SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:1752–1771
Published online 14 October 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1140

RESEARCH ARTICLE

An effective ECC-based user access control scheme with attribute-based encryption for wireless sensor networks

Santanu Chatterjee[1] and Ashok Kumar Das[2]*

[1] Research Center Imarat, Defence Research and Development Organization, Hyderabad 500 069, India
[2] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India

ABSTRACT

For critical applications, real-time data access is essential from the nodes inside a wireless sensor network (WSN). Only the authorized users with unique access privilege should access the specific, but not all, sensing information gathered by the cluster heads in a hierarchical WSNs. Access rights for the correct information and resources for different services from the cluster heads to the genuine users can be provided with the help of efficient user access control mechanisms. In this paper, we propose a new user access control scheme with attribute-based encryption using elliptic curve cryptography in hierarchical WSNs. In attribute-based encryption, the ciphertexts are labeled with sets of attributes and secret keys of the users that are associated with their own access structures. The authorized users with the relevant set of attributes can able to decrypt the encrypted message coming from the cluster heads. Our scheme provides high security. Moreover, our scheme is efficient as compared with those for other existing user access control schemes. Through both the formal and informal security analysis, we show that our scheme has the ability to tolerate different known attacks required for a user access control designed for WSNs. Furthermore, we simulate our scheme for the formal security verification using the widely-accepted automated validation of Internet security protocols and applications tool. The simulation results demonstrate that our scheme is secure. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS

wireless sensor networks; user access control; authentication; security; ECC; AVISPA

***Correspondence**

Ashok Kumar Das, Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India.
E-mail: iitkgp.akdas@gmail.com; ashok.das@iiit.ac.in

1. INTRODUCTION

In present day scenario, the user access control for a hierarchical wireless sensor network (HWSN) becomes an important issue. Access rights for the correct information and resources for different services to the users can be provided with the help of efficient user access control. User access control can identify and impose different access privileges for different types of users with proper user authentication. For accessing the real-time sensing data from the nodes in wireless sensor network (WSN), a user needs to be first authenticated by the nodes and the base station (BS), and hence, the unauthorized access to the nodes can be prevented in the network [1].

Most applications of WSNs are critical in nature, and they are based directly on the real-time data access from the nodes. The users may be interested for accessing the real-time data only from the nodes. We can then allow the users (i.e., the external authorized parties) to access the real-time sensing information directly from the nodes inside WSN as and when they demand, where the involvement of the BS is not needed at all for secure communication between the user and the nodes for this purpose. In a mission-critical application scenario, different types of information belonging to various security levels can be generated by all kinds of sensors. With the proper access privilege, selected types of the authorized users should access proper data. That is, accessibility of a particular type of data to users is based solely on necessity. To allow authorized access of the real-time data from the nodes inside the hierarchical WSNs to the legal users on demand, we need to deploy the user access control before allowing them to access the real-time information from the hierarchical WSNs for which they are permitted. For

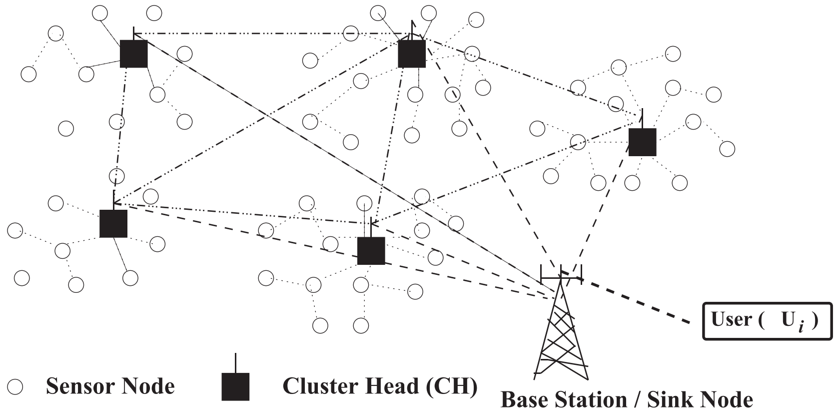

Figure 1. A hierarchical wireless sensor network architecture (Source: [1]).

example, in healthcare applications, monitoring patient's conditions by the expert doctors is very essential. Thus, real-time data sensed by the sensors in a patient's body can be monitored directly by an authorized external user as and when demand is made. Based on critical and emergency situation of the patient, the doctor can take necessary action by instructing the nurses/medical staffs in the hospital for the patient. Hence, before allowing access to the sensitive and private real-time data of the patients, the external user (doctor) must be authenticated for a particular access privilege by the BS (i.e., the medical server) as well as sensor node in the network. In a battlefield surveillance, the user access control is very much essential to give proper access privileges for different types of users. In such application, several sensor nodes can be deployed in the battlefield (target field) via low-flying airplanes or trucks. After that, the deployed sensor nodes monitor the conditions and activities from their surrounding areas and then report the sensing observations to the BS through their neighboring nodes using wireless communication. After collecting the information from the sensors, the BS can conduct a more accurate detection on the activities (for example, possible attacks) of the opposing force. The appropriate decisions and responses are made quickly in the battlefield. It is thus clear that the immediate transmission of critical real-time data from the nodes are required. Moreover, in the battle field scenario, a commander should be able to access all types of data for the purpose of overall coordinating, but a solider may only need to access the type of data relevant to his/her mission. Considering these points, the importance of user access control in HWSN for any applications becomes an important research field.

In this paper, we target to present a new user access control scheme, which is based on the passwords of users and relevant given access privilege to the users to provide different access privileges for different types of users to real-time information by authorizing him/her directly at the nodes and also making it possible for the external users to communicate securely with the nodes to get the responses to their queries.

1.1. Network model

Figure 1 shows the architecture of a HWSN [1]. In this architecture, a hierarchy is presented among the various nodes for their capabilities: *BS*, *cluster heads or group heads*, and *sensor nodes*. In this hierarchy, the sensor nodes are considered as inexpensive and generic wireless devices, and their capabilities are limited. These sensors have limited battery power, memory size and data processing capability, and also have limited radio transmission range. These nodes can be deployed in a cluster, which can communicate among each other in that cluster, and then with their nearby cluster heads. On the other hand, the cluster heads are resource rich than sensor nodes, which have high-power batteries, larger memory space, powerful antenna, and data processing capabilities. They can communicate among each other and route the sensing information of the sensors in their clusters to their neighbor cluster heads and then to the BS. Finally, the BS (BS), also known as the gateway node, is considered as a gateway entity to another network, which is considered as the most powerful data processing/storage center, or also an access point for human interface.

The HWSN model is used for developing our proposed scheme [2]. WSNs are generally distributed event-driven systems. They are different from the traditional wireless networks. They can have extremely large network size, limited energy, redundant low-rate data, and also many-to-one flows. In many WSN applications, we do not always require the connectivity between all sensor nodes. Data centric mechanisms are performed in order to aggregate redundant data to reduce the energy consumption and traffic load. Hence, HWSN has more operational advantages than the distributed WSN model because of the inherent limitations of sensor nodes on their battery power as well as computational ability.

1.2. Motivation

The motivation behind the designing our scheme is as follows. In HWSN, the users (also called the external entities) are allowed to access data as and when they demand, if they are authorized. To provide authorized access of the real-time information from the nodes to the legal users on demand, there is a great need to have the user access control mechanism. In healthcare applications or battlefield surveillance monitoring by the experts is very essential. Thus, real-time data sensed by the sensors can be monitored directly by an authorized external user as and when demand is made. Hence, before allowing access to the sensitive and private real-time data, the external user must be authenticated for a particular access privileges by the BS as well as sensor node in the network.

The elliptic curve cryptography (ECC) is used for our user access control scheme in HWSN. RSA [3] may also be used to authenticate external users and Diffie-Hellman [4] over discrete logarithm problem (DLP) to establish shared keys between external users and sensor nodes in the network. But the evaluation of a 1024-bit modular exponentiation for the DLP of the form 2^x, where x is at least 160 bits, requires more than 50 s [5,6] on both MICA1 and MICA2 motes [7]. Gura *et al.* [8] experimented ECC and RSA on the Atmel ATmega 128 processor [7] using the assembly language, and they pointed out that an 160-bit ECC-point multiplication requires 0.81 s and 1024-bit RSA public-key and private-key operations require 0.43 s and 10.99 s, respectively. The same level of security with smaller size key is then achieved using ECC instead of RSA [9,10]. For examples, 160-bit ECC provides the comparable security to 1024-bit RSA and 224-bit ECC gives the comparable security of 2048-bit RSA [11]. In WSNs, the transmission energy cost is approximately over three orders of magnitude greater than the energy cost for local computation [12]. As a result, the packet size and the number of packets in transmission are the crucial factors in the performance while designing a user access control scheme in WSNs. These motivate us to apply ECC in place of RSA in our user access control, and we can then certainly gain much more energy and bandwidth savings. Furthermore, we also use the efficient symmetric-key cryptographic technique (for example, advanced encryption standard (AES) cryptosystem) along with ECC for achieving the communication and computational efficiency in our scheme.

1.3. Attribute-based encryption

One of the fundamental demand of user access control is that a particular user must have some unique access privilege to some data set as per its access policy. To approach this problem, Sahai and Waters [13] introduced a cryptographic primitive concept, known as the attribute-based encryption (ABE), which is built up on the rudimentary concept of the identity-based encryption proposed by Shamir. According to ABE, a user $\mathcal{U}_i$ creates a secret key and a ciphertext with a set of descriptive attributes that it contains. This ciphertext can be decrypted by some other user $\mathcal{U}_j$, whose key contains matching or overlapping set of attributes above certain number with the ciphertext attributes of $\mathcal{U}_i$.

Based on the concept of ABE, Goyal *et al.* proposed a scheme, called the key-policy attribute-based encryption (KP-ABE) [14]. Each ciphertext is encrypted using a set of descriptive attributes. Each user forms a secret key using a tree-access structure. The leaf-nodes of the user-access structure are associated with the attribute and the non-leaf nodes consist of OR and AND logic gates, which provide the access policy. If the access structure accepts the ciphertext attribute, the user can decrypt the ciphertext. Both ABE and KP-ABE schemes are the foundation for the scheme by Yu *et al.* [15] and the scheme by Ruj *et al.* [16].

We describe in brief the fundamentals of KP-ABE scheme, which is based on the bilinear pairings. Assume that G_1, G_2, and G_3 are the multiplicative cyclic groups of prime order p. Further, assume that g_1 and g_2 are the generators of G_1 and G_2, respectively. A bilinear function $e : G_1 \times G_2 \rightarrow G_3$ is an injective (one-one) function, which has the following properties:

- *Bilinearity:* For all $u \in G_1$, $v \in G_2$, $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$, where $Z_p = \{0, 1, \ldots, p-1\}$.
- *Non-degeneracy:* $e(g_1, g_2) \neq 1$, where g_1 and g_2 are the generators of G_1 and G_2, respectively.
- *Computability:* There exists an efficient algorithm for computing the value $e(u, v)$ for each $u \in G_1$ and $v \in G_2$.

The KP-ABE scheme has the following four major steps to execute:

- *Setup:* The system selects the following parameters: a bilinear group G_1 of prime order p, its generator g, a bilinear map e, a universe of attributes $\mathcal{I}$, a set of prime random numbers t_i for each attribute i, and a unique prime random number. From these parameters, this algorithm outputs the public parameter PK and the master secret key MK.
- *Key generation:* Inputs of this algorithm are a user access tree $\mathcal{P}$ and the master secret key MK. Depending on the leaf node attribute set of the user access tree, it generates a secret key or decryption key SK.
- *Encryption:* Inputs of this algorithm are the message m, a set of attributes $\mathcal{I}_i$ and the public key PK. It encrypts the message m using the public key PK and outputs the ciphertext E.
- *Decryption:* In this step, a user receives the ciphertext E encrypted under the attribute set $\mathcal{I}_i$, the secret key SK (generated from the user access tree $\mathcal{P}$) and the public key PK as input. If the attribute set $\mathcal{I}_i$ matches with the user access structure $\mathcal{P}$, it decrypts the ciphertext E using its secret key SK, and outputs back to the original message m.

