Journal of King Saud University – Computer and Information Sciences (2016) **28**, 68–81

King Saud University

**Journal of King Saud University – Computer and Information Sciences**

www.ksu.edu.sa
www.sciencedirect.com

# A secure effective dynamic group password-based authenticated key agreement scheme for the integrated EPR information system

**Vanga Odelu** [a,b,*], **Ashok Kumar Das** [c], **Adrijit Goswami** [a]

[a] *Department of Mathematics, Indian Institute of Technology, Kharagpur 721 302, India*
[b] *Department of Mathematics, Rajiv Gandhi University of Knowledge Technologies, Hyderabad 500 032, India*
[c] *Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India*

Received 24 July 2013; revised 10 March 2014; accepted 15 April 2014
Available online 7 November 2015

**KEYWORDS**

Cryptanalysis;
Integrated EPR information system;
Dynamic group;
Password;
Authentication;
Security

**Abstract** With the rapid growth of the Internet, a lot of electronic patient records (EPRs) have been developed for e-medicine systems. The security and privacy issues of EPRs are important for the patients in order to understand how the hospitals control the use of their personal information, such as name, address, e-mail, medical records, etc. of a particular patient. Recently, Lee et al. proposed a simple group password-based authenticated key agreement protocol for the integrated EPR information system (SGPAKE). However, in this paper, we show that Lee et al.'s protocol is vulnerable to the off-line weak password guessing attack and as a result, their scheme does not provide users' privacy. To withstand this security weakness found in Lee et al.'s scheme, we aim to propose an effective dynamic group password-based authenticated key exchange scheme for the integrated EPR information system, which retains the original merits of Lee et al.'s scheme. Through the informal and formal security analysis, we show that our scheme provides users' privacy, perfect forward security and known-key security, and also protects online and offline password guessing attacks. Furthermore, our scheme efficiently supports the dynamic group password-based authenticated key agreement for the integrated EPR information system. In addition, we simulate our scheme for the formal security verification using the widely-accepted AVISPA

* Corresponding author at: Department of Mathematics, Indian Institute of Technology, Kharagpur 721 302, India.
E-mail addresses: odelu.vanga@gmail.com, odelu.phd@maths.iitkgp.ernet.in (V. Odelu), iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in (A.K. Das), goswami@maths.iitkgp.ernet.in (A. Goswami).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.04.008
1319-1578 

(Automated Validation of Internet Security Protocols and Applications) tool and show that our scheme is secure against passive and active attacks.



## 1. Introduction

In an integrated EPR (electronic patient record) information system of all the patients, the medical institutions and the academia with most of the patients' information in details for them, can make the corrective decisions and clinical decisions in order to maintain and analyze patients' health. In such systems, the illegal access needs to be avoided as well as the information from theft during transmission over the insecure Internet needs to be prevented.

A dynamic group key agreement protocol provides the mechanisms to process member addition and deletion. Several dynamic group key agreement protocols have been proposed in the literature. We can divide the group key agreement protocols into two categories (Lee et al., 2013). The first one is the group key agreement protocols with public key. For example, key agreement protocols proposed by Tzeng and Tzeng (2000), Tzeng (2002), Boyd and Nieto (2003), Kim et al. (2004), Lee et al. (2006), and Jeong and Lee (2007) employ the public key infrastructure (PKI) and provide higher security. However, they are required to maintain the complex and heavy public key systems and users must hold extra storage for keeping public/private key pairs. The second one is the group password-based key agreement protocols (GPKE) without public key. For example, key agreement protocols of Lee et al. (2004), Abdalla et al. (2006), and Dutta and Barua (2006) provide the same password to all communicating parties. That is, each user does not have his/her own private password, and thus, the user cannot have his/her privacy. However, Zhang et al. (2012) showed that Dutta and Barua's scheme (Dutta and Barua, 2006) is insecure, where their scheme does not satisfy the key independence property (Steiner et al., 2000) and any two malicious users whose logic indexes are not adjacent in the former execution of the protocol may mount a replay attack in new protocol executions. Hence, these password-based approaches are not much suitable for many practical scenarios (Lee et al., 2013).

Boyd and Nieto (2003) described the first conference key agreement protocol, which can be completed in a single round. However, their scheme lacks forward secrecy property. By the forward secrecy property, we mean that when a node (user) leaves the network, it must not read any future messages after its departure. Kim et al. (2004) proposed an efficient and secure constant-round authenticated key agreement protocol (AGKE) for dynamic groups in the random oracle model. Dutta and Barua (2006) proposed a variant of Kim et al.'s scheme (Kim et al., 2004). Dutta–Barua's scheme makes use of the ideal-cipher model, instead of a simple mask, and they claimed that their scheme is secure against dictionary attacks. Unfortunately, their scheme contains another source of redundancy that can be exploited by an attacker (Abdalla et al., 2006). In 2006, Abdalla et al. (2006) proposed the first provably-secure password-based constant-round group key exchange protocol. It is provably-secure in the random-oracle and ideal-cipher models, which makes use of the decisional Diffie–Hellman problem assumption.

