J Ambient Intell Human Comput (2017) 8:101–116
DOI 10.1007/s12652-016-0345-8

ORIGINAL RESEARCH

A privacy-preserving and provable user authentication scheme for wireless sensor networks based on Internet of Things security

Fan Wu[1] · Lili Xu [2] · Saru Kumari[3] · Xiong Li[4,5]

Received: 21 October 2015 / Accepted: 13 January 2016 / Published online: 8 February 2016

Abstract The notion Internet of Things (IoT) means all things in the global network can be interconnected and accessed. Wireless sensor network (WSN) is one of the most important applications of the notion and is widely used in nearly all scopes. In 2014, Hsieh et al. presented an improved authentication scheme for WSNs. But it has several weaknesses, including no session key, lack of mutual authentication and under the insider attack, the off-line guessing attack, the user forgery attack and the sensor capture attack. To avoid the weaknesses, we present a new authentication scheme which is also for WSNs. Then we employ the random oracle model to show the formal proof, and use the protocol analyzing tool Proverif to list the formal verification process. Compared with some recent schemes for WSNs via the aspects of security properties, the proposed scheme overcomes the common problems and fits for the security properties of IoT.

Keywords Internet of Things · Wireless sensor network · Mutual authentication · Formal proof · Smart card

1 Introduction

Internet of Things (IoT) is a popular notion by which we are surrounded in our lives. With a technical view, IoT means the interconnection of every embedded computing device with a unique identity in the Internet covering many sorts of domains, protocols and applications. It satisfies the needs between the development of market, user and information. The appearance of Internet, wireless network and micro-electromechanical systems makes this notion be true and widely applied.

Wireless sensor network (WSN) is one of the most important parts of IoT. The early research focused on the WSNs including a lot of immobile and homogeneous wireless sensors used in a special way. But such description does not meet the development. Nowadays, sensor networks are built with many kinds of sensor nodes which have their own abilities. That is to say, the WSN with heterogeneity is a widespread view in life and it is widely used in many fields, like industrial work monitoring and control, remote health care for patients and alarm for guarding or fire. The wireless sensor network includes three basic parts: the users, the gateway and the sensors. The gateway is the most important core device which is responsible for the security of the wireless sensor network. The users and the sensors must register on it. The gateway can reach and communicate with all the sensors. Users who want the data collected by the sensors should contact the

✉ Fan Wu
conjurer1981@gmail.com

Saru Kumari
saryusiirohi@gmail.com; saru@ccsuniversity.ac.in

Xiong Li
lixiongzhq@163.com

[1] Department of Computer Science and Engineering, Xiamen Institute of Technology, Xiamen 361021, China

[2] School of Information Science and Technology, Xiamen University, Xiamen 361005, China

[3] Department of Mathematics, Ch. Charan Singh University, Meerut 250005, Uttar Pradesh, India

[4] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

[5] Nanjing University of Information Science and Technology, Nanjing 210044, China

gateway. Using an encryption key in the session is a usual method. The main question is how to establish a common session key between the user and the sensor which the user wants to access. If anyone wants to get data from one particular sensor of a WSN, he should first be authenticated to make the access legally. The common way is to do mutual authentication between the user and the special sensor. Researchers have done much work for it in the application layer.

1.1 Related work

As a method to guarantee the security in communication, authentication is usually employed by the researchers, like signatures (Guo et al. 2014; Ren et al. 2015; Wu and Xu 2015) and key agreement (Li et al. 2013a, b; Xu and Wu 2015a, b; Wu et al. 2015b, c). In the recent decade, many authentication schemes for WSN have been proposed (Watro et al. 2004; Das 2009; Khan and Alghathbar 2010; Vaidya et al. 2010; Chen and Shih 2010; Yeh et al. 2011; Yoo et al. 2012; Xue et al. 2013; Choi et al. 2014; Shi and Gong 2013; Hsieh and Leu 2014; Turkanović et al. 2014; Farash et al. 2015; Chang and Le 2015; Wu et al. 2015a). In 2004, (Watro et al. 2004) presented a user authentication scheme for WSN, which was based on RSA and with a name TinyPK. But in 2009, (Das 2009) showed that sensor forgery attacks could be done on the scheme in Watro et al. (2004), and he also proposed an efficient authentication scheme using smart cards. Only hash functions are employed in the scheme. But in 2010, it was criticized by Chen and Shih (2010), He et al. (2010), Khan and Alghathbar (2010) and Vaidya et al. (2010), respectively, due to some security flaws, such as destitute of mutual authentication, and under the impersonation attack and the insider attack. Moreover, (Vaidya et al. 2010) also pointed out that the scheme in (Khan and Alghathbar 2010) was vulnerable to the stolen smart card attack and the sensor node capture attack. Then they presented an enhanced scheme. Kumar and Lee (2011) showed that the scheme in He et al. (2010) was susceptible to information leakage attack, and the following security properties were destitute, including user anonymity, mutual authentication and constructing a session key between the user and the sensor.

According to the review paper Hayouni et al. (2014), elliptic curve cryptography (ECC) has already been used for WSNs in recent years. Some new technology for improving the performance of ECC in sensors appears, like (Liu et al. 2014). In 2011, (Yeh et al. 2011) pointed out that the scheme proposed by Chen and Shih (2010) was under the insider attack and without a password change part. They presented the first authentication scheme using ECC for the WSNs. But (Han 2011) showed that Yeh et al.'s scheme lacked mutual authentication and forward security. In 2013, (Shi and Gong 2013) presented a new two-factor authentication scheme with ECC. But in 2014, (Choi et al. 2014) showed that the scheme in Shi and Gong (2013) had disadvantages such as the session key attack and the off-line password guessing attack. They also showed their new scheme in the paper. Unfortunately, (Wu et al. 2015a) showed that the scheme in Choi et al. (2014) still had some weaknesses, containing under the off-line password guessing attack and the user forgery attack. Moreover, the identity of the user is exposed in the message.

To avoid the common attacks, in 2014, (Turkanović et al. 2014) proposed a scheme to fit for the heterogeneous ad hoc WSNs. According to Chang and Le (2015) and Farash et al. (2015), Turkanović et al. scheme is under the off-line password guessing attack, the sensor node impersonation attack and the stolen verifier attack. Also, the identity of every user in the scheme can be tracked. Here we should mention that the schemes in Chang and Le (2015) and Farash et al. (2015) are all vulnerable to off-line password guessing attack. Also, the first scheme in Chang and Le (2015) does not keep strong forward security.

In 2014, (Hsieh and Leu 2014) showed that Vaidya et al.'s scheme could not withstand the insider attack and the off-line password guessing attack. They also gave their scheme in the paper. But we find that Hsieh and Leu's scheme has disadvantages including under the off-line guessing attack and the sensor node capture attack and also destitution of session key. According to Xue et al. (2013), Turkanović et al. employed the fifth authentication model and they said that it was the only model where the user first contacted the special sensor and asserted that their model was based on the notion of IoT. In fact, according to Fantacci et al. (2014) and Nguyen et al. (2015), which are about the security of IoT, the way that the user contacting the gateway first is more often used. We can see many schemes of this kind (Das 2009; Khan and Alghathbar 2010; Vaidya et al. 2010; Chen and Shih 2010; Yeh et al. 2011; Yoo et al. 2012; Xue et al. 2013; Hsieh and Leu 2014). In fact, the above two structures can both be used in homogeneous WSN. To overcome those weaknesses, we give a new authentication scheme for common WSNs, where the user sends messages to the gateway at first. And we show its security with formal proof, formal verification and informal analysis of security characters. From the explanation, we can see that our scheme is suitable for the IoT security properties.

1.2 Our contribution

1. We point out that Hsieh and Leu's scheme is not secure because it has several disadvantages in security.

2. We present a new two-factor authentication scheme for WSNs also based on ECC, like Choi et al. (2014) and Yeh et al. (2011).
3. We prove our scheme secure with a formal proof in a standard model and a formal verification with Proverif. Also, the concrete security analysis denotes that our scheme meets the IoT security properties.

1.3 Structure of our paper

The remainder of the paper is arranged as follows: Sect. 2 lists the basic knowledge used in the paper. Hsieh and Leu's scheme and its disadvantages are in Sect. 3. Our scheme, its formal proof, formal verification and concrete analysis are in Sects. 4, 5, 6 and 7, respectively. We list the performance comparisons among our scheme and several recent schemes in Sect. 8. Finally, the conclusion is in Sect. 9.

2 Preliminaries

We illustrate some basic knowledge for the whole paper in this Section.

2.1 Symbols used in the paper

We list the symbols throughout the paper in Table 1.

