


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

International Journal of Distributed
Sensor Networks
2018, Vol. 14(1)
© The Author(s) 2018
DOI: 10.1177/1550147718756311
journals.sagepub.com/home/dsn


Jun He¹, Zheng Yang^{1,2} , 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

Date received: 28 September 2017; accepted: 5 January 2018

Handling Editor: Al-Sakib Khan Pathan

Introduction

Wireless sensor networks (WSNs) play more and more important role in many critical practical applications, such as battle field 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

¹School of Computer Science and Engineering, Chongqing University of Technology, Chongqing, China

²Department of Computer Science, University of Helsinki, Helsinki, Finland

Corresponding author:

Zheng Yang, Department of Computer Science, University of Helsinki, Gustaf Hällströmin katu 2b, Helsinki 00014, Finland.

Email: zheng.yang@helsinki.fi



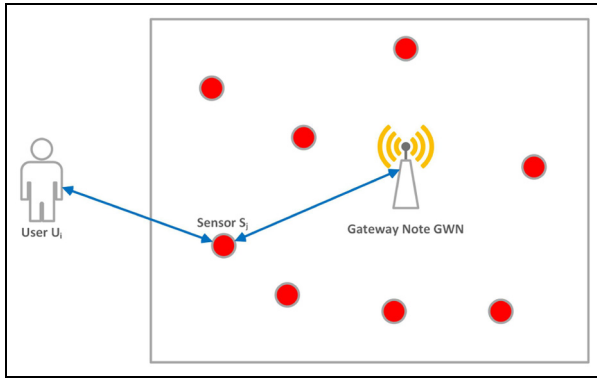


Figure 1. System model.

development of so-called “lightweight” multi-factor user authentication schemes and AKE protocols for WSNs.^{1–15} However, these AKE schemes have various protocol structures determined by underlying wireless sensor networks. For example, the protocols proposed in previous studies^{10,12–17} 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.¹⁸ for more recently proposed lightweight AKE schemes. In this article, we primarily concern about the AKE schemes^{1,2,6,7,9–11} 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^{1,5,7,9,11} to be vulnerable to certain practical active attacks.

Chang and Le⁹ recently showed that a previous protocol proposed by Turkanovic et al.⁷ 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 Le⁹ (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.¹⁹ 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.¹¹ 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^{11,14} 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^{11,14} 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 Le⁹ Theorem 2 is incorrect. We show that the attackers can make use of a corrupted sensor to impersonate arbitrary uncorrupted users to uncompromised sensors as it wishes. We notice that such corruption of a sensor is allowed in the security analysis of P2.⁹ 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 uncorrupted 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.

Table 1. Notations.

U_i	User
GWN	Gateway node
S_j	Sensor node
ID_i	Identity of a party
SID_j	Identity of a sensor
ID_{GWN}	Identity of GWN
ID	General identity which might denote an Identity in $\{ID, SID, ID_{GWN}\}$
pw_i	Password of U_i
SC_i	Smart card of U_i
X_{GWN}	Long-term secret of GWN
T_i	Timestamp
Δ_T	Expected transmission delay
r_i, K_i	Random numbers
P	Point on an elliptic curve
$P \cdot x$	x -axis value of the point P
$\ , \oplus, h()$	Concatenation, XOR, and hash operation

Organizations

The remainder of the article is organized as follows. Section “Security model” discusses the problems of the CL model and introduces our revised security model. Subsequently, in section “On the security of the CL protocol,” we introduce an attack against the CL scheme and our improvement. Section “Performance comparison” shows our experimental and comparison results. Finally, section “Conclusion” concludes this article.

Security model

The CL model⁹ is defined very vaguely and informally. Based on the CL model, it cannot be used to analyze the security and the corresponding proof of the CL protocol. In the sequel, we first give some remarks on relevant important notions which are defined inappropriately in the CL model. Second, we re-define the security definitions to obtain an improved security model. The notations used in the rest of the article are first recalled in Table 1.