Recently, Lee et al. (2013) have proposed a simple group password-based authenticated key protocol without the server's public key, called the SGPAKE protocol, for the integrated EPR information system. Their scheme is based on Abdalla and Pointcheval's scheme (Abdalla and Pointcheval, 2005). Lee et al.'s SGPAKE protocol does not use any long-term key or public-key system. Lee et al. (2013) claimed that SGPAKE protocol provides each user a unique private weak password and resists password-guessing attack, and thus their scheme provides user privacy and data privacy. However, in this paper, we show that any user $U_i$ in a group $S_n$ can derive the private password of the user $U_{i-1}$ by setting the off-line password guessing attack, so that it does not provide the user's privacy. We aim to propose an improvement on Lee et al.'s SGPAKE protocol while retaining the original merits of Lee et al.'s scheme. Through the formal and informal security analysis, we show that our improved scheme provides user's privacy and perfect forward security, and also resists the offline password guessing attack.

The remainder of this paper is organized as follows. In Section 2, we provide the properties of the one-way hash function, discrete logarithm problem and group Diffie–Hellman problem. In Sections 3 and 4, we review Lee et al.'s SGPAKE protocol and then discuss the security flaws of Lee et al.'s SGPAKE protocol, respectively. We explain our improved scheme in Section 5. In Section 6, we provide the security of our improved scheme. Through the informal and formal security analysis, we show that our improved scheme is provably secure against an adversary for protecting the user's privacy and perfect forward security. In Section 7, we simulate our scheme for the formal security verification using the widely-accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool and show that our scheme is secure. In Section 8, we compare the performances of our scheme with other related existing schemes. Finally, we conclude the paper in Section 9.

## 2. Mathematical preliminaries

In this section, we discuss the properties of the one-way hash function, discrete logarithm problem and group Diffie–Hellman problem, which are useful for describing Lee et al.'s SGPAKE protocol (Lee et al., 2013) and its security analysis as well as our improved scheme.

### 2.1. One-way hash function

A one-way collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$ is a deterministic algorithm (Sarkar, 2010; Stinson, 2006) that takes an input as an arbitrary length binary string $x \in \{0,1\}^*$ and outputs a binary string

$h(x) \in \{0,1\}^n$ of fixed-length $n$. The formalization of an adversary $\mathcal{A}$'s advantage in finding collision is given as follows

$$Adv_{\mathcal{A}}^{HASH}(t) = Pr[(x, x') \Leftarrow_R \mathcal{A} : x = x' \quad \text{and} \quad h(x) = h(x')],$$

where $Pr[E]$ denotes the probability of an event $E$ in a random experiment, and $(x, x') \Leftarrow_R \mathcal{A}$ denotes the pair $(x, x')$ is selected randomly by $\mathcal{A}$. In this case, the adversary $\mathcal{A}$ is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by the adversary $\mathcal{A}$ with the execution time $t$. The hash function $h(\cdot)$ is said to be collision-resistant if $Adv_{\mathcal{A}}^{HASH}(t) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$.

An example of a secure one-way function is SHA-1 (Secure Hash Standard, 2010). One of the fundamental properties of a secure one-way hash function is that its outputs are very sensitive to small perturbations in inputs (Das, 2011). Recently proposed hash algorithm, Quark (Aumasson et al., 2010) is an efficient hash function than SHA-1. However, at present, the National Institute of Standards and Technology (NIST) does not recommend SHA-1 for top secret documents anymore. In 2011, Manuel showed that SHA-1 is insecure against collision attacks (Manuel, 2011). In this paper, as in Das and Goswami (2013) and Das et al. (2013), we can use SHA-2 as the secure one-way hash function for achieving top security. However, we use only 160-bits from the hash digest output of SHA-2 in Lee et al.'s SGPAKE scheme and our improved scheme.

### 2.2. Discrete logarithm problem (Das, 2013; Das et al., 2012)

Given an element $g$ in a finite group $G$ whose order is $n$, that is, $n = \#G_g$ ($G_g$ is the subgroup of $G$ generated by $g$) and another element $y$ in $G_g$. The problem is to find the smallest non-negative integer $x$ such that $g^x = y$. In this problem, known as the discrete logarithm problem (DLP), it is relatively easy to calculate discrete exponentiation $y = g^x (\text{mod } p)$ given $g$, $x$ and n using the repeated square-and-multiply algorithm (Stallings, 2003), but it is computationally infeasible to determine $x$ given $y$, $g$ and $n$, when $n$ is large. The formal definition of DLP is given in Definition 1 (Section 6.3).

### 2.3. Group Diffie–Hellman problem (Bresson et al., 2003)

Let $G$ be a cyclic group, whose order be a prime $n$, that is, $\#G = n$. Let $g$ be a generator of $G$. For a given set of values $g^{\Pi x_i}$ for some choice of $x_i$ from the set $\{1, 2, \ldots, n\}$, computing the common group Diffie–Hellman secret $g^{x_1 x_2 \ldots x_n}$ is computationally infeasible, when $n$ is large. The formal definition of the group Diffie–Hellman problem can be found in Bresson et al. (2003).

## 3. Review of Lee et al.'s SGPAKE protocol

In this section, we briefly review the recently proposed Lee et al.'s SGPAKE protocol (Lee et al., 2013) in order to show the cryptanalysis on their scheme.

Lee et al.'s scheme consists of three phases, namely, the user registration phase, authenticated key exchange phase and password change phase. Each user needs to remember his/her weak private password, which is shared with a trusted server $S$. For describing Lee et al.'s scheme, we use the notations listed in Table 1. Assume that $U_i$, $(i = 1, 2, \ldots, n)$ are $n$ communicating parties, $S$ is the trusted integrated EPR information system server. $G_g$ is a multiplicative group generated by the generator $g, p$ is a large prime, and $M, N$ are large numbers, which are made public. $Z_p^* = \{1, 2, 3, \ldots, p-1\}$ is the set of all positive integers less than $p$ and relatively prime to $p$. In other words, $Z_p^* = \{a | 0 < a < n, \gcd(a, n) = 1\}$, where $\gcd(x, y)$ denotes the greatest common divisor (gcd) of two integers $x$ and $y$, and can be calculated efficiently using the repeated applications of the Euclid's division algorithm (Stallings, 2003).

