bits transmitted by 40-60% compared to DIP, the state-of-the-art protocol.

wireless sensor network deployments across application domains has shown that sensor node tasks typically change over time, for instance, to vary sensed parameters, node duty cycles, or support debugging. Such reprogramming is accomplished through wireless communication using reprogrammable devices.

A node must detect if there is a different code item in the network, identify if it is newer, and update its code with minimal reprogramming cost, in terms of convergence speed and energy.

In a static network, sensor nodes run on batteries, leading to inconsistent behaviors, with periods of no operation followed by active periods. Radio communication is also known to be unreliable and dynamic with intermittent connectivity and link asymmetry. Sensor networks are becoming more dynamic due to mobility. Mobile sensor nodes might dynamically leave and join a network, requiring that they also vary their tasks dynamically.

A CCMP allows any node to discover if there is a (key, version) tuple in the network with the same key but a different version compared to its own tuple. A natural approach is to allow a node to keep querying its neighbors, to find if there is a new code item by comparing its own tuple to its neighbors tuples. Hundreds of dynamic nodes and hundreds of code items, a node should ask smart questions at the right time to discover if it needs an update. This requires careful CCMP design to reduce code update latency and to conserve both energy and bandwidth. In DRIP, a node randomly broadcasts each of its own (key, version) tuples to its neighbors separately. DRIP scales linearly with the total number of items (O(T)). DIP improves the total number of messages and latency to O(log(T)) by searching for a different (key, version) using hashes of the (key, version) tuples.

Design Of An Application-Cooperative Management System For Wireless Sensor Networks

It argues for the usefulness of an application-cooperative interactive management system for wireless sensor networks, and presents SNMS, a Sensor
Network Management System. SNMS is designed to be simple and have minimal impact on memory and network traffic, while remaining open and flexible. The system is evaluated in light of issues derived from real deployment experiences.

In these nodes would collect environmental readings every five minutes, and forward them through a multihop network to a base station located between the two trees. One month later, initial examination of the gathered data showed that the nodes in one tree had been entirely unable to contact the base station. Of the 33 remaining nodes, 15% returned no data. Of the 80 deployed nodes, 65% returned no data at all, from the very beginning. This problem was not detected on the day of the deployment for two reasons: the algorithm used to form the collection tree was designed to converge slowly, and with a 5 minute wait between opportunities to contact each node, the deployers were unable to adequately test the network in their allotted time. We suggest that the deployed application was incompatible with the needs of human managers, due to the tightly constrained power budget. However, we argue that every sensor network, running any application, should be able to present a human manager with the ability to quickly determine whether a deployed network is functioning. One week into the Sonoma deployment, another 15% of the nodes died, likely due to premature battery exhaustion caused by a time synchronization failure that prevented the node from entering sleep mode. But, this was later inferred from the collected data, and no records exist of the events that may have caused this failure. Authors of TinyOS components have the ability to check for these failure conditions arising from lower level systems, and combine this data to determine higher level failure conditions. But, this data is rarely available to a human manager. We argue that every sensor network should be able to record these events to permanent local storage for post-mortem analysis, and that this event record should also be accessible to a human manager in real time.

Data Discovery And Dissemination With Dip

A data discovery and dissemination protocol for wireless networks. Prior approaches, such as Trickle or SPIN, have overheads that scale linearly with the number of data items. For T items, DIP can identify new items with O(log(T)) packets while maintaining a O(1) detection latency. To achieve this performance in a wide spectrum of network configurations, DIP uses a hybrid approach of randomized scanning and tree-based directed searches. By dynamically selecting which of the two algorithms to use, DIP outperforms both in terms of transmissions and speed. Simulation and testbed experiments show that DIP sends 20-60% fewer packets than existing protocols and can be 200% faster, while only requiring O(log(log(T))) additional state per data item.

