Securing Network Access in Wireless Sensor Networks*

Kun Sun[§] An Liu[†] Roger Xu[§] Peng Ning[†] Douglas Maughan[‡]

[§] Intelligent Automation Inc.
15400 Calhoun Dr. #400
Rockville, MD, 20855
{ksun, hgxu}@i-a-i.com

[†] Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
{aliu3, pning}@ncsu.edu

[‡] Science & Technology Directorate
Department of Homeland Security
douglas.maughan@dhs.gov

ABSTRACT

In wireless sensor networks, it is critical to restrict the network access only to eligible sensor nodes, while messages from outsiders will not be forwarded in the networks. In this paper, we present the design, implementation, and evaluation of a secure network access system for wireless sensor networks. This paper makes three contributions: First, it develops a network admission control subsystem using Elliptic Curve public key cryptosystem to add new sensor nodes into a sensor network. The admission control subsystem employs a polynomial-based weak authentication scheme to mitigate Denial of Service (DoS) attacks against the public key cryptographic operations. Second, it implements an interface in TinyOS to provide symmetric key cryptography using the hardware security support in IEEE 802.15.4 radio components (e.g., CC2420). The hardware security can satisfy both message authentication and timely delivery requirements in real-time applications. The third contribution is an implementation of a stateless group key update scheme to update a network-wide secret key in a sensor network. We implement all the proposed techniques on Imote2 sensor platform running TinyOS and conduct an evaluation through field experiments.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General-Security and protection

General Terms

Security, Design, Algorithms

Keywords

Sensor Networks, Security, Elliptic Curve Cryptography

1. INTRODUCTION

Wireless sensor networks are comprised of low cost sensor nodes that have the ability to quickly form a mesh network and communicate with each other through digital radios. Recent advances in wireless sensor networks can drastically reduce the cost and physical limitation in wired network. Therefore, wireless sensor networks have been widely used in applications such as process control system, condition monitoring, health and safety monitoring.

Because it is difficult to define and control boundaries and interactions between sensor nodes in wireless sensor networks, as wireless sensor networks are integrated into some critical infrastructures to promote connectivity and remote access capabilities, the possibility of cyber security vulnerabilities and incidents increases significantly. Threats can come from numerous sources, including hostile governments, terrorist groups, malicious intruders, complexities, accidents, natural disasters as well as malicious or accidental actions by insiders.

In wireless sensor networks, it is critical to restrict the network access only to eligible sensor nodes, while messages from outsiders should not be forwarded in the networks. Moreover, outsiders cannot eavesdrop, modify or forge packets from eligible nodes inside the sensor network. Since sensor nodes are highly constrained in terms of resources, satisfying the security protocols in an efficient way (using less energy, computational time and memory space) without sacrificing the strength of their security properties is one of the major challenges.

In this paper, we develop a secure network access system in wireless sensor networks to control the network access to eligible nodes by providing node authentication, packet authentication, packet integrity, and packet confidentiality using standardized cryptosystems. The system works in three stages. In the **Network Admission Control** stage, when a new sensor node is added into an existing sensor network, to communicate with another node already in the network, the two nodes perform a two-way authentication and generate a pairwise secret key using a Self-Certified Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol [6]. We choose Elliptic Curve Cryptography (ECC) because it is more suitable for resource-constrained devices due to its comparable security with much shorter key lengths and less memory requirement. For example, 160-bit ECC can provide the same security as 80-bit symmetric and 1024-bit RSA. Because ECC may suffer from the Denial of Service (DoS) attack, especially on the resource-constrained sensor nodes, we develop a polynomial-based weak authentication scheme to mitigate potential DoS attacks against the Self-Certified ECDH protocol.

For resource constrained sensor nodes, it is expensive to use public key cryptography to secure all the message communications. Moreover, many real-time applications require emergency messages be delivered in a timely manner. The computation delay of the software solutions (e.g., TinySec [13]) is too large and unacceptable for many real-time applications. To solve the above issues, in the **Network Access Control** stage, we

* This work is support by the Department of Homeland Security under grant NBCHC080061.

implement an interface in TinyOS to provide symmetric key cryptography using the hardware security support in IEEE 802.15.4 radio components (e.g., CC2420 [4]). Then, we enforce secure and efficient network access by employing a network-wide secret key, which is only known by eligible nodes, to authenticate all the packets transmitted in the network.

Because all the eligible nodes share a network-wide secret key, when one node is compromised, we must update the secret key among all the remaining eligible nodes. This is essentially a group key update problem. Based on the interdependency of key update messages, group key update schemes can be classified into either stateful ones or stateless ones. In a stateful scheme (e.g., [21]), a legitimate node's state (fail/success) in the current round of group key update will affect its ability to decrypt future group keys. In contrast, the group key update in a stateless scheme is only based on the current group key update message and the node's initial configuration. This property makes stateless group key distribution very useful in situations where some nodes are not constantly on-line, or experience burst packet losses. In the **Network Access Maintenance** stage, to update the network-wide secret key, we extend the stateless group key update schemes in [8] with a packet retransmission scheme, in which a node can recover the current group key without waiting until receiving the next round of key update packets.

After implementing all three stages of the secure access system and the supporting algorithms on Imote2 sensor platform [1] in an experimental sensor network, we evaluate the system performance in terms of code size, computation time, and energy consumption. Through these analyses, we show that our system is a secure and efficient solution to control the network access in wireless sensor networks.

