An Improved Dynamic Password-based User Authentication Scheme for Hierarchical Wireless Sensor Networks

M. Turkanovic¹, M. Holbl¹
¹Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia
muhamed.turkanovic@gmail.com

Abstract—User authentication is an important issue in wireless sensor networks. Das et al. recently proposed a dynamic password-based user authentication scheme for hierarchical wireless sensor networks, which provides high security and a simple authentication approach. In this paper we present a flaw in Das et al.’s scheme that makes it infeasible for real-life implementation. Additionally, we demonstrate that Das et al.’s scheme has redundant elements. To overcome these imperfections we propose an enhanced user authentication scheme based on Das et al.’s, which is both efficient and secure.

Index Terms—Password-based, user authentication, smart card, wireless sensor networks.

I. INTRODUCTION

Wireless sensor networks (WSNs) have become very popular and are being used in variety of applications (e.g., military, health, environment, etc.) [1]. The main goal of a WSN is to monitor, collect and process data from a specific location and deliver it to the end users. The end users, regarding the application, can be anything as a military headquarter, hospital doctor, farmer, etc., thus use the collected data for any kind of a reason (e.g., decision making process). Considering the fact that data collected from the WSN can be very important, it is crucial that it is also secure. The concern for security is even more crucial as communication is done wirelessly, using radio transmission and thus making eavesdropping more probable. To this date numerous researches has been done on the security of WSNs [2], [3]. Since a WSN consist of tiny sensor nodes with low processing power, the balance of efficiency and security is very important but sometimes hard to achieve [4].

One of the most important security issues for WSNs are so-called outside attacks, whereby user authentication is the first line of defense against these attacks [5]. Due to the resource constraint architecture of WSNs, public key infrastructure (PKI) is unsuitable, primarily because of the large energy consumption [6], [7]. Example of such a user authentication scheme based on PKI was presented by Watro, Kong [8], called TinyPK. In 2006, Wong, Zheng [9] proposed a password-based user authentication scheme using only hash functions. However the scheme was later found to be vulnerable to several attacks [10]. In the same paper Das et al. also proposed an improved scheme. Several schemes were later proposed to tackle the security issues of previous ones [11]–[18].

This paper demonstrates that Das et al.’s new dynamic password-based user authentication scheme [14] has a flaw and is infeasible for implementation in real-life environment. Moreover, we show that their scheme can be made more efficient by removing redundant elements. To overcome the flaw and the redundancy of Das et al.’s scheme we propose an improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks.

The rest of the paper is organized as follows: Section 2 presents preliminary concepts and background, Section 3 briefly reviews Das et al.’s scheme, Section 4 elaborates on the flaw and practical weaknesses of Das et al.’s scheme, Section 5 presents our proposed improvement scheme, Section 6 presents the security analysis of the proposed scheme, Sections 7 compares the performance of our scheme with some related schemes and finally we conclude the paper in Section 8.

II. WIRELESS SENSOR NETWORK

This section discusses the architecture of the WSN. Sensor networks consist of one or more base stations and multiple tiny sensor nodes which are equipped with a processor that allows them to sense, process and communicate data. Thus every sensor node is constructed of four basic units, i.e., sensing unit, processing unit, transceiver unit and a power unit. Sensor nodes can be sometimes even smaller than one cubic centimeter and deployed in a bunch of hundreds or even millions. The base stations are in contrary powerful nodes that connect all the sensor nodes and act like a gateway for the end user (e.g., access point, processing center, etc.). The main condition of a sensor node is that they consume extremely low power, have low production cost, are adaptive and autonomous.

There are a variety of sensor node types which can measure or monitor a large scale of conditions like humidity, temperature, light, etc. The advantage of a sensor network is the collaboration of multiple low-cost sensor nodes which can be randomly deployed irrespective of the terrain, thus enable real-time detection of the environment. The sensing
process of sensor nodes can be continuous or event based. Because of the utility of the sensor networks, they can be used in a variety of applications, i.e., military (e.g. battlefield surveillance, targeting help, attack detection, etc.), environmental (e.g. forest fire detection, agriculture, etc.), health (e.g. monitoring human physiological data) [19], etc. The deployment of sensor nodes in a specific field can be done using planes, catapults or placing them one by one. The biggest disadvantage of sensor nodes is the energy efficiency. Further details about WSN can be found in [20], [21].

A. Hierarchical wireless sensor network

Two main architectures of WSN organization are, distributed-flat and hierarchical [22]. Hierarchical wireless sensor networks (HWSN) are organized into several clusters and one main server (i.e., base station). Each cluster consist of one cluster head (CH) and several sensor nodes. CHs communicate with every cluster member (i.e., sensor node) in its cluster, with each CH in the network and with the base station (BS) of the network. Sensor nodes therefore communicate only with the CH and other sensor nodes in the cluster. Finally the hierarchy ends with the BS, whereby it communicates only with the CHs of the network and to the outside world as an access point for the collected data. Sensor nodes are tiny, low-cost, low-processing sensors with short radio transmission range and are responsible only for sensing. CHs in contrary are more powerful thus have more processing capabilities. They are responsible for collecting and processing the data from its cluster members and transferring it to the BS. Further details can be found in [23].