Remarks on the CL security model

We hereby mainly discuss the problems of the CL model from the following two perspectives: (1) freshness and (2) security experiment. Note that all these elements are fundamental to a security model, which are unfortunately not well addressed by the CL model. We review the freshness defined in the CL model:

Freshness. The instance $\Pi_{U_i}^u$ or $\Pi_{S_j}^t$ is fresh if their session key SK has not been revealed to the adversary \mathcal{A} .⁹

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($\Pi_{U_i}^t$) 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 $\Pi_{S_j}^t$ on behalf of U_i and select $\Pi_{S_j}^t$ 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 $\ell, \rho, d \in \mathbb{N}$ be positive integers. In the execution environment, we fix a set of ℓ honest users and ρ sensors at all. We may use a general \mathcal{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 $\{\Pi_{ID_i}^s : i \in [\ell], s \in [d]\}$ for users, oracles $\{\Pi_{SID_j}^t : j \in [\rho], t \in [d]\}$ for sensors, and oracles $\{\Pi_{ID_{GWN}}^z : z \in [(\rho + \ell)d]\}$ for GWN. Each oracle Π_{ID}^s behaves as party \mathcal{ID} carrying out a process to execute the s th protocol instance with some partner (which is

determined during the protocol execution). All oracles of \mathcal{ID} 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_i and pre-specified time interval Δ_T , only one protocol instance can be executed within time period $[T_i, T_i + \Delta_T]$.

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

Adversarial model. An active adversary \mathcal{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 $\Pi_{ID_i}^s$, $\Pi_{SID_j}^t$, and $\Pi_{ID_{GWN}}^u$ 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**(\mathcal{ID}_i, s, m): The adversary can use this query to send any message m of his own choice to the oracle $\Pi_{\mathcal{ID}_i}^s$. The oracle will respond with the next message m^* (if any) to be sent according to the protocol specification and its internal states. Oracle $\Pi_{\mathcal{ID}_i}^s$ would be initiated via sending the oracle the first message $m = (T, \mathcal{ID}_j)$ consisting of a special initialization symbol T and the identity \mathcal{ID}_j of the intended partner. After answering a Send query, the oracle variables will be updated depending on the specific protocol.
- **RevealKey**(\mathcal{ID}_i, s): Oracle $\Pi_{\mathcal{ID}_i}^s$ responds with the contents of the variable $K_{\mathcal{ID}_i}^s$ if and only if the oracle $\Pi_{\mathcal{ID}_i}^s$ has reached an internal state $\Phi_{\mathcal{ID}_i}^s = \text{accept}$ and $\mathcal{ID}_i \neq ID_{GWN}$.
- **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**(\mathcal{ID}_i, s): If oracle $\Pi_{\mathcal{ID}_i}^s$ has state $\Phi_{\mathcal{ID}_i}^s \neq \text{accept}$ or $K_{\mathcal{ID}_i}^s = \emptyset$ or $\mathcal{ID}_i = ID_{GWN}$, then the oracle returns some failure symbol \perp . Otherwise, it flips a random bit b , samples a random key $K_0 \xrightarrow{\$} \mathcal{K}_{\text{ake}}$, and sets $K_1 = K_{\mathcal{ID}_i}^s$, where \mathcal{K}_{ake} is a session key space. Finally, the key K_b is returned. The oracle $\Pi_{\mathcal{ID}_i}^s$ selected by adversary in this query is called as test oracle.

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 Rogaway²² which is specifically introduced for three-party AKE. Note that the patterning notion in Chang and Le⁹ is defined in a similar way (but quite informal).

Let $pf : \mathcal{ID}_i, s, pid_{\mathcal{ID}_i}^s, T_{\mathcal{ID}_i}^s, TM_{\mathcal{ID}_i}^s \rightarrow (\mathcal{ID}_j, t)$ be a partner function²² which is a map on given execution states of $\Pi_{\mathcal{ID}_i}^s$ points to its partner oracle $\Pi_{\mathcal{ID}_j}^t$, where $\mathcal{ID}_j \in pid_{\mathcal{ID}_i}^s$ and $t \in [d]$. The output of partner function should be uniquely determined by $TM_{\mathcal{ID}_i}^s$ of each oracle. In terms of the CL protocol, we realize pf by returning (\mathcal{ID}_j, t) if and only if all of the following conditions are held:

- Both $\Pi_{\mathcal{ID}_i}^s$ and $\Pi_{\mathcal{ID}_j}^t$ accept.
- $\mathcal{ID}_j \in pid_{\mathcal{ID}_i}^s$ and $\mathcal{ID}_i \in pid_{\mathcal{ID}_j}^t$;
- $0 \leq |T_{\mathcal{ID}_i}^s - T_{\mathcal{ID}_j}^t| \leq \Delta_T$;
- $TM_{\mathcal{ID}_i}^s \subseteq TM_{\mathcal{ID}_j}^t$ or $TM_{\mathcal{ID}_j}^t \subseteq TM_{\mathcal{ID}_i}^s$.
- $\Pi_{\mathcal{ID}_j}^t$ is the unique oracle (which meets the above conditions).

