Sensors **2014**, *14*, 21023-21044; doi:10.3390/s141121023

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

A Provably-Secure ECC-Based Authentication Scheme for Wireless Sensor Networks

Junghyun Nam [1], Moonseong Kim [2], Juryon Paik [3], Youngsook Lee [4] and Dongho Won [3,*]

[1] Department of Computer Engineering, Konkuk University, 268 Chungwondaero, Chungju, Chungcheongbukdo 380-701, Korea; E-Mail: jhnam@kku.ac.kr
[2] Information Management Division, Korean Intellectual Property Office, 189 Cheongsaro, Daejeon 302-701, Korea; E-Mail: moonseong@kipo.go.kr
[3] Department of Computer Engineering, Sungkyunkwan University, 2066 Seoburo, Suwon, Gyeonggido 440-746, Korea; E-Mail: wise96@skku.edu
[4] Department of Cyber Investigation Police, Howon University, 64 3-gil, Gunsan, Jeonrabukdo 573-718, Korea; E-Mail: ysooklee@howon.ac.kr

* Author to whom correspondence should be addressed; E-Mail: dhwon@security.re.kr; Tel.: +82-31-290-7213; Fax: +82-31-290-5695.

External Editor: Leonhard M. Reindl

Received: 1 September 2014; in revised form: 30 October 2014 / Accepted: 31 October 2014 / Published: 6 November 2014

Abstract: A smart-card-based user authentication scheme for wireless sensor networks (in short, a SUA-WSN scheme) is designed to restrict access to the sensor data only to users who are in possession of both a smart card and the corresponding password. While a significant number of SUA-WSN schemes have been suggested in recent years, their intended security properties lack formal definitions and proofs in a widely-accepted model. One consequence is that SUA-WSN schemes insecure against various attacks have proliferated. In this paper, we devise a security model for the analysis of SUA-WSN schemes by extending the widely-accepted model of Bellare, Pointcheval and Rogaway (2000). Our model provides formal definitions of authenticated key exchange and user anonymity while capturing side-channel attacks, as well as other common attacks. We also propose a new SUA-WSN scheme based on elliptic curve cryptography (ECC), and prove its security properties in our extended model. To the best of our knowledge, our proposed scheme is the first SUA-WSN scheme that provably achieves both authenticated key exchange and user anonymity. Our scheme is also computationally competitive with other ECC-based (non-provably secure) schemes.

Keywords: wireless sensor network; authentication scheme; authenticated key exchange; user anonymity; smart card; two-factor security

1. Introduction

As various sensors emerge and the related technologies advance, there has been a dramatic increase in the interest in wireless sensor networks (WSNs). Today, billions of physical, chemical and biological sensors are being deployed into various types of WSNs for numerous applications, including military surveillance, wildlife monitoring, vehicular tracking and healthcare diagnostics [1]. A major benefit of WSN systems is that they provide unprecedented abilities to explore and understand large-scale, real-world data and phenomena at a fine-grained level of temporal and spatial resolution. However, providing an application service in a WSN environment introduces significant security challenges to be addressed among the involved parties: users, sensors and gateways. One important challenge is to achieve authenticated key exchange between users and sensors (via the assistance of a gateway), thereby preventing illegal access to the sensor data and their transmissions. Authenticated key exchange in WSNs is more challenging to achieve than in traditional networks due to the sensor network characteristics, such as resource constraints, unreliable communication channel and unattended operation. Another important challenge is to provide user anonymity, which makes authenticated key exchange even harder. As privacy concern increases, user anonymity has become a major security property in WSN applications, as well as in many other applications, like mobile roaming services, anonymous web browsing, location-based services and e-voting. User authentication schemes for WSNs are designed to address these security challenges [2,3], and are a subject of active research in network security and cryptography.

Generally speaking, the design of cryptographic schemes (including user authentication schemes for WSNs) is error-prone, and their security analysis is time-consuming. The difficulty of getting a high level of assurance in the security of cryptographic schemes is well illustrated with examples of flaws discovered in many such schemes years after they were published; see, e.g., [4–6]. The many flaws identified in published schemes over the decades have promoted formal security analyses, which are broadly classified into two approaches [7,8]: the computer security approach and the computational complexity approach. The computer security approach places its emphasis on automated machine specification and analysis mostly in the Dolev–Yao adversarial model [9], where the underlying cryptographic primitives are often used in a black-box manner ignoring some of cryptographic details. The main problem with this automated approach is intractability and undecidability, as the adversary may exhibit a large set of possible behaviors, which leads to a state explosion. Cryptographic schemes proven secure in such a fashion could possibly be flawed, yielding a false positive result. In contrast, the computational complexity approach places its emphasis on deriving a polynomial-time reduction from the problem of breaking the scheme into another problem believed to be hard. A complete computational proof under a well-established cryptographic assumption provides a strong assurance that the security properties of the scheme are satisfied. Accordingly, it has been standard practice for the designers of cryptographic schemes to provide a proven reduction for the security of their schemes in a widely-accepted model [10,11]. Although these human-generated mathematical proofs are usually lengthy and complicated, they are certainly an invaluable tool for getting secure cryptographic schemes.

In 2009, Das [12] proposed a smart-card-based user authentication scheme for wireless sensor networks; throughout the paper, we call such a scheme a SUA-WSN scheme. Since then, the design of SUA-WSN schemes has received significant attention from researchers due to their potential to be widely deployed, and a number of solutions offering various levels of efficiency and security have been subsequently proposed [2,3,13–27]. Early schemes only aimed to achieve mutual authentication [13–15], while later schemes attempted to provide additional security properties, such as authenticated key exchange [2,3,16–27] and user anonymity [2,3,20,22–24,26]. Some schemes [16,21,27] employ elliptic curve cryptography to provide perfect forward secrecy, while others [2,3,12–14,17–20,22–26] only use symmetric cryptography and hash functions to focus on improving the efficiency.

One important security requirement for SUA-WSN schemes is to ensure that only a user who is in possession of both a smart card and the corresponding password can pass the authentication check of the gateway and gain access to the sensor network and data. A SUA-WSN scheme that meets this requirement is said to achieve two-factor security. To properly capture the notion of two-factor security, the adversary against SUA-WSN schemes is assumed to be able to either extract the sensitive information in the smart card of a user possibly via a side-channel attack [28,29] or learn the password of the user through shoulder-surfing or by exploiting a malicious card reader, but not both. Clearly, there is no means to prevent the adversary from impersonating a user if both the password of the user and the information in the smart card are disclosed.

Despite the research efforts over the recent years, it remains a significant challenge to design a robust SUA-WSN scheme that carries a formal proof of security in a widely-accepted model. As summarized in Table 1, most of the published schemes either provide no formal analysis of security [3,12–14,16,20–22,24–26] or fail to achieve important security properties, such as mutual authentication, session-key security, user anonymity, two-factor security and resistance against various attacks [3,13–16,19,21–27,30,31]. Some schemes [2,17–19,23,27] have been proven secure using a computer security approach, which, as mentioned above, suffers from intractability and undecidability and could possibly give a false positive result. To the best of our knowledge, Chen and Shih's scheme [15] is the only SUA-WSN scheme that was proven secure using a computational complexity approach. However, Chen and Shih's scheme does not provide key exchange functionality, but only focuses on mutual authentication (and thus, inherently, cannot carry a proof of authenticated key exchange). Moreover, the security model used for this scheme captures neither the user anonymity property nor the notion of two-factor security.

Table 1. A summary of security results for existing SUA-WSN (smart-card-based user authentication scheme for wireless sensor networks) schemes.

Scheme	**Security Justification**	**Major Weaknesses**
Das [12]	Heuristic arguments	No key-exchange functionality
He *et al.* (2010) [13]	Heuristic arguments	No key-exchange functionality
Khan and Alghathbar [14]	Heuristic arguments	No key-exchange functionality
Chen and Shih [15]	Computational complexity approach (only for entity authentication)	No key-exchange functionality
Yeh *et al.* [16]	Heuristic arguments	Failures of mutual authentication and forward secrecy [30]
Kumar *et al.* (2011) [2]	Computer security approach	Vulnerability to a node capture attack [24]
Kumar *et al.* (2012) [17]	Computer security approach	Failures of authenticated key exchange, user anonymity and two-factor security [3,23]
Yoo *et al.* [18]	Computer security approach	Vulnerability to a man-in-the-middle attack [22]
Vaidya *et al.* [19]	Computer security approach	Failure of user authentication [25]
Xue *et al.* [20]	Heuristic arguments	Vulnerability to a privileged insider attack [26]
Shi and Gong [21]	Heuristic arguments	Failures of authenticated key exchange and two-factor security [27]
Kumar *et al.* (2013) [22]	Heuristic arguments	
He *et al.* (2013) [23]	Computer security approach	
Chi *et al.* [24]	Heuristic arguments	
Kim *et al.* [25]	Heuristic arguments	
Khan and Kumari [3]	Heuristic arguments	
Jiang *et al.* [26]	Heuristic arguments	
Choi *et al.* [27]	Computer security approach	No provision of user anonymity

The contributions of this paper are two-fold:

(1) We present a security model for the analysis of SUA-WSN schemes. Our security model is derived by extending the widely-accepted model of Bellare, Pointcheval and Rogaway [10] to incorporate into it the user anonymity property and the notion of two-factor security. Notice that the original Bellare–Pointcheval–Rogaway (BPR) model for authenticated key exchange (AKE) already captures insider attacks, offline dictionary attacks and other common attacks. We refer readers to [32] to understand how a key exchange scheme that is vulnerable to an offline dictionary attack can be rendered insecure in the BPR model. Our extension of the BPR model provides two security definitions, one for the AKE security and one for the user anonymity property, and both definitions capture the notion of two-factor security. Security properties like authentication,

session-key security, perfect forward secrecy, known-key security and resistance against insider attacks and offline dictionary attacks are implied by the AKE security.

