E-SAS: Enhanced Secure Authentication System for Healthcare Applications using Wireless Medical Sensor Networks

Sang Guun Yoo
Departamento de Ciencias de la Computación
Universidad de las Fuerzas Armadas ESPE
Av. Gral. Rumiñahui s/n, Sangolquí
Ecuador
yysang@espe.edu.ec

Abstract: - One of the fields where wireless sensor networks are being widely applied is the area body sensor networks for healthcare applications. As healthcare applications manage private data of patients, the security becomes one of the most critical requirements to consider. In this aspect, there have been several researches delivering security mechanisms for healthcare services. However, Yoo analyzed several of those approaches and observed that they includes serious security flaws: the scheme proposed for Khan and Kumari presents vulnerabilities to data leakage, man in the middle attack, password guessing attack, and manages insecure keys, while Shin et al.'s scheme includes vulnerabilities to data leakage, password guessing attack, replay attack, and manages insecure keys. To overcome the mentioned security issues, this work proposes an enhanced secure user authentication system for health applications. The presented scheme solves the identified security flaws and excels in terms of performance and efficiency.

Key-Words: - Sensor network, healthcare service, user authentication, two factor authentication, wireless body area network, cryptanalysis, BAN logic, IoT.

1 Introduction

Wireless sensor networks are being applied in different fields [1-5]. One of those fields is the body sensor networks [6]. Body sensor networks is a type of wireless sensor networks with higher sensitivity because the sensors are put on or in user's body. The body sensor network can measure the heart rate, blood pressure, and many other signals, allowing the implementation of different kind of services in the field of healthcare. Due to the importance and criticality of the healthcare applications, body sensor networks must be compliant to the different requirements of security. In this aspect, there have been several works delivering security mechanisms to the wireless sensor networks, such as [7, 8], meanwhile, others have proposed specific security schemes for healthcare services [9-11].

In 2012, Kumar et al. [9] proposed an efficient-strong authentication protocol for healthcare application using wireless medical sensor networks. Later, Khan and Kumari [10] presented their work where they recognized security problems in Kumar et al.'s work and suggested their own authentication scheme using a smart card (two-factor user authentication). After publication of Khan and Kumari proposal, Shin et al. [11] indicated that such mechanism was vulnerable to other attacks and they proposed a new secure authentication system. However, one of our previous work [12] identified that the scheme of Khan-Kumari and Shin et al. are still flawed with a number of security pitfalls. Khan-Kumari scheme is exposed to data leakage, man in the middle attack, password guessing attack, and manages insecure keys, while Shin et al.'s scheme includes vulnerabilities to data leakage, password guessing attack, replay attack, and manages insecure keys. In this situation, this paper continues our contributions made in our previous work [12] and proposes an enhanced user authentication mechanism for body sensor networks which fixes all the aforementioned weaknesses and excels in terms of performance and efficiency.

The rest of paper is organized as follows. Section 2 explains briefly the existing works and details the cryptanalysis of such schemes. Later, Section 3 presents the proposed protocol which solves the vulnerabilities and limitations mentioned in Section 2. Next, Section 4 analyses the proposed protocol in terms of security and performance. Finally, Section 5 concludes this paper.

2 Review of Previous Works

This section reviews briefly schemes proposed by Kumar et al. [9], Khan and Kumari [10], and Shin et al. [11]. This section also reviews our previous work which cryptanalyses the mentioned schemes.

2.1 Review of Kumar et al.'s Scheme

The gateway node *GW* initially chooses three secret keys denoted as *J*, *K*, and *Q* and distributes a secret key $K_{gs}=h(Q||ID_g)$ with all sensor nodes, where ID_g is the identity of *GW*, || is data concatenation, and *h(.)* is a hash function. The proposed scheme is composed of the following phses.

User Registration Phase. The medical professional, which is called as user in this paper, registers him/herself to *GW* by choosing his/her identification ID_i and password PW_i and submitting them to the *GW*, using a secure channel. Then, *GW* computes $C_{ig} = E_J[ID_i||ID_g]$ and $N_i= h(ID_i \oplus PW_i \oplus K)$ where $E_{key}[message]$ is an symmetric encryption and $\oplus$ is the XOR operation. Finally, *GW* delivers a smart card with $\{h(.), C_{ig}, N_i, K\}$ to the user.

Patient Registration Phase. The patient delivers his/her information to the registration center, chooses the sensor kit to use, and selects the medical professional who will monitor his/her physiological data. After patient registration, the registration center sends the information about the recently registered patient to the user.

Login and Authentication Phase. This phase is executed when the users need to check the physiological data of patients. The user inserts the smart card and enters the ID_i and PW_i. The smart card the computes $N_i^*=h(ID_i \oplus PW_i \oplus K)$ and compares N_i^* with N_i to validate the user. If N_i^* and N_i are equal, the smart card calculates $h(ID_i)$ and $CID_i=E_K[h(ID_i)||M||Sn||C_{ig}||T']$ and sends $\{CID_i, T'\}$ to *GW*, where *M* is a random nonce, *Sn* is the sensor node, and *T'* is the current timestamp. Once received the message, the *GW* checks *T'* verifying that $(T''-T') \leq \Delta T$, where *T''* is the current time of *GW* and ΔT is the time interval for the transmission delay. Once verified *T'*, *GW* executes $D_K[CID_i]=(h(ID_i)^*||M||Sn||C_{ig}||T'^*)$ and $D_J[C_{ig}]= (ID_i^*||ID_g^*)$ to get CID_i and C_{ig}, then, compares $h(ID_i)^*$ with $h(ID_i^*)$, ID_g^* with ID_g and T' with T'^*. Once verified the values, the *GW* computes $A_i=E_{SKgs}[ID_i||Sn||M||T'''||T']$ and sends $\{A_i, T'''\}$ to the medical sensor that the user wants to access, where T''' is the current timestamp of *GW*. Upon receiving *GW*'s message, the sensor node validates T''', gets A_i by executing $D_{SKgs}[A_i]=(ID_i^*||Sn^*||M^*||T'''^*||T')$, and checks the validity of *GW* by comparing Sn^* with Sn and T''' with T'''^*. Once validated *GW*, the sensor computes $SK=h(ID_i^*||Sn||M^*||T')$ and $L=E_{SK}[Sn||M^*||T^*]$, and sends $\{L, T^*\}$ to the user, where T^* is sensor's current timestamp T^* and *SK* is the session key. Upon receiving the sensor node's response, the user's system validates the T^* and computes $SK=h(ID_i||Sn||M||T')$. Finally, the user decrypts *L* by executing $D_{SK}(L)=(Sn||M^*||T^*)$ and authenticates the sensor node by comparing Sn^* with Sn and M^* with *M*.

Password Change Phase. The user inserts the smart card and inputs the ID_i and PW_i. The smart card then computes $N_i^*= h(ID_i \oplus PW_i \oplus K)$ and compares N_i^* with N_i to verify the validity of the user. Once validated the user, he/she inputs the new password PW_{inew} and the smart card calculates $N_{inew}=h(ID_i \oplus PW_{inew} \oplus K)$ and replaces the old N_i with the N_{inew}.

2.2 Review of Khan-Kumari Scheme

In [10], the authors proposed their own protocol after analysing the vulnerabilities and limitations of Kumar et al.'s scheme. The authors indicate that Kumar et al.'s scheme has vulnerabilities in user impersonation, password guessing attack, sensor node impersonation attack, and lacks of mutual authentication.

