International Journal of Foundations of Computer Science
Vol. 24, No. 5 (2013) 625–653

DOI: 10.1142/S0129054113500214

World Scientific
www.worldscientific.com

# A NOVEL EFFICIENT ACCESS CONTROL SCHEME FOR LARGE-SCALE DISTRIBUTED WIRELESS SENSOR NETWORKS

ASHOK KUMAR DAS

*Center for Security, Theory and Algorithmic Research*
*International Institute of Information Technology, Hyderabad 500 032, India*
*iitkgp.akdas@gmail.com*
*ashok.das@iiit.ac.in*

SANTANU CHATTERJEE

*Research Center Imarat*
*Defence Research and Development Organization, Hyderabad 500 069, India*
*santanu.chatterjee@rcilab.in*

JAMUNA KANTA SING

*Department of Computer Science and Engineering*
*Jadavpur University, Kolkata 700 032, India*
*jksing@ieee.org*

Received 23 December 2011
Accepted 16 January 2013
Communicated by Albert Zomaya

In a wireless sensor network, we often require the deployment of new nodes to extend the lifetime of the network because some sensor nodes may be lost due to power exhaustion problem or they may be also malicious nodes. In order to protect malicious nodes from joining the sensor network, access control mechanism becomes a major challenging problem in the design of sensor network protocols. Existing access control protocols designed for wireless sensor networks require either high communication overheads or they are not scalable due to involvement of the base station during authentication and key establishment processes. In this paper, we propose a new access control scheme for large-scale distributed wireless sensor networks, which not only identifies the identity of each node but it has also ability to differentiate between old nodes and new nodes. The proposed scheme does not require involvement of the base station during authentication and key establishment processes, and it can be easily implemented as a dynamic access control protocol. In addition, our scheme significantly reduces communication costs in order to authenticate neighbor nodes among each other and establish symmetric keys between neighbor nodes as compared with existing approaches. Further, our scheme is secure against different attacks and unconditionally secure against node capture attacks. The simulation results of our scheme using the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool ensure that our scheme is safe.

*Keywords*: Distributed sensor networks; access control; key establishment; authentication; security; RSA; ECC.

## 1. Introduction

Advancement in wireless communications and electronics over the last few years has enabled the development of networks of low-cost, low-power, multifunctional sensors [1]–[3]. In a distributed wireless sensor network, many tiny computing nodes called sensors, are deployed in a deployment area or target field, for the purpose of sensing important data and transmitting those data to nearby *base stations* for further processing. A sensor node is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. Typically, the transmission between the sensors take place by short range radio communications. The base station is computationally resource-rich whereas the sensor nodes are resource-starved. Each of the deployed sensor nodes has the capabilities to collect data and route data back to the base station. Usually, data are routed back to the base station by a multi-hop infrastructure-less architecture through sensor nodes.

New node deployment in sensor networks is necessary due to the loss of power to the sensor nodes after several weeks or months of operation. As a result, this becomes a necessary requirement [4], [5]. As pointed out in [6], [7], a new deployed node may not always be a legitimate node. For example, a malicious node $B$ can be deployed in the vicinity of a node $A$. In the Sybil attack, a compromised node $B$ claims a new identity $C$ in the vicinity of node $A$. In the node replication attack, a copy of the compromised node $B$ can be deployed in the vicinity of node $A$. The Wormhole attack is that attack in which an adversary tunnels packets between nodes $A$ and $B$ so that one node finds a new neighbor node which is actually the image of the other node. Moreover, a deployed sensor node can be a malicious node directly deployed by an attacker or an introduced new node. In order to prevent malicious nodes from joining the existing sensor network, the access control must be deployed in order to control sensor node deployment.

An access control scheme consists of two tasks: *node authentication* and *key establishment*. In *node authentication*, a deployed node needs to prove its identity to its neighbor nodes and also to prove that it has the right to access the existing sensor network. On the other hand, in *key establishment*, the secret shared keys need to be established between a deployed node and its neighbor nodes to protect secure communications among them. Study of access control has received a little attention in research so far [6], [8], [9]. According to the previous researches [6], [8], [9], we list some essential requirements for evaluating an access control scheme designed for wireless sensor networks as follows.

### Security requirements

- *Withstand external devices to eavesdrop or inject data:* An attacker may try to eavesdrop or inject false reports into the sensor networks. An access control protocol must prevent external devices from such eavesdropping or injecting reports into the existing sensor networks.

- *Resilience against node capture attacks:* The resilience against node capture attack of an access control scheme is measured by estimating the fraction of total secure communications that are compromised by a capture of $c$ sensor nodes *not including* the communication in which the compromised nodes are directly involved. In other words, we wish to find out the effect of $c$ sensor nodes being compromised on the rest of the network. For example, for any two non-compromised sensor nodes $u$ and $v$, we need to find out the probability that the adversary can decrypt the secret communications between $u$ and $v$ when $c$ sensor nodes are already compromised. An access control scheme must be highly resilient against node capture attacks.
- *Resilience against new node deployment attacks:* An access control scheme must defend against malicious node deployment attack, Sybil attack, node replication attack and wormhole attack.

**Functionality requirements**

- An access control scheme should support dynamic nodes addition into the existing sensor network after initial deployment of nodes. This is required due to the loss of power to the sensor nodes after several weeks or months of operation. Further, some nodes could be compromised in the network by the attacker. Thus, new nodes are to deployed to extend the lifetime of the network.
- An access control scheme must provide mutual authentication between any two neighbor sensor nodes for pairwise key establishment.
- An access control scheme should provide very high secure connectivity in the network, that is, any two neighbor nodes should be able to establish secret pairwise key between them.
- An access control scheme should be designed in such a way that it requires minimum number of message/packet transmissions during the authentication and key establishment phase in order to use it for practical applications. In addition, it should be computationally efficient, and the storage requirement in each sensor node must be minimum.
- An access control scheme should not involve the base station (central authority) during the authentication and key establishment phase, and dynamic node addition phase to avoid extra communication and computational overheads. Further, an access control scheme should allow any two neighbor nodes to authenticate and establish secret keys between them locally without involving the base station. Thus, no matter how many nodes are deployed in a sensor network, the communication and computational overheads should remain minimum due to only authentication and establishment of secret keys between their neighbor nodes. As a result, the designed access control scheme needs to be scalable, that is, it should support a large-scale network.

Several symmetric key pre-distribution and authentication protocols have been proposed in the literature to protect sensor networks [10]–[17] (see surveys [18]–[20] for details). These protocols can establish symmetric pairwise secret keys between neighbor nodes in the sensor network with simple computations and they can reduce the risk of entire sensor network. However, most of these schemes can not be easily implemented as dynamic access control because the existing old keys as well as broadcasting messages of existing nodes may be updated once new nodes are deployed in the network. It is also possible to construct symmetric key schemes where such update is not needed. For instance, if a network-wide secret master key is shared by all nodes of the WSN, then it is possible to only accept a new node if this node knows the master secret key. The drawback of such a scheme is that if any node is compromised, then that node could use the master secret key to authenticate new malicious nodes. However, this would not happen in a public key scheme, because the nodes would only know the CA's public key. In this paper, we aim to propose an effective access control scheme to secure sensor networks based on public key technique.

The remainder of this paper is sketched as follows. In Sec. 2, we discuss the existing related access control schemes in sensor networks. We then describe our proposed access control scheme in Sec. 3. In Sec. 4, we analyze theoretically the performance and security of our scheme. In Sec. 5, we simulate our scheme for the formal security analysis using the AVISPA tool [21]. We then compare the performances of our scheme with the existing related access control approaches in Sec. 6. Finally, we conclude the paper in Sec. 7.

## 2. Related Work

In this section, we discuss the following existing related access control schemes proposed in sensor networks.

Zhou *et al.* [6] proposed an access control scheme, which is based on elliptic curve cryptographic techniques for sensor network. Their scheme is more efficient than the schemes based on RSA. Their scheme consists of the following phases. In pre-deployment phase, before a sensor network is deployed, the certificate authority (CA) chooses a set of network parameters and preloads a set of node parameters to each sensor node. In node deployment phase, sensor nodes bootstrap themselves and then start establishing communications among them. During the network operation, if some nodes are lost due to power exhaustion problem or some nodes are detected as malicious, then new nodes need to be deployed in the target field. Each new node has a preset bootstrapping time different from that of the previously deployed nodes. In node authentication phase, every new node broadcasts a message to inform its neighbors for its existence. In this phase, there are two kind of handshakes between nodes: the handshake between new nodes, and the handshake between a new node and an old node. The purpose of these handshakes is to authenticate each node with its neighbor nodes as well as to establish secret keys between

neighbor nodes. Zhou *et al.*'s scheme thus supports new nodes joining in the sensor network dynamically and it supports the key establishment in peer to peer manner by using the bootstrapping time in the preloaded certificates present in the nodes. However, the drawback of their scheme is that it introduces high communication overheads due to exchange of so many messages for the entire protocol during node authentication and key establishment phase.

