Hindawi
Journal of Sensors
Volume 2018, Article ID 7574238, 16 pages
https://doi.org/10.1155/2018/7574238

Research Article

A Lightweight Privacy Protection User Authentication and Key Agreement Scheme Tailored for the Internet of Things Environment: LightPriAuth

Yuwen Chen, Lourdes López, José-Fernán Martínez, and Pedro Castillejo

Departamento de Ingeniería Telemática y Electrónica (DTE), Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST), Universidad Politécnica de Madrid (UPM), C/Nikola Tesla, s/n 28031 Madrid, Spain

Correspondence should be addressed to Yuwen Chen; yuwen.chen@upm.es

Received 18 April 2018; Revised 13 June 2018; Accepted 29 July 2018; Published 18 September 2018

Academic Editor: Antonio Lazaro

Different data are collected by diverse sensors under an Internet of things scenario, such as health data, environmental data, and traffic flow data. People can access data remotely via the Internet easily. Considering the importance and confidentiality of these data, it is necessary to ensure the data security. In this study, we propose an authentication and key establishment scheme for an Internet of things scenario based on low-capability devices. This scheme achieves many security features: user anonymity, sensor anonymity, forward secrecy, resistance to the loss of synchronization problem, and so on We verified these security features using AVISPA and ProVerif; both results show that the scheme is safe enough to achieve the security requirements. Besides, the experiment results elucidate that this scheme gains an advantage in computation and communication costs. It is because of the sole usage of XOR operations and hash functions as well as a minimal amount of asymmetric encryptions to fulfil forward secrecy.

1. Introduction

As sensors are applied to different aspects of our daily life, too much different personal information has been collected by different kinds of sensors; the personal information includes but is not limited to health information, home temperature, and home humidity. The information is so personal that we do not want them to be leaked when we access them remotely via the network. Many lightweight authentication schemes have been proposed to guarantee the safety of remote data access. Figure 1 depicts the structure of these schemes. There are three kinds of entities, the user who wants to read the data of sensors, sensors which are placed in the environment to collect data, and the gateway, which is introduced to authenticate users and sensors and helps the two to build a shared key. After negotiation of the shared key, the user and sensor can communicate with each other without the help of the gateway.

In this paper, we proposed an authentication and key establishment scheme with user anonymity; this scheme is an improved version of the previous scheme of ours [1] and the LifeWear project [2], which is also based on the ECC. To achieve user anonymity, the identity of the user is encrypted using XOR operation; the key for this encryption is generated by the gateway. When a user registers at the gateway, the gateway generates a random number for the user and a unique key based on this number and the gateway's own secret key. This number could be seen as an indicator of the key; the key and this number are sent to the user. The user could encrypt his identity with this key.

However, this is not enough to ensure perfect anonymity, because adversaries can track the user based on this unique number, even though the adversary does not know the real identity of this user. To prevent adversaries from tracking the users, in our scheme, once a user has been authenticated by the gateway, the gateway will assign a new number and a new key for this user. Thus, the adversary is unable to track the user based on the number, because the number has been updated to a new one. Many other schemes adopted this way of protecting the identity privacy of the users [3–10].

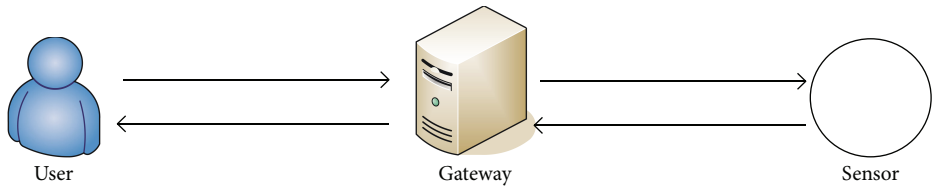

Figure 1: The structure of the model.

Some schemes use an asymmetric encryption method to ensure the anonymity of the authentication scheme. The gateway has a public key that is known by all the members in the scheme; users can use this public key to encrypt their identities; thus, the scheme ensures user anonymity. Our scheme has an advantage compared to the asymmetric encryption method. This is because the asymmetric encryption method requires more computation time compared to our scheme. In our scheme, we encrypt the identity of the user by using the XOR method; the execution time of the XOR operation is minimal compared to an elliptic curve point multiplication [11]. This makes our scheme more suitable for the Internet of things scenario than the asymmetric encryption method.

In the proposed scheme, to enable forward security, the shared key between the user and sensor is generated on an elliptic curve. However, elliptic curve computation needs more computation time compared to the symmetric method; to minimize the computation cost, the proposed scheme only uses four elliptic multiplication operations; as far as we know, this is the least amount needed to build a shared key with perfect forward security. The contribution of this paper is threefold:

(1) The proposed scheme uses the XOR operations, hash operations, and only four elliptic multiplications; the computation cost of the scheme is relatively low, and communication cost decreases at the same time.

(2) The proposed scheme gains various security features: user anonymity, sensor anonymity, users being untraceable, sensors being untraceable, perfect forward secrecy, excellent resistance to the loss of synchronization problem, and so on. Most importantly, the password change phase has been modified to prevent an offline password-guessing attack.

(3) We implement the scheme in HPSL language and test the security features; we analyze the scheme in ProVerif model, too.

2. Related Works

Turkanović et al. discussed the user authentication and key agreement problem for a wireless sensor network [12]. They analyzed the identity protection problem in this scenario; they used a fixed fake identity instead of the real identity to protect the identity of the user. Amin and Biswas proposed an improved scheme [13], which improved several security weaknesses of the protocol of Turkanović et al. They protected the identity privacy by encrypting the identity using a symmetric key that is shared by all the users.

Wu et al. proposed a privacy-preserving and provable user authentication scheme for wireless sensor networks [14]. Every time a user asks access to a sensor's data, the gateway atomically generates a new identity for the user. The identity privacy of the user has been well protected in the scheme, but this scheme faces a loss of synchronization problem. Imagine that the gateway generates a new identity for the user and sends this identity to the user via the Internet but the user does not receive this identity, because either this identity is lost due to poor quality of the network condition or this identity is blocked by an adversary. Thus, when the user logs in the next time using the old identity, he will not be treated as a legal user. Another potential defect of this scheme is that the users in this scheme may be tracked. Even though the adversary does not know the real identity of the user, the adversary can also track the user by the fixed information MI_i, which is used by the user.

Different from [9, 10], Li et al. proposed a three-factor authentication scheme [3] with identity updating based on biometric information. Their scheme can successfully avoid the loss of synchronization problem. They did not update the identity of the users directly. Instead, in their scheme, every time the user logs in, the gateway generates a new key for the user and the user uses this new key to encrypt his identity. The adversary is unable to find out the real identity of the user. Li et al. proposed a similar scheme for wearable sensors in wireless body area networks [4]. In the scheme of Jiang et al., the keys to encrypt the identities of users are the same [6]. Schemes in [5–9] are similar; the key to encrypt the identity is updated when a user logs in. Das proposed a secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks [10]. In this scheme, a temporary identity is generated for the user; every time the user asks for access, the temporary identity will be updated to a new one. However, in this case, the gateway has to store a table of the relationships between this temporary identity and the real identity of the user. This costs extra storage load on the server side.