Here the elliptic curve E is $y^2 = x^3 + ax + b \ mod \ p$ where $a, b \in F_p$ and $4a^3 + 27b^2 \neq 0 \ mod \ p$. We omit $mod \ p$ for convenience in the following part. Some calculation problems are demonstrated as follows:

- Elliptic Curve Discrete Logarithm (ECDL) problem: P and Q are two points in G. It is difficult to compute the special integer $\alpha \in Z_q^*$ satisfying $Q = \alpha P$.
- Elliptic Curve Computational Diffie-Hellman (ECCDH) problem: aP and bP are two points in G. To get abP is hard with only aP and bP.
- Elliptic Curve Decisional Diffie-Hellman (ECDDH) problem: aP, bP and cP are three points in G. It is hard to decide if $abP = cP$.

2.2 Hypotheses for our analysis

Assumption 1 First we list some hypotheses about cryptographical computation.

1. The cryptographic algorithms are secure. $E_k(.)/D_k(.)$ is strong. m is a binary string and no one can decrypt $E_k(m)$ in polynomial time unless he knows k. And the collisions of the hash functions cannot be found in polynomial time.
2. The lengths of the random numbers, hash results and the secret number x are l. The above strings can resist the guessing attacks.

Assumption 2 According to Fan et al. (2011), Wang and Wang (2014), Xu and Wu (2015b) and Wu et al. (2015a), we use the following adversary model:

1. The attacker A has the ability to control the public communication channel under the two-factor environment (Wu et al. 2015a), e.g., forging, blocking, modifying or eavesdropping the messages. Also, A can seize the sensors and get all information from them.

Table 1 Symbols throughout the paper

Symbols	Meaning
p, q	Large primes
$E(F_p)$	A finite field F_p on the elliptic curve E
G	A subgroup in F_p with order q
P	The generator of G
GW, x	The gateway and its secret key
U_i, ID_i, PW_i	The $i-th$ user and his identity and password
S_j, SID_j	The $j-th$ sensor and its identity
sk_u, sk_s	The session keys calculated by the user and the sensor
A	The malicious adversary
$h(.), h_1(.)$	The one-way hash functions
l	The security parameter
$E_k(.)/D_k(.)$	The symmetric encryption/decryption function with key k
$a \oplus b$, $a\|\|b$	The XOR computation and the conjunction with strings a and b
$a? = b$	Whether a equals b

2. A can get all information stored in the smart card, according to the technology proposed in Kocher et al. (1999). But this ability can be only once, since the leakage of data in the smart card means that the two-factor environment does not exist.
3. In papers He et al. (2015) and Wang and Wang (2014), researchers claim that the passwords are in a small dictionary and so do the identities of users. The adversary can choose a pair of (ID^*, PW^*) to guess them until he gets the right pair. We employ this view and consider that the guessing action can be done in polynomial time.
4. A can gain the old session keys.
5. Even though A gets all data in the gateway, sensors and the user's smart card with his identity and password, he still cannot calculate the past session keys. If this item is discussed, the last item cannot be done.

2.3 Security properties of IoT

According to Nguyen et al. (2015), there are five aspects for the IoT security properties. We simply explain them according to our scheme:

1. Confidentiality: A could not obtain useful information between the participants.
2. Integrity: the receiver can detect the change of the received messages.
3. Authentication: the receiver can check the origin of the received messages.
4. Authorization: the receiver can verify if the accessor is authorized.
5. Freshness: no old messages can be used to crack the session.

The concrete applications are illustrated in Sect. 7.

3 Cryptanalysis of Hsieh and Leu's scheme

3.1 Review of Hsieh and Leu's scheme

Hsieh and Leu's scheme consists of three phases: registration, login and authentication, and password change. The last phase has little relation to attacks, so we omit it here. At first, GW and all sensors share a secret number c. Every sensor S_j should has its own identity SID_j, but Hsieh and Leu did not mention this point.

3.1.1 Registration

1. U_i chooses ID_i and PW_i, computes $HPW_i = h(PW_i)$ and sends $\{ID_i, HPW_i\}$ to GW through a secure way.
2. GW generates a nonce w_i, gives a smart card a name ID_s, computes $B_1 = h(h(ID_i||HPW_i||c) \oplus x) \oplus h(w_i)$, $B_2 = h((HPW_i||w_i) \oplus c)$, $B_3 = c \oplus h(ID_i||ID_s||HPW_i)$ and $B_4 = HPW_i \oplus w_i$, and stores $(ID_s, B_1, B_2, B_3, B_4)$ into the smart card. Finally GW sends the card to U_i secretly.

3.1.2 Login and authentication

1. U_i inserts his smart card in the terminal and enters ID_i and PW_i. The smart card computes $HPW_i = h(PW_i)$, $w_i = B_4 \oplus HPW_i$ and $c = B_3 \oplus h(ID_i||ID_s||HPW_i)$ and checks $B_2? = h((HPW_i||w_i) \oplus c)$. If the equation does not hold, the phase will be stopped. Otherwise, the card picks the timestamp T_0, computes $C_0 = B_1 \oplus h(w_i)$, $DID_i = h(ID_i||HPW_i||c) \oplus h(c||T_0)$ and $C_1 = h(C_0||c||T_0)$ and sends $\{DID_i, C_1, T_0\}$ to GW.
2. GW verifies if $T_1 - T_0 \leq \Delta T$, where T_1 is the timestamp. If it is false, the session will be stopped. Otherwise, GW computes $C_2 = DID_i \oplus h(c||T_0)$, and checks $C_1? = h(h(C_2 \oplus x)||c||T_0)$. If it is wrong, the session will be rejected. Otherwise, GW computes $C_3 = h(DID_i||SID_j||c||T_1)$ and sends $\{DID_i, C_3, T_1\}$ to S_j.
3. S_j selects the timestamp T_2 and checks if $T_2 - T_1 \leq \Delta T$ and $C_3? = h(DID_i||SID_j||c||T_1)$. If either of them fails, the phase will be stopped. Then S_j calculates $C_4 = C_3 \oplus c$ and $C_5 = h(C_4||c||T_2)$ and sends $\{C_5, T_2\}$ to GW.
4. GW checks if $T_3 - T_2 \leq \Delta T$ where T_3 is the timestamp. Then it computes $C_4 = C_3 \oplus c$ and checks $C_5? = h(C_4||c||T_2)$. If they are right, GW sends the acceptance to S_j and U_i.

3.2 Weaknesses of Hsieh and Leu's scheme

3.2.1 Insider attack

U_i submits $HPW_i = h(PW_i)$ to GW. The malicious inside attacker guesses a password PW^* and computes $HPW_i? = h(PW^*)$. He can do the above steps until he finds the right password.

3.2.2 Off-line guessing attack

An attacker A as a legal user in the system retrieves all information in his smart card and computes $c = B_{3A} \oplus h(ID_A||ID_s||h(PW_A))$. Then A gets the messages from sessions initialized by U_i and then the data from U_i's smart card, guesses PW^* and computes $w^* = h(PW^*) \oplus B_4$ and checks $h(h(h(PW^*)||w^*) \oplus c)? = B_2$. He can repeat doing this until the correct PW_i is found. Then A guesses

ID^* and checks $B_3? = c \oplus h(ID^*||ID_s||PW_i)$ until he gets the right ID_i.

3.2.3 User forgery attack

Note that A has got the data in U_i's smart card before A gets the right ID_i and PW_i. So once he obtains the correct pair (ID_i, PW_i), he can forge U_i according to the login and authentication steps successfully. The process is described as follows:

A inputs (ID_i, PW_i) and the smart card computes $HPW_i = h(PW_i)$, $w_i = B_4 \oplus HPW_i$ and $c = B_3 \oplus h(ID_i||ID_s||HPW_i)$. Then the card picks the timestamp T_{0A}, computes $C_0 = B_1 \oplus h(w_i)$, $DID_i = h(ID_i||HPW_i||c) \oplus h(c||T_{0A})$ and $C_1 = h(C_0||c||T_{0A})$ and sends $\{DID_i, C_1, T_{0A}\}$ to GW. Since there is no session key formed, sending a legal message means A's success.

3.2.4 Sensor capture attack

Because c is all the same in every sensor, so if anyother sensor S_n is captured, A can fake $C_5 = h(C_4||c||T_2)$ with the current T_2 and c and the corresponding information $C_4 = C_3 \oplus c$.

3.2.5 No session key

There is no session key formed at the end of the authentication. U_i and S_j do not have a secure way to communicate after the authentication.

3.2.6 Lack of mutual authentication

We can see that Hsieh and Leu's scheme is devoid of the property of mutual authentication because the adversary A can forge a legal user through an attack.

4 Outline of our scheme

We present a new scheme to avoid the disadvantages. It includes four phases: initialization, registration, login and authentication, and password change. It is built with the elliptic curve cryptosystem mechanism, like Choi et al. (2014) and Yeh et al. (2011), and is fit for common WSN environment. The login and authentication phase is shown in Table 2.

4.1 Initialization

GW produces an addition group G with a large prime order q on $E(F_p)$. G's generator is P. ID_{GW} is GW's identity. Also, GW chooses a secret key x and two hash functions $h(.)$ and $h_1(.)$.