Correctness. We say an AKE protocol Π is correct if two accepted oracles $\Pi_{\mathcal{ID}_i}^s$ and $\Pi_{\mathcal{ID}_j}^t$ 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 $\Pi_{\mathcal{ID}_i}^s$ be an accepted oracle with intended partner \mathcal{ID}_j , where $(\mathcal{ID}_i, \mathcal{ID}_j) \neq ID_{GWN}$. And let $\Pi_{\mathcal{ID}_j}^t$ be an oracle (if it exists), such that $\Pi_{\mathcal{ID}_i}^s$ is a partner oracle of $\Pi_{\mathcal{ID}_j}^t$, that is, $pf(\mathcal{ID}_i, s, pid_{\mathcal{ID}_i}^s, T_{\mathcal{ID}_i}^s, TM_{\mathcal{ID}_i}^s) \rightarrow (\mathcal{ID}_j, t)$.

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

1. \mathcal{A} queried **RevealKey**(\mathcal{ID}_i, s);
2. If $\Pi_{\mathcal{ID}_j}^t$ exists, \mathcal{A} queried **RevealKey**(\mathcal{ID}_j, t);
3. If \mathcal{ID}_i is user, \mathcal{A} queried either **CorruptSC**(ID_i) or **CorruptS**(SID_j) prior to the acceptance of $\Pi_{\mathcal{ID}_i}^s$;
4. If \mathcal{ID}_i is sensor, \mathcal{A} queried either **CorruptS**(SID_i) or **CorruptSC**(ID_j) prior to the acceptance of $\Pi_{\mathcal{ID}_i}^s$.

Definition 2. Oracle Freshness without Forward Secrecy (OFwoFS)—the oracle $\Pi_{\mathcal{ID}_i}^s$ is said to be OFwoFS–fresh if none of the following conditions holds:

1. \mathcal{A} queried **RevealKey**(\mathcal{ID}_i, s);
2. If $\Pi_{\mathcal{ID}_j}^t$ exists, \mathcal{A} queried **RevealKey**(\mathcal{ID}_j, t);
3. If \mathcal{ID}_i is user, \mathcal{A} queried either **CorruptSC**(ID_i) or **CorruptS**(SID_j);

4. If \mathcal{ID}_i is sensor, \mathcal{A} queried either $\text{CorruptS}(\text{SID}_i)$ or $\text{CorruptSC}(\text{ID}_j)$.

We let $\mathcal{OF} = \{\text{OFFS}, \text{OFwoFS}\}$.

Security experiment $\text{EXP}_{\Pi, \mathcal{A}}^{\text{ake}}(\lambda, \mathcal{OF})$. On input security parameter 1^λ , the security experiment is proceeded as a game between a challenger \mathcal{C} and an adversary \mathcal{A} based on AKE protocol Π , where the following steps are performed:

1. At the beginning of the game, the challenger \mathcal{C} implements the collection of oracles $\{\Pi_{\mathcal{ID}_i}^s : i \in [\ell], s \in [d]\}$ for users and $\{\Pi_{\text{SID}_j}^t : j \in [\rho], t \in [d]\}$ for sensors, and $\{\Pi_{\text{ID}_{\text{GWN}}}^z : z \in [(\rho + \ell)d]\}$ for GWN. All long-term pre-shared key for users, sensors, and GWN are generated, respectively. \mathcal{C} gives the adversary \mathcal{A} all identities as input.
2. \mathcal{A} may issue a polynomial number of queries regarding Execute , Send , CorruptSC , CorruptS , and RevealKey .
3. At some point, \mathcal{A} may issue $\text{Test}(\mathcal{ID}_i, s)$ queries during the experiment. After the Test query, \mathcal{A} can keep asking other queries as it wishes.
4. At the end of the game, \mathcal{A} may terminate and output a bit (b', \mathcal{ID}_i, s) as its guess for b of $\text{Test}(\mathcal{ID}_i, s)$ query. Then the experiment would return a failure symbol \perp if one of the following conditions is held: (1) \mathcal{A} has not issued any Test query, or (2) the $\text{Test}(\mathcal{ID}_i, s)$ query returns a failure symbol \perp , or (3) the test oracle is not \mathcal{OF} -fresh.
5. Finally, the experiment returns 1 if $b = b'$; Otherwise, 0 is returned.

