Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 716265, 14 pages
http://dx.doi.org/10.1155/2013/716265

Research Article

A Hybrid Authenticated Group Key Agreement Protocol in Wireless Sensor Networks

Yue Li,[1] Dehua Chen,[1] Wei Li,[1] Gaoli Wang,[1] and Paul Smith[2]

[1] *School of Computer Science and Technology, Donghua University, No. 2999 North Renmin Road, Songjiang, Shanghai 201620, China*
[2] *Computer Games Academic Team EB.3.13A, Docklands Campus, ADI University of East London, London E16 2RD, UK*

Correspondence should be addressed to Yue Li; frankyueli@dhu.edu.cn

Received 14 October 2012; Revised 28 February 2013; Accepted 1 March 2013

Academic Editor: Motoki Sakai

Copyright © 2013 Yue Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wireless sensor networks are a modern and advanced technology whose applications are fast developing in recent years. Despite being a fascinating topic with various visions of a more intelligent world, there still exist security issues to be resolved in order to make WSNs fully adoptable. Due to the resource constraints of sensor nodes, it is infeasible to use traditional key establishment techniques that find use in fixed communication systems. In this paper, the design of a new hybrid Authenticated Group Key Agreement (AGKA) protocol is described for WSNs. The AGKA protocol reduces the high cost public-key operations at the sensor side and replaces them with efficient symmetric-key based operations. The proposed AGKA protocol is not only efficient but also meets strong security requirements. In order to demonstrate the protocol is verifiably secure and trustworthy, a formal verification of the AGKA protocol is carried out. Furthermore, several experiments are conducted on MICAz and TelosB platforms in order to evaluate the performance of the proposed protocol. The evaluation results show that the AGKA protocol is well suited for use with resource-constrained sensor nodes.

1. Introduction

Wireless sensor networks (WSNs) are viewed as a large number of small sensing self-powered devices/nodes which gather information or detect special events and communicate in a wireless fashion, with the end goal of handing their processed data to a base station. A diverse set of applications for sensor networks encompassing different fields have already emerged including medicine, agriculture, environment, military, electrical power systems, home appliances, toys, and many others.

In these and other vital, life-critical, or security-sensitive applications, secure and fast transmission of sensitive digital information over the sensor network is essential. A solid key management framework is one of the most crucial technologies for achieving secure infrastructure in wireless sensor networks.

Considering the limited resources of both computational ability and power supply of wireless sensor devices, the design of security protocols for wireless sensor networks is a nontrivial challenge given that most public key operations require expensive computations. Therefore, there is a need to employ energy-efficient key agreement protocols in order to prolong each sensor's battery life.

In recent years, symmetric-key-based key establishment schemes have gained popularity due to their small computational overhead. A promising solution for the establishment of symmetric keys in wireless sensor network applications is to use key predistribution protocols such as those studied in various papers [1–3]. Although symmetric mechanisms achieve low computational overhead when compared with public key operations, the key management for symmetric key based protocols is complicated and is always subject to attack by adversaries. Therefore, many public-key-based protocols have been proposed [4–11] for wireless sensor networks which give more flexibility and scalability.

In this paper, we focus on WSN applications involving clusters of wireless sensor nodes. We have designed a new hybrid authenticated group key agreement (AGKA) protocol. The motivation of which was to exploit the difference in capabilities between gateways and sensors and put the cryptographic burden on gateways where the resources are less constrained. We have also implemented the AGKA protocol on

TelosB and MICAz motes and performed several experiments in order to evaluate the performance of the AGKA protocol in terms of its energy consumption and memory usage. The evaluation results show that the proposed protocol is well suited for use with resource-constrained sensor nodes with limited processing power and power resources.

The remainder of this paper is organized as follows. Section 2 describes related works. Some preliminaries and network model are reviewed in Section 3. Section 4 presents our key agreement protocol. In Section 5, the security of the proposed protocol is discussed. We present the performance evaluations in Section 6 and provide our research conclusions in Section 7.

2. Related Works

SPINS [12] is one of the most popular symmetric-key-based security schemes used today. In this memory-efficient scheme, the nodes need only share a key with the base station, and establish keys with other nodes through the base station. This type of scheme is suitable for sensor networks with small numbers of sensor nodes manually deployed around the base station. The big drawback of this scheme is that the base station is a single point of attack, which could result in the compromise of the entire network. Those nodes closest to the base station must forward a high volume of traffic to the base station and this reduces the lifetime of the network as these nodes expend greater energy resources.

Key predistribution is an alternative approach, which distributes the keys to all sensors prior to the deployment of the sensors. Zhu et al. [13] proposed Localized Encryption and Authentication Protocol (LEAP) which supports the establishment of four types of keys for each sensor node including a pair-wise key and a group key (a network-wide shared key).

Eschenauer and Gligor [1] proposed the use of random graph theory, which was used to develop one of the first random predistribution schemes. A random graph is fully connected with a high probability if the average degree of its nodes is above a certain threshold. Generally high-density deployments result in a fully connected network. Hence, key establishment only needs to be performed such that any two neighbors have some probability p of successfully completing key establishment. Eschenauer and Gligor used this theory to develop a framework for key random predistribution protocols. This framework involves three phases: predistribution, shared-key discovery, and path-key establishment.

The computation complexity and energy consumption of those symmetric-key-based protocols are relatively small. However, the key management for pure symmetric-key-based systems can be complicated, a key distribution center (KDC) can be required, or a large number of symmetric keys can be preloaded into devices. Both of these solutions can reduce the scalability of WSNs. In contrast, public-key-based protocols give more flexibility and scalability in large sensor networks where new devices keep entering the cluster. However, public-key-based protocols require more expensive computational power.

In cluster-based wireless sensor networks, the design of secure group key establishment protocols is a foremost security issue. A group key establishment protocol allows participants to construct a group key that is used to encrypt/decrypt transmitted messages among participants over an open channel.

Recently several key agreement protocols have been proposed to offload public-key cryptographic computational requirements to servers and have the low-end devices do less work. Bresson et al. [4] proposed a group key agreement protocol well suited to imbalanced wireless networks consisting of devices with strict energy consumption restrictions and wireless gateways with less stringent restrictions. Their idea was to let a cluster of mobile devices and one wireless gateway dynamically agree on a session key. However, their protocol does not satisfy some important security properties such as mutual authentication and forward secrecy [14].

Nam et al. [15] further improved the mutual authentication of Bresson et al.'s protocol by adopting the Katz-Yung scalable compiler [16] whereby one online signature and $n-1$ verifications must be required; the computational cost, though reduced, is still expensive for resource constrained sensor devices.

Tseng [17] proposed an efficient group key agreement protocol based on the two aforementioned protocols. It employs an online/offline signature scheme [18] and shifts much of the computation to the wireless gateways possessing more computational power and energy. Nevertheless, it does not satisfy some important security properties such as mutual authentication [19].

In recent years, Elliptic-Curve-Cryptography-based-key agreement protocols [5, 9, 10, 20–22] have been designed for use in constrained mobile device environments and wireless sensor networks because of their small key sizes, such as the ECMQV protocol with ECC X.509 certificates [20] and implicit certificates [21] and the ECDSA authenticated key exchange protocol [22]. In 2004, Huang et al. proposed a hybrid authenticated key establishment protocol based on probably secure elliptic curve encryption [5] and the elliptic curve implicit certificate scheme [20]. In 2005, Liu and Ning created TinyECC [23], a software package that provides Elliptic Curve Cryptography (ECC) operations for TinyOS [24]. It supports all elliptic curve operations over prime fields F_p, including point addition, point doubling, and scalar point multiplication, as well as ECDSA operations. In 2011, Ammayappan et al. proposed an ECC-based two-party authenticated key agreement protocol for mobile ad hoc networks, which utilises both RSA and ECC to achieve mutual authentication. This method increases the computation burden on sensor side [10].

Using the concept of Schnorr Signature [25] and based on ECC, Huang et al. in [5] designed a key establishment in the authentication procedure of the access control scheme for WSNs. The new designed key establishment in [11] also used the concept of "timebound" in which once time period has elapsed, the sensor node in the wireless sensor network cannot access any data for a future time period in order to protect future messages. Huang et al. claimed that the authentication procedure and common key generation proposed in [5] offers computational efficiency, energy, and bandwidth savings. Nevertheless, adversaries can still apply a sensor node replication attack in the period of the expiration time. The reason is that the adversary can compromise the sensor node and apply the replication attack before expiration time.

In order to reduce communication cost, some ID-based protocols for wireless sensor networks have been proposed where a sensor node does not need to transmit its implicit certificate [8]. Zhang et al. proposed three protocols for wireless sensor networks [6, 26, 27]. Those protocols offer low communication overhead and low memory requirements by eliminating the public key certificate. But in those protocols, sensor nodes should still perform expensive computation such as Weil/Tate pairing and Map-to-Point operations. Recently, Zhang et al. [8] proposed an efficient ID-based protocol for key agreement in wireless sensor networks. This protocol removes expensive operations from a sensor node side and eliminates the communication overhead of transmitting public-keys, but this protocol is vulnerable to replication attacks, where adversaries can use this weakness to masquerade as a security manager and share the pair-wise key with the sensor node.

From the discussion of the recent representative key agreement protocols designed for wireless sensor networks, we find that those protocols are computationally expensive for sensor nodes or vulnerable to impersonator's attacks. It can be seen that the design of a secure authenticated group agreement protocol well suited to wireless sensor networks is a nontrivial challenge, which inspires us to propose a verifiably secure authenticated group key agreement protocol.

3. Network Model and Notations

Before the discussion of key establishment protocols involving public key cryptography, we will first present the model of the unbalanced cluster-based wireless sensor networks.