Yu *et al.* proposed a scheme to implement the idea of KP-ABE into the field of WSN [15]. The basic objective

Security Comm. Networks 2015; **8**:1752–1771 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

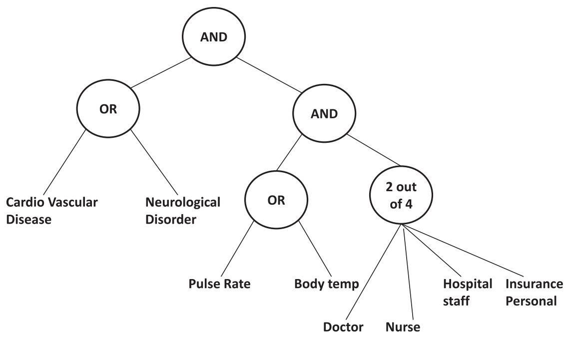

Figure 2. A user access structure, where the user is able to decrypt the sensor node data.

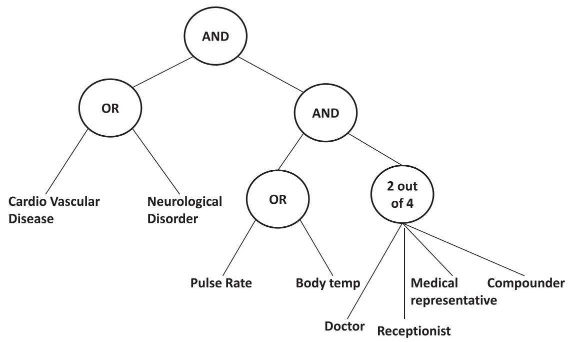

Figure 3. A user access structure, where the user is not able to decrypt the sensor node data.

of the scheme by Yu *et al.* is to provide a fine-grained distributed data access control for WSNs. The proposed scheme exploits the fundamental cryptographic concepts of KP-ABE technique [14]. The scheme is resistant to collusion attack, node compromise attack, and it provides a way of user revocation policy through backward secrecy. Ruj *et al.* [16] extended that scheme to operate upon an environment of multiple distribution centers, which has the same responsibility as a BS where the partial modification of the user access structure can be carried out efficiently.

Following the example provided by Yu *et al.*, we also give an example of a fine grained access control in body sensors. The body sensors are specified with multiple attributes. Consider a situation where the body sensors can detect many in-body diseases such as cardiovascular problem and neurological disorder. Suppose the sensors can also measure some on-body parameters such as body temperature and pulse rate (on-body attribute). Sensors have multiple owners such as doctors, nurses, and hospital staffs. The body sensor can be specified with these attributes as in-body = {cardiovascular disease, neurological disorder, cancer}, on-body = {pulse rate, body temperature}, and owner = {doctor, nurse, hospital staff}. The BS provides each user an access policy via a user access tree. A user then decrypts data from a sensor/cluster head only if he/she has matching attribute with that sensor. For example, a user $\mathcal{U}_i$ with the access structure, provided in Figure 2, can decrypt the sensor data from a sensor node that detects in-body disease like cardiovascular disease or neurological disorder and contains on-body measuring attributes as pulse rate or body temperature, and at least owned by 2-out-of-4 experts like doctor, nurse, hospital staff, or medical insurance person. The user $\mathcal{U}_j$ with user access tree given in Figure 3 is not able to decrypt the sensor node data. The sensor cannot satisfy the '2-out-of-4' threshold structure of the user, because the 'owner' attribute has only one matching data (doctor), and thereby returns a false to the AND logic gate.

1.4. Threat model

The Dolev-Yao threat model [17] is used in our scheme, in which any two nodes can communicate over a public channel [18]. The similar threat model for our scheme is applied, where the channel is considered as insecure, and the end-points (cluster heads or sensor nodes) are not in general trustworthy. An attacker can thus eavesdrop on all traffic, inject packets, and reply old messages previously delivered. The BS is considered as trustworthy and it will never be compromised by an attacker (adversary). We assume that the sensors and cluster heads are not tamper-resistant devices due to cost constraints. If an attacker compromises any sensor or cluster head, he/she can exact all cryptographic information including the secret key materials, data as well as code stored in its memory.

1.5. Our contributions

We aim to propose a new password-based user access control scheme with the help of the attribute-based encryption in HWSNs. The following are the attractive properties achieved in our scheme:

- It provides security and access privilege-wise user authentication depending on the access rights provided for the genuine users in HWSN.
- It preserves high security as compared with other related existing user access control schemes, because our scheme provides mutual authentication between the user, the BS and cluster head, and it also resists known attacks including denial-of-service attack, privileged-insider attack, stolen smart-card attack, replay attack, many logged-in users with the same login-id attack, and man-in-the-middle attack.
- Our scheme supports attribute-based encryption, where the user should have proper key with matching set of attributes to retrieve the information from the network so that the user with lower access privilege cannot access data given for the higher access privileged users.

- It efficiently provides new node addition dynamically after the initial deployment of nodes in WSN. For this purpose, we do not require to update any more information in the user's smart card.
- Our scheme supports the user's password change locally without further contacting the BS.
- In our scheme, both the user and the cluster head establish a session key between them after their successful mutual authentication so that they can use that session key for secure communication.

1.6. Organization of the paper

The roadmap of this paper is sketched as follows. In Section 2, the existing related works are reviewed on user access control in WSN and works on security in wireless body area networks are discussed. In Section 3, we propose a novel ECC-based user access control scheme in HWSNs. Section 4 analyzes the security properties of our proposed scheme and also the performance of our scheme. In Section 5, we simulate our scheme for the formal security verification using the widely-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool and show that our scheme is also secure against passive and active attacks including the replay and man-in-the-middle attacks. In Section 6, the performance of our scheme is compared with other related schemes. Finally, Section 7 concludes the paper.

2. RELATED WORK

In this section, we discuss in brief the existing related user access control schemes and security protocols applicable in resource-constrained WSNs.

Wang *et al.* [19] splitted the access control process into local authentication conducted by a group of sensors physically close to a user, and a lightweight user access control scheme based on the endorsement of the local sensors. They implemented the access control protocols on a testbed of TelosB motes. Based on ECC, they provided the local authentication. In their scheme, using certificate-based authentication, the user access is verified by the sensor node. He *et al.* [20] proposed a distributed privacy-preserving access control scheme for WSNs, which has the characteristics of a single-owner multi-user sensor network and also the requirements of distributed privacy-preserving access control. Wen *et al.* [21] proposed a user access control scheme for wireless multimedia sensor network. In this scheme, the authorized user can access the real-time multimedia data. This scheme is based on the Chinese Remainder Theorem. Li *et al.* [22] discussed various practical issues required for fulfilling the security and privacy requirements in wireless body area networks. They also explored the relevant security solutions in sensor networks and wireless body area sensor network, and also analyzed those applications. They proposed an attribute-based encryption for achieving fine-grained access control.

Mahmud and Morogan [23] proposed a user authentication and access control scheme, which is based on the Identity-Based Signature. They applied the ECC-based digital signature algorithm for the purpose of signing and verifying a message. At the time of initialization, the sensor nodes as well as users are also registered to the BS, and the group identity and access rights of the users are also given by the BS. The user revocation is carried out by expiration of access time of the user assigned by the BS at the time of registration. Without having the proper access right, the authenticated user is not allowed to obtain the requested access. Although this scheme protects the node capture attack and denial-of-service attack, but the user registration as well as password change process are not supported. For a new user addition, the BS needs to broadcast again the user's parameters like userid, groupid, and system timestamp, which incur more communication overheads for the network.

Wang *et al.* [24] proposed an ECC-based user access control scheme. In this proposed scheme, before authentication, a user needs to apply to the key distribution center (KDC) for the access permission. The KDC maintains a user access list pool with the respective user's access privilege. This access privilege consists of the userid, groupid, and user access privilege mask. The multiple users within a same group should have the same access privilege. Based on the ECC, in this scheme, the KDC generates a public key, a private key of the user, and a certificate of user access list based on the user's request. The user then requests the sensor node by sending its certificate and then selects a random number as a session key. Then, the sensor node computes the user's public key as well as generates a temporary public key and then encrypts that session key. The senor node passes the temporary public key and a point on ECC computed by the hash of the session key with the public key of the user along with the message authentication code (MAC) of a random nonce. The user verifies that message decrypts the session key, retrieves the nonce, and then sends the access privilege with that nonce to the sensor node. The sensor node verifies that nonce and replies the information requested by the user, which is again encrypted by the session key. In this scheme, although the user authenticates a sensor node, but that sensor node does not authenticate the user. Thus, the mutual authentication is not provided between the user and the senor node in their scheme. Le *et al.* [25] proposed an access control scheme based on ECC, which is energy efficient. In this scheme, they improved the scheme by Wang *et al.* [26]. This scheme is a public-key cryptography-based access control scheme, where the user needs to take the access permissions from a KDC. An access control list (ACL) pool and the associated user identifications are maintained by the KDC. The user's access privileges are defined in ACL based on user access privileges mask. The public keys between KDC and sensor nodes are mutually exchanged during the pre-deployment phase. The user receives his/her public key and private key after the registration process. A signed certificate of access control list is also issued by the KDC and is sent to the user. The user then has to be authenticated by the sensor node for future secure communications. In this scheme, the user is authenticated by the sensor node, but there is no provision for mutual authentication of the sensor node by the user.

3. THE PROPOSED SCHEME

In this section, we describe our new user access control scheme. The proposed scheme has the seven phases: (i) pre-deployment phase; (ii) post-deployment phase; (iii) registration phase; (iv) login phase; (v) authentication phase; (vi) password change phase; and (vii) dynamic node addition phase, which are discussed in the following subsections.

As in [1], we consider a HWSN model [27,28] (shown in Figure 1) for our scheme. This model consists of a small number of sensors, which are powerful High-end sensors (called the H-sensors) and a large number of resource-starved sensors, called the Low-end sensors (called the L-sensors). For example, we can consider the H-sensors as personal digital assistant (PDA) and the L-sensors as the MICA2-DOT motes [29]. Now-a-days MICA2 motes become obsolete devices, and thus, the L-sensors can be MICAz/IRIS sensors [29]. The target field is a 2D field and divided into a number m of equal-sized disjoint clusters or groups. Each cluster has a cluster head CH_i (H-sensor), which is deployed around the center of that cluster or group, and a number n_i of L-sensors, which are deployed randomly in that cluster. We consider the number n_i of L-sensors nodes in each cluster such that the secure network connectivity in each cluster becomes very high. Depending on the applications, the BS can be located either in the center or at a corner of the deployment field.

We make use of the notations listed in Table I in order to describe and analyze our proposed scheme.

Table I. Notations used in this paper.

Symbol	Description
p	A large prime
$E_p(a,b)$	An elliptic curve over a finite field $GF(p)$
G	A base point in $E_p(a,b)$
U_j	j^{th} user
ID_{U_j}	Identity of U_j
PW_j	Password of user U_i
TS_{U_j}	Registration timestamp of U_j
P_j	An access structure for U_j
n_j	A random nonce generated by U_j
BS	Base station
SN_j	j^{th} sensor node in a cluster
ID_{SN_j}	Identity of sensor SN_j
MK_{SN_j}	Master key of sensor SN_j
CH_i	i^{th} cluster head in HWSN
ID_{CH_i}	Identity of cluster head CH_i
MK_{CH_i}	Master key of cluster head CH_i
$h(\cdot)$	Secure collision-free one-way hash function
$E_k(M)/D_k(M)$	Symmetric encryption/decryption of M using the key k
$X \oplus Y$	Bitwise XORed of data X with data Y
$X\|Y$	Data X concatenates with data Y

3.1. Pre-deployment phase

This phase is used to preload the keying materials to all regular sensor nodes and cluster heads prior to their deployment in a target field. This phase is executed by the BS in offline prior to deployment of sensor nodes and cluster heads in the target field. It consists of the following steps:

Step 1: Before deployment of the nodes, the BS chooses the following *network parameters*. The BS selects a finite field $GF(p)$, where p is a large odd prime of at least 160 bits, an elliptic curve $E_p(a,b)$, which is the set of all points of $y^2 = x^3 + ax + b \pmod p$ such that $a, b \in Z_p = \{0, 1, 2, \ldots, p-1\}$ being the constants with the condition that $4a^3 + 27b^2 \neq 0 \pmod p$, and a base point G in $E_p(a,b)$ whose order is n, where n is at least 160 bits such that $n > 4\sqrt{p}$. The BS chooses a random number y for all registered users, where $y \in Z_n^* = \{1, 2, \ldots, n-1\}$ and then computes $Y = yG$. Depending on the probable user query, the BS chooses a random number $t_i \in Z_n^*$ for each attribute $i \in I$; where I is the universe of all the sensor attributes in a WSN application, and then computes $T_i = t_iG$ for each attribute $i \in I$. For each deployed cluster head CH_i, the BS selects a unique identifier, say ID_{CH_i} and a set I_i of all attributes belonging to a particular cluster heads CH_i. The BS also generates a unique random master key, say MK_{CH_i} for each deployed cluster heads CH_i, which is shared with the BS only. Then, the BS generates a secret key $B_i = b_iG$ for each cluster head CH_i, where b_i is only known to the BS. Further, for each deployed sensor node SN_j, the BS assigns a unique identifier, say ID_{SN_j} and the I_j set of all attributes belonging to a particular sensor node SN_j. The BS also selects a unique random master key, say MK_{SN_j} for each deployed sensor node SN_j.