III. REVIEW OF DAS ET AL.’S SCHEME

This section briefly reviews Das et al.’s scheme [14]. There are four parties included in Das et al.’s scheme: a base station, a cluster head, a sensor node and a user. Sensor nodes are an inactive party since they have no function in the establishment of the secure connection. Their scheme consists of seven phases; the pre-deployment phase, the post-deployment phase, the registration phase, the login phase, the authentication phase, the password change phase and the dynamic node addition phase. The notations used in this paper are summarized in Table I.

### Table I. Notations used in this paper.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH&lt;sub&gt;j&lt;/sub&gt;, ID&lt;sub&gt;CHj&lt;/sub&gt;</td>
<td>Cluster head and the identifier of cluster head</td>
</tr>
<tr>
<td>PW&lt;sub&gt;j&lt;/sub&gt;, ID&lt;sub&gt;j&lt;/sub&gt;</td>
<td>User’s password and identity</td>
</tr>
<tr>
<td>RPW&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Computed masked password</td>
</tr>
<tr>
<td>E&lt;sub&gt;j&lt;/sub&gt;, D&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Symmetric key encryption/decryption algorithm</td>
</tr>
<tr>
<td>X&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Secret information known only to base station</td>
</tr>
<tr>
<td>X&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Secret information shared between user and base station</td>
</tr>
<tr>
<td>Y&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Secret random number known only to user</td>
</tr>
<tr>
<td>T&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Timestamp</td>
</tr>
<tr>
<td>A ⊕ B</td>
<td>Concatenation</td>
</tr>
<tr>
<td>A ⊕ B</td>
<td>XOR operation</td>
</tr>
<tr>
<td>h(·)</td>
<td>Hash operation</td>
</tr>
</tbody>
</table>

A. Pre-deployment phase

Before deploying any cluster head or sensor node into the field, a so-called pre-deployment or initial phase is required to register all cluster heads and sensor nodes with the BS. The BS acts like a setup server in this phase. The setup server assigns a unique identifier (ID<sub>CHj</sub>, ID<sub>j</sub>) and a randomly unique master key (MK<sub>CHj</sub>, MK<sub>j</sub>) for every cluster head (CH<sub>j</sub>) and sensor node (S<sub>j</sub>). The master key MK<sub>CHj</sub> of the CH<sub>j</sub> is known only to the BS and to the CH<sub>j</sub>, likewise the MK<sub>j</sub> of the S<sub>j</sub> is known only to the BS and to the S<sub>j</sub>. At the end, the setup server saves the information {ID<sub>CHj</sub>, MK<sub>CHj</sub>} into the memory of each cluster head and {ID<sub>j</sub>, MK<sub>j</sub>} into the memory of each sensor node.

B. Post-deployment phase

After the sensor nodes and cluster heads are deployed into the target field, the post-deployment phase starts. Each element (i.e., sensor node and cluster head) in the field, within the communication range, locates his physical neighbour. In Das et al.’s scheme [14] it is assumed that the elements in the field establish secure connection between each other. For this purpose a secure pairwise key establishment scheme is used [24]. Finally each element communicates securely with the elements in the communication range.

C. Registration phase

In this phase the user has to register to the BS. The following four steps are required. Step 1. User U<sub>i</sub> selects an identifier ID<sub>i</sub>, a password PW<sub>i</sub> and a random number y which is known only to him/her. Next he/she computes RPW<sub>i</sub> = h(y ⊕ PW<sub>i</sub>) and sends RPW<sub>i</sub> and ID<sub>i</sub> to the BS via a secure channel. Step 2. After receiving RPW<sub>i</sub> and ID<sub>i</sub>, the BS computes f<sub>i</sub> = h(ID<sub>i</sub> ⊕ X<sub>i</sub>). x = h(RPW<sub>i</sub> ⊕ X<sub>i</sub>). r<sub>i</sub> = h(y ⊕ x) and e<sub>i</sub> = f<sub>i</sub> ⊕ x. Step 3. For each m + m’ cluster heads the BS computes K<sub>j</sub> = E<sub>MK<sub>CHj</sub></sub>(ID<sub>i</sub> ⊕ ID<sub>CHj</sub> ⊕ X<sub>i</sub>). where {K<sub>j</sub>, ID<sub>CHj</sub>} ⊥ j ≤ m + m’. m is the number of initially deployed cluster heads and m’ is the number of additionally prepared cluster heads for the purpose of the dynamic node addition phase. Step 4. At the end of the registration phase the BS generates a tamper-proof smart card which stores following information: {ID<sub>i</sub>, y, X<sub>i</sub>, r<sub>i</sub>, e<sub>i</sub>, {K<sub>j</sub>, ID<sub>CHj</sub>} ⊥ j ≤ m + m’}.