3.1. Network Model. The IEEE 802.15.4 low-rate wireless personal area network standard [28] specifies the physical layer and medium access control layer of a low data rate, ultra low power, and low cost sensor network. It defines two device types: a Full Functional Device (FFD) and a Reduced Functional Device (RFD). An RFD takes on the role of an end device, such as a low-power sensor, while an FFD takes the role of a coordinator, a gateway, or a security manager.

The wireless system environment we model is an unbalanced/asymmetric cluster-based wireless sensor network, which consists of some sensor nodes with strict computational capability restrictions and a gateway with less restriction. We consider a set of resource-limited sensor nodes (also called low-power nodes) communicating with a gateway (also called powerful node), in which each low-power node can send messages to the gateway via unicast communication, and the gateway can broadcast or unicast messages to each low-power node. The gateway covers an entire group region called a cell. It is the cluster-head of the group region. In the group region, the data transmission between gateway and its client nodes uses low-power wireless technology such as IEEE 802.15.4 standard and Zigbee. The communication between gateways and the base station could use WiFi and wired LAN technology. The monitoring software on the base station can collect and analyze the sensing data and put the useful information on the web server. All the authenticated users can login to the website to not only get the information of the target object but also maintain the sensor network by performing tasks such as updating/renewing the group key, putting a particular group of sensor nodes into sleep mode or merging the neighboring groups.

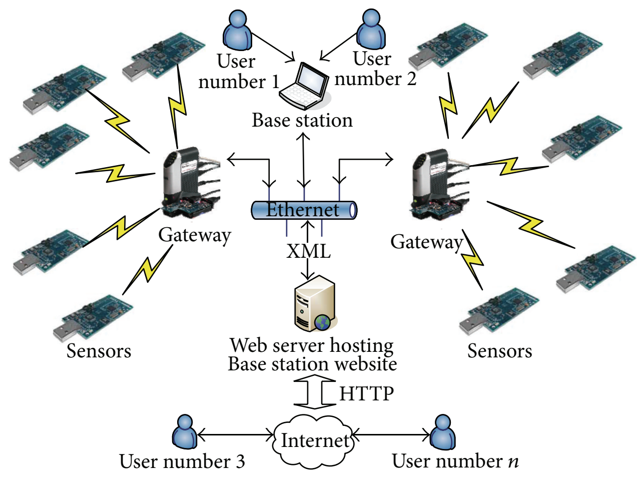


Figure 1: Network model of the asymmetric wireless sensor network.

Figure 1 shows the network model of the asymmetric wireless sensor network.

3.2. Key Notation and Terms. Let $U = 1$ be the initial set of low-power sensor nodes that want to generate a group key with gateway V. In Table 1, we summarize the key notations and terms used in the group key agreement protocol.

4. The Proposed Group Key Agreement Protocol

This section specifies the algorithms and features of the proposed AGKA protocol. The new AGKA protocol is implemented using the elliptic curve version of the Diffie-Hellman problem [29]. In addition to the use of an ECC cryptosystem, the proposed AGKA protocol also adopts a symmetric-key cryptosystem. The protocol reduces the cost of elliptic curve random point scalar multiplications at the sensor side and replaces them with low cost and efficient symmetric-key-based operations. Furthermore, it authenticates the entities based on a combination of the Elliptic Curve Digital Signature Algorithm (ECDSA) [30] and the Message Authentication Code (MAC).

The AGKA protocol consists of four algorithms.

(1) The key generation algorithm *AGKA.Kgen*(ℓ) is a probabilistic algorithm which on input of a security parameter ℓ provides each client $U_i \in \psi_C$ and the gateway with long-lived keys.

(2) The setup algorithm *AGKA.Setup*(ϑ) is an interactive protocol which on input of a set of clients ψ_C sets the

Table 1: Key notation and terms.

Notation	Description
ψ_C	Set of clients
q	A large prime
p	A large prime such that $p = 2q + 1$
P	Denotes a base point of large order n selected for an elliptic curve, which is public to all users
g	A generator for the subgroup G_q
(Q_i, q_i)	Public key and private key pair of a low-power node U_i, $Q_i = g^{q_i} \bmod p$
(Q_V, q_V)	Public key and private key pair of the powerful node V
(D_i, d_i)	Ephemeral public key and private key pair of low-power node
$\mathrm{Sig}_{U_i}(m)$	The signing algorithm based on ECDSA schemes under U_i's private key q_i and the signed message m
$\parallel$	Denotes concatenation
N	Nonce
c	Counter
$\mathrm{MAC}(M, K)$	The computation of a MAC for a message M using MAC key K
GK	Group shared session key

wireless client group to be $\psi_C = \vartheta$ and provides each client U in ψ_C with a secret value sk shared with the gateway.

(3) The join algorithm *AGKA.join*(ϑ) is an interactive protocol which on input of a set of clients ϑ updates the wireless client group ψ_C to be $\psi_C \cup \vartheta$ and provides each client U in ψ_C with a (new) shared secret value sk.

(4) The remove algorithm *AGKA.Remove*(ϑ) is an interactive protocol which on input of subset ϑ of the wireless client group ψ_C updates the latter to be $\psi_C \setminus \vartheta$ and provides each client U in ψ_C with a new shared secret value sk.

Each cluster/group in a hierarchical cluster-based WSN is represented as the set μ, which consists of N sensor devices (also called clients), and a gateway. A nonempty subset of μ is called sensor client group ψ_C, which consists of clients communicating with the gateway. An elliptic curve E defined over prime fields $\mathbb{F}_q$ with coefficients and a base point P of large order n is selected and made public to all users. The protocol considers a signature scheme *SIGN* = (*SIGN.Kgen, SIGN.Sig, SIGN.Ver*). Each client U_i holds a pair of signing private/public key $(\mathrm{SK}_i, \mathrm{PK}_i)$, which are the output of the key generation signature scheme algorithm *SIGN.Kgen*.

4.1. Key Generation. The algorithm *AGKA.Kgen*, on input of the set of clients ψ_C and a security parameter ℓ, performs the following steps.

(1) Execute *SIGN.Kgen*(ℓ) for each client U_i in ψ_C to provide each client with a pair $(\mathrm{SK}_i, \mathrm{PK}_i)$ of signing/verifying keys. The private key SK_i is given to the client U_i in a confidential way, while each public key PK_i is sent to the gateway.

(2) Choose random integer q_v, compute $Q_V = q_V * P$, and set the gateway's private/public keys $(\mathrm{SK}_V, \mathrm{PK}_V) = (q_V, Q_V)$. The private key is given to the gateway in a confidential way, while the public key is certified and sent to the clients. The pair (q_V, Q_V) will be the long-term Diffie-Hellman pair of the gateway.

Basically, for an ECC-based key agreement, each client will generate an ephemeral Diffie-Hellman pair (d_i, D_i), which thus leads to a session key R_i $(R_i = d_i * Q_V)$ shared between the client U_i and the gateway V. Meanwhile, ECDSA signature δ_i is used for authenticating each client node.

4.2. Group Key Setup. As depicted in Figure 2, the group key agreement setup runs as follows.

Step 1. To establish the group key in the cluster, each node $U_i \in \psi_C$ randomly selects a k-bit integer K_i and a $(160 - k)$-bit integer N_i as the nonce. Additionally, U_i randomly picks a random integer $d_i \in [2, n-2]$ as its ephemeral private key and gets the ephemeral public key $D_i = d_i * P$. Then, U_i computes $R_i = d_i * Q_V$ and cipher text $e_i = (K_i \parallel N_i) \oplus R_i \cdot x$, where $R_i \cdot x$ is the x coordinator of R_i. The client node U_i then generates an ECDSA signature $\delta_i = \mathrm{Sig}_{U_i}(D_i \parallel e_i)$ under the private key SK_i of U_i. Finally, each node U_i sends (D_i, e_i, δ_i) to gateway V. Note that generations of the ephemeral public key D_i and the shared secret R_i can be precomputed before the node joins the network, which requires additional memory space but speeds up the protocol's execution.

Step 2. For each node, the gateway first checks if the nonce N_i is fresh and then checks the signature δ_i to authenticate each node U_i. If the authentication holds, it computes $R_i = D_i * q_V$ and then decrypts e_i and gets K_i and N_i. Subsequently, the gateway initializes counter c and computes a group session key $\mathrm{GK} = H(c \parallel K_1 \parallel K_2 \parallel \cdots \parallel K_n)$. The gateway then creates a cipher text $e_{V_i} = \mathrm{GK} \oplus R_i \cdot x$ and sends each client node the cipher text e_{V_i}, counter c, and nonce N_i with $\mathrm{MAC}((e_{V_i} \parallel c), K_i)$. Note that secret key K_i is selected as the MAC key between node U_i and the gateway since K_i is only known to the node U_i and the gateway.

Step 3. Each sensor node U_i first performs the authentication of the gateway through verifying MAC. If the authentication holds, the client calculates the group session key $\mathrm{GK} = e_{V_i} \oplus R_i \cdot x$. HMAC with MD5 hash algorithm is used to calculate the MAC value. MAC is used to verify the integrity of the received message. MAC can also be used to confirm that the received message is sent by the sender who knows the MAC key K_i.

4.3. Algorithm for New Node Joining. The algorithm *AGKA.Join*, on input of the set of appearing client devices ϑ, performs the following steps.

(1) When a new member $U_{i+1} \in \vartheta$ wants to join a group, it must first be authenticated by the base station.

(2) Update the wireless client group $\psi_C = \psi_C \cup \vartheta$.