Reliable data dissemination is a basic building block for sensor network applications. Dissemination protocols reliably deliver data to every node in a network using key, version tuples on top of some variant of the Trickle algorithm. We describe these protocols and their algorithms in greater depth. The key characteristic they share is a node detects a neighbor needs an update by observing that the neighbor has a lower version number for a data item (key). The cost of this mechanism scales linearly with the number of data items: T data items require T version number announcements. Even though a protocol can typically put multiple announcements in a single packet, this is only a small constant factor improvement. Fundamentally, these algorithms scale with O(T) for T total data items. This linear factor introduces a basic cost/latency tradeoff. Nodes can either keep a constant detection latency and send O(T) packets, or keep a constant cost and have an O(T) latency. The key insight in this paper is that dissemination protocols can break this tradeoff by aggregating many data items into a single advertisement. Because these aggregates compress information, they can determine that an update is needed, but cannot always determine which data item needs an update. It outlines existing dissemination algorithms and describes a new algorithm, search, that breaks the cost/latency tradeoff, enabling fast and efficient dissemination.

By using a hash tree of data item version numbers, a protocol using search can discover an update is needed with O(log(T)) transmissions. In simple collision-free and lossless network models, search works well. However, two problems can make the hash tree algorithm perform poorly in real networks. First, packet losses can make it difficult to quickly traverse the tree. Second, the multiple advertisements caused by packet losses are completely redundant: there is typically only one subtree to explore. Through controlled simulation experiments, we find that in cases of very high loss or when a large fraction of items require updates, the underlying constant

factors can cause randomized scans to be more efficient than hash tree searches. It presents an analytical framework to understand these tradeoffs. The analysis shows that whether periodic advertisements or searches is more efficient depends on three factors: network density, packet loss ratios, and the percentage of items that need updates. While they have similar efficiency when reasonably close to their equality point, one can be a factor of two more efficient than the other at the edges.

This analysis indicates that a scalable dissemination protocol can get the best of both worlds by using a hybrid approach, dynamically switching between algorithms based on run-time conditions. It presents such a protocol, called DIP (Dissemination Protocol). DIP continuously measures network conditions and estimates whether each data item requires an updates. Based on this information, it dynamically chooses between a hash tree-based search approach and scoped randomized scanning. DIP improves searching performance by combining hashes over ranges of the key space with a bloom filter. Hashes allow it to detect whether there are version number inconsistencies while Bloom filters let it quickly pinpoint the source of the inconsistency.

It evaluates DIP in simulation and on a mote testbed. In simulated clique networks, DIP sends up to 30-50% fewer packets than either scanning or searching and is correspondingly 30-50% faster. In the Intel Mirage multihop 80 node testbed, DIP sends 60% fewer packets than scanning or searching. In some cases, DIP sends 85% fewer packets than scanning, the dominant algorithm in use today .By Improving its transmission efficiency, DIP is also able to disseminate faster: across real, multihop networks, DIP is 60% faster for a few items and over 200% faster for many items. It presents how DIP relates to prior work. These results show that DIP is significantly more efficient than existing approaches. This improvement comes at a cost of an additional log(log((T)) bits of state per data item for T items .It discusses the implications of these findings. The tradeoffs between scanning and searching touch on a basic tension in sensor net protocol design. While searching can find inconsistencies quickly by exchanging higher-level metadata, its deterministic operation means that it cannot leverage the communication redundancy inherent to wireless protocols. While scanning can take advantage of this redundancy through randomization, it does so by explicitly avoiding any complex metadata exchange. DIP's results suggest the complex tradeoffs between randomized versus deterministic algorithms in wireless networks deserve further study. This paper makes three research contributions. First, it proposes DIP, a n adaptive dissemination protocol that can scale to a large number of items. Second, it introduces using a bloom filter as an optimization to update detection mechanisms in dissemination protocols. Third, it evaluates DIP and shows it outperforms existing dissemination protocols, reducing transmission costs by 60% and latency by up to 40%. These results suggest the complex tradeoffs between randomized versus deterministic algorithms in lossy networks.