Asymmetric encryption is used in some authentication schemes to protect the identity privacy of the users. The gateway has a public key that is known by all the users in the system; when a user logs in, he can use the gateway's public key to encrypt his identity. There are asymmetric encryption methods based on the Elliptic Curve Cryptography (ECC) [15], pairing-based cryptography [16] by Tsai and Lo, and the scheme in [17] by Odelu et al. Liu et al. used ECC [18] to protect the identity privacy of the users. All the real identities are encrypted by the public key of the server; only the

server knows the real identities of the users. In the scheme of Wang et al. [19], they used the ECC public key encryption method to protect the identity privacy. Pairing-based cryptography is another popular asymmetric encryption method. The scheme of Li et al. [20], the scheme of Tsai and Lo [21], and the scheme of Shim [22] are all based on bilinear pairing-based cryptography.

3. The Proposed Scheme

The symbols used in the scheme are listed in Table 1. At the beginning of the scheme, GWN generates the parameters for ECC encryption (p, a, b, G, n, and h) and publishes them to the whole system. GWN generates its secret key X_{GWN} and keeps it as a secret.

3.1. Registration Phase of the User. User U_i chooses a random number r_i and computes $\text{MP}_i = h(r_i \| \text{ID}_i \| \text{PW}_i)$. U_i then sends the registration request message $\{\text{ID}_i, \text{MP}_i\}$ to GWN via a private and secure channel.

When GWN receives the user registration message $\{\text{ID}_i, \text{MP}_i\}$, it computes $d_i = h(\text{ID}_i \| X_{\text{GWN}})$ and $f_i = d_i \oplus \text{MP}_i$. Then, GWN will choose a random number k_i and computes $e_i = h(k_i \| X_{\text{GWN}})$; these random numbers k_i and e_i are used to encrypt the identity information at the login and authentication phase. Finally, GWN sends $\{f_i, l_i, k_i\}$ to U_i in a private and secure channel.

User U_i inserts the random nonce r_i into the smart card and stores $\{f_i, l_i, k_i, \text{MP}_i, r_i\}$. Table 2 provides a depiction of the registration phase of the user.

3.2. Registration Phase of the Sensor. The registration messages of the sensors in the registration phase are sent via a private and secure channel. Sensor S_j sends $\{\text{SID}_j\}$ to GW N. After GWN receives the registration message from S_j, it computes $x_j = h(\text{SID}_j \| X_{\text{GWN}})$ and sends x_j to sensor S_j. Sensor S_j keeps this private key in its memory.

3.3. Login and Authentication Phase. When U_i wants to access a sensor's data via network remotely, U_i has to log in first. A user inserts his smart card (SC) into a card reader and inputs his identity ID_i' and password PW_i'. SC computes a temporary version $\text{MP}_i' = h(r_i \| \text{ID}_i' \| \text{PW}_i')$ using the user inputs PW_i' and ID_i' and the stored value r_i. Then SC computes the information for login.

(1) SC computes $d_i = f_i \bigoplus \text{MP}_i'$ and $e_i = l_i \oplus \text{MP}_i'$ using MP_i'.

(2) SC chooses a random number $k_1 \in [1, n-1]$ and gets $A = k_1 \cdot G$.

(3) SC gets the hash value $M_2 = h(A \| \text{ID}_i \| \text{SID}_j \| d_i \| T_1)$.

(4) SC encrypts ID_i, SID_j, and M_2 with e_i to get $M_1 = e_i \bigoplus (\text{ID}_i \| \text{SID}_j \| M_2)$.

Table 1: Symbols used in the PriAuth.

Symbols	Meaning
GWN	Gateway
U_i, ID_i	The ith user and his identity
S_j, SID_j	The jth sensor node and its identity
$\|\|$	Connects two strings together
$\bigoplus$	XOR operation
X_{GWN}	GWN's secret key
G	The generator of ECC
SK_{ij}, SK_{ij}'	Shared key between U_i and S_j
T_1, T_2	Timestamp
h	General hash function

Table 2: Registration phase of the user.

User ID_i, PW_i	Gateway *master key* X_{GWN}
random number r_i	
$\text{MP}_i = h(r_i \| \text{ID}_i \| \text{PW}_i)$	
$\{\text{ID}_i, \text{MP}_i\}$ →	$d_i = h(\text{ID}_i \| X_{\text{GWN}})$
	$f_i = d_i \bigoplus \text{MP}_i$
	random k_i, $e_i = h(k_i \| X_{\text{GWN}})$
	$l_i = e_i \bigoplus \text{MP}_i$
Stores $\{f_i, l_i, k_i, MP_i, r_i\}$	← $\{f_i, l_i, k_i\}$

(5) SC retrieves the stored k_i, timestamp T_1, and sends Message $1 = \{A, k_i, M_1, T_1\}$ to gateway via a public channel.

When the gateway receives Message $1 = \{A, k_i, M_1, T_1\}$ from a user U_i, GWN first checks the freshness of the message by the timestamp, then GWN checks if this message is from a legal user or not. GWN will abandon this message if it is not from a legal user; otherwise, GWN will forward this request to the sensor SID_j. The process is depicted in the following:

(1) GWN checks the freshness of T_1; if T_1 is not fresh, GWN abandons this message; otherwise, it goes to the next step.

(2) GWN computes $e_i' = h(k_i \| X_{\text{GWN}})$ using the received k_i and its private key X_{GWN}.

(3) GWN decrypts M_1 using e_i' and gets $\text{ID}_i' \| \text{SID}_j' \| M_2' = M_1 \bigoplus e_i'$.

(4) GWN computes d_i' by $d_i' = h(\text{ID}_i' \| X_{\text{GWN}})$.

(5) GWN computes x_j' by $x_j' = h(\text{SID}_j' \| X_{\text{GWN}})$

(6) GWN uses d_i', A, ID_i', SID_j', and T_1 to check if $M_2 = h(A\|\mathrm{ID}_i'\|\mathrm{SID}_j'\|d_i'\|T_1)$. If they are equal, it goes to next step; otherwise, the protocol terminates here.

(7) GWN gets timestamp T_2 and computes $M_3 = h(A\|\mathrm{SID}_j'\|x_j'\|T_2)$.

(8) GWN sends Message $2 = \{A, M_3, T_2\}$ to sensor S_j.

After sensor S_j receives Message $2 = \{A, M_3, T_2\}$, S_j first checks the legitimacy of this message; if it is from the gateway, then S_j replies to the message in the following way.

(1) S_j checks the freshness of the T_2; if T_2 is not fresh, S_j abandons this message; otherwise, it goes to the next step.

(2) S_j checks if $M_3 = h(A\|\mathrm{SID}_j\|x_j\|T_2)$; if they are equal, S_j learns that this information is from the gateway, and it goes to the next step; otherwise, the protocol terminates here.

(3) S_j chooses a random number $k_2 \in [1, n-1]$ and gets $B = k_2 \cdot G$.

(4) S_j calculates the shared key SK_{ij} between U_i and S_j: $\mathrm{SK}_{ij} = h(k_2 \cdot A) = h(k_1 \cdot k_2 \cdot G)$.

(5) S_j calculates the hash value $M_4 = h(B\|\mathrm{SK}_{ij}\|A)$ and $M_5 = h(A\|x_j\|M_3\|M_4\|B)$.

(6) S_j sends Message $3 = \{B, M_4, M_5\}$ to GWN.

After GWN receives Message $3 = \{B, M_4, M_5\}$, first, G WN authenticates the source of the message, then GWN generates the new random number and the new key for the user; afterwards, GWN will send these encrypted information to the user.

(1) GWN checks if $M_5 = h(A\|x_j\|M_3\|M_4\|B)$; if they are equal, it goes to the next step; otherwise, it terminates here.