### 3.1. User registration phase

In the user registration phase, a user needs to register to the server $S$ before accessing the services from $S$. For registering, each new user $U_i$ needs to select his/her chosen random nonce $x_i$. After that the user $U_i$ sends this nonce $x_i$ to the server $S$ via a secure channel.

### 3.2. Authenticated key exchange phase

This phase consists of the following steps::

**Step 1.** $U_i \to S : m_1 = \{g^{x_i} M^{pw_i}\}$

- User $U_i$ chooses a private weak password $pw_i$, which is shared with the trusted server $S$.
- User $U_i$ computes $g^{x_i}$ and $g^{x_i} M^{pw_i}$, using the selected random nonce $x_i$, chosen password $pw_i$ and public key $M$ of $U_i$ in the user registration phase.
- User $U_i$ then sends the message $\langle m_1 \rangle$ to the server $S$ via a public channel, where $m_1 = \{g^{x_i} M^{pw_i}\}$.

**Step 2.** $S \to U_i : m_2 = \{g^{y_i} N^{pw_i}, E_{K_i}[S_n, K_{i-1}, K_i']\}$

- After receiving the message $\langle m_1 \rangle$ from the user $U_i$ in Step 1, the server $S$ chooses a random nonce $y_i$.
- Server $S$ then computes $g^{y_i}$, $g^{y_i} N^{pw_i}$, $g^{x_i} = \frac{g^{x_i} M^{pw_i}}{M^{pw_i}}$, $K_{i-1} \equiv (g^{x_{i-1}})^{y_{i-1}}$ (mod $p$) and $K_i' \equiv (g^{x_{i+1}})^{y_i}$ (mod $p$), for $i = 1, 2, \ldots, n$. Note that $N$ is the public key of $S$.
- After that the server $S$ encrypts the information $(S_n, K_{i-1}, K_i')$ with the key $K_i$ as $E_{K_i}[S_n, K_{i-1}, K_i']$, where $K_i = (g^{x_i})^{y_i} (\text{mod } p)$.
- Finally, the server $S$ sends the message $\langle m_2 \rangle$ to the user $U_i$ via a public channel, where $m_2 = \{g^{y_i} N^{pw_i}, E_{K_i}[S_n, K_{i-1}, K_i']\}$.

**Step 3.** $U_i \to^* : m_3 = \{X_i\}$

- After receiving the message $\langle m_2 \rangle$ in Step 2 from the server $S$, the user $U_i$ computes the key $K_i = \left(\frac{g^{y_i} N^{pw_i}}{N^{pw_i}}\right)^{x_i} = g^{x_i y_i}$ (mod $p$), and then obtains the information $K_{i-1}$ and $K_i'$ by decrypting $E_{K_i}[S_n, K_{i-1}, K_i']$ with key $K_i$ as $(S_n, K_{i-1}, K_i') = D_{K_i}[E_{K_i}[S_n, K_{i-1}, K_i']]$.
- If the user $U_i$ successfully verifies $E_{K_i}[S_n, K_{i-1}, K_i']$, he/she computes $Z_i = (K_{i-1})^{x_i} \equiv g^{x_{i-1} y_{i-1} x_i}$ (mod $p$), $Z_{i+1} = (K_i')^{x_i} \equiv g^{x_i y_i x_{i+1}}$ (mod $p$) and $X_i = \frac{Z_{i+1}}{Z_i} = \frac{g^{x_i y_i x_{i+1}}}{g^{x_{i-1} y_{i-1} x_i}}$.

**Table 1** Notations used in this paper.

| Symbol | Description |
|---|---|
| $S$ | Trusted integrated EPR information system server |
| $U_i$ | A communicating user |
| $ID_i$ | Identity of the user $U_i$ |
| $pw_i$ | The private password of the user $U_i$ |
| $ek_i$ | Symmetric encryption key of the user $U_i$ |
| $S_n$ | A dynamic group of $n$ members |
| $H(\cdot)$ | Secure one-way collision-resistant hash function |
| $p$ | A large prime |
| $g$ | A generator in group $Z_p^*$ |
| $M, N$ | Public keys |
| $E_K(\cdot)/D_K(\cdot)$ | Symmetric encryption/decryption using the key $K$ |
| $A \rightarrow B : X$ | Entity $A$ sends a message $X$ to entity $B$ via a public channel |
| $C_1, C_2$ | Data $C_1$ is concatenated with data $C_2$ |

- Finally, the user $U_i$ broadcasts the message $\langle m_3 \rangle$, where $m_3 = \{X_i\}$, to its group members in $S_n$ via a public channel.

