On the security of a provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks

He, Jun

2018-01-30


http://hdl.handle.net/10138/232579
https://doi.org/10.1177/1550147718756311

Downloaded from Helda, University of Helsinki institutional repository.
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.
On the security of a provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks

Jun He¹, Zheng Yang¹,², Jianxun Zhang¹, Wanping Liu¹ and Chao Liu¹

Abstract
In a recent paper, Chang and Le proposed an efficient smart card–based authenticated key exchange protocol (which is referred to as CL scheme) for heterogeneous ad hoc wireless sensor networks. However, we found that the CL scheme is subject to sensor capture attack which breaks the session key security of the CL scheme. An improved protocol is proposed to fix this problem.

Keywords
Cryptanalysis, authenticated key exchange, wireless sensor network, security attack, security model

Introduction
Wireless sensor networks (WSNs) play more and more important role in many critical practical applications, such as battlefield monitoring, healthy data acquisition, and many others. In WSN-based applications, data collected and stored by sensors are usually valuable and sensitive, which should be protected against unauthorized access from malicious attackers. An ideal solution for providing both secrecy and authentication is to apply the well-known cryptographic primitive—authenticated key exchange (AKE). The “form” of an AKE protocol here is closely related to the system model of the deployed WSNs.

System model
We here consider a system model (shown in Figure 1) for heterogeneous ad hoc WSNs which involve three types of principles, that is, users (U), sensors (S), and a gateway node (GWN). In this system, a user may hold a smart card, who wants to access or configure a sensor (on-site) in terms of actual field conditions. The user’s smart card is used as a security tool to authenticate the user via both memorable password and its stored long-term symmetric authentication key. Meanwhile, an AKE protocol is typically executed among user’s smart card, sensor, and GWN to establish a secure channel between user and sensor. Since the smart card and sensor do not share an authentication key, the online authentication is carried out by the remote GWN which shares symmetric authentication keys with users and sensors, respectively.

Related work
A sensor node is typically limited in power, storage, and computation resources. This has led to the
development of so-called “lightweight” multi-factor user authentication schemes and AKE protocols for WSNs. However, these AKE schemes have various protocol structures determined by underlying wireless sensor networks. For example, the protocols proposed in previous studies are specifically designed for Internet of things (IoT), wherein the GWN stands in the middle of the user and the sensor to take charge of the communication between them. We refer the reader to the work by Ferrag et al. for more recently proposed lightweight AKE schemes. In this article, we primarily concern about the AKE schemes under the system model shown in Figure 1. Note that building a secure AKE protocol for WSNs is non-trivial, since more than two participants and multiple authentication factors might be involved. Any subtle errors may lead to a broken protocol. Specifically, several previous schemes are found to be vulnerable to certain practical active attacks.

Chang and Le recently showed that a previous protocol proposed by Turkanovic et al. is vulnerable to stolen smart card attack, impersonation attack with node capture, sensor node spoofing attack, stolen verifier attack, and backward secrecy attack. As an improvement of Turkanovic et al.’s scheme, two lightweight AKE protocols (i.e. called P1 and P2) were introduced by Chang and Le (which will be referred to as CL scheme), which can provide both authentication and data privacy for the communication between user and sensor. The CL scheme can be considered as a trade-off between security and performance; that is, the CL scheme aims to satisfy the most desirable security attributes but provides high performance with low computation and communication cost (which are important for WSNs). In particular, only one authentication key is required to be stored at GWN. This dramatically improves the storage performance of GWN. Besides the efficiency, one of the advantages of the CL scheme is its provable security. The security of the CL scheme was proven in a model (which will be referred to as CL model) that is derived from the AKE model by Bellare et al. A protocol with security proof is reasonably necessary to formally show that its targeting security goals and properties (specified by the corresponding security model) are satisfied.

However, a security proof itself needs time to be further validated and discussed by researchers to avoid errors that were somehow overlooked. In 2016, Das et al. instigated the weaknesses of CL protocols, that is, the CL protocols cannot resist with smart card lost/stolen attacks and session-specific temporary information attacks. The password update phase of the CL protocols was also criticized by Das et al. by lacking of local password verification during the authentication phase. In order to enhance the security of the CL protocols, an efficient three-factor AKE for WSNs is presented to eliminate the weaknesses of the CL protocols.

Our results

In this work, we further revisit the security results of the CL scheme. We first notice that some key elements (e.g. the freshness and security experiment) in the CL security model are not well-formulated at all. This may lead to an awkward situation that no protocols can be proven secure in the CL model. We hereby clarify the ambiguities and fix up the loopholes in the CL model by showing a revised version. Furthermore, we notice that some recently proposed multi-factor AKE protocols are claimed to satisfy probably security. But similar problems can be also found in the models defined in these literature. Hence, our improved model can serve as a guidance to show how to formulate security definitions for three-party multi-factor AKE protocols. Researchers can simply derive new appropriate security models from ours to formally re-analyze the state-of-the-art “provable secure” AKE protocols for WSNs.