(2) GWN chooses a random number k_3 and computes $e_{\mathrm{inew}} = h(k_3\|X_{\mathrm{GWN}})$.

(3) GWN calculates $M_7 = h(e_{\mathrm{inew}}\|k_3\|d_i'\|T_1\|M_4)$.

(4) GWN computes $M_6 = (e_{\mathrm{inew}}\|k_3\|M_7) \bigoplus e_i'$.

(5) GWN sends Message $4 = \{B, M_6\}$ to U_i.

After U_i receives Message $4 = \{B, M_6\}$, it authenticates if the message is from the gateway; if the message is from the gateway, then U_i updates the information received from the GWN. The whole process is depicted in Table 3.

(1) U_i gets $e_{\mathrm{inew}}'\|k_3'\|M_7' = M_6 \bigoplus e_i$.

(2) U_i computes the shared key SK_{ij}' between U_i and S_j: $\mathrm{SK}_{ij}' = h(k_1 \cdot B) = h(k_1 \cdot k_2 \cdot G)$.

(3) U_i gets the hashed value $M_4' = h(B\|\mathrm{SK}_{ij}'\|A)$ and checks if $M_7 = h(e_{\mathrm{inew}}'\|k_3'\|d_i\|T_1\|M_4')$. If they are equal, U_i accepts SK_{ij}' as the shared key.

(4) U_i updates the identity information $l_i = MP_i' \oplus e_{\mathrm{inew}}'$, $k_i = k_3'$.

3.4. Password Change Phase. To change a user's password, the user first sends a password change request to the SC. After the SC verifies this user, the user can change his password.

In order to prevent the offline password guess attack, in our scheme, the user is only allowed to change his password k times in a time period T. We use a variable counter to record the times a user inputs a wrong password. $\mathrm{TW}_{\mathrm{first}}$ means the first time a user inputs a wrong password. When a user inputs a wrong password more than k times in a time period T, he will not be allowed to input a password anymore in this time period. The whole process is depicted in Figure 2:

(1) User U_i inserts his SC into a card reader and inputs his identity and password: ID_i, PW_i.

(2) SC checks if the counter $\geq k$; if counter $\geq k$, go to step 3. If counter $< k$, go to step 4.

(3) SC continues to check if $|\mathrm{TW}_{\mathrm{first}} - T_{\mathrm{now}}| \leq T$; if $|\mathrm{TW}_{\mathrm{first}} - T_{\mathrm{now}}| \leq T$, user is not allowed to change the password; otherwise, if set counter = 0, go to step 4.

(4) SC computes $h(r_i\|\mathrm{ID}_i\|\mathrm{PW}_i)$ using ID_i and PW_i and the stored r_i. SC compares $h(r_i\|\mathrm{ID}_i\|\mathrm{PW}_i)$ with MP_i stored in the smart card; if they are equal, SC acknowledges the legitimacy of U_i. If they are not equal, go to step 11.

(5) Check if counter == 0? If counter! = 0, set counter = 0.

(6) SC computes $d_i = f_i \bigoplus \mathrm{MP}_i$ using the stored f_i and the user password MP_i.

(7) SC computes $e_i = l_i \oplus \mathrm{MP}_i$ using the stored l_i and the user password MP_i.

(8) User U_i inputs the new password PW_i'.

(9) SC updates MP_i to be $\mathrm{MP}_i' = h(r_i\|\mathrm{ID}_i\|\mathrm{PW}_i')$.

(10) SC uses this new PW_i' to update the stored version of f_i and l_i to get $f_i' = d_i \bigoplus \mathrm{MP}_i'$ and $l_i' = e_i \bigoplus \mathrm{MP}_i'$. Now user has finished the password change phase.

(11) Set counter = counter + 1; if counter == 1, $\mathrm{TW}_{\mathrm{first}}$ is set to be now(), go to the first step.

4. Formal Security Analysis Using ProVerif

ProVerif [23] is an automatic cryptographic protocol verifier, in the formal model (so-called Dolev-Yao model) [24]. It can handle many different cryptographic primitives; it also can handle an infinite number of sessions. We use this tool to prove the secrecy of the shared key and the secrecy of

Table 3: Login and authentication phase.

User ID_i, PW_i	Gateway X_{GWN}	Sensor SID_j, x_j
input ID_i' *and* PW_i'		
$MP_i' = h(r_i \| ID_i' \| PW_i')$		
$d_i = f_i \oplus MP_i'$		
$e_i = l_i \oplus MP_i'$		
random k_1, $A = k_1 \cdot G$		
get k_i, *timestamp* T_1		
$M_2 = h(A \| ID_i \| SID_j \| d_i \| T_1)$		
$M_1 = e_i \oplus (ID_i \| SID_j \| M_2)$		
$\{A, k_i, M_1, T_1\}$ →		
	checks the freshness of T_1	
	$e_i' = h(k_i \| X_{GWN})$	
	$ID_i' \| SID_j' \| M_2' = M_1 \oplus e_i'$	
	$d_i' = h(ID_i' \| X_{GWN})$	
	$M_2'? = h(A \| ID_i' \| SID_j' \| d_i' \| T_1)$	
	$x_j' = h(SID_j' \| X_{GWN})$	
	timestamp T_2	
	$M_3 = h(A \| SID_j' \| x_j' \| T_2)$	
	$\{A, M_3, T_2\}$ →	
		checks the freshness of T_2
		$M_3? = h(A \| SID_j \| x_j \| T_2)$
		random k_2, $B = k_2 \cdot G$
		$SK_{ij} = h(k_2 \cdot A)$
		$M_4 = h(B \| SK_{ij} \| A)$
		$M_5 = h(x_j \| M_3 \| M_4 \| B)$
	$M_5? = h(x_j \| M_3 \| M_4 \| B)$	$\{B, M_4, M_5\}$ ←
	random number k_3	
	$e_{inew} = h(k_3 \| X_{GWN})$	
	$M_7 = h(e_{inew} \| k_3 \| d_i' \| T_1 \| M_4)$	
	$M_6 = (e_{inew} \| k_3 \| M_7) \oplus e_i'$	
$e_{inew}' \| k_3' \| M_7' = M_6 \oplus e_i$	$\{B, M_6\}$ ←	
$SK_{ij}' = h(k_1 \cdot B)$		
$M_4' = h(B \| SK_{ij}' \| A)$		
$M_7'? = h(e_{inew}' \| k_3' \| d_i \| T_1 \| M_4')$		
$l_i = MP_i' \oplus e_{inew}', k_i = k_3'$		

the identity; furthermore, we prove the authentication between the user and the gateway and between the sensor and the gateway. We use ProVerif version 1.98pl1; the simulation was conducted on Ubuntu 14.04 LTS (32-bit) with a memory 1 GB. We show the part of the code implemented in the ProVerif in Appendix A. For more detailed code, please refer to [25].

4.1. Test on the Identity Privacy. To prove that the identity of the user is not known to the attacker, we test the query "*Query not attacker (idi).*" The query result is "*true*," which means the identity of the user is not derivable by the attacker. This proof shows that our scheme can protect the identity privacy of the user. For the protection of the sensor identity, the result is the same. The simulation results are in Box 1.

