Wireless Pers Commun (2015) 82:1377–1404
DOI 10.1007/s11277-015-2288-3

A Secure and Efficient User Anonymity-Preserving Three-Factor Authentication Protocol for Large-Scale Distributed Wireless Sensor Networks

Ashok Kumar Das

Published online: 20 January 2015

Abstract Critical applications in wireless sensor network (WSN) are real-time based applications. Therefore, users are generally interested in accessing real-time information. This is possible, if the users (called the external parties) are allowed to access the real-time data directly from the sensor nodes inside WSN and not from the base station. The sensory information from nodes are gathered periodically by the base station and so, the gathered information may not be real-time. In order to get the real-time information from the sensor nodes, the user needs to be first authorized to the sensor nodes as well as the base station so that the illegal access to nodes do not happen. In this paper, we propose a novel three-factor user authentication scheme suited for distributed WSNs. Our scheme is light-weight, because it only requires the efficient cryptographic hash function, and symmetric key encryption and decryption operations. Further, our scheme is secure against different known attacks which are proved through the rigorous informal and formal security analysis. In addition, we simulate our scheme for the formal security verification using Automated Validation of Internet Security Protocols and Applications tool. The simulation results clearly demonstrate that our scheme is secure against passive and active adversaries.

Keywords Wireless sensor networks · User authentication · Hash function · Password · Biometrics · Smart cards · User anonymity · AVISPA · Security

1 Introduction

In a *distributed wireless sensor network* (*DWSN*) shown in Fig. 1, there is no fixed infrastructure and network topology is not known prior to deployment of the sensor nodes in a target field. Sensor nodes are usually deployed all over the target area randomly. After deployment sensor nodes form an infrastructure-less multi-hop wireless communication between them

A. K. 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

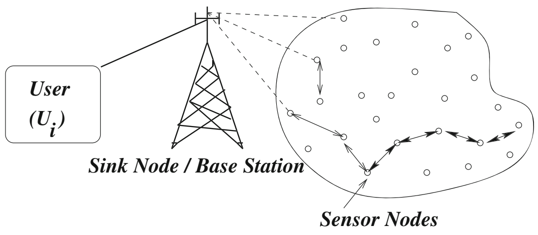

Fig. 1 A distributed wireless sensor network (DWSN) architecture

and data are routed back to the base station (BS). A base station collects sensor readings, performs costly operations on behalf of sensor nodes and manages the network. The base station is assumed to be trustworthy for WSN applications. The transmission between the sensors is done by short range radio communications. The base station is assumed to be computationally well-equipped whereas the sensor nodes are resource-starved.

Usually, the queries in WSN applications are issued at the point of the base station (BS) of the network. However, for critical applications of WSNs, such as military and healthcare applications, there is always a great need to access the real-time information from the sensors directly inside WSN, because the real-time data from the sensor nodes may no longer be accessed through the BS only. Thus, it is expected that the real-time data can be given access directly to the external parties (users) those who are authorized to access data as and when they demand. As a result, user authentication becomes a very important research area in WSN for providing access to real-time data inside WSN as and when an authorized user demands.

Recently, biometric-based user authentication schemes along with passwords have drawn considerable attention in research [13,24,25,40]. Biometric-based user authentication in WSN becomes inherently more reliable and secure than usual traditional password-based user authentication schemes [24]. We have several advantages of using biometric keys (for example, fingerprints, faces, irises, hand geometry and palm-prints, etc.) over traditional passwords, which are listed below [24]:

– Biometric keys can not be lost or forgotten.
– Biometric keys are very difficult to copy or share.
– Biometric keys are extremely hard to forge or distribute.
– Biometric keys can not be guessed easily.
– Someone's biometrics is not easy to break than others.

In this paper, we aim to propose a new three-factor user authentication scheme for DWSN. We show that our scheme is efficient and secure as compared to other existing schemes in WSNs.

According to Tan [34], we also extend the security requirements of two-factor authentication schemes to three-factor authentication schemes in WSNs, which are given below:

– *Mutual authentication.* After run of the protocol, a sensor node should believe that the user is a legitimate registered client. The user also believes that the communicating party is the sensor node which the user intended to login to.
– *BS not knowing password and biometric.* The BS should not have any information about the registered user's password and personal biometric. This requirement is extremely required because several users may apply the same password to access different servers in the real-life applications. As a result, if a privileged insider of the registration center (in our scheme, it is the BS) knows the password or biometrics of a user, he/she may impersonate that user for accessing the services from other servers.

- *Freedom of password and biometric update.* A user should be allowed to change/ update freely and locally his/her password as well as biometric template without contacting the BS. The BS must be totally unaware of the change of the user's password and biometric template.
- *Three-factor security.* In the security model for three-factor authentication scheme, an adversary can have full control over the communication channel between the users, the sensor nodes and the BS during the login phase, and the authentication and key agreement phase. In the three-factor security adversary model, the adversary is modeled to have at most two of the following three abilities, but it is not allowed to have all the three abilities. The adversary can extract the information from the lost or stolen smart card, obtain the password, or access the biometric template of a user.

1.1 Threat Model

We consider the threat model for designing our scheme similar to that in [13]. This model assumes the following:

- The base station (the GW-node) is assumed to be trustworthy and it will never be compromised by an adversary (attacker).
- An adversary can eavesdrop on all traffic, inject packets and reply old messages previously delivered.
- Sensors are not equipped with tamper-resistant hardware due to cost constraints. As a result, if an adversary physically captures a sensor from the target field, the adversary will know all the keying materials stored in that sensor's memory.
- The smart card of a user is not tamper-resistant. Thus, if the smart card of a legal user is lost or stolen by an adversary, he/she will know all the sensitive information stored in that smart card.
- As in [13], we also use the well-known Dolev–Yao threat model in which any two parties communicate over an insecure (public) channel. We adopt the similar threat model in DWSN, where the channel is insecure and also the end-points (users and sensors) cannot in general be trustworthy.

1.2 Our Contributions

Our contributions are listed below:

- We propose a novel secure and efficient user authentication scheme suited for DWSN.
- Our scheme makes use of the user's password and biometric information along with the non-tamper resistant smart card.
- Our scheme allows the user to change/update his/her password and personal biometric locally without further contacting the BS.
- Our scheme allows efficiently new node addition after initial deployment of the nodes in the target field.
- In our scheme, each deployed sensor node in the target field needs to store only one master key prior to its deployment, which enables our scheme for the minimum storage overhead for the resource-constrained sensor nodes.
- Our scheme is shown to be secure against various known attacks through both informal and formal security analysis.

- Our scheme is also secure against passive and active adversaries, which is shown through the simulation results using the widely-accepted Automated Validation of Internet Security Protocols and Applications (AVISPAs) tool.
- Finally, our scheme is efficient as compared to other existing schemes in the literature.

1.3 Roadmap of the Paper

The rest of the paper is organized as follows. We discuss some basic mathematical preliminaries needed for describing and analyzing our scheme in Sect. 2. In Sect. 3, we discuss the existing works on user authentication for WSNs in the literature. In Sect. 4, we propose a novel scheme for three-factor user authentication in DWSN. In Sect. 5, through the rigorous informal and formal security analysis we show that our scheme has the ability to defend various known attacks. Further, we simulate our scheme for formal security verification using the widely-accepted AVISPA tool in Sect. 6. In Sect. 7, we compare the performance of our scheme with other related existing approaches. Finally, we conclude the paper in Sect. 8.

2 Mathematical Background

In this section, we briefly discuss the properties of a one-way hash function and fuzzy extractor.

2.1 One-Way Hash Function

We define the formal definition of a one-way collision-resistant hash function as follows [14,30,33].

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

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

where $Pr[E]$ denotes the probability of a random event E, and $(x, x') \Leftarrow_R \mathcal{A}$ 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. We call the hash function $h(\cdot)$ is collision-resistant, if $Adv_{\mathcal{A}}^{HASH}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$.

2.2 Fuzzy Extractor

We briefly describe the extraction process of data key from the given biometric of a user using a fuzzy extractor method. It is known that the output of a conventional hash function $h(\cdot)$ is sensitive and it may also return completely different outputs even if there is a little variation in inputs [8]. The biometric information is thus prone to various noises during data acquisition, and as a result, the reproduction of actual biometric is hard in common practice. In order to avoid such problem, a fuzzy extractor [4,19,22] is used in the literature, which has the ability to extract a uniformly random string σ and a public information τ from a given biometric template B with the error tolerance t. In the reproduction process, the fuzzy extractor recovers the original biometric data key σ for a noisy biometric B' using the public parameters τ and t. Let $\mathcal{M} = \{0, 1\}^v$ be a finite v-dimensional metric space of biometric data points, $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{Z}^+$ a distance function, which can be used to calculate the distance between two points based on the metric chosen, l the number of bits of the output string b_i, and t the error tolerance, where $\mathbb{Z}^+$ represents the set of all positive integers.

Definition 2 The fuzzy extractor is a tuple $(\mathcal{M}, l, t)$, which is defined by the following two algorithms, called Gen and Rep:

– Gen: It is a probabilistic algorithm, which takes a biometric information $B_i \in \mathcal{M}$ as input and outputs a secret data key $\sigma_i \in \{0, 1\}^l$ and a public reproduction parameter τ_i. In other words, $Gen(B_i) = \{\sigma_i, \tau_i\}$.
– Rep: This deterministic algorithm takes a noisy biometric information $B_i' \in \mathcal{M}$ and a public parameter τ_i related to B_i, and then it reproduces the biometric secret data key σ_i. In other words, $Rep(B_i', \tau_i) = \sigma_i$ provided that $d(B_i, B_i') \leq t$.

A detailed description of the fuzzy extractor and the extraction procedure can be found in [4,19].