In the protocol proposed by Khan and Kumari, the gateway *GW* chooses a secret key *K* and distributes a secret key $K_{gs}=h(K||ID_g)$ with all sensor nodes, where ID_g is *GW*'s identity and *h(,)* is a hash function. The Khan-Kumari scheme is composed of the following phases.

User Registration Phase. The user *U* registers him/herself to the gateway node *GW* by choosing his/her identification ID_u and sending it to the *GW* using a secure channel. Then, *GW* calculates $C_{ug}=E_K(ID_u||ID_g)$, $K_u=h(K||ID_u||ID_g)$, and $K_g=h(ID_g||K)$, where ID_g is the identification of *GW* and *K* is a secret key of *GW*. Later, the *GW* stores $\{h(.), C_{ug}\}$ into a smart card and delivers it with $\{K_u, K_g\}$ to *U* using the secure channel. Once obtained the smart card and $\{K_u, K_g\}$, *U* chooses his/her password PW_u and computes $N_u=h(ID_u||PW_u||K_u)$, $PK_u=K_u \oplus (ID_u||PW_u)$, and $PK_g=K_g \oplus (PW_u||ID_u)$. Finally, *U* stores N_u, PK_u, and PK_g into the smart card.

Patient Registration Phase. The steps executed in this phase are equal to the one proposed by Kumar et al. [9].

Login and Authentication Phase. *U* inserts his/her smart card and inputs his/her ID_u and PW_u. Then, the smart card calculates $K_u=P_{Ku} \oplus (ID_u||PW_u)$, $K_g=P_{Kg} \oplus (PW_u||ID_u)$, and $N_u^*=h(ID_u||PW_u||K_u)$ and verifies the validity of U by comparing N_u* with N_u. Once validated the user, the smart card computes $C_{u1}=C_{ug}\oplus h(K_g)$ and $CID_u=E_{Ku}(h(ID_u)||M||S_n||C_{ug}||T_u)$ where M is a random nonce, T_u is the current timestamp of U, and Sn is the sensor node which the user wants to access. After, U sends $\{CID_u, C_{u1}, T_u\}$ to GW. Once received the authentication request from U, GW validates the freshness of T_u, decrypts CID_U and C_{ug} by executing $D_K(CID_u)=(h(ID_u)^*||M||S_n||C_{ug}^*||T_u^*)$ and $D_K(C_{ug})=(ID_u*||ID_g*)$, and then compares C_{ug}^* with C_{ug}, $h(ID_u)*$ with $h(ID_u^*)$, ID_g* with ID_g, and T_u with T_u* to verifiy the authenticity of U. Once validated U, GW computes $C_{g1}=K_g\oplus(M||T_g||T_u)$ and sends $\{C_{g1}, T_g\}$ to U where T_g is the current timestamp of GW. Additionally, GW computes $C_{g2}=h(K_{gs})\oplus(CID_u||T_g||M||T_u)$ and $A_u=h(CID_u||K_{gs}||T_g||S_n||T_{gs})$, and sends $\{C_{g2}, A_u, T_{gs}\}$ to the sensor node where T_{gs} is the current timestamp. On receiving $\{C_{g1}, T_g\}$ from GW, U verifies the legitimacy of GW by checking the freshness of T_g and comparing $M*$ with M, T_g* with T_g, and T_u* with T_u; values for comparison are obtained from $(M*||T_g*||T_u*)=C_{g1}\oplus K_g$. After this mutual authentication, U and GW compute their session key $K_{sessU\text{-}GW}=h(M||ID_u||T_g)$. On the other hand, on receiving $\{C_{g2}, A_u, T_{gs}\}$ from GW, the sensor node verifies the legitimacy of GW by checking the freshness of T_{gs} and comparing A_u* with A_u; values for comparison are obtained from $(CID_u*||T_g*||M*||T_u*)=C_{g2}\oplus h(K_{gs})$ and $A_u*=h(CID_u*||K_{gs}||T_g*||S_n||T_{gs})$. Once verified the validity of GW, the sensor node gets its current timestamp T_{sg} and computes $C_{s1}=h(T_g||K_{gs}||T_{sg})\oplus h(CID_u||S_n)$, and sends $\{C_{s1}, T_{sg}\}$ to the GW. It also computes $C_{s2}*=h(S_n||M*||T_u*||T_s)$ and sends $\{C_{s2}*, T_s\}$ to U where T_s is another timestamp of the sensor note. On receiving $\{C_{s1}, T_{sg}\}$ from the sensor node, GW checks the freshness of T_{sg} and verifies the legitimacy of the sensor node by comparing $(h(CID_u||Sn))*$ with $h(CID_u||Sn)$; value for comparison is obtained from $(h(CID_u||S_n))*=C_{s1}\oplus h(T_g||K_{gs}||T_{sg})$. After this mutual authentication, GW and sensor node compute their session key $K_{sessGW\text{-}Sn}=h(K_{gs}||T_{sg}||M)$. Finally, on receiving $\{C_{s2}*, T_s\}$ from sensor node, U checks the freshness for T_s and verifies the authenticity of sensor node by calculating $C_{s2}=h(S_n||M||T_u||T_s)$ and comparing $C_{s2}*$ with C_{s2}. After this mutual authentication, U and sensor node compute their session key $K_{sessU\text{-}Sn}=h(M||T_s||S_n)$.

Password Change Phase. When U wants to change the password, he/she insert the smart card and input ID_u and PW_u. Then, the smart card retrieves $K_u=PK_u\oplus(ID_u||PW_u)$, $K_g=PK_g\oplus(PW_u||ID_u)$, calculates $N_u*= h(ID_u||PW_u||K_u)$, and compares N_i* with N_i to validate the authenticity of U. After validation, U inputs his/her new password $(PW_u)_{new}$ and the smart card computes $(N_u)_{new}=h(ID_u||(PW_u)_{new}||K_u)$, $(PK_u)_{new}=K_u\oplus(ID_u||(PK_u)_{new})$ and $(PK_g)_{new}=K_g\oplus((PW_u)_{new}||ID_u)$, and replaces old values with the new ones.

2.3 Cryptanalysis of Khan-Kumari Scheme

One of our previous work [12] has cryptanalyzed the Khan-Kumari scheme based on same the assumptions of [10]. The assumptions indicate that the adversary U_a can obtain a valid smart card to extract values inside it i.e. C_{ug}, N_u, PK_u, PK_g using methods like [13, 14]. However, the GW's secret key K cannot be leaked by any mode. The vulnerability analysis of Khan-Kumari Scheme has finished with the detection of the following security flaws.

Data Leakage Vulnerability. In Login and Authentication phase, GW responds to U's login request with $\{C_{g1}, T_g\}$ where C_{g1} is calculated as $K_g\oplus(M||T_g||T_u)$. Since the number of bits of K_g is less than the number of bits of $M||T_g||T_u$, M is exposed to U_a. Here an example: in the performance analysis section of Khan-Kumari scheme, the authors assumes 128 bits for every data (i.e. random numbers, result of hash functions, timestamps, ID_U, PW_U, and so on.); in such case, $M||T_g||T_u$ will have 384 bits while K_g will have 128 bits. In such conditions, the result of $C_{g1}=K_g\oplus(M||T_g||T_u)$ will contain the plaintext value of M and T_g, exposing the confidential value of M to the attacker. Similar situation occurs for $C_{g2}=h(K_{gs})\oplus(CID_u||T_g||M||T_u)$ where CID_U, T_g, and M will be transmitted in plaintext (see Figure 1).

