The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)

CONFIDENTIAL AND SECURE BROADCAST IN WIRELESS SENSOR NETWORKS

Jaleel Shaheen, Diethelm Ostry
ICT Centre, CSIRO, Australia
Jay.Shaheen@csiro.au, Diet.Ostry@csiro.au

Vijay Sivaraman, Sanjay Jha
University of New South Wales, Sydney, Australia
vijay@unsw.edu.au, sjha@cse.unsw.edu.au

ABSTRACT

Wireless sensor networks need broadcast for operations such as software updates, network queries, and command dissemination. Alongside ensuring authenticity of the source and data, keeping the broadcast data secret is vital in certain applications such as battlefield control, emergency response, and natural resource management. In this paper we propose and prototype a mechanism for ensuring confidentiality and authenticity of broadcast data in single-hop networks, and discuss possible extensions to multi-hop settings. Our scheme uses known low-complexity symmetric encryption techniques for confidentiality, while changing the encryption key on a per-packet basis in a verifiable but non-forgeable way to ensure authenticity. Message integrity, freshness, and semantic security are also provided, and the broadcast data can be dynamic and incrementally processed. We incorporate our security scheme into Deluge, the *de facto* network programming protocol in TinyOS, and quantify the cost in terms of broadcast data transfer time and node memory space on a TelosB mote based platform.

I. INTRODUCTION

Broadcast is an essential feature in any sensor network for critical operations such as network query, software updates, time synchronisation, and network management. Given its importance, there is growing interest in addressing broadcast security [1]. A primary concern is the authenticity of the broadcast source and data: for example, in Deluge [2], the *de facto* network programming protocol distributed with TinyOS today, an arbitrary node can broadcast new versions of the software, disseminate malicious packets, program any number of nodes, and take over the operation of the entire network. Authenticating broadcasts is particularly challenging in wireless sensor networks due to the constrained resources available at sensor nodes, and has recently been posed as an open problem [3]. The symmetric authentication used in point-to-point communication based on a shared key between the two parties cannot be used in broadcast since the key would need to be shared by all parties, which is problematic since receivers should only be able to verify but not originate broadcasts. Asymmetric authentication mechanisms based on digital signatures are impractical on a per-packet basis due to their high computation and communication overheads. Recently, several proposals [4, 5, 6] have emerged to address the challenges in broadcast authentication, particularly in the context of network reprogramming.

Though authentication is arguably the most important security primitive for sensor network broadcasts, confidentiality is also vital in some critical applications, such as command and control signaling in the battlefield. The application that motivates this paper is one undertaken by our organisation, the Commonwealth Scientific and Industrial Research Organisation (CSIRO), to build a Water Resources Observation Network (WRON) [7] to assist in managing and controlling the national water resources of Australia. Confidentiality of various broadcast data and control messages is important in such a scenario. For example, the sensory parameters (such as sampling periods and theresholds) would from time-to-time be updated using broadcast mechanisms, and need to be kept private. Software upgrades should be confidential to prevent exploitation of potential bugs. This paper investigates the feasibility of securing broadcasts and keeping them confidential given the resource constraints at the sensor nodes. We are not aware of any prior proposals that ensure both confidentiality and authentication of broadcast traffic in wireless sensor networks.

Our contribution in this work is to propose and prototype a practical solution for ensuring confidentiality, authenticity, integrity, and freshness of broadcast data, primarily in the context of single-hop wireless sensor networks (extension to multiple hops is possible by sacrificing certain properties). Our scheme primarily uses symmetric key cryptography, shown to be feasible for efficient implementation on resource-constrained sensor nodes, but changes the key on a per-packet basis so that untrusted recipients cannot forge the broadcast data. The keys are derived from a one-way hash chain, whereby a key is verifiable but not forgeable from the previous key. Our scheme allows the broadcast data to be dynamic (i.e. need not be known in its entirety *a priori*), and ensures high message entropy by using a different encryption key for each packet. We prototype our mechanism in the context of the network reprogramming protocol Deluge included in TinyOS, and present experimental results on the associated time and space overheads in a TelosB mote based single-hop network.