In particular, we figure out that the CL scheme cannot provide session key security as claimed by the authors since it is unable to resist with sensor capture attack, namely, the security result of Chang and Le is incorrect. We show that the attackers can make use of a corrupted sensor to impersonate arbitrary uncompromised users to uncompromised sensors as it wishes. We notice that such corruption of a sensor is allowed in the security analysis of P2. It is not hard to see that such attack is very harmful in real-world applications. For example, if an attacker plants a malicious sensor, then it can intercept all communications among uncompromised principles. In order to fix this problem, we give a simple and effective improvement with minimum modifications to P2. The improved scheme is also shown as an example to achieve the provable security with random oracles in our revised security model.
The problems of such freshness definition are summarized as follows:

- In the CL model, there is no RevealKey query (as in the work by Bellare et al.19) which allows the adversary to obtain the session key of a session. Hence, there is no way for an adversary to reveal the session key in the CL security experiment.
- CorruptSC and CorruptS queries are defined without any restrictions on this freshness definition. This implies that an adversary \( \mathcal{A} \) can trivially ask a CorruptSC(\( \Pi_{U_i}^S \)) query to obtain the long-term secret key of \( U_i \), even for the target session. Then \( \mathcal{A} \) is able to run some protocol instance with \( \Pi_{S_i}^U \) on behalf of \( U_i \) and select \( \Pi_{S_i}^U \) as the test session in the security experiment. In this situation, \( \mathcal{A} \) can always win using her own ephemeral secret key and the long-term secret key of \( U_i \).

According to the above discussions, we, first, have the conclusion that no protocol can be secure in the CL model.

Second, the security experiment is not clearly defined. There is no formulation on the timing of oracle queries which are performed in the security experiment. For example, whether an adversary can keep asking oracle queries after the Test query. This would also affect the security definition. Note that the adversary is allowed to ask many Test queries in the CL model. However, the guess bit \( b' \) returned by an adversary is not associated with a specific Test query so that we cannot correctly evaluate the advantage of an adversary.

**Improved security model**

We here define a security model which is suitable for proving the CL protocol. This model follows from indistinguishability-based AKE security models.19,22 In the following, let \( \ell, p, d \in \mathbb{N} \) be positive integers. In the execution environment, we fix a set of \( \ell \) honest users and \( p \) sensors at all. We may use a general \( ID \) in the following to denote one of these kinds of identity in \( \{ID, SID, ID_{GWN}\} \).

We assume that each honest party can only sequentially execute the protocol. This restriction is used to prevent replay attacks. This is characterized by a collection of oracles \( \{\Pi_{GWN}^S : \ell \in [\ell], s \in [d]\} \) for users, oracles \( \{\Pi_{ID}^S : j \in [p], t \in [d]\} \) for sensors, and oracles \( \{\Pi_{ID}^{ID_{GWN}} : z \in ([p + \ell]d)\} \) for GWN. Each oracle \( \Pi_{ID} \) behaves as party \( ID \) carrying out a process to execute the \( sth \) protocol instance with some partner (which is
determined during the protocol execution). All oracles of \( T\) have access to the authentication credential–related information stored on the devices (e.g. the smart card of a user and GWN). For time \( T\) and pre-specified time interval \( \Delta_T\), only one protocol instance can be executed within time period \([T, T + \Delta_T]\).

Moreover, we assume each oracle \( \Pi_{ID} \) maintains a list of independent internal state variables including (1) \( \Phi^i_{ID} \)—execution state \( \Phi^i_{ID} \in \{\text{accept}, \text{reject}\} \), (2) \( pid^i_{ID} \)—identity of the intended communication partner, (3) \( T^i_{ID} \)—timestamp at when the oracle is executed, (4) \( K^i_{ID} \)—session key, and (5) \( TM^i_{ID} \)—the protocol messages orderly sent and received by \( \Pi_{ID} \). Furthermore, we will always assume (for simplicity) that \( K^i_{ID} = \emptyset \) if an oracle has not reached accept state (yet).

**Adversarial model.** An active adversary \( A \) in this model is formalized as a probabilistic polynomial time (PPT) Turing Machine. The active attack powers of an adversary are formulated by allowing it to ask the following queries:

- **Execute\((ID_i, s, SID_j, t, ID_{GWN}, u)\):** If \( \Pi^i_{ID} \), \( \Pi^j_{SID} \), and \( \Pi^p_{ID_{GWN}} \) have not yet been used, this oracle executes the protocol between these instances and gives the transcript of this execution to the adversary. This oracle query models passive eavesdropping of a protocol execution.
- **Send\((ID_i, s, m)\):** The adversary can use this query to send any message \( m \) of his own choice to the oracle \( \Pi^i_{ID} \). The oracle will respond with the next message \( m' \) (if any) to be sent according to the protocol specification and its internal states. Oracle \( \Pi^j_{ID} \) would be initiated via sending the oracle the first message \( m = (T, ID_j) \) consisting of a special initialization symbol \( T \) and the identity \( ID_j \) of the intended partner. After answering a Send query, the oracle variables will be updated depending on the specific protocol.
- **RevealKey\((ID_i, s)\):** Oracle \( \Pi^i_{ID} \) responds with the contents of the variable \( K^i_{ID} \) if and only if the oracle \( \Pi^i_{ID} \) has reached an internal state \( \Phi^i_{ID} = \text{accept} \) and \( ID_i \neq ID_{GWN} \).
- **CorruptSC\((ID_i)\):** This query returns information stored on the smart card of the user \( ID_i \).
- **CorruptS\((SID_j)\):** This query returns information stored on the sensor \( SID_j \).
- **Test\((ID_i, s)\):** If oracle \( \Pi^i_{ID} \) has state \( \Phi^i_{ID} \neq \text{accept} \) or \( K^i_{ID} = \emptyset \) or \( ID_i = ID_{GWN} \), then the oracle returns some failure symbol \( \bot \).

*Secure AKE protocols.* During the protocol execution, two oracles may interact with each other to exchange messages for key establishment. We here define a notion concerning partnership following Bellare and Rogaway which is specifically introduced for three-party AKE. Note that the patterning notion in Chang and Le is defined in a similar way (but quite informal).

Let \( pf : T, s, pid^i_{ID}, T^i_{ID}, TM^i_{ID} \rightarrow (ID_j, t) \) be a partner function which is a map on given execution states of \( \Pi^i_{ID} \) points to its partner oracle \( \Pi^j_{ID} \), where \( ID_j \in pid^i_{ID} \) and \( t \in [d] \). The output of partner function should be uniquely determined by \( TM^i_{ID} \) of each oracle. In terms of the CL protocol, we realize \( pf \) by returning \((ID_j, t)\) if and only if all of the following conditions are held:

- Both \( \Pi^i_{ID} \) and \( \Pi^j_{ID} \) accept.
- \( ID_j \in pid^i_{ID} \) and \( ID_j \in pid^j_{ID} \).
- \( 0 \leq |T^i_{ID} - T^j_{ID}| \leq \Delta_T \).
- \( TM^i_{ID} \subseteq TM^j_{ID} \) or \( TM^j_{ID} \subseteq TM^i_{ID} \).
- \( \Pi^j_{ID} \) is the unique oracle (which meets the above conditions).

**Correctness.** We say an AKE protocol \( II \) is correct if two accepted oracles \( \Pi^i_{ID} \) and \( \Pi^j_{ID} \) are partner, then both oracles hold the same session key.

In order to define the security, we define two notions on **oracle freshness** that describe the active attacks allowed in the following security experiment. Let \( \Pi^i_{ID} \) be an accepted oracle with intended partner \( ID_j \), where \((ID_i, ID_j) \neq ID_{GWN} \). And let \( \Pi^j_{ID} \) be an oracle (if it exists), such that \( \Pi^j_{ID} \) is a partner oracle of \( \Pi^i_{ID} \), that is, \( pf(ID_i, s, pid^i_{ID}, T^i_{ID}, TM^i_{ID}) \rightarrow (ID_j, t) \).

**Definition 1.** Oracle Freshness with Forward Secrecy (OFFS)—the oracle \( \Pi^i_{ID} \) is said to be OFFS – fresh if none of the following conditions holds:

1. \( A \) queried \( \text{RevealKey}(ID_i, s) \);
2. \( \Pi^i_{ID} \) exists, \( A \) queried \( \text{RevealKey}(ID_i, t) \);
3. \( ID_j \) is user, \( A \) queried either \( \text{CorruptSC}(ID_i) \) or \( \text{CorruptS}(SID_j) \) prior to the acceptance of \( \Pi^j_{ID} \);
4. \( ID_j \) is sensor, \( A \) queried either \( \text{CorruptSC}(SID_j) \) or \( \text{CorruptSC}(ID_i) \) prior to the acceptance of \( \Pi^j_{ID} \).

**Definition 2.** Oracle Freshness without Forward Secrecy (OFwoFS)—the oracle \( \Pi^i_{ID} \) is said to be OFwoFS-fresh if none of the following conditions holds:

1. \( A \) queried \( \text{RevealKey}(ID_i, s) \);
2. \( \Pi^i_{ID} \) exists, \( A \) queried \( \text{RevealKey}(ID_j, t) \);
3. \( ID_j \) is user, \( A \) queried either \( \text{CorruptSC}(ID_j) \) or \( \text{CorruptS}(SID_j) \);
4. If \( ID_i \) is sensor, \( A \) queried either CorruptS\((SID_i)\) or CorruptSC\((ID_i)\).

We let \( OF = \{OFFS, OFwoFS\} \).