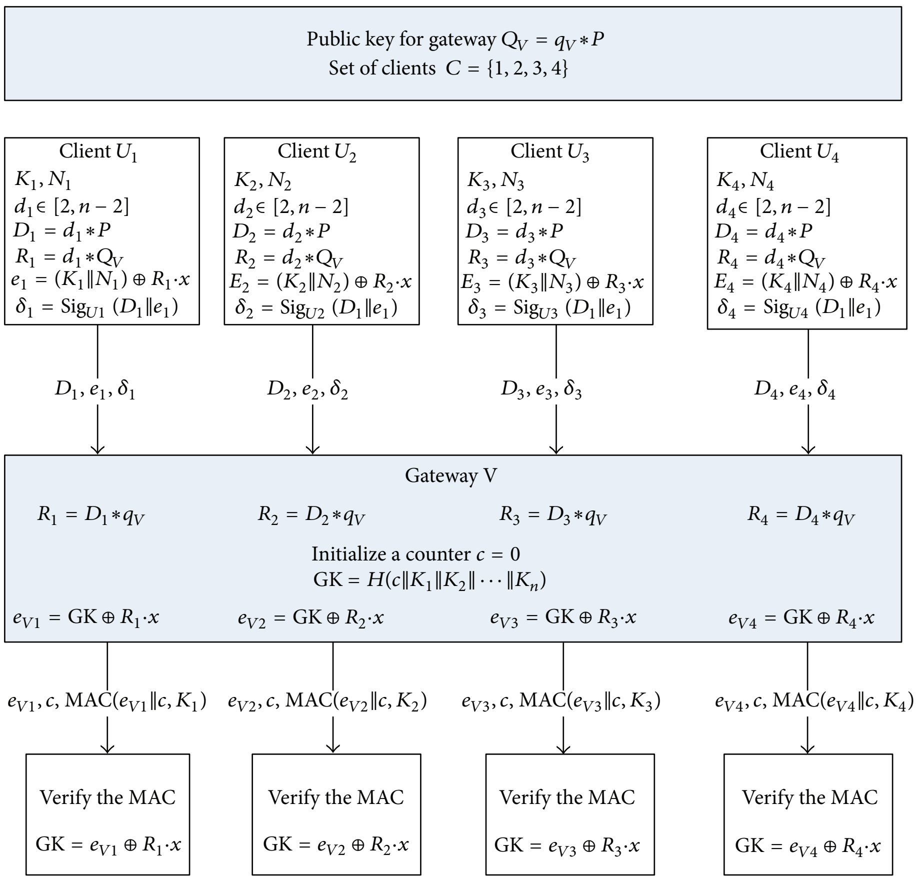

Public key for gateway $Q_V = q_V * P$
Set of clients $C = \{1, 2, 3, 4\}$
Client U_1: K_1, N_1; $d_1 \in [2, n-2]$; $D_1 = d_1 * P$; $R_1 = d_1 * Q_V$; $e_1 = (K_1 \| N_1) \oplus R_1 \cdot x$; $\delta_1 = \mathrm{Sig}_{U1}(D_1 \| e_1)$
Client U_2: K_2, N_2; $d_2 \in [2, n-2]$; $D_2 = d_2 * P$; $R_2 = d_2 * Q_V$; $E_2 = (K_2 \| N_2) \oplus R_2 \cdot x$; $\delta_2 = \mathrm{Sig}_{U2}(D_1 \| e_1)$
Client U_3: K_3, N_3; $d_3 \in [2, n-2]$; $D_3 = d_3 * P$; $R_3 = d_3 * Q_V$; $E_3 = (K_3 \| N_3) \oplus R_3 \cdot x$; $\delta_3 = \mathrm{Sig}_{U3}(D_1 \| e_1)$
Client U_4: K_4, N_4; $d_4 \in [2, n-2]$; $D_4 = d_4 * P$; $R_4 = d_4 * Q_V$; $E_4 = (K_4 \| N_4) \oplus R_4 \cdot x$; $\delta_4 = \mathrm{Sig}_{U4}(D_1 \| e_1)$
D_1, e_1, δ_1 / D_2, e_2, δ_2 / D_3, e_3, δ_3 / D_4, e_4, δ_4
Gateway V
$R_1 = D_1 * q_V$ / $R_2 = D_2 * q_V$ / $R_3 = D_3 * q_V$ / $R_4 = D_4 * q_V$
Initialize a counter $c = 0$
$\mathrm{GK} = H(c \| K_1 \| K_2 \| \cdots \| K_n)$
$e_{V1} = \mathrm{GK} \oplus R_1 \cdot x$ / $e_{V2} = \mathrm{GK} \oplus R_2 \cdot x$ / $e_{V3} = \mathrm{GK} \oplus R_3 \cdot x$ / $e_{V4} = \mathrm{GK} \oplus R_4 \cdot x$
$e_{V1}, c, \mathrm{MAC}(e_{V1} \| c, K_1)$ / $e_{V2}, c, \mathrm{MAC}(e_{V2} \| c, K_2)$ / $e_{V3}, c, \mathrm{MAC}(e_{V3} \| c, K_3)$ / $e_{V4}, c, \mathrm{MAC}(e_{V4} \| c, K_4)$
Verify the MAC: $\mathrm{GK} = e_{V1} \oplus R_1 \cdot x$ / $\mathrm{GK} = e_{V2} \oplus R_2 \cdot x$ / $\mathrm{GK} = e_{V3} \oplus R_3 \cdot x$ / $\mathrm{GK} = e_{V4} \oplus R_4 \cdot x$

FIGURE 2: The AGKA protocol with five devices U_1, U_2, U_3, U_4, and V.

(3) Each appearing client $U_j \in \vartheta$ chooses at random a k-bit integer K_j, a $(160-k)$-bit integer N_j as the nonce, and the ephemeral private key $d_j \in [2, n-2]$ and precomputes the ephemeral public key $D_j = d_j * P$ and the shared session key $R_j = d_j * Q_V$. Then, U_j precomputes the cipher text $e_j = (K_j \parallel N_j) \oplus R_j \cdot x$ and the signature $\delta_j = \mathrm{Sig}_j(D_j \parallel e_j)$ under the private key SK_j.

(4) Each appearing client U_j sends the value (D_j, e_j, δ_j) to the gateway V.

(5) The gateway V verifies the incoming signatures and if correct, operates as in the *Setup* phase with an increased counter c and computes the group session key

$$\mathrm{GK} = H\left(c \parallel \left\{K_j\right\}_{j\in\vartheta}\right). \quad (1)$$

After that, the gateway sends to each client $U_i \in \psi_C$ the counter c, cipher text $e_{V_i} = \mathrm{GK} \oplus R_i \cdot x$, and $\mathrm{MAC}((e_{V_i} \parallel c), K_i)$.

(6) Each client $U_i \in \psi_C$ already holds the value K_i, the shared secret R_i, and the old counter value. So, it first checks that the new counter is greater than the old one and the MAC value, and if the check holds, it simply recovers the group session key $\mathrm{GK} = e_{V_i} \oplus R_i \cdot x$.

4.4. Algorithm for Node Removing. The algorithm *AGKA.Remove*, on input of the set ϑ of disappearing client-sensors, performs the following steps.

(1) Update the sensor group $\psi_C = \psi_C / \vartheta$.

(2) The gateway V operates as in the *Setup* phase. It increases the counter c and computes the shared group session key $\mathrm{GK} = H(c \parallel \{K_i\}_{i\in\psi_C})$.

(3) Then, it sends to each client $U_i \in \psi_C$ the values c, cipher text $e_{V_i} = \mathrm{GK} \oplus R_i \cdot x$, and $\mathrm{MAC}((e_{V_i} \parallel c), K_i)$.

(4) Each client $U_i \in \psi_C$ already holds the value K_i, the shared secret R_i, and the old counter value. So, it first checks that the new counter is greater than the old one and the MAC value, and if the check holds, it simply recovers the group session key $\mathrm{GK} = e_{V_i} \oplus R_i \cdot x$.

5. Security Evaluation

The presented AGKA protocol overcomes the security weaknesses detected in the previously discussed protocols. The security evaluation is discussed in this section.

5.1. Sensor Node Replication Attack. The fresh nonce N_i is used in the message sent from the client node U_i for $i = 1, 2, \ldots n$, so that it can make sure no replayed message (cloning fraud) will be allowed in the protocol. For instance, if an adversary wants to replay the previously transmitted message from one client, it would use the same nonce value in previous round, which will be realized by the gateway who knows the last nonce generated by the client. If an adversary wants to replay the previously transmitted message from the gateway, it would not pass the check of the counter c implemented in Step 3 on the client side. Meanwhile, the signature of the message sent from the client node is also utilized in Step 1 to provide the authentication of the client nodes. Therefore, the proposed protocol prevents the replication attacks.

5.2. Sybil Attack. In this attack, a malicious sensor claims multiple IDs (identities) or locations [31]. In the proposed scheme, each client sensor is authenticated by the base station and gets a unique ID. In addition, each client owns a long-term key pair $(\mathrm{SK}_i, \mathrm{PK}_i)$, where the private key SK_i is used to generate the digital signature of the client. The private key is only known by the private key's owner and kept in secret. A malicious sensor cannot masquerade a forge ID and forge key pair without the base station's authentication. During the *AGKA.Setup* phase, the client's private key is used to sign the sending message, when the gateway in the group receives the signed message from a client node; it will first verify the signature $\delta_j = \mathrm{Sig}_j(D_j \parallel e_j)$ in order to authenticate the identity of the client node. Elliptic Curve Digital Signature Algorithm (ECDSA) is chosen in the proposed protocol to generate and verify the signature of each client. The security of ECDSA is founded in the difficulty of solving the discrete logarithm problem in prime order subgroups of $\mathbb{Z}_p^*$. The adversary cannot masquerade the client U_i and generate the legal signature to pass gateway's authentication without the private key of the client U_i. Even in worst case, the adversary compromise one client sensor N_i but still is not able to claim a new identity N_i' in the vicinity of node N_j because the adversary only knows the private key of the compromised node N_i but not the private key of node N_j. As a result, with the use of ECDSA on the gateway to authenticate the identity of each client sensor, the proposed protocol can withstand the Sybil attack.