4.2. Test of the Authentication of the Scheme. In ProVerif, "*Injective correspondence*" is used to capture the authentication in case of a one-to-one relationship. The event "*event acceptUser (bitstring)*" is used by the user to record the belief

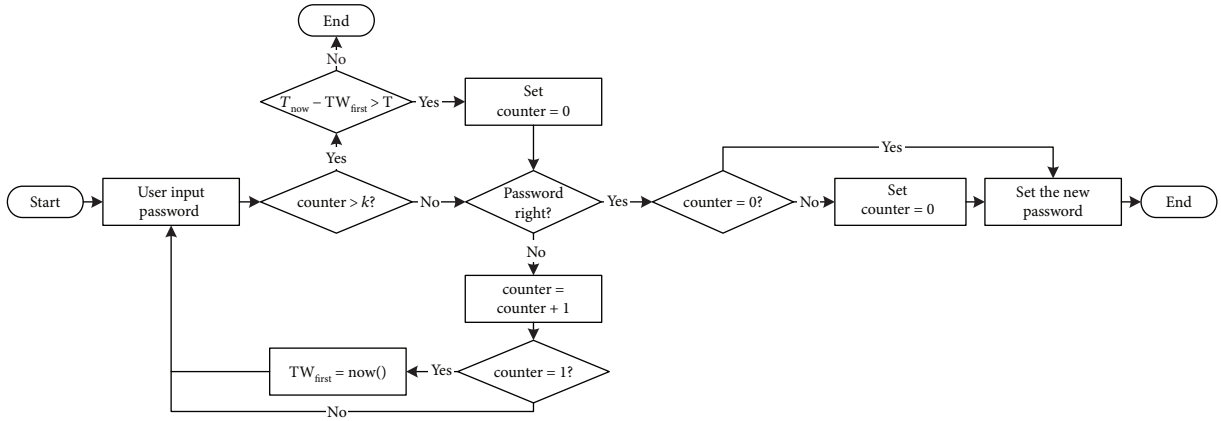

Figure 2: The information flow for a user to change his password.

– Query not attacker(sidj[])……RESULT not attacker(sidj[]) is true.
– Query not attacker(idi[])……RESULT not attacker(idi[]) is true.

Box 1

RESULT inj-event(termGatewaywithSensor(x_71)) ==> inj-event(acceptSensor(x_71)) is true.
RESULT inj-event(termSensor(x_4088)) ==> inj-event(acceptUserbyGateway(x_4088)) is true.
RESULT inj-event(termUser(x_8075)) ==> inj-event(acceptUserbyGateway(x_8075)) is true.
RESULT inj-event(termGatewaywithUser(x_13373)) ==> inj-event(acceptSensor(x_13373)) is true.

Box 2

– Query not attacker(skijs[])……RESULT not attacker(skijs[]) is true.
– Query not attacker(skiju[])……RESULT not attacker(skiju[]) is true.

Box 3

that the user has accepted to run the protocol with the gateway and with the supplied symmetric key. The event "*event termUser(bitstring)*" means that the user believes he has terminated a protocol run using the data type "*bitstring*." The other events have similar meanings. These queries ensure the authentication between the users and the gateway and between the sensors and the gateway (see Box 2).

What is more, we prove the secrecy of the shared key by the queries "*Query not attacker (skijs[])*" and "*Query not attacker(skiju[])*." The result is "true" as shown in the following. The user calculates the shared key between the user and sensor as "*skijs*;" the sensor calculates the shared key as "*skiju*" (see Box 3).

5. AVISPA Verification

AVISPA (Automated Validation of Internet Security Protocols and Applications) is "a push-button tool for the automated validation of Internet security-sensitive protocols and applications" [26]. The AVISPA project aims at developing a push-button, industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications. We write the scheme in HLPSL, which is a role-based language designed explicitly for AVISPA. The code is in Appendix B; we have uploaded the code to [25].

In the HLPSL, the confidentiality goals of the protocol are set to be "*sc_sensor_id*" and "*sc_user_id*," which can ensure the confidentiality of the user identity and the sensor identity. The message authentication goal is set to be "*shared_key*," which can enable the authentication of the shared key. This goal ensures that the users and sensors build a shared key with the help of the gateway.

The running result of the protocol is shown in Table 4. We run the security check based on the CL-based Model-Checker [27] and the On-the-Fly Model-Checker OFMC [28, 29]. The CL-based Model-Checker (CL-AtSe) translates protocol written as transition relation in the IF into a set of constraints which can be used efficiently to find attacks on protocols. While OFMC can be employed not only for efficient falsification of protocols but also for verification, without bounding the messages, an intruder can generate. Both

Table 4: Simulation results.

CL-AtSe back end	OFMC
SUMMARY	% OFMC
SAFE	% Version of 2006/02/13
	SUMMARY
DETAILS	SAFE
BOUNDED_NUMBER_OF_SESSIONS	DETAILS
TYPED_MODEL	BOUNDED_NUMBER_OF_SESSIONS
	PROTOCOL
PROTOCOL	/home/yuwen/avispa/avispa-1.1/testsuite/results/usglight.if
/home/yuwen/avispa/avispa-1.1/testsuite/results/usglight.if	GOAL
GOAL	as_specified
As Specified	BACKEND
	OFMC
BACKEND	COMMENTS
CL-AtSe	STATISTICS
	parseTime: 0.00s
STATISTICS	searchTime: 0.02 s
Analysed: 5 states	visited Nodes: 4 nodes
Reachable: 0 states	
Translation: 0.02 seconds	depth: 2 plies
Computation: 0.00 seconds	

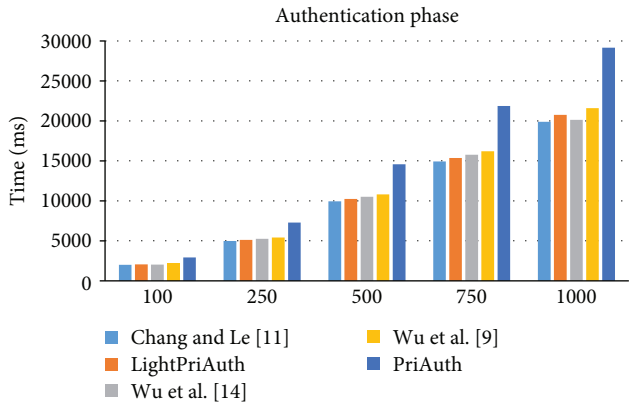

Figure 3: The computation time of different schemes.

Table 5: Computation time of different operations.

Operations	Time (ms)
T_H: one-way hash function	0.0004
$T_{E/D}$: AES encryption/decryption	0.1303
T_{MUL}: ECC multiplications	7.3529

of the two back-end verification tools show that our scheme is safe.

6. Comparison

In this section, we compared our scheme with the other three schemes [1, 9, 14]. We compared them in two folds: computation performance and communication performance.

6.1. Computation Performance. The typical way to compute the execution time of the protocol is to calculate the protocol's computational costs of different operations; the operations' execution time is measured by simulation. In this study, the execution time of the XOR operation is minimal compared to an elliptic curve point multiplication or hash operation, and we neglect it when computing the time approximately [11]. In this section, we first compare different schemes using a benchmark from one previously published paper. Then, we simulate the computation time of these schemes in C++; the result is shown in Figure 3.

The benchmark of MIRACL C/C++ Library used in this study can be found at [9]; we list the results in Table 5. Based on this benchmark, the computation costs of different schemes are calculated; the result is in Table 6. At the user side, our scheme only needs 2 ECC multiplications and 5 hash operations. At the sensor side, our scheme costs 2 ECC multiplications and 5 hash operations, and at the gateway side, our scheme costs 8 hash operations. Our scheme costs the least time at the user side and gateway side. And at the sensor side, our scheme costs the second least time. In all, our scheme costs the least computation time.