**Step 4.** $U_i \rightarrow^* : m_4 = \{Auth_{i1}, Auth_{i2}\}$

- After receiving the message $\langle m_3 \rangle$ in Step 3, the user $U_i$ computes the secret value $sk_i = Z_i^n \times X_{i+1}^{n-1} \times X_{i+2}^{n-2} \times \cdots \times X_1^{i+n-1} = g^{x_1y_1x_2+x_2y_2x_3+\cdots+x_ny_nx_1} \pmod p$.
- $U_i$ computes the key conformation signatures $(Auth_{i1}, Auth_{i2})$, where $Auth_{i1} = H(S_n, sk_i, U_i)$ and $Auth_{i2} = H(S_n, K_i')$.
- $U_i$ then broadcasts the message $\langle m_4 \rangle$, where $m_4 = \{Auth_{i1}, Auth_{i2}\}$, via a public channel.
- Finally, $U_i$ authenticates another user $U_j$ by verifying $Auth_{j1}$, for $i \neq j$, and computes the common session key $SK$ as $SK = H(S_n, sk_i)$. At the same time, the server $S$ also authenticates each user $U_i$ by verifying $Auth_{i2}$. As a result, by verifying $Auth_{i1}$ the mutual authentication among $n$ users in a group $S_n$ is achieved. On the other hand, by verifying $Auth_{i2}$ the server $S$ performs the mutual authentication between the server $S$ and $n$ users in the group $S_n$.

### *3.3. Password change phase*

If a legitimate user $U_i$ wants to change his/her password $pw_i$, he/she sends his/her identity $ID_i$, the old password $pw_i$ and the new password $pw_i'$ to the integrated EPR information system $S$ via a secure channel. The server $S$ then checks the validity of $ID_i$ and the old password $pw_i$. If these are valid, $S$ updates $pw_i$ with the new password $pw_i'$.

The summary of message exchanges during the user registration, authenticated key exchange and password change phases of Lee et al.'s scheme is shown in Table 2.

## 4. Cryptanalysis on Lee et al.'s SGPAKE protocol

In this section, we show that Lee et al.'s SGPAKE protocol is insecure against the offline password guessing attacks.

Lee et al. claimed that their scheme can provide each user with a private weak password and resist the password-guessing attacks. As a result, Lee et al.'s scheme should provide data privacy and user's privacy. However, we show that any user $U_i$ in a group $S_n$ can derive the password $pw_{i-1}$ of another user $U_{i-1}$ in that group $S_n$ through the off-line password-guessing attacks. Thus, we show that Lee et al.'s SGPAKE scheme does not provide the user's privacy. A user $U_i$, being an attacker in a group $S_n$, can obtain the password $pw_{i-1}$ of another user $U_{i-1}$ in that group $S_n$ using the following steps:

**Step 1.** The user $U_i$ computes the key $K_i$ as $K_i = \left(\frac{g^{y_i}N^{pw_i}}{N^{pw_i}}\right)^{x_i} = g^{x_iy_i} \pmod p$ and then obtains $K_{i-1}$ and $K_i'$ by decrypting $E_{K_i}[S_n, K_{i-1}, K_i']$ with key $K_i$ as $(S_n, K_{i-1}, K_i') = D_{K_i}[E_{K_i}[S_n, K_{i-1}, K_i']]$.

**Step 2.** The user $U_i$ obtains $K_{i-2}$ and $K_{i-1}'$ by decrypting $E_{K_{i-1}}[S_n, K_{i-2}, K_{i-1}']$ with the derived key $K_{i-1}$ in Step 1 and then verifies the validity of $K_{i-1}'$ using $Auth_{(i-1)2}$.

**Step 3.** The user $U_i$ computes $g^{y_{i-1}}$ as $g^{y_{i-1}} = (K_{i-1}')^{x_i^{-1}} \pmod p = (g^{x_iy_{i-1}})^{x_i^{-1}} \pmod p$, where $x_i^{-1} \pmod p$ is computed efficiently using the extended Euclid's algorithm (Stallings, 2003).

**Step 4.** The user $U_i$ then computes $N^{pw_{i-1}}$ as $N^{pw_{i-1}} = \frac{g^{y_{i-1}}N^{pw_{i-1}}}{g^{y_{i-1}}} \pmod p$ from the message $\langle m_2 \rangle$ of the user $U_{i-1}$.

**Step 5.** Note that the user $U_{i-1}$'s private password $pw_{i-1}$ is a weak password. A user $U_i$ can set up the off-line password-guessing attack to correctly obtain the private password $pw_{i-1}'$ of the user $U_{i-1}$ such that $N^{pw_{i-1}'} = N^{pw_{i-1}}$ iterating all possible choices of $pw_{i-1}$. This attack has the following steps:

Step 5.1. $U_i$ selects a guessed password $pw_{i-1}'$ for the user $U_{i-1}$.

Step 5.2. Knowing the public information $N$ of the server $S$, $U_i$ computes the value $N^{pw_{i-1}'}$.

Step 5.3. $U_i$ compares the computed value $N^{pw_{i-1}'}$ with the derived value $N^{pw_{i-1}}$.

Step 5.4. If there is a match in Step 5.3, it indicates that the correct guess of the user $U_{i-1}$'s password $pw_{i-1}$. Otherwise, $U_i$ repeats from Step 5.1.

As a result, the user $U_i$, being an insider attacker, can succeed to guess the low-entropy password $pw_{i-1}$ of the user $U_{i-1}$ in his/her own group $S_n$.

**Table 2** Summary of message exchanges during the user registration, authenticated key exchange and password change phases of Lee et al.'s scheme (Lee et al., 2013).

**Registration phase**
$U_i \rightarrow S : pw_i, x_i$
(via a secure channel)

**Authenticated key exchange phase**
$U_i \rightarrow S : m_1 = \{g^{x_i}M^{pw_i}\}$
$S \rightarrow U_i : m_2 = \{g^{y_i}N^{pw_i}, E_{K_i}[S_n, K_{i-1}, K_i']\}$
$U_i \rightarrow^* : m_3 = \{X_i\}$
$U_i \rightarrow^* : m_4 = \{Auth_{i1}, Auth_{i2}\}$

**Password change phase**
$U_i \rightarrow S : ID_i, pw_i, pw_i'$
(via a secure channel)

## 5. The improved scheme

In this section, we first describe the main motivation behind our improved scheme. We then give a threat model under which our scheme is analyzed. Finally, we discuss the various phases related to our improved scheme.