The rest of the paper is organized as follows. Section 2 describes the design of the three stages in the system. Section 3 discusses a few implementation issues. Section 4 provides the security and performance analysis of the system in a small sensor network. Section 5 concludes this paper and points out future research directions.

2. Overview of Proposed System

In the three stages of the secure access control system, the Network Admission Control stage is responsible for adding new eligible nodes into the system; the Network Access Control stage is to guarantee that all traffic in the system is authenticated; and the Network Access Maintenance stage is to revoke compromised nodes, update the group key, and retransmit group keys for out-of-sync nodes.

2.1 Network Admission Control

We use Self-Certified ECDH protocol [6] to establish a pairwise key between a new sensor node and a controller node that is already in the network. The controller node can be a regular sensor node or a more powerful node. Then we use this pairwise key to distribute the current group key used in the network from the controller node to the new node. To avoid possible DoS attacks against the Self-Certified ECDH protocol, we develop a polynomial-based weak authentication scheme [7] in this stage.

As shown in Figure 1, before a new node joins the group, a Certificate Authority (CA) first deploys some keying material onto the sensor node. The controller of the group will broadcast their ID periodically. The new node will listen and choose a controller node with the strongest signal to launch a two-way authentication and establish a pairwise key using Self-Certified ECDH protocol. After establishing a secret key, the controller node uses a push-based method to distribute the current group key to the new node.

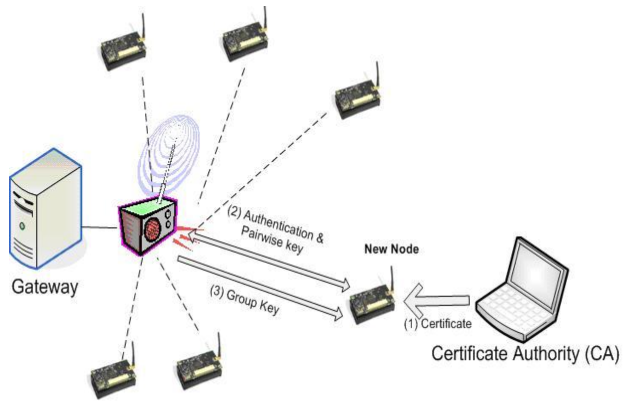

Figure 1: A new node joins the network.

2.1.1 *Self-Certified ECDH*

Self-Certified ECDH is used to facilitate authentication and symmetric key establishment between the new node and the controller node. Compared with ECDH, Self-Certified ECDH protocol [6] only requires two parties exchange their IDs and public keys instead of exchanging IDs, public keys and certificates.

We include two algorithms in Self-Certified ECDH: *Fixed Key* algorithm and *Ephemeral Key* algorithm. In the Fixed Key algorithm, two parties will always establish the same pairwise key if they run this algorithm multiple times. In the Ephemeral Key algorithm, by introducing a new random value, two parties will generate different pairwise keys if they run the algorithm multiple times.

There are three components in this protocol: Certificate Authority (CA), controller node, and regular sensor node. The Certificate Authority generates and distributes private keys and related parameters to all nodes in the network, including controller nodes and regular sensor nodes. Each node in the network will first be initialized by communicating with the CA and receiving the keying materials. A controller node broadcasts its ID periodically, waits for incoming requests from newly deployed sensor nodes, and then runs the Self-Certified ECDH protocol. A newly deployed sensor node sends a request message to a controller node to run the Self-Certified ECDH protocol as the initiator. At the end of the protocol, the sensor node and the controller node establish a common pairwise secret key.

The Self-Certified ECDH protocol is vulnerable to Denial of Service (DoS) attacks due to the expensive scalar point multiplication operation. An attacker may keep sending fake IDs and fake public keys to make controller/sensor nodes busy calculating the secret keys. The underlying reason is that there is no lightweight authentication on exchanged messages in Self-Certified ECDH protocol. To defeat DoS attacks against Self-Certified ECDH protocol, we propose a polynomial-based weak authentication scheme.

2.1.2 Polynomial-based Weak Authentication

The basic idea of polynomial-based weak authentication [7] is shown in Figure 2. The Certificate Authority (CA) first generates a bivariate *t*-degree polynomial f over a finite field GF(*q*), where *q* is a large prime number. Function *f* satisfies the following property: *f(x,y) = f(y,x)*.

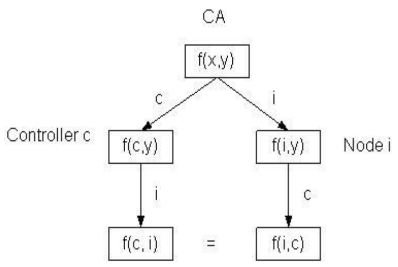

Figure 2: Polynomial-based weak authentication

As shown in Figure 2, for controller c, CA evaluates *x* in the bivariate polynomial *f(x,y)* by *c*, and deploys *f(c,y)* onto *c*. For regular node *i*, CA evaluates *x* in *f(x,y)* by *i*, and deploys *f(i,y)* onto *i*. When node *i* and controller c want to communicate with each other, they can establish a pairwise key based on each other's ID since *f(c,i)* = *f(i,c)*. Then they can use key *f(i,c)* to authenticate the exchanged messages (i.e., ID, U, EV) in Self-Certified ECDH. Evaluation of the polynomial is much faster than scalar point multiplication in Self-Certified ECDH. Both nodes verify the messages before running Self-Certified ECDH to mitigate the DoS attack. Our scheme is *t*-collusion resistant, which means once *t+1* sensor nodes are compromised, the secret polynomial *f* is disclosed. Therefore, it can only provide weak authentication, and cannot replace Self-Certified ECDH to establish secret keys.