In the scheme [9], extra AES encryption/decryption is needed. User, sensor, and gateway need 1, 1, and 2 AES encryptions/decryptions separately. Their proposed scheme needs 6, 3, and 9 more hash operations than our scheme at the user side, gateway side, and in total, respectively.

The scheme [11] needs 2, 1, 1, and 4 more hash operations than our scheme at the user side, sensor side, sensor side, gateway side, and in total, respectively. In PriAuth, the asymmetric encryption method is needed to encrypt the identity of the user. The user and gateway in this scheme both

Table 6: Computation cost of the login and authentication.

Schemes	User	Sensor	Gateway	Total	Total (ms)
Wu et al. [9]	$11T_H + 2T_{\text{MUL}} + 1T_{E/D}$	$4T_H + 2T_{\text{MUL}} + 1T_{E/D}$	$11T_H + 2T_{E/D}$	$26T_H + 4T_{\text{MUL}} + 4T_{E/D}$	29.9432
Wu et al. [14]	$13T_H + 2T_{\text{MUL}}$	$4T_H + 2T_{\text{MUL}}$	$14T_H$	$31T_H + 4T_{\text{MUL}}$	29.4240
Chang and Le [11]	$7T_H + 2T_{\text{MUL}}$	$5T_H + 2T_{\text{MUL}}$	$9T_H$	$21T_H + 4T_{\text{MUL}}$	29.42
Chen et al. [1]	$5T_H + 3T_{\text{MUL}}$	$\mathbf{3T_H + 2T_{MUL}}$	$7T_H + 1T_{\text{MUL}}$	$15T_H + 6T_{\text{MUL}}$	44.1234
LightPriAuth	$\mathbf{5T_H + 2T_{MUL}}$	$4T_H + 2T_{\text{MUL}}$	$\mathbf{8T_H}$	$\mathbf{17T_H + 4T_{MUL}}$	**29.4184**

In this table, the boldface ones are the ones with the least computation time.

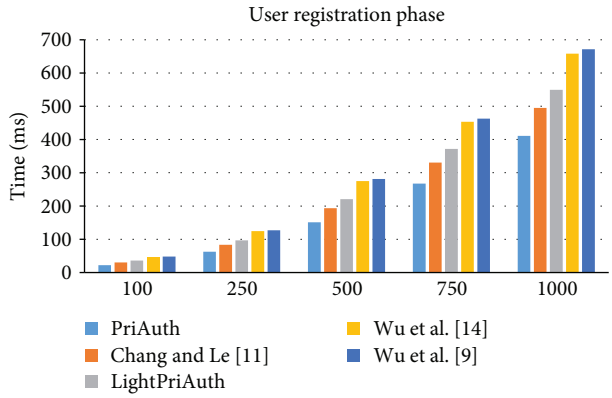

Figure 4: The user registration time of different schemes.

need one more ECC multiplication; in total, PriAuth needs two more ECC multiplications than our scheme. The scheme of Wu et al. [14] is the most similar one with the proposed scheme; however, compared to our scheme, their proposed scheme needs 8, 6, and 14 more hash operations than our scheme at the user side, gateway side, and in total, respectively.

We implement these four different schemes in C++; the running codes are stored at a public repository in http://github.com [25]. We use the MIRACL C/C++ Library [30]. The experiment is conducted in Visual Studio C++ 2017 on a 64-bit Windows 7 operating system, 3.5 GHz processor, 8 GB memory. The hash function is SHA-256, the symmetric encryption/decryption function is AES in MR_PCFB1 form, and the 256-bit-long key for symmetric encryption/decryption function is generated by SHA-256 hash operation. We use the Curve P-192 provided by NIST Digital Signature Standard [31]. The parameters are listed in Appendix C.

The code is compiled in x86 form, and the simulation does not take account of the transmission of the data. We run the login and authentication phase of different schemes 100, 250, 500, 750, and 1000 times. The result is shown in Figure 3. In this figure, the horizontal axis indicates the times the experiment is run and the vertical axis indicates the milliseconds to accomplish the experiment. Our scheme is the second-best one, and the computation time of the PriAuth is the longest.

We run the user registration phase of different schemes. The number of users in the registration phase is set to be 100, 250, 500, 750, and 1000. The result is shown in Figure 4. In this figure, the horizontal axis indicates the number of users and the vertical axis indicates the milliseconds needed to accomplish the experiment. Under all experimental conditions, the running time of the PriAuth is the shortest. Our scheme is the second-best scheme, which is about 1.5 times that of the PriAuth. However, this ratio becomes much smaller when the user number increases. Why did this happen? The computation time is mainly composed of two parts: the hash operation time and the checking time. The numbers of hash operation are listed in Table 7. The checking is performed by the gateway to determine if the user has registered before. The gateway keeps a list of registered users' identity; when the gateway receives a registration request, it has to search the list to check if this user has registered or not. The hash operation time and the checking time are close when the number of the user is smaller. However, the time difference becomes huge with the increasing user numbers.

The running time of the other two schemes is about 2.5 times that of PriAuth. The ratio of hash operations between them is roughly the same as that of the running time. In Table 7, the ratio is computed using the formula Ratio = h_i/H, where H means the number of hash operations needed by PriAuth. h_i means the number of hash operations needed by the other schemes.

We run the sensor registration phase of different schemes. The number of sensors in the registration phase is set to be 100, 250, 500, 750, and 1000. The result is shown in Figure 5. In this figure, the horizontal axis indicates the number of sensors and the vertical axis indicates the milliseconds needed to accomplish the experiment. The running time of our scheme is close to the running time of Wu et al. [9, 14] and Chang and Le [11]; this is mainly because these three schemes need only 1 hash operation in the sensor registration phase. As PriAuth costs 7 hash operations in the sensor registration phase, the running time is close to 7 times that of the other three schemes. The computation time of the PriAuth is the longest. The running time of Wu et al. [9] is a little more than our scheme (Wu et al. [14] and Chang and Le [11]). This is because at the sensor registration phase, the input of the hash operation of our scheme, of Wu et al.'s scheme [14], and of Chang and Le's scheme [11] is the sensor's identity and the gateway's private key, while the input of the hash operation of Wu et al. [9] is the sensor's identity, gateway's private key, and gateway's identity; the hash operation's input is longer.

6.2. Communication Performance. In this section, the communication performance is compared. The identity is set to 8 bytes long [32]. The size of the timestamp is set to 4 bytes

Table 7: The cost of hash operations in the registration phase.

Phase	User registration				Sensor registration			
Entity	User	Gateway	Total	Ratio	Sensor	Gateway	Total	Ratio
Wu et al. [9]	2	3	5	2.5	0	1	1	1/7
Wu et al. [14]	2	3	5	2.5	0	1	1	1/7
Chang and Le [11]	1	2	3	1.5	0	1	1	1/7
Chen et al. [1]	1	1	2	1	3	4	7	1
LightPriAuth	1	2	3	1.5	0	1	1	1/7

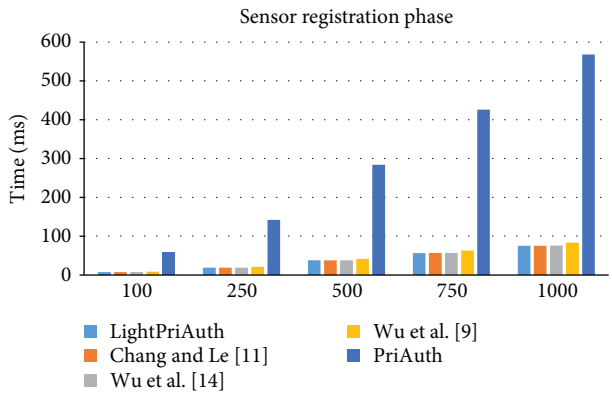

Figure 5: The sensor registration phase of different schemes.