Formally, an instance $\Pi_{\mathcal{ID}_i}^s$ represents an online attack if both following conditions hold at the time of the Test query: (1) at some point, the adversary queried $\text{Send}(\mathcal{ID}_i, s, m^*)$ and ID_i is not corrupted and (2) at some point, the adversary queried $\text{RevealKey}(\mathcal{ID}_i, s)$ or $\text{Test}(\mathcal{ID}_i, s)$. 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 \mathcal{A} can perform to the owner of the test oracle is bound by d .

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

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{ake}}(\lambda, \mathcal{OF}) := \left| \Pr[\text{EXP}_{\Pi, \mathcal{A}}^{\text{ake}}(\lambda, \mathcal{OF}) = 1] - \frac{1}{2} \right|$$

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

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_j . However, Y_i is only computed involving an honest user U_i 's secret f_i and timestamp T_1 . This implies that the Y_i can be re-used 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 \mathcal{A} are described as follows:

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

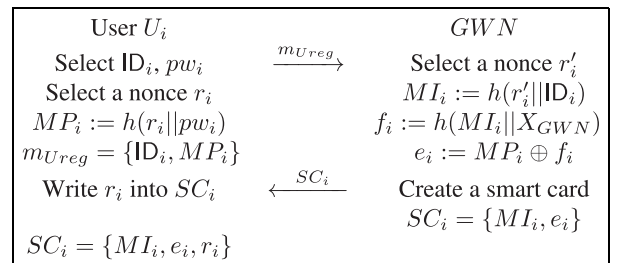


Figure 2. Registration phase of Chang and Le.⁹

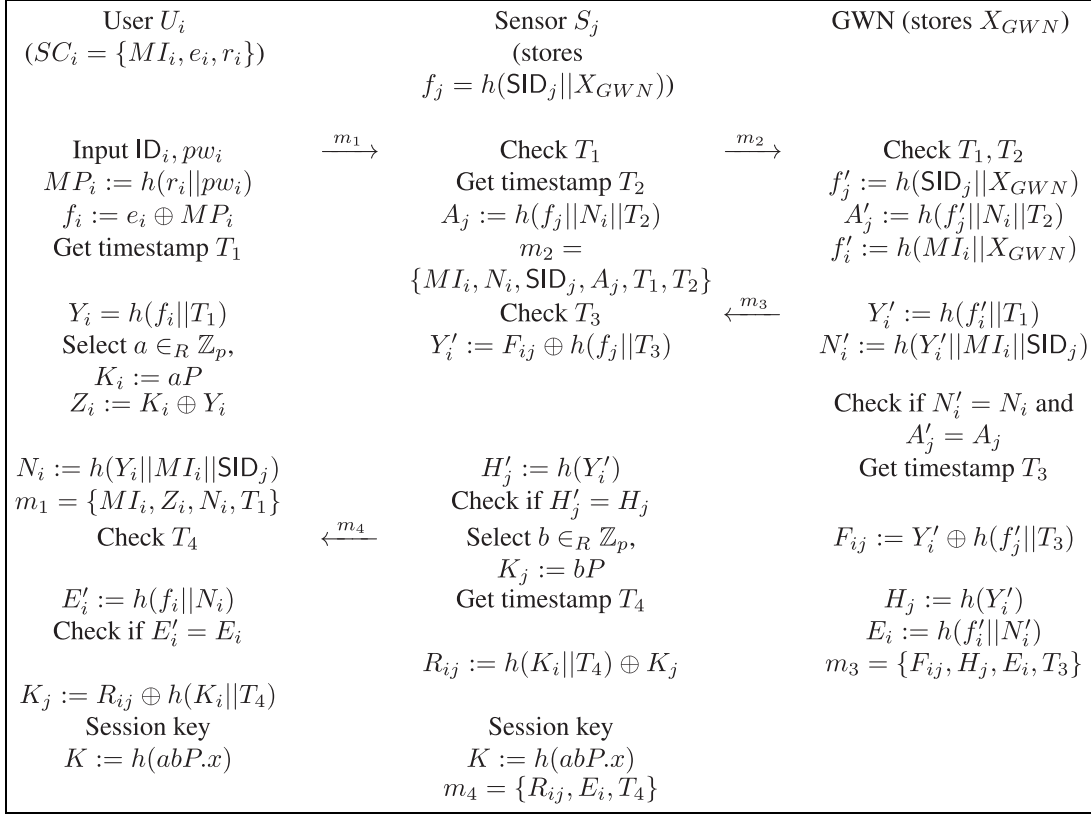


Figure 3. P2 of Chang and Le.⁹

- \mathcal{A} computes $Y_i := F_{ij} \oplus h(f_j || T_3)$. At this point, we stress that the \mathcal{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 Δ_T . This is very like to happen, for example, $\Delta_T = 1$ min.