4.2 Registration

For U_i:

1. U_i selects a random nonce r_0, his own identity ID_i and password PW_i. Then he calculates $MP_i = h(r_0||PW_i)$ and $MI_i = h(r_0||ID_i)$, and sends $\{MP_i, MI_i, ID_i\}$ to GW via a secure channel.
2. GW calculates $e_i = h(ID_{GW}||x||MI_i) \oplus MP_i$ and $f_i = h(MI_i||x) \oplus MI_i$, injects (e_i, f_i, P, p, q) into the smart card, stores ID_i in the database for auditing, and issues the card to U_i via a secure channel.
3. U_i stores $d_i = h(ID_i||PW_i) \oplus r_0$ into the smart card.

For S_j:

1. S_j submits SID_j to GW via a secure channel.
2. GW computes $c_j = h(SID_j||x)$ and sends it to S_j via a secure channel. S_j stores SID_j and c_j.

Meanwhile, if a sensor is changed to be a new one or a new sensor joins the wireless sensor network, the new sensor needs to register on GW like the above steps.

4.3 Login and authentication

1. U_i inserts his smart card and inputs ID_i and PW_i. The smart card calculates $r_1 = d_i \oplus h(ID_i||PW_i)$, $MI_i = h(r_1||ID_i)$ and $MP_i = h(r_1||PW_i)$.
2. U_i selects random numbers $\alpha \in [1, q-1]$, r_2 and r_3, picks up the sensor S_j as the partner, computes $MI_i^{new} = h(r_2||ID_i)$, $B_1 = e_i \oplus MP_i \oplus r_3$, $B_2 = \alpha P$, $B_3 = f_i \oplus MI_i \oplus MI_i^{new} \oplus h(r_3||MI_i)$, $B_4 = h(r_3||MI_i^{new}||B_2) \oplus ID_i$ and $B_5 = h(ID_i||MI_i||MI_i^{new}||SID_j)$, and sends $M_1 = \{MI_i, SID_j, B_1, B_2, B_3, B_4, B_5\}$ to S_j.
3. GW computes $r_3 = B_1 \oplus h(ID_{GW}||x||MI_i)$, $MI_i^{new} = B_3 \oplus h(MI_i||x) \oplus h(r_3||MI_i)$ and $ID_i = B_4 \oplus h(r_3||MI_i^{new}||B_2)$, and checks if ID_i is in the database and $B_5? = h(ID_i||MI_i||MI_i^{new}||SID_j)$. Either failed check leads to the rejection of the session. GW computes $c_j = h(SID_j||x)$ and $D_1 = h(MI_i||SID_j||c_j||B_2)$. Then the message $M_2 = \{MI_i, SID_j, B_2, D_1\}$ is sent to S_j.
4. S_j checks SID_j and $D_1? = h(MI_i||SID_j||c_j||B_2)$. If either checking is incorrect, S_j will stop the session. Otherwise S_j chooses $\beta \in [1, q-1]$, computes $C_1 = \beta P$, $C_2 = \beta B_2$, $sk_s = h_1(B_2||C_1||C_2)$, $C_3 = h(MI_i||SID_j||sk_s)$ and $C_4 = h(c_j||MI_i||SID_j)$, and sends $M_3 = \{C_1, C_3, C_4\}$ to GW.
5. GW first checks $C_4? = h(c_j||MI_i||SID_j)$. If it is right, GW computes $D_2 = h(ID_{GW}||x||MI_i^{new}) \oplus h(MI_i^{new}||r_3)$, $D_3 = h(MI_i^{new}||x) \oplus h(MI_i||r_3)$ and $D_4 = h(ID_i||MI_i||$

Table 2 Login and Authentication

U_i	GW	S_j
input ID_i,PW_i		
compute $r_1 = d_i \oplus h(ID_i\|\|PW_i)$		
$MI_i = h(r_1\|\|ID_i)$ and $MP_i = h(r_1\|\|PW_i)$		
select random numbers $\alpha \in [1, q-1]$, r_2 and r_3		
compute:		
$MI_i^{new} = h(r_2\|\|ID_i)$		
$B_1 = e_i \oplus MP_i \oplus r_3$		
$B_2 = \alpha P$		
$B_3 = f_i \oplus MI_i \oplus MI_i^{new} \oplus h(r_3\|\|MI_i)$		
$B_4 = h(r_3\|\|MI_i^{new}\|\|B_2) \oplus ID_i$		
$B_5 = h(ID_i\|\|MI_i\|\|MI_i^{new}\|\|SID_j)$		
$\xrightarrow{M_1=\{MI_i,SID_j,B_1,B_2,B_3,B_4,B_5\}}$		
	compute:	
	$r_3 = B_1 \oplus h(ID_{GW}\|\|x\|\|MI_i)$	
	$MI_i^{new} = B_3 \oplus h(MI_i\|\|x) \oplus h(r_3\|\|MI_i)$	
	$ID_i = B_4 \oplus h(r_3\|\|MI_i^{new}\|\|B_2)$	
	check ID_i,B_5? $= h(ID_i\|\|MI_i\|\|MI_i^{new}\|\|SID_j)$	
	compute:	
	$c_j = h(SID_j\|\|x)$	
	$D_1 = h(MI_i\|\|SID_j\|\|c_j\|\|B_2)$	
	$\xrightarrow{M_2=\{MI_i,SID_j,B_2,D_1\}}$	
		check SID_j
		check D_1? $= h(MI_i\|\|SID_j\|\|c_j\|\|B_2)$
		choose $\beta \in [1, q-1]$
		compute:
		$C_1 = \beta P$
		$C_2 = \beta B_2$
		$sk_s = h_1(B_2\|\|C_1\|\|C_2)$
		$C_3 = h(MI_i\|\|SID_j\|\|sk_s)$
		$C_4 = h(c_j\|\|MI_i\|\|SID_j)$
		$\xleftarrow{M_3=\{C_1,C_3,C_4\}}$
	check C_4? $= h(c_j\|\|MI_i\|\|SID_j)$	
	compute:	
	$D_2 = h(ID_{GW}\|\|x\|\|MI_i^{new}) \oplus h(MI_i^{new}\|\|r_3)$	
	$D_3 = h(MI_i^{new}\|\|x) \oplus h(MI_i\|\|r_3)$	
	$D_4 = h(ID_i\|\|MI_i\|\|MI_i^{new}\|\|SID_j\|\|D_2\|\|D_3\|\|r_3)$	
	$\xleftarrow{M_4=\{C_1,C_3,D_2,D_3,D_4\}}$	
check D_4? $= h(ID_i\|\|MI_i\|\|MI_i^{new}\|\|SID_j\|\|D_2\|\|D_3\|\|r_3)$		
compute:		
$B_6 = \alpha C_1$		
$sk_u = h_1(B_2\|\|C_1\|\|B_6)$		
check C_4? $= h(MI_i\|\|SID_j\|\|sk_u)$		
compute:		
$d_i^{new} = r_2 \oplus h(ID_i\|\|PW_i)$		
$e_i^{new} = D_2 \oplus h(MI_i^{new}\|\|r_3) \oplus h(r_2\|\|PW_i)$		
$f_i^{new} = D_3 \oplus MI_i^{new} \oplus h(MI_i\|\|r_3)$		
replace (d_i, e_i, f_i) with $(d_i^{new}, e_i^{new}, f_i^{new})$		

$MI_i^{new}||SID_j||D_2||D_3||r_3)$, and sends $M_4 = \{C_1, C_3, D_2, D_3, D_4\}$ to U_i.

6. U_i checks $D_4? = h(ID_i||MI_i||MI_i^{new}||SID_j||D_2||D_3||r_3)$, computes $B_6 = \alpha C_1$ and $sk_u = h_1(B_2||C_1||B_6)$, and checks $C_4? = h(MI_i||SID_j||sk_u)$. If either of checks fails, U_i terminates the session. Then the smart card computes new data $d_i^{new} = r_2 \oplus h(ID_i||PW_i)$, $e_i^{new} = D_2 \oplus h(r_2||PW_i) \oplus h(MI_i^{new}||r_3)$ and $f_i^{new} = D_3 \oplus MI_i^{new} \oplus h(MI_i||r_3)$, and replaces (d_i, e_i, f_i) with $(d_i^{new}, e_i^{new}, f_i^{new})$, respectively.