3 Related Work

Watro et al. [38] proposed a user authentication in WSNs, which is known as TinyPK. TinyPK uses the RSA algorithm [29] and Diffie-Hellman protocol [18]. As pointed out in [17], TinyPK has a security flaw as follows. On receiving the user's public key, an attacker can encrypt a session key along with other parameters and then send the encrypted message to the user. Upon receiving the encrypted message, the user is forced to believe that the message has come from the legitimate sensor node. The user then decrypts the receiving encrypted message using his/her private key and also uses the session key for subsequent operations the attacker intends to perform. Wong et al. [39] proposed an efficient password-based user authentication scheme. However, their scheme is vulnerable to many logged in users with the same login-id and stolen-verifier attacks. M. L. Das's scheme [17] eliminates the flaws of Wong et al.'s scheme. M. L. Das's scheme is based on passwords. However, that scheme cannot also resist denial-of-service attack and node compromise attack. Some improvements on M. L. Das's scheme [17] have been further proposed in [23,26] in order to withstand security flaws found in that scheme.

He et al. [21] later proposed an enhancement on M. L. Das's scheme [17]. Vaidya et al. [35] showed that M. L. Das's scheme [17] and Khan–Alghathbar's scheme [23] have security flaws and they are also vulnerable to various known attacks including stolen smart card attacks. To remedy such security weaknesses found in both schemes [17] and [23], Vaidya et al. suggested an improved two-factor user authentication in WSNs. Fan et al. [20] proposed a simple efficient and Denial-of-Service (DoS) resistant user authentication scheme for two-tiered WSNs. Chen and Shih [6] also pointed out that M. L. Das's scheme [17] fails to achieve mutual authentication. To overcome such problem, they also proposed a robust mutual authentication protocol for WSNs. Das et al. [16] proposed a novel and efficient password-based user authentication scheme for the hierarchical wireless sensor networks. Their scheme was shown to be secure against various known attacks including the replay and man-in-the-middle attacks with the help of formal security verification [12]. Further, an improved version of Das et al.'s scheme [16] has been proposed in [37] in the literature.

Recently, biometric-based user authentication in WSNs has drawn a considerable research attention. Thus, the biometric-based user authentication in WSN becomes inherently more reliable and secure than usual traditional password-based user authentication schemes. Yuan et al.'s scheme [40] provides better security as compared to that for M. L. Das's scheme [17] because the former scheme uses biometrics verification along with the password verification of the user. Yuan et al.'s scheme [40] has same drawbacks as in M. L. Das's scheme [17], which does not resist denial-of-service attack and node compromise attack.

4 The Proposed scheme

In this section, we introduce a novel user authentication scheme suited for distributed wireless sensor networks. For describing and analyzing our scheme, we use the notations listed in Table 1.

The various phases of our scheme are described in detail in the following subsections.

4.1 Pre-deployment Phase

The purpose of this phase is to pre-load the keying materials to all sensor nodes prior to their deployment in the target field. This procedure is performed by the (key) setup server (in our scheme, it is the BS) in offline. The following steps are executed in this phase:

Step PR1: The BS first assigns a unique identity ID_{SN_j} for each deployed sensor node SN_j in the network.

Table 1 Notations used in this paper

Symbol	Description
U_i	ith user
ID_i	Identity of user U_i
PW_i	Password of user U_i
B_i	Biometric information of U_i
K	1,024-bit secret number only known to U_i
BS	Base station (GW-node)
SN_j	jth Sensor node in DWSN
ID_{SN_j}	Short identity of sensor SN_j
MK_{SN_j}	Master key of sensor SN_j
$h(\cdot)$	Secure collision-free one-way hash function
X_s	1,024-bit secret master key of BS
$E_k(M)$	Symmetric encryption of M using the key k
$D_k(M)$	Symmetric decryption of M using the key k
RN_X	A random nonce generated by the entity X
	(Nonce is a one-time random bit-string, usually used to achieve freshness)
T_i	Current system timestamp
ΔT	Interval of the expected time for the transmission delay in DWSN
$X \oplus Y$	Bitwise XORed of data X with data Y
$X\|\|Y$	Data X concatenates with data Y

Step PR2: For each deployed sensor node SN_j, the BS randomly generates a unique master key MK_{SN_j}, which is also shared with the BS.
Step PR3: Finally, the BS loads the following information into the memory of each deployed sensor node SN_j: (i) its own identity, ID_{SN_j} and (ii) its own master key MK_{SN_j}.

Note that the BS knows the identities and master keys of all deployed sensor nodes. The BS is considered as trustworthy and thus, the BS will not reveal the keys to any adversary.

4.2 Post-deployment Phase

As soon as the sensor nodes SN_j are loaded with their keying materials in Sect. 4.1, they can be deployed in the target field. As describe in [9], after deployment each sensor node SN_j can broadcast a HELLO message containing its own identifier ID_{SN_j} to the nodes in its communication range. Every node also receives HELLO message from its neighbor nodes. If d is the average number of neighbor nodes of each sensor node SN_j, it then prepares a list of its all d neighbor nodes as $NL_{SN_j} = \{SN_{j_1}, SN_{j_2}, \ldots, SN_{j_d}\}$. Once each sensor node SN_j finds its neighbors, they can communication each other with their neighbors and then to the BS via a multi-hop wireless communication path.

4.3 Registration Phase

In this phase, a legal user U_i registers with the BS for accessing the real-time data from a particular sensor node SN_j inside DWSN. It consists of the following steps:

Step R1: U_i first inputs his/her chosen identity ID_i, password PW_i, and then imprints the biometric B_i at the sensor of a specific device. U_i then generates a 1,024-bit random number K, which is kept secret to U_i only. U_i uses the fuzzy extractor function $Gen(\cdot)$ on input B_i to produce the biometric data key σ_i and the public parameter τ_i as $Gen(B_i) = (\sigma_i, \tau_i)$, where σ_i is kept secret to U_i only.
Step R2: U_i computes the masked password RPW_i as $RPW_i = h(ID_i||K||PW_i)$ and sends the registration request $\langle ID_i, RPW_i\rangle$ to the BS via a secure channel.
Step R3: After receiving the request, the BS generates a 1,024-bit secret number X_s, which is only known to the BS. The BS computes $r_i = h(ID_i||X_s)$, and issues a smart card, say SC_i to the user U_i containing the information $\{r_i, h(\cdot)\}$ via a secure channel.
Step R4: After receiving the smart card SC_i from the BS, the user U_i computes

$$\begin{aligned} e_i &= h(ID_i||\sigma_i) \oplus K, \\ f_i &= h(ID_i||RPW_i||\sigma_i), \text{ and} \\ r_i^* &= r_i \oplus h(ID_i||K) \\ &= h(ID_i||X_s) \oplus h(ID_i||K). \end{aligned}$$

Finally, U_i replaces r_i with r_i^* in the smart card SC_i and also stores the information $\{e_i, f_i, \tau_i, Gen(\cdot), Rep(\cdot), t\}$ in the memory of SC_i.

The summary of the registration phase of our scheme is given in Table 2.

4.4 Login Phase

In this phase, if the user U_i wants to access the real-time data from the sensor nodes inside DWSN, the following steps need to be executed:

Table 2 Registration phase of our scheme

U_i	BS
Inputs ID_i, PW_i, B_i	
Computes $(\sigma_i, \tau_i) = Gen(B_i)$	
Generates 1,024-bit number K	
Computes $RPW_i = h(ID_i \| K \| PW_i)$	
$\xrightarrow{\langle ID_i, RPW_i \rangle}$ (via a secure channel)	
	Generates 1,024-bit secret X_s
	Computes $r_i = h(ID_i \| X_s)$
	$\xleftarrow{\langle \text{Smart Card}(r_i, h(\cdot)) \rangle}$ (via a secure channel)
Computes $e_i = h(ID_i \| \sigma_i) \oplus K$, $f_i = h(ID_i \| RPW_i \| \sigma_i)$, $r_i^* = r_i \oplus h(ID_i \| K)$	
Replaces r_i with r_i^* in smart card	
Stores $\{e_i, f_i, \tau_i, Gen(\cdot), Rep(\cdot), t\}$ in smart card	

Step L1: The user U_i first inserts his/her smart card SC_i into the card reader of a specific terminal, and then inputs identity ID_i, password PW_i, and also imprints the biometric B_i^* at the sensor.

Step L2: SC_i computes

$$\begin{aligned} \sigma_i^* &= Rep(B_i^*, \tau_i), \\ K^* &= h(ID_i || \sigma_i^*) \oplus e_i, \\ RPW_i^* &= h(ID_i || K^* || PW_i), \\ f_i^* &= h(ID_i || RPW_i^* || \sigma_i^*). \end{aligned}$$

SC_i then checks the condition whether $f_i^* = f_i$. If it holds, it ensures that the user U_i passes successfully both password and biometric verification. Otherwise, this phase is terminated immediately.

Step L3: SC_i further computes $M_1 = r_i^* \oplus h(ID_i || K^*) = h(ID_i || X_s)$ and generates a random nonce RN_{U_i}. U_i selects a sensor-login node, say SN_j from which he/she wants to access the real-time data. SC_i then computes

$$\begin{aligned} M_2 &= M_1 \oplus RN_{U_i} \\ &= h(ID_i || X_s) \oplus RN_{U_i}, \\ M_3 &= h(ID_i || ID_{SN_j} || M_1 || RN_{U_i} || T_1), \end{aligned}$$

where T_1 is the current timestamp of SC_i's system. SC_i finally sends the login request $\langle ID_{SN_j}, M_2, M_3, T_1 \rangle$ to the BS via a public channel.

The login phase of our scheme is summarized in Table 3.