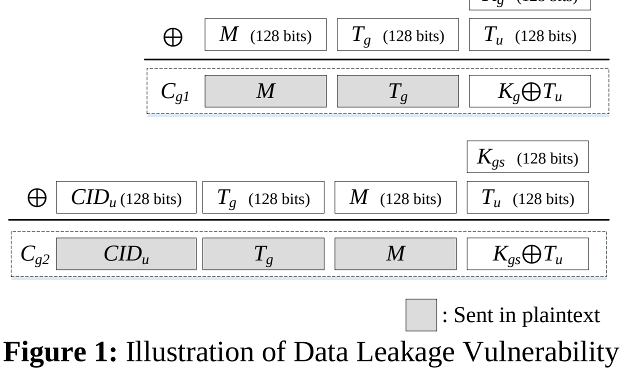

Figure 1: Illustration of Data Leakage Vulnerability in Khan-Kumari Scheme

Man in the middle attack. An adversary U_a can pretend to be a valid GW by executing a man in the middle attack. First, U_a intercepts the message $\{CID_u, C_{u1}, T_u\}$ and resends it to the GW. The GW validates the authenticity of the messages and returns the $\{C_{g1}, T_g\}$ message. U_a also intercepts this message and extracts the M value taking advantage of the data leakage vulnerability as explained before (see Figure 1). Once obtained M, U_a calculates $C_{g1\_a}=K_{g\_a}\oplus(M||T_{g\_a}||Tu)$ and sends $\{C_{g1\_a}, T_{g\_a}\}$ to U, where K_{g_a} is any value with same length of K_g and T_{g_a} is the current timestamp of U_a. Clearly, this message will be successfully authenticated by U because (1) T_{g_a} is fresh and (2) because the comparison between M^* and M will correct (see Figure 2). Now, U_a can pretend to be the correct GW.

This attack becomes more effective if it were executed by a valid user with an authentic smart card because he/she could use K_g instead of K_{g_a}. The valid user can obtain K_g by extracting PK_g from the smart card and calculating $K_g=PK_g\oplus(PW_u||ID_u)$.

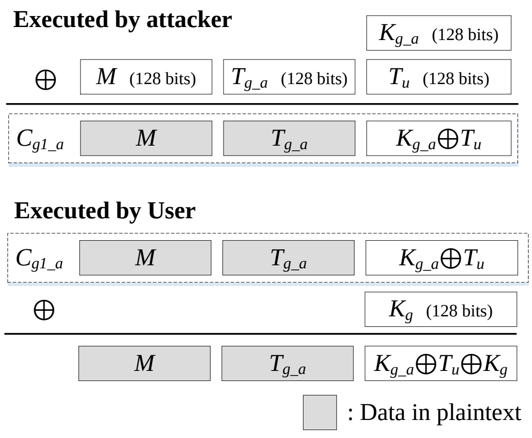

Figure 2: Illustration of man in the middle attack in Khan-Kumari Scheme

Password Guessing Attack. An internal user U_1 with a valid smart card can easily guess the ID_u and PW_u of another user with the following steps. First, the malicious internal user U_1 extracts PK_g from his/her smart card and obtains K_g by calculating $K_g=PK_g\oplus(PW_{u1}||ID_{u1})$ where ID_{u1} and PW_{u1} are the identity and password of U_1. Once obtained K_g, U_1 can guess the ID_u and PW_u of any other user U by extracting PK_g from U's smart card and calculating $(PW_u||ID_u)=PK_g\oplus K_g$.

Insecure K_g and K_{gs}. Most of the security of the Khan-Kumari scheme is based on K_g and K_{gs}. However, those values are shared by all users and sensor nodes. Therefore, if an internal user becomes malicious, K_g could leak allowing any adversary to execute the man in the middle attack to spoof the GW. On the other hand, if an attacker U_a succeeded to discover the K_{gs} value (e.g. using brute force attack) all sensor nodes would be exposed allowing U_a to execute sensor node spoofing attacks. It is important to say that, as all sensor nodes shares the same K_{gs}, the adversary would be more motivated to execute a brute force attack to get such secret value.

2.4 Review of Shin et al.'s Scheme

In the protocol proposed in [11], the gateway node GW initially chooses a secret key K and shares a secret key $K_{gs}=h(K||ID_g)$ with all deployed sensor nodes, where ID_g is the identity of GW. Once executed this initial process, it is possible to execute the following four phases: User registration, patient registration, login and authentication, and password change phases.

User Registration Phase. In this phase, the user U registers him/herself to the gateway node GW. The user chooses his/her identity ID_u and sends it to GW using a secure channel. GW then computes $C_{ug}=E_K(ID_u||ID_g)$, $K_u=h(K||ID_u||ID_g)$, and $K_g=h(ID_g||K)$, where ID_g is the identity of GW and K is the secret key of GW. Later, the GW stores $\{h(.), C_{ug}\}$ into a smart card and delivers it with $\{K_u, K_g\}$ to U using the secure channel. Once obtained the smart card and $\{K_u, K_g\}$, U chooses his/her password PW_u and computes $N_u=h(ID_u||PW_u||K_u)$, $PK_u=K_u\oplus(ID_u||PW_u)$, and $PK_g=K_g\oplus(PW_u||ID_u)$. Finally, U stores N_u, PK_u, and PK_g into the smart card.

Patient Registration Phase. The steps executed in this phase are equal to the one proposed by Kumar et al. [9].

Login and Authentication Phase. U inserts his/her smart card and inputs his/her ID_u and PW_u. The smart card then calculates $K_u=P_{Ku}\oplus(ID_u||PW_u)$, $K_g=P_{Kg}\oplus(PW_u||ID_u)$, and $N_u^*=h(ID_u||PW_u||K_u)$ and verifies the validity of U by comparing N_u^* with N_u. Once validated the user, U generates a random nonce a and calculates $A=g^a$, $C_{u1}=C_{ug}\oplus h(K_g)$ and $CID_u=E_{Ku}(h(ID_u)||A||S_n||C_{ug})$, where g represents the discrete logarithm problem. After, U sends $\{CID_u, C_{u1}\}$ to GW. Upon receiving the authentication request from U, GW decrypts CIU_u and C_{ug} by executing $D_{Ku}(CID_u)=\{h(ID_u)^*, A, S_n^*, C_{ug}^*\}$ and $D_{Ku}(C_{ug})=\{ID_u^*, ID_g\}$, and validates U by comparing C_{ug}^* with C_{ug} and $h(ID_u)^*$ with $h(ID_u^*)$. Once validated U, GW generates a random nonce b and computes $B=g^b$, $C_{g1}=K_g\oplus(A||B)$, the session key $SK_{GW\text{-}U}=A^b$, and $A_m=h(SK_{U\text{-}GW}||A||B)$ and sends $\{C_{g1}, A_m\}$ to U. Additionally, GW computes $C_{g2}=h(K_{gs})\oplus(CID_u||A||B)$ and $A_u=h(K_{gs}||S_n||A||B)$, and sends $\{C_{g2}, A_u\}$ to the sensor node. On receiving $\{C_{g1}, A_m\}$ from GW, U verifies the legitimacy of GW by computing $(A^*||B^*)=C_{g1}\oplus K_g$, $SK_{sessU\text{-}}$