4.4 Password change

1. This step is as same as step 1 of login and authentication phase.
2. U_i selects random numbers r_4 and r_5, computes $MI_i^{new} = h(r_4||ID_i)$, $B_7 = e_i \oplus MP_i \oplus r_5$, $B_8 = f_i \oplus MI_i \oplus MI_i^{new} \oplus h(r_5||MI_i)$, $B_9 = ID_i \oplus h(r_5||MI_i^{new})$ and $B_{10} = h(ID_i||MI_i||MI_i^{new}||r_5)$, and sends $M_5 = \{MI_i, B_7, B_8, B_9, B_{10}\}$ with a password change request to GW.
3. GW computes $r_5 = B_7 \oplus h(ID_{GW}||x||MI_i)$, $MI_i^{new} = B_8 \oplus h(MI_i||x) \oplus h(r_5||MI_i)$ and $ID_i = B_9 \oplus h(r_5||MI_i^{new})$, and checks the validity of ID_i and $B_{10}? = h(ID_i||MI_i||MI_i^{new}||r_5)$. If either of them fails, the request will be abandoned. Otherwise, GW calculates $D_5 = h(ID_{GW}||x||MI_i^{new}) \oplus h(MI_i^{new}||r_5)$, $D_6 = h(MI_i^{new}||x) \oplus h(MI_i||r_5)$ and $D_7 = h(ID_i||r_5||MI_i||MI_i^{new}||D_5||D_6)$, and sends $M_6 = \{D_5, D_6, D_7\}$ to U_i with a grant.
4. U_i checks $D_7? = h(ID_i||r_5||MI_i||MI_i^{new}||D_5||D_6)$ and terminates the session if it is wrong. Otherwise, U_i is asked to input a new password PW_i^{new}. The card computes $MP_i^{new} = h(r_4||PW_i^{new})$, $e_i^{new2} = D_5 \oplus h(MI_i^{new}||r_5) \oplus MP_i^{new}$, $f_i^{new2} = D_6 \oplus h(MI_i||r_5) \oplus MI_i^{new}$ and $d_i^{new2} = h(ID_i||PW_i^{new}) \oplus r_4$, and updates (d_i, e_i, f_i) with $(d_i^{new2}, e_i^{new2}, f_i^{new2})$, respectively.

5 Formal proof

5.1 Basis of formal proof

Our scheme belongs to authenticated key exchange (AKE) scheme, and we give our formal proof based on (Bresson et al. 2003). Some changes are made to fit for our scheme.

In order to analyze our proof simply, we suppose there are three entities in the protocol $\mathcal{P}$: one user U, one sensor S and the gateway GW. If they need not be differentiated, we use I to express the entity.

Each entity owns many instances. We make U^i as the $i-th$ instance of U. And GW^t, S^j and I^k can be defined similarly. We consider an instance as an oracle and employ a simulator to answer the input messages. Three states may occur on an instance: accept, reject and $\perp$. If an oracle gets a normal message, it will turn to be the accept state. If an oracle receives an incorrect message, the state reject will happen. Otherwise if no answer is generated, $\perp$ will appear. Once the oracle U^i or S^j is accepted and calculates a session key, it has the following elements: an identity for session (sid_{U^i} or sid_{S^j}), an identity for its partner (pid_{U^i} or pid_{S^j}) and the session key sk_{U^i} or sk_{S^j}. The situation is called *Partnering* which we will explain later.

Initializations should be done before the simulations. U has his identity ID, password PW and an issued smart card including d, e, f, P, q and p. The password PW is in a dictionary with size N. S owns c, P, p, q and its identity SID. GW contains its identity ID_{GW}, x, P, q and p. Furthermore, ID, SID, ID_{GW}, P, q and p are known to A.

Three definitions and an assumption used in the proof are illustrated below and readers can find the relative queries in Tables 3 and 4.

- *Partnering*: we consider U^i and S^j are partners once the session key is formed between them. At the same time, four conditions should be satisfied: U^i and S^j are accepted; $sid_{U^i} = sid_{S^j}$, $pid_{S^j} = U^i, pid_{U^i} = S^j$ and $sk_{U^i} = sk_{S^j}$.
- $sfs-fresh$(fresh with strong forward security): I^k reaches $sfs-fresh$ if the following queries do not happen:
  1. A $Reveal(I^k)$ occurs;
  2. A $Reveal(pid_{I^k})$ occurs;
  3. Any $Corrupt(I^m)$ query occurs before the *Test* query. Here m can be any legal numbers, including k.
- $sfs-ake$ *security*: if A has the advantage on guessing the coin a on $\mathcal{P}$ after $Test(I^k)$ where I^k is $sfs-fresh$ and A gives a bit a', we define the probability as

$$Adv_{\mathcal{P}}^{sfs-ake}(A) = 2Pr[a = a'] - 1$$

  And the scheme is $sfs-ake$ *secure* based on security length l if $Adv_{\mathcal{P}}^{sfs-ake}(A)$ is negligibly greater than $\frac{O(q_s)}{N}$ and q_s denotes the number of *Send* queries.
- Elliptic Curve Gap Diffie-Hellman(ECGDH) problem assumption: there are two points aP and bP in G. We define $Adv_A^{ECGDH}(t)$ as the probability for A to calculate abP based on the ECDDH oracle with polynomial time t. And in fact, $Adv_A^{ECGDH}(t)$ is ignorable.

Table 3 Simulation of *Send* queries

For a $Send(U^i, init)$ query, the simulator does the following steps:
selects random numbers r_2, r_3 and $\alpha \in [1, q-1]$,
computes $r_1 = d \oplus h(ID\|\|PW)$, $MI = h(r_1\|\|ID)$, $MP = h(r_1\|\|PW)$, $MI^{new} = h(r_2\|\|ID)$, $B_1 = e \oplus MP \oplus r_3$, $B_2 = \alpha P$,
$B_3 = f \oplus MI \oplus MI^{new} \oplus h(r_3\|\|MI)$, $B_4 = h(r_3\|\|MI^{new}\|\|B_2) \oplus ID$ and $B_5 = h(ID\|\|MI\|\|MI^{new}\|\|SID)$.
Returns $M_1 = \{MI, SID, B_1, B_2, B_3, B_4, B_5\}$ as the answer.
For a $Send(U^i, GW^t, M_1)$ query, the simulator does the following steps:
computes $r_3 = B_1 \oplus h(ID_{GW}\|\|x\|\|MI)$, $MI^{new} = B_3 \oplus h(MI\|\|x) \oplus h(r_3\|\|MI)$ and $ID = B_4 \oplus h(r_3\|\|MI^{new}\|\|B_2)$,
and checks $B_5? = h(ID\|\|MI\|\|MI^{new}\|\|SID)$.
If it fails, rejects the query.
Computes $c = h(SID\|\|x)$ and $D_1 = h(MI\|\|SID\|\|c\|\|B_2)$.
Returns $M_2 = \{MI, SID, B_2, D_1\}$ as the answer.
For a $Send(GW^t, S^j, M_2)$ query, the simulator does the following steps:
checks SID, and $D_1? = h(MI\|\|SID\|\|c\|\|B_2)$. If either of them fails, rejects the query.
Selects a random number $\beta \in [1, q-1]$, and computes $C_1 = \beta P$, $C_2 = \beta B_2$, $sk_s = h_1(B_2\|\|C_1\|\|C_2)$, $C_3 = h(MI\|\|SID\|\|sk_s)$ and $C_4 = h(c\|\|MI\|\|SID)$.
Then answers the query with $M_3 = \{C_1, C_3, C_4\}$.
For a $Send(S^j, GW^t, M_3)$ query, the simulator does the following steps:
checks $C_4? = h(c\|\|MI\|\|SID)$. If not, rejects the query.
Computes $D_2 = h(ID_{GW}\|\|x\|\|MI^{new}) \oplus h(MI^{new}\|\|r_3)$, $D_3 = h(MI^{new}\|\|x) \oplus h(MI\|\|r_3)$, and $D_4 = h(ID\|\|MI\|\|MI^{new}\|\|SID\|\|D_2\|\|D_3\|\|r_3)$.
Then answers this query with $M_4 = \{C_1, C_3, D_2, D_3, D_4\}$.
For a $Send(GW^t, U^i, M_4)$ query, the simulator does the following steps:
checks $D_4? = h(ID\|\|MI\|\|MI^{new}\|\|SID\|\|D_2\|\|D_3\|\|r_3)$.
Computes $B_6 = \alpha C_1$ and $sk_u = h_1(B_2\|\|C_1\|\|B_6)$ and checks $C_3? = h(MI\|\|SID\|\|sk_u)$.
If either of them is wrong, stops the query.
Computes $d^{new} = h(ID\|\|PW) \oplus r_2$, $e^{new} = D_2 \oplus h(r_2\|\|PW) \oplus h(MI^{new}\|\|r_3)$ and $f^{new} = D_3 \oplus MI^{new} \oplus h(MI\|\|r_3)$.
Finally replaces d, e and f with d^{new}, e^{new} and f^{new}, respectively.