D. Login phase

After registering with the base station and admission of the smart card, the user U<sub>i</sub> has to login to be able to access the real-time data from the network, i.e., from a specific cluster head CH<sub>j</sub>. The login process is as follows. Step 1. The U<sub>i</sub> inserts the SC into a card reader and inputs the password PW<sub>i</sub>’. Step 2. SC computes RPW<sub>i</sub>’ = h(y ⊕ PW<sub>i</sub>’). x’ = h(RPW<sub>i</sub>’ ⊕ X<sub>i</sub>). r<sub>i</sub>’ = h(y ⊕ x’) and verifies r<sub>i</sub> = r<sub>i</sub>’. If the verification does not hold the session is terminated. Otherwise SC computes N<sub>i</sub> = h(x’ ⊕ T<sub>i</sub>). Step 3. U<sub>i</sub> selects from which CH<sub>j</sub> he/she wants to access the real-time data and according to the selection, the SC selects the corresponding encrypted master key K<sub>j</sub> from its memory. Afterwards SC encrypts the cipher text message (ID<sub>i</sub> ⊕ ID<sub>CHj</sub> ⊕ N<sub>i</sub> ⊕ e<sub>i</sub> ⊕ T<sub>i</sub>) via a public channel to the BS.
E. Authentication phase

The following steps are required in order for the BS to authenticate the user \( U_i \). Step 1. After receiving the message from the \( U_i \), the BS finds the stored master key \( MK_{CH_j} \) of the specific cluster head \( ID_{CH_j} \) and computes \( K = E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel X_2) \). Having the encryption key \( K \), the BS can decrypt the encrypted part of the login message \( D_{SK}(E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel || N_t \parallel e_t \parallel T_2)) \) and thus check if retrieved and received \( ID_i \) and \( ID_{CH_j} \) are equal. Furthermore, the BS checks the validity of the timestamp \( T_1 \) with its current timestamp \( T_1' \). If \( |T_1 - T_1'| < \Delta T \), the BS further computes \( X = h(ID_i \parallel X_2) \), \( Y = e_t \otimes X \) and \( Z = h(Y \parallel T_1) \). At the end of the step, the BS checks if \( Z = ? N_t \). If the verification holds, the BS accepts user’s login request, otherwise the scheme terminates. Step 2. Afterwards, the BS computes \( u = h(Y \parallel T_2) \), whereby \( T_2 = \) the current timestamp of the BS. Using the master key \( MK_{CH_j} \) of the \( CH_j \) as an encryption key, the BS encrypts the message \( (ID_i \parallel ID_{CH_j} \parallel u \parallel T_2 \parallel T_2 \parallel X \parallel e_t) \) and sends following message \( (ID_i \parallel ID_{CH_j} \parallel E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel u \parallel T_2 \parallel T_2 \parallel X \parallel e_t)) \) to the corresponding cluster head \( CH_j \). Step 3. After receiving the message from the BS, the cluster head \( CH_j \), decrypts the encrypted part of the message \( D_{MK_{CH_j}}(E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel u \parallel T_2 \parallel T_2 \parallel X \parallel e_t)) \). Afterwards the \( CH_j \) checks if retrieved and received \( ID_i \) and \( ID_{CH_j} \) are equal and if \( \mid T_2 - T_2' \mid < \Delta T \), \( T_2' \) is the current timestamp of the \( CH_j \) and \( \Delta T \) is the expected time interval for the transmission delay. If all verifications hold, the \( CH_j \) continues and computes \( v = e_t \otimes X, w = h(v \parallel T_2) \) and checks if \( w = ? u \). If it holds the user \( U_i \) is authenticated by the \( CH_j \). Otherwise the scheme terminates. Finally the \( CH_j \) computes the session key as \( SK = h(ID_i \parallel ID_{CH_j} \parallel e_t \parallel T_2) \) and sends the acknowledgment to the user \( U_i \). Step 4. After receiving the acknowledgment from the \( CH_j \), the \( U_i \) can compute the session key \( SK = h(ID_i \parallel ID_{CH_j} \parallel e_t \parallel T_2) \) and thus communicate securely with the \( CH_j \).

F. Password change phase

If the user \( U_i \) wants to change the password, he can do that offline and individually. To manage the change, following steps are required. Step 1. \( U_i \) inserts the SC into a card reader and inputs the current and new password \( PW_i^{old}, PW_i^{new} \). Afterwards the SC computes \( RPW_i' = h(y \parallel PW_i^{old}) \), \( M_1 = h(RPW_i' \parallel X_2) \), \( M_2 = h(y \parallel M_1) \) and compares \( M_2 = ? r_i \). If the verification does not hold, the user inputs an incorrect password and the phase terminates. Otherwise Step 2 follows. Step 2. The SC computes \( M_2 = e_t \otimes M_1, M_2 = h(y \parallel PW_i^{new}), r_i' = h(y \parallel M_2), M_2 = h(M_2 \parallel X_2) \) and \( e_t' = M_2 \otimes M_3 \). Step 3. At the end, the SC replaces the values \( r_i \) and \( e_t \) in the memory, with \( r_i' \) and \( e_t' \).