${}_{GW}=B^a$, *and* $A_m^*=h(SK_{sessU\text{-}GW}||A||B)$ and comparing A^* with A and A_m^* with A_m. On the other *hand*, on receiving $\{C_{g2}, A_u\}$ from GW, the sensor node verifies the legitimacy of GW by calculating $(CID_u^*||A^*||B^*)=C_{g2}\oplus h(K_{gs})$, $A_u^*=h(K_{gs}||S_n||A^*||B^*)$ and comparing A_u^* with A_u. Once verified the validity of GW, the sensor node generates a random nonce f and calculates $F=g^f$, $SK_{MS\text{-}GW}=B^{*f}$, $C_{s1}=h(CID_u||S_n)\oplus F$, $A_{ms}=h(SK_{MS\text{-}U}||S_n||A^*||F)$ and $SK_{MS\text{-}U}=A^{*f}$, and sends $\{C_{s1}, A_{ms}\}$ to the GW. It also computes $C_{s2}=A^*\oplus F$ *and* $A_{msu}=h(SK_{MS\text{-}U}||S_n||A^*||F)$, and sends $\{C_{s2}, A_{msu}\}$ to U. On receiving $\{C_{s1}, A_{ms}\}$ from the sensor node, GW calculates $F^*=C_{s1}\oplus(CID_u||S_n^*)$, $SK_{GW\text{-}MS}=F^{*b}$, $A_{ms}^*=h(SK_{GW\text{-}MS}||K_{gs}||B^*||F)$ and compares A_{ms}^* with A_{ms} to verify the validity of sensor node. On the other hand, on receiving $\{C_{s2}, A_{msus}\}$ from sensor node, U calculates $F^*=C_{s2}\oplus A$, $SK_{U\text{-}MS}=F^*$, $A_{msu}^*=h(SK_{U\text{-}MS}||S_n||A||F^*)$ and compares A_{msu} with A_{msu}^* to verify the legitimacy of sensor node.

Password Change Phase. This phase is identical to the one proposed in Khan-Kumari scheme [10].

2.4 Cryptanalysis of Shin et al.'s Scheme

One of our previous work [12] has cryptanalyzed the Shin et al.' scheme based on same the assumptions presented by authors. The authors explain that the adversary U_a can obtain a valid smart card to extract values inside it i.e. C_{ug}, N_u, PK_u, PK_g.

Data Leakage Vulnerability. In Login and Authentication phase, GW responds to U's login request with $\{C_{g1}, A_m\}$ where $C_{g1}=K_g\oplus(A||B)$. As the number of bits of K_g will be less than $A||B$, part of $A||B$ will be exposed to U_a. Lets explain with an example: in performance analysis of [10], the authors assumes 128 bits for every data (i.e. random numbers, result of hash functions, timestamps, ID_U, PW_U, etc.); in such case, $A||B$ will have 256 bits while K_g will have 128 bits. If you executes $C_{g1}=K_g\oplus(A||B)$, the value of A will be stay in plaintext, exposing the secret A to the adversary. Similar analysis can be made for the message $C_{g2}=h(K_{gs})\oplus(CID_u||A||B)$ where CID_U and A will be transmitted in plaintext (see Figure 3).

Password Guessing Attack. An internal user U_1 with a valid smart card can easily guess the ID_u and PW_u of another user with the following steps. First, the malicious internal user U_1 extracts PK_g from his/her smart card and obtains K_g by calculating $K_g=PK_g\oplus(PW_{u1}||ID_{u1})$ where ID_{u1} and PW_{u1} are the identity and password of U_1. Once obtained K_g, U_1 can guess the ID_u and PW_u of any other user U by extracting PK_g from U's smart card and calculating $(PW_u||ID_u)=PK_g\oplus K_g$.

Insecure K_g and K_{gs}. Most of the security of the Shin et al.'s scheme is based on K_g and K_{gs}. However, those values are shared by all users and sensor nodes. Therefore, if an internal user becomes malicious, K_g could leak easily. On the other hand, if an attacker U_a succeeded to discover the K_{gs} value (e.g. using brute force attack) all sensor nodes would be exposed allowing U_a to execute sensor node spoofing attacks. It is important to say that, as all sensor nodes shares the same K_{gs}, U_a would be more motivated to execute a brute force attack to get such secret value.

Replay attack to GW. In Shin et al.'s scheme, the GW does not verify the freshness of messages sent by U and sensor nodes; therefore, replay attack is possible. During the login and authentication phase, the attacker U_a can eavesdrop the network and capture $\{CID_u, C_{u1}\}$ and U_a can use it again because GW does not verify if such message was used before. Same situation occurs with $\{C_{s1}, A_{ms}\}$ sent by the sensor node.

Replay attack to U. U also does not verify the freshness of messages sent by GW or sensor nodes. Therefore, the attacker could reuse the $\{C_{s2}, A_{msu}\}$ message sent by a sensor node.

Replay attack to sensor nodes. Sensor nodes lack of message freshness' verification. Therefore, the attacker can reuse the $\{C_{g2}, A_u\}$ sent from GW.

Lack of details in creating session keys. The authors indicates that they are using the discrete logarithm problem to compute the session keys. However, they do not explain the details of it.

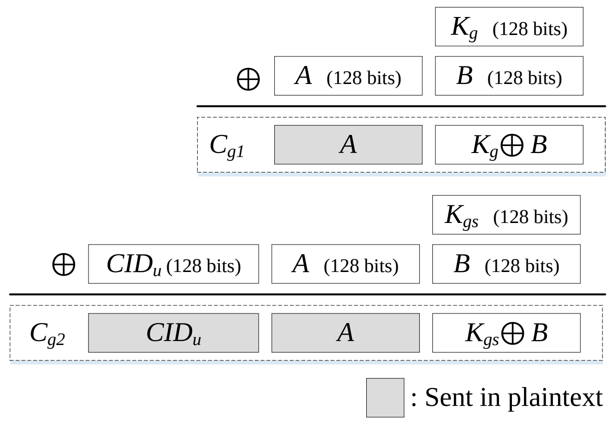

Figure 3: Illustration of Data Leakage Vulnerability in Shin et al.' Scheme

3 THE PROPOSED SCHEME

This section describes the proposed enhanced authentication protocol which repairs the limitations of aforementioned researches. The proposed protocol involves five phases: initialization, user registration, patient registration, login and authentication, and password change phases. It is important to mention that four different entities participate in the proposed protocol i.e. gateway node, user, patient, and sensor nodes.

3.1 Initialization Phase

The Gateway node *GW* stores two secret keys K_1 and K_2. We recommend the use of hardware based security technologies such as TrustZone by ARM [15] and Secure RAM by Freescale [16] to avoid the leakage of such keys. These technologies was applied in many security solution such as [17]. *GW* uses its secret keys to generate an unique key $K_{SG}=h(ID_S||ID_G||K_2)$ for every sensor nodes *S* where ID_S is the identification of *S* and ID_G is the identification of *GW*.

3.2 User Registration Phase

A user *U* selects his/her unique identification ID_U and password PW_U. *U* also creates a random number RN_U and calculates a pseudo password $PPW_U=h(ID_U||PW_U||RN_U)$, where *h(.)* is a hash function, $\oplus$ is an XOR operator and || is a concatenation operator. Once calculated PPW_U, *U* sends ID_U and PPW_U to *GW* using a secure channel. *GW* then calculates $KG_{UG}=h(ID_U||ID_G||K_1)\oplus PPW_U$ where ID_G is the identity of *GW*. Once calculated K_{UG}, *GW* stores KG_{UG} and *h(.)* values inside of a smart card and delivers it to *U* in a secure manner. Finally, *U* stores PPW_U and RN_U into the smart card.