- \mathcal{A} selects a random a^* and $K_i^* = a^*P$ and computes $Z_i^* := K_i^* \oplus Y_i$ and $N_i^* := h(Y_i || MI_i || \text{SID}_v)$. The message $m_1^* = \{MI_i, Z_i^*, N_i^*, T_1\}$ is sent to some oracle of sensor SID_j , say $\Pi_{SID_v}^l$.
- Meanwhile $f_j := h(\text{SID}_j || X_{GWN})$ would run the protocol with GWN honestly. $\Pi_{SID_v}^l$ replies \mathcal{A} with message $m_4^* = \{R_{iv}^*, E_i^*, T_4^*\}$ where $R_{iv}^* = h(K_i^* || T_4) \oplus K_v^*$ and $K_v^* = b^*P$ is chosen by $\Pi_{SID_v}^l$.
- Finally, \mathcal{A} could compute $K_v^* := h(K_i^* || T_4) \oplus R_{iv}^*$ 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 K_i^* is chosen by \mathcal{A} of its own choice.

Note that based on such corrupted sensor, \mathcal{A} is able to launch (unlimited) the above attack to impersonate any honest users U_i to any uncompromised sensors S_v at an arbitrary time T_1 chosen by \mathcal{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_2 and computation time of two hash operations (for computing A_j and Y_i). If the attacker fails within a time $T_1 + \Delta_T$, 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_j into the generation of Y_i , that is, $Y_i = h(f_i || MI_i || \text{SID}_j)$. 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).

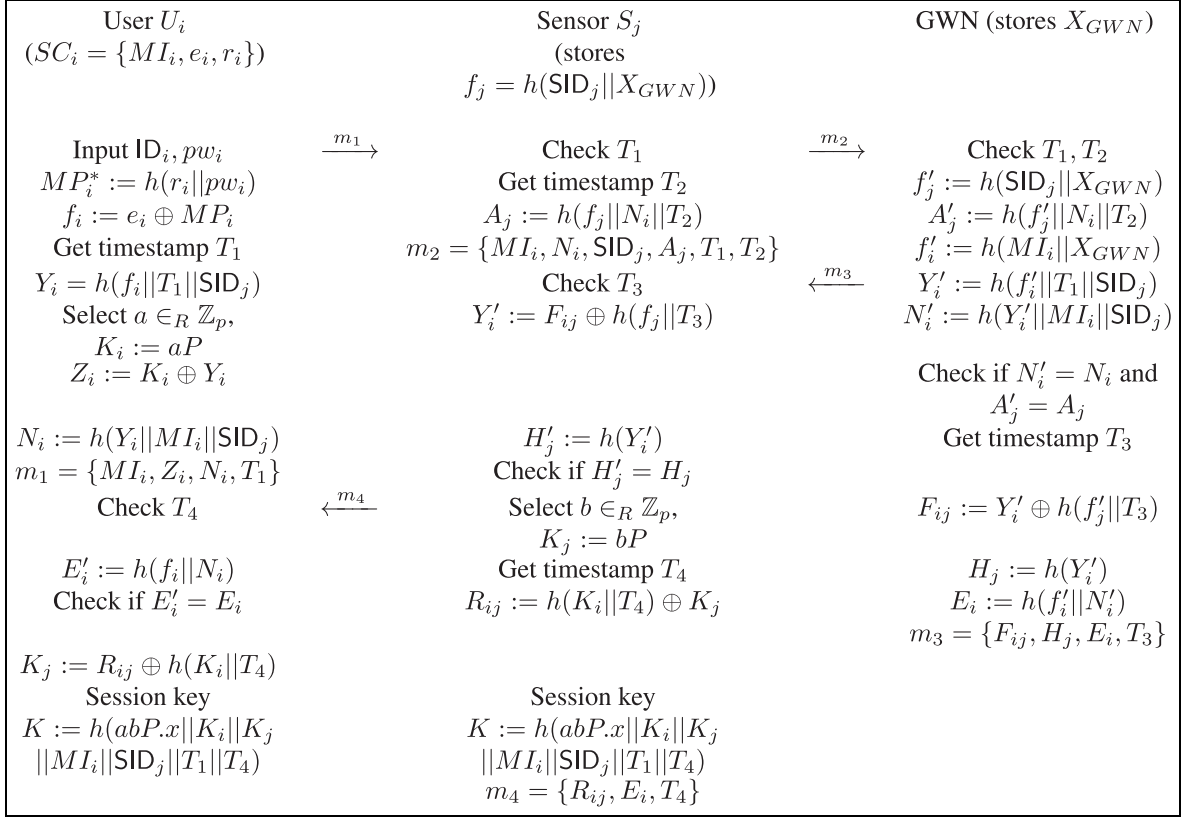


Figure 4. The improved P2.