Security experiment \( EXP_{\Pi, A}(\lambda, OF) \). On input security parameter \( \lambda \), the security experiment is proceeded as a game between a challenger \( C \) and an adversary \( A \) based on AKE protocol II, where the following steps are performed:

1. At the beginning of the game, the challenger \( C \) implements the collection of oracles \( \{\Pi_1^i, i \in [l], s \in [d]\} \) for users and \( \{\Pi_{SID_j}^i, j \in [\rho], t \in [d]\} \) for sensors, and \( \{\Pi_{ID_{IDGWN}}^i, z \in [\rho + \ell d]\} \) for GWN. All long-term pre-shared key for users, sensors, and GWN are generated, respectively. \( C \) gives the adversary \( A \) all identities as input.

2. \( A \) may issue a polynomial number of queries regarding Execute, Send, CorruptSC, CorruptS, and RevealKey.

3. At some point, \( A \) may issue Test\((ID_{IDGWN}, s)\) queries during the experiment. After the Test query, \( A \) can keep asking other queries as it wishes.

4. At the end of the game, \( A \) may terminate and output a bit \( b' \) \( (b', ID_{IDGWN}, s) \) as its guess for \( b \) of Test\((ID_{IDGWN}, s)\) query. Then the experiment would return a failure symbol \( \bot \) if one of the following conditions is held: (1) \( A \) has not issued any Test query, or (2) the Test\((ID_{IDGWN}, s)\) query returns a failure symbol \( \bot \), or (3) the test oracle is not \( OF\)–fresh.

5. Finally, the experiment returns 1 if \( b = b' \); Otherwise, 0 is returned.

Formally, an instance \( \Pi_1^{ID} \) represents an online attack if both following conditions hold at the time of the Test query: (1) at some point, the adversary queried Send\((ID_{IDGWN}, s, m')\) and \( ID_i \) is not corrupted and (2) at some point, the adversary queried RevealKey\((ID_{IDGWN}, s)\) or Test\((ID_{IDGWN}, s)\). The number of online attacks represents a bound on the number of passwords the adversary could have tested in an online fashion. Let \( D \) be the maximum bit length of a user’s password. The maximum number of online attacks that \( A \) can perform to the owner of the test oracle is bound by \( d \).

Definition 3. Session Key Security—Given a correct AKE protocol II and an adversary \( A \) which runs the above security experiment without failure, we then define the advantage of \( A \) as follows

\[
Adv_{\Pi, A}^{\text{ake}}(\lambda, OF) := \Pr[EXP_{\Pi, A}^{\text{ake}}(\lambda, OF) = 1] - \frac{1}{2}
\]

We say that a correct AKE protocol II is session-key-secure if for all adversaries \( A \) the advantage such that \( Adv_{\Pi, A}^{\text{ake}}(\lambda, OF) \leq d/t/D + \text{negl}(\lambda) \) where \( \text{negl}(\lambda) \) is a negligible function in the security parameter \( \lambda \).

**On the security of the CL protocol**

A sensor capture attack against the CL protocol

**CL Protocol.** We first briefly review the P2 protocol.9 The user registration phase is shown in Figure 2. The protocol execution of P2 is depicted in Figure 3.

A sensor capture attack against P2. We here describe a sensor capture attack against the session key security of P2.9 The protocol P1 has the same problem.9 In this attack, we will make use of some compromised sensor to impersonate an arbitrary honest user. Note that the corruption of sensor is allowed in Theorem 2.9 In real-world applications, such compromised sensor might be also registered by an attacker as well. The core attack idea is to extract the secret \( Y_i = h(f_i || T_1) \) which is used to compute the authentication message \( H_i \). However, \( Y_i \) is only computed using an honest user \( U_i \)’s secret \( f_i \) and timestamp \( T_1 \). This implies that the \( Y_i \) can be reused many times in different sessions (to impersonate \( U_i \) to any uncompromised sensors) during \( T_1 + \Delta T \).

The concrete attack steps performed by an adversary \( A \) are described as follows:

- \( A \) corrupts some sensor say \( SID_j \) to obtain its long-term secret \( f_j \), that is, via CorruptS\((SID_j)\).
  - The goal of \( A \), for instance, is to impersonate some honest user \( U_i \) to communicate with another uncompromised sensor \( S_s \).
- \( A \) asks \( U_i \) to run a protocol instance \( \Pi_1^{ID} \) with sensor \( SID_j \), that is, via sending \( \Pi_1^{ID} \), the first message as Send\((ID_{IDGWN}, s, (T, SID_j))\). \( A \) intercepts the message \( m_1 = \{MI_i, Z_i, N_i, T_1\} \).
- \( A \) gets timestamp \( T_2 \) and honestly computes \( A_i := h(f_i || N_i || T_2) \) which is computable via compromised \( f_i \).
- \( A \) sends \( m_2 = \{MI_i, N_i, SID_j, A_i, T_1, T_2\} \) to GWN and receives \( m_3 = \{F_i, H_i, E_i, T_3\} \).