The rest of this paper is organised as follows: Section II. defines the problem setting, solution requirements, and prior approaches from the literature. Our solution, and possible extensions, are discussed in Section III.. Section IV. describes our prototype and experimental results, while Section V. summarises our work and points to directions for future work.

II. PROBLEM OVERVIEW AND PRIOR SOLUTIONS

This section defines the operating environment and threat model, outlines the solution requirements, and discusses relevant prior work in wireless sensor network broadcast security.

A. Operating Environment and Threat Model

We assume a single-hop wireless sensor network in which a single source of broadcast data, called the base-station, can directly communicate with all sensor nodes (extension to multi-

ple hops will be discussed in Section C.). We assume that the base-station has abundant computation resources, and cannot be compromised. The base-station is also assumed to hold a public/private key pair, of which the public key is known to all nodes. Each sensor node is also assumed to be equipped with a public/private key pair, and the base-station knows the public key of each sensor node that is a target of the broadcast data. As we will see later, the public keys are required only during bootstrapping to establish initial trust; thereafter shared symmetric keys are used for data encryption.

If the application warrants confidentiality of the broadcast data, the sensor nodes are expected to be protected against physical compromise. The sensor nodes in our WRON project are expected either to be physically inaccessible to attackers (e.g. in secured areas), or hardened by incorporation of tamper-resistant hardware. If one or more nodes in the network are not compromise-resistant, confidentiality of the broadcasts is sacrificed, though our scheme will still ensure authenticity of all broadcasts.

We assume an underlying broadcast data delivery protocol that guarantees reliable in-order delivery of packets. For bulk operations such as software upgrades, broadcast protocols such as Deluge have built-in mechanisms to guarantee this. For mobile applications such as battlefield command and control, it is conceivable that a node is out of range for an extended period of time and then rejoins the network. At such time our scheme permits the node to re-establish synchrony (of keys) with the base-station by going through the bootstrap process.

The wireless medium is assumed to be insecure in that a passive eavesdropper can listen to all transmissions. An active intruder can transmit arbitrary messages, or replay a valid captured message at a later time. We make no assumptions about the number of intruders, their locations, their radio range, or the degree of collusion amongst them. In the case where nodes are not hardened against physical compromise, no assurances on data confidentiality can be given since an intruder can extract the cryptographic keys. Nevertheless, our scheme does assure authenticity of the broadcast source and data, under the assumption that the intruder does not have the capability to block reception of packets at an uncompromised node; such jamming effectively makes the network multi-hop, and requires enhancement of our solution as discussed later. Finally, we do not address denial-of-service or battery-drain attacks.

B. Solution Requirements

We seek a security mechanism that provides the following properties for the broadcast traffic:

- Confidentiality: The broadcast data should be kept secret from eavesdroppers. Confidentiality cannot be guaranteed if one or more nodes in the network are physically compromised.
- Authenticity: Messages not originating from the base-station should be discarded (ensuring source authenticity), as should messages that have been tampered with (ensuring data authenticity, also known as message integrity).
- Freshness: Packets that have been captured and replayed at a later time should be ignored by the sensor nodes.
- Semantic security: Even if the broadcast messages are chosen from a small finite set, the encryption should produce ciphertext that does not give information to an intruder on which of these messages was sent.
- Dynamic data: The sequence of broadcast messages need not be known before-hand in its entirety by the base-station.
- Delay Tolerance: No time synchronisation should be required in the system.
- Incremental processing: Each packet must be verifiable without having to wait for additional data.

C. Prior Proposals

We are not aware of any prior work that specifically addresses confidentiality of broadcast traffic in a wireless sensor network. However, we briefly mention some relevant schemes for unicast encryption and broadcast authentication.

TinySec [8] develops mechanisms for link-level encryption. It is however not strong enough for broadcast traffic since the encryption key would typically have a long lifetime and would need to be shared by all parties, which is problematic when receivers are untrusted and could use the key to generate broadcasts.

The μTESLA [4] protocol uses symmetric-key encryption with time-varying keys. The base-station constructs a key chain by repeatedly applying a hash function to an initial random value, and the root-key is distributed to each node securely based on a pre-distributed symmetric key. The chain construction allows nodes to verify that disclosed keys are authentic. Loose time-synchronisation of the network into regular time intervals is assumed, and the base-station uses a single key from the key chain for the whole duration of a time interval. The key is revealed by the base-station at a later time, when nodes can verify that the key is a valid member of the chain, the message authentication codes (MACs) of stored packets are correct, and that the time delay is such that only the base-station could have constructed the received pages.