Step 2: Finally, the BS pre-loads the following information into the memory of each cluster head CH_i before its deployment in the target field: (i) ID_{CH_i}; (ii) $E_p(a,b)$; (iii) the base point G; (iv) the secret key K_i for CH_i; (v) a secure collision-resistant hash function $h(\cdot)$; (vi) MK_{CH_i}; (vii) the secret key B_i; (viii) the public attribute key T_i belongs to that particular CH_i, where $i \in I_i$; and (ix) Y, which is used for the master key encryption. The BS also pre-loads ID_{SN_j}, I_j, and MK_{SN_j} to each deployed sensor node SN_j in a particular cluster having the cluster head CH_i.

3.2. Post-deployment phase

This phase remains same as in [1]. After deployment, each node first finds its physical neighbors within its communication range. The deployed sensor nodes establish the pairwise keys with their neighbor sensor nodes and the cluster head in their corresponding cluster. For this purpose, the unconditionally secure key management scheme [28] can be used. After key establishment, each node can securely communicate with its neighbor nodes as well as cluster head in their corresponding cluster. The cluster heads securely communicate with their neighbor cluster heads, and finally, to the BS in the network.

3.3. Registration phase

In the registration phase, a legal user U_j needs to register with the BS of the network in order to access the real-time data from a specified cluster head CH_i. This phase has the following steps:

Step 1: User U_j selects his/her identity ID_{U_j}, password PW_j, and a randomly generated number x_j.

Step 2: After that, U_j computes the masked password $RPW_j = h\left(ID_{U_j}||x_j||PW_j\right)$ and sends the registration request message $\left\langle ID_{U_j}||RPW_j\right\rangle$ to the BS via a secure channel. Note that '||' denotes the concatenation operator.

Step 3: The BS computes A_j for each user U_j, where $A_j = h\left(ID_{U_j}||TS_{U_j}\right)$, where TS_{U_j} is the registration timestamp of the user U_j. The BS then calculates the secret shared masked password $R_{U_j} = h(RPW_j||A_j)$ for that user U_j.

Step 4: Suppose m cluster heads, say $CH_1, CH_2, \ldots, CH_m$, will be deployed during the initial deployment. The BS then computes the m key-plus-id combinations $\left\{\left(s_{ij}, ID_{CH_i}\right) \mid 1 \le i \le m\right\}$, where $s_{ij} = h\left(TS_{U_j}||\left(T_{CH_i} \oplus W_{CH_i}\right)\right)$, where T_{CH_i} is the bootstrapping of the cluster head CH_i, and W_{CH_i} the expiration time of the cluster head CH_i. The BS further generates a random secret number $y_j \in Z_p$ and computes the public key $Y_j = y_jG$ for each registered user U_j.

This phase has further divided two sub-phases, which are given in the succeeding text:

3.3.1. Access structure generation.

Step 1: The BS selects an access structure P_j for each user U_j. After receiving the registration information from the valid users, the BS assigns each user U_j the access structure P_j. These access structures are implemented via an access tree. Every leaf node of the access tree is labeled with an attribute. The internal nodes are considered as the threshold gates. We the represent the access structures using the logic expressions over the attributes. With the help of the access tree data access, the privilege of each user U_j can be defined. For example, Figure 2 illustrates a particular access structure used for wireless body area sensor network. For each node x in P_j, the BS needs to construct a $d_x + 1$ degree polynomial q_x applying the Lagrange interpolation, where d_x is the degree of the node x.

Step 2: For each non-root node x in P_j, it sets $q_x(0) = q_{parent(x)}(index(x))$, where $parent(x)$ is the parent of x and x the $index(x)$-th child of its parent. The BS then assigns $q_r(0) = y$, where $q_r(0)$ is the polynomial of the root of the access tree of the user U_j.

Step 3: Finally, the BS computes $k_{i_{U_j}} = (q_i(0) - t_i) \pmod p$ for each leaf node $i \in P_j$.

3.3.2. Smart card generation.

The BS issues a smart card for the user U_j containing the following parameters: (i) G; (ii) TS_{U_j}; (iii) P_j; (iv) $h(\cdot)$; (v) R_{U_j}; (vi) $k_{i_{U_j}}$ for each leaf node $i \in P_j$; (vii) Y_j; and (viii) $m + m'$ key-plus-id combinations $\left\{\left(s_{ij}, ID_{CH_i}\right) \mid 1 \le i \le m + m'\right\}$. We assume that after initial deployment, m' cluster heads may be added into the network later. Finally, the smart card is sent to the user U_j via a secure channel by the BS.

After receiving the smart card securely from the BS, the user U_j stores x_j into the smart card. This registration phase of our scheme is summarized in Figure 4.

User (U_j)	BS
Selects ID_{U_j} and PW_j, random value x_j. Computes $RPW_j = h(ID_{U_j}\|\|x_j\|\|PW_j)$. $\xrightarrow{\langle ID_{U_j}\|\|RPW_j\rangle}$ (via a secure channel)	
	Computes $R_{U_j}, k_{i_{U_j}}$. $\xleftarrow{\langle Smartcard(G, TS_{U_j}, P_j, Y_j, h(\cdot), R_{U_j}, k_{i_{U_j}}, \{(s_{ij}, CH_i) \mid 1 \le i \le m+m'\}\rangle}$ (via a secure channel)

Figure 4. Registration phase of our proposed scheme.

Security Comm. Networks 2015; **8**:1752–1771 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

Remark 1. As in [1], we choose the value of $m+m'$ based on the available memory of the smart card. We can store $m + m' = 200$ keys plus identifiers of the cluster heads in a smart card's memory. In practice, only a small number of cluster heads are expected to deploy for a large-scale HWSN. For instance, if each cluster has 200 sensor nodes deployed in that cluster, an HWSN will form a large-scale network with 20 000 sensor nodes, in which we require only 100 cluster nodes. As pointed out in [1], this is a practical assumption to store the computed $m = 100$ keys for the initial deployment of cluster heads and after that $m' = 100$ keys for dynamically added cluster heads into the smart card's memory.

Remark 2. We assume that the password PW_j of a user U_j is a very high entropy or strong password so that the offline/online password guessing attack will be difficult for an adversary without knowing both ID_{U_j} and PW_j. According to our threat model, the BS is considered as trustworthy. The BS keeps all the master keys of the deployed sensors and cluster heads. So, the BS will not reveal the master keys.

3.4. Login phase

The purpose of this phase is to login to the system by a valid user who wants to access the real-time data from the specified cluster head CH_i in HWSN. The user U_j executes the following steps:

Step 1: At the time of login, the user U_j first inserts the smart card into the card reader of a specific terminal, and then inputs his/her identity ID_{U_j} and password PW_j. The smart card then computes the masked password $RPW'_j = h\left(ID_{U_j}||x_j||PW_j\right)$ and $A'_j = h\left(ID_{U_j}||TS_{U_j}\right)$, using the stored value TS_{U_j}, which is the registration timestamp of the user U_j. The smart card then computes $R'_{U_j} = h\left(RPW'_j||A'_j\right)$ and checks the condition $R'_{U_j} = R_{U_j}$. If this verification does not hold, it means that U_j has entered incorrect identity and password. In this case, the smart card will terminate this phase immediately.

Step 2: The smart card selects a secret 160-bit random nonce n_j and computes $N_j = n_jG$.

Step 3: After that the smart card computes $SK_j = n_jY_j$ and $M_j = h\left(R'_{U_j}\left|\right|T_{U_j}\left|\right|N_{j_x}\right)$, where T_{U_j} is the current time stamp of the user U_j and N_{j_x} the x-coordinate of the point N_j.

Step 4: U_j then selects the cluster head CH_i from which he or she wants to access the real-time information inside HWSN. After selecting, the user U_j computes $T_{ij} = T_{U_j} \oplus s_{ij}$.

Step 5: Finally, the user U_j sends the login request message $\left\langle N_j||E_{SK_{jx}}\left(ID_{U_j}\left|\right|ID_{CH_i}\left|\right|T_{ij}||M_j\right)\right\rangle$ to the BS via a public channel, where SK_{jx} denotes the secret key, which is the x-coordinate of the point SK_j.

User (U_j)	BS
Inserts the smart card and inputs ID_{U_j}, PW_j. Computes $RPW'_j = h(ID_{U_j}\|\|x_j\|\|PW_j)$, $A'_j = h(ID_{U_j}\|\|TS_{U_j})$, $R'_{U_j} = h(RPW'_j\|\|A'_j)$. Verifies if $R'_{U_j} = R_{U_j}$? If so, selects n_j and computes $N_j = n_jG$, $SK_j = n_jY_j$ and $M_j = h(R'_{U_j}\|\|T_{U_j}\|\|N_{j_x})$. Selects CH_i and computes $T_{ij} = T_{U_j} \oplus s_{ij}$. $\xrightarrow{\langle N_j\|\|E_{SK_{jx}}(ID_{U_j}\|\|ID_{CH_i}\|\|T_{ij}\|\|M_j)\rangle}$ (via a public channel)	

Figure 5. Login phase of our proposed scheme.

The summary of our login phase is provided in Figure 5.

3.5. Authentication phase

When the login message $\left\langle N_j||E_{SK_{jx}}\left(ID_{U_j}\left|\right|ID_{CH_i}\right|\right|$ $T_{ij}||M_j)\rangle$ is received from the user U_j, the BS performs the following steps:

Step 1: The BS first computes $SK'_j = y_jN_j$ and decrypts the encrypted received message $E_{SK_{jx}}\left(ID_{U_j}\left|\right|ID_{CH_i}\right|\right|$ $T_{ij}||M_j)$ using the computed shared key SK'_{jx} of the point SK'_j.

Step 2: The BS then computes $T'_{U_j} = T_{ij} \oplus s_{ij}$, and using T'_{U_j} and N_j it then computes $M'_j = h\left(R_{U_j}\left|\right|T'_{U_j}\left|\right|N_{j_x}\right)$. The BS then checks the value of M'_j with the received value M_j. If there is a match, the user U_j is considered as a legitimate user. Otherwise, this phase is terminated immediately.

Step 3: The BS computes $T_1 = h\left(T'_{U_j}||T_{BS}\right)$, where T_{BS} represents the current timestamp of the BS. The BS then sends the authentication request message $\left\langle E_{MK_{CH_i}}\left(ID_{U_j}\left|\right|ID_{CH_i}\left|\right|T_{ij}||T_{BS}||N_j||T_1||Y_i||TS_{U_j}\right)\right\rangle$ to the corresponding cluster head CH_i via a public channel.

Step 4: When CH_i receives the message in Step 3, it decrypts $E_{MK_{CH_i}}\left(ID_{U_j}\left|\right|ID_{CH_i}\right|\right|T_{ij}||T_{BS}||N_j||$ $T_1||Y_i||TS_{U_j})$ using its own master key MK_{CH_i} as $D_{MK_{CH_i}}\left[E_{MK_{CH_i}}\left(ID_{U_j}\left|\right|ID_{CH_i}\right|\right|T_{ij}||T_{BS}||N_j||T_1||Y_i||$ $TS_{U_j})\big] = \left(ID_{U_j}||CH_i||T_{ij}||T_{BS}||N_j||T_1||Y_i||TS_{U_j}\right)$. CH_i then checks if the retrieved CH_i is equal to the received CH_i. If this holds, CH_i further checks if $\left|T_{BS} - T^*_{BS}\right| < \triangle T_{BS}$, where T^*_{BS} is the current system timestamp of the CH_i and $\triangle T_{BS}$ the expected time interval for the transmission delay. If this condition holds true, CH_i computes

BS	CH_i
Computes $SK'_j = y_j N_j$ and retrieves ID_{CH_i}, T_{ij}, M_j using SK'_{jx}. Computes $T'_{U_j} = T_{ij} \oplus s_{ij}$ and using T'_{U_j} computes $M'_j = h(R_{U_j} \|\| T'_{U_j} \|\| N_{j_x})$ Verifies if $M'_j = M_j$? If so, sends $\xrightarrow{\langle E_{MK_{CH_i}}(ID_{U_j} \|\| ID_{CH_i} \|\| T_{ij} \|\| T_{BS} \|\| N_j \|\| T_1 \|\| Y_i \|\| TS_{U_j})\rangle}$	
	Retrieves $ID_{U_j}, ID_{CH_i}, T_{ij}, T_{BS}, N_j, T_1, Y_i, TS_{U_j}$. Checks if $\mid T_{BS} - T^*_{BS} \mid < \triangle T_{BS}$? If so, computes $s_{ij} = h(TS_{U_j} \|\| (T_{CH_i} \oplus W_{CH_i}))$, $T'_{U_j} = T_{ij} \oplus T_{CH_i}, T'_1 = h(T'_{U_j} \|\| T_{BS})$. Verifies if $T'_1 = T_1$? If it is valid, executes master key encryption phase.