(2) We propose the first SUA-WSN scheme whose AKE security, as well as user anonymity are formally proven in a widely-accepted model. Our scheme employs elliptic curve cryptography (ECC) to provide perfect forward secrecy, but differs from other ECC-based schemes [16,21,27] in that it provides user anonymity. We prove the security properties of our scheme in the random oracle model under the elliptic curve computational Diffie–Hellman (ECCDH) assumption. We also show that our provably-secure scheme is computationally competitive compared with other ECC-based (non-provably secure) schemes.

Table 2 shows the basic notation that is used consistently throughout this paper.

Table 2. Basic notation.

Symbol	**Description**
UR	User
SR	Sensor
GW	Gateway
$ID_{UR}, ID_{SR}, ID_{GW}$	Identities of UR, SR and GW
pw_{UR}	Password of U
sk	Session key
$\mathcal{A}$	Probabilistic polynomial-time adversary
$L(\cdot), H(\cdot), F(\cdot)$	Cryptographic hash functions
$\mathsf{Enc}_k(\cdot)/\mathsf{Dec}_k(\cdot)$	Symmetric encryption/decryption under key k
MAC	Message authentication code
$\mathsf{Mac}_k(\cdot)/\mathsf{Ver}_k(\cdot)$	MAC generation/verification under key k
$\oplus$	Bitwise exclusive-or (XOR) operation
$\|$	String concatenation operation
$\{0,1\}^n$	Bit strings of length n

The remainder of this paper is structured as follows. Section 2 describes our extended security model for the analysis of SUA-WSN schemes. Section 3 presents the proposed SUA-WSN scheme along with cryptographic primitives on which the security of the scheme relies and then compares our scheme with other ECC-based schemes, both in terms of efficiency and security. Section 4 provides proofs of the user anonymity property and the AKE security for our scheme. Section 5 concludes the paper, summarizing our result and presenting some interesting future work.

2. Our Extended Security Model for SUA-WSN Schemes

In this section, we present a security model extended from the BPR model [10] to capture the security properties of SUA-WSN schemes.

Participants and long-lived keys. Let GW be the gateway and $\mathcal{SRS}$ and $\mathcal{URS}$ be the sets of all sensors and users, respectively, registered with GW. Let $\mathcal{E} = \{GW\} \cup \mathcal{SRS} \cup \mathcal{URS}$. We identify each entity

$E \in \mathcal{E}$ by a string, and interchangeably use E and ID_E to refer to this identifier string. To properly capture the user anonymity property, we assume that: (1) each user $UR \in \mathcal{URS}$ has its pseudo identity PID_{UR} (as well as its true identity ID_{UR}); and (2) the adversary $\mathcal{A}$ is given only PID_{UR}, but not ID_{UR}. A user UR may run multiple sessions of the authentication and key exchange protocol of the scheme (hereafter simply called the protocol), either serially or concurrently, to anonymously establish a session key with a sensor $SR \in \mathcal{SRS}$ via the assistance of the gateway GW. Therefore, at any given time, there could be multiple instances of the entities UR, SR and GW. We use Π_E^i to denote instance i of entity $E \in \mathcal{E}$. Instances of UR and SR are said to accept when they compute a session key in an execution of the protocol. We denote the session key of Π_E^i by sk_E^i. Before the protocol is ever executed,

- GW generates its master secret(s), issues a smart card to each $UR \in \mathcal{URS}$ and establishes a shared key with each $SR \in \mathcal{SRS}$; and
- each $UR \in \mathcal{URS}$ chooses its private password pw_{UR} from the set of all possible passwords.

Partnering. Informally, we say that two instances are partners (or partnered) if they participate together in a protocol session and establish a shared key. Formally, the partner relationship between instances is defined in terms of the notion of the session identifier. A session identifier (sid) is literally an identifier of a protocol session and is typically defined as a function of the messages exchanged in the session. Let sid_E^i denote the sid of instance Π_E^i. We say that two instances, Π_{UR}^i and Π_{SR}^j, are partners if: (1) both instances have accepted; and (2) $sid_{UR}^i = sid_{SR}^j$.

Adversary capabilities. The adversary $\mathcal{A}$ is a probabilistic polynomial-time (PPT) machine, which has full control of all communications between entities. More specifically, the PPT adversary $\mathcal{A}$ is able to: (1) eavesdrop, modify, intercept, delay and delete the protocol messages; (2) ask entities to open up access to session keys and long-term keys; and (3) extract the sensitive information on the smart cards of users. These capabilities of $\mathcal{A}$ are modeled using a pre-defined set of oracles to which $\mathcal{A}$ is allowed to ask queries. We assume that, when making oracle queries directed at (instances of) UR, the adversary $\mathcal{A}$ uses the pseudo identity PID_{UR}, since it does not know the true identity ID_{UR}. The oracle queries are described as follows:

- Execute(Π_{UR}^i, Π_{SR}^j, Π_{GW}^k): This query models passive eavesdropping on the protocol messages. It prompts a protocol execution among the instances Π_{UR}^i, Π_{SR}^j and Π_{GW}^k and returns the transcript of the protocol execution to $\mathcal{A}$.
- Send(Π_E^i, m): This query sends a message m to an instance Π_E^i, modeling active attacks against the protocol. Upon receiving m, the instance Π_E^i proceeds according to the protocol specification. Any message generated by Π_E^i is output and given to $\mathcal{A}$. A query of the form Send(Π_{UR}^i, `start`) prompts Π_{UR}^i to initiate a protocol session.
- Reveal(Π_E^i): This query captures known key attacks. Upon receiving this query, the instance Π_E^i returns its session key sk_E^i back to $\mathcal{A}$ (if it has accepted).
- CorruptLL(E): This query returns the long-lived secret(s) of entity E, capturing the notion of forward secrecy, as well as resistance to unknown key share attacks and insider attacks.
- CorruptSC(UR): This query captures side-channel attacks (*i.e.*, the notion of two-factor security) and returns the information stored in the smart card of UR.

- TestAKE(Π_E^i): This query is used for defining the indistinguishability-based security of session keys. The output of the query depends on a random bit b chosen by the oracle; in response to the query, either the real session key sk_E^i if $b = 1$ or a random key drawn from the session-key space if $b = 0$ is returned to $\mathcal{A}$.
- TestID(UR): This query is used for determining whether the protocol provides user anonymity or not. Depending on a random bit b chosen by the oracle, $\mathcal{A}$ is given either the identity actually used for UR in the protocol sessions (when $b = 1$) or a random identity drawn from the identity space (when $b = 0$).

SR and GW are said to be corrupted when they are asked a CorruptLL query, while UR is considered as corrupted if it has been asked both CorruptLL and CorruptSC queries.

Authenticated key exchange (AKE). We define the AKE security of the authentication and key exchange protocol P by using the notion of freshness of instances. Informally, a fresh instance refers to an instance whose session key should be kept indistinguishable from a random key to the adversary $\mathcal{A}$, and an unfresh instance refers to an instance that holds a session key that can be distinguishable from a random key by trivial means. A formal definition of freshness follows:

Definition 1 (Freshness)**.** *An instance Π_E^i is fresh unless one of the following occurs:*

1. *$\mathcal{A}$ queries* Reveal(Π_E^i) *or* Reveal($\Pi_{E'}^j$)*, where $\Pi_{E'}^j$ is the partner of Π_E^i;*
2. *$\mathcal{A}$ queries* CorruptLL(SR) *or* CorruptLL(GW) *before Π_E^i accepts.*
3. *$\mathcal{A}$ queries both* CorruptLL(UR) *and* CorruptSC(UR)*, for some $UR \in \mathcal{URS}$, before Π_E^i accepts.*

The AKE security of the protocol P is defined in the context of the following two-phase experiment:

Experiment **ExpAKE**$_0$:

Phase 1. $\mathcal{A}$ freely asks any oracle queries, except that:

1. $\mathcal{A}$ is not allowed to ask queries of the TestID oracle.
2. $\mathcal{A}$ is not allowed to ask the TestAKE(Π_E^i) query if the instance Π_E^i is not fresh.
3. $\mathcal{A}$ is not allowed to ask the Reveal(Π_E^i) query if it has already asked a TestAKE query of Π_E^i or its partner instance.

Phase 2. When Phase 1 is over, $\mathcal{A}$ outputs a bit b' as a guess of the random bit b selected by the TestAKE oracle. $\mathcal{A}$ succeeds if $b = b'$.