Huang [8] proposed an efficient access control protocol based on elliptic curve cryptography (ECC) and hash chain. This scheme is suitable for resource-constrained sensor nodes and could be easily implemented as a dynamic access control because all the old secrets and broadcasting information in existing deployed nodes should not be updated once a new node is added. This scheme requires involvement of the base station during the initialization phase, and also node authentication and key establishment phase. The limitation of this scheme is that it may not support a large-scale network and thus it may not be scalable.

Further, Kim and Lee [9] showed that Huang's scheme [8] is insecure against the replay attack and an active attack known as new node masquerading attack, and has the lack of hash chain renewability. In order to remedy the weaknesses in Huang's scheme, Kim and Lee [9] proposed an enhanced access control protocol over sensor networks. Their scheme consists of the initialization phase, the authentication and key establishment phase, and the new node injection phase. They have used the renewal of hash chain phase to overcome weaknesses in Huang's scheme for the hash chain exhausted nodes. Due to renewal of hash chain, existing nodes need to communicate with the base station and as a result, it introduces high communication overheads. Similar to Huang's scheme, Kim-Lee's scheme is also not scalable to support a large-scale sensor network. However, Shen *et al.* [22] showed that their scheme is also vulnerable to a fatal weakness, where their scheme is insecure against an active attack, called the man-in-the-middle attack.

Huang [23] proposed a simple dynamic access control protocol to prevent malicious nodes from joining sensor networks. This scheme [23] uses the existing Schnorr signature [24] during authentication phase. This scheme also uses the expiration time for each deployed sensor node so that once the time period elapses, the sensor nodes in the network cannot access any data for future time period. However, if an adversary captures a sensor node and deploys another fake node using the captured node's information, the deployed fake node can still authenticate and establish successfully with its neighbor nodes until expiration time period elapses. Further, this scheme requires high storage and computational overheads.

## 3. The Proposed Scheme

In this section, we first propose a threat model for our scheme. We then provide the notations, which are used in our scheme. We discuss the main motivation behind the development of our novel access control scheme in sensor networks. Finally, we discuss the various phases related to our proposed scheme.

### 3.1. *Threat model*

We assume that due to the hostile environments in the deployment field, sensor nodes can be physically captured by an attacker. We also assume that sensor nodes are not equipped with tamper-resistant hardware due to cost constraints and as a result, once a node is captured by an attacker, all the sensitive data as well as cryptographic information including secret keys stored in its memory are revealed to the attacker. The threats associated to security protocols in sensor networks that are implemented in devices located in "hostile" environments. Most of those threats come from the fact that any potential attacker has physical access to the device implementing the protocols. Hence, it is possible that (s)he has the ability of dumping memory contents, reading firmware etc. Depending on the attacker motivation, hardware invasive or semi-invasive techniques could be applied to recover secret keys. If such keys belong to ordinary devices, then false data could be injected in the WSN. However, we assume that in any case, the base station (BS), which is the central authority (CA) in our case, will not be compromised by an attacker and thus, the attacker does not have any ability to know the private key of the CA. We further assume that an attacker can directly deploy malicious nodes in the deployment field after the initial deployment of nodes.

As in [25], we make use of the Dolev-Yao threat model [26] in which two communicating parties (nodes) communicate over an insecure channel. We adopt the similar threat model for WSNs where the channel is insecure and the end-points (sensor nodes) cannot in general be trustworthy. Finally, we assume that an attacker has only the ability to eavesdrop on all traffic and reply old messages previously delivered.

### 3.2. *Notations*

We use the notations for describing our proposed access control scheme as shown in Table 1. Random nonce is a one-time random bit-string which is usually used to achieve freshness. The public key of CA is $Q = kG$, where $kG = G + G + \ldots + G$ ($k$ times) is called the elliptic curve scalar multiplication in $E_p(a, b)$. If $nG = \mathcal{I}$, where $\mathcal{I}$ is the point at infinity or zero point [27], then we call $n$ is the order of the base point $G$ in $E_p(a, b)$. We may use SHA-1 [28] as the hash function $H(\cdot)$.

### 3.3. *Motivation*

Our proposed access control scheme is motivated by the following considerations. Compared to RSA, elliptic curve cryptography (ECC) can achieve the same level of security with smaller key size [29]. In wireless sensor networks, the transmission energy consumption rate is approximately over three orders of magnitude greater than the energy consumption rate for computing [30]. However, currently there exist few transceivers with lower communication for transmission and reception. An example of such a transceiver is CC2420 [31]. The packet size and the number

Table 1. Notations used in the proposed scheme.

| Symbol | Description |
|---|---|
| $E_p(a, b)$ | An elliptic curve over finite field $GF(p)$: $y^2 = x^3 + ax + b \pmod p$ where $a$ and $b \in Z_p$ are constants such that $4a^3 + 27b^2 \neq 0 \pmod p$, and $p$ is prime. |
| $G$ | A base point on $E_p(a, b)$ |
| $k$ | Private key of CA (only known to CA) |
| $Q$ | $Q = kG$, public key of CA |
| CA | Certification authority (Base station) |
| $id_u$ | Identifier of node $u$ |
| $Cert_u$ | Certificate of node $u$ issued by CA |
| $RN_u$ | Random nonce generated by node $u$ |
| $K_{u,v}$ | Symmetric secret key shared between nodes $u$ and $v$ |
| $u \rightarrow v : M$ | Message $M$ sent from node $u$ to node $v$ |
| $E_K(M)$ | Symmetric key encryption of message $M$ using the key $K$ |
| $D_K(M)$ | Symmetric key decryption of message $M$ using the key $K$ |
| $H(\cdot)$ | Secure one way hash function |
| $A\|\|B$ | Data $A$ concatenates with data $B$ |

of packets in transmission play a crucial role for the performance while designing an access control protocol in sensor networks. If a node is preloaded with the certificate by the CA (in our scheme, it is the base station), then verifying RSA signature in the certificate takes less time than that for ECC signature verification in the certificate, since the signature will be generated in offline by the CA prior to deployment of sensor nodes in the target field. However, compared to a 1024-bit RSA signature [32], if we use ECC-based signature [33], [34] in certificate, then we require only 320-bit signature when 160-bit ECC is used in the proposed scheme. This motivates us to use ECC instead of RSA in our proposed access control scheme so that we can achieve much more energy and bandwidth savings. Our scheme uses the symmetric key cryptographic techniques along with ECC in order to achieve communication and computational efficiency.

Zhou *et al.*'s scheme [6] is not secure against node compromise attack. Further, their scheme assumes that each node can sustain a tolerance time interval before it can be compromised [6], [35] and thus, their scheme may not be convenient for practical implementations [8]. Moreover, their scheme requires high communication and computational overheads due to exchange of so many messages for the entire protocol. Though Huang's scheme [8] is efficient than [6], it is insecure against the replay attack and an active attack such as new node masquerading attack, and it has the lack of hash chain renewability [9]. Kim-Lee's scheme [9] is an enhancement of Huang's scheme [8]. However, their scheme is again vulnerable to active attack. Though the recently proposed Huang's scheme [23] is secure against different attacks except node capture attack, it requires high storage and computational overheads. In this paper, we aim to devise an efficient and secure access control scheme. Our scheme is secure against different attacks and also requires less communication, computation, and storage overheads as compared to other existing schemes.

Further, our scheme does not require any involvement of the base station during the authentication and key establishment phase as well as dynamic node addition phase.

In our scheme, we use a preloaded certificate in each sensor node prior to its deployment in the target field. The preloaded certificate in each node contains a version number which is different for each deployment phase for sensor nodes, a unique certificate serial number, the issuer name (CA, that is, base station), bootstrapping time and the node's identifier. In addition, we use the latest version verified field in each node so that with the help of the bootstrapping time and version present in the certificate, each node will be able to detect a malicious node in the network. After successful authentication with neighbor nodes, each node establishes distinct pairwise symmetric keys with its neighbors for their future secure communication. Further, the resilience against node compromise attack of our scheme is much higher than other existing schemes. In addition, our scheme prevents malicious node deployment attack, sybil attack, node replication attack and wormhole attack as compared to other schemes. Thus, higher security along with lower communication, computation and storage overheads make our scheme much suitable for practical applications in WSN.

### 3.4. *Different phases*

In this section, we describe the various phases related to our scheme.

#### 3.4.1. *Pre-deployment phase*

This phase is performed by the CA (the base station in our scheme) in offline before deployment of sensor nodes in a particular deployment field (target field). The pre-deployment phase consists of the following steps:

Step 1: Prior to deployment of sensor nodes in the target field, the CA chooses a set of *network parameters* which includes (i) a finite field $GF(p)$, where $p$ is a large odd prime of at least 160-bits; (ii) an elliptic curve $E_p(a, b)$, which is the set of all points of $y^2 = x^3 + ax + b \,(\mathrm{mod}\, p)$ such that $a, b \in Z_p = \{0, 1, 2, \ldots, p-1\}$ are constants with $4a^3 + 27b^2 \neq 0 \,(\mathrm{mod}\ p)$; (iii) a base point $G$ in $E_p(a, b)$ whose order is $n$, where $n$ is at least 160-bit number such that $n > 4\sqrt{p}$ as in [6]; (iv) the CA's private key $k \in Z_n^*$, where $Z_n^* = \{1, 2, \ldots, n-1\}$; (v) the CA's public key $Q = kG$.