![Figure 2. Registration phase of Chang and Le.9](image-url)
\[ Y_i = h(f_i || T_1) \]

Select a \( a \in \mathbb{R} \) and \( K_i = aP \)

\[ Z_i = K_i \oplus Y_i \]

\[ N_i = h(Y_i || MI_i || SID_i) \]

\[ m_1 = \{ MI_i, Z_i, N_i, T_1 \} \]

Check \( T_3 \)

\[ Y'_i = F_{ij} \oplus h(f_j || T_3) \]

Check if \( H'_j = h(Y'_i) \)

Select \( b \in \mathbb{R} \) and \( K_j = bP \)

Get timestamp \( T_4 \)

\[ R_{ij} = h(K_i || T_4) \oplus K_j \]

Session key

\[ K = h(abP \cdot x) \]

\[ m_4 = \{ R_{ij}, E_i, T_4 \} \]

Figure 3. P2 of Chang and Le.⑨

- \( A \) computes \( Y_i := F_{ij} \oplus h(f_j || T_3) \). At this point, we stress that the \( A \) is able to impersonate \( U_i \) to \( SID_v \) at time \( T_1 \). We next show that the attack is possible if the time consumption of the above execution is less than \( \Delta_r \). This is very likely to happen, for example, \( \Delta_r = 1 \) min.

- \( A \) selects a random \( a^* \) and \( K'_i = a^*P \) and computes \( Z'_i = K'_i \oplus Y_i \) and \( N'_i = h(Y_i || MI_i || SID_i) \).

The message \( m_1' = \{ MI_i, Z'_i, N'_i, T_1 \} \) is sent to some oracle of sensor \( SID_i \), say \( \Pi'_{SID} \).

- Meanwhile \( f_j := h(SID_j || X_{GW}) \) would run the protocol with GWN honestly. \( \Pi'_{SID} \) replies \( A \) with message \( m_3' = \{ R_{ij}, E_i, T_3 \} \) where \( R_{ij} = h(K_i || T_4) \oplus K_j \) and \( K'_i = b^*P \) is chosen by \( A \).

- Finally, \( A \) could compute \( K' := h(K'_i || T_4) \oplus R_{ij} \) and session key \( K' := h(a^*b^*P \cdot x) \), where \( a^*b^*P \cdot x \) is the x-coordinate of the point \( a^*b^*P \).

Note that based on such corrupted sensor, \( A \) is able to launch (unlimited) the above attack to impersonate any honest users \( U_i \) to any uncompromised sensors \( S_i \) at an arbitrary time \( T_1 \) chosen by \( A \). This would enable us to break the session key security of P2. It is not hard to see that this attack is also very harmful in practice. We highlight that the similar attack can be also mounted to P1.

Since the attack mainly exploits the corrupted sensor to obtain the secret \( Y_i \) which can be extracted from the ciphertext \( F_{ij} \) based on the corrupted secret key \( f_j \). Therefore, the attack time roughly equals to the message delivery time of \( m_1 \) and \( m_3 \) and computation time of two hash operations (for computing \( A_j \) and \( Y_i \)). If the attacker fails within a time \( T_1 + \Delta_r \), it can just choose another time \( T_1' \) to start the attack again.

**An improved protocol**

In order to fix the problem of P2 (cf. P1), the computation of \( Y_i \) should be modified. Specifically, we suggest adding the sensor’s identity \( SID_i \) into the generation of \( Y_i \), that is, \( Y_i = h(f_j || MI_i || SID_i) \). We also include the messages exchanged between user and sensor into the session key generation so that the resultant session key is uniquely bound to a specific session. The rest of protocol execution of P2 remains the same. The improved protocol is shown in Figure 4 with minimum modifications. The improved protocol preserves all the properties of the original protocol such as session key security and perfect forward secrecy (PFS).
The improved protocol consists of four phases which are described as follows:

1. **User registration phase.** The user registration phase is shown in Figure 2.
   - **The user** $U_i$ selects an identity $ID_i$ and a password $pw_i$ and generates a random nonce $r_i$. $U_i$ computes $MP_i := h(r_i || pw_i)$ and sends the registration message $m_{\text{reg}} = \{ID_i, MP_i\}$ to the GWN over some secure channel.
   - Upon receiving $m_{\text{reg}}$, GWN verifies the user’s identity and generates a random nonce $r_i$. Then, GWN computes a pseudonym $MI_i := h(r_i || ID_i)$ for $U_i$ and uses its long-term key $X_{GW}$ to compute $f_i := h(MI_i || X_{GW})$ and $e_i := MP_i \oplus f_i$. Finally, GWN creates a smart card $SC_i = \{MI_i, e_i\}$ and issues it to the user $U_i$ over some secure channel.
   - After receiving the smart card $SC_i$, $U_i$ writes the value $r_i$ into it. Eventually, the smart card $SC_i$ stores $\{MI_i, e_i, r_i\}$.

2. **Sensor registration phase.** The sensor registration phase should be also run over some secure channel. As for a sensor node $S_j$, GWN first selects a unique identity $SID_j$. Then, GWN computes $f_j := h(SID_j || X_{GW})$. Meanwhile, the tuple $(SID_j, f_j)$ is stored by GWN. Finally, the identity $SID_j$ and the authentication key $f_j$ are written into the sensor node $S_j$.