### 5.1. Motivation

We have shown that any user $U_i$ in a group $S_n$ can derive the password $pw_{i-1}$ of another user $U_{i-1}$ in that group, $S_n$ through the off-line password-guessing attack. Thus, Lee et al.'s SGPAKE scheme does not provide the user's privacy. Hence, we feel that there is a need to propose an improvement of Lee et al.'s scheme to withstand the security flaw found in Lee et al.'s SGPAKE scheme, while retaining the original merits of Lee et al.'s SGPAKE scheme. Through the formal and informal security analysis, we show that our improved scheme provides user's privacy and perfect forward security, and resists the offline password guessing attack. We also show that our scheme is efficient as compared to Lee et al.'s SGPAKE scheme and other related existing schemes.

### 5.2. Threat model

As in Das and Goswami (2013), we use the Dolev–Yao threat model (Dolev and Yao, 1983) in our improved scheme in which two communicating parties communicate over an insecure channel. Any adversary (attacker or intruder) can then eavesdrop the transmitted messages over the public insecure channel and he/she can modify, delete or change the contents of the transmitted messages. We adopt the similar threat model for our scheme, since the channel is insecure and the end-points (users and server) cannot in general be trustworthy.

### 5.3. Description of our improved scheme

Our scheme consists of three phases, namely, the user registration phase, the authenticated key exchange phase and the password change phase. As in Lee et al.'s protocol, each user also needs to only remember his/her weak password shared with a trusted server $S$. However, even if the password of a user is weak, our scheme resists the offline password guessing attack as compared to Lee et al.'s scheme. For describing our scheme, we also use the notations listed in Table 1. We assume that $U_i$, $(i = 1, 2, \ldots, n)$, are $n$ communicating parties and $S$ is the trusted integrated EPR information system server. $G_g$ is a multiplicative group generated by the generator $g, p$ is a large prime so that it is intractable to a discrete logarithm, and $M, N$ are large public keys, which are made public, and $Z_p^* = \{1, 2, 3, \ldots, p-1\}$.

We describe the user registration phase, the authenticated key exchange phase and the password change phase in detail in the following subsections.

#### 5.3.1. User registration phase

As in Lee et al.'s scheme, in our proposed improved scheme a user $U_i$ first needs to register to the trusted server $S$ before accessing the services from $S$. For registering, each new user $U_i$ needs to generate his/her chosen random nonce $x_i$, choose a private password $pw_i$ and select an identity $ID_i$. Then the user $U_i$ sends the registration request message $\langle ID_i, x_i, pw_i \rangle$ to the server $S$ via a secure channel.

#### 5.3.2. Authenticated key exchange phase

Our authenticated key exchange phase consists of the following steps:

**Step 1.** $U_i \to S : m_1 = \{g^{x_i} M^{pw_i}\}$

- Step 1.1. Each user $U_i$ chooses a random nonce $x_i$.
- Step 1.2. $U_i$ computes $g^{x_i}$ and $g^{x_i} M^{pw_i}$, using the generator $g$ and public key $M$ of the user $U_i$.
- Step 1.3. $U_i$ then sends the message $\langle m_1 \rangle$, where $m_1 = \{g^{x_i} M^{pw_i}\}$, to the trusted integrated EPR information system server $S$ via a public channel.

**Step 2.** $S \to U_i : m_2 = \{g^{y_i} N^{pw_i}, E_{ek_i}[S_n, K_{i-1}, K_i']\}$

- Step 2.1. $S$ chooses a random nonce $y_i$.
- Step 2.2. $S$ computes $g^{y_i}, g^{y_i} N^{pw_i}, g^{x_i} = \frac{g^{x_i} M^{pw_i}}{M^{pw_i}}, K_{i-1} \equiv (g^{x_{i-1}})^{y_{i-1}^2} \pmod p$, and $K_i' \equiv (g^{x_{i+1}})^{y_i^2} \pmod p$, using the password $pw_i$ of the user $U_i$, which is already sent securely to the server $S$ by the user $U_i$ during the registration phase, and $N$ the public key of the server $S$.
- Step 2.3. $S$ then computes encryption key $ek_i \equiv (g^{x_i})^{y_i} \pmod p$, for $i = 1, 2, \ldots, n$. $S$ encrypts the information $(S_n, K_{i-1}, K_i')$ using the computed encryption key $ek_i$.
- Step 2.4. Finally, $S$ sends the message $\langle m_2 \rangle$, where $m_2 = \{g^{y_i} N^{pw_i}, E_{ek_i}[S_n, K_{i-1}, K_i']\}$, to the user $U_i$ via a public channel.