Let SuccAKE$_0$ be the event that $\mathcal{A}$ succeeds in the experiment **ExpAKE**$_0$. Let $\mathsf{Adv}_P^{\mathrm{AKE}}(\mathcal{A})$ denote the advantage of $\mathcal{A}$ in breaking the AKE security of protocol P and be defined as $\mathsf{Adv}_P^{\mathrm{AKE}}(\mathcal{A}) = 2 \cdot \mathsf{Pr}_{P,\mathcal{A}}[\mathsf{SuccAKE}_0] - 1$.

Definition 2 (AKE security)**.** *The authentication and key exchange protocol P is AKE-secure if* $\mathsf{Adv}_P^{\mathrm{AKE}}(\mathcal{A})$ *is negligible for any* PPT *adversary $\mathcal{A}$.*

User anonymity. The AKE security does not imply user anonymity. In other words, an authentication and key exchange protocol that does not provide user anonymity may still be rendered AKE secure.

Hence, a new, separate definition is necessary to capture the user anonymity property. Our definition of user anonymity is based on the notion of cleanness.

Definition 3 (Cleanness)**.** *A user $UR \in \mathcal{URS}$ is clean unless one of the following occurs:*

1. *$\mathcal{A}$ queries* $\mathsf{CorruptLL}(GW)$.
2. *$\mathcal{A}$ queries both* $\mathsf{CorruptLL}(UR)$ *and* $\mathsf{CorruptSC}(UR)$.

Note that this definition of cleanness does not impose any restriction on asking a $\mathsf{CorruptLL}$ query to SR. This reflects our objective to achieve user anonymity even against the sensor SR.

Now, consider the following experiment to formalize the user anonymity property:

Experiment $\mathbf{ExpID}_0$:

Phase 1. $\mathcal{A}$ freely asks any oracle queries, except that:

1. $\mathcal{A}$ is not allowed to ask queries of the $\mathsf{TestAKE}$ oracle.
2. $\mathcal{A}$ is not allowed to ask the $\mathsf{TestID}(UR)$ query if the user UR is not clean.
3. $\mathcal{A}$ is not allowed to ask $\mathsf{CorruptLL}$ and $\mathsf{CorruptSC}$ queries against GW and UR if it has already asked the $\mathsf{TestID}(UR)$ query.

Phase 2. When Phase 1 is over, $\mathcal{A}$ outputs a bit b' as a guess on the random bit b selected by the TestID oracle. $\mathcal{A}$ succeeds if $b = b'$.

Let SuccID_0 be the event that $\mathcal{A}$ succeeds in the experiment $\mathbf{ExpID}_0$. Then, we define the advantage of $\mathcal{A}$ in attacking the user anonymity of protocol P as $\mathsf{Adv}_P^{\mathrm{ID}}(\mathcal{A}) = 2 \cdot \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_0] - 1$.

Definition 4 (User anonymity)**.** *The authentication and key exchange protocol P provides user anonymity if* $\mathsf{Adv}_P^{\mathrm{ID}}(\mathcal{A})$ *is negligible for any* PPT *adversary $\mathcal{A}$.*

3. The Proposed SUA-WSN Scheme

This section presents our ECC-based user authentication scheme for wireless sensor networks. Our scheme consists of three phases: the registration phase, the authentication, the key exchange phase and the password update phase. We begin by describing the cryptographic primitives on which the security of our scheme relies.

3.1. Preliminaries

Elliptic curve computational Diffie–Hellman (ECCDH) problem. Let $\mathbb{G}$ be an elliptic curve group of prime order q. Typically, $\mathbb{G}$ will be a subgroup of the group of points on an elliptic curve over a finite field. Let P be a generator of $\mathbb{G}$. Informally stated, the ECCDH problem for $\mathbb{G}$ is to compute $xyP \in \mathbb{G}$ when given two elements $(xP, yP) \in \mathbb{G}^2$, where x and y are chosen at random from $\mathbb{Z}_q^*$. We say that the ECCDH assumption holds in $\mathbb{G}$ if it is computationally intractable to solve the ECCDH problem for $\mathbb{G}$. More formally, we define the advantage of an algorithm $\mathcal{A}$ in solving the ECCDH problem for $\mathbb{G}$ as $\mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(\mathcal{A}) = \Pr[\mathcal{A}(\mathbb{G}, P, xP, yP) = xyP]$. We say that the ECCDH assumption holds in $\mathbb{G}$

if $\mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(\mathcal{A})$ is negligible for all PPT algorithms $\mathcal{A}$. We use $\mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(t)$ to denote the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(\mathcal{A})$ over all algorithms $\mathcal{A}$ running in time at most t.

Symmetric encryption schemes. A symmetric encryption scheme Γ is a pair of efficient algorithms (Enc, Dec) where: (1) the encryption algorithm Enc takes as input an ℓ-bit key k and a plain text message m and outputs a ciphertext c; and (2) the decryption algorithm Dec takes as input a key k and a ciphertext c and outputs a message m. We require that $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$ holds for all $k \in \{0,1\}^\ell$ and all $m \in \mathcal{M}$, where $\mathcal{M}$ is the plain text space. For an eavesdropping adversary $\mathcal{A}$ against Γ and for an integer $n \geq 1$ and a random bit $b \in_R \{0,1\}$, consider the following indistinguishability experiment:

```
Experiment $\mathbf{Exp}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A}, n, b)$
    for $i = 1$ to $n$
        $k_i \in_R \{0,1\}^\ell$
        $(m_{0,i}, m_{1,i}) \leftarrow \mathcal{A}(\Gamma)$
        $c_i \leftarrow \mathsf{Enc}_{k_i}(m_{b,i})$
        $\mathcal{A}(c_i)$
    $b' \leftarrow \mathcal{A}$, where $b' \in \{0,1\}$
    return $b'$
```

Let $\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A})$ be the advantage of an eavesdropper $\mathcal{A}$ in violating the indistinguishability of Γ, and let it be defined as:

$$\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A}) = |\Pr[\mathbf{Exp}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A}, n, 0) = 1] - \Pr[\mathbf{Exp}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A}, n, 1) = 1]|.$$

We say that Γ is secure if $\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A})$ is negligible in ℓ for any PPT adversary $\mathcal{A}$. We define $\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(t)$ as $\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(t) = \max_{\mathcal{A}} \{\mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(\mathcal{A})\}$, where the maximum is over all PPT adversaries $\mathcal{A}$ running in time at most t.

Message authentication codes. A message authentication code (MAC) scheme Δ is a pair of efficient algorithms (Mac, Ver) where: (1) the MAC generation algorithm Mac takes as input an ℓ-bit key k and a message m and outputs a MAC δ; and (2) the MAC verification algorithm Ver takes as input a key k, a message m and a MAC δ and outputs one if δ is valid for m under k or outputs zero if δ is invalid. Let $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A})$ be the advantage of an adversary $\mathcal{A}$ in violating the strong existential unforgeability of Δ under an adaptive chosen message attack. More precisely, $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A})$ is the probability that an adversary $\mathcal{A}$, who mounts an adaptive chosen message attack against Δ with oracle access to $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Ver}_k(\cdot)$, outputs a message/MAC pair (m, δ), such that: (1) $\mathsf{Ver}_k(m, \delta) = 1$; and (2) δ was not previously output by the oracle $\mathsf{Mac}_k(\cdot)$ as a MAC on the message m. We say that the MAC scheme Δ is secure if $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A})$ is negligible for every PPT adversary $\mathcal{A}$. Let $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(t)$ denote the maximum value of $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A})$ over all adversaries $\mathcal{A}$ running in time at most t.

Cryptographic hash functions. Our scheme uses three cryptographic hash functions $L : \{0,1\}^* \rightarrow \{0,1\}^\ell$, $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ and $F : \{0,1\}^* \rightarrow \{0,1\}^\varepsilon$, where ℓ is as defined for Δ and Γ, κ is the bit-length of session keys and ε is the bit-length of SID_{UR} (see Section 3.2.1 for the definition of SID_{UR}). These hash functions are modeled as random oracles in our security proofs.

3.2. Description of the Scheme

The public system parameters for our scheme include:

1. an elliptic curve group $\mathbb{G}$ with a generator P of prime order q,
2. a symmetric encryption scheme $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$,
3. a MAC scheme $\Delta = (\mathsf{Mac}, \mathsf{Ver})$, and
4. three hash functions L, H and F.

We assume that these public system parameters are fixed during an initialization phase and are known to all parties in the network. As part of the initialization, the gateway GW chooses two master keys $x, y \in \mathbb{Z}_q^*$, computes its public key $X = xP$ and establishes a shared secret key $k_{GS} = L(ID_{SR}\|y)$ with each sensor SR.