G. Dynamic node addition phase

When an element (i.e., sensor node, cluster head) of the WSN needs to be deployed (e.g., changing a broken element or replacing a captured element) into the field after the initial pre-deployment and post-deployment phase, following is required. If a new sensor node \( S_i \) or a new cluster head \( CH_j \) is about to be additionally deployed into the deployment field, the BS assigns a unique identifier \( ID_{S_i} \) or \( ID_{CH_j} \) and a randomly generated unique master key \( MK_{S_i} \) or \( MK_{CH_j} \). The generated information is then loaded into the memory of the sensor node \( S_i \) or cluster head \( CH_j \). Afterwards the elements can be deployed in the field, whereby the user \( U_i \) gets informed by the BS about the new addition to the network.

IV. COMMENTS ON THE FLAW AND REDUNDANCY OF DAS ET AL.’S SCHEME

This section highlights the flaw of Das et al.’s scheme [14] and shows why the scheme is inappropriate for real-life environment. Additionally, we highlight redundant parts of the scheme and explain why they are redundant. The flaw and redundant parts are as follows.

In Das et al.’s scheme, at the registration phase, the user \( U_i \) selects a random number \( y \) which is known only to him. While continuing with the registration process, the user also selects an identifier \( ID_i \) and a password \( PW_i \). Afterwards, a computed masked password \( RPW_i = h(y \parallel PW_i) \) is computed, using the secret random number \( y \) and the password \( PW_i \). Next, the user \( U_i \) provides \( \{RPW_i, ID_i \} \) to the BS, whereby the secret random number \( y \) is not provided. After receiving the information the BS computes among others \( r_i = h(y \parallel x) \), whereby \( x = h(RPW_i \parallel X_2) \). However, the BS cannot derive \( r_i \), since \( y \) is a secret random number which is known only to the user \( U_i \) and it was not provided in the message \( \{RPW_i, ID_i \} \) or in any other way to the BS. Therefore we conclude that Das et al.’s scheme has a flaw, hence the BS cannot know the value of \( y \) and cannot compute the parameter \( r_i \). This flaw if further reflected in the scheme as follows. In the registration phase of Das et al.’s scheme, the BS generates a tamper-proof smart card SC with the following parameters \( \{ID_i, y, X_2, r_i, e_t, \{K_j, ID_{CH_j} \} \mid 1 \leq j \leq m + m' \} \). Since the BS does not know the value of \( y \), it cannot compute \( r_i = h(y \parallel x) \). It is therefore infeasible for the BS to generate a smart card SC which such parameters. Additionally, the fault is linked further with the login phase of Das et al.’s scheme, where the SC tries to authenticate the user \( U_i \) by checking his inputted password \( PW_i' \). The SC computes \( RPW_i' = h(y \parallel PW_i') \) or in any other way to the BS. Again this is impossible, because the random number \( y \) cannot be stored in the SC as the BS was not in possession of the secret random number \( y \) while generated information was stored on the SC. In addition to the login phase, the SC computes \( r_i' = h(y \parallel x') \), whereby \( x' = h(RPW_i' \parallel X_2) \). Afterwards it verifies whether \( r_i = ? r_i' \) in order to verify the password, thus trying to find out if the user \( U_i \) entered a correct password. Once more, this verification is infeasible, hence \( r_i' \) cannot be computed by the BS and stored into the SC as shown in the previous comments. Also \( r_i' \) cannot be computed, hence the SC cannot compute \( RPW_i' \) without the secret random number \( y \). The fault is further linked with password change phase of Das et al.’s scheme, where the smart card SC computes \( RPW_i' = h(y \parallel PW_i^{old}) \), whereby \( PW_i^{old} \) is the current password of the user \( U_i \) and was...
inputted by him/her along with a new password $PW_1^{\text{new}}$. As already presented, the $SC$ cannot compute $RPW_i^*$, since the random number $y$ cannot be stored in it, because the $BS$ which was generating the $SC$ cannot be in possession of the random number $y$. Therefore we conclude that the password change phase is infeasible.

In the authentication phase of Das et al.’s scheme, the cluster head $CH_j$ tries to validate the user $U_i$ by verifying $w = ?u$. The value $u$ is part of the encrypted message sent by the $BS$ to the cluster head $CH_j$. The $BS$ computes $u = h(Y \| T_2)$, whereby $Y = e_i \oplus h(ID_i \| X_s)$. After receiving the message $\{ID_i \| ID_{CH_j} \| E_{MK_{CH_j}}(ID_i \| ID_{CH_j} \| u \| T_1 \| T_2 \| X \| e_j)\}$ from the $BS$, the cluster head $CH_j$ computes $w = h(v \| T_2)$, whereby $v = e_j \oplus X$. As it is noticeable, the cluster head $CH_j$ computes $w$ by using values $e_i$, $X$ and $T_2$ which it got from the message of the $BS$. Furthermore, after computing $w$ it compares $w$ with $u$, which is also already contained in the message $\{ID_i \| ID_{CH_j} \| E_{MK_{CH_j}}(ID_i \| ID_{CH_j} \| u \| T_1 \| T_2 \| X \| e_j)\}$. We conclude that this part of authentication phase of Das et al.’s scheme is redundant, since it is an unnecessary verification of parameters which are all in the same encryption message.