3. **Authentication and key exchange phase.** The following protocol steps are performed:
   1. **Message 1 (m1):** $U_i$ to $S_j$
      - $U_i$ inputs its identity $ID_i$ and password $pw_i$.
      - Upon receiving $\{ID_i, pw_i\}$, the smart card $SC_i$ computes $MP_i := h(r_i || pw_i)$ and the corresponding authentication key $f_i := e_i \oplus MP_i$.
      - $SC_i$ gets the current timestamp $T_1$ and a ciphertext $Z_i := K_i \oplus Y_i$ and an authentication message $N_i := h(Y_i || MI_i || SID_j)$.
      - The message $m_1 = \{MI_i, Z_i, N_i, T_1\}$ is sent to the sensor $S_j$.
   2. **Message 2 (m2):** $S_j$ to GWN
      - Upon receiving $m_1$, $S_j$ checks whether or not $T_1$ is a valid timestamp.

![Figure 4. The improved P2.](image-url)
• $S_j$ gets a timestamp $T_2$ and computes an authentication message $A_j := h(f_j||N_j||T_2)$.

• The message $m_2 = \{M_l, N_l, SID_l, A_j, T_1, T_2\}$ is sent to GWN.

3. **Message 3 (m3): GWN to $S_j$**

• Upon receiving $m_2$, GWN aborts if either $T_1$ or $T_2$ is not valid.

• GWN computes verification messages $f_j := h(SID_l||X_{GWN})$ and $A_j := h(f_j||N_l||T_2)$, $f_i := h(M_l||X_{GWN})$, and $N_i := h(Y_i||M_l||SID_l)$. GWN rejects if $N_i \neq N_j$ and $A_j \neq A_j$.

• Next, GWN gets a timestamp $T_3$ and computes an authentication key $Y_i := h(f_j||T_1||SID_l)$ and a ciphertext $F_y := Y_i \oplus h(f_j||T_2)$, and authentication messages $H_i := h(Y_i)$ and $E_i := h(f_j||N_j)$.

• The message $m_3 = \{F_y, H_i, E_i, T_3\}$ is sent to the sensor $S_j$.

4. **Message 4 (m4): $S_j$ to $U_i$**

• Upon receiving $m_3$, $S_j$ checks $T_3$, and computes the verification key $Y_i := F_y \oplus h(f_j||T_3)$. $S_j$ rejects if $H_i \neq h(Y_i)$.

• $S_j$ selects a random value $b \in \mathbb{Z}_p$ and computes an ephemeral public key $K_j := bP$. Next, $S_j$ gets a timestamp $T_4$ and computes a ciphertext $R_y := h(K_j||T_4) \oplus K_j$.

• The session key is computed as $K := h(abP.x||K_i||K_j||M_l||SID_l||T_1||T_4)$, where $abP.x$ is the $x$-coordinate of the point $abP$. The input of the hash function covers almost all protocol messages exchanged between user and sensor, so that the session key is uniquely bound to a session identified by such transcript.

• The message $m_4 = \{R_y, E_i, T_4\}$ is sent to the user $U_i$.

5. **Finalize: User $U_i$**

• Upon receiving message $m_4$, $U_i$ checks the timestamp $T_4$ and computes the verification message $E_i := h(f_j||N_i)$ and rejects if $E_i \neq E_i$.

• $U_i$ computes the session key $K := h(abP.x||K_i||K_j||M_l||SID_l||T_1||T_4)$.

4. **User authentication key change phase.** Suppose the user has been authenticated in an AKE session with his or her old password $pw_i$, $SC_i$ has $MP_i$ in that session. Then the user $U_i$ inputs his new password $pw_i^{new}$. After this, $SC_i$ computes $MP_i^{new} := h(r_i||pw_i^{new})$ and $e_i^{new} := e_i \oplus MP_i \oplus MP_i^{new}$. Finally, $SC_i$ stores $e_i^{new}$.

The **decisional Diffie–Hellman assumption.** Let $\mathbb{G}$ be a group of prime order $p$ under an elliptic curve. Let $P$ be a random generator of $\mathbb{G}$. The decisional Diffie–Hellman (DDH) problem is stated as follows: given a tuple $(P, aP, bP, cP)$ for $a, b, c \rightarrow \mathbb{Z}_p^*$ as input, it is hard to distinguish whether $c = ab$.