4.5 Authentication and Key Agreement Phase

In this phase, the BS authenticates the user U_i and also a sensor node SN_j authenticates U_i before U_i is given access to the real-time data from SN_j inside DWSN. At the end of successful

Table 3 Login phase of our scheme

U_i/SC_i	BS
Inserts the smart card SC_i and inputs ID_i, PW_i, and B_i^*	
Computes $\sigma_i^* = Rep(B_i^*, \tau_i)$, $K^* = h(ID_i\|\|\sigma_i^*) \oplus e_i$, $RPW_i^* = h(ID_i\|\|K^*\|\|PW_i)$, $f_i^* = h(ID_i\|\|RPW_i^*\|\|\sigma_i^*)$	
Checks if $f_i^* = f_i$? If so, computes $M_1 = r_i^* \oplus h(ID_i\|\|K^*)$	
Generates a random nonce RN_{U_i} and computes $M_2 = M_1 \oplus RN_{U_i}$ and $M_3 = h(ID_i\|\|ID_{SN_j}\|\|M_1\|\|RN_{U_i}\|\|T_1)$	
$\xrightarrow{\langle ID_{SN_j}, M_2, M_3, T_1\rangle}$ (via a public channel)	

mutual authentication, both the user U_i and the sensor node SN_j establish a common session key between them. After receiving the login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ from the user U_i by the BS via a public channel, the following steps are executed:

Step A1: The BS first checks the validity of T_1 in the message as follows. If the login request is received at time T_2, the BS checks the condition $|T_1 - T_2| < \Delta T$, where ΔT is the interval of the expected time for the transmission delay in DWSN. If it holds, the BS further checks ID_{SN_j} of the sensor node SN_j from which the user U_i is looking for accessing the real-time data. If it is valid, the BS computes

$$\begin{aligned} M_4 &= h(ID_i||X_s), \\ M_5 &= M_2 \oplus M_4 \\ &= h(ID_i||X_s) \oplus RN_{U_i} \oplus h(ID_i||X_s) \\ &= RN_{U_i}, \\ M_6 &= h(ID_i||ID_{SN_j}||M_4||M_5||T_1). \end{aligned}$$

The BS then checks the condition $M_6 = M_3$. If it holds, the BS authenticates the user U_i as a valid user and proceeds to next step. Otherwise, this phase is terminated immediately.
Step A2: The BS computes $M_7 = E_{MK_{SN_j}}[ID_i, ID_{SN_j}, M_5, h(M_4), T_1, T_3]$ using the master key MK_{SN_j} of the sensor-login node SN_j, and T_3 is the current timestamp of the BS. Finally, the BS sends the authentication request $\langle ID_{SN_j}, M_7\rangle$ to the sensor-login node SN_j via a public channel.
Step A3: Suppose the authentication request $\langle ID_{SN_j}, M_7\rangle$ is received by the sensor SN_j at time T_4. SN_j decrypts M_7 using its own master key MK_{SN_j} in order to retrieve the information ID_i, ID_{SN_j}, M_5, $h(M_4)$, T_1, and T_3. SN_j then checks the validity of ID_{SN_j}, and also checks the condition $|T_3 - T_4| < \Delta T$. If these conditions hold, the message is treated as a valid message and U_i is considered as a valid user by SN_j. Otherwise, this phase is terminated immediately.
Step A4: SN_j generates a random nonce RN_{SN_j}. Further, SN_j computes the session key SK_{ij} shared between the user U_i and the sensor node SN_j as $SK_{ij} = h(ID_i||ID_{SN_j}||h(M_4)||M_5||RN_{SN_j}||T_1||T_5)$, where T_5 is the current timestamp of the sen-

sor SN_j. SN_j also computes

$$\begin{aligned} M_8 &= h(SK_{ij}), \\ M_9 &= M_5 \oplus RN_{SN_j} \oplus ID_i \\ &= RN_{U_i} \oplus RN_{SN_j} \oplus ID_i. \end{aligned}$$

SN_j then sends the authentication reply $\langle M_8, M_9, T_5 \rangle$ to the user U_i via a public channel.
Step A5: After receiving the authentication reply from SN_j in Step A4, the smart card SC_i of the user U_i first checks the validity of the timestamp T_5 attached with the message by the condition $|T_5 - T_6| < \Delta T$, where T_6 is the time when the message is received by SC_i. If it is valid, SC_i then computes

$$\begin{aligned} M_{10} &= M_9 \oplus RN_{U_i} \oplus ID_i \\ &= RN_{SN_j}, \\ SK'_{ij} &= h(ID_i||ID_{SN_j}||h(M_1)||RN_{U_i}||M_{10}||T_1||T_5), \\ M_{11} &= h(SK'_{ij}). \end{aligned}$$

SC_i checks the condition $M_{11} = M_8$. If this verification passes, U_i treats the sensor node SN_j as a legitimate sensor node. Otherwise, this phase is terminated immediately. Note that $SK_{ij} = SK'_{ij}$. Finally, SN_j stores SK_{ij} and $SC_i(U_i)$ stores SK'_{ij} as the secret session key shared between them for their future secure communication.

This phase is summarized in Table 4.

4.6 Password and Biometric Update Phase

In this phase, we describe the procedure of updating old password and biometric by a legal user U_i locally without contacting the BS. This phase involves the following steps:

Step PB1: U_i first inserts his/her smart card SC_i into the card reader of a specific terminal, and then inputs his/her identity ID_i, old password PW_i^{old} and also imprints the biometric B_i^{old} at the sensor. SC_i then computes

$$\begin{aligned} \sigma_i^{old} &= Rep(B_i^{old}, \tau_i), \\ K^* &= h(ID_i||\sigma_i^{old}) \oplus e_i, \\ RPW_i^{old} &= h(ID_i||K^*||PW_i^{old}), \\ f_i^{old} &= h(ID_i||RPW_i^{old}||\sigma_i^{old}). \end{aligned}$$

If the condition $f_i^{old} = f_i$ is met, SC_i then asks the user U_i to input his/her chosen new password and biometric. Otherwise, this phase is terminated immediately.
Step PB2: U_i enters his/her new password PW_i^{new} and imprints the new biometric B_i^{new} at the sensor. SC_i then computes

$$\begin{aligned} (\sigma_i^{new}, \tau_i^{new}) &= Gen(B_i^{new}), \\ e_i^{new} &= h(ID_i||\sigma_i^{new}) \oplus h(ID_i||K^*), \\ RPW_i^{new} &= h(ID_i||K^*||PW_i^{new}), \\ f_i^{new} &= h(ID_i||RPW_i^{new}||\sigma_i^{new}). \end{aligned}$$

Finally, SC_i updates e_i, f_i, and τ_i with e_i^{new}, f_i^{new}, and τ_i^{new}, respectively, in its memory.

Table 4 Authentication and key agreement phase of our scheme

BS	SN_j
Checks the validity of T_1 and ID_{SN_j} If they are valid, computes $M_4 = h(ID_i\|\|X_s)$, $M_5 = M_2 \oplus M_4$, $M_6 = h(ID_i\|\|ID_{SN_j}\|\|M_4\|\|M_5\|\|T_1)$ Checks if $M_6 = M_3$? If so, computes $M_7 = E_{MK_{SN_j}}[ID_i, ID_{SN_j}, M_5, h(M_4), T_1, T_3]$ $\xrightarrow{\langle ID_{SN_j}, M_7\rangle}$ (via a public channel)	
U_i/SC_i	SN_j
	Decrypts M_7 using its master key MK_{SN_j} to retrieve $ID_i, ID_{SN_j}, M_5, h(M_4), T_1, T_3$ Checks the validity of ID_{SN_j} and T_3 If they are valid, computes $SK_{ij} = h(ID_i\|\|ID_{SN_j}\|\|h(M_4)\|\|M_5\|\|RN_{SN_j}\|\|T_1\|\|T_5)$, $M_8 = h(SK_{ij})$, $M_9 = M_5 \oplus RN_{SN_j}\|\|ID_i$ $\xleftarrow{\langle M_8, M_9, T_5\rangle}$ (via a public channel)
Checks the validity of T_5. If so, computes $M_{10} = M_9 \oplus RN_{U_i} \oplus ID_i$, $SK'_{ij} = h(ID_i\|\|ID_{SN_j}\|\|h(M_1)\|\|RN_{U_i}\|\|M_{10}\|\|T_1\|\|T_5)$, $M_{11} = h(SK'_{ij})$ Checks if $M_{11} = M_8$? If so, U_i ensures that SN_j is valid Stores SK_{ij} as session key	Stores SK'_{ij} as session key

4.7 Dynamic Node Addition Phase

Sometimes some deployed sensor nodes become faulty because these are power exhausted or compromised by an attacker. In either case, there is a need to deploy some fresh sensor nodes to the target field in order to continue the services in DWSN. This phase is very simple and executed in offline by the BS. For a new deployed sensor node SN_j^{new} in the target field, the BS assigns a unique identity $ID_{SN_j^{new}}$ and also generates a unique master key $MK_{SN_j^{new}}$ randomly, which is different from all the master keys of deployed sensor nodes. Finally, the BS pre-loads $ID_{SN_j^{new}}$ and $MK_{SN_j^{new}}$ in the memory of SN_j^{new} prior to its deployment in the target field. The BS also needs to inform the user U_i about the addition of the new sensor node SN_j^{new} in the network so that U_i can later access the real-time data from that node.

5 Security Analysis of the Proposed Scheme

In this section, through both informal and formal security analysis, we show that our scheme protects various known attacks.

5.1 Informal Security Analysis

The following subsections show that our scheme has the ability to tolerate various known attacks.

5.1.1 *Stolen Smart Card Attack*

Suppose the smart card SC_i of a legal user U_i is lost or stolen. Then an adversary can easily extract all the sensitive information e_i, f_i, and r_i^* from the lost/stolen smart card SC_i, where $e_i = h(ID_i||\sigma_i) \oplus K$, $f_i = h(ID_i||RPW_i||\sigma_i)$, $r_i^* = r_i \oplus K\ h(ID_i||X_s) \oplus h(ID_i||K)$, K being a 1,024-bit secret number known to the user U_i only, since the smart card in our scheme is not tamper-resistant. Note that ID_i, K, PW_i and σ_i are unknown to the adversary. Due to collision-resistant property of the one-way hash function $h(\cdot)$, it is computationally infeasible to derive the password PW_i and the biometric secret data key σ_i (consequently, the biometric B_i) of the user U_i, and also the secret information X_s of the BS. Thus, our scheme has the ability to prevent the stolen smart card attack.