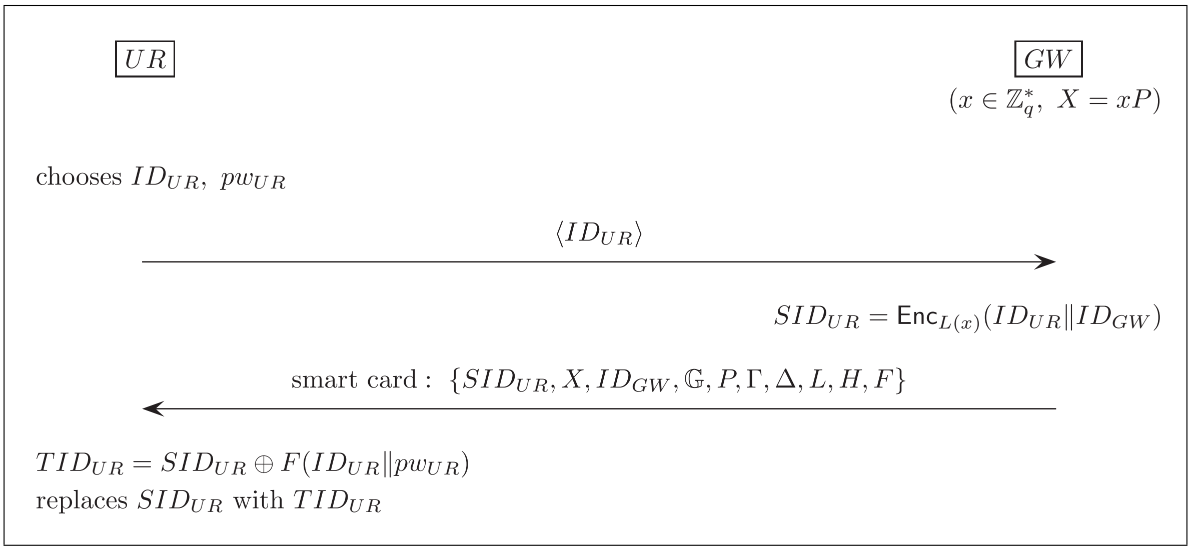
3.2.1. Registration Phase

A user UR registers itself with the gateway GW as follows:

1. UR chooses its identity ID_{UR} and password pw_{UR} freely and submits the identity ID_{UR} to GW via a secure channel.
2. GW computes $SID_{UR} = \mathsf{Enc}_{L(x)}(ID_{UR}\|ID_{GW})$ and issues UR a smart card loaded with $\{SID_{UR}, X, ID_{GW}, \mathbb{G}, P, \Gamma, \Delta, L, H, F\}$. (We assume that q is implicit in $\mathbb{G}$.)
3. UR replaces SID_{UR} with $TID_{UR} = SID_{UR} \oplus F(ID_{UR}\|pw_{UR})$.

This phase of user registration is depicted in Figure 1.

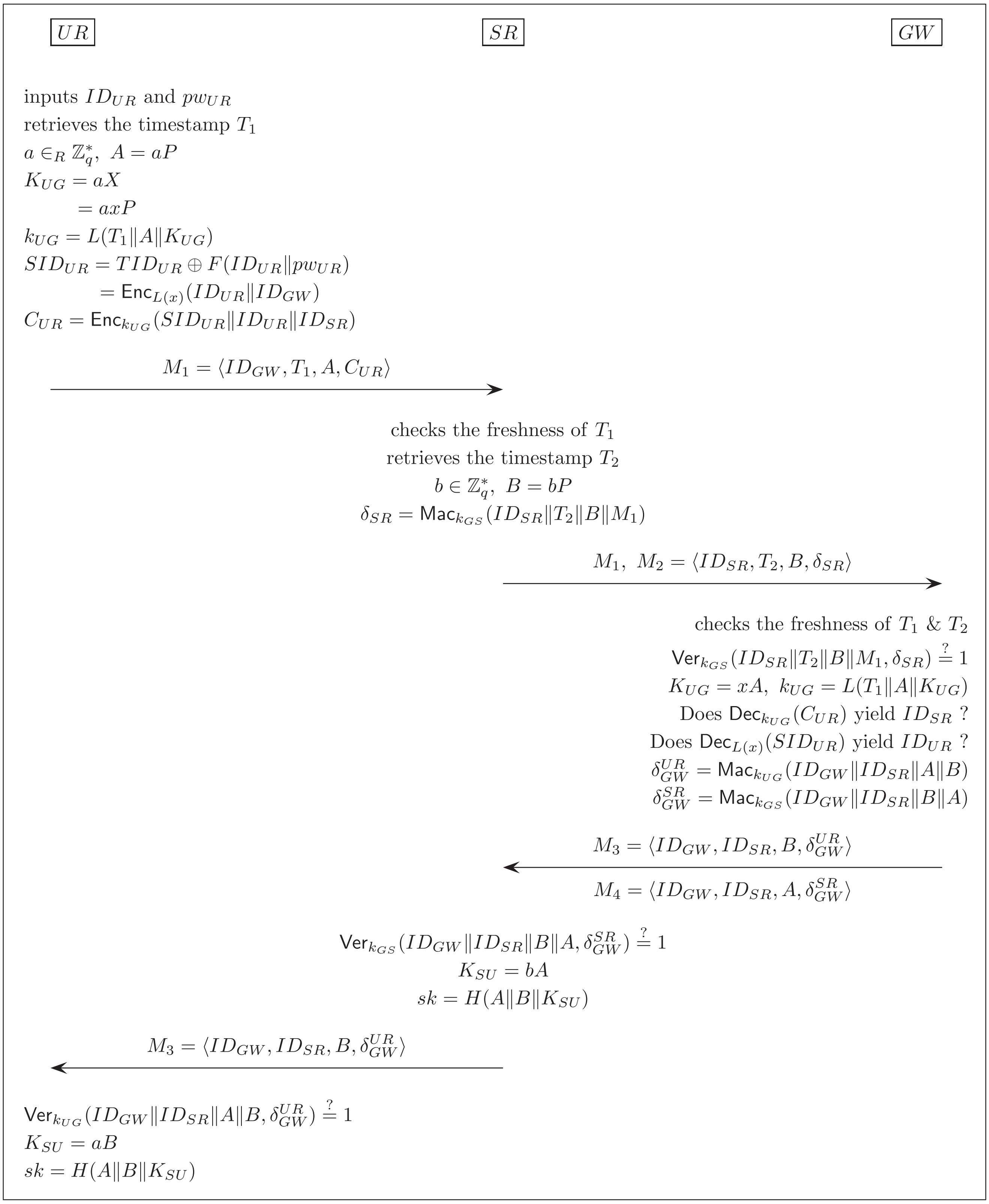
Figure 1. User registration.


UR
GW
$(x \in \mathbb{Z}_q^*,\ X = xP)$
chooses $ID_{UR},\ pw_{UR}$
$\langle ID_{UR} \rangle$
$SID_{UR} = \mathsf{Enc}_{L(x)}(ID_{UR}\|ID_{GW})$
smart card : $\{SID_{UR}, X, ID_{GW}, \mathbb{G}, P, \Gamma, \Delta, L, H, F\}$
$TID_{UR} = SID_{UR} \oplus F(ID_{UR}\|pw_{UR})$
replaces SID_{UR} with TID_{UR}

3.2.2. Authentication and Key Exchange Phase

UR needs to perform this phase with SR and GW whenever it wishes to gain access to the sensor network and data. The steps of the phase are depicted in Figure 2 and are described as follows:

Figure 2. The authentication and key exchange protocol.



$\boxed{UR}$ $\boxed{SR}$ $\boxed{GW}$

UR:

inputs ID_{UR} and pw_{UR}
retrieves the timestamp T_1
$a \in_R \mathbb{Z}_q^*,\ A = aP$
$K_{UG} = aX$
$\quad = axP$
$k_{UG} = L(T_1 \| A \| K_{UG})$
$SID_{UR} = TID_{UR} \oplus F(ID_{UR} \| pw_{UR})$
$\quad = \mathsf{Enc}_{L(x)}(ID_{UR} \| ID_{GW})$
$C_{UR} = \mathsf{Enc}_{k_{UG}}(SID_{UR} \| ID_{UR} \| ID_{SR})$

$$\xrightarrow{M_1 = \langle ID_{GW}, T_1, A, C_{UR} \rangle}$$

SR:

checks the freshness of T_1
retrieves the timestamp T_2
$b \in \mathbb{Z}_q^*,\ B = bP$
$\delta_{SR} = \mathsf{Mac}_{k_{GS}}(ID_{SR} \| T_2 \| B \| M_1)$

$$\xrightarrow{M_1,\ M_2 = \langle ID_{SR}, T_2, B, \delta_{SR} \rangle}$$

GW:

checks the freshness of T_1 & T_2
$\mathsf{Ver}_{k_{GS}}(ID_{SR} \| T_2 \| B \| M_1, \delta_{SR}) \overset{?}{=} 1$
$K_{UG} = xA,\ k_{UG} = L(T_1 \| A \| K_{UG})$
Does $\mathsf{Dec}_{k_{UG}}(C_{UR})$ yield ID_{SR} ?
Does $\mathsf{Dec}_{L(x)}(SID_{UR})$ yield ID_{UR} ?
$\delta_{GW}^{UR} = \mathsf{Mac}_{k_{UG}}(ID_{GW} \| ID_{SR} \| A \| B)$
$\delta_{GW}^{SR} = \mathsf{Mac}_{k_{GS}}(ID_{GW} \| ID_{SR} \| B \| A)$

$$\xleftarrow[M_4 = \langle ID_{GW}, ID_{SR}, A, \delta_{GW}^{SR} \rangle]{M_3 = \langle ID_{GW}, ID_{SR}, B, \delta_{GW}^{UR} \rangle}$$