Table 4 Simulation of other queries

For a hash query $h(s)$, if a record (s, ω) has already in L_h, ω is returned as the answer.
Otherwise, the simulator chooses $\omega \in \{0,1\}^l$, outputs the answer ω and writes (s, ω) in L_h.
For $h_1(s)$, the steps are similar. The record is $(1, s, \omega)$
For an $Execute(U^i, GW^t, S^j)$ query, the *Send* queries are done successively and the messages (M_1, M_2, M_3, M_4) are returned.
For a $Reveal(I^k)$ query, if I^k has been *Partnering*, returns sk_u or sk_s. Here I means U or S.
Otherwise a $\perp$ is the answer.
For a *Corrupt*(*smart card*) query, all information in U's smart card is retrieved.
For a $Corrupt(I^k)$ query, all information of I^k is returned. Since A knows some information at first,
the concrete results are below:
(1) $I = U$: All information in U's smart card with PW is obtained by A.
(2) $I = GW$: A can get the secret key x.
(3) $I = S$: A can obtain c.
For a $Test(I^k)$ query, if I^k is not $sfs-fresh$, a $\perp$ is got. Here I is U or S, like $Corrupt(I^k)$.
Otherwise, the simulator tosses a coin a.
If $a = 1$, returns the session key. If $a = 0$, returns a random string with the length l.

5.2 Security proof

Theorem 1 In the scheme $\mathcal{P}$, the cyclic addition group G in the finite field $E(F_p)$ has a large prime order q. The passwords are in a set with N elements. A can make at most *Send*, *Execute* and hash queries for q_s, q_e and q_h times, respectively. l is the security length. We make T_m as the time cost of a scalar multiplication in G and know

$$Adv_{\mathcal{P}}^{sfs-ake}(A) \leq \frac{(q_s+q_e)^2}{q-1} + \frac{q_h^2+(q_s+q_e)^2}{2^l} + \frac{12q_h+7q_s}{2^{l-1}} + \frac{2q_s}{N} + 4q_h((q_s+q_e)^2+1)Adv_A^{ECGDH}(t+(4q_e+2q_s)T_s)$$

Proof The proof contains a sequence of games, from G_0 to G_8. The event A guesses the coin a correctly in the test session in Game G_i is denoted as $Succ_i$. A need not take time in guessing the user's identity since there is only one user in the proof.

- Game G_0: this game is for the real attacks with random oracles. If any of the following things happens, a random bit a' is selected instead of the answer of *Test*.
  1. A does not guess a' when the game aborts or stops.
  2. A uses more queries than the pre-determined quantities.
  3. A uses more time than the pre-determined time.

  According to the definition,

  $$Adv_{\mathcal{P}}^{sfs-ake}(A) = 2Pr[Succ_0] - 1.$$

- Game G_1: all the oracles are simulated in this game. Three lists are defined for some queries. L_h stores the answers to hash queries. If the hash query is asked by A, the answer is stored in L_A. And $L_{\mathcal{P}}$ stores the transcripts of all messages. The queries are demonstrated in Tables 3 and 4. A can do queries using the oracles to break the privacy of authentication processes and the session keys. So G_1 and G_0 are indistinguishable and $Pr[Succ_1] = Pr[Succ_0]$.
- Game G_2: we try to eliminate the collisions in the messages. In light of birthday paradox, we show the three collisions as follows:
  1. The random numbers $\alpha, \beta \in [1, q-1]$ in different sessions may be the same and the whole probability is bounded by $\frac{(q_s+q_e)^2}{2(q-1)}$.
  2. The random numbers r_1, r_2 and r_3 may have collisions and the whole probability is $\frac{(q_s+q_e)^2}{2^{l+1}}$.
  3. The probability of collisions in hash results is upper bounded by $\frac{q_h^2}{2^{l+1}}$.

  So we can see that $|Pr[Succ_2] - Pr[Succ_1]| \leq \frac{(q_s+q_e)^2}{2(q-1)} + \frac{(q_s+q_e)^2+q_h^2}{2^{l+1}}$.
- Game G_3: we think about the probability of forging M_1 without random oracles in this game. Since the simulator gives response as S, we add steps in $Send(U^i, GW^t, M_1)$: the simulator needs to check if $M_1 \in L_{\mathcal{P}}$ and $(ID||*,*)$, $(*||ID, MI)$, $(*||MI,*)$, $(*||ID,*)$, $(*||B_2,*)$ and $(ID||MI|| * ||SID, B_5)$ are in L_A. If any of them fails, the query will be terminated. Because S does not know PW or MI^{new}, $(r_1||PW,*)$ cannot be examined. The probabilities for $(*||ID, MI)$ and $(ID||MI|| * ||SID, B_5)$ are all bounded by $\frac{q_s}{2^l}$ and for the others are bounded by $\frac{q_h}{2^l}$. If the examinations are considered, we can see $|Pr[Succ_3] - Pr[Succ_2]| \leq \frac{5q_h+2q_s}{2^l}$.
- Game G_4: we think about the probability of forging M_2 without random oracles and add steps on $Send(GW^t, S^j, M_2)$: checks if $M_2 \in L_{\mathcal{P}}$ and $(SID||*, c)$ and $(MI||SID||c||B_2, D_1)$ are in L_A. The probability for $(MI||SID||c||B_2, D_1)$ is bounded by $\frac{q_s}{2^l}$ while for $(SID||*, c)$ is $\frac{q_h}{2^l}$. So $|Pr[Succ_4] - Pr[Succ_3]| \leq \frac{q_h+q_s}{2^l}$.
- Game G_5: we think about the probability of forging M_3 without random oracles and add steps on $Send(S^j, GW^t, M_3)$: checks if $M_3 \in L_{\mathcal{P}}$ and $(1, B_2||C_1||*,*)$, $(MI||SID||*, C_3)$ and $(c||MI||SID, C_4)$ are in L_A. The probabilities for $(MI||SID||*, C_3)$ and $(c||MI||SID, C_4)$ are both bounded by $\frac{q_s}{2^l}$. And it is bounded by $\frac{q_h}{2^l}$ at most for $(1, B_2||C_1||*,*)$. So $|Pr[Succ_5] - Pr[Succ_4]| \leq \frac{q_h+2q_s}{2^l}$.
- Game G_6: here is the game for forging M_4 without random oracles and we add steps on $Send(GW^t, U^i, M_4)$: checks if $M_4 \in L_{\mathcal{P}}$ and $(ID_{GW}|| * ||MI^{new},*)$, $(MI^{new}|| r_3,*)$, $(MI^{new}||*,*)$, $(MI||r_3,*)$, $(1, B_2 ||C_1||*,*)$, $(MI||SID||*, C_3)$ and $(ID||MI||MI^{new}||SID||D_2||D_3|| r_3, D_4)$ are in L_A. The last two data have the upper-bound probability $\frac{q_s}{2^l}$, respectively and each of the others has at most $\frac{q_h}{2^l}$. So $|Pr[Succ_6] - Pr[Succ_5]| \leq \frac{5q_h+2q_s}{2^l}$.
- Game G_7: the ECGDH problem is brought in and A can use the random oracles. Once A gets the real session key via the hash oracle h_1, we will say we use A to work out ECGDH problem. We change the h_1 oracle as follows: if A asks $(1, \alpha P||\beta P||\lambda)$, the simulator checks if $(1, \alpha P||\beta P||*, sk) \in L_A$. If it exists, returns sk. Otherwise, ECDDH oracle is used to check $\lambda? = \alpha\beta P$. The query should be terminated if the check fails. Otherwise, the simulator selects $sk \in \{0,1\}^l$, answers with it, and adds $(1, \alpha P||\beta P||\lambda, sk)$ into L_A. Here we divide the game into two aspects. At the beginning A queries *Corrupt*(*smart card*) and obtains all information from the card.

1. It simulates active attacks. A chooses a password PW^* from the dictionary with size N. Then he can forge messages to start the session. As A can send at most q_s *Send* queries, the probability of guessing the right password is $\frac{q_s}{N}$.
2. It simulates passive attacks. There are two cases:
   (a) A asks *Execute* queries and finally queries h_1 successfully to break the ECGDH problem. $(1, \alpha P||\beta P||\alpha\beta P, sk)$ can be retrieved from L_A with the probability bounded by $\frac{1}{q_h}$. So the probability of this case is at most $q_h Adv_A^{ECGDH}(t + 4q_e T_m)$.
   (b) A asks *Send* queries in order to simulate the *Execute* query and repeats querying. Similar as the last case, we can get the probability $q_h Adv_A^{ECGDH}(t + 2q_s T_m)$ for this case.

So we know

$$
\begin{aligned}
&|Pr[Succ_7] - Pr[Succ_6]| \\
&\leq \frac{q_s}{N} + q_h Adv_A^{ECGDH}(t + 4q_e T_m) + q_h Adv_A^{ECGDH}(t + 2q_s T_m) \\
&\leq \frac{q_s}{N} + 2q_h Adv_A^{ECGDH}(t + (4q_e + 2q_s)T_m)
\end{aligned}
$$

– Game G_8: it is for strong forward security. A can ask all *Corrupt* oracles. However, in the light of the $sfs - fresh$ notion, $Corrupt(I^m)$ query should occur after *Test*. Therefore, A can use the old sessions only. Like G_7, we can find $(1, \alpha P||\beta P||\alpha\beta P, sk)$ from L_A. And the probability of obtaining αP and βP in the same session is $\frac{1}{(q_s+q_e)^2}$. So $|Pr[Succ_8] - Pr[Succ_7]| \leq 2q_h(q_s + q_e)^2 Adv_A^{ECGDH}(t + (4q_e + 2q_s)T_m)$. Now A has no advantage and $Pr[Succ_8] = \frac{1}{2}$.