5.1.2 *Password and Biometric Change Attack*

Assume that an adversary acquires the smart card SC_i of a legal user U_i. Note that to update the new password and biometric, the adversary needs to know ID_i and K. Without these information, it is computationally infeasible task to compute $e_i^{new} = h(ID_i||\sigma_i^{new}) \oplus K$, $RPW_i^{new} = h(ID_i||K||PW_i^{new})$, and $f_i^{new} = h(ID_i||\ RPW_i^{new}||\sigma_i^{new})$, where $(\sigma_i^{new}, \tau_i^{new}) = Gen(B_i^{new})$, even if the adversary supplies his/her chosen password PW_i^{new} and imprints a new biometric B_i^{new} at the sensor. Due to collision-resistant property of $h(\cdot)$, the adversary has no feasible way to update password and biometric of the user U_i.

5.1.3 *Replay Attack*

In this attack, an adversary tries to pretend to be a valid/authorized user by logging to the BS by sending the messages which were previously delivered by a legal user U_i. Let the adversary intercept the login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ during the login phase, where $M_1 = r_i^* \oplus h(ID_i||K) = h(ID_i||X_s)$, $M_2 = h(ID_i||X_s) \oplus RN_{U_i}$, and $M_3 = h(ID_i||ID_{SN_j}||M_1||RN_{U_i}||T_1)$. Assume that the adversary replies this login request to the BS. The BS can easily check the validity of this login request by means of checking the validity of T_1, the timestamp of U_i's smart card, SC_i. If the verification fails, it ensures this login request is a replay one, and the BS will immediately reject it. As a result, our scheme protects the replay attack.

5.1.4 *Man-in-the-Middle Attack*

Suppose an adversary eavesdrops a valid login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ during the login phase. In order to change M_2 and M_3, the adversary needs to know $M_1 = h(ID_i||X_s)$ and ID_i. Since both ID_i and X_s are protected by a collision-resistant hash function $h(\cdot)$, it is a computationally infeasible task for the adversary to change M_2 and M_3 as $M_2' = h(ID_i||X_s) \oplus RN_{U_i}'$ and $M_3' = h(ID_i||ID_{SN_j}||h(ID_i||X_s)||\ RN_{U_i}'||T_1')$, where T_1' is the current timestamp of the adversary's system, and RN_{U_i}' is the random nonce generated by the adversary. This clearly shows that the adversary does not have any ability to change or modify the login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ to a fake login request $\langle ID_{SN_j}, M_2', M_3', T_1'\rangle$, and hence, our scheme prevents the man-in-the-middle attack.

5.1.5 Denial-of-Service Attack

In our scheme, the BS sends the authentication request $\langle ID_{SN_j}, M_7\rangle$ to a sensor-login node SN_j. In reply, SN_j sends the authentication reply $\langle M_8, M_9, T_5\rangle$ to the user U_i. If an adversary blocks the messages from reaching SN_j and U_i, both SN_j and U_i will know about malicious dropping of such control messages. On the other hand, if the adversary does the malicious flooding of the authentication requests to the sensor node SN_j, it needs to perform only one symmetric decryption and two hash operations. Due to the computational efficiency of such operations, checking the authenticity of the malicious authentication requests does not occupy too much energy, memory as well as computational resources. Thus, we say that our scheme has also the ability to resist the denial-of-service attack.

5.1.6 Many Logged-in Users with the Same Login-ID Attack

In our scheme, the hash value of a user's password PW_i is embedded with his/her personal biometric data key σ_i and the random number K of U_i. As a result, even if two users U_i and U_j have the same password, say PW, the hash values $f_i = h(ID_i||RPW_i||\sigma_i)$ and $f_j = h(ID_j||RPW_j||\sigma_j)$ are also distinct, where $RPW_i = h(ID_i||K_1||PW)$, $RPW_j = h(ID_j||K_2||PW)$, $Gen(B_i) = (\sigma_i, \tau_i)$ and $Gen(B_j) = (\sigma_j, \tau_j)$. Here ID_i and ID_j are the identities of U_i and U_j respectively, K_1 and K_2 are the secret random numbers of U_i and U_j respectively, and B_i and B_j are the biometrics of U_i and U_j respectively. Further, even if the password PW is same for both U_i and U_j, in order to login to the BS (network) they need to pass biometric verification. Hence, our scheme prevents the many logged-in users with the same login-id attack.

5.1.7 Privileged-Insider Attack

During the registration phase, an insider being an attacker at the BS may try to derive PW_i and the biometric data key σ_i of a legal user U_i. However, note that in our scheme the user U_i sends the registration request $\langle ID_i, RPW_i\rangle$ to the BS securely, where $RPW_i = h(ID_i||K||PW_i)$. Since the 1,024-bit secret number K and the password PW_i are unknown to the insider attacker, it is a computationally infeasible to derive the password PW_i due to the collision-resistant property of the one-way hash function $h(\cdot)$.

We also assume that a legal user U_i is a malicious attacker. U_i can extract the information $\{e_i, f_i, r_i^*, \tau_i, Gen(\cdot), Rep(\cdot), t\}$ from his/her smart card. Then using his/her ID_i and B_i, U_i can compute $\sigma_i = Rep(B_i, \tau_i)$, $K = h(ID_i||\sigma_i) \oplus e_i$, and $h(ID_i||X_s) = r_i^* \oplus h(ID_i||K)$. It can be noted that the BS's master secret key X_s can not be extracted within polynomial time from $h(ID_i||X_s)$, since the probability of guessing X_s is $\frac{1}{2^k}$, which is negligible, where k is the bit-length of X_s (in our scheme, $k = 1{,}024$). Hence, the proposed scheme is secure against privileged-insider attacker.

5.1.8 Protection of User Anonymity

Suppose an attacker intercepts the login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ during the login phase, and the authentication request $\langle ID_{SN_j}, M_7\rangle$ and the authentication reply $\langle M_8, M_9, T_5\rangle$ during the authentication and key agreement phase. Note that these values are protected by the one-way collision-resistant hash function $h(\cdot)$ and also determined by two random nonces RN_{U_i} and RN_{SN_j}, and two timestamps T_1 and T_5. Due to this, these messages are different in each protocol run and as a result, the attacker does not have any ability to link two login messages of a particular user U_i. Hence, our scheme preserves the user anonymity property.

5.1.9 Session Key Security

Suppose an attacker intercepts the login request $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ during the login phase, and the authentication request $\langle ID_{SN_j}, M_7\rangle$ and the authentication reply $\langle M_8, M_9, T_5\rangle$ during the authentication and key agreement phase. The secret session key $SK_{ij} = h(ID_i||ID_{SN_j}||h(M_4)||M_5||RN_{SN_j}||T_1||T_5)$ is embedded with ID_i, $M_4 = h(ID_i||X_S)$, RN_{U_i} and RN_{SN_j}, and also protected by the one-way hash function $h(\cdot)$. In order to compute SK_{ij}, an adversary needs to know ID_i, X_s, RN_{U_i}, and RN_{SN_j}. Due to the collision-resistant one-way property of $h(\cdot)$, it is a computationally infeasible problem for the attacker to derive SK_{ij}. Thus, our scheme provides the session key security.

5.1.10 Stolen-Verifier Attack

One of the interesting characteristics of our proposed scheme is that our scheme does not store any verifier/password table for verification. The insider of the network can not get/steal user's password and biometrics because the BS and sensor nodes do not maintain any password/verifier table in order to validate user's login request. Our scheme is then resilient against the stolen-verifier attack.

5.1.11 Three-Factor Security

As described in [34], in the three-factor security model the goals of adversary, who has learned at most two components of the triple {password, smart card, biometric}, are to mount an impersonation attack (as a legal user or the BS) in order to obtain the last component or to compromise the user anonymity. Note that our scheme preserves the user anonymity property which is shown in Sect. 5.1.8. Even if the user's smart card is stolen, the adversary can not know ID_i, X_s, PW_i, σ_i, which is evident from Sect. 5.1.1. In order to impersonate the adversary does not have any ability to modify the login request or authentication request/reply due to the collision-resistant property of $h(\cdot)$. Therefore, our scheme satisfies the three-factor security property.

5.1.12 Resilience Against Node Capture Attack

We measure the resilience against node capture attack of a user authentication scheme in WSN by estimating the fraction of total secure communications that are compromised by a capture of c nodes *not including* the communication in which the compromised nodes are directly involved [16]. In other words, we need to find out the effect of c sensor nodes being compromised on the rest of the network. For example, for any non-compromised sensor node SN_j, we need to compute the probability that the adversary can decrypt the secure communication between a sensor-login node SN_j and a legal user U_i, when c sensor nodes are already compromised in the network? If we denote this probability by $P_e(c)$, we call such a scheme is *perfectly secure* or *unconditionally secure against node capture attack* when $P_e(c) = 0$. Assume that an adversary physically captures a sensor node, say SN_j randomly in the target field from which the user U_i accesses the real-time data from SN_j. Since the sensor nodes are not equipped with tamper-resistant hardware, the adversary can easily compromise all the secret information including the captured sensor node's master key and session key shared with the user U_i. The session key is generated using the random nonce RN_{U_i} and the timestamps of the user U_i and the sensor node SN_j, and thus each established session key between a user and a sensor node is distinct throughout the network. Also, each sensor node is loaded with a unique random master key prior to its deployment in the target field. As a result, the adversary has the ability to compromise the master key of that captured sensor node only. However, other non-compromised sensor nodes can still communicate securely with the actual real-time data to their corresponding legitimate users. The compromise of a sensor node does not reveal any other information about other sensor nodes and users in order to compromise any other secure communication between the users and the non-compromised nodes in the network. In other words, we have $P_e(c) = 0$. Hence, our scheme is unconditionally secure against node capture attack.

5.2 Formal Security Analysis

In this section, we show that our scheme is secure against an adversary using the formal security analysis under the random oracle model. We use the proof of the formal security by the method of contradiction as in [7]. We follow the similar analysis as in [11,14,15,27], [28]. Note that one can also prove the formal security in the standard model. However, in this paper, we have performed the formal security analysis under the generic group model of cryptography.