V. PROPOSED IMPROVED SCHEME

This section proposes an improved dynamic password-based user authentication scheme for HWSNs to overcome the flaw and redundancy of Das et al.’s scheme [14]. We will not describe the pre-deployment, the post-deployment and the dynamic node addition phases of the proposed scheme since they are same as in Das et al.’s scheme. An overview of the scheme is depicted in Fig. 1.

A. Registration phase

The user authentication starts with the registration phase. For the user $U_i$ to register successfully the following steps are required. Step 1. The user $U_i$ selects an identifier $ID_i$, a password $PW_i$ and a random number $y$ which is known only to him/her. Using the random number $y$, the $U_i$ computes a masked password $RPW_i^* = h(y \| PW_i^*)$ and then sends the values $ID_i$ and $RPW_i^*$ to the $BS$ via a secure channel. Step 2. After receiving $ID_i$ and $RPW_i^*$, the $BS$ computes $f_i = h(ID_i \| X_s)$ using the secret information $X_s$ which is known only to the $BS$. Using the secret information $X_s$ which is shared between the $U_i$ and the $BS$, the $BS$ computes $x = h(RPW_i^* \| X_A)$ and $e_i = f_i \oplus x$. Step 3. For every initially deployed cluster head in the network ($CH_1, CH_2, ..., CH_m$), the $BS$ computes a combination $\{(K_j, ID_{CH_j}) | 1 \leq j \leq m\}$. $K_j$ is an encryption key computed as $K_j = E_{MK_{CH_j}}(ID_i \| ID_{CH_j} \| X_s)$, using the master key $MK_{CH_j}$ of a specific $CH_j$. Additionally, the $BS$ computes another $m'$ number of the combinations, whereby $m'$ is the number of additionally prepared $CH$s for the option of dynamic addition of $CH$s. If afterwards a new cluster head ($CH_{m+1}$) is being deployed into the field, a user can already use the pre-computed combination ($K_{m+1}, ID_{CH_{m+1}}$) for the authentication process. Step 4. In the final step, the $BS$ generates a smart card $SC$ containing $\{ID_i, X_A, e_i, f_i, \{ (K_j, ID_{CH_j}) | 1 \leq j \leq m + m'\}\}$. Step 5. Finally, the user $U_i$ adds the random number $y$ into the smart card $SC$ and thus ends the registration phase.

B. Login phase

In the login phase, following steps are performed. Step 1. The user $U_i$ inserts his smart card $SC$ into a card reader and inputs his password $PW_i^*$. Step 2. The $SC$ computes the masked password $RPW_i^* = h(y \| PW_i^*)$, using the inputted password and the random number $y$ which is already stored by the user $U_i$ in the $SC$. Next, $x = h(RPW_i^* \| X_A)$ is computed using the stored secret information $X_A$ and $RPW_i^*$. In addition, the $SC$ computes the initial $x = e_i \oplus f_i$ using the stored information $e_i$ and $f_i$ and compares $x = ?x'$. If the verification does not hold, the user $U_i$ has inputted an incorrect password and the scheme terminates. If the above verification holds, the $BS$ computes $N_i = h(x' \| T_2)$, where $T_1$ is the system’s timestamp. Step 3. For the user $U_i$ to access real-time data from a specific cluster head, he/she needs to select the appropriate cluster head $CH_j$, his identifier $ID_{CH_j}$ and the associated encryption key $K_j$ from the list of all combinations already saved in the $SC$. Then the $SC$ uses the $K_j$ to encrypt a message $E_{K_j}(ID_i \| ID_{CH_j} \| N_i \| e_i \| T_1)$. Finally, the user $U_i$ sends the message $\{ ID_i \| ID_{CH_j} \| E_{K_j}(ID_i \| ID_{CH_j} \| N_i \| e_i \| T_1) \}$ to the $BS$ via a public channel.