3.3 Patient Registration Phase

The patient goes to the registration center and delivers his/her information. After, he/she chooses the suitable sensor kit and designates professionals/users of the medical staff. Later, the registration center sends to the patient the sensor kit which will monitor his/her health conditions.

3.4 Login and Authentication Phase

This phase is executed when *U* wants to communicate with the *GW* and *S*, and it is composed of Login and Authentication Phases.

3.4.1 Login Phase

U inserts the smart card and inputs his/her ID_U and PW_U. Then, the smart card calculates $PPW_U{*}=h(ID_U||PW_U||RN_U)$ and compares the computed $PPW_U{*}$ with PPW_U stored in the smart card. The next step is executed only if those values are equal, on the contrary, the login process is terminated. Otherwise, the smart card generates a random nonce *M*, calculates $K_{UG}=KG_{UG}\oplus PPW_U$ and computes a temporary key $TK1=h(ID_U||ID_G||T_U||K_{UG})$ and $CID_U=E_{TK1}(ID_U||ID_G||ID_S||T_U||M)$ where T_U is the current timestamp, ID_S is the identity of the sensor node *S* which *U* wants to access, and $E_x(y)$ is a symmetric encryption of *y* using key *x*. After calculating those values, *U* sends $\{ID_U, ID_G, T_U, CID_U\}$ to *GW*. ID_G is incorporated only if there are various gateways in the system.

3.4.2 Authentication Phase

Once received the login request message at time T_U', *GW* verifies the freshness and validity of T_U. If $(T_U'-T_U)>\Delta T$, *GW* terminates the authentication process. ΔT represents the communication delay threshold. Once verified T_U, *GW* calculates $TK1{*}=h(ID_U||ID_G||T_U||K_{UG})$ and decrypts CID_U by $(ID_U{*}||ID_G{*}||ID_S||T_U{*}||M)=D_{TK1*}(CID_U)$ where $D_x(y)$ is a symmetric decryption of *y* using key *x*. Once decrypted CID_U, *GW* compares ID_U with $ID_U{*}$, ID_G with $ID_G{*}$ and T_U with $T_U{*}$. The authentication continues only if those values match. Once verified the authenticity of *U*, *GW* gets its current timestamp T_{G1}, calculates $C_{G1}=h(ID_U||T_{G1}||M||K_{UG})$ and send $\{C_{G1}, T_{G1}\}$ to *U*. *U* then computes $C_{G1}{*}=h(ID_U||T_{G1}||M||K_{UG})$ and compares $C_{G1}{*}$ with the received C_{G1} to verify the correct authentication with *GW*. Finally, *U* and *GW* calculates their session key $K_{sessUG}=h(M||T_U||T_{G1})$. It is important to indicate that timestamps are usable only once to eliminate parallel session attacks.

On the other hand, after sending $\{C_{G1}, T_{G1}\}$ to *U*, *GW* also calculates another temporary key $TK2=h(ID_S||ID_G||T_{G2}||K_{SG})$, generates a random nonce *N*, computes $C_{G2}=E_{TK2}(ID_S||ID_G||ID_U||T_{G2}||T_U||M||N)$, and sends $\{ID_S, ID_G, T_{G2}, C_{G2}\}$ to *S* where T_{G2} is the current timestamp of *GW*. Upon receiving $\{ID_S, ID_G, T_{G2}, C_{G2}\}$ at time T_{G2}', *S* verifies the freshness of T_{G2}. If $(T_{G2}'-T_{G2})>\Delta T$, *S* terminates the authentication process, where ΔT represents the communication delay threshold. Once verified T_{G2}, *S* calculates $TK2{*}=h(ID_S||ID_G||T_{G2}||K_{SG})$ and decrypts C_{G2} by $(ID_S{*}||ID_G{*}||ID_U||T_{G2}{*}||T_U||M||N)=D_{TK2*}(C_{G2})$.

Once decrypted C_{G2}, *S* compares ID_S with $ID_S{*}$, ID_G with $ID_G{*}$ and T_{G2} with $T_{G2}{*}$. The authentication continues only if those values match. Once verified the authenticity of *GW*, *S* gets its current timestamp T_{S1}, calculates $C_{S1}=h(ID_S||T_{S1}||N||K_{SG})$ and send $\{C_{S1}, T_{S1}\}$ to *GW*. *GW* then computes $C_{S1}{*}=h(ID_S||T_{S1}||N||K_{SG})$ and compares $C_{S1}{*}$ with the received C_{S1} to verify the correct authentication with *S*. Finally, *S* and *GW* calculates their session

key $K_{sessSG}=h(N||T_{S1}||T_{G2})$. Again, it is important to indicate that timestamps are usable only once to eliminate parallel session attacks.
After authentication with *GW*, *S* gets its current timestamp T_{S2}, calculates $C_{S2}=h(ID_S||T_{S2}||M)$ and sends $\{C_{S2}, T_{S2}\}$ to *U*. *U* then calculates $C_{S2}{*}=h(ID_S||T_{S2}||M)$ and compares C_{S2} with $C_{S2}{*}$ to verify the validity of *S*. Finally *U* and *S* computes their session key $K_{sessUS}=h(M||T_U||T_{S2})$.

3.5 Password Change Phase
U inserts the smart card and input his/her ID_U, PW_U and $(PW_U)_{New}$. Then, the smart card calculates $PPW_U{*}=h(ID_U||PW_U||RN_U)$ and compares the computed $PPW_U{*}$ with PPW_U stored in the smart card. The next step of the password change phase is executed only with the correct comparison result. Once, verified the authenticity of PPW_U, the smart card generates $(PPW_U)_{New}=h(ID_U||(PW_U)_{New}||RN_U)$ and replaces PPW_U with $(PPW_U)_{New}$.

4 Analysis of the Proposed Scheme
This part of the paper executes the security and performance analysis of the proposed protocol.

4.1 Security Analysis
This part executes a formal verification and analysis of possible attacks. The initialization, registration and password change phases were excluded from analysis because they are performed in a protected environment. For the analysis of the login and authentication phase, the well-known Dolev-Yao [18] threat model was used, which considers an insecure communication channel among participating entities.

4.1.1 Formal Proof Based on BAN Logic
This part executes a formal verification using the widely used BAN logic [19, 20]. BAN logic has been used in different works such as [8], [21-23] to reason about their security validation. The basic notations of BAN logic mentioned in this section are detailed in the Table 1. Additionally, below are the explanation of logical postulates to be used in proofs and their descriptions.

Message-meaning rule: If the entity *P* shares a secret key *K* with the entity *Q*, and *P* receives a message *X* encrypted or hashed with *K*, then the entity *P* believes that *X* was sent by *Q*.

$$\frac{P \mid\equiv P \xleftrightarrow{K} Q, P \lhd \{X\}_K}{P \mid\equiv Q \mid\sim X}, \quad \frac{P \mid\equiv P \xleftrightarrow{K} Q, P \lhd (X)_K}{P \mid\equiv Q \mid\sim X}$$

Table 1: Notations of BAN Logic

Notation	Description
$P \mid\equiv X$	The entity *P* believes in *X*.
$\#(X)$	The content of *X* is fresh.
$P \Rightarrow X$	The entity *P* has jurisdiction over the *X*.
$P \lhd X$	The entity *P* has *received* the message *X*.
$P \mid\sim X$	The entity *P* sent a message *X*.
(X, Y)	*X* or *Y* is one *part* of the formula (*X*, *Y*).
$\{X\}_K$	The content of *X* is encrypted using a key *K*
$(X)_K$	The content *X* is hashed using a *K*.
$P \xleftrightarrow{K} Q$	Entities *P* and *Q shares* a secret key *K* only known by them.