Figure 6. Authentication phase of our proposed scheme.

$s_{ij} = h\left(TS_{U_j} \| \left(T_{CH_i} \oplus W_{CH_i}\right)\right)$ with its own bootstrapping time T_{CH_i} and expiration time W_{CH_i}, and the received value of registration timestamp TS_{U_j} of the user U_j.

Step 5: The cluster head CH_i computes $T'_{U_j} = T_{ij} \oplus T_{CH_i}$ and $T'_1 = h\left(T'_{U_j} \| T_{BS}\right)$ using the retrieved value of T'_{U_j}, and then checks if $T'_1 = T_1$. If it does not hold, this phase is terminated immediately. Otherwise, the user U_j is considered as a valid user by CH_i. Then, CH_i executes the master key encryption phase for the user U_j. U_j also performs the data decryption phase after receiving the authentication reply message from CH_i.

The authentication phase of our scheme is summarized in Figure 6.

3.5.1. Master key and data encryption phase.

This phase has the following steps:

Step 1: The cluster head CH_i computes $K_3 = (B_i + Y) \pmod p$ using the stored values B_i and Y, and then computes $K_{2i} = (B_i + T_i) \pmod p$ for all attributes i, where $\forall i \in I_i$ for that particular cluster head CH_i.

Step 2: As per the user U_j request, the cluster head CH_i computes $K_{ij} = h\left(ID_{CH_i} \left|\right| ID_{U_j} \left|\right| T_{U_j} \left|\right| N_{j_x} \left|\right| T_{CH_i}\right)$ and $K_{sj} = h(K_{ij} \| K_3)$ for the user U_j. CH_i then encrypts the sensing information (plaintext message) M using K_{sj} as $M' = E_{K_{sj}}(M)$.

Step 3: CH_j then sends the authentication reply message $\left\langle M' \left|\right| TS_{CH_i} \left|\right| \left(T_{CH_i} \oplus W_{CH_i}\right) \left|\right| T_{CH_i} \left|\right| h\left(ID_{U_j} \left|\right| ID_{CH_i} \left|\right| TS_{CH_i} \| M'\right) \| K_{2i}, \forall i \in I_i \right\rangle$ to the user U_j with its current time stamp TS_{CH_i} via a public channel.

We summarize the master key and data encryption phase of our proposed scheme in Figure 7.

CH_i	User (U_j)
Selects K_s and computes $K_{ij} = h(ID_{CH_i} \|\| ID_{U_j} \|\| T_{U_j} \|\| N_{j_x} \|\| T_{CH_i})$ and $K_{sj} = h(K_s \|\| K_{ij} \|\| K_3)$. Encrypts $M' = E_{K_{sj}}(M)$. $\xrightarrow{\langle M' \|\| TS_{CH_i} \|\| (T_{CH_i} \oplus W_{CH_i}) \|\| T_{CH_i} \|\| h(ID_{U_j} \|\| ID_{CH_i} \|\| TS_{CH_i} \|\| M') \|\| K_{2i}, \forall i \in I_i \rangle}$	

Figure 7. Master key and data encryption phase of our scheme.

3.5.2. Data decryption phase.

This phase has the following three steps:

Step 1: After receiving the message from the cluster head CH_i during the master key and data encryption phase, the user U_j first computes the hash value $h\left(ID_{U_j} \left|\right| ID_{CH_i} \left|\right| TS_{CH_i} \| M'\right)$ with the received values of TS_{CH_i} and M'. U_j then checks this hash value with the received hash value. If the computed hash value is not equal to the received hash value, this phase is terminated immediately. Otherwise, the user U_j checks if $\left|TS_{CH_i} - TS^*_{CH_i}\right| < \triangle TS_{CH_i}$, where $TS^*_{CH_i}$ is the current system timestamp of the user U_j and $\triangle TS_{CH_i}$ the expected time interval for the transmission delay. If it holds, the user U_j computes $W_{CH_i} = \left(T_{CH_i} \oplus W_{CH_i}\right) \oplus T_{CH_i}$, and checks if $W_{CH_i} \geq TS_{CH_i}$. If it does not hold, this phase is terminated immediately. Otherwise, the cluster head CH_i is considered as a legitimate node by the user U_j.

Step 2: After authenticating the cluster head CH_i, the user U_j decrypts the message to recover the original message M as follows. The decryption process is executed from the leaf nodes of its own access tree, and in the bottom-up approach. The user U_j computes F_a for each leaf node a in P_j using the following logic: if $a \in I_i$, then $F_a = Access\left(k_{iU_j} g + K_{2i}\right)$; else $F_a = \perp$ (invalid).

Security Comm. Networks 2015; **8**:1752–1771 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

For the user U_j, the access tree is P_j for which the root node is r. If P_{j_x} is the sub-tree of P_j, P_{j_x} is rooted at the node x. If a set of attributes I_i satisfies the access tree P_{j_x}, then only we obtain $Access\left(k_{iU_j}g + K_{2i}\right) = (y + b_i)g \pmod p$, which is shown in the succeeding text:

$$\begin{aligned} Access\left(k_{iU_j}g + K_{2i}\right) &= ((q_x(0) - t_i)g \\ &\quad + (B_i + T_i)) \pmod p \\ &= ((q_r(0) - t_i)g \\ &\quad + (b_i + t_i)g \pmod p \\ &= (y + b_i)g \pmod p \\ &= yg + b_i g \pmod p \\ &= K_3', say \end{aligned}$$

where $q_x(0) = q_{parent(x)}(index(x))$, $q_r(0) = y$ and $q_x(0) = q_{parent(x)}(index(x))$ is executed in a recursive way as explained in [14].

Step 3: The user U_j computes $K_{ij} = h\left(ID_{CH_i}\left|\right|ID_{U_j}\left|\right| T_{U_j}\left|\right|N_{j_x}\left|\right| T_{CH_i}\right)$ and $K_{sj} = h\left(K_{ij}||K_3'\right)$, and then decrypts M' using K_{sj} to retrieve M as $D_{K_{sj}}(M') = M$.

3.6. Password change phase

This phase helps a legal user U_j to update his/her current password freely and completely locally without contacting the BS for security reasons. For this purpose, we have the following steps:

Step 1: U_j inputs the smart card into a card reader of the specific terminal. After that, he/she provides his/her identity ID_{U_j} and the old password PW_j^{old} and new changed password PW_j^{new}.The smart card computes $RPW_j^{old} = h\left(ID_{U_j}||x_j||PW_j^{old}\right)$, $A_j' = h\left(ID_{U_j}\left|\right|TS_{U_j}\right)$ using the stored value TS_{U_j}, and $R_{U_j}' = h\left(RPW_j^{old}\left|\right|A_j'\right)$.

Step 2: The smart card then checks if $R_{U_j}' = R_{U_j}$. If there is a mis-match, it ensures that U_j inputs his/her old password incorrectly, and as a result, the phase is then terminated immediately. Otherwise, the smart card executes Step 3.

Step 3: The smart card computes the new masked password $RPW_j^{new} = h\left(ID_{U_j}||x_j||PW_j^{new}\right)$ and $R_{U_j^{new}} = h\left(RPW_j^{new}\left|\right|A_j'\right)$.

Step 4: Finally, the smart card replaces R_{U_j} with the new masked password $R_{U_j}^{new}$ into its memory.

3.7. New node addition phase

The purpose of this phase is to deploy fresh nodes to join in the existing network. Addition of news nodes are extremely important, because some sensor nodes or cluster heads may be captured by an attacker or some nodes may expire due to energy problem.

Case I. Joining new sensor nodes
In this case, if a sensor node SN_i is deployed in a cluster C_i, prior to its deployment, the BS needs to assign the unique identifier ID_{SN_i} and also a randomly generated unique master key MK_{SN_i}. After that, ID_{SN_i} and MK_{SN_i} are loaded in the memory of the node SN_i.

Case II. Joining new cluster heads
The first selects a cluster head CH_i' whose information are already in the user U_j's smart card. Assume that the cluster head CH_i' be deployed in the cluster C_i. Prior to its deployment, the BS loads the unique identifier $ID_{CH_i'}$ and the unique master key $MK_{CH_i'}$ as described in Section 3.3.

After deployment of new sensor nodes in their corresponding cluster along with its new cluster head, the BS needs to inform the user U_j about the addition of the cluster heads. As a result, it is clear that we do not require to store any other information regarding cluster heads addition in the user's smart card.

4. ANALYSIS OF OUR SCHEME

In this section, we evaluate the functionality and security of our proposed user access control scheme.

4.1. Computational overhead

We consider the computational overhead of our scheme for the login and authentication phases. Let t_{ecm}, t_h, t_i, t_{enc}, t_{dec}, and t_{eca} denote the time for performing an ECC-point multiplication, a one-way hash function $h(\cdot)$, a modular inverse, a symmetric key encryption, a symmetric key decryption and an ECC-point addition, respectively. In our scheme, the cluster head CH_j requires one symmetric-key decryption t_{dec}, one symmetric-key encryption t_{enc}, five hash operations t_h, and four ECC-point addition t_{eca} for the login and authentication purposes. The user U_j needs seven t_h, four ECC-point multiplication t_{ecm}, three ECC-point additions, and one symmetric-key encryption t_{enc} for login and authentication phases. The BS requires one ECC-point multiplication t_{ecm}, one symmetric-key decryption t_{dec}, two hash operations t_h, and one symmetric-key encryption t_{enc} for the login, and authentication purposes. As a total, the computational overhead is $14t_h + 3t_{enc} + 2t_{dec} + 5t_{ecm} + 7t_{eca}$.

4.2. Communication overhead

From the user registration, login and authentication phases described in Section 3, it is clear that for our proposed scheme, the BS, cluster head and user need to exchange only four messages among them. However, the cluster heads are involved for only two message exchanges among them for query response to the user U_j. We show the bitwise and packetwise communication overheads for our proposed scheme. For computing the number of packets required, we have considered CC2420 transmitter [30]. CC2420 transmitter supports a packet size of 128 bytes (1024 bits).

We also assume that the universe of all sensor attributes $\mathcal{I}$ contains 16 attributes and the average number of sensor attributes of any cluster head CH_i is 3, that is, $|I_i| = 3$. It requires 320 bits to store the sensor attributes $\mathcal{I}_i$ of a cluster head CH_i. It is to be noticed that the parameters K_{2i} take $3 \times 320 = 960$ bits, $\forall i \in I_i$.

Table II shows the different parameters in bits used in our scheme. In Table III, we have calculated the number of bits and packets required for each message in our user access control scheme for different phases. We see that only two packets are transmitted, when the cluster heads are involved.

4.3. Security analysis

In this section, through both informal and formal security analysis, we show that our scheme has the ability to tolerate different known attacks.

4.3.1. Informal security analysis.

Our scheme has the ability to defend the following attacks:

Stolen-verifier attack: The systems, which store the verifier/password table for password verification, are generally susceptible to the stolen-verifier attack. Note that our scheme does not need to store any verifier/password table for password verifications. Thus, our scheme has the ability to prevent such attack.

Many logged-in users with the same login-id attack: In this attack, if more than one legitimate user have same login-id and password, any of those users can launch this attack. But, in our proposed scheme, verifier/password tables are not required for password verifications. In our scheme, during the login phase, the user U_j gives his/her identity ID_{U_j} and password PW_j, and the smart card then computes the masked password $RPW'_j = h\left(ID_{U_j}||x_j||PW_j\right)$ and $A'_j = h\left(ID_{U_j}||TS_{U_j}\right)$ using the stored value TS_{U_j}, which is the registration timestamp of the user U_j. The smart card then computes $R'_{U_j} = h\left(RPW'_j||A'_j\right)$ and checks if $R'_{U_j} = R_{U_j}$. If two different users have the same password, then also the registration timestamp and the random number x_j stored in the smart card are not be same so that the value of R_{U_j} is always different for any other user. As a result, our scheme resists this attack.