2.2 Network Access Control

All nodes in the group will use the same group key to protect packets transmitted in wireless sensor networks. On the sender side, the sending node generates a message integrity code (MIC) for each outgoing packet using the group key. On the receiver side, the receiving node uses the group key to verify the MIC included in each incoming packet. If the MIC can be verified, the receiver forwards the received packet up in the radio stack. Otherwise, the receiver simply discards the packet.

The software implementation of encryption, decryption, and MIC generation and verification (e.g., TinySec [13]) will incur a large delay in packet transmission in the wireless sensor networks. To minimize such delay, we exploit the hardware security support in IEEE 802.15.4 radio component (e.g. CC2420 RF chip [4]), which provides 128-bit AES encryption/decryption. CC2420 features hardware security with two types of operations: stand-alone encryption operation and in-line security operation. The stand-alone encryption operation provides a plain AES encryption, with 128 bit plaintext and 128 bit keys. To encrypt a plaintext, a node first writes the plaintext to the stand-alone buffer SABUF, and then issues a SAES command to initiate the encryption operation. When the encryption is complete, the cipher-text is written back to the stand-alone buffer, overwriting the plaintext.

The in-line security operation can provide encryption, decryption, and authentication on frames within the receive buffer (RXFIFO) and the transmit buffer (TXFIFO) of CC2420 on a frame basis. It supports three modes of security: counter mode (CTR), CBC-MAC mode, and CCM mode. CTR mode performs encryption on the outgoing frames in the TXFIFO buffer, and performs decryption on the incoming frames in the RXFIFO buffer. CBC-MAC mode can generate and verify the MIC of the messages. The length of MIC is variable with even values between 4 bytes and 16 bytes. CCM mode combines CTR mode encryption and CBC-MIC authentication in one operation. All the three security modes are based on AES encryption/decryption using 128 bit keys.

2.3 Network Access Maintenance

When some nodes are compromised, a key manager must update the group key to revoke the compromised nodes from the network. We design and implement a stateless group key update scheme to update the group key in the network. A stateless group key update scheme can guarantee a legitimate node to get the more recent group key as long as the node receives the corresponding key update message, even if the node is offline for a while, or misses several previous rounds of key updates.

In the stateless group key update schemes, each sensor node is pre-assigned a unique ID and some personal secret keys that never change during the lifetime of the group. To revoke a node or to update the group key, the key manager encrypts a new group key separately, using a set of secret keys only known to the non-revoked nodes. The manager creates a key update message consisting of the resulting cipher-texts and some auxiliary information (e.g., the IDs of the encryption keys), and then broadcasts this message to the network. After receiving a key update message, a non-revoked node uses its personal secret key (or a key derived from its personal secret key) to decrypt a certain part of the message (indicated by the node ID), and obtains the new group key.

2.3.1 Stateless Group Key Update

We develop two stateless group key update schemes. In the basic stateless group key update scheme, controller c uses the pair-wise secret key $K_{c,i}$ shared with each non-revoked node *i* to encrypt the new group key. This scheme works well for small groups, but cannot scale to large groups, due to the high communication and computation overheads on the controller node. It requires performing *n-r* AES encryption operations and sending *n-r* messages, where *n* is the total number of group members, *r* is the number of the revoked nodes.

To reduce the communication and computation overheads in the basic scheme, we adopt Subset Difference Method (SDM) [8] in our system. SDM is a type of Subset-Cover algorithm. Given *r* revoked nodes in a total of *n* nodes, the non-revoked receivers will be partitioned into at most 2*r*-1 subsets (or 1.25*r* on average). SDM consists of two major algorithms: subset finding algorithm and key assignment algorithm. The subset finding algorithm partitions non-revoked nodes into disjointed groups, and the key assignment algorithm guarantees only the non-revoked nodes can derive the key and then decrypt the new group key.

The basic idea of SDM is that, whenever we want to revoke a set of compromised nodes, we can always find subset covers that only cover eligible nodes in the group. For example, we want to revoke two compromised nodes (node 4 and node 6) in Figure 3, where the labels of node 4 and node 6 in the tree are 11 and 13, respectively. Subset $S_{2,11}$ covers nodes 1, 2, 3, but not node 4; subset $S_{3,13}$ covers nodes 5, 7, 8, but not node 6. If we want to revoke node 4 and node 6, we encrypt new group key using subset key of $S_{2,11}$ and $S_{3,13}$ and flood this message through the network. Only non-revoked nodes have the keying materials to decrypt the message and obtain the new group key.