Freshness-conjuncatenation rule: If the entity *P* trusts freshness of the message *X*, then the entity *P* **trusts** freshness of (*X*, *Y*).

$$\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X, Y)}$$

Nonce-verification rule: If the entity *P* believes that **the** message *X* is fresh and if *P* believes that *X* was sent by *Q*, then *P* believes that *Q* believes *X*.

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

Jurisdiction rule: If entity *P* believes that the entity *Q* has **jurisdiction** over the message *X*, then the *P* believes *Q* on the validity of *X*.

$$\frac{P \mid\equiv Q \Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

In the following, the security proof process is detailed. To reach the formal proof, the proposed protocol must satisfy the following goals:

$$U \mid\equiv U \xleftrightarrow{K_{sessUG}} GW \quad \text{(G.1)}$$
$$U \mid\equiv GW \mid\equiv U \xleftrightarrow{K_{sessUG}} GW \quad \text{(G.2)}$$
$$GW \mid\equiv U \xleftrightarrow{K_{sessUG}} GW \quad \text{(G.3)}$$
$$GW \mid\equiv U \mid\equiv U \xleftrightarrow{K_{sessUG}} GW \quad \text{(G.4)}$$
$$S \mid\equiv S \xleftrightarrow{K_{sessSG}} GW \quad \text{(G.5)}$$
$$S \mid\equiv GW \mid\equiv S \xleftrightarrow{K_{sessSG}} GW \quad \text{(G.6)}$$
$$GW \mid\equiv S \xleftrightarrow{K_{sessSG}} GW \quad \text{(G.7)}$$
$$GW \mid\equiv S \mid\equiv S \xleftrightarrow{K_{sessSG}} GW \quad \text{(G.8)}$$

$$U \mid\equiv U \xleftrightarrow{K_{sessUS}} S \qquad (G.9)$$
$$U \mid\equiv S \mid\equiv U \xleftrightarrow{K_{sessUS}} S \qquad (G.10)$$
$$S \mid\equiv U \xleftrightarrow{K_{sessUS}} S \qquad (G.11)$$
$$S \mid\equiv U \mid\equiv U \xleftrightarrow{K_{sessUS}} S \qquad (G.12)$$

First, communication messages are converted to the idealized form:

$$U \rightarrow GW: \quad ID_U, ID_G, T_U, ((ID_U||ID_G||ID_S||T_U||M)_{TK1} \qquad (M.1)$$
$$U \leftarrow GW: \quad \{h(ID_U||T_{G1}||M||K_{UG})\}_{KUG}, T_{G1} \qquad (M.2)$$
$$GW \rightarrow S: \quad ID_S, ID_G, T_{G2}, (ID_S||ID_G||ID_U||T_{G2}||T_U||M||N)_{TK2} \qquad (M.3)$$
$$GW \leftarrow S: \quad \{h(ID_S||T_{S1}||N||K_{SG})\}_{KSG}, T_{S1} \qquad (M.4)$$
$$U \leftarrow S: \quad \{h(ID_S||T_{S2}||M)\}_M, T_{S2} \qquad (M.5)$$

Second, we define the assumptions how the protocol will start.

$$GW \mid\equiv \#(T_U) \qquad (A.1)$$
$$U \mid\equiv \#(T_{G1}) \qquad (A.2)$$
$$S \mid\equiv \#(T_{G2}) \qquad (A.3)$$
$$GW \mid\equiv \#(T_{S1}) \qquad (A.4)$$
$$U \mid\equiv \#(T_{S2}) \qquad (A.5)$$
$$U \mid\equiv U \xleftrightarrow{K_{UG}} GW \qquad (A.6)$$
$$GW \mid\equiv U \xleftrightarrow{K_{UG}} GW \qquad (A.7)$$
$$S \mid\equiv S \xleftrightarrow{K_{SG}} GW \qquad (A.8)$$
$$GW \mid\equiv S \xleftrightarrow{K_{SG}} GW \qquad (A.9)$$
$$U \mid\equiv M \qquad (A.10)$$

Finally, the previously listed assumptions and BAN logic postulates are used to execute the proof steps (see Table 5). The goals listed previously were achieved by (S.15) ~ (S.22) and (S.26) ~ (S.29). In conclusion, the proposed scheme generates fresh and secure session key and delivers mutual authentication.

4.1.2 Security Verification from Possible Attacks. This subsection analyses the security of the proposed protocol against possible attacks. As explained previously, this work assumes that common communication channels are insecure and there are adversaries that can intercept messages coming from *U*, *S*, and *GW*. In addition, this work assumes that the adversary can acquire an authentic smart card of a valid user and can extract values stored inside it [14, 15].

Data Leakage Vulnerability. Previous works included this vulnerability because they made use of simple XOR operations to hide sensible data. In the proposed mechanism, XOR operations for hiding sensible data was omitted and all sensible data sent in the insecure channel is secured using a one-way hash function with a secret key i.e. K_{UG} and K_{SG} or they are encrypted using a secure symmetric cryptographic function.

Password Guessing Attack. This attack is not possible in the proposed scheme because PW_U is not stored anywhere. Instead, a variant value PPW_U is used for validity user's password. The adversary, even with a valid smart card, cannot guess the PW_U because it is protected using a secure one-way hash function.

Man in the middle attack. In previous works, this attack was possible because of the data leakage vulnerability. In the proposed protocol, this attack is not possible because the adversary U_a cannot estimate the message $\{C_{G1}, T_{G1}\}$ since C_{G1} is created based on a secret key K_{UG} that U_a cannot obtain.

Insecure Secret Keys. Previous works had this problem because they used secret values shared by all users and sensor nodes i.e. K_g and K_{gs}. In the proposed protocol, each user has his/her own secret key $KG_{UG}=h(ID_U||ID_G||K_1)\oplus PPW_U$ and each sensor node has its own secret key $K_{SG}=h(ID_S||ID_G||K_2)$. Therefore, leakage of one secret key will not affect to the rest of users or sensor nodes.

Replay Attack. Several well-known mechanism such as timestamps and random nonces are included to avoid all possibility of replay attacks. A timestamp mechanism is used for delivering freshness of request at the initial steps of each authentication request. After, a stronger technique based on random nonces *M* and *N* is used to verify the validity of *GW*.

User Impersonation Attack. Even though another legal user of the system eavesdrops on *U*'s message $\{ID_U, ID_G, T_U, CID_U\}$, he/she cannot obtain a valid CID_U because CID_U is computed using a temporal key TK_1 which was derived from a unique secret key of the user K_{UG}. The proposed mechanism solved the problem by generating a unique secret key K_{UG} for each user of the system.