5.3. Mutual Authentication. The signature of the message sent from the client node is generated in Step 1, which is verified by the gateway in Step 2. This provides the authentication of the client node. Meanwhile, a Message Authentication Code (MAC) is applied in Step 2. This will provide proof of authentication and integrity for the sent message. In the proposed protocol, the MAC key K_i is generated by client node U_i and sent to the gateway in a confidential way, where K_i is encrypted by $e_i = (K_i \parallel N_i) \oplus R_i \cdot x$. Only the gateway with the private key of q_V can decrypt the encrypted message and recover K_i. Thus, only the gateway V and client node U_i knows the MAC key K_i. Therefore, the MAC code $\mathrm{MAC}((e_{V_i} \parallel c), K_i)$ can be used to authenticate the identity of the gateway. As a result, the AGKA protocol provides the authentication between the client nodes and the gateway.

5.4. Perfect Forward Secrecy. A key agreement protocol offers forward secrecy if compromisation of a long-term key cannot result in the compromisation of previously established session keys. As mentioned in Step 1 of the AGKA protocol, $(d_i, D_i, R_i, e_i, \delta_i)$ is stored in the memory storage of the low-power node and each tuple $(d_i, D_i, R_i, e_i, \delta_i)$ is used only once. In this case, $(d_i, D_i, R_i, e_i, \delta_i)$ must be erased as soon as they are no longer useful. Obviously, since the low-power nodes' long-term keys SK_i are used only for authentication and they are not used for hiding the group key, the leakage of any client node's long-term key does not reveal anything about the group key. Furthermore, strong (partial) forward-secrecy (where any internal data is revealed, that is, the signing key but also the d_i, k_i, and R_i) is also achieved if the d_i's and R_i's are erased as soon as they are no longer useful (the client has left from the group). As a consequence, no information about previous session keys can be found in the memory of the low-power sensor nodes.

6. Formal Verification of the AGKA Protocol

Traditionally, cryptographic protocols have been designed and verified using informal and intuitive techniques. However, an absence of formal verification has proven [32, 33] to lead to flaws and security errors remaining undetected in a protocol. Formal verification aims at providing a rigid and thorough means of testing the correctness of a cryptographic protocol so that even subtle defects can be uncovered. A number of formal techniques have been developed for this purpose. This section first discusses the Coffey-Saidha-Newe (CSN) logical technique [32] and then formally analyzes and verifies the proposed group key agreement protocol using this logic.

6.1. CSN Modal Logic. The CSN logic provides a means of verifying hybrid cryptographic protocols. The logic can analyze the evolution of both knowledge and belief during a protocol execution, and is therefore useful in addressing issues of both security and trust. The inference rules provided are the standard inferences required for natural deduction and the axioms of the logic are sufficiently low-level to express the fundamental properties of hybrid cryptographic protocols, such as the ability of a principal to encrypt/decrypt based on knowledge of a cryptographic key. The logic is capable of analyzing a wide variety of hybrid cryptographic protocols because the constructs of the logic areof general purpose and therefore provide the user with increased flexibility allowing him to develop his own theorem.

The underlying assumptions of the logic can also be stated as follows. The communication environment is hostile but reliable; the cryptosystems used are ideal. That is, the encryption and decryption functions are completely noninvertible without knowledge of the appropriate cryptographic key and are invertible with knowledge of the appropriate cryptographic key. Keys used by the system are considered valid if they have not exceeded their validity period and only known by the rightful owner(s).

6.1.1. The CSN Logic Language

$a, b, c, \ldots$: general propositional variables

Φ: an arbitrary statement

Σ and Ψ: arbitrary entities

i and j: individual entities

ENT: the set of all possible entities

k: a cryptographic key. In particular, k_Σ is the public key of entity Σ and k_Σ^{-1} is the corresponding private key of entity Σ

$t, t', t'', \ldots$: moments in time. For example, $t1$ represents time after Step 1 of protocol has completed

$e(x, k_\Sigma)$: encryption function, encryption of x using key k_Σ

$d(x, k_\Sigma^{-1})$: decryption function, decryption of x using key k_Σ^{-1}

$\mathrm{ks}_{(\Sigma,\Psi)}$: shared secret key for entities Σ and Ψ

$\mathrm{KS}_{\{\Sigma,\Psi\}}$: set of good shared keys for entities Σ and Ψ

$\mathrm{ss}_{(\Sigma,\Psi)}$: shared secret for entities Σ and Ψ (secret can be fresh)

$\mathrm{SS}_{\{\Sigma,\Psi\}}$: set of good shared secrets for entities Σ and Ψ

$E(x, \mathrm{ks}_{(\Sigma,\Psi)})$: encryption of plaintext message x using the shared secret key of entities Σ and Ψ

$D(x, \mathrm{ks}_{(\Sigma,\Psi)})$: decryption of ciphertext message x using the shared secret key of entities Σ and Ψ

K: propositional knowledge operator (true or false evaluation) of Hintikka [34]

$K_{\Sigma,t}\Phi$: Σ knows statement Φ at time t

L: knowledge predicate (assigns an object a property). $L_{\Sigma,t}x$ means that Σ knows and can reproduce object x at time t

B: belief operator. $B_{\Sigma,t}\Phi$ means that Σ believes at time t that statement Φ is true

C: "Contains" operator. $C(x, y)$ means that the object x contains the object y. The object y may be cleartext or ciphertext in x

S: emission operator. $S(\Sigma, t, x)$ means that Σ sends message x at time t

R: reception operator. $R(\Sigma, t, x)$ means that Σ receives message x at time t

A: authentication operator. $A(\Sigma, t, \Psi)$ means that Σ authenticates Ψ at time t.

The language includes the classical logical connectives of conjunction ($\wedge$), disjunction ($\vee$), complementation ($\neg$), and material implication ($\rightarrow$). The symbols $\forall$ and $\exists$ denote universal and existential quantification, respectively. The symbol $\in$ indicates membership of a set and / denotes set exclusion. The symbol $\vdash$ denotes a logical theorem. The logic does not contain specific temporal operators, but the knowledge, belief, and message transfer operators are time-indexed.

6.1.2. Inference Rule.

6.1.2. Inference Rule. The logic incorporates the following rules of inference.

(R1) From $\vdash p$ and $\vdash (p \rightarrow q)$ infer $\vdash q$.

(R2) (a) From $\vdash p$ infer $\vdash K_{\Sigma,t}p$;
(b) from $\vdash p$ infer $\vdash B_{\Sigma,t}p$.

(R1) is the *Modus Ponens* and states that if p can be deduced and $(p \rightarrow q)$ can be deduced, then q can also be deduced. (R2) consists of the generalisation rules which state that if p is a theorem, then knowledge and belief in p are also theorems.

The logic also includes the following standard propositional rules of natural deduction.

(R3) From $(p \wedge q)$ infer p.

(R4) From p and q infer $(p \wedge q)$.

6.1.3. Axioms. Two types of axioms are used in this logic, logical and nonlogical. Logical axioms are general statements made in relation to any system, while non-logical are system specific.

Logical Axioms. The logic includes the following standard modal axioms for knowledge and belief:

(A1) $\exists t \exists p \exists q (K_{\Sigma,t}p \wedge K_{\Sigma,t}(p \rightarrow q) \rightarrow K_{\Sigma,t}q)$;

(A2) $\exists t \exists p (K_{\Sigma,t}p \rightarrow p)$.

The axiom (A1) is application of the Modus Ponens to the knowledge operator. The axiom (A2) is called the knowledge axiom and is said to logically characterise knowledge. If something is known, then it is true. This property distinguishes between knowledge and belief. Consider

(A3) (a) $\exists t \exists x \exists i, i \in \{\mathrm{ENT}\}(L_{i,t}x \rightarrow \forall t', t' \geq t\ L'_{i,t}x)$;
(b) $\exists t \exists x \exists i, i \in \{\mathrm{ENT}\}(K_{i,t}x \rightarrow \forall t', t' \geq t\ K'_{i,t}x)$.

Axioms (A3)(a) and (A3)(b) assert that knowledge, once gained, cannot be lost. Consider

(A4) $\exists t \exists x \exists y (\exists i, i \in \{\mathrm{ENT}\} L_{i,t}y \wedge C(y, x) \rightarrow \exists j, j \in \{\mathrm{ENT}\} L_{j,t}x)$.

If a piece of data is constructed from other pieces of data, then each piece of data involved in the construction must be known to some entity.

Nonlogical Axioms. The non-logical axioms reflect the underlying assumptions of the logic. These assumptions relate to the emission and reception of messages and to the use of encryption and decryption in these messages. Consider

(A5) $\exists t \exists x (S(\Sigma, t, x) \rightarrow L_{\Sigma,t}x \wedge \exists i, i \in \{\mathrm{ENT}/\Sigma\} \exists t', t' > t\ R(i, t', x))$.

The emission axiom (A5) states that if Σ sends a message x at time t, then Σ knows x at time t and some entity i other than that Σ will receive x at time t' subsequent to t. Consider

(A6) $\exists t \exists x (R(\Sigma, t, x) \rightarrow L_{\Sigma,t}x \wedge \exists i, i \in \{\mathrm{ENT}/\Sigma\} \exists t', t' < t\ S(i, t', x))$.

The reception axiom (A6) states that: if Σ receives a message x at time t, then Σ knows x at time t and some entity i other than that Σ has sent x at time t' prior to t. Consider

(A7) (a) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(L_{i,t}x \wedge L_{i,t}k_\Sigma \rightarrow L_{i,t}(e(x, k_\Sigma)))$;
(b) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(L_{i,t}x \wedge L_{i,t}k_\Sigma^1 \rightarrow L_{i,t}(d(x, k_\Sigma^{-1})))$.