Secured Deluge [5] adds security to the Deluge network programming protocol by using a hash-chain to verify authenticity of packets. The base-station sends the code update in a sequence of packets, each of which includes the hash of the next packet to be sent. A node receiving the broadcast stores this hash value, and compares it to the hash of the next page when it is received, allowing an immediate decision as to whether the page is authentic and in sequence. The intial packet is signed so the intial hash value is authenticated. Sluice [6] is very similar to Secured Deluge except that the hash in the chain is computed over pages rather than packets.

III. Our Approach

The heart of our mechanism lies in the use of a chain of keys, one key per packet, as described below.

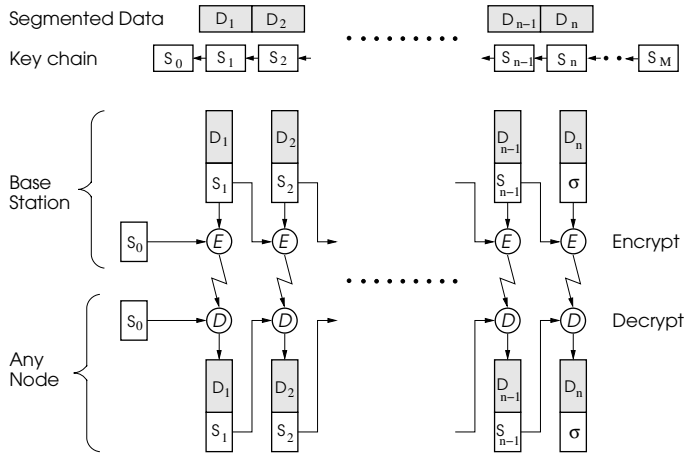

Figure 1: Key chain encryption of broadcast packets

A. *The Procedure*

The process by which our security mechanism works for a single-hop system is depicted in Fig. 1, and described by the steps below:

1. **Key-chain generation:** The base-station (BS) selects an arbitrary random number s_M, and from it generates a key chain s_M, s_{M-1}, ..., s_1, s_0, where s_{i-1} is a hash (SHA1 or MD5) of s_i for $i = 1, \ldots, M$ (see Fig. 1). Further, the length M of the chain can be arbitrary but no less than the number of packets n that the base-stations wants to broadcast (in this transaction).

2. **Bootstrapping:** The "root-key" s_0 must be securely conveyed to each target sensor node. The root-key could be programmed into the sensor nodes prior to deployment (if the key-chain in the previous step is long enough to be used for the lifetime of the node), or one of several key management schemes [9] can generate dynamic keys for secure distribution of the root key to all nodes. We have chosen to use an authenticated Diffie-Hellman exchange to generate a secure channel for the root-key transfer. The Diffie-Hellman exchange generates a symmetric shared key between the base-station and a particular node; the base-station uses this shared key to encrypt s_0 for transmission to that sensor node. The dynamic nature of the Diffie-Hellman exchange precludes capture-and-replay attacks, while authentication via digital signatures (using elliptic curve methods, discussed later) prevents an intruder from masquerading as the base-station or as a sensor node, and protects against man-in-the-middle attacks.

3. **Data transmission:** Once all target sensor nodes have the root key, the base-station creates a packet by concatenating the broadcast data and the successor key s_1, and encrypts the message with a symmetric encryption technique using key s_0 (see Fig. 1). The encryption scheme must ensure that modification of the encrypted data also destroys the key, i.e. that the encrypted data and encrypted key are not separable in ciphertext. Such message integrity is guaranteed, for example, by the offset code-book (OCB) mode [10] of block cipher encryption. The encrypted packet is then broadcast to all nodes.

4. **Data reception:** A receiver sensor node can decrypt the message using key s_0 to reveal the data as well as the successor key s_1. If the key s_1 hashes to s_0, authenticity and integrity the packet is assured (by use of an appropriate block cipher mode) and the packet is accepted (see Fig. 1). The key s_0 is now discarded by the node and the new key s_1 stored in its place.