C. Authentication phase

After the login phase, the $BS$ has to authenticate the user $U_i$ and so enable him to compute the session key $SK$ and thus communicate securely with the cluster head $CH_j$. Following steps are required for a successful authentication. Step 1. After receiving the login request message from the user $U_i$, the $BS$ firstly has to compute the encryption key in order to read the encrypted part of the login request message. This is done using the $BS$’s secret information $X_s$ and the values $ID_i$ and $ID_{CH_j}$. Knowing the $ID_{CH_j}$, the $BS$ can find the associated master key $MK_{CH_j}$ of the appropriate cluster head $CH_j$. Thus the $BS$ computes $K = E_{MK_{CH_j}}(ID_i \| ID_{CH_j} \| X_s)$. Having $K$, the $BS$ can now decrypt the encrypted part of the login request message $D_{K}(E_{K_j}(N_i \| e_i \| T_1))$. Step 2. When the login request message is decrypted successfully, the $BS$ uses the current timestamp $T_1^*$ and checks if $|T_1^* - T_1| < \Delta T_1$, whereby $\Delta T_1$ is the expected interval for the transmission delay. Furthermore, the $BS$ checks if retrieved and received $ID_i$ and $ID_{CH_j}$ are equal. If the verifications do not hold, the scheme terminates. Otherwise, the $BS$ computes $Z = h(e_i \oplus h(ID_i \| X_s) \| T_3)$, using $e_i, ID_i$ and $T_3$ from the login request message and its secret information $X_s$. Afterwards, the $BS$ verifies $Z = ?N_i$. If the verification does not hold, the scheme terminates. Otherwise, the $BS$ acknowledges user $U_i$ as a valid user and proceeds as follows. Step 3. Using the master key $MK_{CH_j}$ of the appropriate cluster head $CH_j$ as an encryption key, the $BS$ encrypts the message $E_{MK_{CH_j}}(ID_i \| ID_{CH_j} \| T_2 \| T_2 \| e_i)$, whereby $T_2$ is the current timestamp of the system. After the message is constructed, the $BS$ sends it to the appropriate cluster head $CH_j$ via a public channel.
Fig. 1. Depiction of the proposed scheme through the phases and according to the communication between the actors of the WSN.

Step 4. After receiving the message \( \{ID_i \parallel ID_{CH_j} \parallel E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel T_1 \parallel T_2 \parallel e_i) \} \) from the BS, the \( CH_j \) uses its master key \( MK_{CH_j} \) to decrypt the message \( D_{MK_{CH_j}}(E_{MK_{CH_j}}(ID_i \parallel ID_{CH_j} \parallel T_1 \parallel T_2 \parallel e_i)) \). It then checks if \( |T_2 - T_2^*| < \Delta T_2 \), whereby \( T_2^* \) is the current timestamp of the \( CH_j \) and \( \Delta T_2 \) is the expected interval for the transmission delay. The BS also checks if retrieved and received \( ID_i \) and \( ID_{CH_j} \) are equal. If the verifications hold, the \( CH_j \) computes the session key \( SK = h(ID_i \parallel ID_{CH_j} \parallel e_i \parallel T_2) \) with the values he/she is already in possession of. Using the secret session key \( SK \), the user \( U_i \) can now communicate and securely access real-time data from the cluster head \( CH_j \).

D. Password change phase

For a user \( U_i \) to change his/her password, no connection with the BS or any of the cluster heads \( CH_j \) is needed. The phase can be done offline, using only the SC. The phase contains the following steps. Step 1. The user \( U_i \) inserts his/her \( SC \) into a card reader and inputs his/her old and new password \( PW_i^{old}, PW_i^{new} \). Step 2. Using the random number \( y \) stored in the SC and the old password \( PW_i^{old} \), the SC computes the masked password \( RPW_i^{old} = h(y \parallel PW_i^{old}) \). Additionally, using the secret information \( X_A \), the SC computes the masked password \( RPW_i^{new} = h(RPW_i^{old} \parallel X_A) \). The SC then...
computes the initial $x = e_i \oplus f_i$ using the stored values $e_i$ and $f_i$ and compares $x = ? M_i$. If the verification does not hold, the user $U_i$ has inputted an incorrect password and the scheme terminates. Step 3. Otherwise, the SC starts the process of replacing the password, whereby the new masked password is computed $\text{RP}W_i^{\text{new}} = h(y \perp \text{PW}_i^{\text{new}})$ using the new inputted password $\text{PW}_i^{\text{new}}$. Furthermore, the SC computes $M_2' = h(\text{RP}W_i^{\text{new}} \perp X_3)$. Finally the new value $e_i' = f_i \oplus M_2'$ is computed and replaced with the stored $e_i$ in the memory of the SC.

Our proposed protocol not only eliminates the flaw but is also less computationally costly and thus more appropriate for use. In Section VII we demonstrate the advantages of our proposed protocol in comparison to competitive ones.

VI. SECURITY ANALYSIS AND DISCUSSION OF THE PROPOSED SCHEME

This section provides discussion and a security analysis of the proposed scheme. We show that our scheme overcomes all the practical flaws of Das et al.’s scheme and is equally resilient to various possible attacks.