Resilience against node capture attack: Assume that c nodes are randomly captured by an attacker in the sensor network. Let $P_e(c)$ denote the probability that the adversary compromises a fraction of total secure communications using c compromised nodes. If $P_e(c) = 0$, our user access control scheme is called unconditionally secure against node capture attack. Assume that the attacker captures a sensor node or a cluster head. Then, the attacker will know the master key along with other information from its memory, because the sensor nodes or cluster heads are not equipped with tamper-resistant hardware. Each node is preloaded with a unique randomly generated master key prior to its deployment. Further, each cluster head can establish a secret session key with a user, which is different from other session keys. As a result, the attacker has the ability to respond with false data to a legitimate user by capturing a cluster head from which the user wants to access data. Other non-captured cluster heads have the ability to communicate with 100% secrecy with the actual real-time data to the legitimate users. Consequently, the compromise of a sensor node or cluster head does not lead to compromise of any other secure communication between the user and the non-compromised cluster heads in the network. In this way,

Table II. Bitsize of the parameters used in our scheme.

Type	Bitwise size
User identifier, ID_{U_j}	16
Bootstrapping time, T_i	32
Node identifier ID_{CH_i}	16
Hash value (SHA-1)	160
Symmetric-key encryption/decryption	128

Table III. Message size and number of packets to be transmitted per message for the proposed access control scheme.

Message	Exchange between	Size (bits)	# of packets required
$\left\langle ID_{U_j}\|\|RPW_j\right\rangle$	U_j to BS	176	1
$\left\langle N_j\|\|E_{SK_{jx}}\left(ID_{U_j}\|\|ID_{CH_i}\|\|T_{ij}\|\|M_j\right)\right\rangle$	U_j to BS	448	1
$\left\langle E_{MK_{CH_i}}\left(ID_{U_j}\|\|ID_{CH_i}\|\|T_{ij}\|\|T_{BS}\|\|N_j\|\|T_1\|\|Y_i\|\|TS_{U_j}\right)\right\rangle$	BS to CH_i	128	1
$\left\langle M'\|\|TS_{CH_i}\|\|(T_{CH_i}\oplus W_{CH_i})\|\|T_{CH_i}\|\|h\left(ID_{U_j}\|\|ID_{CH_i}\|\|TS_{CH_i}\|\|M'\right) \|\| K_{2i}, \forall i \in I_i\right\rangle$	CH_i to U_j	1360	2
	Total	2112	5

the unconditional security against node capture attack is achieved in our scheme.

Masquerade attack: In such an attack, an attacker may try to fabricate the fake login request message in order to cheat the BS and then convince that the fake message is a legal request during the login phase. However, in our scheme at the time of login phase, the legal user U_j needs to insert his/her smart card in a card reader and then provide his/her identity ID_{U_j} and password PW_j. The smart card then computes the masked password $RPW'_j = h\left(ID_{U_j}||x_j||PW_j\right)$ and the hash value $A'_j = h\left(ID_{U_j}\middle|\middle|TS_{U_j}\right)$ using the stored registration timestamp TS_{U_j}. The smart card then computes $R'_{U_j} = h\left(RPW'_j\middle|\middle|A'_j\right)$ and checks whether $R'_{U_j} = R_{U_j}$. If this verification is successful, the user U_j computes SK_j based on Y_j stored in the smart card and M_j using the computed value R'_{U_j}, and then sends the login request message $\left\langle N_j||E_{SK_j}\left(ID_{U_j}\,||ID_{CH_i}||\,T_{ij}||M_j\right)\right\rangle$ to the BS. In order to convince the BS that this is a legal remote login request, the illegal user has to know the value of T_{ij} as well as PW_j, ID_{U_j}, and TS_{U_j}. As a result, the attacker does not have then any ability to create a fake login request message on behalf of the original user U_j. Thus, our scheme also resists this attack.

Replay and man-in-the-middle attacks: In this attack, an attacker may try to prove as a valid user logging to the BS by sending messages that were previously delivered by a legal user U_j. However, our scheme makes use of the current system timestamp during the login and authentication phases. Comparison of previous timestamp with current timestamp of the receiver system withstands the replay attack, because the expected time interval for the transmission delay is very short. Moreover, in the login phase, the user U_j sends the value of M_j, where $M_j = H\left(R'_{U_j}\middle|\middle|T_{U_j}\middle|\middle|N_{j_x}\right)$, where T_{U_j} is the current time stamp of the user U_j, to the BS. Because the attacker can not change the hash value M_j, he or she cannot also change the value of T_{U_j}. Thus, an attacker does not have any ability to successfully replay previously used messages during the login and authentication phases. As a result, our scheme resists both replay and man-in-the-middle attacks.

Offline password guessing attack: Suppose the smart card of a legal user U_j has lost or stolen. Then, the attacker/intruder can extract its all stored sensitive information such as G, TS_{U_j}, P_j, Y_j, $h(\cdot)$, R_{U_j}, x_j, and $k_{i_{U_j}}$ for each leaf node $i \in P_j$, and also $m + m'$ key-plus-id combinations $\{(s_{ij}, ID_{CH_i}) \mid 1 \leq i \leq m + m'\}$. However, the attacker does not have any ability to know the user U_j's password PW_j from R_{U_j} from the hash value $R_{U_j} = h(RPW_j||A_j)$ due to collision-resistant one-way property of the hash function $h(\cdot)$. Hence, the attacker needs to guess the user U_j's correct password PW_j as well as the identity ID_{U_j}. Because both PW_j and ID_{U_j} are unknown to the attacker, it is a computationally infeasible problem for that attacker to guess PW_j correctly. As a result, our scheme prevents this attack through the stolen smart card attack.

Denial-of-service attack: In our scheme, after deployment, the user U_j first sends a login request message to the BS. At the time of authentication, the BS sends the authentication request message to a specific cluster head CH_i, from which the user U_j wants to access real-time data inside HWSN. After receiving the request message from the BS, the cluster head CH_i sends an acknowledgment (authentication reply) to the user U_i for mutual successful authentication. If an attacker blocks the messages from reaching the BS and cluster heads, the BS and cluster heads will know about malicious dropping of such control messages. Hence, the denial-of-service attack is prevented in our scheme.

Privileged-insider attack: Note that during the registration phase of our proposed scheme, the user U_j does not send his/her password PW_j in plaintext. The user U_j sends the masked password $RPW_j = h\left(ID_{U_j}||x_j||PW_j\right)$ along with the identity ID_{U_j} to the BS via a secure channel. Without knowing the secret random number x_j, which is only known to the user U_j, it is a computationally infeasible problem to retrieve PW_j from RPW_j due to collision-resistant one-way property of the hash function $h(\cdot)$. Thus, a privileged-insider of the BS does not have any ability to know the password PW_j of the user U_j. As a result, our scheme protects such an attack from revealing the user U_j's password.

Protection of user anonymity: Note that during the login phase, a legal user U_j sends the login request message $\left\langle N_j||E_{SK_{jx}}\left(ID_{U_j}\,||ID_{CH_i}||\,T_{ij}||M_j\right)\right\rangle$ to the BS. The BS sends the authentication request message $\left\langle E_{MK_{CH_i}}\left(ID_{U_j}\,||ID_{CH_i}||\,T_{ij}||T_{BS}||N_j||T_1||Y_i||TS_{U_j}\right)\right\rangle$ to the cluster head CH_i for authentication. In reply, CH_i sends the authentication reply message $\langle M'\,||TS_{CH_i}||\left(T_{CH_i} \oplus W_{CH_i}\right)||T_{CH_i}||\, h\left(ID_{U_j}\,||ID_{CH_i}||\,TS_{CH_i}||M'\right)||K_{2i}, \forall i \in I_i\rangle$ back to the user U_j. Note that none of the messages include the identity ID_{U_j} of the user U_j in plaintext and it is protected by the one-way hash function $h(\cdot)$ and symmetric-key encryption. Due to the collision-resistant one-way property of $h(\cdot)$ and secure symmetric-key cryptosystem (for example, AES [31]), the adversary does not have any ability to derive the identity ID_{U_j} of U_j. Hence, our scheme protects the user anonymity property.

4.3.2. Formal security analysis.

In this paper, we follow the formal security analysis by the method of contradiction as in [32,33]. For the formal security analysis of our scheme, we first define the one-way collision resistant hash function as follows:

Definition 1 (Formal definition of one-way collision resistant hash function [34,35]). *A collision-resistant one-way hash function $h : A \rightarrow B$, where $A = \{0,1\}^*$ and $B = \{0,1\}^n$, is a deterministic algorithm that takes an input as an arbitrary length binary string $x \in A$ and produces an output $y \in B$ as a binary string of fixed-length, n. Let $Adv_{\mathcal{A}}^{HASH}(t_1)$ denote an adversary (attacker) $\mathcal{A}$'s advantage in finding collision. Then, we have*

$$Adv_{\mathcal{A}}^{HASH}(t_1) = Pr[(u,v) \Leftarrow_R \mathcal{A} : u \neq v \text{ and } h(u) = h(v)],$$

where $Pr[E]$ denotes the probability of a random event E, and $(u,v) \Leftarrow_R \mathcal{A}$ denotes the pair (x,x') is selected randomly by $\mathcal{A}$. In this case, the adversary $\mathcal{A}$ is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by the adversary $\mathcal{A}$ with the execution time t_1. $h(\cdot)$ is then called collision-resistant, if $Adv_{\mathcal{A}}^{HASH}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$.

The elliptic curve DLP (ECDLP) is formally defined as in [33] as follows.

Definition 2 (Formal definition of ECDLP [33]). *Let $E_p(a,b)$ denote an elliptic curve over a prime finite field Z_p, and $P \in E_p(a,b)$ and $Q = kP \in E_p(a,b)$ be two points, where $k \in_R Z_p$ (We use the notation $x \Leftarrow_R T$ to denote that the number x is chosen randomly from the set T).*

Instance: (P,Q,l) for some $k,l \in_R Z_p$.
*Output: **yes**, if $Q = lP$, that is, $k = l$, and output **no**, otherwise.*
Consider the following two distributions

$$D_{real} = \{k \Leftarrow_R Z_p, X = P, Y = Q, Z = k : (A,B,C)\},$$
$$D_{rand} = \{k,l \Leftarrow_R Z_p, X = P, Y = Q, Z = l : (A,B,C)\}.$$

The advantage of any probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$ in solving ECDLP on $E_p(a,b)$ is defined as $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP} = |Pr[(X,Y,Z) \leftarrow D_{real} : \mathcal{D}(X,Y,Z) = 1] - Pr[(X,Y,Z) \leftarrow D_{rand} : \mathcal{D}(X,Y,Z) = 1]|$, where the probability $Pr[\cdot]$ is considered over the random choices of k and l. We then call $\mathcal{D}$ is a (t_2,ϵ_2)-ECDLP distinguisher for $E_p(a,b)$ if $\mathcal{D}$ runs at most in time t_2 such that $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t_2) \geq \epsilon_2$.

ECDLP assumption: There exists no (t_2,ϵ_2)-ECDLP distinguisher for $E_p(a,b)$. In other words, for every probabilistic, polynomial-time 0/1-valued distinguisher $\mathcal{D}$, we have $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t_2) \leq \epsilon_2$, for any sufficiently small $\epsilon_2 > 0$.

The following two oracles are defined for our formal security analysis:

- *Reveal1*: It will unconditionally output the input x from the corresponding hash value $y = h(x)$.
- *Reveal2*: Given $P \in E_p(a,b)$ and $Q = kP \in E_p(a,b)$, it will unconditionally output the discrete logarithm k.

Theorem 1. *Under the ECDLP assumption, our proposed scheme is secure against an adversary for deriving the password PW_j of a legal user U_j even if the smart card of U_j is lost or stolen, if the one-way hash function $h(\cdot)$ closely behaves like an oracle.*

Proof. In this proof, we aim to construct an adversary, say $\mathcal{A}$ who will have the ability to derive the password PW_j of a legal user U_j. We assume that the adversary $\mathcal{A}$ attains the smart card of the user U_j and can extract all the stored information from that smart card. We follow the similar proof as in [32,33]. The adversary $\mathcal{A}$ uses the oracles *Reveal*1 and *Reveal*2 for running the experimental algorithm, say $EXP_{\mathcal{A},UACS}^{HASH,ECDLP}$ provided in Algorithm 1 for our proposed user access control scheme, say *UACS*.