SR:

$\mathsf{Ver}_{k_{GS}}(ID_{GW} \| ID_{SR} \| B \| A, \delta_{GW}^{SR}) \overset{?}{=} 1$
$K_{SU} = bA$
$sk = H(A \| B \| K_{SU})$

$$\xleftarrow{M_3 = \langle ID_{GW}, ID_{SR}, B, \delta_{GW}^{UR} \rangle}$$

UR:

$\mathsf{Ver}_{k_{UG}}(ID_{GW} \| ID_{SR} \| A \| B, \delta_{GW}^{UR}) \overset{?}{=} 1$
$K_{SU} = aB$
$sk = H(A \| B \| K_{SU})$

Step 1. UR inserts its smart card into a card reader and inputs its identity ID_{UR} and password pw_{UR}. Given ID_{UR} and pw_{UR}, the smart card retrieves the current timestamp T_1, selects a random $a \in \mathbb{Z}_q^*$ and computes:

$$\begin{aligned} A &= aP, \\ K_{UG} &= aX \\ &= axP, \\ k_{UG} &= L(T_1\|A\|K_{UG}), \\ SID_{UR} &= TID_{UR} \oplus F(ID_{UR}\|pw_{UR}) \\ &= \mathsf{Enc}_{L(x)}(ID_{UR}\|ID_{GW}), \\ C_{UR} &= \mathsf{Enc}_{k_{UG}}(SID_{UR}\|ID_{UR}\|ID_{SR}). \end{aligned}$$

After the computations, the smart card sends the message $M_1 = \langle ID_{GW}, T_1, A, C_{UR}\rangle$ to the sensor SR.

Step 2. Upon receiving M_1, SR first checks the freshness of T_1. If T_1 is not fresh, SR aborts the protocol. Otherwise, SR retrieves the current timestamp T_2, chooses a random $b \in \mathbb{Z}_q^*$ and computes B and δ_{SR} as follows:

$$\begin{aligned} B &= bP, \\ \delta_{SR} &= \mathsf{Mac}_{k_{GS}}(ID_{SR}\|T_2\|B\|M_1). \end{aligned}$$

Then, SR sends the message $M_2 = \langle ID_{SR}, T_2, B, \delta_{SR}\rangle$ along with M_1 to GW.

Step 3. After having received M_1 and M_2, GW verifies that: (1) T_1 and T_2 are fresh; and (2) $\mathsf{Ver}_{k_{GS}}(ID_{SR}\|T_2\|B\|M_1, \delta_{SR}) = 1$. If any of the verifications fails, GW aborts the protocol. Otherwise, GW computes $K_{UG} = xA$ and $k_{UG} = L(T_1\|A\|K_{UG})$, decrypts C_{UR} with key k_{UG} and checks if the decryption produces the same ID_{SR} as contained in M_2. GW aborts if the check fails. Otherwise, GW decrypts SID_{UR} with key $L(x)$ and checks if this decryption yields the same ID_{UR} as produced through the decryption of C_{UR}. If only the two IDs match, GW computes:

$$\begin{aligned} \delta_{GW}^{UR} &= \mathsf{Mac}_{k_{UG}}(ID_{GW}\|ID_{SR}\|A\|B), \\ \delta_{GW}^{SR} &= \mathsf{Mac}_{k_{GS}}(ID_{GW}\|ID_{SR}\|B\|A), \end{aligned}$$

and sends two messages $M_3 = \langle ID_{GW}, ID_{SR}, B, \delta_{GW}^{UR}\rangle$ and $M_4 = \langle ID_{GW}, ID_{SR}, A, \delta_{GW}^{SR}\rangle$ to SR.

Step 4. When receiving M_3 and M_4, SR verifies that $\mathsf{Ver}_{k_{GS}}(ID_{GW}\|ID_{SR}\|B\|A, \delta_{GW}^{SR}) = 1$. If the verification fails, SR aborts the protocol. Otherwise, SR forwards the message M_3 to UR and computes the shared secret $K_{SU} = bA$ and the session key $sk = H(A\|B\|K_{SU})$.

Step 5. Upon receiving M_3, UR checks if $\mathsf{Ver}_{k_{UG}}(ID_{GW}\|ID_{SR}\|A\|B,\ \delta_{GW}^{UR}) = 1$. UR aborts the protocol if the check fails. Otherwise, UR computes $K_{SU} = aB$ and $sk = H(A\|B\|K_{SU})$.

Since $K_{SU} = bA = aB = abP$, UR and SR will compute the same session key $sk = H(A\|B\|abP)$ in the presence of a passive adversary.

3.2.3. Password Update Phase

One of the general guidelines to get better password security is to ensure that passwords are changed at regular intervals. Our scheme allows users to update their passwords at will.

1. UR inserts his smart card into a card reader and enters the identity ID_{UR}, the current password pw_{UR} and the new password pw'_{UR}.
2. The smart card computes $TID'_{UR} = TID_{UR} \oplus F(ID_{UR}\|pw_{UR}) \oplus F(ID_{UR}\|pw'_{UR})$ and replaces TID_{UR} with TID'_{UR}.

3.3. Performance and Security Comparison

Table 3 compares our scheme with other ECC-based SUA-WSN schemes in terms of the computational requirements, the AKE security and user anonymity. For fairness of comparison, SUA-WSN schemes that use only lightweight symmetric cryptographic primitives are not considered in the table since they cannot achieve forward secrecy, but have a clear efficiency advantage over the ECC-based schemes.

The scalar-point multiplication and map-to-point operation are much more expensive than the other operations considered in the table, such as symmetric encryption/decryption, MAC generation/verification and hash function evaluation. The total number of modular exponentiations and map-to-point operations required in Yeh *et al.*'s scheme [16] is 10, while the number is reduced to six in the other schemes. Therefore, the overall performance of Yeh *et al.*'s scheme is not as good as those of the other schemes.

Table 3. A comparison of elliptic curve cryptography (ECC)-based SUA-WSN schemes. AKE, authenticated key exchange.

Scheme	**Computation**		**Security**	
	SR	$UR + SR + GW$	**AKE**	**Anonymity**
Our scheme	$2M + 2A + 1H$	$6M + 3E + 6A + 6H$	Proven	Proven
Choi *et al.* [27]	$2M + 5H$	$6M + 18H$	Proven using a computer security approach	No
Shi and Gong [21]	$2M + 4H$	$6M + 15H$	Broken [27]	No
Yeh *et al.* [16]	$2M + 1P + 2H$	$8M + 2P + 9H$	Broken [30]	No

M: scalar-point multiplication; P: map-to-point operation; E: symmetric encryption/decryption; A: MAC generation/verification; H: hash function evaluation.

From the viewpoint of the computational burden on the sensor SR, our scheme is competitive with Choi *et al.*'s scheme [27] and Shi and Gong's scheme [21], since a MAC generation/verification is almost as fast as a hash function evaluation. According to Crypto++ benchmarks, HMACwith SHA-1 takes 11.9 cycles per byte, while SHA-1 takes 11.4 cycles per byte (see Table 4).

Table 4. A result of Crypto++ benchmarks for HMAC, SHA-1 and AES.

Algorithm	**HMAC (SHA-1)**	**SHA-1**	**AES/CTR**	**AES/CBC**	**AES/OFB**	**AES/ECB**
Cycles Per Byte	11.9	11.4	12.6	16.0	16.9	16.0

Another point we wish to make is that a hash function evaluation with a long input string may not be faster than a symmetric encryption with a relatively short plain text input, though the opposite is generally true for the same length of inputs. For example, the computation of the ciphertext C_{UR} in our scheme is unlikely to be more expensive than the computations of the hash values β, γ and δ, which are defined in both Choi *et al.*'s scheme and Shi and Gong's scheme. In this sense, it is fair to say that our scheme is competitive also in terms of the overall computational cost.

As is obvious from the table, our scheme is the only one that provides user anonymity (regardless of whether it is proven or not). This explains how the other schemes could have been designed without using any form of encryption algorithm. Choi *et al.* [27] prove that their scheme achieves the AKE security, but only using a computer security approach. In contrast, we use a computational complexity approach in proving both the AKE security and the user anonymity property.

4. Security Results

Let P denote the authentication and key exchange protocol of our scheme depicted in Figure 2. This section proves that the protocol P is AKE-secure and provides user anonymity (against any party other than the gateway GW); see Section 2 for the formal definitions of the AKE security and the user anonymity property.