**Step 3.** $U_i \to^* : m_3 = \{X_i\}$

- Step 3.1. $U_i$ computes the encryption key $ek_i = \left(\frac{g^{y_i} N^{pw_i}}{N^{pw_i}}\right)^{x_i} = g^{x_i y_i} \pmod p$, and obtains $K_{i-1}$ and $K_i'$ by decrypting $E_{ek_i}[S_n, K_{i-1}, K_i']$ with the computed key $ek_i$.
- Step 3.2. If $U_i$ successfully verifies $E_{ek_i}[S_n, K_{i-1}, K_i']$, he/she then computes $Z_i = (K_{i-1})^{x_i} \equiv g^{x_{i-1} x_i y_{i-1}^2} \pmod p$, $Z_{i+1} = (K_i')^{x_i} \equiv g^{x_i x_{i+1} y_i^2} \pmod p$, and $X_i = \frac{Z_{i+1}}{Z_i} = \frac{g^{x_i x_{i+1} y_i^2}}{g^{x_{i-1} x_i y_{i-1}^2}} \pmod p$.
- Step 3.3. Finally, $U_i$ broadcasts the message $\langle m_3 \rangle$, where $m_3 = \{X_i\}$.

**Step 4.** $U_i \to^* : m_4 = \{Auth_{i1}, Auth_{i2}\}$

- Step 4.1. After receiving the message $\langle m_3 \rangle$ in Step 3, $U_i$ computes the secret value $sk_i$ as

$$sk_i = Z_i^n \times X_i^{n-1} \times X_{i+1}^{n-2} \times \cdots \times X_{i-2}^1$$
$$= g^{x_1 y_1^2 x_2 + x_2 y_2^2 x_3 + \cdots + x_n y_n^2 x_1} \pmod p.$$

- Step 4.2. $U_i$ computes the key conformations $(Auth_{i1}, Auth_{i2})$, where $Auth_{i1} = H(S_n, sk_i, U_i)$ and $Auth_{i2} = H(S_n, K_i')$.
- Step 4.3. $U_i$ broadcasts the message $\langle m_4 \rangle$, where $m_4 = \{Auth_{i1}, Auth_{i2}\}$.
- Step 4.4. Finally, the user $U_i$ authenticates another user $U_j$ in a group $S_n$ by verifying $Auth_{j1}$, for $i \neq j$, and also computes the common session key $SK$ as $SK = H(S_n, sk_i)$. At the same time, the server $S$ authenticates each user $U_i$ by verifying $Auth_{i2}$.

**Table 3** Summary of the user registration and authenticated key exchange phases of our scheme.

| User $U_i$ | Server $S$ |
|---|---|
| **Registration phase** | |
| Generates random nonce $x_i$, selects identity $ID_i$ and password $pw_i$. | |
| $\xrightarrow{\langle ID_i, x_i, pw_i \rangle}$ (via a secure channel) | |
| **Authenticated key exchange phase** | |
| Computes $m_1 = g^{x_i} M^{pw_i}$. | |
| $\xrightarrow{\langle m_1 \rangle}$ | |
| | Chooses a random nonce $y_i$, computes $g^{y_i} N^{pw_i}$, $g^{x_i} = \frac{g^{x_i} M^{pw_i}}{M^{pw_i}}$, $K_{i-1} \equiv (g^{x_{i-1}})^{y_{i-1}^2} (\text{mod } p)$, and $K'_i \equiv (g^{x_{i+1}})^{y_i^2} (\text{mod } p)$. Computes key $ek_i \equiv (g^{x_i})^{y_i}$ $(\text{mod } p)$, for $i = 1, 2, \ldots, n$, and encrypts $(S_n, K_{i-1}, K'_i)$ using $ek_i$. |
| | $\xleftarrow{\langle m_2 = \{g^{y_i} N^{pw_i}, E_{ek_i}[S_n, K_{i-1}, K'_i]\} \rangle}$ |
| Computes key $ek_i = \left(\frac{g^{y_i} N^{pw_i}}{N^{pw_i}}\right)^{x_i} = g^{x_i y_i}$ $(\text{mod } p)$, retrieves $[S_n, K_{i-1}, K'_i] = D_{ek_i}[E_{ek_i}[S_n, K_{i-1}, K'_i]]$. Verifies $E_{ek_i}[S_n, K_{i-1}, K'_i]$. If it is successful, computes $Z_i = (K_{i-1})^{x_i} \equiv g^{x_{i-1} x_i y_{i-1}^2} (\text{mod } p)$, $Z_{i+1} = (K'_i)^{x_i} \equiv g^{x_i x_{i+1} y_i^2}$ $(\text{mod } p)$, and $X_i = \frac{Z_{i+1}}{Z_i} = \frac{g^{x_i x_{i+1} y_i^2}}{g^{x_{i-1} x_i y_{i-1}^2}}$ $(\text{mod } p)$. | |
| Broadcasts the message $\langle m_3 = \{X_i\} \rangle$ | |
| Computes $sk_i = Z_i^n \times X_i^{n-1} \times \cdots \times X_{i-2}^1$ $= g^{x_1 y_1^2 x_2 + x_2 y_2^2 x_3 + \cdots + x_n y_n^2 x_1}$ $(\text{mod } p)$. | |
| Computes $Auth_{i1} = H(S_n, sk_i, U_i)$ and $Auth_{i2} = H(S_n, K'_i)$. | |
| Broadcasts the message $\langle m_4 = \{Auth_{i1}, Auth_{i2}\} \rangle$. | |
| Verifies $Auth_{j1}$, for $i \neq j$ for each user $U_j$. | |
| | Verifies $Auth_{i2}$ for each user $U_i$. |
| Computes common session key $SK = H(S_n, sk_i)$. | |

Note that by verifying $Auth_{i1}$, the mutual authentication among $n$ users in a group $S_n$ is achieved. On the other hand, by verifying $Auth_{i2}$, the mutual authentication between the server $S$ and $n$ users in the group, $S_n$ is also achieved.