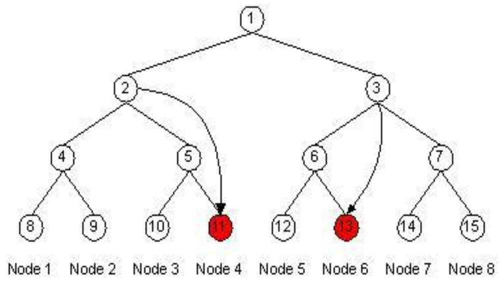

Figure 3: SDM tree

In the original SDM scheme, a group key update message may not fit into one packet when the number of revoked node *t* becomes large, so a message may be divided into multiple packets for delivery. We must guarantee all the packets for one message can be correctly received by sensor nodes. However, this message dissemination faces threats from both external attackers and potentially compromised nodes. For example, the adversary may attempt to modify or replace one or more group key update packets being propagated to sensor nodes. As another example, the adversary may inject bogus group key updated packets and force normal sensor nodes to verify and/or forward them, thus exhausting their limited battery power.

We develop a secure message dissemination mechanism to provide security protection for packet distribution, including the integrity protection of message and resistance to the potential DoS attacks. The key issue is to provide immediate authentication of the multiple packets for a large group key update message. We arrange packets and their hash images using Merkle Hash Tree [17] to provide immediate authentication of the packets.

(a) Construction of packets

Figure 4 illustrates our authentication scheme for the group key update message. We split each group key update message into N fixed-size packets, denoted as Pkt_1 through Pkt_N. We use Merkle hash tree [17] to facilitate the authentication of the hash images of the packets. Specifically, we calculate the hash images of each packet to have $V_i = H(Pkt_i)$, $(i = 1, 2, ...,N)$, and construct a Merkle hash tree using V_1, V_2, ..., and V_N as leaf nodes. Figure 4 shows the construction of the Merkle hash tree when N = 8. We compute $e_i = H(V_i)$ $(i = 1, 2, ...,N)$, and build a binary tree by computing internal nodes from adjacent children nodes. Each internal node is the hash image of the two children nodes. For example, $e_{1-2} = H(e_1||e_2)$, and $e_{1-4} = H(e_{1-2}||e_{3-4})$.

We then construct N packets using this Merkle hash tree. Specifically, we construct one packet for each V_i, where i = 1, 2, …, N; each packet consists of packet Pkt_i and the values in V_i's authentication path (i.e., the siblings of the nodes in the path from Vi to the root) in the Merkle hash tree. For example, a first packet consists of Pkt_1, e_2, e_{3-4}, and e_{5-8} in Figure 4. We include the root of the Merkle hash tree and a signature over all of them in a signature packet.

(b) Transmission & Authentication of packets

Our packet construction provides the capability of immediate packet authentication on receivers. This property is critical for sensor nodes to prevent DoS attacks aimed at exhausting receivers' buffers.

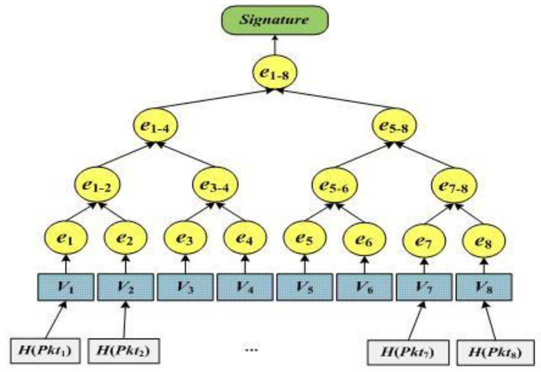

Figure 4. The Merkle hash tree constructed for 8 Packets

The controller node first broadcasts the signature packet, which serves as the advertisement of the new group key update message. Upon receiving a signature packet, each node verifies the signature to authenticate the root of the Merkle hash tree. This root allows the node to authenticate each packet upon receipt, using the values in the authentication path included in the same packet. For example, in Figure 4, if e_{1-8} has been authenticated in the signature packet, upon receiving a packet consisting of Pkt_1, e_2, e_{3-4}, and e_{5-8}, a node can immediately verify whether $H(H(H(H(Pkt_1)||e_2)||e_{3-4})||e_{5-8}) = e_{1-8}$. If the answer is yes, the received packet is accepted; otherwise, it must be a forged packet and should be discarded right away.

2.3.2 *Group Key in One-way Key Chain*

We further customize the stateless group key update schemes for managing network access control keys. This customization is based on the observation that network access control keys are intended for authenticating wireless packets only. As a result, backward secrecy is not a concern. In other words, once network access control keys are updated, the secrecy of previous network access keys is not necessary any more. Based on this observation, we organize the network access control keys in a one-way key chain to facilitate the authentication of future keys based on previous ones.

$$k_0 \xleftarrow{H} k_1 \xleftarrow{H} \quad \cdots \quad \xleftarrow{H} k_{n-1} \xleftarrow{H} k_n$$

Figure 5. Organize Keys into one-way key chain

A one-way key chain is a chain of keys generated through repeatedly applying a one way hash function H on a random number (key seed). For instance, $k_{n-1} = H(k_n)$, ..., $k_0 = H(k_1)$. The property of one-way means, given a latest released key k_i from a one way key chain, it is computationally infeasible for an adversary to find any unreleased key k_j such that $H_{j-i}(k_j)$ equals k_i. However, it allows a receiver to easily verify that a later key k_x really belongs to the key chain by checking that $H_{x-i}(k_x)$ equals k_i. Using the one-way key chain to organize the network access control keys, authentication of later keys based on a previous one can be trivially performed at each node by hashing a newly distributed key a few times and verify if it matches a previously known one. Figure 5 illustrates this approach.