4.1. Proof of AKE Security

Theorem 1. *Our authentication and key exchange protocol P is AKE-secure in the random oracle model under the ECCDH assumption in $\mathbb{G}$ and the security of the MAC scheme Δ.*

Proof. Assume a PPT adversary $\mathcal{A}$ against the AKE security of the protocol P. We prove the theorem by making a series of modifications to the original experiment $\mathbf{ExpAKE}_0$, bounding the effect of each change in the experiment on the advantage of $\mathcal{A}$ and ending up with an experiment where $\mathcal{A}$ has no advantage (*i.e.*, $\mathcal{A}$ has a success probability of 1/2). Let $\mathsf{SuccAKE}_i$ denote the event that $\mathcal{A}$ correctly guesses the random bit b selected by the $\mathsf{TestAKE}$ oracle in experiment $\mathbf{ExpAKE}_i$. Let t_i be the maximum time required to perform the experiment $\mathbf{ExpAKE}_i$ involving the adversary $\mathcal{A}$.

Experiment $\mathbf{ExpAKE}_1$. In this first modified experiment, the simulator answers the queries to the L oracle as follows:

<u>Simulation of the L oracle</u>

For each query of L on a string m, the simulator first checks if an entry of the form (m, l) is in a list called LList, which is maintained to store input-output pairs of L. If it is, the simulator outputs l as the answer to the hash query. Otherwise, the simulator chooses a random ℓ-bit string str, answers the query with str and adds the entry (m, str) to LList.

This is the only difference between $\mathbf{ExpAKE}_1$ and $\mathbf{ExpAKE}_0$; the simulator answers all other oracle queries of $\mathcal{A}$ as in the original experiment $\mathbf{ExpAKE}_0$. Then, since $\mathbf{ExpAKE}_1$ is perfectly indistinguishable from $\mathbf{ExpAKE}_0$, it follows that:

Claim 1. $\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_1] = \Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_0]$.

Experiment $\mathbf{ExpAKE}_2$. In this experiment, we modify the computations of X and A as follows:

The $\mathbf{ExpAKE}_2$ modification

- The simulator chooses two random elements $Y, Y' \in \mathbb{G}$ and sets $X = Y'$.
- For every fresh instance, the simulator chooses a random $r \in \mathbb{Z}_q^*$ and sets $A = rY$. For other instances, the simulator computes A as in experiment $\mathbf{ExpAKE}_1$.

Due to the modification, the simulator does not know the master secret x. The simulator aborts the experiment if $\mathcal{A}$ makes the $\mathsf{CorruptLL}(GW)$ query. However, in this case, $\mathcal{A}$ cannot gain any advantage, as no instance is considered fresh. In this experiment, the simulator simply sets each k_{UG} to a random ℓ-bit string, since it does not know the ephemeral secret a and, thus, cannot compute the secret K_{UG}. This means that the success probability of $\mathcal{A}$ may be different between $\mathbf{ExpAKE}_1$ and $\mathbf{ExpAKE}_2$ if it asks an $L(T_1\|A\|K_{UG})$ query. However, this difference is bounded by Claim 2.

Claim 2. $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_2] - \Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_1]\right| \leq 1/q_L \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(t_2)$, *where* q_L *is the number of queries made of the* L *oracle.*

Proof. We prove the claim via a reduction from the ECCDH problem, which is believed to be hard, to the problem of distinguishing two experiments $\mathbf{ExpAKE}_1$ and $\mathbf{ExpAKE}_2$. Assume that the success probability of $\mathcal{A}$ is non-negligibly different between $\mathbf{ExpAKE}_1$ and $\mathbf{ExpAKE}_2$. Then, we construct an algorithm $\mathcal{A}_{\mathrm{ECCDH}}$ that solves the ECCDH problem in $\mathbb{G}$ with a non-negligible advantage. The objective of $\mathcal{A}_{\mathrm{ECCDH}}$ is to compute and output the value $W = uvP \in \mathbb{G}$ when given an ECCDH-problem instance $(U = uP, V = vP) \in \mathbb{G}$. $\mathcal{A}_{\mathrm{ECCDH}}$ runs $\mathcal{A}$ as a subroutine while simulating all of the oracles on its own.

$\mathcal{A}_{\mathrm{ECCDH}}$ handles all of the oracle queries of $\mathcal{A}$ as specified in experiment $\mathbf{ExpAKE}_2$, but using U and V in place of X and Y. When $\mathcal{A}$ outputs its guess b', $\mathcal{A}_{\mathrm{ECCDH}}$ chooses an entry of the form $(T_1\|A\|K, l)$ at random from LList and terminates outputting K/r. >From the simulation, it is not hard to see that $\mathcal{A}_{\mathrm{ECCDH}}$ outputs the desired result $W = uvP$ with probability at least $1/q_L$ if $\mathcal{A}$ makes a $L(T_1\|A\|K_{UG})$ query for some fresh user instance. This completes Claim 2. □

Before proceeding further, we define the event Forge as follows:

Forge: The event that the adversary $\mathcal{A}$ asks a Send query of the form $\mathsf{Send}(\Pi_E^i, E'\|msg)$ for uncorrupted E and E', such that msg contains a MAC forgery.

Experiment $\mathbf{ExpAKE}_3$. This experiment is different from $\mathbf{ExpAKE}_2$ in that it is aborted and the adversary $\mathcal{A}$ does not succeed if the event Forge occurs. Then, we have:

Claim 3. $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_3] - \Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_2]\right| \leq q_{\mathrm{send}} \cdot \mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(t_3)$, *where* q_{send} *is the number of queries made for the oracle* Send.

Proof. Assume that the event Forge occurs with a non-negligible probability. Then, we construct an algorithm $\mathcal{A}_{\mathsf{forge}}$ who generates, with a non-negligible probability, a forgery against the MAC scheme Δ. The algorithm $\mathcal{A}_{\mathsf{forge}}$ is given access to the $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Ver}_k(\cdot)$ oracles. The objective of $\mathcal{A}_{\mathsf{forge}}$ is to produce a message/MAC pair (m, δ), such that: (1) $\mathsf{Ver}_k(m, \delta) = 1$; and (2) δ has not been output by the oracle $\mathsf{Mac}_k(\cdot)$ on input m.

Let n_k be the total number of MAC keys used in the sessions initiated via a Send query. Clearly, $n_k \leq q_{\text{send}}$. $\mathcal{A}_{\mathsf{forge}}$ begins by selecting a random $i \in \{1, \ldots, n_k\}$. Let k_i denote the $i-\mathrm{th}$ key among all of the n_k MAC keys and Send_i be any Send query that is expected to be answered and/or verified using k_i. $\mathcal{A}_{\mathsf{forge}}$ runs $\mathcal{A}$ as a subroutine and answers the oracle queries of $\mathcal{A}$ as in experiment $\mathbf{ExpAKE}_2$, except that: it answers all Send_i queries by accessing its $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Ver}_k(\cdot)$ oracles. As a result, the $i-\mathrm{th}$ MAC key k_i is not used during the simulation. If Forge occurs against an instance who holds k_i, $\mathcal{A}_{\mathsf{forge}}$ halts and outputs the message/MAC pair generated by $\mathcal{A}$ as its forgery. Otherwise, $\mathcal{A}_{\mathsf{forge}}$ terminates with a failure indication.

If the guess i is correct, then the simulation is perfect and $\mathcal{A}_{\mathsf{forge}}$ achieves its goal. Namely, $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A}_{\mathsf{forge}}) = \mathsf{Pr}[\mathsf{Forge}]/n_k$. Since $n_k \leq q_{\text{send}}$, we get $\mathsf{Pr}[\mathsf{Forge}] \leq q_{\text{send}} \cdot \mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A}_{\mathsf{forge}})$. Since $\mathcal{A}_{\mathsf{forge}}$ runs in time at most t_3, it follows, by definition, that $\mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(\mathcal{A}_{\mathsf{forge}}) \leq \mathsf{Adv}_{\Delta}^{\mathrm{EU-CMA}}(t_3)$. This completes the proof of Claim 3. $\square$

Experiment $\mathbf{ExpAKE}_4$. We next modify the way of answering queries of the H oracle as follows:

> <u>Simulation of the H oracle</u>
>
> For each H query on a string m, the simulator first checks if an entry of the form (m, h) is in a list called HList, which is maintained to store input-output pairs of H. If it is, h is the answer to the hash query. Otherwise, the simulator chooses a random κ-bit string str, answers the query with str and adds the entry (m, str) to HList.

Other oracle queries of $\mathcal{A}$ are handled as in experiment $\mathbf{ExpAKE}_3$. Since $\mathbf{ExpAKE}_4$ is perfectly indistinguishable from $\mathbf{ExpAKE}_3$, it is clear that:

Claim 4. $\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_4] = \Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_3]$.

Experiment $\mathbf{ExpAKE}_5$. We finally modify the experiment so that, for each fresh instance of SR, the computation of B is done as follows:

> <u>The $\mathbf{ExpAKE}_5$ modification</u>
>
> The simulator selects a random $r' \in \mathbb{Z}_q^*$ and computes $B = r'X$.