Table 8: Communication comparison.

	Hash	ECC	Id	T	R	Total bytes	Com
Wu et al. [9]	10	4	4	0	1	564	+168
Wu et al. [14]	13	4	2	0	0	624	+228
Chang and Le [11]	9	2	1	5	0	412	+16
Chen et al. [1]	9	4	0	3	0	492	+96
LightPriAuth	5	4	2	2	1	396	0

Hash: means a general result data; ECC: means a random point on the elliptic curve, Id: means the identity of a sensor or a user; T: means a timestamp; R: means a general random number; Com: comparison between our scheme and the other schemes.

[33]. Moreover, the byte length of a random number is set to be 20 bytes [9]. The result of SHA-256 is 256 bits, which is 32 bytes. The sizes of a point on the elliptic curve with a 192-bit elliptic curve is 384 bits, which is 48 bytes [9]. The sum of each type of variable length in bytes is calculated for comparison of the communication cost.

Table 8 shows the number of different types of data used in the scheme. It is not hard to find that the communication cost of our scheme is the least. The cost of our scheme, LightPriAuth, is 396 bytes; the costs of Wu et al.'s [9, 14], Chang and Le's [11] and Chen et al.'s [1] schemes are 564, 624, 412, and 492 bytes, respectively; they are 168, 228, 16, and 96 bytes higher, respectively, than the proposed scheme. The main reason is that LightPriAuth transmits only 5 hash result data. While the other schemes of Wu et al. [9, 14], Chang and Le [11], and Chen et al. [1] have to transmit 10, 13, 9, and 9 hash result data, respectively. They are 5, 8, 4, and 4 more, respectively, than the proposed scheme.

7. Other Security Feature Analyses

In this section, we analyze the security features of different schemes. At the end of this section, we conclude the results into a table.

7.1. User Anonymity/Sensor Anonymity. Regarding user anonymity, we find that all the schemes could enable user anonymity, as the identities of the users are encrypted. For sensor anonymity, the identity of the sensor is transmitted transparently in the scheme [11, 14]; adversaries could get the identity easily.

7.2. User Anonymity to Sensor. In the scheme of [9], the identity of the user is sent to the sensors directly; once a user accesses a sensor's data, this user's identity is known by the sensor; the sensor can learn the identity of the user. Apparently, this is not good for the identity privacy of the user. In the proposed scheme, LightPriAuth, the identity of the user need not be sent to the sensor; thus, this could avoid the potential identity leaking problem. We describe this "user anonymity to sensor."

7.3. Loss of Synchronization Problem. Similar to the scheme [3], when a user logs in, the gateway will generate a new identity for the user and the old identity will not be used anymore. However, if adversaries block this identity from being sent to the user, the user cannot receive this identity, when he logs in the next time using the old identity, he will not be treated as a legal user anymore. The scheme in [14] has this problem.

7.4. Offline Dictionary Attack. For most of the schemes, an adversary is unable to launch an offline dictionary attack in the login and authentication phase. However, an adversary is able to launch an offline dictionary attack in the password-changing phase.

In the password change phase of [19], if the adversary types in a random identity ID_i' and a random password PW_i', he will get a reply from the SC. Based on the replied message, the adversary can judge if the identity and password are correct or not. If the adversary guesses a correct key pair by accident, then he could set a new password. Thus, the adversary is able to launch an offline dictionary attack.

In the proposed scheme, we set a limitation on the user, if the user inputs a wrong identity and password pair more than k times in a time period T, he is not allowed to log in in this period of time. Thus, our scheme can avoid the offline dictionary attack in the password change phase.

Table 9: Security feature comparison.

Security feature	Wu et al. [9]	Wu et al. [14]	Chang and Le [11]	Chen et al. [1]	LightPriAuth
User anonymity	✓	✓	✓	✓	✓
User being tracked	✓	X	X	✓	✓
Sensor anonymity	✓	X	X	✓	✓
User anonymity to sensor	X	✓	✓	✓	✓
Loss of synchronization	✓	X	✓	✓	✓
With timestamp	X	X	✓	✓	✓
Offline dictionary attack	X	X	X	X	✓
Computation cost (ms)	29.9432	29.424	29.42	44.1234	29.4184
Communication cost (bytes)	564	624	412	492	396

```
(* Role of the user*)
let processUser(idi: bitstring, sidj: bitstring, pwi: bitstring) =
   (* registration phase of the user *)
      new ri:bitstring;
   let mpi = hash(con(ri,con(idi,pwi))) in
   out(scUser,(idi,mpi));
   in(scUser,(fi:bitstring,li:bitstring,ki:bitstring));
   let (ei:bitstring) = xor(li, mpi) in
      let (di:bitstring) = xor(fi, mpi) in
      (* Real start of the role *)
         (* Message 1*)
         new k1:exponent;
         let A = exp(g,k1) in
         new T1:bitstring;
         let m2 = hash(con(g2 h(A),con(idi,(con(sidj,con(di,T1)))))) in
         let m1 = xor(ei,con(idi,con(sidj,m2))) in
         out(cug, (A, ki, m1, T1));
         (* Message 4 *)
         in(cug, (B:G, M6:bitstring));
         let (tem:bitstring) = xor(M6, ei) in
         let (einew':bitstring, tem1: bitstring) = Split(tem) in
         let (k3':bitstring, m7':bitstring) = Split(tem1) in
         let skiju = hash(g2h(exp (B, k1))) in
         let m4' = hash(con(g2h(B),con(skiju,g2h(A)))) in
         if (m7') = hash(con(einew',con(k3',con(di,con(T1,m4'))))) then
                  event acceptUser(di);
            let li = xor(mpi,einew') in
            let ki = k3' in
                  event termUser(di).
```

Algorithm 1

7.5. Security Feature Comparison. Finally, we get in Table 9 the comparison of security features; we can find that compared to other schemes, the proposed scheme has more security features. Besides, the computation cost and the communication cost of the proposed scheme are lower.

8. Conclusions

With different sensors collecting different data around us, it is vital not only to ensure the safety of these data but also to protect the privacy of the data. In this paper, we propose an authentication and key establishment scheme between users and sensors. We analyzed the security features using ProVerif and AVISPA; the formal verifications show that the proposed scheme has achieved all the desired security features. Through comparison, we find that the proposed scheme is comparable to the related works regarding the computation cost and more efficient in communication cost. Our work is part of the LifeWear project, in which we focus on the safety of data transmission and identity privacy problem.

```
(* Role of the Sensor*)
let processSensor(sidj: bitstring) =
      (* registration phase of the sensor *)
      out(scSensor,sidj);
   in (scSensor,xj: bitstring);
      (* Real start of the role *)
   (* Message 2*)
   in(csg,(A:G,m3:bitstring,T2:bitstring));
   (* Message 3*)
   if (m3) = hash(con(g2h(A),con(sidj,con(xj,T2)))) then
      event acceptSensor(xj);
   new k2:exponent;
   let B = exp (g, k2) in
   let skijs = hash(g2h(exp (A, k2))) in
   let m4 = hash(con(g2h(B),con(skijs,g2h(A)))) in
   let m5 = hash(con(xj,con(m3,con(m4,g2h(B))))) in
   out(csg,(B,m4,m5));
      event termSensor(xj).
```