5. **Iterate:** Steps 3 and 4 are repeated for successive packets, using key s_{i-1} in lieu of s_0, and s_i in lieu of s_1, till all broadcast packets have been successfully received by all target nodes. Care must be taken that successive packets are transmitted at a rate which gives nodes sufficient time to extract the data payload and prepare for the next packet.

B. *Discussion*

Our use of a key chain is similar to the scheme used by μTESLA [4], with some differences: our scheme uses the keys for encrypting data, while μTESLA uses the keys for computing message authentication codes (MACs) to validate the data. μTESLA discloses keys some time after the data has been transmitted (requiring storage of packets), whereas we send the key to decrypt a packet in the preceding packet. Lastly, unlike our scheme, μTESLA uses network-wide loose time-synchronisation, with a single key from the key chain being used for the whole duration of a time interval, while our scheme changes the key from packet to packet.

The schemes proposed in [5, 6] hash chain the data for authentication. We do not include hash chains for two reasons: first, construction of the hash chain requires *a priori* knowledge of the sequence of broadcast data, which may not be available in dynamic situations. Second, though our key chain is constructed independently of the data, it does, in a single-hop network, authenticate the entire packet, since the successor key in the packet is non-forgeable.

A feature common to all the above broadcast security protocols is an initial commitment step: the signed first packet in [5] or first page in [6] commits to a data hash chain, while the root-key in μTESLA and our scheme commits to a key chain. Confidentiality of the broadcast data requires the initial commitment to be transmitted secretly by the base-station to each target node individually. While this is computationally expensive, we note that this operation can be overlapped in nodes so that for large networks, the time to carry out the initialisation step is limited by communication time requirements rather than the computational load.

Our choice of authenticated Diffie-Hellman for the bootstrapping phase (communicating the root-key to each node) is based on its dynamic nature (i.e., choice of different random number in each attempt) that protects against replay attacks. A simple asymmetric encryption of the root-key by the base-station using the target node's public key is susceptible to capture and replay by an adversary at a later time. Authenticated Diffie-Hellman exchange is implemented via elliptic-

curve public-key cryptographic operations, which have been show to be feasible for resource-constrained sensor nodes [11].

C. Extension to Multiple Hops

In a single-hop scenario in which all nodes can directly hear all base-station transmissions, our scheme assures authenticity of the broadcast message by entangling the validity of the data with the validity of the key. However, in a multi-hop network, a compromised transit node, instead of relaying the messages it receives, can disentangle the keys from the data, use these valid keys to encrypt malicious data, and forward these packets into the remaining network. Though there are several ways in which our scheme can be extended to ensure authenticity in multi-hop networks with compromised nodes, none seems to preserve all desired properties listed in section B..

One approach is to use a data hash-chain similar to [5, 6], whereby the hash of the next packet is included in each packet before encryption. In fact the hash might even be used as the encryption key. However, such an approach requires all the broadcast data to be know *a priori*, which is not suitable for applications in which the data is dynamic.

Another approach is to ensure that a broadcast packet has been received by all nodes before the next one is transmitted by the base-station (say by having the underlying broadcast mechanism provide packet-by-packet reliability) or by withholding transmission of the next packet for a fixed period of time, at which time the key contained in the current packet loses validity. Unfortunately, this approach requires some form of network-wide synchronisation, similar to μTESLA [4]. A more thorough exploration of multi-hop extensions is deferred to future work.

IV. Implementation and Experimentation

We prototyped our security scheme as an extension to the Deluge network programming protocol [2] that is distributed with TinyOS. Deluge uses broadcast to efficiently distribute a new program to multiple target sensor nodes. Internally it divides the program into pages of 1104 bytes each; a page is transmitted in 48 packets, each containing 23 bytes of data payload. A target sensor node upon successful receipt of a page stores it in flash memory. Deluge works over multiple hops, and guarantees reliable in-order delivery of packets. It however lacks in-built security mechanisms, which we implement using the scheme proposed in this paper.