In order to use the method of contradiction proof [7] for our formal security analysis, we assume that there exists the following oracle for an adversary:

– $Reveal$: This oracle will unconditionally output the input string x from the corresponding hash value $y = h(x)$.

Theorem 1 *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like an oracle, our scheme is provably secure against an adversary for deriving the identity ID_i of a legal user U_i and the secret information X_s of the BS.*

Proof We follow the proof of this theorem similar to that in [11,14,15,27,28]. We need to construct an adversary $\mathcal{A}$ who will have the ability to derive the identity ID_i of a legal user U_i and the secret information X_s of the BS. The adversary $\mathcal{A}$ uses the $Reveal$ oracle in the experiment $EXP1^{HASH}_{\mathcal{A},UAS}$ provided in Algorithm 1 for our proposed secure and efficient user anonymity-preserving three-factor authentication scheme, say UAS. The success probability of $EXP1^{HASH}_{\mathcal{A},UAS}$ is defined by $Succ1 = |Pr[EXP1^{HASH}_{\mathcal{A},UAS} = 1]-1|$. The advantage function for this experiment becomes $Adv1\ (et_1, q_R) = max_{\mathcal{A}}\{Succ1\}$, where the maximum is taken over all $\mathcal{A}$ with execution time et_1, and the number of queries q_R made to the $Reveal$ oracle. Our scheme is provably secure against the adversary $\mathcal{A}$ for deriving the identity ID_i of a legal user U_i and the secret information X_s of the BS, if $Adv1 \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Consider the experiment $EXP1^{HASH}_{\mathcal{A},UAS}$ provided in Algorithm 1. According to this experiment, if the adversary $\mathcal{A}$ has the ability to invert a one-way hash function $h(\cdot)$, he/she can successfully derive the identity ID_i of a legal user U_i and the secret information X_s of the BS and win the game. However, by Definition 1, it is a computationally hard problem to invert $h(\cdot)$ due to collision-resistant property, that is, $Adv^{HASH}_{\mathcal{A}}(t) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$. As a result, $Adv1\ (et_1, q_R) \leq \epsilon$, since it is dependent on $Adv^{HASH}_{\mathcal{A}}(t)$. Hence, our scheme is provably secure against an adversary for deriving the identity ID_i of a legal user U_i and the secret information X_s of the BS. $\square$

Algorithm 1 $EXP1^{HASH}_{\mathcal{A},UAS}$

```
1: Intercept the login request $\langle ID_{SN_j}, M_2, M_3, T_1 \rangle$ during the login phase.
2: Call $Reveal$ oracle on input $M_3$ to retrieve the information $ID_i, ID_{SN_j}, M_1, RN_{U_i}$, and $T_1$ as
   $(ID'_i, ID'_{SN_j}, M'_1, RN'_{U_i}, T'_1) \leftarrow Reveal(M_3)$.
3: Compute $M_2^* = M'_1 \oplus RN'_{U_i}$.
4: if $(ID'_{SN_j} = ID_{SN_j})$ and $(M_2^* = M_2)$ and $(T'_1 = T_1)$ then
5:    Call $Reveal$ oracle on input $M'_1$ to retrieve $ID_i$ and $X_s$ as $(ID''_i, X''_s) \leftarrow Reveal(M'_1)$.
6:    if $(ID''_i = ID_i)$ then
7:       Accept $ID''_i$ as the correct $ID_i$ of the user $U_i$.
8:       Accept $X''_s$ as the correct $X_s$ of the BS.
9:       return 1 (Success)
10:   else
11:      return 0 (Failure)
12:   end if
13: else
14:   return 0 (Failure)
15: end if
```

Theorem 2 *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like an oracle, our scheme is provably secure against an adversary for deriving the password PW_i and the biometric data key σ_i of a legal user U_i, and the secret information X_s of the BS, even if the smart card SC_i of U_i is lost or stolen.*

Proof We follow the proof of this theorem similar to that in Theorem 1. We assume that the smart card SC_i of a legal user U_i is stolen or lost. An adversary $\mathcal{A}$ thus knows all the sensitive information stored in that smart card according to our threat model provided in Sect. 1.1. We construct the adversary $\mathcal{A}$ who will have the ability to derive the password PW_i and the biometric key σ_i of a legal user U_i, and the secret information X_s of the BS. The adversary $\mathcal{A}$ uses the oracle $Reveal$ in the experiment $EXP2^{HASH}_{\mathcal{A},UAS}$ provided in Algorithm 2 for our proposed scheme, UAS. The success probability of $EXP2^{HASH}_{\mathcal{A},UAS}$ is given by $Succ2 = |Pr[EXP2^{HASH}_{\mathcal{A},UAS} = 1] - 1|$. The advantage function for this experiment then becomes $Adv2\ (et_2, q_R) = max_{\mathcal{A}}\{Succ2\}$, where the maximum is taken over all $\mathcal{A}$ with execution time et_2, and the number of queries q_R made to the $Reveal$ oracle. Our scheme is called provably secure against the adversary $\mathcal{A}$ for deriving the the password PW_i and the biometric data key σ_i of a legal user U_i, and the secret information X_s of the BS, if $Adv2 \leq \epsilon$, for any sufficiently small $\epsilon > 0$. According to the experiment given in Algorithm 2, the adversary $\mathcal{A}$ can derive successfully PW_i, σ_i and X_s, if he/she can invert a one-way hash function $h(\cdot)$ and in that case, he/she will win the game. However, by Definition 1, it is a computationally hard problem to invert $h(\cdot)$ due to collision-resistant property, that is, $Adv^{HASH}_{\mathcal{A}}(t) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$. We then have, $Adv2\ (et_2, q_R) \leq \epsilon$, since it is dependent on $Adv^{HASH}_{\mathcal{A}}(t)$. As a result, our scheme is also provably secure against an adversary for deriving PW_i, σ_i and X_s. □

6 Simulation for Formal Security Verification of the Proposed Scheme Using AVISPA Tool

In this section, we evaluate our scheme for the formal security verification using the widely-accepted AVISPA tool.

Algorithm 2 $EXP2^{HASH}_{\mathcal{A},UAS}$

```
1: Extract all the sensitive information $e_i$, $f_i$, and $r_i^*$ from the lost/stolen smart card of the legal user $U_i$,
   where $e_i = h(ID_i||\sigma_i) \oplus K$, $f_i = h(ID_i||RPW_i||\sigma_i)$, $r_i^* = r_i \oplus h(ID_i||K) = h(ID_i||X_s) \oplus h(ID_i||K)$,
   $K$ being a 1,024-bit secret number known to the user $U_i$ only.
2: Call $Reveal$ oracle on input $f_i$ to retrieve $ID_i$, $RPW_i$ and $\sigma_i$ as $(ID_i', RPW_i', \sigma_i') \leftarrow Reveal(f_i)$.
3: Compute $K' = e_i \oplus h(ID_i'||\sigma_i')$.
4: Call $Reveal$ oracle on input $RPW_i'$ to retrieve $ID_i$, $K$ and $PW_i$ as $(ID_i'', K'', PW_i'') \leftarrow Reveal(RPW_i')$.
5: if $(K'' = K')$ then
6:     Accept $PW_i''$ as the correct password $PW_i$ and $\sigma_i'$ as the correct biometric key of the user $U_i$.
7:     Compute $u = r_i^* \oplus K''$.
8:     Call $Reveal$ oracle on input $u$ to retrieve $ID_i$ and $X_s$ as $(ID_i^*, X_s^*) \leftarrow Reveal(u)$.
9:     if $(ID_i^* = ID_i'')$ then
10:        Accept $X_s^*$ as the correct secret information of the BS.
11:        return 1 (Success)
12:    else
13:        return 0 (Failure)
14:    end if
15: else
16:    return 0 (Failure)
17: end if
```

AVISPA is known as a push-button tool for the automated validation of Internet security-sensitive protocols and applications. AVISPA provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques [1]. We use the widely-accepted AVISPA tool for our formal security verification [5,10,11,13,14]. AVISPA implements four back-ends and abstraction-based methods that are integrated through the high level protocol specific language, known as HLPSL [36]. A static analysis is performed to check the executability of the protocol. The protocol along with the intruder actions are compiled into an intermediate format (IF). IF is the start point for the four automated protocol analysis techniques, which is a lower-level language than HLPSL and is read directly by the back-ends to the AVISPA tool. The first back-end, known as the On-the-fly Model-Checker (OFMC), does several symbolic techniques to explore the state space in a demand-driven way [3]. The second back-end, known as the CL-AtSe (Constraint-Logic-based Attack Searcher), provides a translation from any security protocol specification written as transition relation in intermediate format into a set of constraints which are effectively used to find whether there are attacks on protocols. The third back-end, known as the SAT-based Model-Checker (SATMC), builds a propositional formula which is then fed to a state-of-the-art SAT solver and any model found is translated back into an attack. Finally, the fourth back-end, known as TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols), approximates the intruder knowledge by using regular tree languages.

The output format (OF) of AVISPA is generated by using one of the four back-ends explained above. When the analysis of a protocol has been successful (by finding an attack or not), the output describes precisely what is the result, and under what conditions it has been obtained. In OF, there are the following sections.

– The first printed section SUMMARY indicates that whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive.

- The second 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 are 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 [1]:

- *agent:* Values of type *agent* represent principal names. The intruder is always assumed to have the special identifier i.
- *public_key:* These values represent agents' public keys in a public-key cryptosystem. For example, given a public (respectively private) key pk, its inverse private (respectively public) key is obtained by inv_pk.
- *symmetric_key:* Variables of this type represent keys for a symmetric-key cryptosystem.
- *text:* In HLPSL, *text* values are often used as nonces. These values can be used for messages. If Na is of type *text (fresh)*, then Na' will be a fresh value which the intruder cannot guess.
- *nat:* The *nat* type represents the natural numbers in non-message contexts.
- *const:* This type represents constants.
- *hash_func:* The base type *hash_func* represents cryptographic hash functions. The base type function also represents functions on the space of messages. It is assumed that the intruder cannot invert hash functions (in essence, that they are one-way).
- *bool:* Boolean values are useful for modeling, for instance, binary flags.