Algorithm 2

```
(*gateway *)
let processGateway(xgwn: bitstring) =
   (* Message 2*)
   in(cug,(A:G, ki:bitstring, M1:bitstring, T1:bitstring));
   let ei' = hash(con(ki,xgwn)) in
   let (tem:bitstring) = xor(M1, ei') in
   let (idi':bitstring, tem1: bitstring) = Split(tem) in
   let di'= hash(con(idi',xgwn)) in
   let (sidj': bitstring, m2': bitstring) = Split(tem1) in
   if (m2') = hash(con(g2h(A),con(sidj',hash(con(idi',xgwn))))) then
      event acceptUserbyGateway(di');
   new T2: bitstring;
   let xj'=hash(con(sidj',xgwn)) in
   let m3 = hash(con(g2h(A),con(sidj',con(xj',T2)))) in
   out(csg,(A,m3,T2));
   (* Message 4*)
   in(csg,(B:G, m4:bitstring, m5:bitstring));
   if (m5) = hash(con(xj',con(m3,con(m4,g2h(B))))) then
   new k3:bitstring;
   let einew = hash(con(k3,xgwn)) in
   let m7 = hash(con(einew,con(k3,con(di',con(T1,m4))))) in
      event acceptSensorbyGateway(xj');
      event termGatewaywithSensor(xj');
   let m6 = xor(con(einew,con(k3,m7)),ei') in
   out(cug,(B,m6));
      event termGatewaywithUser(di').
(* User registration *)
let processUserRegistration(xgwn: bitstring) =
         in(scUser, (idi: bitstring, mpi:bitstring));
   let di = hash(con(idi,xgwn)) in
   let fi = xor(di,mpi) in
   new ki:bitstring;
      let ei= hash(con(idi,xgwn)) in
      let li= xor(ei,mpi) in
   out(scUser,(fi,li,ki)).
(* Sensor registration *)
```

Algorithm 3: Continued.

```
let processSensorRegistration(xgwn: bitstring) =
   in(scSensor, sidj:bitstring);
   let xj = hash(con(sidj,xgwn)) in
   out(scSensor,xj).
```

Algorithm 3

```
role user (Ui, Sj, GW: agent,
           Kdi,Kei: symmetric_key,
           H: hash_func
           P: text,
           SND_US,RCV_US: channel (dy))
played_by Ui
def=
  local State: nat,
  T1,K1,A,M1,M2,Ki,B,K3,K2,IDi,SIDj,M4,M6,M7,SK,Keinew: text
  const shared_key,sc_user_id,sc_sensor_id:protocol_id
  init State:= 0
  transition
      1. State = 0        RCV_US(start)=|>
      State': = 2        ⋀ T1':= new()
                         ⋀ K1':= new()
                         ⋀ A':= exp(P,K1')
                         ⋀ M2':= H(A'.IDi.SIDj.Kdi.T1')
                         ⋀ M1':= xor(Kei,(IDi.SIDj. H(A'.IDi.SIDj.Kdi.T1')))
                         ⋀ SND_UG(A'.M1'.Ki.T1')
      2. State = 7       ⋀ RCV_UG(B′
                                  .xor((Keinew'.K3'.H(Kei.K3'.Kdi.T1.M4')),Kei)
                                  )=|>
      State':= 8        ⋀ Kei':= Keinew'
                         ⋀ Ki':= K3'
                         ⋀ SK':= H(exp(B′,K1))
                         ⋀ witness(Ui,Sj,user_sensor_sk,SK')
                         ⋀ request(Ui,Sj,user_sensor_sk,SK')
    end role
```

Algorithm 4

```
role sensor (Ui, Sj, GW: agent,
             Kxj: symmetric_key,
             H: hash_func,
             P: text,
             SND_SG,RCV_SG: channel(dy))
played_by Sj
def=
  local State: nat,
  T2,K2,A,B,SK,M3,M4,M5,SIDj :text
  const shared_key:protocol_id
  init State:= 4
  transition
    1. State = 4      ⋀ RCV_SG(A'
                               .H(A'.SIDj'.Kxj.T2')
                               .T2'
                               )=|>
```

Algorithm 5: Continued.

```
        State':= 3        ⋀ K2':= new()
                            ⋀ B′:= exp(P,K2')
                            ⋀ SK':= H(exp(A',K2))
                            ⋀ M4':= H(B′.SK.A')
                            ⋀ M5':= H(Kxj.H(A'.SIDj'.Kxj.T2').M4'.B′)
                            ⋀ SND_SG(B′.M4'.M5')
                            ⋀ witness(Ui,Sj,shared_key,SK')
                            ⋀ request(Ui,Sj,shared_key,SK')
    end role
```

Algorithm 5

```
role gateway (Ui, Sj, GW: agent,
              Kdi,Kei, Kxj, Xgwn: symmetric_key,
              H: hash_func,
              SND_UG,RCV_UG,SND_SG,RCV_SG: channel(dy))
played_by GW
def=
  local State: nat,
  T1,T2,A,B,IDi,SIDj,M2,M3,M4,M6,M7,Ki,Keinew,K3: text
  const sc_user_id,sc_sensor_id:protocol_id
  init State: = 2
  transition
  1. State = 2     ⋀ RCV_UG(A'.
                          xor(H(Ki'.Xgwn),(IDi'.SIDj'.H(A'.IDi'.SIDj'.Kdi.T1'))).
                          Ki'.
                          T1'
                          )=|>
  State':= 3       ⋀ T2': = new()
                     ⋀ M3':= H(A'.SIDj'.Kxj.T2')
                     ⋀ SND_SG(A'.M3'.T2')
                     ⋀ secret(IDi,sc_user_id,{Ui,GW})
                     ⋀ secret(SIDj,sc_sensor_id,{Ui,GW})
  2. State = 5     ⋀ RCV_SG(B′
                          .M4'
                          .H(Kxj.M3.M4'.B′)
                          )=|>
  State':=6        ⋀ K3':= new()
                          ⋀ Keinew':= H(K3'.Xgwn)
                          ⋀ M7':= H(Kei.K3'.Kdi.T1.M4')
                          ⋀ M6':= xor((Keinew'.K3'.M7'),Kei)
                          ⋀ SND_UG(B′.M6')
end role
```

Algorithm 6

```
role session(Ui, Sj, GW: agent,
             Kdi,Kei, Kxj,Xgwn: symmetric_key,
             H: hash_func,
             P: text
             )
def=
  local      SSG,RSG,
             SUG,RUG:channel(dy)
  composition
     user(Ui,Sj,GW, Kdi,Kei,              H,P, SUG,RUG)
⋀ sensor(Ui,Sj,GW, Kxj,                  H,P, SSG,RSG)
⋀ gateway(Ui,Sj,GW, Kdi,Kei,Kxj,Xgwn,   H,   SUG,RUG,SSG,RSG)
end role
```

Algorithm 7

```
role environment()
def=
   const ui, sj, gw: agent,
         kdi,kei, kxj,xgwn, kig,keig: symmetric_key,
         shared_key, sc_user_id,
         sk_sensor_gwn, sc_sensor_id: protocol_id,
         h,expp: hash_func,
         p: text
intruder_knowledge={ui, sj, gw, kig, h, p}
composition
         session(ui,sj,gw,  kdi,kei,kxj,xgwn,  h,p)
      ⋀ session(ui, i,gw,  kdi,kei,kig,xgwn,  h,p)
      ⋀ session( i,sj,gw,  kig,keig,kxj,xgwn, h,p)
end role
```

Algorithm 8

```
% Confidentiality (G12)
   secrecy_of sc_sensor_id,sc_user_id
% Message authentication (G2)
   authentication_on shared_key
```

Algorithm 9

```
A = -3
B = 64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1
P = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF
Gx = 188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012
Gy = 07192B95FFC8DA78631011ED6B24CDD573F977A11E794811
```

Algorithm 10

Appendix

A. Appendix

Algorithm 1 describes the role of the user.