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 $m_{U_{reg}} = \{ID_i, MP_i\}$ to the GWN over some secure channel.
 - Upon receiving $m_{U_{reg}}$, 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_{GWN} to compute $f_i := h(MI_i || X_{GWN})$ 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 $f_j := h(SID_j || X_{GWN})$. 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 selects a random value $a \in_R \mathbb{Z}_p$. Next, SC_i computes an ephemeral public key $K_i := aP$, 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.

- S_j gets a timestamp T_2 and computes an authentication message $A_j := h(f_j || N_i || T_2)$.
 - The message $m_2 = \{MI_i, N_i, SID_j, A_j, T_1, T_2\}$ 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 $f_j' = h(SID_j || X_{GWN})$ and $A_j' := h(f_j' || N_i || T_2)$, $f_i' := h(MI_i || X_{GWN})$, and $N_i' := h(Y_i' || MI_i || SID_j)$. GWN rejects if $N_i' \neq N_i$ and $A_j' \neq A_j$.
 - Next, GWN gets a timestamp T_3 and computes an authentication key $Y_i' := h(f_i' || T_1 || SID_j)$ and a ciphertext $F_{ij} := Y_i' \oplus h(f_j' || T_3)$, and authentication messages $H_j := h(Y_i')$ and $E_i := h(f_i' || N_i)$.
 - The message $m_3 = \{F_{ij}, H_i, E_i, T_3\}$ 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 $Y_i' := F_{ij} \oplus h(f_j' || T_3)$. S_j rejects if $H_j \neq h(Y_i')$.
 - S_j selects a random value $b \in_R \mathbb{Z}_p$ and computes an ephemeral public key $K_j := bP$. Next, S_j gets a timestamp T_4 and computes a ciphertext $R_{ij} := h(K_i || T_4) \oplus K_j$.
 - The session key is computed as $K := h(abP \cdot x || K_i || K_j || MI_i || SID_j || T_1 || T_4)$, where $abP \cdot x$ 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 $m_4 = \{R_{ij}, E_i, T_4\}$ 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 $E_i' := h(f_i || N_i)$ and rejects if $E_i' \neq E_i$.
 - U_i computes the session key $K := h(abP \cdot x || K_i || K_j || MI_i || SID_j || T_1 || T_4)$.
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 tuple (P, aP, bP, cP) for $a, b, c \xrightarrow{\$} \mathbb{Z}_p^*$ 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:

$$\text{EXP}_{\mathbb{G}, p, \mathcal{D}}^{\text{ddh}}(\lambda)$$

$$P \xleftarrow{\$} \mathbb{G}; (a, b, \gamma) \xleftarrow{\$} \mathbb{Z}_p^*; \beta \xleftarrow{\$} \{0, 1\};$$
 if $\beta = 1$ then $\Gamma \leftarrow abP$, otherwise $\Gamma \leftarrow \gamma P$;
 $\beta' \leftarrow \mathcal{D}(\mathbb{G}, p, P, aP, bP, \Gamma)$; if $\beta = \beta'$ then return 1, otherwise return 0.

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

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

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 λ .

Theorem 1. Suppose that the hash function h is modeled as random oracle, the output bit length of the hash function is ν , and the DDH problem relative to a group \mathbb{G} with prime order p holds, then the fixed protocol $P2'$ is session-key-secure such that $\text{Adv}_{P2', \mathcal{A}}^{\text{ake}}(\lambda, \text{OFFS}) \leq d/D + q_h/2^{\nu-3} + (d\ell)^2 \cdot \text{Adv}_{\mathbb{G}, p, \mathcal{D}}^{\text{ddh}}(\lambda)$, where q_h is the number of allowed random oracle query.

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

Proof. The proof is proceeded following the game-based approach.²³ Let Adv_{ξ} denote the advantage of \mathcal{A} in Game ξ .