Step 2: Once the set of network parameters are selected, the CA preloads a set of *node parameters* for each sensor node $u_i$ prior to its deployment in offline. This set contains (i) a unique node identifier $id_{u_i}$ of the node $u_i$; (ii) the elliptic curve $E_p(a, b)$; (iii) the base point $G$; (iv) the certificate $Cert_{u_i}$ for node $u_i$ shown in Table 2; (v) the CA's public key $Q$; (vi) a secure one-way hash function $H(\cdot)$; (vii) a variable called $latest\_version\_verified$.

Table 2. Certificate of a node $u$.

| Field name | Data type |
|---|---|
| Version ($V$) | Integer |
| Certificate serial number ($SN$) | Integer |
| Issuer name ($CA$) | String |
| Bootstrapping time ($T_u$) | Time |
| Node identifier ($id_u$) | String |
| Signature on all above fields ($sig$) | ECDSA signature |

A preloaded public key certificate in each sensor node $u$ prior to its deployment is used to prove its own identity to its neighbor nodes. The purpose of the preloaded certificate in each sensor's memory is that when a new node is deployed in the sensor network, its neighbor nodes can verify the certificate in order to check whether the new node is a legitimate or not. The structure of a preloaded certificate $Cert_u$ in a sensor node $u$ is shown in Table 2. The version field ($V$) is different for each certificate preloaded in sensor nodes. The version $V$ is initialized as follows:

$$V = \begin{cases} 1, \text{ if } u \text{ is deployed during initial deployment phase} \\ i, \text{ if } u \text{ is deployed during } i\text{-th dynamic nodes addition phase.} \end{cases} \quad (1)$$

Each certificate will have a unique serial number ($SN$). In our scheme, the issuer ($CA$) in each certificate is basically the base station. Each certificate will contain a bootstrapping time, $T_u$ so that the node $u$ bootstraps itself to join the sensor network. The certificate will contain the unique node identifier $id_u$ for a sensor node $u$. Finally, the signature on all above fields is computed using the elliptic curve digital signature algorithm (ECDSA) [33], [34] with the help of the private key $k$ of the CA. Since the private key $k$ of the CA is only known to the CA and no one (including the sensor nodes and attackers) can derive $k$ from the public key $Q = kG$ of the CA due to difficulty in solving the elliptic curve discrete logarithm problem (ECDLP), the CA can only generate the valid certificates for deployed sensor nodes.

The purpose of using the variable, $latest\_version\_verified$ is that when a new node is deployed in the network, the old existing nodes will expect that their $latest\_version\_verified$ should be at least the version of the certificate of the new deployed node. This variable of a node $u$ is initialized by the CA prior to deployment as $latest\_version\_verified = V$.

After deployment, once the node $u$ authenticates with its neighbor nodes and establishes the pairwise symmetric secret keys with neighbors, the variable $latest\_version\_verified$ will be updated by $u$ as discussed in Sec. 3.4.2.

### 3.4.2. *Authentication and key establishment phase*

This phase is executed by each deployed sensor node in the network. Initially, a large number of sensor nodes are deployed in the target field. We can deploy some

new sensor nodes in the existing sensor network when some nodes may exhaust their power or they can be compromised by an attacker.

In this phase, each sensor node will authenticate its neighbor nodes in its communication range and also establish secret pairwise symmetric keys with its neighbors after successful authentication. The first task of a deployed sensor node is to locate its neighbor nodes in its communication range. In order to know neighbor nodes, each node $u$ broadcasts a HELLO message containing its own identifier $id_u$. If the node $u$ receives $d$ HELLO messages, then it prepares a list of neighbors as $NL_u = \{v_1, v_2, \ldots, v_d\}$, where $id_{v_1}$, $id_{v_2}$, ..., $id_{v_d}$ are the corresponding identifiers of the neighbor nodes $v_1$, $v_2$, ..., $v_d$, respectively. The node $u$ will authenticate its $d$ neighbors in $NL_u$ and if the authentication be successful, it will establish secret pairwise symmetric keys with them.

Let $u$ and $v$ be two neighbor nodes. The authentication and key establishment procedure between $u$ and $v$ involves the following steps:

Step 1: Node $u$ generates a random nonce $RN_u$ and a random secret number $r_u (< n)$. Note that $n$ is the order of the base point $G$ in the elliptic curve $E_p(a, b)$. $r_u$ is $u$'s private key. It computes the public key $Q_u = r_u G$. $u$ then sends the following message to its neighbor node $v$:

$$u \rightarrow v : id_u || RN_u || Q_u || Cert_u \,. \tag{2}$$

Step 2: After receiving the message from $u$, $v$ verifies the certificate $Cert_u$ of $u$ by means of verifying the signature present in that certificate using the ECDSA signature verification algorithm by the CA's public key $Q$. If the signature verification is successful, then $v$ further verifies $id_u$ with the received identity of the certificate of $u$. If they match, $v$ assumes that node $u$'s identity is valid.

We have then the following three cases for the node $v$.

Case 1: $T_v = T_u$

In this case, the bootstrapping time $T_v$ of node $v$ present in its certificate is equal to the bootstrapping time $T_u$ of node $u$ present in the certificate. If $Cert_v.V = Cert_u.V$ and the value of $latest\_version\_verified$ for node $v$ is also equal to $Cert_u.V$, then node $v$ ensures that node $u$ is deployed during the same deployment phase. Here $Cert_v.V$ represents the version in the certificate of node $v$. Thus, nodes $u$ and $v$ are both new nodes. Node $v$ accepts node $u$ as a legitimate node in the network.

Case 2: $T_v > T_u$

Node $v$ verifies whether $Cert_v.V > Cert_u.V$ and the value of $latest\_version\_verified$ for node $v$ is greater than or equal to the version available in $Cert_u$. If both conditions are satisfied, then $u$ is accepted as a legitimate node by the node $v$. In such case, node $u$ is considered as old node and $v$ is considered as new deployed node in the network.

Case 3: $T_v < T_u$

Node $v$ verifies whether $Cert_v.V < Cert_u.V$ and the value of $latest\_version\_verified$ for node $v$ is less than or equal to the version available in $Cert_u$. If both conditions are satisfied, then $u$ is also accepted as a legitimate node by the node $v$. Here, node $v$ is considered as old node and node $u$ is considered as new deployed node in the network.

Node $v$ now generates a random nonce $RN_v$ and a random secret number $r_v(< n)$, which is considered as the secret key of node $v$. $v$ computes the public key $Q_v = r_vG$. After that $v$ computes $r_vQ_u = (K_{x_{vu}}, K_{y_{vu}})$ and computes the secret symmetric shared key $K_{v,u}$ with node $u$ as

$$K_{v,u} = H(id_u||id_v||RN_u||RN_v||T_u||T_v||K_{x_{vu}}||K_{y_{vu}}). \tag{3}$$

For the challenge-response protocol, $v$ can create a puzzle message, say $PM$ and then computes the encrypted puzzle using its computed key $K_{v,u}$ as $E_{K_{v,u}}(PM)$ and the hash value $H(K_{v,u}||PM||RN_u)$. Finally, $v$ sends the following message to node $u$:

$$v \rightarrow u : id_v||RN_u||RN_v||Q_v||Cert_v||E_{K_{v,u}}(PM)||H(K_{v,u}||PM||RN_u)\,. \tag{4}$$

Step 3: After receiving the message from node $v$, node $u$ proceeds as follows. $u$ first verifies the certificate $Cert_v$ of $v$ received in the message by means of verifying the signature containing in $Cert_v$ using the ECDSA signature verification algorithm by the CA's public key $Q$. If the signature verification is successful, $u$ further verifies the identity $id_v$ of node $v$ with the identity present in $Cert_v$ and the received random nonce $RN_u$ in the message with its own previously generated random nonce for authentication with node $v$. If these verifications are successful, then $u$ assumes that node $v$'s identity is valid.

Similar to Step 2, we have also the following three cases for the node $u$.

Case 1: $T_u = T_v$

If $Cert_u.V = Cert_v.V$ and the value of $latest\_version\_verified$ for node $u$ is equal to $Cert_v.V$, then node $u$ also ensures that node $v$ is deployed during the same deployment phase. In this case, nodes $u$ and $v$ are both new nodes. Node $u$ then accepts node $v$ as a legitimate node in the network.

Case 2: $T_u > T_v$

In this case, $u$ verifies whether $Cert_u.V > Cert_v.V$ and the value of $latest\_version\_verified$ for node $u$ is greater than or equal to the version available in $Cert_v$. If both conditions are satisfied, then $v$ is accepted as a legitimate node by the node $u$, and as a result, node $v$ is considered as old node and node $u$ is considered as new deployed node in the network.