We define the success probability for the experiment $EXP_{\mathcal{A},UACS}^{HASH,ECDLP}$ provided in Algorithm 1 as $Succ_{\mathcal{A},UACS}^{HASH,ECDLP} = \left|2Pr\left[EXP_{\mathcal{A},UACS}^{HASH,ECDLP} = 1\right] - 1\right|$. The advantage function for this experiment then becomes $Adv_{\mathcal{A},UACS}^{HASH,ECDLP}(t, q_{R_1}, q_{R_2}) = max_{\mathcal{A}}\left\{Succ_{\mathcal{A},UACS}^{HASH,ECDLP}\right\}$, where the maximum is taken over all $\mathcal{A}$ with execution time t, and q_{R_1} and q_{R_2} are the number of queries made to the *Reveal*1 and *Reveal*2 oracles, respectively. We that call our scheme is provably secure against an adversary for deriving the password PW_j of a legal user U_j if $Adv_{\mathcal{A},UACS}^{HASH,ECDLP}(t, q_{R_1}, q_{R_2}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Otherwise, the adversary $\mathcal{A}$ will win the game.

According to the experiment, the adversary $\mathcal{A}$ can derive the password PW_j of the user U_j. However, deriving the input from a given hash value is a computationally infeasible task, because the hash function $h(\cdot)$ is collision resistant, that is, $Adv_{\mathcal{A}}^{HASH}(t_1) \leq \epsilon_1$ (provided in Definition 1), and due to difficulty of solving ECDLP, that is, $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t_2) \leq \epsilon_2$ (provided in Definition 2). As a result, $Adv_{\mathcal{A},UACS}^{HASH,ECDLP}(t, q_{R_1}, q_{R_2}) \leq \epsilon$, because $Adv_{\mathcal{A},UACS}^{HASH,ECDLP}(t, q_{R_1}, q_{R_2})$ is dependent on both $Adv_{\mathcal{A}}^{HASH}(t_1)$ and $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t_2)$. Hence, our scheme is provably secure against an adversary for deriving the password of any user. $\square$

Algorithm 1 $EXP^{HASH,ECDLP}_{\mathcal{A},UACS}$

Extract all the information G, TS_{U_j}, P_j, $h(\cdot)$, R_{U_j}, x_j, Y_j, $k_{i_{U_j}}$ for each leaf node $i \in P_j$, and $m + m'$ key-plus-id combinations $\{(s_{ij}, ID_{CH_i}) \mid 1 \leq i \leq m + m'\}$, from the stolen/lost smart card of the user U_j. Note that $RPW_j = h(ID_{U_j} \| x_j \| PW_j)$, $A_j = h(ID_{U_j} \| TS_{U_j})$ and $R_{U_j} = h(RPW_j \| A_j)$.
Intercept the login request message $\langle N_j \| E_{SK_{jx}}(ID_{U_j} \| ID_{CH_i} \| T_{ij} \| M_j)\rangle$ during the login phase.
Call *Reveal*2 oracle on input N_j to retrieve n_j as $n'_j \leftarrow Reveal2(N_j)$.
Compute $SK'_j = n'_j Y_j$ using the public key Y_j of the BS.
Decrypt $u = E_{SK_{jx}}(ID_{U_j} \| ID_{CH_i} \| T_{ij} \| M_j)$ using the computed key SK'_{jx} to retrieve the information $ID_{U_j}, ID_{CH_i}, T_{ij}, M_j$ as $\left(ID'_{U_j}, ID'_{CH_i}, T'_{ij}, M'_j\right) = D_{SK'_{jx}}[u]$.
Using the extracted TS_{U_j} and computed ID'_{U_j}, compute $A'_j = h\left(ID'_{U_j} \| TS_{U_j}\right)$.
Call *Reveal*1 oracle on input R_{U_j} to retrieve RPW_j and A_j as $\left(RPW^*_j, A^*_j\right) \leftarrow Reveal1(R_{U_j})$.
if $\left(A^*_j = A'_j\right)$ **then**
    Call *Reveal*1 oracle on input RPW^*_j to retrieve ID_{U_j}, x_j and PW_j of the user U_j as $\left(ID^*_{U_j}, x^*_j, PW^*_j\right) \leftarrow Reveal1\left(RPW^*_j\right)$.
    if $\left(x^*_j = x_j\right)$ and $\left(ID^*_{U_j} = ID'_{U_j}\right)$ **then**
        Accept PW^*_j as the correct password PW_j of the user U_j.
        return 1 (Success)
    else
        return 0 (Failure)
    end if
else
    return 0 (Failure)
end if

5. SIMULATION RESULTS FOR FORMAL SECURITY VERIFICATION OF OUR SCHEME USING AVISPA TOOL

In this section, we first describe in brief the overview of AVISPA tool along with the high-level protocol specification language (HLPSL). We then give the implementation details of our scheme in HLPSL. Finally, we discuss the analysis of the simulation results using AVISPA back-end.

5.1. AVISPA tool

We have used the widely-accepted AVISPA back-ends for our formal security verification [36–39]. AVISPA is considered as a push-button tool for the automated validation of Internet security-sensitive protocols and applications [40,41]. It consists of four back-ends: (i) On-the-fly Model-Checker (OFMC); (ii) Constraint-Logic-based Attack Searcher; (iii) SAT-based Model-Checker; and (iv) Tree Automata based on Automatic Approximations for the Analysis of Security Protocols.

In AVISPA, the protocols are implemented in HLPSL [42], which is based on roles: basic roles for representing each participant role, and composition roles for representing scenarios of basic roles. In HLPSL, the intruder is modeled using the Dolev-Yao model [17] (as in the threat model used in our scheme), who takes a legitimate role in a protocol run.

The output format of AVISPA is generated by using one of the four back-ends: OFMC, Cl-AtSe, SAT-based Model-Checker and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols. In output format, there are the following sections:

- SUMMARY section indicates that whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive.
- Next section, called DETAILS, either explains under what condition the tested protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive.
- Other sections such as PROTOCOL, GOAL, and BACKEND represent the name of the protocol, the goal of the analysis and the name of the back-end used, respectively.
- Finally, after some comments and statistics, the trace of an attack (if any) is also printed in the standard Alice-Bob format.

The basic types available in HLPSL are [40]:

- *agent: agent* denotes a principal name. The intruder has always the special identifier *i*.
- *public_key:* It denotes the agents' public keys in a public-key cryptosystem. As an example, given a public (respectively, private) key *pu*, its inverse private (respectively, public) key is denoted by *inv_pu*.
- *symmetric_key:* It says the key for a symmetric-key cryptosystem.
- *text: text* values are often used as nonces. They can be also applied for messages. If N denotes the type *text (fresh)*, then N' becomes a fresh value, which the intruder cannot guess easily.
- *nat:* It is used for denoting the natural numbers in non-message contexts.
- *const:* This type represents constants.
- *hash_func:* The base type *hash_func* represents cryptographic hash functions. It is assumed that the intruder cannot invert hash functions (in essence, that they are one-way).

```
role alice (Uj, CHi, BS   : agent, SKubs  : symmetric_key,
   SKj  : symmetric_key, MKchi :symmetric_key,
         H : hash_func, Snd, Rcv: channel(dy))
played_by Uj
def=
 local State  : nat,
      IDj, IDsi,Tij,TSuj,Tuj,Tchi,TSchi,Tbs,Wchi, RPWj,
      PWj,Xj,Pj,Kiuj,Nnj,G,Njx,Mj,M,Aj,T1,Bi,Y,Ks,K3,
      Kij,Ksj,Sij,Ruj:  text
 const alice_server,bs_bob,bob_alice, subs, sub3, sub1,
      sub2,sub4 : protocol_id
init  State := 0
transition
  1.State = 0      /\ Rcv(start)=|>
    State' :=1      /\ RPWj' :=H(IDj.Xj.PWj)
     /\ Snd(Uj.BS.{IDj.RPWj'}_SKubs)
                  /\ secret({PWj}, sub1, Uj)
                  /\ secret({SKubs}, sub3, {BS,Uj})
                  /\ secret({SKj}, subs, {BS,Uj})
 2. State   = 1     /\ Rcv({IDj.TSuj.Pj.H(H(Xj.PWj).
   H(IDj.TSuj')).Kiuj.H(TSuj'.xor(Tchi',Wchi')).IDsi}_SKubs) =|>
    State' := 2   /\ Tuj':= new()
   /\ Tij':= xor(Tuj',H(TSuj'.xor(Tchi',Wchi')))
     /\ Njx':=H(Nnj.G)
   /\ Mj':=H(H(H(IDj.Xj.PWj).H(IDj.TSuj')).Tuj'.Njx')
                /\ Snd(H(Nnj.G).{IDj.IDsi.Tij'.Mj'}_SKj)
                /\ witness(Uj, BS, alice_server, Tuj')
3.State   = 2        /\ Rcv(IDsi.{M}_H(Ks.H(IDj.IDsi.Tuj.H(Nnj.G).
           Tchi).H(Bi.Y)).TSchi'.xor(Tchi',Wchi').Tchi.H(IDj.IDsi.TSchi'.
            {M}_H(Ks.H(IDj.IDsi.Tuj.H(Nnj.G).Tchi).H(Bi.Y))))=|>
 State' := 3      /\ request(CHi,Uj, bob_alice, TSchi')
        /\ secret({Bi,Y}, sub4, {BS,CHi})
 end role
```

Figure 8. Role specification for the user U_j of our scheme.

```
role bs (Uj, CHi, BS   : agent, SKubs  : symmetric_key,
SKj  : symmetric_key, MKchi :symmetric_key,
      H : hash_func, Snd, Rcv: channel(dy))
played_by BS
def=
local State  : nat,
      IDj, IDsi,TSuj,Tchi,Tij,Tuj,TSchi,Tbs,Wchi,
      RPWj,PWj,Xj,Pj,Kiuj,Nnj,G,Njx,Mj,M,Aj,T1,Bi,
      Y,Ks,K3,Kij,Ksj,Sij,Ruj:  text
const alice_server,bs_bob,bob_alice, subs, sub3,
     sub1, sub2,sub4 : protocol_id
 init  State := 0
transition
1. State   = 0 /\ Rcv(Uj.BS.{IDj.H(IDj.Xj.PWj)}_SKubs) =|>
  State':=1  /\ TSuj':= new()
    /\ Tchi':= new()
    /\ Wchi':= new()
    /\ Aj':= H(IDj.TSuj')
     /\ Ruj':=H(H(IDj.Xj.PWj).Aj')
          /\ Sij':= H(TSuj'.xor(Tchi',Wchi'))
    /\ secret({PWj}, sub1, Uj)
          /\ secret({SKubs}, sub3, {BS,Uj})
          /\ secret({SKj}, subs, {BS,Uj})
    /\ secret({Bi,Y}, sub4, {BS,CHi})
          /\ Snd({IDj.TSuj'.Pj.Ruj'.Kiuj.Sij'.IDsi}_SKubs)
2. State=1    /\ Rcv(H(Nnj.G).{IDsi.xor(Tuj',H(TSuj'.
  xor(Tchi',Wchi'))).H(H(H(IDj.Xj.PWj).H(IDj.TSuj')).Tuj'.Njx')}_SKj) =|>
  State':= 2 /\ Tbs':= new()
          /\ T1':= H(Tuj'.Tbs')
          /\ Snd({IDj.IDsi.xor(Tuj',H(TSuj'.
            xor(Tchi',Wchi'))).Tbs'.H(Nnj.G).T1'.Tchi'}_MKchi)
         /\ witness(BS,CHi, bs_bob, Tbs')
         /\ request(Uj, BS, alice_server, Tuj')
end role
```

Figure 9. Role specification for the base station of our scheme.

```
role bob (Uj, CHi, BS   : agent, SKubs : symmetric_key,
SKj  : symmetric_key, MKchi :symmetric_key,
 H : hash_func, Snd, Rcv: channel(dy))
played_by CHi
def=
local State  : nat,
     IDj, IDsi,TSuj,Tuj,Tij,Tchi,TSchi,Tbs,Wchi, RPWj,
     PWj,Xj,Pj,Kiuj,Nnj,G,Njx,Mj,M,Aj,T1,Bi,Y,Ks,K3,
     Kij,Ksj,Sij,Ruj:  text
const alice_server,bs_bob,bob_alice, subs, sub3,
     sub1, sub2,sub4: protocol_id
init  State := 0
transition
1. State   = 0 /\ Rcv({IDj.IDsi.xor(Tuj',H(TSuj'
  .xor(Tchi',Wchi'))).Tbs'.H(Nnj.G).H(Tuj'.Tbs').Tchi'}_MKchi)  =|>
  State' := 1 /\ TSchi' := new()
    /\ K3' :=H(Bi.Y)
          /\ Kij':=H(IDj.IDsi.Tuj'.H(Nnj.G).Tchi')
          /\ Ksj' :=H(Ks.Kij'.K3')
    /\ secret({Bi,Y}, sub4, {BS,CHi})
          /\ secret(PWj, sub1, Uj)
          /\ secret(Ks, sub2, CHi)
          /\ secret({SKubs}, sub3, {BS,Uj})
          /\ secret({SKj}, subs, {BS,Uj})
          /\ request(BS, CHi, bs_bob, Tbs')
          /\ Snd(IDsi.{M}_Ksj'.TSchi'.xor(Tchi',Wchi').
            Tchi.H(IDj.IDsi.TSchi'.{M}_Ksj'))
          /\  witness(CHi,Uj, bob_alice, TSchi')
end role
```

Figure 10. Role specification for the cluster head CH_i of our scheme.