Axioms (A7)(a) and (A7)(b) refer to the ability of an entity to encrypt or decrypt a message when it has knowledge of a public or private cryptographic key. Consider

(A8) (a) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(\neg L_{i,t}k_\Sigma \wedge \forall t', t' < t \neg L'_{i,t}(e(x, k_\Sigma)) \wedge \neg(\exists y(R(i,t,y) \wedge C(y, e(x, k_\Sigma)))) \rightarrow \neg L_{i,t}(e(x, k_\Sigma)))$;
(b) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(\neg L_{i,t}k_\Sigma^{-1} \wedge \forall t', t' < t \neg L'_{i,t}(d(x, k_\Sigma^{-1})) \wedge \neg(\exists y(R(i,t,y) \wedge C(y, d(x, k_\Sigma^{-1})))) \rightarrow \neg L_{i,t}(d(x, k_\Sigma^{-1})))$.

Axioms (A8)(a) and (A8)(b) refer to the impossibility of encrypting or decrypting a message without knowledge of the correct key. Axiom (A8)(a) states that if an entity does not know k at t and does not know, prior to t, the encryption $e(x, k_\Sigma)$ and also does not receive $e(x, k_\Sigma)$ at t in a message, then the entity cannot know $e(x, k_\Sigma)$ at time t. Axiom (A8)(b) makes a similar statement for the decryption of a message x without knowledge of the decryption key. Consider

(A9) $\forall t(\forall i, i \in \{\text{ENT}\}L_{i,t}k_i^{-1} \wedge \forall j, j \in \{\text{ENT}/i\}\neg L_{j,t}k_i^{-1})$.

The key secrecy axiom (A9) states that the private keys used by the system are known only to their rightful owners. Consider

(A10) $\exists t \exists x(\exists i, i \in \{\text{ENT}\}L_{i,t}d(x, k_\Sigma^{-1}) \rightarrow L_{\Sigma,t}x)$.

Axiom (A10) states that if an entity knows and can reproduce $d(x, k_\Sigma^{-1})$ and k_Σ at time t; then it knows and can reproduce x, and this implies that this entity knows at time t that Σ knows and can reproduce x prior to t. Consider

(A11) (a) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(L_{i,t}x \wedge L_{i,t}\text{ks}_{(\Sigma,\Psi)} \rightarrow L_{i,t}(E(x, \text{ks}_{(\Sigma,\Psi)})))$;
(b) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(L_{i,t}y \wedge C(y, E(x, \text{ks}_{(\Sigma,\Psi)})) \wedge L_{i,t}\text{ks}_{(\Sigma,\Psi)} \rightarrow L_{i,t}(D(x, \text{ks}_{(\Sigma,\Psi)})))$.

Axiom (A11) refers to the ability an entity has to encrypt or decrypt a message using a symmetric system when it has knowledge of a secret key. Consider

(A12) (a) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(\neg L_{i,t}\text{ks}_{(\Sigma,\Psi)} \wedge \forall t', t' < t, \neg L'_{i,t}(E(x, \text{ks}_{(\Sigma,\Psi)})) \wedge \neg(\exists y(R(i,t,y) \wedge C(y, E(x, \text{ks}_{(\Sigma,\Psi)})))) \rightarrow \neg L_{i,t}(E(x, \text{ks}_{(\Sigma,\Psi)})))$;
(b) $\exists t \exists x \exists i, i \in \{\text{ENT}\}(\neg L_{i,t}\text{ks}_{(\Sigma,\Psi)} \wedge \forall t', t' < t, \neg L'_{i,t}(D(x, \text{ks}_{(\Sigma,\Psi)})) \wedge \neg(\exists y(R(i,t,y) \wedge C(y, D(x, \text{ks}_{(\Sigma,\Psi)})))) \rightarrow \neg L_{i,t}(D(x, \text{ks}_{(\Sigma,\Psi)})))$.

Axiom (A12) refers to the inability of an entity to encrypt or decrypt data without knowledge of the appropriate shared secret key. Consider

(A13) $\forall t((\forall i, i \in \{\text{ENT}/\Sigma, \Psi\}\neg L_{i,t}\text{ks}_{(\Sigma,\Psi)} \wedge \exists j, j \in \{\Sigma, \Psi\}L_{j,t}\text{ks}_{(\Sigma,\Psi)}) \rightarrow \text{ks}_{(\Sigma,\Psi)} \in \{\text{KS}_{\{\Sigma,\Psi\}}\})$.

Axiom (A13) states that only the rightful owners of a shared secret key know that key; this implies that this key is a good key. Consider

(A14) $\forall t((\forall i, i \in \{\text{ENT}/\Sigma, \Psi\}\neg L_{i,t}\text{ss}_{(\Sigma,\Psi)} \wedge \exists j, j \in \{\Sigma, \Psi\}L_{j,t}\text{ss}_{(\Sigma,\Psi)}) \rightarrow \text{ss}_{(\Sigma,\Psi)} \in \{\text{SS}_{\{\Sigma,\Psi\}}\})$.

Axiom (A14) states that only the rightful owners of a shared secret know that secret; this implies that this is a good secret. Finally

(A15) (a) $\exists x \exists t(A(\Sigma, t, \Psi) \rightarrow (L_{\Sigma,t}\text{ss}_{(\Sigma,\Psi)} \wedge \text{ss}_{(\Sigma,\Psi)} \in \{\text{SS}_{\{\Sigma,\Psi\}}\} \wedge R(\Sigma, t, x) \wedge C(x, \text{ss}_{(\Sigma,\Psi)}) \wedge \forall t', t' < t, \neg S(\Sigma, t', x) \rightarrow K_{\Sigma,t}(S(\Psi, t', x))))$;
(b) $\exists x \exists t(A(\Sigma, t, \Psi) \rightarrow (L_{\Sigma,t}k_\Psi \wedge L_{\Sigma,t}x \wedge R(\Sigma, t, y) \wedge C(y, e(\text{x}, k_\Psi^{-1}))) \rightarrow (\forall t', t' < t, K_{\Sigma,t}(S(\Psi, t', y))))$.

(A15)(a) states that if Σ knows a secret $\text{ss}_{(\Sigma,\Psi)}$ that it shares with Ψ (the secret can be fresh), and this secret is a good secret, and Σ receives a message containing $\text{ss}_{(\Sigma,\Psi)}$ at t that it did not send, then Σ knows that Ψ sent this message prior to t.

(A15)(b) states that if Σ knows the public key of Ψ ($k\Psi$) and message x, and if Σ receives a message y containing $e(x, k_\Psi^{-1})$, then Σ knows that Ψ sent message y prior to t.

6.2. Formal Verification of the Proposed Protocol.

To provide assurance that the new AGKA protocol is verifiably secure and trustworthy, a formal verification on its specifications is performed in this section. CSN logic was adopted to perform formal verifications of security protocols in Chapter 6, and is therefore adopted here to perform the formal verification of the new proposed group key agreement protocol.

6.2.1. Goals of the Proposed AGKA Protocol.

The goals of the key-agreement protocol are defined as follows:

Goal 1: $K_{V,t1}$ $(\exists t, t < t1, S(U_i, t, X) \wedge C(X, (D_i, K_i \parallel N_i)))$, for $i = 1, \ldots, n$;

Goal 2: $K_{Ui,t2}$ $(\exists t, t1 < t < t2, S(V, t, X) \wedge C(X, (\text{GK}, c, N_i)))$, for $i = 1, \ldots, n$.

Goal 1 states that the gateway V knows that it will obtain a signed message from U_i containing the ephemeral public key D_i and the concatenation value $K_i \parallel N_i$ prior to the end of Step 1.

Goal 2 states that the low power node U_i will obtain a message from V containing the group key GK, the counter c, and the nonce N_i after Step 1 but before the end of Step 2.

6.2.2. Initial Assumptions.

Consider the following:

(1) $\forall i, \forall t, i \in \{\text{ENT}\}(L_{i,t}Q_V \wedge L_{i,t}Q_i)$;
(2) $\forall i, \forall j, \forall t, i \in \{\text{ENT}/V\}\neg L_{i,t}q_V \wedge j \in \{\text{ENT}/U_i\}\neg L_{j,t}q_i$;
(3) $K_{Ui,t0}$ $(\forall j, \forall t, j \in \{\text{ENT}/U_i\}, t < t1, \neg L_{j,t}N_i)$;

(4) $K_{Ui,t0}$ $(\forall j, \forall t, j \in \{\mathrm{ENT}/U_i\}, t < t1, \neg L_{j,t} R_i \cdot x)$;

(5) $\forall i, \forall j, \forall t, ((t > t1\ i \in \{U_i, V\} L_{i,t} R_i \cdot x) \wedge (j \in \{\mathrm{ENT}/U_i, V\} \neg L_{j,t} R_i \cdot x)) \rightarrow (R_i \cdot x \in \mathrm{SS}_{\{U_i,V\}})$;

(6) $L_{Ui,t0} K_i \wedge K_{Ui,t0} (\forall i, \forall t, i \in \{\mathrm{ENT}/U_i\}, t < t1, \neg L_{j,t} K_i) \rightarrow (K_i \in \mathrm{KS}_{\{Ui,V\}})$.

Assumption (1) states that the public keys Q_V and Q_i, where $i = 1 \ldots n$, are known to all entities.

Assumption (2) states that the private keys of U_i and V are known only to its owner and not known to any other entity.

Assumption (3) refers to the timely revelation of the random nonce N_i by the client U_i.

Assumption (4) refers to the timely revelation of the shared key $R_i \cdot x$ by the client U_i.