Case 3: $T_u < T_v$

Node $v$ verifies whether $Cert_u.V < Cert_v.V$ and the value of $latest\_version\_verified$ for node $u$ is less than or equal to the version available in $Cert_v$. Now, if both these conditions are satisfied, then $v$ is also accepted as a legitimate node by the node $u$. Hence, node $u$ is considered as old node and node $v$ is considered as new deployed node in the network.

Once the node $u$ considers the node $v$ as a legitimate node, $u$ computes $r_u Q_v = (K_{x_{uv}}, K_{y_{uv}})$. $u$ then computes the symmetric secret key $K_{u,v}$ shared with $v$ as

$$K_{u,v} = H(id_u||id_v||RN_u||RN_v||T_u||T_v||K_{x_{uv}}||K_{y_{uv}}). \tag{5}$$

In order to solve the puzzle, $u$ first decrypts the encrypted puzzle $E_{K_{v,u}}(PM)$ using its own computed key $K_{u,v}$ and retrieves the puzzle as $PM' = D_{K_{u,v}}(E_{K_{v,u}}(PM))$ and computes the hash value using the retrieved puzzle $PM'$, its own computed key $K_{u,v}$ and its own random nonce $RN_u$ as $H(K_{u,v}\ ||PM'||RN_u)$. If this computed hash value matches with the incoming hash value $H(K_{v,u}||\ PM\ ||RN_u)$ received in the message, then $u$ ensures that it has solved the puzzle successfully. $u$ then executes Step 4.

Step 4: To make sure that node $u$ has successfully computed the same secret key shared with node $v$ and solved the puzzle sent by node $v$, it computes the hash value as $H(K_{u,v}||PM'||RN_v)$. Node $u$ then sends the following message to $v$ as a response of the previous challenge sent by node $v$:

$$u \rightarrow v : id_u||id_v||RN_v||H(K_{u,v}||PM'||RN_v)\,. \tag{6}$$

Step 5: After receiving the message from node $u$, node $v$ first verifies whether the random nonce received in the message matches with its own random nonce $RN_v$. If so, node $v$ computes the hash value $H(K_{v,u}||PM||RN_v)$ using its own computed key $K_{v,u}$, previously created puzzle $PM$ and random nonce $RN_v$. If this hash value matches with the hash value received in the message, then $v$ also ensures that node $u$ has the correct same secret key shared between them and it has solved the puzzle successfully.

Node $u$ stores the secret key $K_{u,v}$ in its memory for future secret communication with node $v$. Similarly, node $v$ also stores the secret key $K_{v,u}$ in its memory for future secret communication with node $u$.

This phase is summarized in Table 3. We note that our scheme provides mutual authentication between any two neighbor nodes in the network.

In order to thwart against node capture attacks, node $u$ deletes $RN_u$, $r_u$, $Q_u$ from its memory and node $v$ also deletes $RN_v$, $r_v$, $Q_v$ immediately after their key establishment. After successful authentication and key establishment with neighbor nodes, each sensor node $u$ will first set its variable $latest\_version\_verified$ by the latest version of a certificate of its neighbors nodes and then increment it. For

Table 3. Authentication and key establishment phase of our proposed scheme.

| Node $u$ | Node $v$ |
|---|---|
| 1. Generates $RN_u$, $r_u$. Computes $Q_u = r_u G$. $id_u \| \| RN_u \| \| Q_u \| \| Cert_u \xrightarrow{}$ | |
| | 2. Verifies certificate $Cert_u$. Verifies version, bootstrapping time present in $Cert_u$, $latest\_version\_verified$ of $v$. 3. If above verifications hold, generates $RN_v$, $r_v$, computes $r_v Q_u = (K_{x_{vu}}, K_{y_{vu}})$, $K_{v,u} = H(id_u \| \| id_v \| \| RN_u \| \| RN_v \| \| T_u \| \| T_v \| \| K_{x_{vu}} \| \| K_{y_{vu}})$. Generates puzzle $PM$, computes $E_{K_{v,u}}(PM)$ and $H(K_{v,u} \| \| PM \| \| RN_u)$. $id_v \| \| RN_u \| \| RN_v \| \| Q_v \| \| Cert_v \| \| E_{K_{v,u}}(PM) \| \| H(K_{v,u} \| \| PM \| \| RN_u) \xleftarrow{}$ |
| 4. Verifies $Cert_v$. Verifies version, bootstrapping time present in $Cert_v$, $latest\_version\_verified$ of $u$. 5. If above verifications hold, computes $r_u Q_v = (K_{x_{uv}}, K_{y_{uv}})$, $K_{u,v} = H(id_u \| \| id_v \| \| RN_u \| \| RN_v \| \| T_u \| \| T_v \| \| K_{x_{uv}} \| \| K_{y_{uv}})$. Retrieves puzzle as $PM' = D_{K_{u,v}}(E_{K_{v,u}}(PM))$, and computes $H(K_{u,v} \| \| PM' \| \| RN_u)$. Verifies puzzle comparing this hash value with received hash value in message. If verification is successful, $u$ accepts $v$ as a legitimate node and then computes $H(K_{u,v} \| \| PM' \| \| RN_v)$. $id_u \| \| id_v \| \| RN_v \| \| H(K_{u,v} \| \| PM' \| \| RN_v) \xrightarrow{}$ | |
| | 6. Verifies received random nonce $RN_v$ with its own random nonce. If so, computes $H(K_{v,u} \| \| PM \| \| RN_v)$ and compares it with the received hash value in the message. If they match, $v$ accepts $u$ as a legitimate node. |

example, $u$ updates its variable $latest\_version\_verified$ by 2 after authenticating and establishing secret keys with its neighbors during the initial deployment phase. This means that $u$ is now ready to authenticate and establish secret keys with only a new deployed node during $i$-th dynamic nodes addition phase ($i \geq 2$) whose certificate's version is greater than or equal to 2.

### 3.4.3. *Dynamic nodes addition phase*

Deployment of new nodes in sensor networks is inevitable due to the loss of sensor nodes because after several weeks or months of operation some sensor nodes in the

network may exhaust their power. Even some nodes may be compromised and we need to deploy new nodes in the network. Assume that one or more new nodes to be deployed in a dynamic nodes addition phase.

Let a new sensor node $u$ be deployed in the $i$-th dynamic nodes addition phase. Prior to its deployment in the target field, using the pre-deployment phase discussed in Sec. 3.4.1, the CA will preload a set of *node parameters* in offline. This set contains (i) a unique node identifier $id_u$ of the node $u$; (ii) the elliptic curve $E_p(a, b)$; (iii) the base point $G$; (iv) the certificate $Cert_u$ for node $u$ shown in Table 2; (v) the CA's public key $Q$; (vi) a secure one-way hash function $H(\cdot)$; (vii) a variable called $latest\_version\_verified$ initialized to $i$, for the $i$-th dynamic nodes addition phase.

After deployment, $u$ authenticates and establishes pairwise symmetric secret keys with its neighbor nodes using the authentication and key establishment phase as described in Sec. 3.4.2. Thus, in our scheme, dynamic nodes addition phase is very simple and efficient, and it does not require any involvement of the base station after deployment.

## 4. Analysis of the Proposed Scheme

In this section, we perform the functionality analysis and security analysis including the formal security analysis of our proposed access control scheme.

### 4.1. *Functionality analysis*

#### 4.1.1. *Correctness proof*

We show that any two neighbor nodes in the network can establish correctly a symmetric secret key shared between them. This correctness proof is given in Theorem 1.

**Theorem 1.** *In our access control scheme, any two neighbor sensor nodes always establish correctly the same symmetric secret key shared between them after successful mutual authentication between them.*

**Proof.** Let $u$ and $v$ be two neighbor sensor nodes. From Sec. 3.4.2, it is clear that node $u$ establishes a symmetric secret key with node $v$ as $K_{u,v} = H(id_u||id_v||RN_u||RN_v||T_u||T_v||K_{x_{uv}}||K_{y_{uv}})$, where $r_uQ_v = (K_{x_{uv}}, K_{y_{uv}})$. On the other hand, node $v$ also establishes a symmetric secret key with node $u$ as $K_{v,u} = H(id_u||id_v||RN_u\ ||RN_v||T_u||T_v||K_{x_{vu}}||K_{y_{vu}})$, where $r_vQ_u = (K_{x_{vu}}, K_{y_{vu}})$. In order to prove $K_{u,v} = K_{v,u}$, we have to show that $K_{x_{uv}} = K_{x_{vu}}$ and $K_{y_{uv}} = K_{y_{vu}}$, that is, it suffices to show that $r_uQ_v = r_vQ_u$.

Now we have,

$$\begin{aligned} r_u Q_v &= r_u(r_v G) \\ &= r_v(r_u G) \\ &= r_v Q_u. \end{aligned}$$

Hence, $u$ and $v$ always establish correctly the same symmetric secret key shared between them. □

4.1.2. *Secure connectivity*

We measure the secure connectivity of our access control scheme by the probability that any two neighbor sensor nodes can establish a secret pairwise key (secure link) between them. It is noted that any two neighbor sensor nodes can establish a symmetric secret key shared between them, if they authenticate each other successfully. As a result, the secure connectivity probability becomes 1.0. Hence, our scheme provides 100% secure connectivity in the network.