The simulator sets the session key sk to a random κ-bit string for each pair of fresh instances, as it cannot compute K_{SU}. Accordingly, the success probability of $\mathcal{A}$ may be different between $\mathbf{ExpAKE}_4$ and $\mathbf{ExpAKE}_5$ if it asks an $H(A\|B\|K_{SU})$ query. Claim 5 below bounds the difference:

Claim 5. $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_5] - \Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_4]\right| \leq 1/q_H \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(t_5)$, *where q_H is the number of queries made of the H oracle.*

Proof. Suppose that the difference in the advantage of $\mathcal{A}$ between $\mathsf{SuccAKE}_4$ and $\mathsf{SuccAKE}_5$ is non-negligible. Then, from $\mathcal{A}$, we construct an algorithm $\mathcal{A}_{\text{ECCDH}}$ that solves the ECCDH problem in $\mathbb{G}$ with a non-negligible advantage. The objective of $\mathcal{A}_{\text{ECCDH}}$ is to compute and output the value $W = uvP \in \mathbb{G}$ when given an ECCDH-problem instance $(U = uP, V = vP) \in \mathbb{G}$.

$\mathcal{A}_{\text{ECCDH}}$ runs $\mathcal{A}$ as a subroutine while answering all of the oracles queries by itself. $\mathcal{A}_{\text{ECCDH}}$ handles the queries of $\mathcal{A}$ as specified in the $\mathbf{ExpAKE}_5$ experiment, but using U and V in place of X and Y. When $\mathcal{A}$ terminates and outputs its guess b', $\mathcal{A}_{\text{ECCDH}}$ selects an entry of the form $(A\|B\|K, h)$ at random from HList and outputs K/rr'. If $\mathcal{A}$ makes a $H(A\|B\|K_{SU})$ query, $\mathcal{A}_{\text{ECCDH}}$ outputs the desired result $W = uvP$ with probability at least $1/q_H$. This completes Claim 5. □

In experiment $\mathbf{ExpAKE}_5$, the adversary $\mathcal{A}$ obtains no information on the random bit b selected by the $\mathsf{TestAKE}$ oracle, since the session keys of all fresh instances are selected uniformly at random from $\mathbb{G}$. Therefore, it follows that $\Pr_{P,\mathcal{A}}[\mathsf{SuccAKE}_5] = 1/2$. This result combined with Claims 1–5 completes the proof of Theorem 1. □

4.2. Proof of User Anonymity

Theorem 2. *The authentication and key exchange protocol P provides user anonymity in the random oracle model under the ECCDH assumption in $\mathbb{G}$ and the security of the symmetric encryption scheme Γ.*

Proof. Assume a PPT adversary $\mathcal{A}$ against the user anonymity property of the protocol P. As in the proof of Theorem 1, we make a series of modifications to the original experiment $\mathbf{ExpID}_0$, bounding the difference in the success probability of $\mathcal{A}$ between two consecutive experiments and then ending up with an experiment where $\mathcal{A}$ has a success probability of 1/2 (*i.e.*, $\mathcal{A}$ has no advantage). We use SuccID_i to denote the event that $\mathcal{A}$ correctly guesses the random bit b selected by the TestID oracle in experiment $\mathbf{ExpID}_i$. Let t_i be the maximum time required to perform the experiment $\mathbf{ExpID}_i$ involving the adversary $\mathcal{A}$.

Experiment $\mathbf{ExpID}_1$. This experiment is different from $\mathbf{ExpID}_0$ in that the random oracle L is simulated as follows:

<u>Simulation of the L oracle</u>

For each query to L on a string m, the simulator first checks if an entry of the form (m, l) is in a list called LList, which is maintained to store input-output pairs of L. If it is, the simulator outputs l as the answer to the hash query. Otherwise, the simulator chooses a random ℓ-bit string str, answers the query with str and adds the entry (m, str) to LList.

Other oracle queries of $\mathcal{A}$ are answered as in the original experiment $\mathbf{ExpID}_0$. Then, since L is a random oracle, $\mathbf{ExpID}_1$ is perfectly indistinguishable from $\mathbf{ExpID}_0$, and Claim 6 immediately follows.

Claim 6. $\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_1] = \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_0]$.

Experiment $\mathbf{ExpID}_2$. Here, we modify the experiment so that A is computed as follows:

The **ExpID**$_2$ modification

- The simulator chooses a random exponent $y \in \mathbb{Z}_q^*$ and computes $Y = yP$.
- For each instance of users, the simulator chooses a random $r \in \mathbb{Z}_q^*$ and sets $A = rY$.

As a result of the modification, each K_{UG} is set to $xyrP$ for some random $r \in \mathbb{Z}_q^*$. Since the view of $\mathcal{A}$ is identical between **ExpID**$_2$ and **ExpID**$_1$, it follows that:

Claim 7. $\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_2] = \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_1]$.

Experiment **ExpID**$_3$. In this experiment, we modify the computations of X and A as follows:

The **ExpID**$_3$ modification

- The simulator chooses two random elements $Y, Y' \in \mathbb{G}$ and sets $X = Y'$.
- For instances of every clean user, the simulator chooses a random $r \in \mathbb{Z}_q^*$ and sets $A = rY$. For other instances, the simulator computes A as in experiment **ExpID**$_2$.

As a result, the simulator does not know the master secret x. The simulator aborts the experiment if $\mathcal{A}$ makes the $\mathsf{CorruptLL}(GW)$ query. However, in this case, $\mathcal{A}$ cannot gain any advantage, as no user is considered clean (see Definition 3). In this experiment, the simulator simply sets each k_{UG} to a random ℓ-bit string, since it does not know the ephemeral secret a and, thus, cannot compute the secret K_{UG}. This means that the success probability of $\mathcal{A}$ may be different between **ExpID**$_2$ and **ExpID**$_3$ if it asks an $L(T_1\|A\|K_{UG})$ query. However, this difference is bounded by Claim 8.

Claim 8. $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_3] - \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_2]\right| \leq 1/q_L \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECCDH}}(t_3)$, *where q_L is the number of queries made to the L oracle.*

Proof. We prove the claim via a reduction from the ECCDH problem, which is believed to be hard, to the problem of distinguishing two experiments **ExpID**$_2$ and **ExpID**$_3$. Assume that the success probability of $\mathcal{A}$ is non-negligibly different between **ExpID**$_2$ and **ExpID**$_3$. Then, we construct an algorithm $\mathcal{A}_{\mathrm{ECCDH}}$ that solves the ECCDH problem in $\mathbb{G}$ with a non-negligible advantage. The objective of $\mathcal{A}_{\mathrm{ECCDH}}$ is to compute and output the value $W = uvP \in \mathbb{G}$ when given an ECCDH-problem instance $(U = uP, V = vP) \in \mathbb{G}$. $\mathcal{A}_{\mathrm{ECCDH}}$ runs $\mathcal{A}$ as a subroutine while simulating all of the oracles on its own.

$\mathcal{A}_{\mathrm{ECCDH}}$ handles all of the oracle queries of $\mathcal{A}$ as specified in experiment **ExpID**$_3$, but using U and V in place of X and Y. When $\mathcal{A}$ outputs its guess b', $\mathcal{A}_{\mathrm{ECCDH}}$ chooses an entry of the form $(T_1\|A\|K, l)$ at random from LList and terminates outputting K/r. >From the simulation, it is clear that $\mathcal{A}_{\mathrm{ECCDH}}$ outputs the desired result $W = uvP$ with a probability of at least $1/q_L$ if $\mathcal{A}$ makes a $L(T_1\|A\|K_{UG})$ query for some clean $UR \in \mathcal{URS}$. This completes Claim 8. □

Experiment **ExpID**$_4$. We finally modify the experiment so that, for each clean user $UR \in \mathcal{URS}$, a random identity ID'_{UR} drawn from the identity space is used in place of the true identity ID_{UR} in generating C_{UR}.

Claim 9. $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_4] - \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_3]\right| \leq \mathsf{Adv}_{\Gamma}^{\mathrm{IND-EAV}}(t_4)$.

Proof. We prove the claim by constructing an eavesdropper $\mathcal{A}_{\text{eav}}$ who attacks the indistinguishability of Γ with advantage equal to $\left|\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_4] - \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_3]\right|$.

$\mathcal{A}_{\text{eav}}$ begins by choosing a random bit $b \in \{0,1\}$. Then, $\mathcal{A}_{\text{eav}}$ invokes the adversary $\mathcal{A}$ and answers all of the oracle queries of $\mathcal{A}$ as in experiment $\mathbf{ExpID}_3$, except that, for each clean user $UR \in \mathcal{URS}$, it generates C_{UR} by accessing its own encryption oracle as follows:

> $\mathcal{A}_{\text{eav}}$ outputs $(SID_{UR}\|ID_{UR}\|ID_{SR}, SID_{UR}\|ID'_{UR}\|ID_{SR})$ as its plain text pair in the indistinguishability experiment $\mathbf{Exp}_{\Gamma}^{\text{IND-EAV}}$. Let c be the ciphertext received in return for the plain text pair. $\mathcal{A}_{\text{eav}}$ sets C_{UR} equal to the ciphertext c.