A. Flaw correction

In our scheme, the BS, at the time of the registration, does not compute and store $r_j$ into the smart card SC, but stores $f_j$, which is afterwards used as a verification token in the login and password change phase. Furthermore, the user $U_i$ stores the secret random number $y$ into the SC, hence neither the BS, nor any of CHs can know the value of $y$. In the authentication phase of our scheme, before sending a message to the CH, the BS does not encrypt the parameters $u$ and $X$ into the message, since they would be only needed if CH would verify $w = ? u$, whereby $w = h((e_i \oplus X) \perp T_2)$. We already described in the comments of Section IV that this verification is redundant or an unnecessary verification of parameters which would all be in the same encryption message. Therefore $u$ and $X$ are removed along with the CH’s computation of $w$ and $v$.

B. Replay attack

Even if an attacker would intercept a login request message $\{ ID_1 \perp ID_{CH_j} \perp E_{K_j}(ID_1 \perp ID_{CH_j} \perp N_i \perp e_i \perp T_1) \}$ and try to replay it to the BS, the BS checks the freshness of the timestamps verifying $|T_1 - T_1^*| < \Delta T_1$. Moreover, even if the verification would hold and the BS would authenticate the attacker, he/she cannot compute the session key $SK$, since he/she cannot know the values $e_i$ and $T_1$, which are encrypted with the encryption key $K$. Hence our scheme is resilient against replay attacks.

C. Gateway node bypassing attack

The proposed scheme can withstand the gateway node bypassing attack, because the user $U_i$ and a cluster head $CH_j$ need to establish a session key $SK$ in order to communicate (request - response) securely. To compute the $SK$ the user $U_i$ needs to register and authenticate with the BS.

D. Stolen-verifier attack

Stolen-verifier attack is an attack where an adversary steals a user’s verifier from a server and tries to impersonate a legitimate user with the stolen verifier. Our proposed scheme is free from password tables or any kind of verifier tables, hence neither the BS nor the CHs keep them to authenticate the users, thus making the scheme resilient against stolen-verifier attack.

E. Impersonation attack

In an impersonation attack an adversary tries to impersonate a legitimate user by forging a valid login request. The adversary could forge the login request using some eavesdropped message or using information stored on the smart card. Since in our scheme the messages are encrypted, the adversary cannot forge a valid request message without knowing the encryption key $K$. Since the encryption key is computed using the secret information $X_3$ which is known only to the BS it is impossible for an adversary to compute it. Furthermore, even if the adversary would steal a valid smart card and derive the encryption key $K_j$ from it, he/she would not be able to compute $N_i$ without the password $\text{PW}_i$ of the user $U_i$. Without $N_i$ the adversary cannot forge a valid login request, thus making the scheme resilient against the impersonation attack.

F. Offline and online password guessing attack

In an offline password guessing attack [25] an adversary eavesdrops the communication over a public channel between a legitimate user and the authentication server. He/She then uses the eavesdropped message and tries to generate a valid password by using a brute force or dictionary attack and compare it with the message from the legitimate user. For an adversary to use an offline password guessing attack in our proposed scheme, he/she would first need to have a user’s smart card. Moreover, messages sent over a public channel are in our proposed scheme protected using an symmetric encryption key $K$. Additionally, the password $\text{PW}_i$ in our proposed scheme can only be found hidden inside the parameters $e_i = f_i \oplus h(y \perp \text{PW}_i) \perp X_3$ and $N_i = h(h(y \perp \text{PW}_i) \perp X_3) \perp T_1$, thus making it computationally infeasible to extract it due to the one-way property of the hash function. In an online password guessing attack an adversary tries to find a valid password by attempting to login or authenticate online. To use this attack the adversary would need to have a valid smart card, whereby it is assumed that the smart card itself would block the password guessing after multiple wrong password inputs. Therefore, we can conclude that our proposed scheme is resilient against offline or online password guessing attacks.

G. Smart card breach attack

Although we assume that a smart card is tamper-resistant and cannot be breached, we consider a scenario where a legitimate user’s smart card is being stolen or lost and eventually found and somehow cracked by an adversary. This would mean that the adversary obtained the information $\{y, ID, X_E,e_i,f_i,\{ (K_f, ID_{CH_j}) \mid 1 \leq j \leq m \}$ from the SC. Fortunately, the adversary cannot use the retrieved information from the SC to impersonate a legitimate user $U_i$, since he/she needs to know the $U_i$’s password $\text{PW}_i$ in order to successfully accomplish an authentication. Furthermore, there is no feasible way for an adversary to obtain the $\text{PW}_i$ from $e_i = f_i \oplus h(h(y \perp \text{PW}_i) \perp \text{X}_E)$. 

114
$X_A$) due to the one-way property of a hash function. There is also no feasible way for an adversary to crack the encryption key $K$, since he/she does not know the secret information $X_S$, which is known only to the BS. Therefore we can conclude that the scheme is resilient against smart card breach attack.

H. Password change attack

If a legitimate user $U_i$ wants to change the password $PW_i$, he/she can accomplish that offline, using its smart card SC and without contacting the BS, as described in the Password change phase of the proposed scheme. Thus for an adversary to change the password of the user $U_i$, he/she needs to be in the possession of $U_i$’s smart card. Furthermore, if an adversary could come into the possession of a legitimate user’s smart card, he/she would need to know the old password $PW_i^{old}$ of the $U_i$ in order to change it. As we already described earlier, our scheme can withstand the smart card breach attack, thus making it impossible for an attacker to accomplish a password change attack.