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

$$\text{Adv}_{\text{II}, \mathcal{A}}^{\text{ake}}(\lambda) = \text{Adv}_0$$

Meanwhile, \mathcal{C} will simulate the queries of \mathcal{A} as follows:

- $\text{Execute}(ID_i, s, SID_j, t, ID_{GWN}, u)$ and $\text{Send}(\mathcal{ID}_i, s, m)$ queries will be honestly simulated as defined in section “Security model” following our protocol specification.
- $\text{RevealKey}(\mathcal{ID}_i, s)$: The session key returned by this query is $SK = h(abP.x || K_i || K_j || MI_i || SID_j || T_1 || T_4)$.
- $\text{CorruptSC}(ID_i)$: This query returns $SC_i = \{MI_i, e_i, r_i\}$ of ID_i .
- $\text{CorruptS}(SID_j)$: This query returns the contents of $f_j = h(SID_j || X_{GWN})$ of SID_j .
- $\text{Test}(\mathcal{ID}_i, s)$: This query will be honestly simulated as defined in section “Security model.” However, the test oracle (asked by this query) should keep fresh throughout the security experiment in the sense of Definition 1.

In this game, the challenger just simulates these queries following the protocol specification without any modification. In the subsequent games, we may change them step by step till the advantage of the adversary is zero.

Game 1. In this game, the challenger aborts if the adversary \mathcal{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 $\text{Send}(ID_i, s, m)$ 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 . Hence, 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 \mathcal{A} correctly guesses is about $1/D$. Note that \mathcal{A} can try $d\ell$ times which is the maximum number of Send query to all ℓ 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 \mathcal{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 \mathcal{A} may be able to ask the random oracle $h(Y_i || MI_i || SID_j)$

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 \mathcal{A} is bound to $1/2^{\nu-1}$. 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 S_j . The security of A_j is quite similar to that of N_i .

Since there are two such authentication messages, N_i and A_j , each of which has two related hash operations, \mathcal{A} can ask at most q_h random oracle queries. We therefore have that

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

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

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

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 build an algorithm \mathcal{D} to solve the DDH problem. Given a DDH challenge instance (xP, yP, cP) , the job of \mathcal{D} is to distinguish $c = ?xy$. Meanwhile, \mathcal{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 \mathcal{D} ’s guess is correct. Then \mathcal{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

$$Adv_3 \leq Adv_4 + Adv_{\mathbb{G}, p, \mathcal{D}}^{ddh}(\lambda)$$

Table 2. Computation cost.

	User	Sensor	Gateway
SG ⁴	3mul + 6h 11.442 ms	2mul + 3h 7.621 ms	1mul + 5h 3.835 ms
CLK+ ⁶	3mul + 7h 11.449 ms	2mul + 6h 7.642 ms	1mul + 5h 3.835 ms
TBH ⁷	6h 0.042 ms	7h 0.028 ms	7h 0.049 ms
DKO+ ¹¹	2mul + 13h 7.691 ms	2mul + 9h 7.663 ms	10h 0.07 ms
Ours	2mul + 6h 7.642 ms	2mul + 5h 7.635 ms	8h 0.056 ms

As the bit in the Test query is not used anymore, the advantage of an adversary in this game is zero, that is, $Adv_4 = 0$. To sum up all probabilities in the above games, we get the result of this theorem.

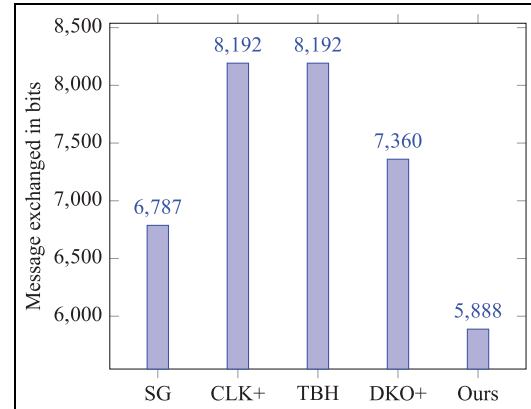
Performance comparison

In this section, we compare our improved scheme with related existing similar lightweight AKE protocols of Shi and Gong⁴ (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 t_{mul} and t_h 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 $t_{mul} = 3.8$ ms and $t_h = 0.007$ 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.'s⁷ 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.

**Figure 5.** Communication cost.

Conclusion

In this article, we revisited a recently proposed AKE scheme⁹ 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.'s¹¹ 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

province for enterprise informatization and IoT (grant no. 2015WZJ02), Research Foundation of the Natural Foundation of Chongqing City (grant nos cstc2013jcyjA0076, cstc2016jcyjA40019, and cstc2017jcyjAX0277), China Postdoctoral Science Foundation (grant no. 2017M612911), and Scientific and Technological Research Program of Chongqing Municipal Education Commission (grant nos KJ1600932 and KJ1600928).