4.1.3. *Computational overhead*

We consider the computational complexity for a sensor node $u$ for authentication and key establishment of a symmetric secret key with a neighbor node $v$. Let $T_{ecm}$, $T_h$, $T_i$, $T_{enc}$, and $T_{dec}$ denote the time for performing an elliptic curve scalar multiplication, a one-way hash function $H(\cdot)$, a modular inverse, a symmetric key encryption, and a symmetric key decryption, respectively. Node $u$ needs the computational complexity due to the following: computation of the public key $Q_u$, which requires $T_{ecm}$; verification of the certificate of $v$, which requires $2T_{ecm} + T_i + T_h$; computation of symmetric key, which requires $T_{ecm} + T_h$; computation for the challenge-response protocol, which requires $T_{enc}/T_{dec} + 2T_h$. Summing up all these complexity terms together, $u$ requires $4T_{ecm} + T_i + T_{enc}/T_{dec} + 4T_h$.

4.1.4. *Communication overhead*

It is clear from the authentication and key establishment phase described in Sec. 3.4.2 that any two neighbor nodes need to exchange only three messages in order to authenticate with each other and establish a symmetric secret key between them. We use Table 4 for different parameters used for our access control protocol. We have calculated the size of each message in bits and then the number of packets required to transmit per message. For calculating the number of packets required for a message, we have considered the packet size of 128 bytes, that is, 1024 bits for CC2420. From Table 5, we see that our scheme requires only 4 packet transmissions during the the authentication and key establishment phase.

Table 4. Bit size of the parameters used for our proposed access control scheme.

| Type | Bit size |
|---|---|
| Version ($V$) | 8 |
| Serial number ($SN$) | 32 |
| Issuer name ($CA$) | 8 |
| Bootstrapping time ($T_u$) | 32 |
| Node identifier ($id_u$) | 16 |
| Signature (ECDSA) | 320 |
| Random nonce ($RN_u$) | 32 |
| Hash digest | 160 |
| Latest version verified ($latest\_version\_verified$) | 8 |

Table 5. Message size in bits and number of packets needed to transmit during authentication and key establishment phase of our proposed access control scheme.

| Message | # bits required | #packets required |
|---|---|---|
| $id_u \| \| RN_u \| \| Q_u \| \| Cert_u$ | 784 | 1 |
| $id_v \| \| RN_u \| \| RN_v \| \| Q_v \| \| Cert_v \| \| E_{K_{v,u}}(PM)$ $\| \| H(K_{v,u} \| \| PM \| \| RN_u)$ | 1104 | 2 |
| $id_u \| \| id_v \| \| RN_v \| \| H(K_{u,v} \| \| PM' \| \| RN_v)$ | 224 | 1 |

#### 4.1.5. *Storage overhead*

Prior to deployment in the pre-deployment phase, each sensor node $u$ requires the storage space due to storing the node parameters: a unique node identifier $id_u$ of the node $u$; the elliptic curve $E_p(a, b)$; the base point $G$; the certificate $Cert_u$ for node $u$ shown in Table 2; the CA's public key $Q$; and the variable $latest\_version\_verified$. Using Table 4, the storage overhead for each node $u$ of our scheme becomes 1560 bits.

### 4.2. *Security analysis*

We show that our scheme can effectively defend various attacks, which are discussed in the following subsections.

#### 4.2.1. *Withstand external devices to eavesdrop or inject data*

In our scheme, when a new sensor node passes the successful authentication procedure, it has already established the shared symmetric secret keys with its neighbors. These secret keys can be used to secure communications among its neighbor sensor nodes. As a result, external devices are prevented from eavesdropping or injecting false reports into the sensor network.

4.2.2. *Resilience against node capture attacks*

In the following, we now evaluate the ability of our access control scheme to tolerate compromised sensor nodes in the network. Let $P_e(c)$ denote the probability that an adversary compromises a fraction of total secure communications by capturing $c$ number of sensor nodes in the network. If $P_e(c) = 0$, we call an access control scheme as *unconditionally secure against node capture attack.*

Our scheme ensures that any two neighbor sensor nodes can establish a symmetric secret key between them for their future secure communications. Moreover, since the pairwise key between any two neighbor nodes is generated using random private keys by the nodes, the established secret keys among sensor nodes are different throughout the network. If we have more neighbor nodes of each node, the number of secure links in the network also increases. Now, if a sensor node is captured or compromised, then the attacker is only able to compromise secret keys of its neighbor nodes. The effect of a compromised node does not lead to compromise secure communications among non-compromised nodes in the network. In other words, even some nodes are compromised, the security of the entire sensor network is not compromised, that is, non-compromised nodes still communicate among each other with 100% security. Hence, our scheme is unconditionally secure against node capture attack.

4.2.3. *Resilience against new node deployment attacks*

With the help of formal security proof we show that our access control scheme can prevent different new node deployment attacks by an adversary.

**Theorem 2.** *Under the elliptic curve discrete logarithm problem (ECDLP) assumption, our proposed scheme prevents malicious node deployment attack, Sybil attack, node replication attack and Wormhole attack by an adversary.*

**Proof.** We first define the elliptic curve discrete logarithm problem (ECDLP) formally as follows [36]. Let $E_p(a, b)$ be an elliptic curve modulo a prime $p$. Let $P \in E_p(a, b)$ and $Q = kP \in E_p(a, b)$ be two points, where $k \in_R Z_p$ ($a \in_R S$ denotes that $a$ is chosen randomly from $S$), and $Z_p = \{0, 1, 2, \ldots, p-1\}$.

*Instance:* $(P, Q, r)$ for some $k, r \in_R Z_p$.

*Output:* **YES**, if $Q = rP$, i.e., $k = r$, and **NO**, otherwise.

Consider the following two distributions

$$\Delta_{real} = \{k \in_R Z_p, X = P, Y = Q(= kP), Z = k : (X, Y, Z)\},$$
$$\Delta_{rand} = \{k, r \in_R Z_p, X = P, Y = Q(= kP), Z = r : (X, Y, Z)\}.$$

The advantage of any probabilistic, polynomial-time, 0/1-valued (false/true-valued) distinguisher $\mathcal{D}$ in solving ECDLP on $E_p(a, b)$ is defined as

$$Adv^{ECDLP}_{\mathcal{D}, E_p(a,b)} = |Pr[(X, Y, Z) \leftarrow \Delta_{real} : \mathcal{D}(X, Y, Z) = 1]$$
$$- Pr[(X, Y, Z) \leftarrow \Delta_{rand} : \mathcal{D}(X, Y, Z) = 1]|,$$

where the probability $Pr[\cdot]$ is taken over the random choices of $k$ and $r$. $\mathcal{D}$ is a $(t, \epsilon)$-ECDLP distinguisher for $E_p(a, b)$ if $\mathcal{D}$ runs at most in time $t$ such that $Adv^{ECDLP}_{\mathcal{D}, E_p(a,b)}(t) \geq \epsilon$.

*ECDLP assumption:* There exists no $(t, \epsilon)$-ECDLP distinguisher for $E_p(a, b)$. In other words, for every probabilistic, polynomial-time 0/1-valued distinguisher $\mathcal{D}$, we have $Adv^{ECDLP}_{\mathcal{D}, E_p(a,b)}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

We use the method of contradiction proof [37] for proving this theorem. Assume that an adversary can determine the private key $k$ of the CA from the public elliptic curve parameters and the CA's public key $Q = kG$. Thus, the adversary has the ability to generate a valid certificate with proper bootstrapping time, version of the certificate, certificate serial number and node identifier for a new deployed malicious node in the network on behalf of the CA.

We define the following two oracles for the adversary $\mathcal{A}$:

- **RevealKey:** This unconditionally outputs the private key $k$ of the CA using the public elliptic curve parameters and the CA's public key $Q = kG$.
- **CreateCertificate:** This query is allowed at any time during the attacker's execution. This oracle computes a valid certificate of a new deployed malicious node on behalf of the CA.

Now, the adversary $\mathcal{A}$ can run the following experimental algorithm, say, $Experiment^{ECDLP}_{ACP,\mathcal{A}}$ for the proposed access control protocol $ACP$.