The space of legal messages are defined as the closure of the basic types. For a given message *Msg* and encryption key *key*, we denote $\{Msg\}\_key$ as the symmetric/ public-key encryption. The associative "$\cdot$" operator is used for concatenations.

The "*played_by A*" declaration indicates that the agent named in variable A will play in the role. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions have the form $X = | > Y$, which relate an event X and an action Y. This means that whenever we take a transition that is labeled in such a way as to make the event predicate X true, we must immediately (that is, simultaneously) execute action Y. If a variable V remains permanently secret, it is expressed by the goal *secrecy_of V*. If V is ever obtained or derived by the intruder, a security violation will result.

6.1 Specifying our Scheme

We have implemented our proposed scheme in HLPSL language. In our implementation, we have three basic roles: alice, bs and bob, which represent the three participants: the user U_i, the base station BS and the sensor node SN_j, respectively. We have further specified the session and environment in our implementation.

In Fig. 2, we have specified the role specification for the user U_i of our scheme in HLPSL. During the registration phase, the user U_i sends the registration request $\langle ID_i, RPW_i\rangle$ to the BS via a secure channel using the $Snd(\,)$ operation. Note that the type declaration $channel\ (dy)$ indicates that the channel is for the Dolev–Yao threat model. After that U_i receives the smart card containing the information $\{r_i, h(\cdot)\}$ via a secure channel from the BS using the $Rcv(\,)$ operation. During the login phase, U_i sends the login message $\langle ID_{SN_j}, M_2, M_3, T_1\rangle$ to the BS. In HLPSL, secret(Xs, subs2, BS) declares that the secret information X_s of the BS

```
role alice (Ui, BS, SNj : agent,
          SKuibs : symmetric_key,
          Snd, Rcv: channel(dy))
played_by Ui
def=
local State  : nat,
     IDi, IDsnj, RNui, RNsnj, T1, T3, T5 :  text,
     K, PWi, RPWi, Xs : text,
% Bi: biometric secret key
% Ti: biomteric public parameter
     Bi, Ti : text,
     M1, M2, M3, M5, M6, M7: text,
     MKsnj : symmetric_key,
     H : hash_func, Gen, Rep : hash_func
const alice_server_t1, alice_server_rnui,
     server_bob_t3, bob_alice_rnsnj,
     bob_alice_t5, subs1, subs2,
     subs3, subs4, subs5 : protocol_id
init  State := 0
transition
1. State = 0 /\ Rcv(start) =|>
% Registration phase
State' := 1 /\  RPWi' := H(IDi.K.PWi)
% Ui sends the registration request to the BS via a secure channel
          /\ Snd({IDi.RPWi'}_SKuibs)
          /\ secret({PWi,Bi,K},subs1,Ui)
          /\ secret(Xs, subs2, BS)
          /\ secret(MKsnj, subs3, {BS,SNj})
          /\ secret(SKuibs, subs4, {Ui,BS})
          /\ secret(IDi, subs5, {Ui,BS,SNj})
% Ui receives the smart card from the BS
2. State = 1 /\ Rcv({H(IDi.Xs)}_SKuibs) =|>
% Login phase
State' := 2 /\ RNui' := new()
          /\ T1' := new()
          /\ M1' := H(IDi.Xs)
          /\ M2' := xor(M1',RNui')
          /\ M3' := H(IDi.IDsnj.M1'.RNui'.T1')
% Ui sends the login message to the BS
          /\ Snd(IDsnj.M2'.M3'.T1')
% Ui has freshly generated the values T1 and RNui for BS
          /\ witness (Ui,BS,alice_server_t1, T1')
          /\ witness (Ui,BS,alice_server_rnui, RNui')
% Authentication phase
% Ui receives the message from SNj
3. State = 2 /\ Rcv(H(H(IDi.IDsnj.H(H(IDi.Xs)).RNui'.RNsnj'.T1'.T5')).
            xor(xor(RNui',RNsnj'),IDi).T5') =|>
% Ui's acceptance of the values RNsnj and T5 generated for Ui by SNj
  State' := 3 /\ request(SNj, Ui, bob_alice_rnsnj, RNsnj)
            /\ request(SNj, Ui, bob_alice_t5, T5')
end role
```

Fig. 2 Role specification in HLPSL for the user U_i of our scheme

is permanently kept secret to the BS only characterized by the protocol id subs2. By the declaration witness(A, B, id, E), we mean it is for a (weak) authentication property of A by B on E, which declares that agent A is witness for the information E; this goal will be identified by the constant id in the goal section. This means that the agent named in variable B has freshly generated the value E for the agent named in variable A. The id term is a new constant that identifies the message term upon which the goal should authenticate. On

```
role server (Ui, BS, SNj : agent,
            SKuibs : symmetric_key,
            Snd, Rcv: channel(dy))
played_by BS
def=
  local State  : nat,
     IDi, IDsnj, RNui, RNsnj, T1, T3, T5 :  text,
     K, PWi, RPWi, Xs : text,
% Bi: biometric secret key
% Ti: biomteric public parameter
     Bi, Ti : text,
     M1, M2, M3, M4, M5, M6, M7, M8, M9: text,
     MKsnj : symmetric_key,
     H : hash_func, Gen, Rep : hash_func
const alice_server_t1, alice_server_rnui,
     server_bob_t3, bob_alice_rnsnj,
     bob_alice_t5, subs1, subs2,
     subs3, subs4, subs5 : protocol_id
init  State := 0
transition
% Registration phase
% BS receives the registration requestfrom Ui via a secure channel
1. State   = 0 /\ Rcv({IDi.H(IDi.K.PWi)}_SKuibs)=|>
% BS issues a smart card to Ui
  State' := 1 /\ secret(Xs, subs2, BS)
            /\ secret(MKsnj, subs3, {BS,SNj})
            /\ secret(SKuibs, subs4, {Ui,BS})
            /\ secret(IDi, subs5, {Ui,BS,SNj})
            /\ Snd({H(IDi.Xs)}_SKuibs)
% Login phase
2. State = 1  /\ Rcv(IDsnj.xor(H(IDi.Xs),RNui').
              H(IDi.IDsnj.M1'.RNui'.T1').T1')=|>
% Authentication phase
  State' := 2 /\ T3' := new()
            /\ M4' := H(IDi.Xs)
            /\ M5' := RNui'
            /\ M7' := {IDi.IDsnj.M5'.H(M4').T1'.T3'}_MKsnj
            /\ Snd(IDsnj.M7')
% BS has freshly generated the value T3 for SNj
            /\ witness (BS,SNj,server_bob_t3, T3')
% BS's acceptance of the values T1 and RNui generated for BS by Ui
            /\ request(Ui, BS, alice_server_t1, T1')
            /\ request(Ui,BS,alice_server_rnui, RNui')
end role
```

Fig. 3 Role specification in HLPSL for the BS of our scheme

the other hand, request(B, A, id, E) indicates for a strong authentication property of A by B on E, declares that agent B requests a check of the value E; this goal will be identified by the constant id in the goal section. This formalizes A's acceptance of the value E as having generated for him/her by the agent named in B. During the authentication and key agreement phase, U_i waits for the authentication reply $\langle M_8, M_9, T_5 \rangle$ from the sensor node SN_j. authentication_on alice_server_rnui declares that U_i (SC_i) generates a random nonce RN_{U_i}, where RN_{U_i} is only known to U_i. When the BS receives RN_{U_i} from other messages from U_i, the BS performs a strong authentication for U_i based on RN_{U_i}. In a similar way, the role specifications of the BS and the sensor node SN_j along with all the exchanged messages are shown in Figs. 3 and 4, respectively.

Fig. 4 Role specification in HLPSL for the sensor node SN_j of our scheme

```
role bob (Ui, BS, SNj : agent,
          SKuibs : symmetric_key,
          Snd, Rcv: channel(dy))
played_by SNj
def=
  local State  : nat,
      IDi, IDsnj, RNui, RNsnj, T1, T3, T5 :  text,
      K, PWi, RPWi, Xs, SKij : text,
% Bi: biometric secret key
% Ti: biomteric public parameter
      Bi, Ti : text,
      M1, M2, M3, M5, M6, M7, M8, M9: text,
      MKsnj : symmetric_key,
      H : hash_func, Gen, Rep : hash_func
const alice_server_t1, alice_server_rnui,
      server_bob_t3, bob_alice_rnsnj,
      bob_alice_t5, subs1, subs2,
      subs3, subs4, subs5 : protocol_id
init  State := 0
transition
% Authentication phase
% SNj receives the authentication request from BS
1. State = 0 /\ Rcv(IDsnj.{IDi.IDsnj.RNui'.
               H( H(IDi.Xs)).T1'.T3'}_MKsnj)=|>
  State' := 1 /\ secret(Xs, subs2, BS)
             /\ secret(MKsnj, subs3, {BS,SNj})
             /\ secret(SKuibs, subs4, {Ui,BS})
             /\ secret(IDi, subs5, {Ui,BS,SNj})
             /\ Snd({H(IDi.Xs).IDsnj}_SKuibs)
             /\ RNsnj' := new()
             /\ T5' := new()
% SNj sends an acknowledgement to Ui
            /\ SKij' := H(IDi.IDsnj.H(H(IDi.Xs))
                      .RNui'.RNsnj'.T1'.T5')
            /\ M8' := H(SKij')
            /\ M9' := xor(xor(RNui',RNsnj'),IDi)
            /\ Snd(M8'.M9'.T5')
% SNj has freshly generated the values T5 and RNsnj for Ui
           /\ witness (SNj,Ui,bob_alice_t5, T5')
           /\ witness (SNj,Ui,bob_alice_rnsnj, RNsnj')
% SNj's acceptance of the value T3 generated for SNj by BS
           /\ request(BS, SNj, server_bob_t3, T3')
end role
```