Combining all above games, Theorem 1 is proved. □

6 Formal verification

Researchers have developed many tools to verify the cryptographic schemes. Proverif is one of them for the verification. It can prove characters such as observational equivalence, reachability properties and correspondence assertions, which are very useful to computer network security. Attacks can be reconstructed by the tool. An execution trace is described when one security character is not satisfied. We also make a formal verification of the Login and authentication phase for our scheme with Proverif and the codes are illustrated.

6.1 Premises in the verification

First some premises including channels, shared keys, constants, functions and equations are listed for the process of protocol. These are illustrated in Table 5.

The four events are used to test the correspondence relations for the user and the sensor in Login and authentication phase. Moreover, the first two queries are for checking the session keys' security and the last two are for checking if the relations of the events are right. They are shown in Table 6.

6.2 Processes

Every participant uses its own process and the scheme is modeled as the parallel execution:

Table 5 Definitions of Proverif code

```
(*Channels and shared keys are listed below*)
free ch1: channel. (*the public channel between the user and the sensor*)
free ch2: channel. (*the public channel between the sensor and GW*)
free sch1: channel [private]. (*the secret channel between the user and GW*)
free sch2: channel [private]. (*the secret channel between the sensor
and GW*)
free sku: bitstring [private]. (*the user's session key*)
free sks: bitstring [private]. (*the sensor's session key*)
(*Constants are listed below*)
free x:bitstring [private]. (*the private key of GW*)
free IDi:bitstring [private]. (*Ui's identity*)
free PWi:bitstring [private]. (*Ui's password*)
const IDGW:bitstring. (*GW's identity*)
const P:bitstring. (*the generator P*)
const SIDj:bitstring. (*Sj's identity*)
table d(bitstring). (*database in GW*)
(*Functions and equations are listed below:*)
fun h(bitstring):bitstring. (*hash function*)
fun h1(bitstring):bitstring. (*hash function*)
fun mul(bitstring,bitstring):bitstring. (*scalar multiplication function*)
fun xor(bitstring,bitstring):bitstring. (*X-OR function*)
fun con(bitstring,bitstring):bitstring. (*string concatenation*)
equation forall m:bitstring,n:bitstring; xor(xor(m,n),n)=m.
(*X-OR computation*)
equation forall m:bitstring,n:bitstring; mul(mul(P,m),n)
= mul(mul(P,n),m).(*scalar multiplication*)
```

Table 6 Events and queries in Proverif code

```
Events
event UserStart(bitstring)
event UserAuth(bitstring)
event SensorStart(bitstring)
event SensorAuth(bitstring)
Queries
query attacker(sku)
query attacker(sks)
query id:bitstring; inj-event(UserAuth(id))
== > inj-event(UserStart(id)).
query sid:bitstring; inj-event(SensorAuth(sid))
== > inj-event(SensorStart(sid)).
```

process $!User|!GW|!Sensor$

The processes of the user, the sensor and the gateway are in Tables 7, 8 and 9, respectively. The processes of the user and the sensor can be divided into two parts: registration and authentication. The process of the gateway contains three parts: two for registration and one for authentication.

6.3 Results of the verification

We demonstrate the main results as follows. It is easy to see that the session keys are secure via the verification:

Starting query inj-event(SensorAuth(sid))
== > inj-event(SensorStart(sid))
RESULT inj-event(SensorAuth(sid))
== > inj-event(SensorStart(sid)) is true.
Starting query inj-event(UserAuth(id))
== > inj-event(UserStart(id))
RESULT inj-event(UserAuth(id))
== > inj-event(UserStart(id)) is true.
Starting query not attacker(sks[]) RESULT not attacker(sks[]) is true.
Starting query not attacker(sku[]) RESULT not attacker(sku[]) is true.

7 Analysis of security properties

We discuss the security properties in this section and show the comparison with some recent schemes (Choi et al. 2014; Shi and Gong 2013; Turkanović et al. 2014; Hsieh and Leu 2014; Chang and Le 2015; Farash et al. 2015) in Table 10. In Chang and Le (2015), there are two versions: one is based on hash functions and the other is based on ECC. We use P1 and P2 to tell them apart. The results are in Table 10. Each security character may belong to one IoT security character demonstrated in Sect. 2.3, and we also demonstrate this for one column in Table 10.

Table 7 Code for the user

```
The user's process:
let User=
new r0:bitstring;
let MPi=h(con(r0,PWi)) in
let MIi=h(con(r0,IDi)) in
out(sch1,(MPi,MIi,IDi));
in(sch1,(xei:bitstring,xfi:bitstring));
let ei = xei in
let fi = xfi in
let di = xor(h(con(IDi,PWi)),r0) in
!
(
event UserStart(IDi);
let r1 = xor(di,h(con(IDi,PWi))) in
let MIi' = h(con(r1,IDi)) in
let MPi' = h(con(r1,PWi)) in
new alpha:bitstring;
new r2:bitstring;
new r3:bitstring;
let MInew = h(con(r2,IDi)) in
let B1= xor(xor(ei,MPi'),r3) in
let B2 = mul(P,alpha) in
let B3 = xor(xor(xor(fi,MIi'),MInew),h(con(r3,MIi'))) in
let B4 = xor(IDi,h(con(con(r3,MInew),B2))) in
let B5 = h(con(con(con(IDi,MIi'),MInew),SIDj)) in
let M1 =(MIi',SIDj,B1,B2,B3,B4,B5) in
out(ch1,M1);
in (ch1,(xC1:bitstring,xC3:bitstring,xD2:bitstring,xD3:bitstring,
xD4:bitstring));
if xD4 = h(con(con(con(con(con(con(IDi,MIi'),MInew),SIDj),
xD2),xD3),r3)) then
let B6 = mul(xC1,alpha) in
let sku = h1(con(con(B2,xC1),B6)) in
if xC3 = h(con(con(MIi',SIDj),sku)) then
let dinew = xor(r2,h(con(IDi,PWi))) in
let einew = xor(xor(xD2,h(con(MInew,r3))), h(con(r2,PWi))) in
let finew = xor(xor(xD3,MInew),h(con(MIi',r3))) in
let di = dinew in
let ei = einew in
let fi = finew in
0).
```

7.1 Resistant to the insider attack

In registration phase, MP_i submitted by the user is a hash result with a random number r_0 and the password PW_i as input. A malicious adversary A cannot guess the real PW_i from MP_i due to lack of r_0. It meets the confidentiality property for IoT security.

Table 8 Code for the sensor

```
The sensor's process:
let Sensor =
out(sch2,SIDj);
in(sch2, xxcj:bitstring);
!
(
in(ch2,(uMIi:bitstring,uSIDj:bitstring,uB2:bitstring,uD1:bitstring));
if uSIDj = SIDj then
if uD1 = h(con(con(con(uMIi,uSIDj),xxcj),uB2)) then
event SensorStart(uSIDj);
new beta:bitstring;
let C1 = mul(P,beta) in
let C2 = mul(uB2,beta) in
let sks = h1(con(con(uB2,C1),C2)) in
let C3 = h(con(con(uMIi,SIDj),sks)) in
let C4 = h(con(con(con(xxcj,uMIi),SIDj),Yj)) in
let M3 = (C1,C3,C4) in
out(ch2,M3);
0
).
```

Table 9 Code for the gateway

```
User registration
let GWReg1 =
in(sch1,(xMPi:bitstring,xMIi:bitstring,xIDi:bitstring));
let ei'= xor(con(con(IDGW,x),xMIi),xMPi) in
let fi'= xor(h(con(xMIi,x)),xMIi) in
insert d(xIDi);
out (sch1,(ei',fi')).
Sensor registration
let GWReg2 =
in(sch2,(ySIDj:bitstring));
let cj = h(con(ySIDj,x)) in
out(sch2,(cj)).
Authentication
let GWAuth =
in(ch1,(xxMIi:bitstring,xxSIDj:bitstring,xxB1:bitstring,
xxB2:bitstring, xxB3:bitstring,xxB4:bitstring,xxB5:bitstring));
let xr3 = xor(xxB1,con(con(IDGW,x),xxMIi)) in
let xMIinew = xor(xor(xxB3,h(con(xxMIi,x))),
  h(con(xr3,xxMIi))) in
let xIDi = xor(xxB4,h(con(con(xr3,xMIinew),xxB2))) in
get d(=xIDi) in
if xxB5 = h(con(con(con(xIDi,xxMIi),xMIinew),xxSIDj)) then
event UserAuth(xIDi);
let pcj = h(con(xxSIDj,x)) in
let xxD1 = h(con(con(con(xxMIi,xxSIDj),pcj),xxB2)) in
let M2 =(xxMIi,xxSIDj,xxB2,xxD1) in
out (ch2,M2);
in (ch2,(xxC1:bitstring,xxC3:bitstring,xxC4:bitstring));
if xxC4 = h(con(con(pcj,xxMIi),xxSIDj)) then
event SensorAuth(xxSIDj);
let D2 = xor(h(con(con(IDGW,x),xMIinew)),
  h(con(xMIinew,xr3))) in
let D3 = xor(h(con(xMIinew,x)), h(con(xxMIi,xr3))) in
let D4 =
  con(con(con(con(con(con(xIDi,xxMIi),xMIinew),xxSIDj),
D2),D3),xr3) in
let M4 = (xxC1,xxC3,D3,D4,D5) in
out(ch1,M4).
```