The security scheme in this work was implemented using industry standard cryptographic primitives and parameters. We set the symmetric key size to 8 bytes, to be compatible with the RC5 encryption algorithm implementation in TinySec [8]. For ease of implementation we chose to use the available RC5 in TinySec even though its CFB (cipher feedback) mode does not sufficiently protect against tampering of the encrypted data. A more sophisticated block cipher scheme like OCB mode which does offer tamper protection has comparable time complexity [10], and will be investigated in our future implementations. Our key chain was generated by hashing the key using the SHA1 algorithm. Of the 20-byte value generated by SHA1, only the lower 8 bytes were used as the RC5 encryption key.

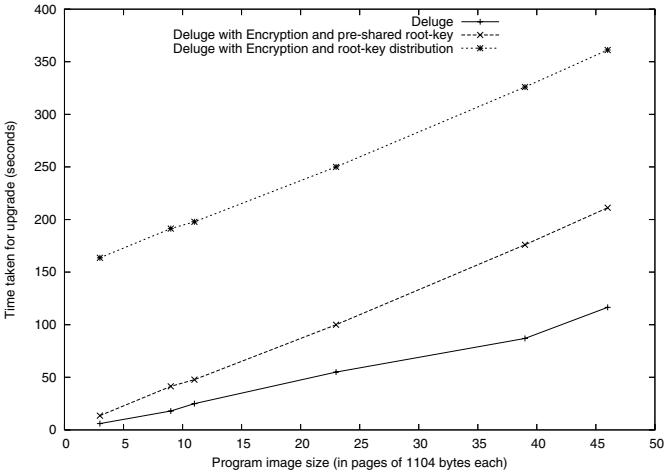

Figure 2: Transfer time from Base-Station to one node

The packet structure of Deluge was modified so that in addition to the 23 bytes of data, 8 bytes of key was included corresponding to the successor key in the key chain, and the packet was encrypted with the current key using RC5. The additional 8 bytes per packet constitutes an overhead of 384 bytes per page.

Our platform for testing comprised a PC (running the *cygwin* environment on Windows XP) acting as a base-station and TelosB motes (commerically available from Crossbow Technology Inc.) as target sensor nodes. The base-station was implemented in Java using the BouncyCastle JCE provider [12].

The bootstrap phase of our security scheme involves establishing the initial shared key (the root-key) between the base-station and each target node. We implemented an authenticated Diffie-Hellman key exchange to securely share the root-key, using primitives from the TinyECC package [13] for implementing EC-DSA (Elliptic Curve Digital Signature Algorithm), and the EccM package [11] for the ECDH (Elliptic Curve Diffie-Hellman). We note that at the moment we have not optimised our cryptographic operations for performance. Currently our code requires 42 KB of ROM space (which is around 22 KB more than raw Deluge) and 4 KB of RAM for data storage (around 3 KB more than raw Deluge), and these fit comfortably within the memory available in the TelosB motes (48 KB of ROM program space, 10 KB of data storage, and 1 MB of flash storage).

In our first experiment, we profile the time taken for upgrading a single node with a new image using Deluge and our security-enhanced version of Deluge. Fig. 2 shows the upgrade time for six program images (supplied as samples in the TinyOS distribution) of varying sizes: Blink (6 KB), Oscilloscope (9 KB), Pong (11 KB), TinyECC (23 KB), secure-Deluge (39 KB), and a dummy program (46 KB). Each experiment was performed several times and found to have consistent timing results, so error-bars are not shown. The bottom curve shows the time taken for the upgrade when original Deluge is used, namely without any security mechanisms. The curve above it corresponds to our scheme whereby the data is encrypted using the key-chain but the root-key has been pre-shared. It shows that the time required grows proprotionally to the image size,