Algorithm 2 describes the role of the sensor.

Algorithm 3 describes the role of the gateway.

B. Appendix

Note that we write our scheme in HLPSL from the authentication phase; supposing that the users and sensors have registered at the gateway secretly, and successfully get the registered information, the role of the user is described in Algorithm 4.

The role of the sensor is described in Algorithm 5.
The role of the gateway is described in Algorithm 6.
The role of the session is described in Algorithm 7.
The role of the environment is described in Algorithm 8.

The role of the goal is divided into two parts. The first part is the "*secrecy_of sc_sensor_id*" and "*sc_user_id*"; this means we want to keep the identity of the user and sensor confidential between them and the gateway. The second part "*authentication_on user_sensor_sk*" means the authentication of the shared key between a user and a sensor (Algorithm 9).

C. Appendix

The parameters of the Curve P-192 by NIST is described in Algorithm 10.

Data Availability

The experimental data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work presented in this paper has been supported by the LifeWear project (funded by the Spanish Ministry of Industry, Energy and Tourism with Reference TSI-010400-2010-100). The work has also been supported by the Chinese Scholarship Council (CSC) with File no. 201507040027.

References

[1] Y. Chen, J.-F. Martínez, P. Castillejo, and L. López, "A privacy protection user authentication and key agreement scheme tailored for the Internet of things environment: PriAuth," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 5290579, 17 pages, 2017.

[2] J. Rodríguez-Molina, J.-F. Martínez, P. Castillejo, and L. López, "Combining wireless sensor networks and semantic middleware for an Internet of things-based sportsman/woman monitoring application," *Sensors*, vol. 13, no. 2, pp. 1787–1835, 2013.

[3] X. Li, J. Niu, S. Kumari, J. Liao, W. Liang, and M. K. Khan, "A new authentication protocol for healthcare applications using wireless medical sensor networks with user anonymity," *Security and Communication Networks*, vol. 9, no. 15, 2655 pages, 2016.

[4] X. Li, M. H. Ibrahim, S. Kumari, A. K. Sangaiah, V. Gupta, and K.-K. R. Choo, "Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks," *Computer Networks*, vol. 129, pp. 429–443, 2017.

[5] S. Kumari and M. K. Khan, "Cryptanalysis and improvement of "a robust smart-card-based remote user password authentication scheme"," *International Journal of Communication Systems*, vol. 27, no. 12, pp. 3939–3955, 2014.

[6] Q. Jiang, J. Ma, Z. Ma, and G. Li, "A privacy enhanced authentication scheme for telecare medical information systems," *Journal of Medical Systems*, vol. 37, no. 1, p. 9897, 2013.

[7] Q. Jiang, J. Ma, G. Li, and L. Yang, "An efficient ticket based authentication protocol with unlinkability for wireless access networks," *Wireless Personal Communications*, vol. 77, no. 2, pp. 1489–1506, 2014.

[8] M. H. Ibrahim, S. Kumari, A. K. Das, M. Wazid, and V. Odelu, "Secure anonymous mutual authentication for star two-tier wireless body area networks," *Computer Methods and Programs in Biomedicine*, vol. 135, pp. 37–50, 2016.

[9] F. Wu, L. Xu, S. Kumari, and X. Li, "A new and secure authentication scheme for wireless sensor networks with formal proof," *Peer-to-Peer Networking and Applications*, vol. 10, no. 1, pp. 16–30, 2017.

[10] A. K. Das, "A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 9, no. 1, pp. 223–244, 2016.

[11] C. C. Chang and H. D. Le, "A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 357–366, 2016.

[12] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of things notion," *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.

[13] R. Amin and G. P. Biswas, "A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks," *Ad Hoc Networks*, vol. 36, Part 1, pp. 58–80, 2016.

[14] F. Wu, L. Xu, S. Kumari, and X. Li, "A privacy-preserving and provable user authentication scheme for wireless sensor networks based on Internet of things security," *Journal of Ambient Intelligence and Humanized Computing*, vol. 8, no. 1, pp. 101–116, 2017.

[15] Y. Choi, D. Lee, J. Kim, J. Jung, J. Nam, and D. Won, "Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography," *Sensors*, vol. 14, no. 6, pp. 10081–10106, 2014.

[16] J. L. Tsai and N. W. Lo, "Secure anonymous key distribution scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 906–914, 2016.

[17] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1900–1910, 2018.

[18] J. Liu, Z. Zhang, X. Chen, and K. S. Kwak, "Certificateless remote anonymous authentication schemes for wirelessbody area networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 332–342, 2014.

[19] C. Wang, G. Xu, and J. Sun, "An enhanced three-factor user authentication scheme using elliptic curve cryptosystem for wireless sensor networks," *Sensors*, vol. 17, no. 12, p. 2946, 2017.

[20] C.-T. Li, T.-Y. Wu, C.-L. Chen, C.-C. Lee, and C.-M. Chen, "An efficient user authentication and user anonymity scheme with provably security for IoT-based medical care system," *Sensors*, vol. 17, no. 7, p. 1482, 2017.

[21] J. L. Tsai and N. W. Lo, "A privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. 9, no. 3, pp. 805–815, 2015.

[22] K.-A. Shim, "S^2DRP: secure implementations of distributed reprogramming protocol for wireless sensor networks," *Ad Hoc Networks*, vol. 19, pp. 1–8, 2014.

[23] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proceedings. 14th IEEE Computer Security Foundations Workshop*, pp. 82–96, Cape Breton, NS, Canada, June 2001.

[24] http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. February 5, 2018.

[25] https://github.com/SevenBruce/UAuth. March 1, 2018.

[26] A. Armando, D. Basin, Y. Boichut et al., "The AVISPA tool for the Automated Validation of Internet Security Protocols and Applications," in *Computer Aided Verification. CAV 2005. Lecture Notes in Computer Science*, pp. 281–285, Springer Berlin Heidelberg, 2005.

[27] M. Turuani, "The CL-Atse Protocol Analyser," in *Term Rewriting and Applications. RTA 2006. Lecture Notes in Computer Science*, F. Pfenning, Ed., Springer, Seattle, WA, USA, 2006.

[28] D. Basin, S. Mödersheim, and L. Viganò, "Constraint differentiation: a new reduction technique for constraint-based analysis of security protocols," *Proceedings of CCS'03*, V. Atluri and P. Liu, Eds., , pp. 335–344, ACM Press, 2003, http://www.avispa-project.org.

[29] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: a symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, no. 3, pp. 181–208, 2005.

[30] https://libraries.docs.miracl.com/miracl-user-manual/about. March 1, 2018.

[31] https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf. April 3, 2018.

[32] M. S. Farash, M. Turkanović, S. Kumari, and M. Hölbl, "An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the Internet of things environment," *Ad Hoc Networks*, vol. 36, Part 1, pp. 152–176, 2016.

[33] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.