Mutual Authentication. The proposed protocol delivers mutual authentication to all participant entities i.e. mutual authentication between *U* and *GW*, and between *S* and *GW*.
(1) *U* and *GW*: *GW* verifies the authenticity of *U* by comparing ID_U, ID_G and T_U sent in plaintext with

the ones included in CID_U. GW can be sure of U because CID_U can be created only by the authentic U who knows the secret value K_{UG}. On the other hand, U verifies the authenticity of GW by comparing C_{G1} sent by GW with $C_{G1}*$ calculated by U. C_{G1} can only be calculated by the authentic GW because it is based on the secret value K_{UG}. Additionally, U can also be sure of the authenticity and freshness of the message because C_{G1} was calculated based on the random secret nonce M sent by the same U.

(2) S and GW: S verifies the authenticity of GW by comparing ID_S, ID_G and T_{G2} sent in plaintext with the ones included in C_{G2}. S can be sure of GW because C_{G2} can be created only by the authentic GW who knows the secret value K_{SG}. On the other hand, S verifies the authenticity of GW by comparing C_{S1} sent by S with $C_{S1}*$ calculated by GW. C_{S1} can only be calculated by the authentic S because it is based on the secret value K_{SG}. Additionally, GW can also be sure of the authenticity and freshness of the message because C_{S1} was calculated based on the random secret nonce N sent by GW.

Table 2: Enhanced security features of the proposed scheme

Security Feature	Propo-sed	Kumar et al.	Khan-Kumari	Shin et al.
Protection against User impersonation attack	Yes	No	Yes	Yes
Mutual authentication	Yes	No	Yes	Yes
Protection against Data Leakage Vulnerability	Yes	Yes	No	No
Protection against Man in the middle attack	Yes	No	No	Yes
Protection against password guessing attack	Yes	Yes	No	No
Secure individual keys	Yes	No	No	No
Protection against replay attacks	Yes	Yes	Yes	No
Clear session key creation	Yes	Yes	Yes	No

Privileged-Insider Attack. In the proposed solution, U never transmits his/her PW_U because the mechanism uses PPW_U instead of PW_U. An adversary cannot calculate PW_U using the PPW_U because PPW_U is calculated using a secure one-way hash function.

Stolen Verifier Attack. The proposed protocol does not make use of password tables. Therefore, it is strong against this type of attack.

Many Logged-in Users with the Same Login ID. The usage of smart card delivers stronger protection than only password based schemes. Assuming that U's smart card is not cloned, the proposed scheme avoids this threat since the login steps requires calculations inside the smart card.

Brute Force Attack. An adversary can try two kinds of brute force attacks. (1) First, the attacker can attempt to authenticate by sending random or sequential messages $\{ID_U, ID_G, T_U, CID_U\}$ to GW. However, as well as explained in the replay attack, this attack becomes infeasible because each authentication process uses a different timestamp. (2) On the other hand, a malicious internal user can attempt to determine the secret values K_1 or K_2 using his/her smart card. However, the obtainment of such data is not possible since they are stored as non-reversible hash values.

Session Key Generation. The proposed solution offers a simple but practical methods for generation of session keys among U, S, and GW.

Table 2 compares the security features of the proposed protocol with the previous ones. Such table indicates how our solution provides better security features.

4.2 Performance Analysis

For this analysis, this work has followed the same conditions of [10]. Table 3 exhibits the performance in terms of memory space required by smart card while Table 4 shows the computational complexity/cost in compared schemes. For calculation of memory needed in smart card, this work assumed that the identity ID_U, PW_U, random numbers, timestamps, hash functions are 128 bits long to follow the same conditions proposed in [10]. Table 3 shows how the proposed solution requires 20% less space that the latest solutions and requires the same space that the Kumar et al. scheme. On the other hand, Table 4 indicates that the proposed protocol requires less operations in registration and password change phases than other solutions and requires similar number of operations in login and authentication phase. In summary, the proposed solution provides enhanced security features without any important overhead in terms of performance.

Table 3: Memory needed by smart card in different schemes

Proposed	Kumar et al.	Khan-Kumari	Shin et al.
512bits	512bits	640 bits	640bits

5 Conclusions

This paper has proposed an enhanced user authentication protocol that eliminates the security flaws found our previous work and provides a strong security, as demonstrated using formal method (BAN logic) and detailed security analysis. The proposed solution provides enhanced security features without any important overhead in terms of performance. The proposed scheme applies the benefit of the two-factor authentication mechanism to deliver a secure authentication system offering balanced features in terms of security and performance.

References:

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, *In Proceeding of the 1st ACM international workshop on wireless sensor networks and applications*, 2002, pp. 88-97.

[2] C. Otto, A. Milenkovic, C. Sanders, E. Jovanov, System architecture of a wireless body area sensor network for ubiquitous health monitoring, *Journal Mobile Multimedia*, Vol. 1, Issue 4, 2006, pp. 307-326.

[3] S. Lee, Wireless sensor network design for tactical military applications: Remote large-scale environments, *In Proceedings of 2009 IEEE Military Communications Conference MILCOM2009*, 2009, pp. 1-7.

[4] G. Werner-Allen, Fidelity and yield in a volcano monitoring sensor network, *In Proceedings of the 7th symposium on Operating systems design and implementation OSDI '06*, 2006, pp. 381-396.

[5] N. Xu, A Survey of Sensor Network Applications, *IEEE Communications Magazine*, 2002, pp. 1-9.

[6] A. Wang, A Configurable Energy-Efficient Compressed Sensing Architecture With Its Application on Body Sensor Networks, *IEEE Transactions on Industrial Informatics*, Vol. 12, Issue 1, 2015, pp. 15-27.

[7] S. Yoo, S. Kang, J. Kim, SERA: A Secure Energy-Reliability Aware Data Gathering for Sensor Networks, *Multimedia Tools and Applications*, Vol. 73, Issue 2, 2014, pp. 617-646.

[8] S. Yoo, K. Park, J. Kim, A Security-Performance Balanced User Authentication Scheme for Wireless Sensor Networks, *International Journal of Distributed Sensor Networks*, Vol. 2012, Article ID 382810, 2012.

[9] P. Kumar, S. Lee, H. Lee, E-SAP: Efficient-Strong Authentication Protocol for Healthcare Applications Using Wireless Medical Sensor Networks, *Sensors*, Vol. 12, Issue 2, 2012, pp. 1625-1647.

[10] M. Khan, S. Kumari, An Improved User Authentication Protocol for Healthcare Services via Wireless Medical Sensor Networks, International Journal of Distributed Sensor Networks, Vol. 2014, article ID 347169, 2014.

[11] S. Shin, S. Lee, H. Kim, Authentication Protocol for Healthcare Services over Wireless Body Area Networks, *International Journal of Computer and Communication Engineering*, Vol. 5, Issue 1, 2016, pp.50-60.

[12] S. Yoo, Cryptanalysis of Several Authentication Schemes for Healthcare Applications using Wireless Medical Sensor Networks, *In Proceedings of the 2016 International Conference on Network, Communication and Computing (ICNCC 2016)*, 2016.

[13] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, *In Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'99)*, 1999, pp. 388-397.

[14] T. Messerges, E. Dabbish, R. Sloan, Examining smart-card security under the threat of power analysis attacks, *IEEE Transactions on Computers*, Vol. 51, 5, 2002, pp. 541-552.

[15] A. Alvez, D. Felton, TrustZone: Integrated Hardware and Software Security. Enabling Trusted Computing in Embedded Systems, *White Paper*, 2004.

[16] Freescale Semiconductors, Security features in the i.MX31 and i.MX31L multimedia application processors, *White Paper*, IMX31SECURITYWP Rev. 0.0, 2005.