#### *5.3.3. Password change phase*

If a legitimate user $U_i$ wants to change his/her password $pw_i$, he/she needs to send his/her identity $ID_i$, the old password $pw_i$ and the new password $pw'_i$ to the integrated EPR information system $S$ via a secure channel. After receiving the password change request, the server $S$ verifies the pair $(ID_i, pw_i)$. If it is a valid pair, the server $S$ then replaces the old password $pw_i$ with the new password $pw'_i$ in its database.

The summary of our scheme is given in Table 3.

## 6. Analysis of our proposed scheme

In this section, we first show that all users in a group $S_n$ have the same secret key. After that through the informal and formal security analysis, we show that our scheme is secure against different attacks.

### *6.1. Correctness of the protocol*

The correctness proof of our scheme is given in Theorem 1.

**Theorem 1.** *In our improved scheme, all users $U_i$ $(i = 1, 2, \ldots, n)$ in a group $S_n$ have the same secret session key $SK$.*

**Proof.** Let $U_i$ be a user in a group $S_n$ consisting of $n$ members. The user $U_i$ in a group $S_n$ computes the secret value $sk_i$ as

$$
\begin{aligned}
sk_i &= Z_i^n \times X_i^{n-1} \times X_{i+1}^{n-2} \times \cdots \times X_{i-2}^1 \\
&= Z_i^n \times (Z_{i+1}/Z_i)^{n-1} \times (Z_{i+2}/Z_{i+1})^{n-2} \times \cdots \times (Z_{i-1}/Z_{i-2})^1 \\
&= \prod_{i=1}^{n} Z_i \\
&= g^{x_1 y_1^2 x_2 + x_2 y_2^2 x_3 + \cdots + x_n y_n^2 x_1} \ (\text{mod } p).
\end{aligned}
$$

As a result, all users $U_i$ $(i = 1, 2, \ldots, n)$ in the group $S_n$ have the same secret session key $SK$ as $SK = H(S_n, sk_i)$. $\square$

### 6.2. Informal security analysis

In this section, we show that our scheme is secure against various known attacks. For security analysis, we use the threat model described in Section 5.2.

#### 6.2.1. Session key security

In our scheme, no adversary can compute $Z_i(= g^{x_{i-1}x_i y_{i-1}^2} \pmod p)$ even if he/she has the knowledge of $g^{x_{i-1}}, g^{x_i}$ and $g^{y_{i-1}^2}$. This problem is computationally difficult due to difficulty of solving the group Diffie–Hellman problem (Bresson et al., 2003). Moreover, the secret value $sk_i$ depends on the one-time random secrets $x_i$ and $y_i$ $(i = 1, 2, \ldots, n)$. Thus, computing the secret session key $SK = H(S_n, sk_i)$ without computing $Z_i$ is a computationally infeasible for the adversary. Therefore, our scheme provides the session key security.

#### 6.2.2. Mutual authentication

As in Lee et al.'s scheme (Lee et al., 2013), each legal user $U_i$ can decrypt the information in message $\langle m_2 \rangle$ using its own password $pw_i$, and then compute the one-time encryption key $ek_i$. Additionally, each user $U_i$ can authenticate the server $S$ by verifying the ciphertext $E_{ek_i}[S_n, K_{i-1}, K'_i]$, whereas the server $S$ can also authenticate each user $U_i$ by verifying $Auth_{i2}$. Furthermore, each user $U_i$ can authenticate the other users $U_j$'s $(i \neq j)$ by verifying $Auth_{j1}$ in a group $S_n$. Our scheme then achieves the mutual authentication between the users and the server, and also among all the users in a group $S_n$.

#### 6.2.3. Perfect forward security

By the forward security, we mean that when a node (user) leaves the network, it must not read any future messages after its departure. Forward secrecy thus ensures that the subsequent shared session keys cannot be derived even if an adversary knows the contiguous subset of old session keys. In our scheme, the session key $SK$ is the secret value $sk_i = g^{x_1 y_1^2 x_2 + x_2 y_2^2 x_3 + \cdots + x_n y_n^2 x_1} \pmod p$, which depends on only the one-time random secrets $x_i$ and $y_i$'s $(i = 1, 2, \ldots, n)$. Since the random secrets are not dependent on the private password $pw_i$ of a user $U_i$, no adversary can compute the previous session keys even if he/she knows private password $pw_i$ of that user $U_i$. Thus, our proposed scheme provides the perfect forward security to the established session key $SK$.