In HLPSL, the space of legal messages are considered as the closure of the basic types. For a given plaintext message M and the encryption key k, $\{M\}\_k$ means the symmetric/public-key encryption. The associative '·' operator is useful for concatenation purpose.

The declaration '*played_by U*' says that the agent named in variable U plays in the role. A knowledge declaration (generally in the top-level *Environment* role) is required in order to specify the intruder's initial knowledge. $X = | > Y$ means an immediate reaction transition, which relates an event X and an action Y. Whenever we take a transition that is labeled in such a way as to make the event predicate X true, one must immediately (that is, simultaneously) execute action Y. If we want to keep a variable A to be permanently secret, it must be considered by the goal *secrecy_of A*. Thus, if the variable A is ever obtained or derived by the intruder, a security violation will result in HLPSL. A detailed description on AVISPA and HLPSL can be found in [40,41].

5.2. Specifying our scheme

We have implemented our scheme in HLPSL language, where the three basic roles: alice, bs, and bob, represent the participants, namely the user U_j, the BS and cluster head CH_i, respectively. We have then defined the session and environment in our scheme.

Figure 8 illustrates the role specification for the user U_j in HLPSL. During the registration phase, U_j sends the registration request message $\langle ID_{U_j} || RPW_j \rangle$ securely to the BS with the *Snd*( ) operation. The type declaration *channel* (*dy*) indicates that the channel is for the Dolev-Yao threat model (as described in our threat model in Section 1.4). U_j then waits for the smart card securely from the BS from the *Rcv*( ) operation. In HLPSL, the intruder has the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. During the login phase, U_j sends the login request message $\langle N_j || E_{SK_j}(ID_{U_j} || ID_{CH_i} || T_{ij} || M_j) \rangle$ to the BS. After receiving

```
role session(Uj,CHi, BS: agent,SKubs : symmetric_key,
SKj  : symmetric_key, MKchi :symmetric_key,
       H : hash_func )
def=
  local  SI, SJ, RI, RJ,BI,BJ: channel (dy)
composition
     alice(Uj, BS,CHi,SKubs,SKj,MKchi,H, SI, RI)
/\  bs(Uj, BS, CHi, SKubs,SKj,MKchi,H, BI, BJ)
/\  bob(Uj,BS,CHi, SKubs, SKj,MKchi,H, SJ, RJ)
end role

role environment()
def=
  const uj,chi,bs : agent, skubs: symmetric_key,
       skj  : symmetric_key, mkchi :symmetric_key,
       h   : hash_func, idj, idsi,tsuj,tuj,tchi,
       tschi,tij,tbs,wchi, rpwj,pwj,ruj,sij,xj,pj,
       kiuj,nnj,g,njx,m,mj,aj,t1,bi,y,ks,k3,kij,ksj: text,
       alice_server, bs_bob, bob_alice, subs, sub3, sub1,
       sub2, sub4 : protocol_id
intruder_knowledge = {uj, chi, idsi, h}
  composition
session(uj,bs,chi,skubs,skj,mkchi,h) /\
session(uj,bs,chi,skubs,skj,mkchi,h)
end role

goal
  secrecy_of subs, sub1, subs2, subs3, subs4
  authentication_on alice_server
  authentication_on bs_bob
  authentication_on bob_alice
end goal
environment()
```

Figure 11. Role specification for the goal and environment of our scheme.

this message, the BS starts the authentication process. After successful authentication, the user U_j receives the authentication reply message $\langle M' || TS_{CH_i} || (T_{CH_i} \oplus W_{CH_i}) || T_{CH_i} || h(ID_{U_j} || ID_{CH_i} || TS_{CH_i} || M') || K_{2i}, \forall i \in I_i \rangle$ from the cluster head CH_i by the $Rcv(\,)$ operation.

Figure 9 shows the role specification for the BS in HLPSL language. During the registration phase after receiving the registration request message $\langle ID_{U_j} || RPW_j \rangle$ securely from the user U_j, the BS sends securely the smart card to the user U_j. In the login phase, when the BS receives the login request message $\langle N_j || E_{SK_{jx}}(ID_{U_j} || ID_{CH_i} || T_{ij} || M_j) \rangle$ from the user U_j, the BS first authenticates the user and then sends the authentication request message $\langle E_{MK_{CH_i}}(ID_{U_j} || ID_{CH_i} || T_{ij} || T_{BS} || N_j || T_1 || Y_i || TS_{U_j}) \rangle$ to the cluster head CH_i with the $Snd(\,)$ operation.

Figure 10 shows the implementation of the role specification for the cluster head CH_i in HLPSL. During the authentication phase, the cluster head CH_i receives the authentication request message $\langle E_{MK_{CH_i}}(ID_{U_j} || ID_{CH_i} || T_{ij} || T_{BS} || N_j || T_1 || Y_i || TS_{U_j}) \rangle$ from the BS by the $Rcv(\,)$ operation. Then, the cluster head CH_i sends the authentication reply message $\langle M' || TS_{CH_i} || (T_{CH_i} \oplus W_{CH_i}) || T_{CH_i} || h(ID_{U_j} || ID_{CH_i} || TS_{CH_i} || M') || K_{2i}, \forall i \in I_i \rangle$ back to the user U_j with the $Snd(\,)$ operation.

The declaration witness(A,B,id,E) is used for a (weak) authentication property of A by B on E. It declares that the agent A is witness for the information E. This goal is specified by the constant *id* in the goal section. The other declaration, request(B,A,id,E) is used for a strong authentication property of A by B on E. It means that the agent B requests a check of the value E. This goal is identified by the constant *id* in the goal section. In HLPSL, i is always denoted for the intruder.