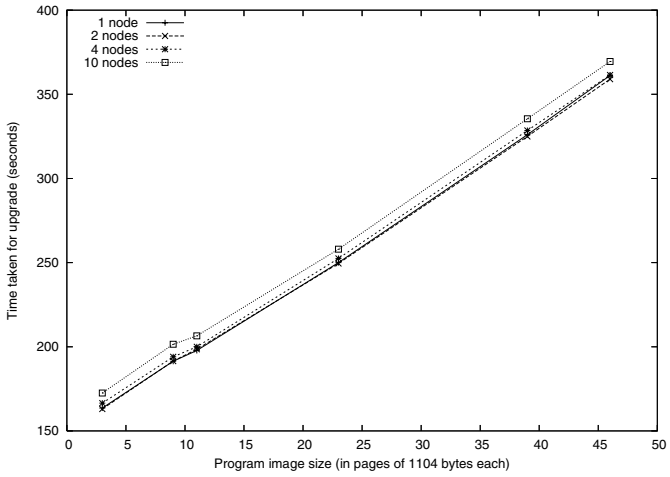

Figure 3: Transfer time from Base-Station to multiple nodes

but stays within twice that required by Deluge, an acceptable price to pay. The top curve shows the time taken for the upgrade including the initial dynamic root-key exchange phase. This phase adds approximately 150 seconds to each image upgrade operation, and as expected, is independent of the size of the code image.

In our second experiment we measure the time taken by our security-enhances Deluge to upgrade multiple nodes (note that the vertical scale in this plot starts at 150 seconds). Fig. 3 shows that upgrading each additional node incurs a small additional cost, emphasising that the advantages of broadcast are retained by our security mechanism.

V. Conclusions and Future Work

Critical wireless sensor networks deployed in defence and resource management applications require confidentiality of broadcast traffic in addition to protection against attacks. In this paper we develop efficient mechanisms to meet these stringent requirements in a single-hop setting. Our scheme uses low-complexity symmetric-key cryptographic techniques to guarantee confidentiality, authenticity, freshness, and semantic security of data, while allowing it to be dynamic and incrementally processed. We implemented a prototype of our mechanism as an add-on to the Deluge network reprogramming protocol in off-the-shelf TelosB motes running TinyOS. Our experiments show that the time required to broadcast an image confidentially and securely to multiple nodes using our scheme is no more than twice needed by the standard Deluge, with an additional 150 seconds for the initial bootstrap phase. We believe this is an acceptable price to pay for ensuring confidentiality and security of wireless sensor network broadcasts.

Our future work will address the multi-hop scenario where nodes may be compromised. We also intend to evaluate alternative methods for the root-key establishment.

VI. Acknowledgement

The authors would like to thank Dr. Peter Lamb of CSIRO for his valuable suggestions on an earlier draft of this paper.

References

[1] A. Perrig and J. D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2002.

[2] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *ACM SenSys*, Baltimore, MD, Nov 2004, pp. 81–94.

[3] M. Luk, A. Perrig, and B. Whillock, "Seven Cardinal Properties of Sensor Network Broadcast Authentication," in *ACM SASN*, Alexandria, VA, Oct 2006, pp. 147–156.

[4] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[5] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the Deluge Network Programming System," in *Proc. Intl. Conf. Inf. Proc. Sensor Networks (IPSN'06)*, Nashville, TN, Apr 2006.

[6] P. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure Dissemination of Code Updates in Sensor Networks," in *Proc. Intl. Conf. Distributed Computing Systems (ICDCS)*, Lisboa, Portugal, Jul 2006.

[7] "Water Resources Observation Network," CSIRO news [Online]. Available: *http://www.csiro.au/news/newsletters/0604_water/ecos.pdf*

[8] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," in *ACM SenSys*, Baltimore, MD, Nov 2004, pp. 162–175.

[9] Y. Jiang, C. Lin, M. Shi, and X. Shen, *Security in Sensor Networks*. Auerbach Publications, 2007, ch. Key Management Schemes for Wireless Sensor Networks, pp. 113–143.

[10] P. Rogaway, M. Bellare, and J. Black, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," *ACM Trans. Information and System Security*, vol. 6, no. 3, pp. 365–403, Aug 2003.

[11] D. J. Malan, M. Welsh, and M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography," in *IEEE SECON*, Santa Clara, CA, Oct 2004, pp. 71–80.

[12] Bouncy-Castle, "Cryptographic APIs," [Online]. Available: *http://www.bouncycastle.org/*

[13] Cyber-Defense-Laboratory, "TinyECC: Elliptic Curve Cryptography for Sensor Networks,", 2007 [Online]. Available: *http://discovery.csc.ncsu.edu/software/TinyECC/*