Assumption (5) states that only the entities U_i and V will know the shared key $R_i \cdot x$ after Step 1, and this implies that $R_i \cdot x$ is a good secret.

Assumption (6) states that U_i generates the shared MAC key K_i and that U_i knows that no other entity knows this key prior to $t1$, and that the key is a good key.

6.2.3. Formal Analysis

Step 1. $K_{V,t1}(R(V, t1, X) \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x), e(\mathrm{Mes}, q_i)))$ where $\mathrm{Mes} = (D_i \parallel E(K_i \parallel N_i, R_i \cdot x))$.

This states that V knows at time $t1$, it will receive a message X containing the ephemeral public key D_i and encrypted message $E(K_i \parallel N_i, R_i \cdot x)$. And this message will be signed by the private key of the client.

By application of Axiom (A2),

$$R(V, t1, X) \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x), e(\mathrm{Mes}, q_i))). \tag{2}$$

Applying Axiom (A6) and Inference Rule (R2),

$$\begin{aligned} &L_{V,t1} X \wedge K_{V,t1} \left(\exists j, j \in \left\{\frac{\mathrm{ENT}}{V}\right\}\right), \\ &\exists t, t < t1, S(j, t, X) \\ &\qquad \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x), e(\mathrm{Mes}, q_i))). \end{aligned} \tag{3}$$

Applying Inference Rule (R3),

$$\begin{aligned} &K_{V,t1} \left(\exists j, j \in \left\{\frac{ENT}{V}\right\}\right), \\ &\exists t, t < t1, S(j, t, X) \\ &\qquad \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x), e(\mathrm{Mes}, q_i))). \end{aligned} \tag{4}$$

Using Assumption (4) which states that only U_i has knowledge of $R_i \cdot x$ before $t1$ and Assumption (3) which states that only U_i has knowledge of N_i before $t1$,

$$\begin{gathered} K_{V,t1} (\exists t, t < t1), \\ S(U_i, t, X) \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x), e(\mathrm{Mes}, q_i))). \end{gathered} \tag{5}$$

Using Axioms (A7)/(A8)/(A15)(b) which reflect the ability of an entity to authenticate another entity when it has knowledge of its public key and a message with a signature of the message, Assumption (1) which states that the public key of U_i is known to all entities, and Assumption (2) which states the private key of U_i is only known to its owner U_i, we get,

$$\begin{gathered} A(V, t1, U_i) \longrightarrow K_{V,t1} (\exists t, t < t1), \\ S(U_i, t, X) \wedge C(X, (D_i, E(K_i \parallel N_i, R_i \cdot x))). \end{gathered} \tag{6}$$

This shows that the client U_i is authenticated at Step 1 of the protocol since only it could have encrypted *Mes* with its secret key q_i and *Mes* contains the ephemeral public key D_i, and the cipher text $E(K_i \parallel N_i, R_i \cdot x)$.

Using Axioms (A11) and (A12), which reflect the ability of an entity to decrypt a message when it has knowledge of the secret key, and Assumption (5) which states that $R_i \cdot x$ is a good secret key only known to U_i, and V we get,

$$K_{V,t1} (\exists t, t < t1, S(U_i, t, X) \wedge C(X, (D_i, K_i \parallel N_i))), \quad \text{: satisfying Goal 1.} \tag{7}$$

Step 2.

$$\begin{aligned} &K_{Ui,t2} (R(U_i, t2, X)) \\ &\qquad \wedge C(X, (E(\mathrm{GK}, R_i \cdot x), c, N_i, \mathrm{MAC}(\mathrm{Mes2}, K_i))), \end{aligned} \tag{8}$$

where $\mathrm{Mes2} = E(\mathrm{GK}, R_i \cdot x) \| c \| N_i$.

This states that U_i knows at time $t2$ that it will receive a message X containing cipher text $E(\mathrm{GK}, R_i \cdot x)$, nonce N_i, and a message authentication code of this message.

By application of Axiom (A2),

$$\begin{aligned} &R(U_i, t2, X) \\ &\qquad \wedge C(X, (E(\mathrm{GK}, R_i \cdot x), c, N_i, \mathrm{MAC}(\mathrm{Mes2}, K_i))). \end{aligned} \tag{9}$$

Applying Axiom (A6) and Inference Rule (R2),

$$\begin{gathered} L_{Ui,t2} X \wedge K_{Ui,t2} (\exists j, j \in \{\mathrm{ENT}/U_i\}, \exists t, t < t2), \\ S(j, t2, X) \wedge C(X, (E(\mathrm{GK}, R_i \cdot x), c, N_i, \mathrm{MAC}(\mathrm{Mes2}, K_i))). \end{gathered} \tag{10}$$

Applying Inference Rule (R3)

$$\begin{gathered} K_{Ui,t2} (\exists j, j \in \{\mathrm{ENT}/U_i\}, \exists t, t < t2), \\ S(j, t2, X) \wedge C(X, (E(\mathrm{GK}, R_i \cdot x), c, N_i, \mathrm{MAC}(\mathrm{Mes2}, K_i))). \end{gathered} \tag{11}$$

Applying Axioms (A11)/(A12) and (A13) and Assumption (4), and using Assumption (5) which states that $R_i \cdot x$ is a good secret key only known to entities U_i and V:

$$\begin{gathered} K_{Ui,t2} (\exists t, t < t2), \\ S(V, t2, X) \wedge C(X, (E(\mathrm{GK}, R_i \cdot x), c, N_i, \mathrm{MAC}(\mathrm{Mes2}, K_i))). \end{gathered} \tag{12}$$

Using Assumption (3) which states the timely revelation of N_i (after time $t1$) by U_i, we get

$$K_{Ui,t2}\left(\exists t, t1 < t < t2\right), \\ S\left(V, t2, X\right) \wedge C\left(X, \left(E\left(\mathrm{GK}, R_i \cdot x\right), c, N_i, \mathrm{MAC}\left(\mathrm{Mes2}, K_i\right)\right)\right). \quad (13)$$

The client V is authenticated at this point of the protocol since only U_i and V could have encrypted *Mes2* and generate the Message Authentication Code MAC(Mes2, K_i) with its secret key K_i (Axioms (A11)/(A12)/(A15)(a) and Assumption (6)), and *Mes2* contains the cipher text $E(\mathrm{GK}, R_i \cdot x)$, the nonce N_i, and the counter c; therefore

$$A\left(U_i, t1, V\right) \longrightarrow K_{Ui,t2}\left(\exists t, t1 < t < t2\right), \\ S\left(V, t2, X\right) \wedge C\left(X, \left(E\left(\mathrm{GK}, R_i \cdot x\right), c, N_i\right)\right). \quad (14)$$

Using Axioms (A11) and (A12), which reflect the ability of an entity to decrypt a message when it has knowledge of the secret key, and Assumption (5) which states that $R_i \cdot x$ is a good secret key only known to U_i and V, we get

$$K_{Ui,t2}\left(\exists t, t1 < t < t2, S\left(V, t2, X\right) \wedge C\left(X, \left(\mathrm{GK}, c, N_i\right)\right)\right), \\ : \text{satisfying Goal 2.} \quad (15)$$

From the analysis it can be seen that all goals of the proposed group key agreement protocol are achieved and no security flaw is detected. This indicates that the proposed protocol is verifiably secure and trustworthy.

7. Implementation and Performance Evaluation

In order to evaluate the suitability of our protocol in sensor networks, we carried out a set of experiments based on the TelosB [35] and MICAz [36] mote platforms. Table 2 lists the configuration and the architecture of TelosB and MICAz motes.

A low-end PC (1.0 GHz Intel Pentium III processor, 512 MB RAM, and 30 GB hard drive) with a mote attached is used to simulate the gateway. The TelosB mote or the MICAz mote attached to the PC is responsible for transmitting and receiving messages. Using the PC as the security manager enables the security manager to implement all operations by the Java program and store all members' public keys in the local memory device without worrying about memory constraints. This method reduces the execution time of the protocol and releases the memory and power constraints existing in sensor nodes. Most cryptographic algorithms, such as ECDSA, RC5, and Skipjack, are supported by Java, and these algorithms can be found in the Java security packages or the third-party security packages. Another reason for using the PC to simulate the gateway is that the handshaking messages and execution process can be displayed on PC, which eases the researchers in tracing the messages received from the group members and the authentication process during the AGKA protocol.

7.1. Implementation. The implementation is divided into two modules, the client (group member) module and the security manager module.

(i) The client module implements all the operations required by the proposed protocol on the client side, which involves ECC point multiplication, ECDSA signature generation, and MAC generation.

(ii) The security manager module has two parts. The first part powernode.nc is written in nesC code and implemented on the MICAz and TelosB that are attached to the security manager (computer), and the other part is securitymanger.java which is written in Java and implemented on the security manager (computer). These two parts are linked by a Java class MoteIF which enables Java applications to send and receive the message through Universal Asynchronous Receiver/Transmitter (UART).

In software, we implemented our protocol by the use of the nesC programming language and work with the TinySec [37] module and the TinyECC [23] software package, implemented specifically for TinyOS.

TinySec is the first fully implemented link layer security architecture for wireless sensor networks. It is also a research platform that is easily extendable and has been incorporated into higher level protocols. Some well-studied cryptographic primitives are applied in TinySec, such as Message Authentication Codes (MACs), Initialization Vectors (IVs), and Cipher Block Chaining (CBC). It is noteworthy that TinySec was distributed with official releases of TinyOS version 1.x. It has proven that efficient secure communication in wireless sensor networks is a feasible reality. Table 3 summarizes the security characteristics of TinySec.

The TinyECC package supports all elliptic curve operations over prime fields F_p, including point addition, point doubling, and scalar point multiplication, as well as ECDSA operations. It also includes elliptic curve parameters recommended by Stands for Efficient Cryptography Group (SECG), such as secp160k1, secp160r1, and secp160r2. The natural number operations in TinyECC are based on RSAREF2.0 [23, 38].