Trickle: A Selfregulating Algorithm For Code Propagation And Maintenance In Wireless Sensor Networks

In this Trickle, an algorithm for propagating and maintaining code updates in wireless sensor networks. Borrowing techniques from the epidemic/gossip, scalable multicast, and wireless broadcast literature, Trickle uses a "polite gossip" policy, where motes periodically broadcast a code summary to local neighbors but stay quiet if they have recently heard a summary identical to theirs. When a mote hears an older summary than its own, it broadcasts an update. Instead of flooding a network with packets, the algorithm controls the send rate so each mote hears a small trickle of packets, just enough to stay up to date. We show that with this simple mechanism, Trickle can scale to thousand-fold changes in network density, propagate new code in the order of seconds, and impose a maintenance cost on the order of a few sends an hour.

Composed of large numbers of small, resource constrained computing nodes ("motes"), sensor networks often must operate unattended for months or years. As requirements and environments evolve in lengthy deployments, users need to be able to introduce new code to re task a network. The scale and embedded nature of these systems – buried in bird burrows or collared on roving herds of zebras for months or years – requires network code propagation. Networking has a tremendous energy cost, however, and defines the system lifetime: laptops can be recharged, but sensor networks die. An effective reprogramming protocol must send few packets. While code is propagating, a network can be in a useless state due to there being multiple programs running concurrently. Transition time is wasted time, and wasted time is wasted energy. Therefore, an effective reprogramming protocol must also propagate new code quickly. The cost of transmitting new code must be measured against the duty cycle of an application. For some applications, sending a binary image (tens of kilobytes) can have the same cost as days of operation. In these applications, programs are 20-400 bytes long, a handful of packets. Wireless sensor networks may operate at a scale of hundreds, thousands, or more. Unlike Internet based systems, which represent a wide range of devices linked through a common network protocol, sensor networks are independent, application specific deployments. Symmetric links are common, and prior work has shown network behavior to often be worse indoors than out, predominantly due to multi-path effects. Motes come and go, due to temporary disconnections, failure, and network repopulation. As new code must eventually propagate to every mote in a network, but network membership is not static, propagation must be a continuous effort. Propagating code is costly; learning when to propagate code is even more so. Motes must periodically communicate to learn when there is new code.

To reduce energy costs, motes can transmit metadata to determine when code is needed. Even for binary images, this periodic metadata exchange overwhelms the cost of transmitting code when it is needed. Sending a full TinyDB binary image costs approximately the same as transmitting a forty byte metadata summary once a minute for a day. In Mat´e, Tiny Diffusion, Tiny DB, and similar systems, this tradeoff is even more pronounced: sending a few metadata packets costs the same as sending an entire program. The communication to learn when code is needed overwhelms the cost of actually propagating that code. The first step towards sensor network reprogramming, then, is an efficient algorithm for determining when motes should propagate code, which can be used to trigger the actual code transfer.

Efficient And Secure Source Authentication For Multicast

One of the main challenges of securing multicast communication is source authentication, or enabling receivers of multicast data to verify that the received data originated with the claimed source and was not modified enroute. The problem becomes more complex in common settings where other receivers of the data are not trusted, and where lost packets are not retransmitted. Several source authentication schemes for multicast have been suggested in the past, but none of these schemes is satisfactorily efficient in all prominent parameters. We recently proposed a very efficient scheme, TESLA, that is based on initial loose time synchronization between the sender and the receivers, followed by delayed release of keys by the sender. This paper proposes several substantial modifications and improvements to TESLA. One modification allows receivers to authenticate most packets as soon as they arrive (whereas TESLA requires buffering packets at the receiver side, and provides delayed authentication only). Other modifications improve the scalability of the scheme, reduce the space overhead for multiple instances, increase its resistance to denial-of-service attacks, and more.