7.2 Resistant to the off-line guessing attack

According to Sect. 2.2, A could obtain d_i, e_i and f_i from U_i's smart card and the messages M_1^{old}, M_2^{old}, M_3^{old} and M_4^{old} in the last session from the channel. Then he guesses the pair (ID^*, PW^*) and computes $r^* = d_i \oplus h(ID^*||PW^*)$, $MI^* = h(r^*||ID^*)$ and $MP^* = h(r^*||PW^*)$. Here A can use three equations: $B_4^{old} = h(r_3^{old}||MI^*||B_2^{old}) \oplus ID^*$, $e_i \oplus h(MI^*||r_3^{old}) = D_2^{old} \oplus MP^*$ and $f_i \oplus MI^* = D_3^{old} \oplus h(MI_i^{old}||r_3^{old})$. The random number r_3^{old} in the last session is needed in all equations. But $r_3^{old} = h(ID_{GW}||x||MI_i^{old}) \oplus B_1$ and x is kept in GW secretly. So A has no chance to calculate $h(ID_{GW}||x||MI_i^{old})$. Due to this reason, A cannot finish the guessing. For IoT security, it meets the confidentiality property.

7.3 Resistant to the user forgery attack

If A wants to forge M_1 to start a session, he should generate $B_1 = h(ID_{GW}||x||MI_i) \oplus r_3$ and $B_3 = h(MI_i||x) \oplus MI_i^{new} \oplus h(r_3||MI_i)$. Since x is kept secretly in the gateway, A has no ability to forge suitable B_1 and B_3. For IoT security, it meets the integrity property.

7.4 Resistant to the gateway forgery attack

GW can get the random number r_3 which is produced by U_i with the secret key x and calculate D_1, D_2, D_3 and D_4. So it is hard for A to generate M_2 and M_3 in order to forge GW. For IoT security, it meets the integrity property.

7.5 Resistant to the sensor capture attack

Every sensor owns SID_j and the corresponding secret number c_j. Stealing the two strings from one sensor does not affect the others. So our scheme can resist to the sensor capture attack. For IoT security, it meets the integrity property.

Table 10 Comparison of security characters

	IoT security properties	Ours	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Resistant to the insider attack	(1)	✓	×	✓	✓	✓	✓	✓	✓
Resistant to the off-line guessing attack	(1)	✓	×	×	×	×	×	×	×
Resistant to the user forgery attack	(2)	✓	×	×	×	×	✓	✓	✓
Resistant to the gateway forgery attack	(2)	✓	✓	✓	✓	✓	✓	✓	✓
Resistant to the sensor capture attack	(2)	✓	×	×	✓	✓	✓	✓	✓
Resistant to the de-synchronization attack	(2)	✓	✓	✓	✓	✓	✓	✓	✓
Resistant to the replay attack	(5)	✓	✓	✓	✓	✓	✓	✓	✓
Resistant to the known-key attack	(1)	✓	?	✓	✓	✓	✓	✓	✓
Useful input identity	(4)	✓	✓	×	✓	✓	✓	✓	✓
User anonymity	(1)	✓	✓	×	×	×	✓	✓	✓
Mutual authentication	(3)	✓	×	×	×	×	✓	✓	✓
Form a session key	(1)	✓	×	✓	✓	✓	✓	✓	✓
Strong forward security	(1)	✓	?	×	✓	✓	×	✓	✓

[1]:Hsieh and Leu (2014);[2]:Turkanović et al. (2014);[3]:Shi and Gong (2013);[4]:Choi et al. (2014);
[5]:P1 in Chang and Le (2015);[6]:P2 in Chang and Le (2015);[7]: Farash et al. (2015)
(1):Confidentiality;(2):integrity;(3):authentication;(4):authorization;(5):freshness

7.6 Resistant to the de-synchronization attack

The de-synchronization attack means that the gateway denies the legal user's login and authentication. In our scheme, password must be checked in a session with the gateway before changing. That blocks inputting wrong password by mistake. Also, inconsistent data between the gateway and the user may cause this attack. The gateway does not store any information about the concrete users except the identity for audit. Data change always happens on the user side only. It is impossible that inconsistent data appear between the user and the gateway. Thus, our scheme avoids the de-synchronization attack. For IoT security, it meets the integrity property.

7.7 Resistant to the replay attack

Even if A replays U_i and GW's old messages M_1 and M_2 to repeat the login and authentication phase, S_j will choose a fresh random number β and produce the new C_1 and C_3. So A cannot calculate the session key to crack the session. For IoT security, it meets the freshness property.

7.8 Resistant to the known-key attack

The session key in our scheme is built by the ECCDH problem. There is no relation among any session keys. So even if A has got some session keys by accident, other keys will not be affected. For IoT security, it meets the confidentiality property.

7.9 Useful input identity

Unlike paper (Turkanović et al. 2014), ID_i should be input at the beginning of login and it is useful in the process of login and authentication in our scheme. For IoT security, it meets the authorization property.

7.10 User anonymity

We use a dynamic hash result MI_i as the identity of U_i and it is updated in every authentication and password change phase. The attacker cannot track the user by MI_i, not to say getting the real identity of U_i. It meets the confidentiality property for IoT security.

7.11 Mutual authentication

We demonstrate the explanations of mutual authentication between U_i, GW and S_j:

- GW authenticates U_i by checking B_5, and S_j by checking C_4.
- S_j checks D_1 to verify GW and authenticates U_i indirectly by GW.
- U_i checks D_4 to verify GW and C_4 to verify S_j.

It meets the authentication property for IoT security.

7.12 Form a session key

Unlike paper (Hsieh and Leu 2014), the user and the sensor share a session key at last in our scheme. The latter

conversation can be encrypted by the temporary key. It meets the confidentiality property for IoT security.

7.13 Strong forward security

If A gets all information from U_i, S_j and GW, he cannot calculate the historical session keys, as our scheme employs ECCDH problem. It meets the confidentiality property for IoT security.

8 Performance comparison

In this section we show the performance of proposed scheme and compare it with schemes in Choi et al. (2014), Shi and Gong (2013), Turkanović et al. (2014), Hsieh and Leu (2014), Chang and Le (2015) and Farash et al. (2015).

We illustrate the basis of comparison:

- T_s is the time cost of one scalar multiplication in G and T_h is the time for hash function. According to Xu and Wu (2015a), $T_s = 7.3529ms$, $T_h = 0.0004ms$. We can see that $T_s \gg T_h$.
- We assume that the points in G has 320 bits in total. And we also assume the security parameter l is 160. That is to say, the lengths for the secret numbers, e.g., x in the gateway, the hash results, random numbers, timestamps and SID_j are 160 bits. Q_u and Q_s are the quantities of the users and the sensors in the WSN. $|P|$, $|p|$ and $|q|$ are lengths for the parameters P, p and q. According to Hankerson et al. (2004), $|p| = 160$, $|q| \approx 160$.

We explain every aspect in Table 11 which lists the comparison results.

- Our scheme is in the middle by comparing time cost of user, better than (Choi et al. 2014; Shi and Gong 2013).
- By comparing time cost on the sensor, our scheme is the same as (Shi and Gong 2013; Choi et al. 2014), and better than P2 in Chang and Le (2015), which also employs ECC. The main reason is that our scheme employs ECCDH problem to obtain strong forward security while schemes based on hash functions do not have this property, like Hsieh and Leu (2014) and Turkanović et al. (2014).
- Our scheme is also in the middle by comparing time cost of the gateway. Also, it is better than (Choi et al. 2014; Shi and Gong 2013).
- Our scheme has less communication cost than (Choi et al. 2014; Shi and Gong 2013; Turkanović et al. 2014) and it is also in the middle.
- Our scheme takes more storage cost in the user's smart card than (Hsieh and Leu 2014; Turkanović et al. 2014; Farash et al. 2015) and P1 in Chang and Le (2015), same as P2 in Chang and Le (2015), and less than (Choi et al. 2014; Shi and Gong 2013). The reason is that the corresponding parameters P, p and q should be stored. Among the four schemes with ECC mechanism, ours is the best.
- For the aspect of secret number storage in the gateway, our scheme is same as (Hsieh and Leu 2014), and better than the others. The storage cost for schemes in Turkanović et al. (2014) and Chang and Le (2015) is determined by the quantities of users and sensors. Even if there is only one user and one sensor, it takes more space in the gateway to store the secret bit strings in Turkanović et al. (2014) and Chang and Le (2015) than all of the other schemes for this index.
- The most important evaluation index is the security. The proposed scheme perfectly wins because it supports all required IoT security properties.