Bouncy Castle [39] is a collection of APIs used in cryptography. It includes APIs for both the Java and the C# programming languages. It provides a Java library to implement all elliptic curve operations over F_p, including point addition, point doubling, and scalar point multiplication, as well as ECDSA operations. In order to implement ECDSA operations in Java, a number of Bouncy Castle classes are imported into our implementation.

7.2. Experimental Setup. The performance evaluation is performed on both TelosB and MICAz motes. We set two experimental networks, both consist of groups of seven client motes and a single gateway. The performance of the protocol in each network is evaluated. As mentioned in Section 4, some values such as D_i and R_i can be pre-computed before the sensor node *AKGA.SETUP* phase. This is to facilitate a

Table 2: Configuration of TelosB and MICAz motes.

Mote	Manufacturer	Microcontroller	Clock frequency	RAM	Program memory	Data memory	Radio
MICAz	Crossbow	Atmega 128	7.37 MHz	4 kB	128 kB	512 kB	CC2420
TelosB	Moteiv	TI MSP430	4 MHz	10 kB	48 kB	1 MB	CC2420

Table 3: TinySec security characteristics.

	Encryption	Block cipher	Code requirement	Auth. provided	Cost (time/energy)	Key agreement
TinySec	Optional—CBC mode (with CTS)	Skipjack/RC5	7146 Bytes Max.	Yes—CBC-MAC	0.38 ms/9.1% Max.	No

speeding up of the protocol's operation. The impact of the use of precomputation methods will be evaluated.

To enable TelosB and MICAz motes to execute the ECC computations required by the AGKA protocol, the 128-bit and 160-bit ECC parameters recommended by SECG [40] are chosen for use in the tests presented in the experiment, while the 192-bit ECC parameters are not included in the evaluation. This is because the 192-bit ECC requires 48 bytes to represent the point (public key pair) on the curve, which results in 120 bytes payload in the communication message; such large payload size exceeds the maximum TinyOS payload size of 114 bytes.

The following evaluating measurements are used in our performance evaluation experiments:

(i) ROM consumption;

(ii) RAM consumption;

(iii) execution time;

(iv) energy consumption.

7.3. Evaluation Results. A comparison between the results on the TelosB and the results on the MICAz, as well as between the results with pre-computation disabled and with pre-computation enabled, will now be presented.

7.3.1. Execution Time. The execution time can be one of the most meaningful attributes when evaluating security protocols, especially with regard to resource-constrained sensor nodes. The execution time is measured using an oscilloscope.

In comparing two different mote architectures with the same protocol running, it can be seen that the resulting execution time depends on the clock frequency of the microcontroller on the sensor platform.

Figure 3 plots the average execution times for the AGKA protocol implemented on both the TelosB and the MICAz motes with different elliptic curves.

From Figure 3, it can be seen that the value for the execution time on the MICAz mote is about half that of the TelosB mote results, and this can be attributed to the clock frequency of the MICAz being 8 MHz which is double the clock frequency of the TelosB mote. Different elliptic curves affect the execution time of the protocol, and this can be seen in the fact that there is at least a 1.00 second difference

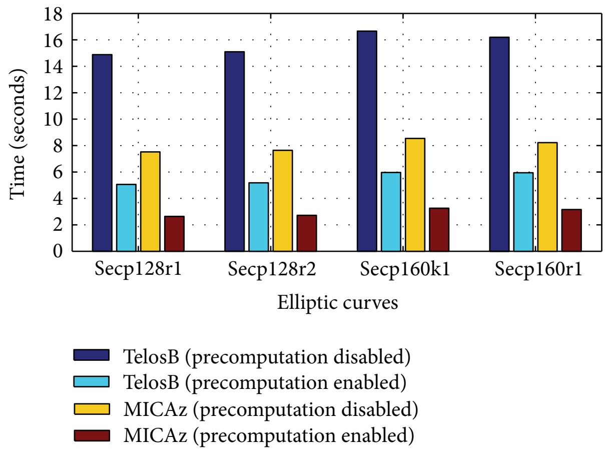


Figure 3: Comparison of the execution time on TelosB and MICAz motes.

with 128-bit elliptic curves implemented compared with 160-bit elliptic curves. It is noticeable that the execution time is significantly reduced when pre-computation is enabled; the reason for this is that two public-key generations are pre-computed and the corresponding results are installed in the memory before the nodes join the network. This saves at least 9 seconds in execution time for the TelosB mote and saves at least 4.50 seconds in execution time for the MICAz mote. The fastest execution time observed from the experimental results is 2.64 seconds, when the AGKA protocol with the secp128k1 elliptic curve was implemented on the MICAz motes. Although pre-computation speeds up the protocol, considerable increases in ROM usage are traded.

7.3.2. Memory Usage. Due to the limited storage available on the sensor nodes, memory usage is an important attribute when evaluating the new key agreement protocol. As already mentioned, the pre-computation method improves the execution speed of the protocol; however, extra memory required is the tradeoff. The check_size script provided by the TinyOS is used to obtain the ROM and RAM sizes required by the AGKA protocol in each experiment.

The experiment evaluates the increases in ROM requirements of the proposed AGKA protocol with pre-computation enabled. Table 4 illustrates the ROM consumption for the

TABLE 4: ROM usage for the AGKA protocol on the TelosB and MICAz motes.

Elliptic curves	First run		Second run		Third run		Fourth run	
	ROM (bytes)	RAM (bytes)	ROM (bytes)	RAM (bytes)	ROM (bytes)	RAM (bytes)	ROM (bytes)	RAM (bytes)
Secp128r1	27716	2492	27938	2560	28216	2628	28514	2702
Secp128r2	27684	2492	27962	2560	28228	2628	28636	2702
Secp160k1	28876	2868	29214	2952	29536	3036	29874	3110
Secp160r1	28844	2868	29182	2952	29516	3036	29842	3110

AGKA protocol on the TelosB and MICAz motes when the pre-computation method is enabled.

It can be seen that the ROM consumption increases with a rise in the number of *AGKA.Setup* algorithms run. The reason for that is discussed in the following. In Step 1, each low-power node U_i uses the offline pre-computing technique to compute $D_i = d_i * p$, $R_i = d_i * Q_V$, $e_i = (K_i \parallel N_i) \oplus R_i \cdot x$ and a signature $\delta_i = \mathrm{Sig}_{U_i}(D_i \parallel e_i)$. Certainly, some tuples $(d_i, D_i, R_i, e_i, \delta_i)$ should be stored in the memory storage of the low-power node U_i in advance. When the proposed protocol plans to run four *AGKA.Setup* algorithms, it will store 4 tuples $(d_i, D_i, R_i, e_i, \delta_i)$ in the memory at beginning and give each tuple a sequence number; for example, the tuple 1 is named as $(d_i, D_i, R_i, e_i, \delta_i)^1$ and tuple 2 is named as $(d_i, D_i, R_i, e_i, \delta_i)^2$. This is the reason why the ROM consumption increases with a rise in the number of *AGKA.Setup* algorithms run. After each run, the proposed protocol will remove the corresponding used tuple $(d_i, D_i, R_i, e_i, \delta_i)$; for example, the protocol will remove the tuple 1 $(d_i, D_i, R_i, e_i, \delta_i)^1$ at the end of the first execution of the *AGKA.Setup* algorithm.

7.3.3. Energy Consumption. Another important evaluation measurement besides the memory usage and the execution time is the energy consumption. The energy consumption by the AGKA protocol is measured by the using of the Agilent mobile communication DC Source (DCS). Figure 4 illustrates the energy consumption for the AGKA protocol implemented on the TelosB and the MICAz motes with specific elliptic curves.

It is shown that the protocol with 128-bit elliptic curves consumes less energy than with 168-bit elliptic curves. This is attributed to a reduction in computational complexity and shorter message size when the protocol uses the 128-bit elliptic curves. With the same elliptic curve, the energy consumed by the protocol on the MICAz is less than that on the TelosB. The reason for this is that the execution times on the MICAz are about half that on the TelosB. Furthermore, with the same elliptic curve, at least 35 μWH of energy is saved with pre-computation enabled on the MICAz mote, while at least 32 μWH of energy is saved with pre-computation enabled on the TelosB mote.

7.4. Limitation and Further Improvement. The comparison results identify that execution time and energy consumption are reduced with short elliptic curves, and those measurements are also improved with pre-computation enabled,

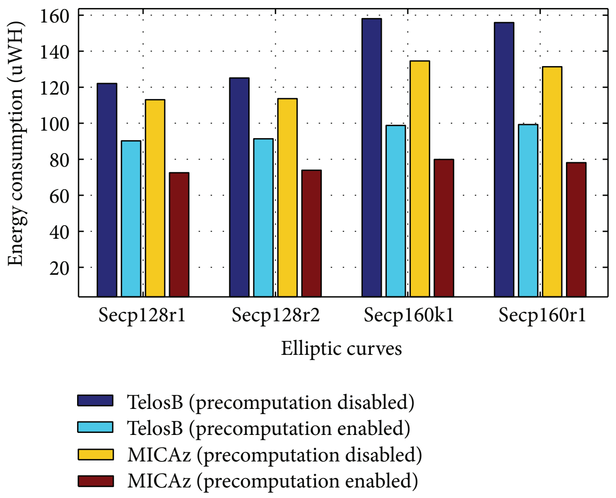


FIGURE 4: Comparison of energy consumption on TelosB and MICAz motes.

while the significant increases in memory usage is the critical tradeoff. Therefore, further improvements and optimizations on memory usage need to be implemented in future work.