**Definition 4.** For a group $\mathbb{G}$ of prime order $p$ and an adversary $\mathcal{D}$, we define the following experiment:

\[
\text{EXP}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda) = P \leftarrow \mathbb{G}; (a, b, c) \xleftarrow{\mathcal{U}} \mathbb{Z}_p^*; \beta \leftarrow \mathcal{D}(P, aP, bP, cP); \text{if } \beta = 1 \text{ then } \Gamma \leftarrow abP, \text{ otherwise } \Gamma \leftarrow \gamma P; \beta' \leftarrow \mathcal{D}(P, aP, bP, cP); \text{ if } \beta = \beta' \text{ then return } 1, \text{ otherwise return } 0.
\]

The advantage of $\mathcal{D}$ in above experiment is defined as follows

\[
\text{Adv}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda) := \left| \text{Pr}[\text{EXP}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda) = 1] - \frac{1}{2} \right|
\]

We say that the DDH problem relative to a group $\mathbb{G}$ of prime order $p$ holds, if for all PPT adversaries $\mathcal{D}$ the advantage $\text{Adv}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda)$ is a negligible function in $\lambda$.

**Theorem 1.** Suppose that the hash function $h$ is modeled as random oracle, the output bit length of the hash function is $\nu$, and the DDH problem relative to a group $\mathbb{G}$ with prime order $p$ holds, then the fixed protocol $P'_{2i}$ is session-key-secure such that $\text{Adv}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda, OFFS) \leq \frac{d_i}{D} + \frac{q_h}{2^{\nu-3}} + (d_i)^2 \cdot \text{Adv}_{\mathbb{G}, p, \mathcal{D}}^{\text{DDH}}(\lambda)$, where $q_h$ is the number of allowed random oracle queries.

Due to this modification, each credential $Y_i$ is particularly bound to an execution between the user $ID_i$ and the sensor $SID_i$ at time $T_1$. Thus, the attacker is unable to abuse $Y_i$ involving some malicious sensor $SID_i$ to impersonate $ID_i$ to some honest sensor $SID_j$.

**Proof.** The proof is proceeded following the game-based approach.$^{25}$ Let $\text{Adv}_{\xi}$ denote the advantage of $\mathcal{A}$ in Game $\xi$.

**Game 0.** The first game is the real security experiment which is run between an adversary $\mathcal{A}$ and an AKE challenger $\mathcal{C}$. Thus, we have that

\[
\text{Adv}_{\mathbb{G}, \mathcal{C}, \mathcal{A}}^{\text{AKE}}(\lambda) = \text{Adv}_{\mathcal{A}}(\lambda)
\]
Meanwhile, C will simulate the queries of A as follows:

- Execute(ID_i, s, SID_j, t, ID_GWN, u) and Send(ID_i, s, m) queries will be honestly simulated as defined in section “Security model” following our protocol specification.
- RevealKey(ID_i, s): The session key returned by this query is $SK = h(abP.x||K_j||MI_i||SID_j||T_1||T_2)$. Hence, the probability that the adversary can correctly generate a valid $N_i$ for $A$ is bound to $1/2^{n-1}$. Furthermore, we add another similar abort rule that the challenger aborts if an oracle of $GWN$ receives a valid $A_j$ which is not sent by any fresh oracle of $GWN$. With the similar arguments in the previous game, we have that

$$Adv_0 \leq Adv_1 + \frac{d\ell}{D}$$

Game 1. In this game, the challenger aborts if the adversary $A$ asks the random oracle $h(\cdot)$ with a user $U_i$'s password $pw_i$ and its randomness $r_i$ as input, that is, $MI_i = h(r_i||pw_i)$. This implies that the adversary can ask a Send(ID_i, s, m) query, such that $pw_i \in m$. Recall that $pw_i$ is used to compute $MP_i$, which is used to decrypt $f_i = MI_i \oplus e_i$. In this case, the adversary must be able to succeed in online attacks.

The length of the password is assumed to be $D$. If the adversary can input the correct $pw_i$, then it is able to impersonate the user (e.g. after obtaining the victim's smart card without compromising the secret information stored in the smart card). The probability that $A$ correctly guesses is about $1/D$. Note that $A$ can try $d\ell$ times which is the maximum number of Send query to all $\ell$ parties. The winning probability of a successful online attack is bound to $d\ell/D$. Thus, we have that

$$Adv_0 \leq Adv_1 + \frac{d\ell}{D}$$

Game 2. In this game, the challenger first aborts if a fresh oracle of $GWN$ receives a valid $N_i$ which is not sent by any fresh oracle of $U_i$ at a valid time. To generate a valid $N_i$, the adversary $A$ has to either (1) break the one-wayness of the hash function or (2) randomly guess (correctly) the output of the random oracle $h(\cdot)$ for generating $N_i$. If the case (1) holds, the adversary $A$ may be able to ask the random oracle $h(Y||MI_i||SID_j)$ and $h(f_i)$ with the correct $f_i$ but without asking the corresponding $CorruptSC$ query. The abort event implies that the adversary is able to either find the pre-image of the hash value (after obtaining the protocol transcript via Execute query) or forge the protocol message. Since each $N_i$ is used only once, either $f_i$ or $h(f_i||T_1)$ is protected by the one-wayness of the hash function $h$. Hence, the probability of correctly generating a valid $N_i$ for $A$ is bound to $1/2^{n-1}$. Furthermore, we add another similar abort rule that the challenger aborts if an oracle of $GWN$ receives a valid $A_j$ which is not sent by any fresh oracle of $GWN$. With the similar arguments in the previous game, we have that