With the growth and commercialization of the Internet, simultaneous transmission of data to multiple receivers becomes a prevalent mode of communication. Often the transmitted data is streamed and has considerable bandwidth. To avoid having to send the data separately to each receiver, several multicast routing protocols have been proposed and deployed, typically in the IP layer. The underlying principle of multicast communication is that each data packet sent from the source reaches a number of receivers. Securing multicast communication introduces a number of difficulties that are not encountered when trying to secure unicast communication. It is used for a taxonomy of multicast security concerns and some solutions. A major concern is source authentication, or allowing a receiver to ensure that the received data is authentic (i.e., it originates with the source and was not modified on the way), even when none of the other receivers of the data is trusted. Providing source authentication for multicast communication is the focus of this work. Simply deploying the standard point-to-point authentication mechanism (i.e appending a message authentication code to each packet, computed using a shared key) does not provide source authentication in the case of multicast. The problem is that any receiver that has the shared key can forge data and impersonate the sender. Consequently, it is natural to look for solutions based on asymmetric cryptography to prevent this attack, namely digital signature schemes. Indeed, signing each data packet provides good source authentication; however, it has high overhead, both in terms of time to sign and verify, and in terms of bandwidth. Several schemes were proposed that mitigate this overhead by amortizing a single signature over several packets. However, none of these schemes is fully satisfactory in terms of bandwidth and processing time, especially in a setting where the transmission is lossy and some data packets may never arrive.

Even though some schemes amortize a digital signature over multiple data packets, a serious denial-of-service attack is usually possible where an attacker floods the receiver with bogus packets supposedly containing a strong signature. Since signature verification is computationally expensive, the receiver is overwhelmed verifying the signatures. Another approach to providing source authentication uses only symmetric cryptography, more specifically on message authentication codes (MACs), and is based on delayed disclosure of keys by the sender. This technique was first used by Cheung in the context of authenticating communication among routers. It was then used in the Guy Fawkes protocol for interactive unicast communication .In articular, the TESLA scheme described in was presented to the reliable multicast transport (RMT) working group of the IETF and the secure multicast (SMuG) working group of the IRTF and was favorably received.

TESLA is particularly well suited to provide the source authentication un functionality for the MESP header , or for the ALC protocol proposed by the RMT. The main idea of TESLA, is to have the sender attach to each packet a MAC computed using a key k known only to itself. The receiver buffers the received packet without being able to authenticate it. If the packet is received too late, it is discarded. A short while later, the sender discloses k and the receiver is able to authenticate the packet. Consequently, a single MAC per packet suffices to provide source authentication, provided that the receiver has synchronized its clock with the sender ahead of time.

IMPLEMENTATION AND EXPERIMENTAL SETUP

We have written programs that execute the functions of the network owner, user and sensor node. The network owner and user side programs are C programs using OpenSSL [18] and running on laptop PCs (with 2 GB RAM) under Ubuntu 11.04 environment with different computational power. Also, the sensor node side programs are written in nesC and run on resource-limited motes (MicaZ and TelosB). The MicaZ mote features an 8-bit 8-MHz Atmel microcontroller with 4-kB

RAM, 128-kB ROM, and 512 kB of flash memory. Also, the TelosB mote has an 8-MHz CPU, 10-kB RAM, 48- kB ROM, 1MB of flash memory, and an 802.15.4/ZigBee radio. Our motes run TinyOS 2.x. Additionally, SHA-1 is used, and the key sizes of ECC are set to 128 bits, 160 and 192 bits, respectively. Throughout this paper, unless otherwise stated, all experiments on PCs (respectively, sensor nodes) were repeated 100,000 times (respectively, 1,000 times) for each measurement in order to obtain accurate average results. To implement DiDrip with the data hash chain method (rsp. the Merkle hash tree method), the following functionalities are added to the user side program of Drip: construction of data hash chain (rsp. Merkle hash tree) of a round of dissemination data, generation of the signature packet and all data packets. For obtaining version number of each data item, the DisseminatorC and DisseminatorP modules in the Drip nesC library has been modified to provide an interface called DisseminatorVersion. Moreover, the proposed hash tree method is implemented without and with using the message specific puzzle approach presented in Appendix, resulting in two implementations of DiDrip; DiDrip1 and DiDrip2. In DiDrip1, when a node receives a signature/data packet with a new version number, it authenticates the packet before broadcasting it to its next-hop neighbours. On the other hand, in DiDrip2, a node only checks the puzzle solution in the packet before broadcasting the packets.