**Algorithm 1** $Experiment^{ECDLP}_{ACP,\mathcal{A}}$

Call RevealKey oracle. Let $k' \leftarrow RevealKey(E_p(a, b), n, G, Q)$ be the output, which be the private key of the CA.
Call CreateCertificate oracle. Let $Cert_w \leftarrow CreateCertificate(k', E_p(a, b), n, G)$ be the output, which be a certificate generated for a new malicious deployed node $w$ by the adversary $\mathcal{A}$ using the private key computed by the RevealKey oracle.
**if** (CreateCertificate oracle generates correctly a valid certificate for the node $w$) **then**
  **return** 1
**else**
  **return** 0
**end if**

We define $Succ^{ECDLP}_{ACP,\mathcal{A}} = 2Pr[Experiment^{ECDLP}_{ACP,\mathcal{A}} = 1] - 1$. Then the advantage function is defined by

$$Adv^{ECDLP}_{ACP}(t, q_R, q_C) = \max_{\mathcal{A}}\{Succ^{ECDLP}_{ACP,\mathcal{A}}\},$$

where the maximum is taken over all $\mathcal{A}$ with execution time $t$, $q_R$ is the number of queries to the RevealKey oracle and $q_C$ the number of queries to the CreateCertificate oracle. We say that our proposed access control scheme is secure against

malicious node deployment attack, Sybil attack, node replication attack and Wormhole attack, if $Adv_{ACP}^{ECDLP}(t, q_R, q_C) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

Note that according to the experiment given in Algorithm 1, the adversary $\mathcal{A}$ can compute correctly the private key $k$ of the CA. Once the private key $k$ of the CA is known to the adversary $\mathcal{A}$, he/she can easily generate a valid certificate for a new malicious deployed node. Now, the new deployed node can authenticate and establish successfully the secret pairwise keys with its neighbor nodes in the network. Thus, by including the proper bootstrapping time and version of the certificate a new malicious node can easily join to the network. However, this is a contradiction because of the ECDLP assumption defined above. Computing the private key $k$ of the CA from the public elliptic curve parameters and the public key $Q = kG$ of the CA, it is computationally infeasible problem. As a result, $Adv_{ACP}^{ECDLP}(t, q_R, q_C) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Because the adversary does not know the private key of the CA, he/she cannot falsify certificates for malicious nodes. Since our scheme uses the proper bootstrapping time and version of the certificate, a new malicious deployed node is not allowed to join the sensor network. New malicious deployed nodes are prevented from falsifying the latest bootstrapping time and version of the certificate because they do not match with their certificates. Hence, our access control scheme defends against malicious node deployment attack, Sybil attack, node replication attack and Wormhole attack by an adversary. □

***Remark.*** Let $u$ and $v$ be two nodes. If $T_u < T_v$, then $u$ is considered as old node and $v$ as new. In our scheme, node $u$ rejects new node $v$, if $Cert_v.V \not> Cert_u.V$ and the value of $latest\_version\_verified$ for node $v$ is not greater than or equal to the version available in $u$'s certificate $Cert_u$. Moreover, if a non-compromised node, say $u$ wants to communicate with compromised node, say $v$ or vice-versa, then in our scheme both nodes $u$ and $v$ must authenticate to each other and also establish a symmetric key between them so that they can use that key for future secret communication. However, $v$ could not be able to pass successfully authentication procedure with $u$ due to timestamp, version and the value of $latest\_version\_verified$ verifications, because $v$ will not have a valid certificate with proper timestamp and version in the certificate. Hence, our scheme has ability to defend such cases.

## 5. Simulation for Formal Security Analysis of the Proposed Protocol

We have verified and analyzed the communication part of our protocol by using a model checker, known as AVISPA (Automated Validation of Internet Security Protocols and Applications) [21]. Model checking methods are used to search for states of the system whether some properties are violated or not [38]. Using model checking it can only specify that whether an attack is possible or not for the specified system which states the number of principals, number of concurrent protocol runs, principal roles and so on. Therefore, the way in which the system is specified plays a

crucial part in detecting these attacks. Model checking tools have been successfully employed to detect attacks on security protocols [39]. We have used AVISPA model checker for our formal security analysis. AVISPA implements four different tools and abstraction-based methods which are integrated through a common high level protocol specific language, known as HLPSL [40]. A static analysis is performed to check the executability of the protocol, and then the protocol and the intruder actions are compiled into an intermediate format (IF). This intermediate format is the starting point for the automated protocol analysis techniques. IF is a lower-level language than HLPSL and is read directly by the back-ends of the AVISPA tool. The back-ends are used to provide protocol falsification, bounded and unbounded verification. The first back-end, called the On-the-fly Model-Checker (OFMC), performs several symbolic techniques to explore the state space in a demand-driven way. The second back-end, CL-AtSe, known as the Constraint-Logic-based Attack Searcher, provides a translation from any security protocol specification written as transition relation in intermediate format into a set of constraints which are effectively used to find whether there are attacks on protocols. Third back-end, called the SAT-based Model-Checker (SATMC), builds a propositional formula which is then fed to a state-of-the-art SAT solver and any model found is translated back into an attack. Finally, TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) is the last back-end, which approximates the intruder knowledge by using regular tree languages.

Protocols are described using a high level language, called the HLPSL language, which is a role-oriented language. Each principal is implemented in transitional roles in which the transitions of a principal takes place during the protocol run as specified. The protocol session is a parallel composition of these transitional roles. The intruder is modeled using the Dolev-Yao threat model [26] (according to our threat model in Sec. 3.1) with the possibility for the intruder to assume a legitimate role in a protocol run. The role system defines the number of sessions, the number of principals and the roles. In this analysis, the role system consists of two sessions in which the principals, say Alice and Bob take alternatively the roles for initiator and responder, respectively.

We have implemented our scheme under AVISPA model checkers using SPAN (Security Protocol ANimator for AVISPA). Figure 1 shows the specification in HLPSL language for the initiator, sensor node $u$. Node $u$ first sends the message $\langle id_u||RN_u||Q_u||Cert_u\rangle$ to the responder, node $v$. $u$ receives the message $\langle id_v||RN_u||RN_v||Q_v||Cert_v||E_{K_{v,u}}(PM)||H(K_{v,u}||PM||RN_u)\rangle$ from $v$ and then sends the message $\langle id_u||id_v||RN_v||H(K_{u,v}||PM'||RN_v)\rangle$ to $v$.

In Fig. 2, the specification in HLPSL language for the responder (node $v$) is given. After receiving the message from $u$, it sends the message $\langle id_v||RN_u||RN_v||Q_v||Cert_v||E_{K_{v,u}}(PM)||H(K_{v,u}||PM||RN_u)\rangle$ to $u$. It then waits for an acknowledgment $\langle id_u||id_v||RN_v||H(K_{u,v}||PM'||RN_v)\rangle$ from $u$. In the roles, mutual authentication between $u$ and $v$ are performed.

```
role alice (U,V   : agent,
      Kca : public_key,
      % inv(Kca) = private key of CA
      % H is hash function
       H    : function,
      F    : function,
      IDu, Vru, Cu, Sru, Tu: text,
      Snd, Rcv: channel(dy))
played_by U
def=
 local State  : nat,
      IDv, Vrv, Cv, Srv,Tv, Qu,  Qv, P, Nv, Nu, G,
      Ru, Rv :  text
 const alice_bob_nv,  bob_alice_nu, subs1, subs2 : protocol_id
 init  State := 0

 transition
  1. State  = 0  ∧ Rcv(start) =|>
    State' := 1  ∧ Nu' := new()
                 ∧ Ru' := new()
                  ∧ Qu' := F(Ru'.G)
                 ∧ secret({Ru'}, subs1, U)
                 ∧ Snd(U. Nu'.Qu'.Vru.Sru.Cu.Tu.IDu.
                   {Vru.Sru.Cu.Tu.IDu}_inv(Kca))
                 ∧ witness(U, V, bob_alice_nu, Nu')
  2. State  = 1 ∧ Rcv(V.Nv'.Nu'.Qv'.Vrv.Srv.Cv.Tv.IDv
                .{ Vrv.Srv.Cv.Tv.IDv}_inv(Kca).
                {P'}_H(IDu.IDv.Nu'.Nv'.Tu.Tv.Qu').
                H(H(IDu.IDv.Nu'.Nv'.Tu.Tv.Qu').P'.Nu')) =|>
  State' := 3 ∧ Snd(U.V.Nv'.H(H(IDv.IDu.Nv'.Nu'.Tv.Tu.Qv).P'.Nv'))
                ∧ request(V,U,alice_bob_nv,Nv')
end role
```

Fig. 1. Role specification for the initiator (sensor node $u$) for our protocol.

We assume that the intruder has knowledge of all public parameters including $ID_u$, $ID_v$, $T_u$, $T_v$, $RN_u$, $RN_v$, $h(\cdot)$ and the public key of CA. We have simulated our protocol using SPAN for OFMC and CL-AtSe model checkers. The results of the analysis using OFMC and CL-AtSe are shown in Figs. 3 and 4. We have obtained the test results as follows:

- OFMC reports the protocol is safe.
- CL-AtSe reports the protocol is also safe.

## 6. Performance Comparison with Related Schemes

In this section, we compare the performance of our scheme with Zhou *et al.*'s scheme [6], Huang's scheme [8], Kim-Lee's scheme [9] and Huang's scheme [23].