2.3.3 Packet Retransmission Scheme

Some nodes may not receive the group key update message due to the limited transmission range and topology change. We call these nodes using old group key as out-of-sync nodes. When a node is out-of-sync, it cannot access the network until the next round of group key update, since all its packets won't have the right MIC.

To bring the out-of-sync nodes back to the network, we exploit the stateless feature of the SDM scheme we are using. Each node buffers the most recent key update message it has received, and it can transmit this buffered message to the out-of-sync nodes that have not been updated to the newest session key. Then, the out-of-sync nodes can extract the new session key from the message, because a stateless group key update scheme allows a node to obtain the updated group key as long as the node has the corresponding key update message. Therefore, two key-unsynchronized nodes that want to communicate can synchronize their session keys before the next round of group key update.

3. System Implementation

We implement the secure network access system on Imote2 [1] sensor platform. We develop the system components on sensor nodes in nesC language on TinyOS. Our implementation assumes that powerful nodes such as Laptops serve as the CA and the gateway that pre-distributes the keying materials to the sensor nodes and collects/processes the data from the sensor nodes in a wireless sensor network. We use Java to develop the CA and gateway applications.

3.1 Hardware Modules

Imote2, the hardware platform used in our implementation, is an advanced wireless sensor node platform, which is built around the low power PXA271 XScale processor and integrates an 802.15.4 radio (CC2420) with a built-in 2.4GHz antenna.

The radio chip of Imote2, CC2420, features hardware IEEE 802.15.4 security operations. All security operations are based on AES encryption/decryption using 128 bit keys. Although CC2420 provides hardware security support, TinyOS does not provide interfaces or modules to use these security operations. In our system, we implement a new interface called *CC2420AES* and related supporting modules to make AES encryption and decryption available for sensor applications.

3.2 Software Modules

The software modules of our system consist of two major parts: Java modules on the CA and Gateway nodes and TinyOS modules on sensor nodes, as shown in Figure 6.

Java modules include three components: CA, polynomial scheme, and SDM scheme.

- The CA generates public/private key pairs using Bouncy Castle Crypto APIs [12] and then pre-distributes (1) keying materials by generating polynomial share, (2) key pair for Self-Certified ECDH, (3) the commitment of one-way group key chain, and (4) labels in subset tree according to the node ID. CA must convert the above keying materials in nesC code before the pre-distribution.
- The polynomial generation scheme is responsible for generating polynomials.
- SDM scheme builds a full binary tree and a Steiner Tree, and provides the subset cover finding algorithm using Java Data Structure Library (JDSL) [20].

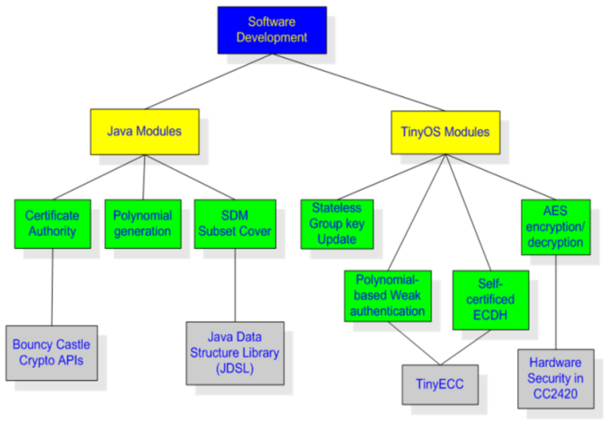
Software Development
Java Modules
TinyOS Modules
Certificate Authority
Polynomial generation
SDM Subset Cover
Stateless Group key Update
AES encryption/ decryption
Polynomial-based Weak authentication
Self-certified ECDH
Bouncy Castle Crypto APIs
Java Data Structure Library (JDSL)
TinyECC
Hardware Security in CC2420

Figure 6: System modules

TinyOS modules include the implementations for the polynomial-based weak authentication, Self-Certified ECDH protocol, AES encryption/decryption interface, and SDM scheme.

- Polynomial-based weak authentication uses the big number operation in TinyECC 1.0 [5].
- Self-Certified ECDH uses the big number operation, scalar point multiplication operation, and point addition operation in TinyECC 1.0.
- Because TinyOS does not provide an interface to use the hardware security component in CC2420, we develop an AES encryption/decryption interface that supports CTR, CCM, and CBC-MAC modes. Although hardware security in CC2420 only implements AES encryption (no AES decryption in hardware), the above three modes do not need AES decryption.
- SDM scheme provides functions that encrypt new group keys and generate group key update messages on the controller node. It also provides functions that derive subset keys from subsets and pre-distributed keying materials, and decrypt new group key using the derived subset key.

3.2.1.1 TinyECC

Because both Self-Certified ECDH and polynomial-based weak authentication protocols are based on TinyECC [5], we would like to give a review on TinyECC. Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The hardness of solving elliptic curve discrete logarithm problem (ECDLP) allows several cryptographic schemes based on elliptic curves. TinyECC [5] is to provide a ready-to-use, publicly available software package for ECC-based PKC operations that can be flexibly configured and integrated into sensor network applications.