Finally, the specifications in HLPSL language for the role of session, goal, and environment are specified in Figure 11. In the session segment, all the basic roles: alice, server, and bob are instanced with concrete arguments. The top-level role (environment) is always defined in the specification of HLPSL language. This role contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The standard authentication and secrecy goals are supported by the current version of HLPSL. In our scheme, five secrecy goals and three authentications are verified.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web-interface-computation/
  ./tempdir/workfile55k0vL.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.31s
  visitedNodes: 13 nodes
  depth: 4 plies
```

Figure 12. The result of the analysis using On-the-fly Model-Checker of our scheme.

Table IV. Time complexity of various operations in terms of t_{mul}.

$t_{ecm} \approx 1200t_{mul}$	$t_{sigver} \approx 2405.36t_{mul}$
$t_i \approx 3T_{mul}$	t_{add} is negligible
$t_h \approx 0.36t_{mul}$	$t_{enc} \approx 0.15t_{mul}$
$t_{dec} \approx 0.15t_{mul}$	$t_{ecenc} \approx 2405t_{mul}$
$t_{ecdec} \approx 1205t_{mul}$	$t_{mac} \approx t_h$
$t_{siggen} \approx 1204.36t_{mul}$	$t_{eca} \approx 5t_{mul}$

5.3. Analysis of results

We have chosen the back end OFMC for an execution test and a bounded number of sessions model checking [43]. For the replay attack checking, this back end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that this back end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev-Yao model check, this back end also checks whether there is any man-in-the-middle attack possible by the intruder. We have simulated our scheme for formal secu-

Table V. Computational cost comparison between our scheme and other schemes for the login and authentication phases.

Phase	User/Node	[25]	[24]	[23]	Ours
L	U_j	$t_h + t_{sigver}$ $+t_{mac}$	$t_{ecm} + 2t_{mac}$	$t_h + t_{sigver}$ $+t_{siggen}$	$7t_h + 4t_{ecm}$ $3t_{eca} + t_{enc}$
+	BS	$2t_{sigver} + 2t_{mac}$ $+2t_h$	–	–	$2t_h + t_{ecm} + t_{dec}$ $+t_{enc}$
A	SN_i/CH_j	$3t_{mac} + t_h$	$t_{eca} + 3t_{ecm}$ $+t_h + 2t_{mac}$	$2t_h + t_{siggen}$ $+t_{sigver}$	$5t_h + t_{dec}$ $+t_{enc} + 4t_{eca}$
	Total cost	$4t_h$ $+3t_{sigver} + 6t_{mac}$	$t_h + 3t_{ecm}$ $+t_{eca} + 4t_{mac}$	$3t_h + 2t_{siggen}$ $+2t_{sigver}$	$14t_h + 5t_{enc}/t_{dec}$ $+5t_{ecm} + 7t_{eca}$
	Rough estimation	$7,220t_{mul}$	$4,807t_{mul}$	$7,221t_{mul}$	$6,046t_{mul}$

Note: L, login phase; A, authentication phase.

rity verification using OFMC back end under the AVISPA web tool [40]. The simulation result for the formal security verification of our scheme using OFMC is shown in Figure 12. It is clear that our scheme is secure against passive and active attacks including the replay and man-in-the-middle attacks.

6. PERFORMANCE COMPARISON WITH EXISTING SCHEMES

This section provides the performance comparison of our scheme with the relevant existing access control schemes, such as the schemes by Mahmud *et al.* [23], Wang *et al.* [24] and Le *et al.* [25].

6.1. Comparison of computational costs

We have used the notations for comparison of computational cost between our scheme and other schemes provided in Table V. Let t_{ecm}, t_{eca}, t_i, t_{add}, t_{mul}, t_h, t_{enc}, t_{dec}, t_{ecenc}, t_{ecdec}, t_{mac}, t_{siggen}, and t_{sigver} denote the time taken for performing one ECC point multiplication, ECC point addition, modular inverse, modular addition, modular multiplication over finite field $GF(2^{163})$, hashing operation $h(\cdot)$, AES encryption, AES decryption, ECC encryption, ECC decryption over finite field $GF(2^{163})$, MAC operation, ECC signature generation, and ECC signature verification over finite field $GF(2^{163})$, respectively. For simplicity, we have considered the time taken for one MAC operation as that for one hashing operation. The quantitative analysis of [44] shows that approximately 1200 field multiplications are required for the computation of an ECC-point multiplication; an elliptic curve point addition is equivalent to one field inversion and two-field multiplications; approximately three-field multiplications are necessary for the computation of a field inversion; the computation of elliptic curve encryption and decryption require approximately 2405 and 1205 field multiplications, respectively [45,46]; and the cost of field addition is negligible. Furthermore, a 1024-bit modular multiplication takes 41 times longer than a field multiplication in finite field $GF(2^{163})$. The results of Wong *et al.* [47] show the speed for AES encryption and decryption, hash function using SHA-1 and 1024-bit modular multiplication. In Table IV, the time complexity of various operations in terms of t_{mul} are listed according to the analysis results reported in [48].

Table V shows the computational complexity comparison among existing schemes [23–25], and our scheme during the login and authentication phases. In this table, we have shown both formulated results and rough quantitative analysis. It is clear that compared with the other existing schemes [23,25], the computational cost of our scheme is much less. Our scheme requires little more computational overhead as compared to Wang *et al.*'s [24]. However, in the scheme by Wang *et al.*, although the user authenticates a sensor node, that sensor node does not authenticate the user, and as a result, the mutual authentication is not provided between the user and the senor node in their scheme. In our scheme, the resource-constrained sensor nodes do not require any computational overhead, and only resource-rich cluster heads need to perform computation during the authentication phase. Hence, our scheme is more suitable for the resource-constrained sensor nodes as compared with other schemes.

Table VI. Communication cost comparison between our scheme and other schemes.

Scheme	I_1	I_2	I_3
Le *et al.* [25]	2208	7	7
Mahmud-Morogan [23]	1132	5	5
Wang *et al.* [24]	2544	6	6
Ours	2144	5	4

Note: I_1, total number of bits transmission required for all the phases; I_2, total number of packets transmission during all phases; I_3, total number of message transmission during all the phases.

6.2. Comparison of communication costs

Table VI provides the communication cost comparison among our scheme and the other related schemes [24,25]

Security Comm. Networks 2015; **8**:1752–1771 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

Table VII. Energy cost comparison between our scheme and other schemes required for a sensor node or cluster head during the login and authentication phases.

Scheme	Sensor node/cluster head's energy cost
Le *et al.* [25]	three MAC operations + one hash operation + three message transmissions
Mahmud-Morogan [23]	one ECC-point addition + three ECC-point multiplication + one hash operation + two MAC operations + three message transmissions
Wang *et al.* [24]	two hash operations + one ECC-signature generation + two ECC-signature verifications + two message transmissions
Ours	five hash operations + two symmetric-key encryption/decryptions + $(k+1)$ ECC-point additions ($k = \|I_i\|$) + one message transmission

and [23]. In this comparison, we have considered the total number of bits and the total number of packets required for transmissions during all phases. From this table, we see that our scheme requires four exchanged messages and among them a cluster head is only directly involved with two message exchanges. In addition, we have calculated the total number of bits required for all the messages during all phases for the access control schemes. We have calculated the number of packets required for transmission of a message for the CC2420 transceiver [30], which supports a packet of size 128 bytes, that is, 1024 bits. The results shown in Table VI clearly demonstrates that our scheme is efficient compared to other related schemes.

6.3. Comparison of energy costs

Because sensor nodes are resource-constrained, we need to consider the energy cost of a sensor node during the login and authentication phases. Table VII compares the energy cost of a sensor node or cluster head during the login and authentication phases among our scheme and other schemes, such as the schemes by Le *et al.* [25], Mahmud-Morogan [23], and Wang *et al.* [24]. As in [1,36], we have taken the energy cost of a sensor node or cluster head due to both computational and communication costs involved during the login and authentication phases. In wireless communication, the energy spent by sensor nodes or cluster heads are mainly due to transmissions and receptions of messages/packets rather than local computation. Note that our scheme does not require exchange of any message/packet during the login and authentication phases for a sensor node as compared to other schemes. This clearly indicates that the energy spent by a sensor node is significantly less as compared to other exiting schemes. Moreover, the energy cost of a cluster head in our scheme is also less due to application of efficient hash function, ECC and symmetric-key cryptosystem (AES). Because of resource-rich cluster heads, our scheme is efficient as compared with existing schemes.

7. CONCLUSION

In this paper, we have proposed a new user access control scheme for HWSNs. In our scheme, a user is authenticated by both the BS and the cluster heads inside HWSN under certain access privilege. After successful authentication, the cluster head sends the encrypted real-time data as per the user request using the attribute-based encryption method. Authenticated users can only retrieve the symmetric key, if he or she has the proper access privilege for decrypting the message send by the respective cluster head. Our scheme supports new node addition dynamically. For this purpose, the user's smart card is not required to updated any further stored information for accessing the real-time data from the newly joined cluster heads. Mahmud-Morogan's scheme is inefficient as the user registration and password change phases are not supported. In the scheme by Le *et al.*, the user is authenticated by the sensor node, but there is no provision for mutual authentication of the sensor node by the user. In the scheme by Wang *et al.*, although the user authenticates a sensor node, that sensor node does not authenticate the user. Thus, the mutual authentication is not provided between the user and the senor node in their scheme. Using the widely-accepted AVISPA tool, we have shown that our scheme is secure with respect to the passive attacks and the active attacks including the replay attack and the man-in-the-middle attack. Furthermore, through the rigorous informal and formal security analysis, our scheme is shown to be secure against various known attacks. Our scheme has the ability to change the password by a legal user freely without contacting the BS at any time. Our scheme also supports efficiently new node addition dynamically after the initial deployment of the nodes in the network as compared with other existing schemes. In addition, our scheme preserves the user anonymity property. Our scheme is also efficient in terms of communication and computational overheads. Overall, high security, low communication and computational costs, and extra important features supported by our scheme make our scheme much appropriate for practical applications in user access control of WSNs. In the future, we would like to validate our scheme for the energy, time, storage, and communication demand measurements using a real testbed simulation environment in WSN.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers and the Editor, Dr Xiangyun Zhou, which have improved the content and the presentation of this paper.

REFERENCES

1. Das AK, Sharma P, Chatterjee S, Sing JK. A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Journal of Network and Computer Applications* 2012; **35**(5): 1646–1656.
2. Cheng Y, Agrawal DP. An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Ad Hoc Networks* 2007; **5**(1): 35–48.
3. Rivest RL, Shamir A, Adleman LM. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 1978; **21**: 120–126.
4. Diffie W, Hellman ME. New directions in cryptography. *IEEE Transactions on Information Theory* 1976; **22**: 644–654.
5. Malan DJ, Welsh M, Smith MD. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography, *Proceedings of First IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, Santa Clara, California, USA, October 2004; 59–64.
6. Watro R, Kong D, Cuti S, Gardiner C, Lynn C, Kruus P. TinyPK: securing sensor networks with public key technology, *Proceedings of the 2nd ACM Workshop on Security of ad hoc and Sensor Networks (SASN'04)*, Washington, DC, USA, October 2004; 59–64.
7. Atmel Corporation. Available from: http://www.atmel.com. [Accessed on November 2010].
8. Gura N, Patel A, Wander A, Eberle H, Shantz SC. Comparing elliptic curve cryptography and RSA on 8-bit CPUs, *Proceedings of 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'04), LNCS 3156*, Cambridge (Boston), USA, 2004; 119–132.
9. Johnson D, Menezes A. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Technical Report CORR 99-34*, Dept. of C & O, University of Waterloo, Canada, August 23, 1999.
10. Liao HZ, Shen YY. On the elliptic curve digital signature algorithm. *Tunghai Science* 2006; **8**: 109–126.
11. Vanstone S. Responses to NIST's proposal. *Communications of the ACM* 1992; **35**: 50–52.
12. Carman DW, Kruus PS, Matt BJ. Constraints and approaches for distributed sensor network security. *NAI Labs Technical Report No. 00-010*, dated September 1, 2000.
13. Sahai A, Waters B. Fuzzy identity-based encryption. *Advances in Cryptology - EUROCRYPT, LNCS* 2005; **3494**: 457–473.
14. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data, *ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, 2006; 89–98.
15. Yu S, Ren K, Lou W. FDAC: Toward fine-grained distributed data access control in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(4): 673–686.
16. Ruj S, Nayak A, Stojmenovic I. Distributed fine-grained access control in wireless sensor networks, *IEEE International Parallel and Distributed Processing Symposium*, Anchorage (Alaska) USA, 2011; 352–362.
17. Dolev D, Yao A. On the security of public key protocols. *IEEE Transactions on Information Theory* 1983; **29**(2): 198–208.
18. Das ML. Two-factor user authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications* 2009; **8**(3): 1086–1090.
19. Wang H, Sheng B, Tan CC, Li Q. Comparing symmetric-key and public-key based security schemes in sensor networks: a case study of user access control, *Proceedings of 28th International Conference on Distributed Computing Systems*, Beijing, China, 2008; 11–18.
20. He D, Bu J, Zhu S, Chan S, Chen C. Distributed access control with privacy support in wireless sensor networks. *IEEE Transactions on Wireless Communications* 2011; **10**: 3473–3481.
21. Wen M, Lei J, Li J, Wang Y, Chen K. Efficient user access control mechanism for wireless multimedia sensor networks. *Journal of Computational Information Systems* 2011; **7**(9): 3325–3332.
22. Li M, Lou W, Ren K. Data security and privacy in wireless body area networks. *IEEE Wireless Communications* 2010: 51–58.
23. Mahmud AAl, Morogan MC. Identity-based authentication and access control in wireless sensor networks. *International Journal of Computer Applications* 2012; **41**(13): 18–24.
24. Wang H, Sheng B, Li Q. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 127–137.

25. Le H, Lee S, Butun I, Khalid M, Sankar R, Kim M, Han M, Lee Y-K, Lee H. An energy-efficient access control scheme for wireless sensor networks based on elliptic curve cryptography. *Journal of Communications and Networks* 2009; **11**(6): 599–606.
26. Das AK. An efficient random key distribution scheme for large-scale distributed sensor networks. *Security and Communication Networks* 2011; **4**(2): 162–180.
27. Das AK. An unconditionally secure key management scheme for large scale heterogeneous wireless sensor networks, *First IEEE International Conference on Communication Systems and Networks (COMSNETS 2009)*, Bangalore, India, 2009; 1–10.
28. Das AK, Sengupta I. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials, *3rd IEEE International Conference on Communication Systems Software and Middleware (COMSWARE 2008)*, Bangalore, India, 2008; 9–16.
29. Crossbow Technology Inc. Wireless Sensor networks. http://www.xbow.com. [Accessed on September 2011].
30. *rmCC2420 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver*. Available from: http://www.ti.com/product/cc2420. [Accessed on September 2011].
31. Advanced Encryption Standard (AES). FIPS PUB197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf [Accessed on January 2013].
32. Odelu V, Das AK, Goswami A. A secure effective key management scheme for dynamic access control in a large leaf class hierarchy. *Information Sciences* 2014; **269**(C): 270–285.
33. Das AK, Paul NR, Tripathy L. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences* 2012; **209**(C): 80–92.
34. Sarkar P. A simple and generic construction of authenticated encryption with associated data. *ACM Transactions on Information and System Security* 2010; **13**(33): 1–16.
35. Stinson DR. Some observations on the theory of cryptographic hash functions. *Designs, Codes and Cryptography* 2006; **38**(2): 259–277.
36. Chatterjee S, Das AK, Sing JK. An enhanced access control scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks* 2014; **21**(1-2): 121–149.
37. Das AK. A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. *Networking Science* 2013; **2**(1-2): 12–27.
38. Das AK, Chatterjee S, Sing JK. A novel efficient access control scheme for large-scale distributed wireless sensor networks. *International Journal of Foundations of Computer Science* 2013; **24**(5): 625–653.
39. Das AK, Massand A, Patil S. A novel proxy signature scheme based on user hierarchical access control policy. *Journal of King Saud University - Computer and Information Sciences* 2013; **25**(2): 219–228.
40. Avispa Tool Documentation. Automated validation of internet security protocols and applications. http://www.avispa-project.org/ package/ usermanual.pdf. [Accessed on March 2013].
41. Armando A, Basin D, Boichut Y, *et al.* The avispa tool for the automated validation of internet security protocols and applications, *17th International Conference on Computer Aided Verification (CAV'05), LNCS 3576*, Scotland, UK, 2005; 281–285.
42. von Oheimb D. The high-level protocol specification language hlpsl developed in the eu project avispa, *Proceedings of APPSEM 2005 Workshop*, Tallinn, 2005; 1–17.
43. Basin D, Modersheim S, Vigano L. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security* 2005; **4** (3): 181–208.
44. Kim H-S, Lee S-W. Enhanced novel access control protocol over wireless sensor networks. *IEEE Transactions on Consumer Electronics* 2009; **55** (2): 492–498.
45. DeWin E, Bosselaers A, Vandenberghe S, De Gersem P, Vandewalle J. A fast software implementation for arithmetic operations in $GF(2^n)$. In *Proceedings of Advances in Cryptology - ASIACRYPT '96*, vol. 1163, Lecture Notes in Computer Science. Springer-Verlag: Kyongju, Korea, 1996; 65–76.
46. Schroeppel R, Orman H, O'Malley S, Spatscheck O. Fast key exchange with elliptic curve systems, *Proceedings of Advances in Cryptology - CRYPTO '95*, Santa Barbara, California, USA, 1995; 43–56.
47. Wong DS, Fuentes HH, Chan AH. The performance measurement of cryptographic primitives on palm devices, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, New Orleans, Louisiana, USA, 2001; 92–101.
48. Wu S, Chen K. An efficient key-management scheme for hierarchical access control in E-medicine system. *Journal of Medical Systems* 2012; **36**(4): 2325–2337.