The experiment only evaluates the protocol with a group size of seven. With increasing the group size, the execution time will increase. The major reason is that the clients' handshaking packets will queue in the transceiver of the security manager and may cause the jam in the communication channel. Further experiments and simulations on protocol performance versus group size should be carried out.

8. Conclusion and Future Work

In this paper, a secure authenticated group key agreement protocol well suited for wireless sensor networks has been proposed. We showed that the proposed protocol provides forward secrecy and mutual authentication between low-power nodes and the powerful node (gateway). We also demonstrated that the proposed protocol is verifiably secure against node replication attacks and Sybil attacks. Meanwhile, the implementation of the protocol on the TelosB and the MICAz motes was also described in detail. In addition to the implementation of the protocol, a number of evaluation experiments were developed and performed on the motes and described. The experimental results were analyzed based on the following evaluation metrics: execution time, memory usage, and energy consumption. The evaluation results indicate that the protocol is suitable for use with energy-constrained sensor networks. We plan to further investigate the reduction method that can be used to reduce the bit-length of the pre-computed key pairs and signatures, which will in turn reduce the memory usage of the proposed protocol. In addition, we plan to carry out a further evaluation of the proposed protocol with a larger number of group members than used in this study.

Acknowledgments

This research work was supported by the National Natural Science Foundation of China (61103238 and 61003278). This research is also supported by the Fundamental Research Funds for the Central Universities.

References

[1] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41–47, November 2002.

[2] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Security And Privacy*, pp. 197–213, May 2003.

[3] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61, October 2003.

[4] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual authentication and group key agreement for low-power mobile devices," *Computer Communications*, vol. 27, no. 17, pp. 1730–1737, 2004.

[5] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks," in *Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '03)*, pp. 141–150, September 2003.

[6] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 247–260, 2006.

[7] Y. H. Kim, H. Lee, J. H. Park, L. T. Yang, and D. H. Lee, "Key establishment scheme for sensor networks with low communication cost," in *Proceedings of the 4th International Conference on Autonomic and Trusted Computing: Bringing Safe, Self-x and Organic Computing Systems into Reality (ATC '07)*, vol. 4610, pp. 441–448, Hong Kong, Hong Kong, 2007.

[8] L. P. Zhang and Y. Wang, "An ID-based authenticated key agreement protocol for wireless sensor networks," *Journal of Communications*, vol. 5, no. 8, pp. 620–626, 2010.

[9] H. L. Yeh, T. H. Chen, P. C. Liu, T. H. Kim, and H. W. Wei, "A secured authentication protocol for wireless sensor networks using Elliptic Curves Cryptography," *Sensors*, vol. 11, no. 5, pp. 4767–4779, 2011.

[10] K. Ammayappan, A. Negi, V. N. Sastry, and A. K. Das, "An ECC-based two-party authenticated key agreement protocol for mobile Ad Hoc networks," *Journal of Computers*, vol. 11, pp. 2408–2416, 2011.

[11] H. F. Huang, "A new design of access control in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2011, Article ID 412146, 7 pages, 2011.

[12] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[13] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, Washington, DC, USA, October 2003.

[14] J. Nam, S. Kim, and D. Won, "A weakness in the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme for low-power mobile devices," *IEEE Communications Letters*, vol. 9, no. 5, pp. 429–431, 2005.

[15] J. Nam, J. Lee, S. Kim, and D. Won, "DDH-based group key agreement in a mobile environment," *Journal of Systems and Software*, vol. 78, no. 1, pp. 73–83, 2005.

[16] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," *Journal of Cryptology*, vol. 20, no. 1, pp. 85–113, 2007.

[17] Y. M. Tseng, "A secure authenticated group key agreement protocol for resource-limited mobile devices," *Computer Journal*, vol. 50, no. 1, pp. 41–52, 2007.

[18] A. Shamir and Y. Tauman, "Improved on-line/off-line signature schemes," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp. 355–367, Springer, Berlin, Germany, 2001.

[19] Y. Li and T. Newe, "On the logical verification of a group key agreement protocol for resource constrained mobile devices," in *Proceedings of the Australasian Telecommunication Networks and Applications Conference (ATNAC '07)*, pp. 277–281, Christchurch, New Zealand, December 2007.

[20] R. Struik and G. Rasor, *Mandatory ECC Security Algorithm Suite*, IEEE P802.15 Wireless Personal Area Networks, 2002.

[21] SECG, *SEC1: Elliptic Curve Cryptography, Standards For Efficient Cryptography Group*, Certicom Research, 2000.

[22] M. Aydos, T. Yan, and C. K. Koc, "A High-speed ECC-based wireless authentication protocol on an ARM microprocessor," in *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC '00)*, New Orleans, La, USA, 2000.

[23] A. Liu and P. Ning, "TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 245–256, April 2008.

[24] U. Berkeley, *TinyOS Community Forum: An Open Source OS for the Networked Sensor Regime*, 2007.

[25] C. P. Schnorr, "Efficient signature generation by smart cards," in *Proceedings of the 9th Annual International Cryptology Conference (CRYPTO '89)*, vol. 434 of *Lecture Notes in Computer Science*, pp. 688–689, Santa Barbara, Calif, USA, January 1991.

[26] Y. Zhang, W. Liu, Y. Fang, and D. Wu, "Secure localization and authentication in ultra-wideband sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 829–835, 2006.

[27] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing sensor networks with location-based keys," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '05)*, pp. 1909–1914, March 2005.

[28] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks- Specific Requirements-Part 15.4: Wireless*

Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs), 2003.

[29] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[30] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, pp. 36–63, 2001.

[31] J. Douceur, "The sybil attack," in *Procceeding of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2002.

[32] T. Coffey and P. Saidha, "Logic for verifying public-key cryptographic protocols," *IEE Computers and Digital Techniques*, vol. 144, pp. 28–32, 1997.

[33] M. Burrows, M. Abadi, and R. Needham, "Logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[34] J. Hintikka, *Knowledge and Belief: An Introduction to the Logic of Two Notions*, Cornell University Press, Ithaca, NY, USA, 1962.

[35] Crossbow. TelosB product http://www.xbow.com/Products/productsdetails.aspx?sid=147>.

[36] Crossbow-Technology. *MICAz Datasheet*, 2008, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[37] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, Baltimore, Md, USA, November 2004.

[38] *RSA Laboratories, RSAREF: A cryptographic toolkit (version 2.0)*, 1994.

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

[40] CC2431. Texas Instruments Incorporated, 2008, http://focus.ti.com/docs/prod/folders/print/cc2431.html.

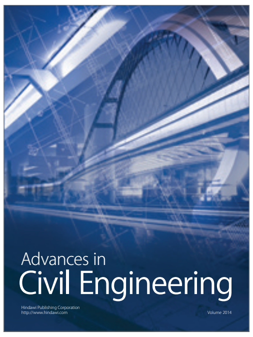
Journal of Engineering

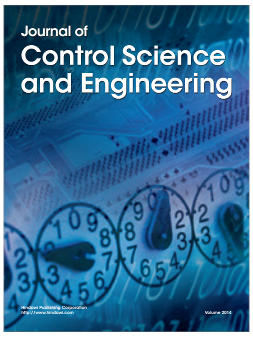
The Scientific World Journal

International Journal of Rotating Machinery

Journal of Sensors

International Journal of Distributed Sensor Networks


Advances in Civil Engineering

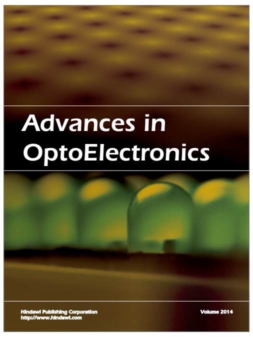

Journal of Control Science and Engineering

Hindawi

Submit your manuscripts at
http://www.hindawi.com

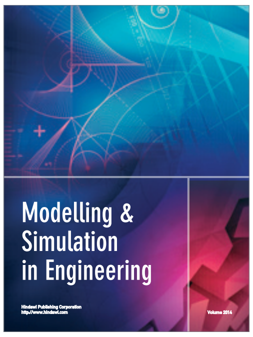
Journal of Robotics

Journal of Electrical and Computer Engineering

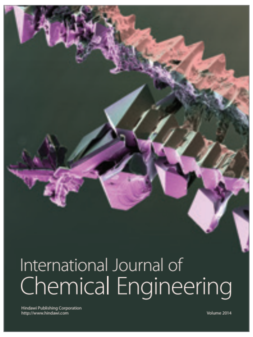

Advances in OptoElectronics

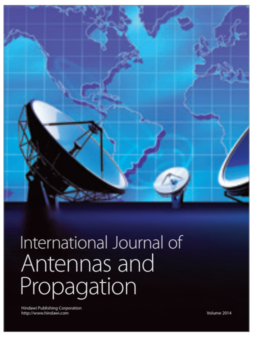
VLSI Design

International Journal of Navigation and Observation

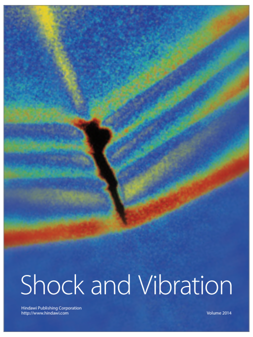

Modelling & Simulation in Engineering

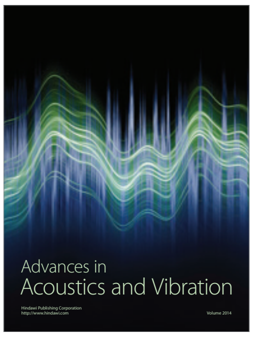
International Journal of Aerospace Engineering


International Journal of Chemical Engineering


International Journal of Antennas and Propagation

Active and Passive Electronic Components


Shock and Vibration


Advances in Acoustics and Vibration