TinyECC is implemented on TinyOS, which is a popular, open-source OS for networked sensors. All the TinyECC components have nesC implementations, though some modules also include inline assembly code, which can be turned on for faster execution on some sensor platforms. This allows TinyECC to be compiled and used on any sensor platform that can run TinyOS. TinyECC has been tested successfully on MICAz, TelosB, Tmote Sky, and Imote2. TinyECC adopts almost all existing optimizations for ECC operations. These optimizations can be turned on or off to balance the efficiency and the resource requirements.

3.2.1.2 *Self-Certified ECDH*

We implement Self-Certified ECDH protocol using the Big Number, scalar multiplication, and point addition operations in TinyECC. Our implementation consists of three components, Certificate Authority, controller node, and regular sensor node.

The CA generates and distributes private keys and related parameters to all nodes in the network, including controller nodes and regular sensor nodes. In our experiment, the CA is implemented on a laptop. We assume the laptop is secure and won't be compromised. Whenever we deploy a sensor node i, we run a java program to generate private key for node i and save the key in nesC code. Thus, when we compile and install the nesC code for node i, it can get its private key and related parameters for Self-Certified ECDH protocol. We use the bouncy castle provider for java [12] to provide the elliptic curve operations.

A regular sensor node sends a request message to a controller node to run the Self-Certified ECDH protocol as the initiator. The controller node waits for incoming requests from newly deployed sensor nodes, and then runs the Self-Certified ECDH protocol. We implement both fixed key agreement algorithm and ephemeral key agreement algorithm in Self-Certified ECDH .

3.2.1.3 *Polynomial based Weak Authentication*

For polynomial-based weak authentication, we implement a java program to facilitate the operations of the CA on the laptop. First, it generates a polynomial and saves the coefficients of the polynomial in a symmetric matrix. Then, it generates polynomial share according to the node ID and saves polynomial share in nesC code.

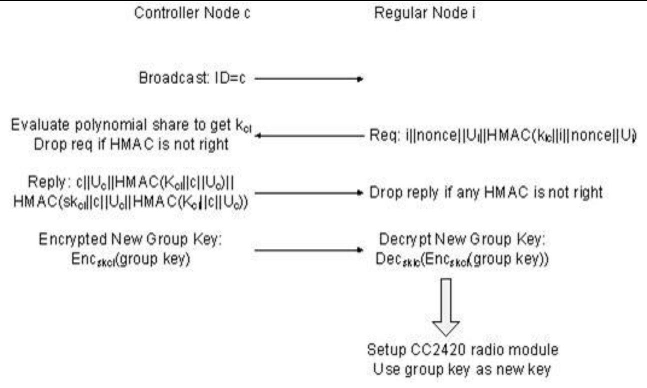
Controller Node c
Regular Node i
Broadcast: ID=c
Evaluate polynomial share to get k_{ci}
Drop req if HMAC is not right
Req: i||nonce||U_i||HMAC(k_{ic}||i||nonce||U_i)
Reply: c||U_c||HMAC(K_{ci}||c||U_c)||
HMAC(sk_{ci}||c||U_c||HMAC(K_{ci}||c||U_c))
Drop reply if any HMAC is not right
Encrypted New Group Key:
Enc_{skci}(group key)
Decrypt New Group Key:
Dec_{skic}(Enc_{skci}(group key))
Setup CC2420 radio module
Use group key as new key

Figure 7: Network Admission Control procedure

The Network Admission Control stage is shown in Figure 7. Controller nodes periodically broadcast their IDs. A sensor node *i* picks a controller node with the strongest signal strength (RSSI) and sends a request message to the controller node. This request message contains its Self-Certified ECDH public key U_i, nonce, and it's ID. Node i also appends a HMAC using the pairwise key k_{ic} generated by the polynomial scheme. Once the controller node receives this request message, it evaluates polynomial share using *i*, derives the key k_{ci}, and verifies the HMAC. If the HMAC is not correct, it simply drops the request message. Otherwise, the controller node performs Self-Certified ECDH to establish a pairwise key, sk_{ci}, with node *i*. Next, it sends back a reply message containing its Self-Certified ECDH public key U_c with one HMAC using k_{ci} and a new HMAC using sk_{ci}. After some random delay, controller node *c* sends out the new group key encrypted by using key sk_{ci}. At this time, node i should have finished its Self-Certified ECDH operations and obtained sk_{ic}. Then node i decrypts new group key and configures its CC2420 radio to use the new group key.

3.2.1.4 *Stateless Group Key Update*

The stateless group key update scheme consists of several components to implement SDM. Certificate Authority (CA) generates and saves the full binary tree for SDM in Java. Because JDK doesn't provide binary tree data structure, we use the data structure library in java (http://www.jdsl.org/) [22] to implement the binary tree. The deployed sensor nodes don't need to save the whole binary tree. For each sensor node *i*, CA generates nesC code that saves the labels in the binary tree for node i to derive subset key.

The controller node has a copy of the latest group key update message, thus it knows which node has been revoked and which node is normal. We implement an algorithm to check which node is out-of-sync. The basic idea works as follows: whenever the controller node finds a packet MIC verification failure, if the source node ID is not in the revoked list, it records the node ID and increases a specific counter by one. Whenever the controller node finds a packet MIC verification success, if the source node ID is not in the revoked list, the controller node clears the counter for that node ID. Once the counter for a node ID is larger than a threshold (e.g., 20), the controller node sends the buffered group key update message to this node. In this way, we limit the message overhead for helping the out-of-sync nodes obtain the new group key before the next round of group key update.