Table 11 Comparison of performance

	Our scheme	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Time of the user	$2T_m + 13T_h$	$8T_h$	$7T_h$	$3T_m + 5T_h$	$3T_m + 7T_h$	$7T_h$	$2T_m + 7T_h$	$11T_h$
Time of the sensor	$2T_m + 4T_h$	$2T_h$	$5T_h$	$2T_m + 4T_h$	$2T_m + 4T_h$	$5T_h$	$2T_m + 5T_h$	$7T_h$
Time of the gateway	$14T_h$	$5T_h$	$7T_h$	$T_m + 4T_h$	$T_m + 4T_h$	$8T_h$	$9T_h$	$14T_h$
Communication cost (bits)	3680	1280	4000	3840	4220	2720	3040	3520
Storage in the smart card (bits)	≈ 1120	800	800	≈ 1280	≈ 1440	480	≈ 1120	640
Secret number stored in the gateway (bits)	160	160	$160(Q_u + Q_s + 1)$	320	320	$160(Q_u + Q_s + 1)$	$160(Q_u + Q_s + 1)$	160
Security for IoT	Yes	No	No	No	No	No	No	No

[1]:Hsieh and Leu (2014);[2]:Turkanović et al. (2014);[3]:Shi and Gong (2013);[4]:Choi et al. (2014)

[5]:P1 in Chang and Le (2015);[6]:P2 in Chang and Le (2015);[7]: Farash et al. (2015)

9 Conclusion

In this paper, we point out that Hsieh and Leu scheme is insecure. To eliminate the weaknesses, we propose a new authentication scheme for WSN. Then we prove it secure by the formal proof and the formal verification. Also, we analyze the security properties of our scheme. The scheme not only provides mutual authentication between the user, the sensor and the gateway, but also withstands security analysis. Furthermore, it owns all IoT security properties. The results of comparison with the other recent schemes show that our scheme is well-performed and more practical.

Acknowledgments The authors thank the anonymous reviewers for their valuable comments. This research is supported by Fujian Education and Scientific Research Program for Young and Middle-aged Teachers under Grant No. JA14369, the National Natural Science Foundation of China under Grant No. 61300220, and it is also supported by PAPD and CICAEET.

References

Bresson E, Chevassut O, Pointcheval D (2003) Security proofs for an efficient password-based key exchange. In: Proceedings of the 10th ACM conference on Computer and communications security, ACM, p 241–250

Chang CC, Le HD (2015) A provably secure, efficient and flexible authentication scheme for ad hoc wireless sensor networks. IEEE Trans Wirel Commun. doi:10.1109/TWC.2015.2473165

Chen TH, Shih WK (2010) A robust mutual authentication protocol for wireless sensor networks. Etri J 32(5):704–712

Choi Y, Lee D, Kim J, Jung J, Nam J, Won D (2014) ecurity enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. Sensors 14(6):10,081–10,106

Das ML (2009) Two-factor user authentication in wireless sensor networks. Wirel Commun IEEE Trans 8(3):1086–1090

Fan R, Dj He, Xz Pan (2011) An efficient and dos-resistant user authentication scheme for two-tiered wireless sensor networks. J Zhejiang Univ Sci C 12(7):550–560

Fantacci R, Pecorella T, Viti R, Carlini C (2014) A network architecture solution for efficient iot wsn backhauling: challenges and opportunities. IEEE Trans Wirel Commun 21(4):113–119

Farash MS, Turkanović M, Kumari S, Hölbl M (2015) An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. Ad Hoc Netw. doi:10.1016/j.adhoc.2015.05.014

Guo P, Wang J, Geng XH, Kim CS, Kim JU (2014) A variable threshold-value authentication architecture for wireless mesh networks. J Internet Technol 15(6):929–935

Han W (2011) Weakness of a secured authentication protocol for wireless sensor networks using elliptic curves cryptography. IACR Cryptol ePrint Arch 2011:293

Hankerson D, Vanstone S, Menezes AJ (2004) Guide to elliptic curve cryptography. Springer Science & Business Media

Hayouni H, Hamdi M, Kim TH (2014) A survey on encryption schemes in wireless sensor networks. In: Advanced Software Engineering and Its Applications (ASEA), 2014 7th International Conference on, p 39–43

He D, Gao Y, Chan S, Chen C, Bu J (2010) An enhanced two-factor user authentication scheme in wireless sensor networks. Ad Hoc Sens Wirel Netw 10(4):361–371

He D, Kumar N, Chen J, Lee CC, Chilamkurti N, Yeo SS (2015) Robust anonymous authentication protocol for health-care applications using wireless medical sensor networks. Multimed Syst 21(1):49–60. doi:10.1007/s00530-013-0346-9

Hsieh WB, Leu JS (2014) A robust user authentication scheme using dynamic identity in wireless sensor networks. Wirel Pers Commun 77(2):979–989

Khan MK, Alghathbar K (2010) Cryptanalysis and security improvements of two-factor user authentication in wireless sensor networks. Sensors 10(3):2450–2459

Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Advances in Cryptology-CRYPTO 99, Springer, p 388–397

Kumar P, Lee HJ (2011) Cryptanalysis on two user authentication protocols using smart card for wireless sensor networks. In: Wireless Advanced (WiAd), IEEE, p 241–245

Li X, Ma J, Wang W, Xiong Y, Zhang J (2013a) A novel smart card and dynamic id based remote user authentication scheme for multi-server environments. Math Comput Model 58(1):85–95

Li X, Niu J, Khan MK, Liao J (2013b) An enhanced smart card based remote user password authentication scheme. J Netw Comput Appl 36(5):1365–1371

Liu Z, Wenger E, Groβ schä dl J (2014) Mote-ecc: energy-scalable elliptic curve cryptography for wireless-sensor-networks. In: Boureanu I, Owesarski P, Vaudenay S (eds) Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol 8479, Springer International Publishing, p 361–379, DOI 10.1007/978-3-319-07536-5_22

Nguyen KT, Laurent M, Oualha N (2015) Survey on secure communication protocols for the internet of things. Ad Hoc Netw 32:17–31

Ren Y, Shen J, Wang J, Han J, Lee S (2015) Mutual verifiable provable data auditing in public cloud storage. J Internet Technol 16(2):317–323

Shi W, Gong P (2013) A new user authentication protocol for wireless sensor networks using elliptic curves cryptography. Int J Distrib Sens Netw 2013:730831. doi:10.1155/2013/730831

Turkanović M, Brumen B, Hölbl M (2014) A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. Ad Hoc Netw 20:96–112

Vaidya B, Makrakis D, Mouftah HT (2010) Improved two-factor user authentication in wireless sensor networks. In: Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on, IEEE, p 600–606

Wang D, Wang P (2014) Understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks. Ad Hoc Netw 20:1–15

Watro R, Kong D, Cuti Sf, Gardiner C, Lynn C, Kruus P (2004) Tinypk: securing sensor networks with public key technology. In: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, ACM, p 59–64

Wu F, Xu L (2015) An improved and provable self-certified digital signature scheme with message recovery. Int J Commun Syst 28(2):344–357

Wu F, Xu L, Kumari S, Li X (2015a) A new and secure authentication scheme for wireless sensor networks with formal proof. Peer-to-Peer Netw Appl. doi:10.1007/s12083-015-0404-5

Wu F, Xu L, Kumari S, Li X (2015b) A novel and provably secure biometrics-based three-factor remote authentication scheme for mobile client-server networks. Comput Electr Eng 45:274–285

Wu F, Xu L, Kumari S, Li X, Alelaiwi A (2015c) A new authenticated key agreement scheme based on smart cards

providing user anonymity with formal proof. Secur Commun Netw 8(18):3847–3863

Xu L, Wu F (2015a) Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care. J Med Syst 39(2):1–9

Xu L, Wu F (2015b) An improved and provable remote user authentication scheme based on elliptic curve cryptosystem with user anonymity. Secur Commun Netw 8(2):245–260

Xue K, Ma C, Hong P, Ding R (2013) A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. J Netw Comput Appl 36(1):316–323

Yeh HL, Chen TH, Liu PC, Kim TH, Wei HW (2011) A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. Sensors 11(5):4767–4779

Yoo SG, Park KY, Kim J (2012) A security-performance-balanced user authentication scheme for wireless sensor networks. Int J Distrib Sens Netw 2012:382810. doi:10.1155/2012/382810