Fig. 5 Role specification in HLPSL for the session of our scheme

```
role session(Ui,BS,SNj: agent,
    SKuibs : symmetric_key)
def=
  local  US, UR, SS, SR, VS, VR: channel (dy)
  composition
         alice(Ui, BS, SNj, SKuibs, US, UR)
       /\ server(Ui, BS, SNj, SKuibs, SS, SR)
       /\  bob(Ui, BS, SNj, SKuibs, VS, VR)
end role
```

In Figs. 5 and 6, we have implemented the specifications in HLPSL language for the role of session and environment of our scheme, respectively. 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 HLPSL. This role contains global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate

Fig. 6 Role specification in HLPSL for the goal and environment of our scheme

```
role environment()
def=
  const ui, bs, snj  : agent,
        skuibs : symmetric_key,
        h : hash_func,
        gen, rep : hash_func,
        idsnj, ti: text,
        alice_server_t1, alice_server_rnui,
        server_bob_t3, bob_alice_rnsnj,
        bob_alice_t5, subs1, subs2,
        subs3, subs4, subs5 : protocol_id
intruder_knowledge = {bs, snj, h, gen, rep, ti, idsnj}
composition
session(ui, bs, snj, skuibs)
     /\ session(ui, bs, snj, skuibs)
     /\ session(ui, bs, snj, skuibs)
end role

goal
secrecy_of subs1
secrecy_of subs2
secrecy_of subs3
secrecy_of subs4
secrecy_of subs5
authentication_on alice_server_rnui
authentication_on alice_server_t1
authentication_on server_bob_t3
authentication_on bob_alice_rnsnj
authentication_on bob_alice_t5
end goal
environment()
```

user. The intruder, which is always denoted by i, also participates in the execution of protocol as a concrete session.

6.2 Analysis of Results

We have chosen the OFMC backend for an execution test and a bounded number of sessions model checking [3] for our simulation. For the replay attack checking, OFMC checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that this back-end provides 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 using the AVISPA web tool [2]. The simulation result for the formal security verification of our scheme using OFMC is shown in Fig. 7. In this figure, the first printed section, called the SUMMARY, indicates whether the protocol is safe, unsafe, or whether the analysis is inconclusive. It is clear that our scheme is safe from the printed SUMMARY section. DETAILS section explains under what condition the protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive. It is also noted that our scheme is declared as safe, and no attack is found in our scheme. As a result, the result in this figure ensures that our scheme is secure against passive and active attacks including the replay and man-in-the-middle attacks.