I. Many logged-in users with the same login-id attack

As Das et al.’s scheme can withstand the many logged-in users with the same login-in attack, also can our proposed scheme. For a user to be able to login, he/she needs his smart card SC, whereby every SC has a random number $y$ stored in it. So even if two or more users have the same login credentials $(ID_i, PW_i)$, their computed masked password $RPW_i = h(y \parallel PW_i)$ will be different.

J. Privileged insider attack

For a privileged-insider attack [26] to take place, a privileged person who can access a server (e.g., administrator or system manager), could use his/her privileges to obtain a password of a user (e.g., from a password table or from a login request message) and then try to impersonate the same user on some other server, where the user could also be registered. In our case the server is the BS, but as already described in Stolen-verifier and password guessing attack, our scheme is free from password tables, thus making it impossible for anyone to obtain a password from the BS. Furthermore, even if a privileged insider of the BS would monitor the login request from the user, he/she cannot obtain his/her password, since he/she does not send the password $PW_i$ in plaintext, but rather in form of a computed masked password $RPW_i = h(y \parallel PW_i)$. Because the password is concatenated with the random number $y$ and hashed with a one-way hash function it is computationally infeasible for an adversary to obtain the password, thus making the scheme secure against the privileged-insider attack.

K. DoS attack

Denial-of-service attack [27] is useless against our proposed scheme since acknowledgement about a successful authentication from the BS is being sent over the CH to the user $U_i$.

VII. PERFORMANCE ANALYSIS AND COMPARISON

This section summarizes the performance and functionality of our proposed scheme and compares it with some recent and related user authentication schemes for WSN. In Table II we compare the functionality of our proposed scheme with other related schemes. The comparison demonstrates that our scheme can achieve the same functionalities as Das et al.’s scheme, therefore much more than other schemes.

| Function- | Our pro- | [14] | [16] | [17] | [13] | [12] | [18] |
|alleles | posed protocol | | | | | | |
| Supports | password change | Yes | Yes | Yes | Yes | No | No |
| Mutual authentication | Yes | Yes | No | No | Yes | Yes | Yes |
| Resilient against DoS attack | Yes | Yes | No | No | Yes | Yes | No |
| Resilient against node capture attack | Yes | Yes | No | No | Yes | No | No |
| Session key between user and node | Yes | Yes | No | No | Yes | No | No |
| Supports dynamic node addition | Yes | Yes | No | No | Yes | No | No |

Furthermore, Table III shows the computational-cost comparison of our scheme and other schemes. We summarized only the registration, login and authentication phases, since they are the important ones for user authentication. It can be seen that our scheme requires less computations (i.e., four hashing operations less) than Das et al.’s scheme, whereby both ours and Das et al.’s are more computational-costly than other schemes. However, additional encryption/decryption operations in our and Das et al.’s scheme are worth the additional functionalities which are derived.

<table>
<thead>
<tr>
<th>Authentication scheme</th>
<th>Registration phase</th>
<th>Login + authentication phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our proposed scheme</td>
<td>$3T_h + (m + m')T_E$</td>
<td>$7T_h + 3T_E + 2T_D$</td>
</tr>
<tr>
<td>Das, Sharma [14]</td>
<td>$4T_h + (m + m')T_E$</td>
<td>$10T_h + 3T_E + 2T_D$</td>
</tr>
<tr>
<td>Huang, Chang [16]</td>
<td>$4T_h$</td>
<td>$11T_h$</td>
</tr>
<tr>
<td>He, Gao [17]</td>
<td>$6T_h$</td>
<td>$11T_h$</td>
</tr>
<tr>
<td>Vaidya, Makrakis [13]</td>
<td>$5T_h$</td>
<td>$13T_h$</td>
</tr>
<tr>
<td>Fan, Ping [12]</td>
<td>$6T_h$</td>
<td>$19T_h$</td>
</tr>
<tr>
<td>Chen and Shih [28]</td>
<td>$3T_h$</td>
<td>$10T_h$</td>
</tr>
</tbody>
</table>

VIII. CONCLUSIONS

This paper shows that because of a serious flaw, Das et al.’s dynamic password-based user authentication scheme for HWSN is inappropriate for implementation in real-life environment. Moreover, we present some redundant parts
which unnecessarily slow the scheme down. To overcome the flaw and redundancy of Das et al.’s scheme, we proposed an enhanced dynamic password-based user authentication scheme for HWSN to remedy these practical weaknesses. The scheme which we proposed satisfies all the requirements for a HSWN user authentication scheme and is robust for a real-life environment. Furthermore, it can also withstand various attacks without losing any functionality of Das et al.’s scheme, thus retains its advantages and is less computationally-costly (i.e., four hashing operations less (∇Tq)).

REFERENCES