The description of the notations $T_{ecm}$, $T_i$, $T_h$, $T_{enc}$ and $T_{dec}$ are already provided in Sec. 4.1.3. We denote $T_{ecenc}$ and $T_{ecdec}$ as the time for performing public key encryption and decryption using the ECC encryption and decryption algorithms, respectively. $T_{eca}$ is the time for performing the elliptic curve point addition in $E_P(a, b)$, $T_{mul}$ is the time for executing a modular multiplication overt the finite field $GF(2^{163})$ and $T_{add}$ is the time for executing a modular addition in $GF(2^{163})$.

The elliptic curve scalar multiplication and modular inverse operations are computational expensive, whereas the hashing computation is much efficient than those

```
role bob (V,U : agent,
        Kca : public_key,
        % inv(Kca) = private key of CA
        % H is hash function
         H: function,
        F: function,
        IDv, Vrv, Cv, Srv,Tv : text,
        Snd, Rcv: channel(dy))
played_by V
def=
 local State  : nat,
      IDu, Vru, Cu, Sru, Tu, Qv, Qu, P, Nv, Nu,
      G, Ru, Rv : text
 const bob_alice_nu, alice_bob_nv,
       subs1, subs2  : protocol_id
 init State := 0

 transition
  1. State  = 0 /\ Rcv(U.Nu'.Qu'.Vru.Sru.Cu.Tu.IDu.
                {Vru.Sru.Cu.Tu.IDu}_inv(Kca)) =|>
    State' := 1/\ Nv' := new()
                /\ Rv' := new()
                /\ Qv' := F(Rv'.G)
                 /\ P' := new()
                /\ secret({Rv'}, subs2, V)
                /\ Snd(V.Nv'.Nu'.Qv'.Vrv.Srv.Cv.Tv.IDv.
                  { Vrv.Srv.Cv.Tv.IDv}_inv(Kca)
                  .{P'}_ H(IDu.IDv.Nu'.Nv'.Tu.Tv.Qu')
                  .H(H(IDu.IDv.Nu'.Nv'.Tu.Tv.Qu').P'.Nu'))
                /\ witness(V, U, alice_bob_nv, Nv')
  2. State  =1 /\ Rcv(U.V.Nv'.H(H(IDv.IDu.Nv'.Nu'.Tv.Tu.Qv')
                  .P.Nv')) =|>
      State' := 3 /\ request(U,V,bob_alice_nu,Nu')
end role
```

Fig. 2. Role specification for the responder (sensor node $v$) for our protocol.

```
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite\results\UAP_correct.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.01s
  visitedNodes: 4 nodes
  depth: 2 plies
```

Fig. 3. The result of the analysis using OFMC of our protocol.

computations [23]. In addition, the elliptic curve encryption and decryption are also computationally expensive as compared to those for symmetric key encryption and decryption (for example, AES encryption and decryption). For a rough estimation of the computational complexity for different schemes, we have measured the computational cost of the schemes in terms of $T_{mul}$ as in [41], [42]. The rough estimation