3.2.1.5 *Pairwise Key Based Group Key Update*

After a controller node and a sensor node establish a pairwise key using the Self-Certified ECDH protocol, the controller node uses simple push-based method to distribute current group key to the sensor node.

For each sensor node kept in the controller node's memory, the controller node retrieves the pairwise key shared with the regular one, encrypts and authenticates the current group key using the pairwise key, and sends the encrypted message to the regular node. The controller node starts a timer to wait for the ACK from the regular node. It will retransmit up to certain times before it receives an ACK (authenticated with the pairwise key) from the sensor node. Upon receiving a group key update message, a sensor node verifies the MIC and decrypts the message to retrieve the current group key. The sensor node performs the decryption operation and accepts the new group key only when the MIC can be verified correctly. Then, the sensor node sends an ACK message back to the controller node. The ACK message is authenticated with the pairwise key,

After a sensor node obtains the current group key from the controller node, it will use the group key to protect its messages, verify and forward the messages from other nodes. Among many sensor network applications, it is more important to provide message authentication than message confidentiality, because we must guarantee the data collected from the sensor networks are sent from the eligible nodes and have not been modified by attackers.

4. Performance Evaluation

We use a small wireless sensor network with 7 Imote2 to evaluate the overhead and efficiency of our system. We set the following parameters in our system. For Elliptic Curve Cryptography, we use the secp160r1 parameter set [11]. We choose 128-AES for symmetric key cryptography. We run our system on Imote2 at four different frequencies: 13 MHz, 104 MHz, 208 MHz, and 416 MHz.

4.1 Code Size

The code size of our system on Imote2 is shown in Table 1. Compared with the memory size of Imote2 (256 KB RAM, 32 MB ROM), our system is small and affordable.

Table 1. Code size on Imote2

Graphics	ROM (bytes)	RAM(bytes)
Controller module	179,836	18,788
Regular sensor module	180,188	16,388

4.2 Computation Time

We measure the execution time of polynomial-based scheme and Self-Certified ECDH protocol, and show the results in Figure 8 and Figure 9. We find that polynomial-based pairwise key establishment is very fast. It only needs around 2 *ms* to establish a pair-wise key using the polynomial-based key establishment scheme. It is 425 times faster than the fixed key establishment algorithm and 700 times faster than the ephemeral key establishment algorithm in Self-Certified ECDH. It explains why polynomial-based weak authentication can mitigate DoS attacks against Self-Certified ECDH.

The computation delay in our system is shown in Figure 10. *Request–ack* is the waiting time for a regular sensor node from sending out a request message to receiving a reply message from a controller node. *ECDH key establish* is the time to compute Self-Certified ECDH key. *Group key msg process* is the time for the regular node to decrypt new group key using the pairwise key generated in Self-Certified ECDH. The above three computation processes only happen when a sensor node joins a sensor network for the first time. *Revoke msg process* is the time for a sensor node to derive the subset key and decrypt the new group key using SDM. When we run the sensor nodes in higher frequency, we can reduce the computation delay accordingly. However, when we test the sensor nodes using 416 MHz frequency, we find that the nodes are not very stable due to the high frequency setting.

The partial packet transmission delay caused by CC2420 hardware security operations is shown in Figure 11. The delay introduced by the security operations is small. The packet payload size is 25 bytes, which is reasonable and efficient. We notice that the most expensive operation CCM mode is 1.38 times slows than the packet transmission without any security operation.