[17] S. Yoo, K. Park, J. Kim, Confidential information protection system for mobile devices, *Security and communication networks*, Vol. 5, 2012, pp. 1452-1461.

[18] D. Dolev, A. Yao, On the security of public-key protocols. *IEEE Transactions on Information Theory*, Vol. 29, no.2, 1983, pp. 198-208.

[19] M. Burrows, M. Abadi, R. Needhan, Logic of authentication, *ACM Transactions on Computer Systems*, Volume 8. no. 1, 1990, pp. 18-36.

[20] Y. Kim, C. Moon, D. Jeong, D. Baik, Formal Verification of Bundle Authentication Mechanism in OSGi Service Platform: BAN Logic, *International Journal of Software Engineering*, Vol. 16, no. 2, 2006, pp. 153-173.

[21] W. Tsaur, J. Li, W. Lee, An efficient and secure multiserver authentication scheme with key agreement, *Journal of Systems and Software*, Vol. 85, no. 4, 2012, pp. 876–882.

[22] S. Wang, Q. Ma, Y. Zhang, Y. Li, An authentication protocol for RFID tag and its simulation, *Journal of Networks*, Vol. 6, no. 3, 2011, pp. 446–453.

[23] J. Tsai, T. Wu, K. Tsai, New dynamic ID authentication scheme using smart cards, *International Journal of Communication Systems*, Vol. 23, 2010, pp. 1449–1462.

Table 4: Summary of performance analysis i.e. number of operations (h: hash, se: symmetric encryption, sd: symmetric decryption, xor: xor operation, dlo: discrete logarithm operation)

Phase	Entity	Proposed	Kumar et al.	Khan-Kumari	Shin et al.
Registration Phase	User	1h	-	1h+2xor	1h+2xor
	GW	1h+1xor	1se+1h +2xor	1se+2h	1se+2h
	Sensor	-	-	-	-
Login and Authentication Phase	User	6h+1xor+1se	1se + 2h+2xor	6h+1se +4xor	2xor+3h +1dlo+1lp
	GW	6h+1sd+1se	1se + 1sd + 3h	7h+2sd +2xor	2sd+3dlo +2xor+6h
	Sensor	4h+1sd	1se + 1sd+1h	7h+2xor	5h+3xor +3dlo
Password Change Phase	User	2h	2h+4xor	2h+2xor	2h+2xor
	GW	-	-	-	-
	Sensor	-	-	-	-

Table 5: BAN Logic Proof Steps

$GW \triangleleft ID_U, ID_G, T_U, (ID_U \Vert ID_G \Vert ID_S \Vert T_U \Vert M)_{TK1}$	(S.1)	// by (M.1)
$GW \mid\equiv \#(TK1), GW \mid\equiv TK1$	(S.2)	// by (S.1), (A.1), (A,7), and *TK1* generation algorithm shared between *U* and *GW*
$GW \mid\equiv U \mid\sim M$	(S.3)	// by (S.1), (S.2), and message-meaning rule
$GW \mid\equiv U \mid\equiv M$	(S.4)	// by (S.2), (S.3), (A.1), freshness-conjugation rule, and nonce-verification rule
$U \triangleleft \{h(ID_U \Vert T_{G1} \Vert M \Vert K_{UG})\}_{K_{UG}}, T_{G1}$	(S.5)	// by (M.2)
$U \mid\equiv GW \mid\sim (T_{G1}, M)$	(S.6)	// by (S.5), (A.6), and message-meaning rule
$U \mid\equiv GW \mid\equiv (T_{G1}, M)$	(S.7)	// by (S.6), (A.2), freshness-conjugation rule, and nonce-verification rule
$S \triangleleft ID_S, ID_G, T_{G2}, (ID_S \Vert ID_G \Vert ID_U \Vert T_{G2} \Vert T_U \Vert M \Vert N)_{TK2}$	(S.8)	// by (M.3)
$S \mid\equiv \#(TK2), S \mid\equiv TK2$	(S.9)	// by (S.8), (A.3), (A.8), and *TK2* generation algorithm shared between *S* and *GW*
$S \mid\equiv GW \mid\sim (T_{G2}, M, N)$	(S.10)	// by (S.8), (A.9), and message-meaning rule
$S \mid\equiv GW \mid\equiv (T_{G2}, M, N)$	(S.11)	// by (S.10), (A.3), freshness-conjugation rule, and nonce-verification rule

$GW \triangleleft \{h(ID_S \| T_{S1} \| N \| K_{SG})\}_{KSG}, T_{S2}$	(S.12)	// by (M.4)
$GW \mid\equiv S \mid\sim (T_{S1}, N)$	(S.13)	// by (S.12), (A.9), and message-meaning rule
$GW \mid\equiv S \mid\equiv (T_{S1}, N)$	(S.14)	// by (S.13), (A.4), freshness-conjugation rule, and nonce-verification rule
$GW \mid\equiv \#(Ksess_{UG}), GW \mid\equiv Ksess_{UG}$	(S.15)	// once verified (S.4), *GW* computes $Ksess_{UG}$
$U \mid\equiv \#(Ksess_{UG}), U \mid\equiv Ksess_{UG}$	(S.16)	// once verified (S.7), *U* computes $Ksess_{UG}$
$S \mid\equiv \#(Ksess_{SG}), S \mid\equiv Ksess_{SG}$	(S.17)	// once verified (S.11), *S* computes $Ksess_{SG}$
$GW \mid\equiv \#(Ksess_{SG}), GW \mid\equiv Ksess_{SG}$	(S.18)	// once verified (S.14), *GW* computes $Ksess_{SG}$
$U \mid\equiv GW \mid\equiv Ksess_{UG}$	(S.19)	// by (S.15), (S.16), and $Ksess_{UG}$ generation algorithm shared between *U* and *GW*
$GW \mid\equiv U \mid\equiv Ksess_{UG}$	(S.20)	// by (S.15), (S.16), and $Ksess_{UG}$ generation algorithm shared between *U* and *GW*
$S \mid\equiv GW \mid\equiv Ksess_{SG}$	(S.21)	// by (S.17), (S.18), and $Ksess_{SG}$ generation algorithm shared between *S* and *GW*
$GW \mid\equiv S \mid\equiv Ksess_{SG}$	(S.22)	// by (S.17), (S.18), and $Ksess_{SG}$ generation algorithm shared between *S* and *GW*
$U \triangleleft \{h(ID_S \| T_{S2} \| M)\}_M, T_{S2}$	(S.23)	// by (M.5)
$U \mid\equiv S \mid\sim (T_{S2}, M)$	(S.24)	// by (S.23), (A.10) and message meaning rule
$U \mid\equiv S \mid\equiv (T_{S2}, M)$	(S.25)	// by (S.24), (A.5), freshness-conjugation rule, and nonce-verification rule
$S \mid\equiv \#(Ksess_{US}), S \mid\equiv Ksess_{US}$	(S.26)	// once verified (S.11), *S* computes $Ksess_{US}$
$U \mid\equiv \#(Ksess_{US}), U \mid\equiv Ksess_{US}$	(S.27)	// once verified (S.25), *S* computes $Ksess_{US}$
$U \mid\equiv S \mid\equiv Ksess_{US}$	(S.28)	// by (S.26), (S.27), and $Ksess_{US}$ generation algorithm shared between *U* and *S*
$S \mid\equiv U \mid\equiv Ksess_{US}$	(S.29)	// by (S.26), (S.27), and $Ksess_{US}$ generation algorithm shared between *U* and *S*