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 C:\progra~1\SPAN\testsuite\results\UAP_correct.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 4 states
 Reachable : 4 states
 Translation: 0.01 seconds
 Computation: 0.00 seconds
```

Fig. 4. The result of the analysis using CL-AtSe of our protocol.

Table 6. Time complexity of various operations in terms of $T_{mul}$.

| | | |
|---|---|---|
| $T_{ecm} \approx 1200T_{mul}$ | $T_{eca} \approx 5T_{mul}$ | $T_i \approx 3T_{mul}$ |
| $T_{add}$ is negligible | $T_h \approx 0.36T_{mul}$ | $T_{enc} \approx 0.15T_{mul}$ |
| $T_{dec} \approx 0.15T_{mul}$ | $T_{ecenc} \approx 2405T_{mul}$ | $T_{ecdec} \approx 1205T_{mul}$ |

of different operations in terms of $T_{mul}$ are shown in Table 6. From this table, it is noted that the computation of an ECC point multiplication requires approximately 1200 field multiplications; an ECC point addition requires one field inversion and two field multiplications; the computation of a field inversion requires approximately three field multiplications; the computation of elliptic curve encryption and decryption require approximately 2405 and 1205 field multiplications respectively [43], [44]; and the cost of field addition is negligible. A 1024-bit modular multiplication takes 41 times longer than a field multiplication in finite field $GF(2^{163})$; AES encryption/decryption and hashing operation using SHA-1 take 0.15 and 0.36 field multiplications, respectively.

The computational cost for each sensor node to achieve authentication and establishment of a secret key with its neighbor node for our scheme and other schemes are given in Table 7. It is noted that our scheme performs better than Zhou *et al.*'s scheme [6] and Huang's scheme [23], while our scheme is also comparable with that for other schemes. However, in Huang's scheme [8] each sensor node needs to update the broadcasted hash chain after each successful authentication. Further, in Kim-Lee's scheme [9] each node requires to update the broadcasted hash chain after each successful authentication and also to renew the hash chain.

Table 7. Comparison of computational overheads among different access control schemes.

| Scheme | Time complexity | Rough estimation |
|---|---|---|
| Zhou *et al.* [6] | $3T_{ecm} + T_i + T_h + 2T_{ecenc}/T_{ecdec}$ | $7213T_{mul}$ |
| Huang [8] | $2T_{ecm} + 4T_h$ | $2401T_{mul}$ |
| Kim-Lee [9] | $2T_{ecm} + 9T_h$ | $2409T_{mul}$ |
| Huang [23] | $5T_{ecm} + 9T_h$ | $6001T_{mul}$ |
| Ours | $4T_{ecm} + T_i + 4T_h + T_{enc}/T_{dec}$ | $4805T_{mul}$ |

Table 8. Comparison of communication costs among different access control schemes.

| Scheme | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|
| Zhou *et al.* [6] | 15232 | 21 | 20 |
| Huang [8] | 3904 | 10 | 10 |
| Kim-Lee [9] | 4136 | 16 | 12 |
| Huang [23] | 3392 | 10 | 8 |
| Ours | 4224 | 6 | 8 |

$I_1$ : Total number of bits transmission required for messages of all phases for schemes; $I_2$ : Total number of message transmissions for the scheme; $I_3$ : Total number of packets transmissions during authentication and key establishment phase, and dynamic nodes addition phase for the scheme.

We have computed the total number of bits required for transmission of messages for all phases, the total number of message transmissions of all phases, and total number of packet transmissions during authentication and key establishment phase, and dynamic nodes addition phase for our scheme and other schemes. Communication costs of our scheme and other schemes are shown in Table 8. In wireless sensor networks, the transmission energy consumption rate is greater than the energy consumption rates for computing [30]. From the table, it is clear that Zhou *et al.*'s scheme [6] requires a lot of communication overhead compared with Huang's scheme [8], Kim-Lee's scheme [9], Huang's scheme [23] and our scheme. Further, our scheme outperforms in term of communication overhead compared to Huang's scheme [8], Kim-Lee's scheme [9] and Huang's scheme [23], because our scheme requires a few number of message transmissions only. Moreover, Huang's scheme [8] and Kim-Lee's scheme require the involvement of the base station during the authentication and key establishment phase, because the hash chain needs to be broadcasted by the base station to all the existing sensor nodes in the network, whereas our scheme, Zhou *et al.*'s scheme [6] and Huang's scheme [23] do not require to involve the base station during that phase.

We have then computed the number of bits required to store information prior to a node's deployment for authentication and key establishment purpose for our scheme and other schemes. The results are given in Table 9. It is evident from the table that our scheme requires minimum number of bits prior to a node's deployment in each sensor node for authentication and key establishment with its neighbor nodes as compared to that for other schemes except Huang's scheme [8].

Table 9. Comparison of storage overheads among different access control schemes.

| Scheme | Storage complexity required to store information prior to a node's deployment (in bits) |
|---|---|
| Zhou *et al.* [6] | 1824 |
| Huang [8] | 1456 |
| Kim-Lee [9] | 1616 |
| Huang [23] | 1648 |
| Ours | 1560 |

Finally, we have considered the energy required for a sensor node during the authentication and key establishment phase in order to authenticate its neighbor node and then to establish a secret session key with that neighbor node for our scheme and other schemes. As in [45], the energy cost of a sensor node is considered due to both computational and communication costs involved during the authentication and key establishment phase. Comparison of the energy cost of a sensor node during the authentication and key establishment phase between our scheme and other schemes is shown in Table 10. In wireless communication, energy of sensor nodes goes mainly for transmissions and receptions of messages/packets rather than computing. Since our scheme requires less number of message or packet transmissions as compared to other schemes, the energy spent by sensor nodes is less as compared to other schemes. Due to involvement of the base station during the authentication and key establishment phase, in Huang's scheme [8] and Kim-Lee's scheme [9] a sensor node needs to further spend energy to receive the broadcasted information by the base station. Moreover, Huang's scheme [8] and Kim-Lee's scheme [9] are insecure, whereas our scheme is secure against different attacks which are shown theoretically and through simulation results using AVISPA tool. Hence, considering efficiency and security, our scheme is significantly better than the existing access control schemes.

## 7. Conclusion

We have proposed an effective access control scheme, which is suitable for wireless sensor networks. Using the preloaded certificate, each sensor node can authenticate and establish symmetric secret keys with its neighbor nodes. We have shown that our scheme is unconditionally secure against node capture attacks. In our scheme,

Table 10. Comparison of energy cost of a sensor node during the authentication and key establishment phase between our scheme and other schemes.

| Scheme | Sensor node's energy cost |
|---|---|
| [6] | bootstrapping time validation + one hash operation for parameter generation + ECDSA signature verification to verify node's identity, length of bootstrapping phase and node's public key + two ECC encryption/decryption for random nonce encryption and validation + five message transmissions |
| [8] | one ECC point multiplication + one random number generation + two hash operations for parameters generation + one ECC point multiplication for computing shared session key + two hash operations for hash chain validation + four message transmissions |
| [9] | one ECC point multiplication + one random number generation + four hash operations for parameters generation + one ECC point multiplication for computing shared session key + four hash operations for hash chain validation + one hash operation for hash chain renewal + one broadcasting for hash chain renewal + four message transmissions |
| [23] | one ECC point multiplication + one random number generation + two hash operations for signature generation and verification + timestamp validation + two hash operations for parameters verification+ one ECC point multiplication for computing secret shared key + three ECC point multiplication for signature verification + four message transmissions |
| Ours | two ECC point multiplication for public key computation and secret key generation + ECDSA signature verification + value of latest version verified validation, one hash operation for secret key generation, and two hash operations for puzzle message generation and validation + one symmetric-key encryption/decryption for puzzle message encryption/ validation + three message transmissions |

the external devices are prevented from injecting false reports into the sensor network. Through the formal security analysis we have further shown that our scheme can defend different node deployment attacks by an adversary. In our scheme, we require only a few message transmissions as compared to that for Zhou *et al.*'s scheme, Huang's scheme and Kim-Lee's scheme, and as a result, our scheme is more appropriate for practical applications. Moreover, our scheme is efficient in computation for authentication and key establishment of symmetric secret keys with its neighbor nodes. Dynamic nodes addition in our scheme does not require any involvement of the base station during the authentication and key establishment phase as in Huang's scheme and Kim-Lee's scheme. Zhou *et al.*'s scheme requires each sensor node to sustain a tolerance time interval before it is compromised. However, our scheme, Huang's scheme and Kim-Lee's scheme do not have this constraint. In addition, our scheme can be easily implemented as a dynamic access control because

all the old node parameters stored in each existing deployed sensor node need not be changed/updated once a new node is deployed or an old node is lost. We have also simulated our protocol for formal security analysis using AVISPA tool and it is evident from the simulation results that our protocol is secure.

## Acknowledgments

We gratefully acknowledge the insightful comments of the anonymous reviewers, the Editor, Prof. Albert Y. Zomaya, and the Managing Editor, Prof. Sartaj Sahni, which have improved the quality and presentation of the paper.

## References

[1] M. Ilyas and I. Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC, 2005.

[2] P. Baronti, P. Pillai, V.W.C. Chook, S. Chessa, A. Gotta, and Y. F. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, 30(7):1655–1695, 2007.

[3] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[4] B. Parno, A. Perrig, and V. Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *Proceedings of the* 2005 *IEEE Symposium on Security and Privacy* (*S&P'*05), 2005.

[5] Y. Hu, A. Perrig, and D. B. Johnson. A defense against wormhole attacks in wireless networks. In *Proceedings of Annual IEEE International Conference on Computer and Communications* (*IEEE INFOCOM'*03), 2003.

[6] Y. Zhou, Y. Zhang, and Y. Fang. Access control in wireless sensor networks. *Ad Hoc Networks*, 5:3–13, 2007.

[7] H. Wen, J. Luo, and L. Zhou. Lightweight and effective detection scheme for node clone attack in wireless sensor networks. *IET Wireless Sensor Systems*, 1(3):137–143, 2011.

[8] H.-F. Huang. A novel access control protocol for secure sensor networks. *Computer Standards & Interfaces*, 31:272–276, 2009.

[9] H.-S. Kim and S.-W. Lee. Enhanced novel access control protocol over wireless sensor networks. *IEEE Transactions on Consumer Electronics*, 55(2):492–498, 2009.

[10] L. Eschenauer and V. D. Gligor. A Key Management Scheme for Distributed Sensor Networks. In *9th ACM Conference on Computer and Communication Security*, pages 41–47, November 2002.

[11] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, Berkeley, California, 2003.

[12] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security* (*CCS*), pages 52–61, Washington DC, Oct 27-31 2003.

[13] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In *ACM Conference on Computer and Communications Security* (*CCS'*03), pages 42–51, Washington DC, USA, October 27-31 2003.

[14] A. K. Das. An Efficient Random Key Distribution Scheme for Large-Scale Distributed Sensor Networks. *Security and Communication Networks*, 4(2):162–180, 2011.

[15] M. Eltoweissy, M. Moharram, and R. Mukkamala. Dynamic key management in sensor networks. *IEEE Communications Magazine*, 44(4):122–130, April 2006.

[16] O.D. Mohatar, A.F. Sabater, and J.M. Sierra. A light-weight authentication scheme for wireless sensor networks. *Ad Hoc Networks*, 9(5):727–735, 2011.

[17] A. K. Das. A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. *International Journal of Information Security*, 11(3):189–211, 2012.

[18] A. K. Das. A Survey on Analytic Studies of Key Distribution Mechanisms in Wireless Sensor Networks. *Journal of Information Assurance and Security*, 5(5):526–553, 2010.

[19] Y. Wang, G. Attebuty, and B. Ramamurthy. A Survey of Security Issues in Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 8(2):2–23, 2006.

[20] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway. A survey of key management schemes in wireless sensor networks. *Computer Communications*, 30 (11-12):2314–2341, 2007.

[21] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P.-C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigano, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *17th International Conference on Computer Aided Verification (CAV'05), LNCS* 3576, pages 281–285, 2005.

[22] J. Shen, S. Moh, and I. Chung. Comment: "Enhanced novel access control protocol over wireless sensor networks". *IEEE Transactions on Consumer Electronics*, 56(3):2019–2021, 2010.

[23] H.-F. Huang. A New Design of Access Control in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2011. Article ID 412146, 7 pages doi:10.1155/2011/412146.

[24] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology (Crypto 89), Lecture Notes in Computer Science, Springer-Verlag*, volume 435, pages 339–351, 1990.

[25] M. L. Das. Two-Factor User Authentication in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 8(3):1086–1090, 2009.

[26] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[27] N. Koblitz. Elliptic Curves Cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[28] Secure Hash Standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995.

[29] S. Vanstone. Responses to NIST's proposal. *Communications of the ACM*, 35:50–52, 1992.

[30] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and Approaches for Distributed Sensor Network Security. dated September 1, 2000. NAI Labs Technical Report No. 00-010.

[31] CC2420: 2.4 GHz IEEE 802.15.4/ZigBee-Ready RF Transceiver. Available from: http://www.ti.com/product/cc2420. Accessed on September 2011.

[32] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[33] D. Johnson and A. Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical Report CORR 99-34, Dept. of C & O, University of Waterloo, Canada, August 23, 1999.

[34] H.Z. Liao and Y.Y. Shen. On the Elliptic Curve Digital Signature Algorithm. *Tunghai Science*, 8:109–126, 2006.

[35] 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*, 24(2):247–260, 2006.

[36] R. Dutta and R. Barua. Provably Secure Constant Round Contributory Group Key Agreement. *IEEE Transactions on Information Theory*, 54(5):2007–2025, May 2008.

[37] Y.-H Chuang and Y.-M. Tseng. An efficient dynamic group key agreement protocol for imbalanced wireless networks. *International Journal of Network Management*, 20(4):167–180, 2010.

[38] E. Clarke, J. Somesh, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of the IFIP Working Conference on Programming Concepts and Methods*, 1998.

[39] D. Basin, S. Modersheim, and L. Vigano. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[40] D. von Oheimb. The high-level protocol specification language hlpsl developed in the eu project avispa. In *Proceedings of APPSEM* 2005 *Workshop*, 2005.

[41] A. K. Das. A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. *Networking Science*, 2012. in press.

[42] S. Wu and K. Chen. An Efficient Key-Management Scheme for Hierarchical Access Control in E-Medicine System. *Journal of Medical Systems*, 36(4):2325–2337, 2012.

[43] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. In *Proc. of Advances in Cryptology - CRYPTO '95, Lecture Notes in Computer Science, Springer-Verlag*, volume 963, pages 43 – 56, 1995.

[44] E. DeWin, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle. A fast software implementation for arithmetic operations in $GF(2^n)$. In *Proc. of Advances in Cryptology - ASIACRYPT '96, Lecture Notes in Computer Science, Springer-Verlag*, volume 1163, pages 65–76, 1996.

[45] A. K. Das, P. Sharma, S. Chatterjee, and J. K. Sing. A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Journal of Network and Computer Applications*, 35(5):1646–1656, 2012.