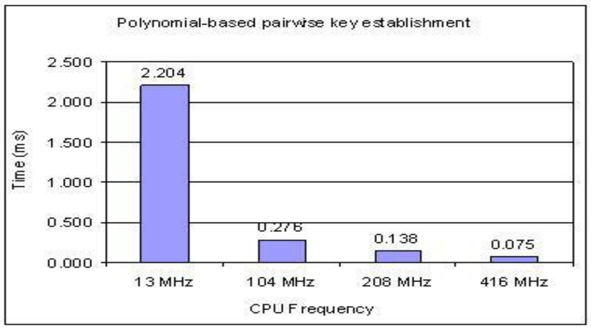

Figure 8. Time delays in polynomial-based scheme

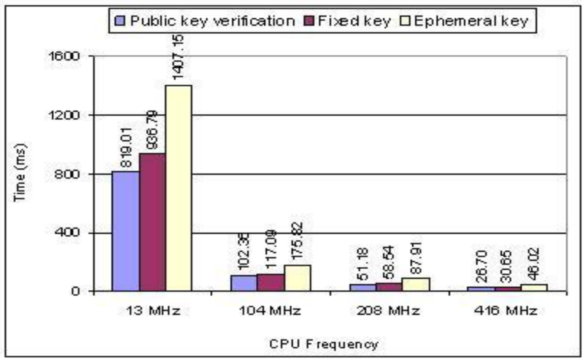

Figure 9. Time delay in Self-Certified ECDH

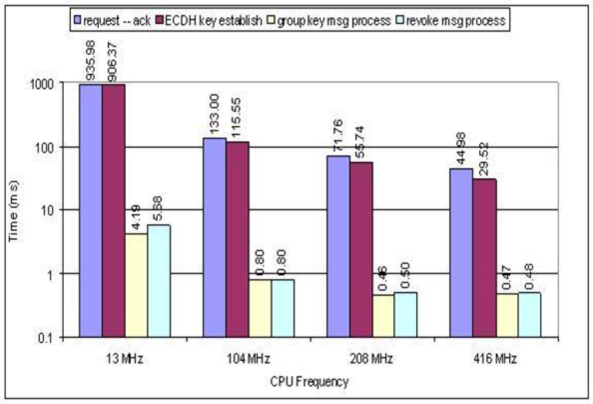

Figure 10. Computation delay

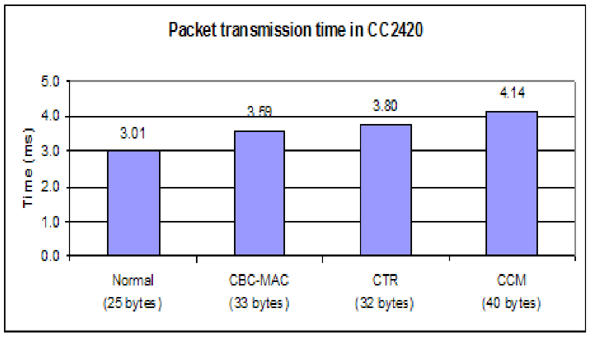

Figure 11. Packet transmission time in CC2420

4.3 Packet Communication Delay

The communication delays for three different security modes are listed in Table 2. In our experiment, we add 8 bytes as MIC in the CBC-MAC, which will introduce 0.256 *ms* of delay for transmitting the extra bytes through the radio.

Table 2. Packet communication delay

	Packet length (bytes)	Transmission time (ms)
Non-security	L	D=L*8/250
CTR	L+7	D+0.224
CBC-MAC	L+8	D+0.256
CCM	L+15	D+0.48

4.4 Energy Consumption

Because the computation and communication overhead of our secure system is very small, the extra energy consumption introduced by our secure system is small too. For the PXA271 microprocessor on Imote2 sensor nodes, it costs 390 *uA* in deep sleep mode, and 44 *mA* (13MHz) and 66 *mA* (104MHz) in active mode. For the CC2420 radio which is designed for very low current consumption, it only costs18.8 *mA* in receiving mode and 17.4 *mA* in transmitting mode.

5. Conclusion

In this paper, we have presented a prototype of secure network access control system in wireless sensor networks. Our design and implementation have successfully proven the feasibility of the proposed technical approaches. As our future work, we will extend the secure access system to support large wireless mesh networks.

6. REFERENCES

[1] http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf

[2] http://www.meshnetics.com/Building_Subsidence_Monitoring_Case_Study.pdf

[3] www.xbow.com

[4] SmartRF CC2420 Datasheet. http://focus.ti.com/lit/ds/symlink/cc2420.pdf

[5] A. Liu, P. Kampanakis, and P. Ning, ``TinyECC: Elliptic curve cryptography for sensor networks (version 0.3)," http://discovery.csc.ncsu.edu/software/TinyECC/.

[6] Arazi, B. (1999). Certification of dl/ec keys. In Proceedings of the IEEE P1363 Study Group for Future Public-Key Cryptography Standards.

[7] Donggang Liu, Peng Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in Proc. of ACM Conference on Computer and Communications Security (CCS), 2003

[8] D. Naor, M. Naor and J. Lotspiech "Revocation and Tracing Schemes for Stateless Receivers", CRYPTO '2001

[9] D. Hankerson, A. Menezens, and S. Vanstone. Guide to Elliptic Curve Cryptography. Springer, 2004.

[10] Certicom Research. Standards for efficient cryptography – SEC1: Elliptic curve cryptography. http://www.secg.org/download/aid-385/sec1_final.pdf

[11] Certicom Research. Standards for efficient cryptography – SEC2: Recommended Elliptic Curve Domain Parameters. http://www.secg.org/download/aid-386/sec2_final.pdf

[12] Bouncy Castle provider, http://www.bouncycastle.org/

[13] Chris Karlof, Naveen Sastry, and David Wagner., "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", ACM SenSys 2004,

[14] Y. Hu, A. Perrig, and D. B. Johnson. Wormhole detection in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, Dec 2001.

[15] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, May 2000.

[16] Sangwon Hyun, Peng Ning, An Liu, Wenliang Du, "Seluge: Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks," in Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), IP Track, pages 445--456, April 2008.

[17] R. Merkle. Protocols for public key cryptosystems. In Proceedings of the IEEE Symposium on Research in Security and Privacy, Apr 1980.

[18] Additional ECC Groups For IKE, http://www1.tools.ietf.org/html/draft-ietf-ipsec-ike-ecc-groups-06

[19] Donggang Liu, Peng Ning. Improving Key Pre-Distribution with Deployment Knowledge in Static Sensor Networks. In ACM Transactions on Sensor Networks (TOSN), Vol. 1, No. 2, pages 204-239, November 2005.

[20] IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs) http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[21] 25. C. Wong, M. Gouda, and S. Lam. Secure Group communications Using Key Graphs. In Proceedings of the ACM SIGCOMM '98, Vancouver, B.C, 1998.

[22] Java Data Structure Library (JDSL) http://www.jdsl.org/