Based on the design of DiDrip, we implement the verification function for signature and data packets based on the ECDSA verify function and SHA-1 hash function of TinyECC 2.0 library [19] and add them to the Drip nesC library. Also, in our experiment, when a network user (i.e., a laptop computer) disseminates data items, it first sends them to the serial port of a specific sensor node in the network which is referred to as repeater. Then, the repeater carries out the dissemination on behalf of the user using DiDrip Similar to [7], we use a circuit to accurately measure the power consumption of various cryptographic operations executed in a mote. The Tektronix TDS 3034C digital oscilloscope accurately measures the voltage Vr across the resistor. Denoting the battery voltage as Vb (which is 3 volts in our experiments), the voltage across the mote Vm is then Vb _ Vr. Once Vr is measured, the current through the circuit I can be obtained by using Ohm's law. The power consumed by the mote is then VmI. By also measuring the execution time of the cryptographic operation, we can obtain the energy consumption of the operation by multiplying the power
and execution time

CONCLUSION AND FUTURE WORK

In this paper, we have identified the security vulnerabilities in data discovery and dissemination when used in WSNs, which have not been addressed in previous research. Also, none of those approaches support distributed operation. Therefore, in this paper, a secure and distributed data discovery and dissemination protocol named DiDrip has been proposed. Besides analyzing the security of DiDrip, this paper has also reported the evaluation results of DiDrip in an experimental network of resource-limited sensor nodes, which shows that DiDrip is feasible in practice. We have also given a formal proof of the authenticity and integrity of the disseminated data items in DiDrip. Also, due to the open nature of wireless channels, messages can be easily intercepted. Thus, in the future work, we will consider how to ensure data confidentiality in the design of secure and distributed data discovery and dissemination protocols

REFERENCES

[1] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst., 2004, pp. 81–94.
[2] D. He, C. Chen, S. Chan, and J. Bu, "DiCode: DoS-resistant and distributed code dissemination in wireless sensor networks," IEEE Trans. Wireless Commun., vol. 11, no. 5, pp. 1946–1956, May 2012.
[3] T.Dang,N. Bulusu,W. Feng, and S. Park, "DHV:Acode consistency maintenance protocol for multi-hop wireless sensor networks," in Proc. 6th Eur. Conf.Wireless Sensor Netw., 2009, pp. 327–342.
[4] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in Proc. Eur. Conf. Wireless Sensor Netw., 2005, pp. 121–132.
[5] K. Lin and P. Levis, "Data discovery and dissemination with DIP," in Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw., 2008, pp. 433–444.
[6] M. Ceriotti, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment," in Proc. IEEE Int. Conf. Inf. Process. Sensor Netw., 2009, pp. 277–288.
[7] D. He, S. Chan, S. Tang, and M. Guizani, "Secure data discovery and dissemination based on hash tree for wireless sensor networks," IEEE Trans. Wireless Commun., vol. 12, no. 9, pp. 4638–4646, Sep. 2013

[8] M. Rahman, N. Nasser, and T. Taleb, "Pairing-based secure timing synchronization for heterogeneous sensor networks," in Proc. IEEE Global Telecommun. Conf., 2008, pp. 1–5.
[9] Geoss. [Online]. Available: http://www.epa.gov/geoss/
[10] NOPP. [Online]. Available: http://www.nopp.org/
[11] ORION. [Online]. Available: http://www.joiscience.org/ocean observing/advisors
[12] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks," in Proc. 1st Conf. Symp. Netw. Syst. Design Implementation, 2004, pp. 15–28.
[13] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," in Proc. Netw. Distrib. Syst. Security Symp., 2001, pp. 35–46.
[14] Y. Chen, I. Lin, C. Lei, and Y. Liao, "Broadcast authentication in sensor networks using compressed bloom filters," in Proc. 4th IEEE Int. Conf. Distrib. Comput. Sensor Syst., 2008, pp. 99–111.
[15] R. Merkle, "Protocols for public key cryptosystems," in Proc. IEEE Security Privacy, 1980, pp. 122–134.
[16] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," in Proc. Adv. Cryptology, 1997, pp. 56–73.
[17] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in Proc. IEEE Security Privacy, 2000, pp. 56–73.
[18] OpenSSL. [Online]. Available: http://www.openssl.org
[19] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic

curve cryptography in wireless sensor networks," in Proc. ACM/
IEEE Inf. Process. Sensor Netw., 2008, pp. 245–256.