$$Adv_1 \leq Adv_2 + \frac{q_h}{2^{n-2}}$$

Game 3. In this game, the challenger aborts if one of the following conditions holds: (1) a fresh oracle of $S_j$ receives a valid $H_i$ which is not sent by any fresh oracle of $GWN$ or (2) a fresh oracle of $U_i$ receives a valid $E_i$ which is not sent by any fresh oracle of $GWN$. With the similar arguments in the previous game, we have that

$$Adv_2 \leq Adv_3 + \frac{q_h}{2^{n-2}}$$

As a result in this game, we have the fact that the test oracle must have a partner oracle; otherwise, the challenger has aborted.

Game 4. In this game, we try to reduce the security to the DDH hard problem. We change the session key of the test oracle to be a random value. If there exists an adversary which can distinguish this game from the previous game, then we can make use of it to build an algorithm $D$ to solve the DDH problem. Given a DDH challenge instance $(xP, yP, cP)$, the job of $D$ is to distinguish $c = ?xy$. Meanwhile, $D$ first guesses the test oracle and its partner oracle. If its guess is incorrect then it aborts. The abort probability is bound to $1/(d\ell)^2$. In the following, we assume that $D$’s guess is correct. Then $D$ sets the ephemeral DH key of the test oracle to $xP$ and its partner oracle’s DH key as $yP$. The session key of the test oracle is computed as $h(cP||xP||yP||MI_i||SID_j||T_1||T_4)$. The other simulations are similar to the previous games. So that if $c = ab$, then the game is identical to the previous game. Otherwise, it is equivalent to this game. Due to the hardness of the DDH problem, we have that

$$Adv_3 \leq Adv_4 + Adv_{D_{G,P,D}(\lambda)}$$
As the bit in the Test query is not used anymore, the advantage of an adversary in this game is zero, that is, \(Adv = 0\). To sum up all probabilities in the above games, we get the result of this theorem.

**Performance comparison**

In this section, we compare our improved scheme with related existing similar lightweight AKE protocols of Shi and Gong\(^4\) (SG), Choi et al.\(^6\) (CLK+), Turkanovic et al.\(^7\) (TBH), and Das et al.\(^11\) (DKO+) which are run under the same system model.

We approximately test the performance of compared protocols on PC with Intel Core i7, 4.2 GHz + 8GB RAM, and Python 2.7. Let \(tmul\) and \(th\) denote the estimated experimental time of 224-bits elliptic curve point multiplication and a SHA256 hash function operation (for 128 bytes message), respectively. We omit the cost of XOR in the comparison, which is much smaller than the other two types of operations. We specifically have that \(tmul = 3.8\) ms and \(th = 0.007\) ms. We consider a biometric computation required by Das et al.’s scheme as a hash operation. The comparison of computational cost is listed in Table 2.

In Figure 5, we compare the communication cost of our scheme with existing schemes. We make reasonable assumptions that each identity is 128 bits, each random nonce is 256 bits, each timestamp is 64 bits, each elliptic curve cryptography (ECC) group value is 224 bits, and each hash value is 256 bits.

Turkanovic et al.’s\(^7\) scheme (TBH) is more efficient than all other listed protocols but it does not provide PFS. Our improved scheme inherits the high performance of \(P2\),\(^9\) which provides a trade-off between security and performance. Although the computation cost of our scheme is slightly more expensive than that of Shi et al.’s scheme (SG), our scheme is the most efficient one for the user and GWN among those compared protocols which satisfy PFS. Furthermore, our scheme has the best communication performance.

**Conclusion**

In this article, we revisited a recently proposed AKE scheme\(^9\) for WSNs. We have shown that there is a security vulnerability in this scheme. We also demonstrated a concrete attack and gave a solution for avoiding this attack. Some improvements for the security model have been given for analyzing the improved protocol. Since a good AKE protocol might need to provide as many security properties as possible, it might be interesting (as a future work) to further improve our protocol and Das et al.’s\(^11\) scheme by incorporating more security attributes as introduced in recent literature.\(^{12,24}\) Of course, one could also strengthen our security model to cover more active attacks for specific protocols.

Among those security properties, PFS has been a de facto standard property of AKE. Recently proposed lightweight AKE protocols,\(^{4,6,9,12,25}\) all achieve PFS based on Diffie–Hellman key exchange (DHKE). Since two exponential operations are required in DHKE, it is not efficient for either smart card or sensor due to its low processing resources. Another open problem is to construct lightweight AKE protocols which provide PFS without relying on Diffie–Hellman–like primitives.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was supported by the Research Project of the Humanities and Social Sciences of the Ministry of Education of China (grant nos 16YJC870018 and 15YJC790061), National Natural Science Foundation of China (grant nos 11647097, 61502064, and 61503052), Open project of Key laboratory of higher education of Sichuan
References