Fig. 7 The result of the analysis using OFMC of our scheme

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web−interface−computation/
  ./tempdir/workfilegyLULF.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.79s
  visitedNodes: 55 nodes
  depth: 6 plies
```

Table 5 Functionality comparison between our scheme and other schemes

Functionality	[38]	[39]	[17]	[40]	[21]	[35]	[20]	[6]	Ours
F_1	No	No	No	Yes	Yes	Yes	No	No	Yes
F_2	Yes	No	No	No	No	Yes	Yes	Yes	Yes
F_3	No	No	No	No	No	No	No	No	Yes
F_4	No	No	No	Yes	No	No	No	No	Yes
F_5	No	No	No	No	No	Yes	Yes	No	Yes
F_6	No	No	No	No	No	Yes	No	No	Yes
F_7	No	No	No	No	No	No	No	No	Yes
F_8	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
F_9	Yes	No	No	No	No	No	Yes	No	Yes
F_{10}	No	No	No	No	No	No	No	No	Yes
F_{11}	No	No	No	No	No	No	No	No	Yes

F_1, whether supports password change or not; F_2, whether supports mutual authentication or not; F_3, whether supports biometric update or not; F_4, whether provides non-repudiation or not; F_5, whether resists denial-of-service attack or not; and F_6, whether resilient against node capture attack or not; F_7, whether provides three-factor security or not; F_8, whether preserves user anonymity or not; F_9, whether provides key agreement or not; F_{10}, whether provides formal security analysis or not; F_{11}, whether supports formal security verification using AVISPA tool or not

7 Performance Comparison with Related Schemes

In this section, we compare the performance of our scheme with other related existing schemes.

In Table 5, we have compared the functionalities of our scheme with other related schemes, such as Watro et al.'s scheme [38], Wong et al.'s scheme [39], M. L. Das's scheme [17], Yuan et al.'s scheme [40], He et al.'s scheme [21], Vaidya et al.'s scheme [35], Fan et al.'s scheme [20] and Chen-Shih's scheme [6]. From this table, it is clear that our scheme is superior with respect to all the functionality provided by our scheme such as preventing denial-of-service attack, supporting password and biometric update phase, mutual authentication and key agreement, non-repudiation because of employing the biometric of a user, user anonymity,

Table 6 Comparison of computation costs in different phases between our scheme and other schemes

Phase	User/node	[38]	[39]	[17]	[40]	[21]	[35]	[20]	[6]	Ours
R	U_i	$t_{pu} + t_{pr}$	–	–	–	t_h	t_h	–	–	$3t_h + t_{fe}$
	BS	t_{pr}	$3t_h$	$3t_h$	$4t_h$	$5t_h$	$4t_h$	$6t_h$	$3t_h$	t_h
	SN_j	–	–	–	–	–	–	–	–	–
L	U_i	$2t_{pr} + t_h$	–	$4t_h$	$4t_h$	$5t_h$	$6t_h$	$7t_h$	$4t_h$	$7t_h + t_{fe}$
+	BS	–	t_h	$4t_h$	$4t_h$	$5t_h$	$5t_h$	$2t_h$	$5t_h$	$2t_h + t_{enc}$
AK	SN_j	$2t_{pu} + t_h$	$3t_h$	t_h	t_h	t_h	$2t_h$	$2t_h$	t_h	$2t_h + t_{dec}$

R registration phase, *L* loin phase, *AK* authentication and key agreement phase

Table 7 Comparison of communication costs between our scheme and other schemes

Scheme	Communication cost
Watro et al. [38]	2 Messages (3,072 bits)
Wong et al. [39]	4 Messages (608 bits)
M. L. Das [17]	3 Messages (704 bits)
Yuan et al. [40]	3 Messages (704 bits)
He et al. [21]	3 Messages (736 bits)
Vaidya et al. [35]	5 Messages (944 bits)
Fan et al. [20]	5 Messages (1,232 bits)
Chen-Shih [6]	4 Messages (944 bits)
Ours	3 Messages (736 bits)

three-factor security as well as formal security analysis and verification, when compared those functionalities with other related schemes.

In Table 6, we have compared the computation cost of our scheme with other schemes during the registration phase, login phase, and authentication and key agreement phase. We have used the following notations for computing the computational costs of our scheme and other schemes: t_{pu}: public-key computation, t_{pr}: private-key computation, t_h: hash computation (SHA-1 [31]), t_{enc}: symmetric encryption (AES encryption [32]), t_{dec}: symmetric decryption (AES decryption [32]), t_{fe}: time for executing a fuzzy extractor function ($Gen(\cdot)$ and $Rep(\cdot)$). Note that during the registration phase, in our scheme a user U_i and the BS require the computation costs $3t_h + t_{fe}$ and t_h, respectively, whereas no computation cost is involved during this phase for a resource-constrained sensor node SN_j. During the login phase and the authentication and key agreement phase of our scheme, the computation costs for U_i, BS and SN_j are $7t_h + t_{fe}$, $2t_h + t_{enc}$ and $2t_h + t_{dec}$, respectively. Since the fuzzy extractor method is efficient, U_i and the BS do not require much computation costs. On the other hand, due to efficiency of one-way hash function $h(\cdot)$ and symmetric encryption/decryption, the sensor node SN_j does not require much computation cost and as a result, our scheme is very suited for the resource-constrained sensor nodes.

The communication costs in terms of the number of exchanged messages and bits for a successful user authentication for our scheme and the other schemes are shown in Table 7. We have used the number of bits required for the following fields for computation of communication costs between our scheme and other schemes. The identifiers of sensor node, base station (GW-node) and user are each 16 bits. The random nonce is 32 bits. The timestamp

Table 8 Comparison of sensor node's energy cost between our scheme and other schemes

Scheme	Sensor node's energy cost
Watro et al. [38]	Nonce validation + checksum generation and verification + two public-key operations + response to the user's query
Wong et al. [39]	Lookup table query + three hash operations for parameters generation + waiting for the GW-node's response + response to the user's query
M. L. Das [17]	Timestamp validation + one hash operation for parameter generation + response to the user's query
He et al. [21]	Timestamp validation + one hash operation for parameter generation + response to the user's query
Vaidya et al. [35]	Timestamp validation + one hash operation for parameter verification + one hash operation for parameter generation + response to the user's query + waiting for the GW-node's response
Fan et al. [20]	One hash operation for random nonce validation + one hash operation for session key generation + response to the user's query
Chen-Shih [6]	Timestamp validation + one hash operation for parameter generation + response to the user's query
Yuan et al. [40]	Timestamp validation + one hash operation for parameter generation + response to the user's query
Ours	One symmetric-key decryption + timestamp validation + two hash operations for session key generation and validation + response to the user's query

field is 32 bits. The public key encryption and decryption using RSA in Watro et al.'s scheme require each 1,024 bits. The AES encryption and decryption require each 128 bits. If we use SHA-1 as the one-way hash function, the message digest is 160 bits. From Table 7, it is clear that a successful user authentication process in our scheme requires 736 bits, while Watro et al.'s scheme, Wong et al.'s scheme, M. L. Das's scheme, Yuan et al.'s scheme, He et al.'s scheme, Vaidya et al.'s scheme, Fan et al.'s scheme and Chen-Shih's scheme require 3,072, 608, 704, 704, 736, 944, 1,232 and 944 bits, respectively. Though our scheme requires little more communication overhead than Wong et al.'s scheme, M. L. Das's scheme and Yuan et al.'s scheme, these schemes are insecure against some known attacks, which are discussed in Sect. 3.

Finally, we have compared the energy cost required for a sensor node among our scheme and other schemes in Table 8. Note that a sensor node's energy cost is mainly due to both computation and communication costs involved in the schemes. For Watro et al.'s scheme, a sensor node consumes battery due to nonce validation, checksum generation and verification, two public-key operations and then response to the user's query. In Wong et al.'s scheme, a sensor node consumes battery for a lookup table query, three hash operations for parameters generation and then waiting for the GW-node's response before responding to the user's query. In both M. L. Das's scheme and Yuan et al.'s scheme, a sensor node consumes battery due to timestamp validation and one hash operation for parameter generation and finally for responding to the user's query. In He et al.'s scheme, a sensor node consumes battery due to timestamp validation, one hash function for parameter generation and response to the user's query. In Vaidya et al.'s scheme, battery consumption for a sensor node comes due to timestamp validation, one hash function for parameter generation, another hash function for parameter verification, and then response to the user's query and waiting for the GW-node's response. Fan et al's scheme requires battery consumption for a sensor node due to one hash function for random-nonce validation, another hash function for session key generation and then response to the user's query. Chen-Shih's scheme involves battery consumption for a sensor node due to time-stamp validation, one hash function for parameter generation and response to the user's query. Finally, in our scheme, a sensor node consumes battery due to one symmetric decryption, timestamp validation, two hash operations for session key generation and validation and response to the user's query. Due to efficient hash and symmetric-key operations, a sensor node's energy cost in our scheme is comparable with that for the other schemes.

8 Conclusion

In this paper, we have addressed the user authentication problem by introducing a novel three-factor scheme for the resource-constrained DWSN. Our scheme supports efficiently updating password and biometric change phase without contacting the BS and dynamic node addition phase, which are considered as crucial factors in this area. Our scheme is shown to be secure against possible known attacks, which is evident through both informal and formal security analysis and verification. In addition, our scheme is very suited for resource-constrained sensor nodes due to the computation and communication efficiency as compared to those for other related schemes proposed in the literature. Overall, higher security along with low communication and computation costs make our scheme appropriate for practical WSN applications.

Acknowledgments The author would like to acknowledge the many helpful suggestions of the anonymous reviewers and the Editor, which have improved the content and the presentation of this paper.

References

1. AVISPA. Automated Validation of Internet Security Protocols and Applications. http://www.avispa-project.org/. Accessed on January 2013.
2. AVISPA. AVISPA Web Tool. http://www.avispa-project.org/web-interface/expert.php/. Accessed on May 2014.
3. Basin, D., Modersheim, S., & Vigano, L. (2005). OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, *4*(3), 181–208.
4. Burnett, A., Byrne, F., Dowling, T., & Duffy, A. (2007). A biometric identity based signature scheme. *International Journal of Network Security*, *5*(3), 317–326.
5. Chatterjee, S., Das, A. K., & Sing, J. K. (2014). An enhanced access control scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, *21*(1–2), 121–149.
6. Chen, T.-H., & Shih, W.-K. (2010). A robust mutual authentication protocol for wireless sensor networks. *ETRI Journal*, *32*(5), 704–712.
7. Chuang, Y.-H., & Tseng, Y.-M. (2010). An efficient dynamic group key agreement protocol for imbalanced wireless networks. *International Journal of Network Management*, *20*(4), 167–180.
8. Das, A. K. (2011). Analysis and improvement on an efficient biometric-based remote user authentication scheme using smart cards. *IET Information Security*, *5*(3), 145–151.
9. Das, A. K. (2012). A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. *International Journal of Information Security*, *11*(3), 189–211.
10. Das, A. K. (2013). A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. *Networking Science*, *2*(1–2), 12–27.

11. Das, A. K., Chatterjee, S., & Sing, J. K. (2013). A novel efficient access control scheme for large-scale distributed wireless sensor networks. *International Journal of Foundations of Computer Science*, *24*(5), 625–653.
12. Das, A. K., Chatterjee, S., & Sing, J. K. (2013). Formal security verification of a dynamic password-based user authentication scheme for hierarchical wireless sensor networks. In: *International Symposium on Security in Computing and Communications (SSCC 2013), Communications in Computer and Information Science Series (CCIS)*. (Vol. 377, pp. 243–254).
13. Das, A. K., Chatterjee, S., & Sing, J. K. (2014). A New Biometric-Based Remote User Authentication Scheme in Hierarchical Wireless Body Area Sensor Networks. In: *Ad Hoc & Sensor Wireless Networks* (in press).
14. Das, A. K., & Goswami, A. (2013). A secure and efficient uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. *Journal of Medical Systems*, *37*(3), 1–16.
15. Das, A. K., Paul, N. R., & Tripathy, L. (2012). Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences*, *209*(C), 80–92.
16. Das, A. K., Sharma, P., Chatterjee, S., & Sing, J. K. (2012). A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Journal of Network and Computer Applications*, *35*(5), 1646–1656.
17. Das, M. L. (2009). Two-factor user authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications*, *8*(3), 1086–1090.
18. Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654.
19. Dodis, Y., Reyzin, L., & Smith, A. (2004). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: *Proceedings of the Advances in Cryptology (Eurocrypt'04), LNCS*, (Vol. 3027, pp. 523–540).
20. Fan, R., Ping, L.-D., Fu, J.-Q., & Pan, X.-Z. (2010). A secure and efficient user authentication protocol for two-tier wireless sensor networks. In *Second Pacific-Asia conference on circuits, communications and system (PACCS 2010)* (pp. 425–428).
21. He, D., Gao, Y., Chan, S., Chen, C., & Bu, J. (2010). An enhanced two-factor user authentication scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, *10*(4), 361–371.
22. He, D., Kumar, N., Lee, J.-H., & Sherratt, R. S. (2014). Enhanced three-factor security protocol for consumer USB mass storage devices. *IEEE Transactions on Consumer Electronics*, *60*(1), 30–37.
23. Khan, M. K., & Alghathbar, K. (2010). Cryptanalysis and security improvements of two-factor user authentication in wireless sensor networks. *Sensors*, *10*(3), 2450–2459.
24. Li, C.-T., & Hwang, M.-S. (2010). An efficient biometric-based remote authentication scheme using smart cards. *Journal of Network and Computer Applications*, *33*(1), 1–5.
25. Li, X., Niu, J.-W., Ma, J., Wang, W.-D., & Liu, C.-L. (2011). Cryptanalysis and improvement of a biometrics-based remote user authentication scheme using smart cards. *Journal of Network and Computer Applications*, *34*(1), 73–79.
26. Nyang, D. H., & Lee, M.-K. (2009). Improvement of Das's two-factor authentication protocol in wireless sensor networks. In *Cryptology ePrint Archive*, Report 2009/631.
27. Odelu, V., Das, A. K., & Goswami, A. (2013). An effective and secure key-management scheme for hierarchical access control in E-medicine system. *Journal of Medical Systems*, *37*(2), 1–18.
28. Odelu, V., Das, A. K., & Goswami, A. (2014). A secure effective key management scheme for dynamic access control in a large leaf class hierarchy. *Information Sciences*, *269*(C), 270–285.
29. Rivest, R. L., Shamir, A., & Adleman, L. M. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126.
30. Sarkar, P. (2010). A simple and generic construction of authenticated encryption with associated data. *ACM Transactions on Information and System Security*, *13*(4), 33.
31. Secure Hash Standard. FIPS PUB 180–1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995.
32. Stallings, W. (2003). *Cryptography and Network Security: Principles and Practices* (3rd ed.). Pearson Education India . Gaithersburg, USA.
33. Stinson, D. R. (2006). Some observations on the theory of cryptographic hash functions. *Designs, Codes and Cryptography*, *38*(2), 259–277.
34. Tan, Z. (2014). A user anonymity preserving three-factor authentication scheme for telecare medicine information systems. *Journal of Medical Systems*, *38*(3), 1–9.
35. Vaidya, B., Makrakis, D., & Mouftah, H. T. (2010). Improved two-factor user authentication in wireless sensor networks. In: *Second international workshop on network assurance and security services in ubiquitous environments* (pp. 600–606).

36. von Oheimb, D. (2005). The high-level protocol specification language HLPSL developed in the EU project AVISPA. In: *Proceedings of APPSEM 2005 Workshop*.
37. Wang, D., & Wang, P. (2014) Understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks. *Ad Hoc Networks* (in press). doi:10.1016/j.adhoc.2014.03.003.
38. Watro, R., Kong, D., Cuti, S., Gardiner, C., Lynn, C., & Kruus, P. (2004, October). Tinypk: Securing sensor networks with public key technology. In: *Proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks, SASN 2004, Washington, DC, USA* (pp. 59–64).
39. Wong, K., Zheng, Y., Cao, J., & Wang, S. (2006). A dynamic user authentication scheme for wireless sensor networks. In: *Proceedings of IEEE international conference on sensor networks, ubiquitous, and trustworthy computing, IEEE Computer Society* (pp. 244–251).
40. Yuan, J., Jiang, C., & Jiang, Z. (2010). A biometric-based user authentication for wireless sensor networks. *Wuhan University Journal of Natural Sciences*, *15*(3), 272–276.

Ashok Kumar Das is currently working as an Assistant Professor in the Center for Security, Theory and Algorithmic Research of the International Institute of Information Technology (IIIT), Hyderabad 500 032, India. He received his Ph.D. degree in Computer Science and Engineering, M.Tech. degree in Computer Science and Data Processing, and M.Sc. degree in Mathematics, all from the Indian Institute of Technology, Kharagpur, India. He received the Institute Silver Medal from the Indian Institute of Technology, Kharagpur, India. His current research interests include cryptography, wireless sensor network security, proxy signature, hierarchical access control, data mining and remote user authentication. He has published more than 60 papers in international journals and conferences in these areas. For more details, visit http://sites.google.com/site/iitkgpakdas/, http://www.iiit.ac.in/people/faculty/ashokkdas.