ORCID iD

Zheng Yang:  <https://orcid.org/0000-0001-8610-9936>

References

1. He D, Gao Y, Chan S, et al. An enhanced two-factor user authentication scheme in wireless sensor networks. *Ad Hoc Sens Wirel Netw* 2010; 10(4): 361–371.
2. Das AK, Sharma P, Chatterjee S, et al. A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *J Netw Comput Appl* 2012; 35(5): 1646–1656.
3. Xue K, Ma C, Hong P, et al. A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. *J Netw Comput Appl* 2013; 36(1): 316–323.
4. Shi W and Gong P. A new user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Int J Distrib Sens N* 2013; 2013: 730831.
5. Turkanovic M and Holbl M. An improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Elektron Elektrotech* 2013; 19(6): 109–116.
6. Choi Y, Lee D, Kim J, et al. Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* 2014; 14(6): 10081–10106.
7. Turkanovic M, Brumen B and Hlbl M. 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* 2014; 20(2): 96–112.
8. Chang CC, Hsueh WY and Cheng TF. A dynamic user authentication and key agreement scheme for heterogeneous wireless sensor networks. *Wireless Pers Commun* 2016; 89(2): 447–465.
9. Chang CC and Le HD. A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. *IEEE T Wirel Commun* 2016; 15(1): 357–366.
10. Jung J, Kim J, Choi Y, et al. An anonymous user authentication and key agreement scheme based on a symmetric cryptosystem in wireless sensor networks. *Sensors* 2016; 16(8): E1299.
11. Das AK, Kumari S, Odelu V, et al. Provably secure user authentication and key agreement scheme for wireless sensor networks. *Secur Commun Netw* 2016; 9: 3670–3687.
12. Challa S, Wazid M, Das AK, et al. Secure signature-based authenticated key establishment scheme for future IoT applications. *IEEE Access* 2017; 5: 3028–3043.
13. Jiang Q, Zeadally S, Ma J, et al. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access* 2017; 5: 3376–3392.
14. Park K, Park Y, Park, et al. Provably secure and efficient authentication protocol for roaming service in global mobility networks. *IEEE Access* 2017; 5: 25110–25125.
15. Ali R, Pal AK, Kumari S, et al. A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Gener Comp Sy*. Epub ahead of print 15 July 2017. DOI: 10.1016/j.future.2017.06.018.
16. Shim K-A. BASIS: a practical multi-user broadcast authentication scheme in wireless sensor networks. *IEEE T Inf Foren Sec* 2017; 12(7): 1545–1554.
17. Wazid M, Das AK, Kumar N, et al. Design of lightweight authentication and key agreement protocol for vehicular ad hoc networks. *IEEE Access* 2017; 5: 14966–14980.
18. Ferrag MA, Maglaras LA, Janicke H, et al. Authentication protocols for Internet of Things: a comprehensive survey. *Secur Commun Netw* 2017; 2017: 6562953.
19. Bellare M, Pointcheval D and Rogaway P. Authenticated key exchange secure against dictionary attacks. In: *Advances in cryptology—EUROCRYPT*, Bruges, 14–18 May 2000, pp.139–155. Heidelberg: Springer.
20. Jager T, Kohlar F, Schäge S, et al. On the security of TLS-DHE in the standard model. In: *Advances in cryptology—CRYPTO*, Santa Barbara, CA, 19–23 August 2012, pp.273–293. Heidelberg: Springer.
21. Yang Z, Yang W, Zhu L, et al. Towards modelling perfect forward secrecy in two-message authenticated key exchange under ephemeral-key revelation. *Secur Commun Netw* 2015; 8(18): 3356–3371.
22. Bellare M and Rogaway P. Provably secure session key distribution: the three party case. In: *27th annual ACM symposium on theory of computing*, Las Vegas, NV, 29 May–1 June 1995, pp.57–66. New York: ACM Press.
23. Shoup V. Sequences of games: a tool for taming complexity in security proofs. *Cryptology ePrint Archive*, Report 2004/332, 2004, <http://eprint.iacr.org/>
24. Yaqoob I, Ahmed E, ur Rehman MH, et al. The rise of ransomware and emerging security challenges in the Internet of Things. *Computer Networks* 2017; 129: 444–458.
25. Chen Y, Martínez JF, Castillejo P, et al. A privacy protection user authentication and key agreement scheme tailored for the Internet of Things environment: PriAuth. *Wirel Commun Mob Com* 2017; 2017: 5290579.