That is, $\mathcal{A}_{\text{eav}}$ sets C_{UR} to the encryption of either $SID_{UR}\|ID_{UR}\|ID_{SR}$ or $SID_{UR}\|ID'_{UR}\|ID_{SR}$. Now, when $\mathcal{A}$ terminates and outputs its guess b', $\mathcal{A}_{\text{eav}}$ outputs one if $b = b'$, and zero otherwise. Then, it is clear that:

- The probability that $\mathcal{A}_{\text{eav}}$ outputs one when the first plain texts are encrypted in the experiment $\mathbf{Exp}_{\Gamma}^{\text{IND-EAV}}$ is equal to the probability that $\mathcal{A}$ succeeds in the experiment $\mathbf{ExpID}_3$.
- The probability that $\mathcal{A}_{\text{eav}}$ outputs one when the second plain texts are encrypted in the experiment $\mathbf{Exp}_{\Gamma}^{\text{IND-EAV}}$ is equal to the probability that $\mathcal{A}$ succeeds in the experiment $\mathbf{ExpID}_4$.

That is, $\mathsf{Adv}_{\Gamma}^{\text{IND-EAV}}(\mathcal{A}_{\text{eav}}) = \left|\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_4] - \Pr_{P,\mathcal{A}}[\mathsf{SuccID}_3]\right|$. Note that in the simulation, $\mathcal{A}_{\text{eav}}$ eavesdrops at most q_{send} encryptions, which is polynomial in the security parameter ℓ. This completes the proof of Claim 9. □

In the experiment $\mathbf{ExpID}_4$, the adversary $\mathcal{A}$ cannot gain any information on the random bit b selected by the TestID oracle, because the identities of all clean users are chosen uniformly at random from the identity space. It, therefore, follows that $\Pr_{P,\mathcal{A}}[\mathsf{SuccID}_4] = 1/2$. This result combined with Claims 6–9 yields the statement of Theorem 2. □

5. Concluding Remarks

We have extended the widely-accepted security model of Bellare, Pointcheval and Rogaway [10] to formally capture the security requirements for SUA-WSN schemes—smart-card-based user authentication schemes for wireless sensor networks. Our extended model provides formal definitions of the AKE security and the user anonymity property, while capturing the notion of two-factor security. We have also proposed a new SUA-WSN scheme and proved that it achieves user anonymity, as well as the AKE security in the extended model. To the best of our knowledge, our scheme is the first SUA-WSN scheme that is proven secure in a widely-accepted model.

We believe that our result lays a solid foundation for designing provably-secure two-factor authentication schemes for mobile roaming services, where user anonymity, as well as authenticated key exchange are also of critical security importance; see, e.g., the recent work of He *et al.* [33,34]. A concrete design of such a provably-secure roaming authentication scheme would be interesting future work. We also leave it as future work to present a formal treatment of security properties for three-factor authentication schemes [35].

Acknowledgments

This work was supported by Konkuk University.

Author contributions

M.K. and Y.L. conceived and designed the experiments; M.K. and Y.L. performed the experiments; J.P. and D.W. analyzed the data; J.N. and D.W. proved the security of the protocol; J.N. and J.P. wrote the paper.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Rawat, P.; Singh, K.; Chaouchi, H.; Bonnin, J. Wireless sensor networks: A survey on recent developments and potential synergies. *J. Supercomput.* **2014**, *68*, 1–48.
2. Kumar, P.; Choudhury, A.; Sain, M.; Lee, S.; Lee, H. RUASN: A robust user authentication framework for wireless sensor networks. *Sensors* **2011**, *11*, 5020–5046.
3. Khan, M.; Kumari, S. An improved user authentication protocol for healthcare services via wireless medical sensor networks. *Int. J. Distrib. Sens. Netw.* **2014**, *2014*, No. 347169.
4. Choo, K.K.R.; Boyd, C.; Hitchcock, Y. Errors in computational complexity proofs for protocols. In Proceedings of ASIACRYPT 2005, Chennai, India, 4–8 December 2005; pp. 624–643.
5. Nam, J.; Paik, J.; Won, D. A security weakness in Abdalla *et al.*'s generic construction of a group key exchange protocol. *Inf. Sci.* **2011**, *181*, 234–238.
6. Nam, J.; Choo, K.K.R.; Kim, M.; Paik, J.; Won, D. Dictionary attacks against password-based authenticated three-party key exchange protocols. *KSII Trans. Internet Inf. Syst.* **2013**, *7*, 3244–3260.
7. Choo, K.K.R. Refuting security proofs for tripartite key exchange with model checker in planning problem setting. In Proceedings of 19th IEEE Computer Security Foundations Workshop, Venice, Italy, 5–7 July 2006; pp. 297–308.
8. Choo, K.K.R. *Secure Key Establishment*; Springer: Berlin, Germany, 2008.
9. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208.
10. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated key exchange secure against dictionary attacks. In Proceedings of EUROCRYPT 2000, Bruges, Belgium, 14–18 May 2000; pp. 139–155.
11. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, Innsbruck, Austria, 6–10 May 2001; pp. 453–474.
12. Das, M. Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1086–1090.

13. He, D.; Gao, Y.; Chan, S.; Chen, C.; Bu, J. An enhanced two-factor user authentication scheme in wireless sensor networks. *Adhoc Sens. Wirel. Netw.* **2010**, *10*, 361–371.
14. Khan, M.; Alghathbar, K. Cryptanalysis and security improvements of "two-factor user authentication in wireless sensor networks". *Sensors* **2010**, *10*, 2450–2459.
15. Chen, T.; Shih, W. A robust mutual authentication protocol for wireless sensor networks. *ETRI J.* **2010**, *32*, 704–712.
16. Yeh, H.; Chen, T.; Liu, P.; Kim, T.; Wei, H. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* **2011**, *11*, 4767–4779.
17. Kumar, P.; Lee, S.; Lee, H. E-SAP: Efficient-strong authentication protocol for healthcare applications using wireless medical sensor networks. *Sensors* **2012**, *12*, 1625–1647.
18. Yoo, S.; Park, K.; Kim, J. A security-performance-balanced user authentication scheme for wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2012**, *2012*, No. 382810.
19. Vaidya, B.; Makrakis, D.; Mouftah, H. Two-factor mutual authentication with key agreement in wireless sensor networks. *Secur. Commun. Netw.* **2012**, doi:10.1002/sec.517.
20. Xue, K.; Ma, C.; Hong, P.; Ding, R. A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. *J. Netw. Comput. Appl.* **2013**, *36*, 316–323.
21. Shi, W.; Gong, P. A new user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, No. 730831.
22. Kumar, P.; Gurtov, A.; Ylianttila, M.; Lee, S.; Lee, H. A strong authentication scheme with user privacy for wireless sensor networks. *ETRI J.* **2013**, *35*, 889–899.
23. He, D.; Kumar, N.; Chen, J.; Lee, C.; Chilamkurti, N.; Yeo, S. Robust anonymous authentication protocol for health-care applications using wireless medical sensor networks. *Multimed. Syst.* **2013**, doi:10.1007/s00530-013-0346-9.
24. Chi, L.; Hu, L.; Li, H.; Chu, J. Analysis and improvement of a robust user authentication framework for ubiquitous sensor networks. *Int. J. Distrib. Sens. Netw.* **2014**, *2014*, No. 637684.
25. Kim, J.; Lee, D.; Jeon, W.; Lee, Y.; Won, D. Security analysis and improvements of two-factor mutual authentication with key agreement in wireless sensor networks. *Sensors* **2014**, *14*, 6443–6462.
26. Jiang, Q.; Ma, J.; Lu, X.; Tian, Y. An efficient two-factor user authentication scheme with unlinkability for wireless sensor networks. *Peer-to-Peer Netw. Appl.* **2014**, doi:10.1007/s12083-014-0285-z.
27. Choi, Y.; Lee, D.; Kim, J.; Jung, J.; Nam, J.; Won, D. Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* **2014**, *14*, 10081–10106.
28. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of CRYPTO 1999, Santa Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
29. Messerges, T.; Dabbish, E.; Sloan, R. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552.
30. Han, W. Weakness of a secured authentication protocol for wireless sensor networks using elliptic curves cryptography. Available online: http://eprint.iacr.org/2011/293 (accessed on 15 August 2014).

31. Kumar, P.; Lee, H. Cryptanalysis on two user authentication protocols using smart card or wireless sensor networks. In Proceedings of the IEEE Wireless Advanced (WiAd), London, UK, 20–22 June 2011; pp. 241–245.
32. Nam, J.; Choo, K.K.R.; Paik, J.; Won, D. Password-only authenticated three-party key exchange proven secure against insider dictionary attacks. *Sci. World J.* **2014**, *2014*, No. 802359.
33. He, D.; Kumar, N.; Khan, M.; Lee, J. Anonymous two-factor authentication for consumer roaming service in global mobility networks. *IEEE Trans. Consum. Electron.* **2013**, *59*, 811–817.
34. He, D.; Zhang, Y.; Chen, J. Cryptanalysis and improvement of an anonymous authentication protocol for wireless access networks. *Wirel. Pers. Commun.* **2014**, *74*, 229–243.
35. He, D.; Kumar, N.; Lee, J.; Sherratt, R. Enhanced three-factor security protocol for USB mass storage devices. *IEEE Trans. Consum. Electron.* **2014**, *60*, 30–37.