#### 6.2.4. Off-line password guessing attack

In the following, we explain how our improved scheme has the ability to resist the weaknesses of Lee et al.'s SGPAKE scheme. From the received message $\langle m_2 \rangle$ in Step 2 during our authenticated key exchange phase, a user $U_i$ in a group $S_n$ can compute the encryption key $ek_i = g^{x_i y_i} \pmod p$, and obtain $K_{i-1}$, and $K'_i$ by decrypting $E_{ek_i}[S_n, K_{i-1}, K'_i]$ with the computed encryption key $ek_i$, where $K_{i-1} \equiv (g^{x_{i-1}})^{y_{i-1}^2} \pmod p$ and $K'_i \equiv (g^{x_{i+1}})^{y_i^2} \pmod p$. Since $x_{i-1}$ (chosen by the user) and $y_{i-1}$ (chosen by the server) are random nonces corresponding to the user $U_{i-1}$, the user $U_i$ cannot compute $g^{x_{i-1}}$ and $g^{y_{i-1}}$ using the derived $K_{i-1}$ and $K'_i$, due to difficulty of solving discrete logarithm problem. The user $U_i$ needs to guess both private password, say $pw'_{i-1}$ and one-time random secret, say $x'_{i-1}$ of the user $U_{i-1}$ such that the condition $g^{x'_{i-1}} M^{pw'_{i-1}} \pmod p = g^{x_{i-1}} M^{pw_{i-1}} \pmod p$ holds. Then, the guessed pair $(pw'_{i-1}, x'_{i-1})$ may probably be the original pair. However, as pointed out in Das and Goswami (2013), the probability of guessing a correct password $pw'_{i-1}$ of composed $n$ characters is $\approx \frac{1}{2^{6n}}$ and the probability of guessing a correct random nonce $x_{i-1}$ of 1024-bit is $\approx \frac{1}{2^{1024}}$. Moreover, to guess a correct $pw'_{i-1}$, the attacker has to guess the correct $x'_{i-1}$, which is of 1024-bit. Thus, the probability of guessing the probably correct pair $(pw'_{i-1}, x'_{i-1})$ is $\approx \frac{1}{2^{6n+1024}}$. If $n = 10$, the success probability is approximately $\frac{1}{2^{60+1024}}$, which is negligible. The user $U_i$ can not compute $M^{pw_{i-1}}$ from message $\langle m_1 = \{g^{x_{i-1}} M^{pw_{i-1}}\} \rangle$ and $N^{pw_{i-1}}$ from $g^{y_{i-1}} N^{pw_{i-1}}$ without knowing $g^{x_{i-1}}$ and $g^{y_{i-1}}$, respectively. Therefore, the user $U_i$ has no way to guess the password correctly of other users in a group $S_n$ with the available public parameters, and hence, our scheme resists the off-line password-guessing attacks.

#### 6.2.5. Undetectable on-line password guessing attack

Suppose an adversary, distinguished as the user $U_i$, guesses a password, say $pw'_i$ and communicates with the server $S$. But in Step 4 of our authenticated key exchange phase, to compute the authentication parameters $Auth_{i1}(= H(S_n, sk_i, U_i))$ and $Auth_{i2}(= H(S_n, K'_i))$, the adversary needs to retrieve the original $K_{i-1}$ and $K'_i$ from the ciphertext $E_{ek_i}[S_n, K_{i-1}, K'_i]$. Computing $K_{i-1}$ and $K'_i$ without knowledge of the encryption key $ek_i(= g^{x_i y_i} \pmod p)$ and computing $Z_i$ (used in computing $sk_i$) without knowing the correct pair $(pw_i, x_i)$ are computationally infeasible tasks to the attacker. Thus, a failed password guessing will be detected by other users as well as the server. On other hand, suppose an adversary, distinguished as the server $S$, guesses a password, say $pw'_i$ and communicates with the users $U_i$'s in a group $S_n$. After receiving the message $\langle m_1 \rangle$ from the user $U_i$, the adversary needs to compute $g^{x_i} = \frac{g^{x_i} M^{pw'_i}}{M^{pw'_i}}$ using a guessed password $pw'_i$. Then, the adversary needs to compute $K_{i-1}$ and $K'_i(= (g^{x_{i+1}})^{y_i^2} \pmod p)$, $K_{i-1}(= (g^{x_{i-1}})^{y_{i-1}^2} \pmod p)$ and the encryption key $ek_i(= g^{x_i y_i} \pmod p)$. Moreover, the adversary has to compute the ciphertext $E_{ek_i}[S_n, K_{i-1}, K'_i]$ and send the message, say $m'_2(= \{g^{x_i} M^{pw'_i}, E_{ek_i}[S_n, K_{i-1}, K'_i]\})$ to the user $U_i$. Since the user $U_i$ can verify $\langle m'_2 \rangle$ in Step 3 of our authenticated key exchange phase, a failed password guessing will be detected by the user $U_i$. As a result, our scheme prevents the undetectable on-line password guessing attack.

#### 6.2.6. Data privacy and users' privacy

In our improved scheme, computing a session key is computationally infeasible task as described in Section 6.2.1. So, no adversary can decrypt the transmitted data without the common session key. Moreover, our scheme provides private password to each user in a group and also prevents the password guessing attacks. Therefore, our scheme provides data privacy as well as users' privacy, whereas Lee et al.'s scheme does not provide the users' privacy as their scheme is vulnerable to off-line password-guessing attacks.

#### 6.2.7. Known-key security

Each run of our authenticated key exchange phase between a specific user $U_i$ and the server $S$ produces a unique session secret key. This property ensures that when the protocol has known key security, the knowledge of previous session keys does not allow an adversary to compromise other previous session keys or future session keys (Ammayappan et al., 2011). Since the session key is different for different sessions and they are independent among each protocol execution, our improved scheme also exhibits the known-key security property.

### 6.3. Formal security analysis

We follow the formal security proof of our improved scheme as in Das (2013), Das et al. (2012) and Odelu et al. (2014). For this purpose, we first formally define the discrete logarithm problem (DLP). We present the formal security proof of our scheme only for users' privacy and perfect forward security in Theorem 2, whereas other security analyses are already provided in Section 6.2. For the formal security analysis, we follow the method using the random oracle model as used in Chatterjee et al. (2014), Das et al. (2013), Islam and Biswas (2013) and Islam and Biswas, 2014.

**Definition 1.** (Formal definition of discrete logarithm problem (Das, 2013)) Let $G$ be a cyclic group of order $q, g$ a generator of $G$, and $A_1$ an algorithm that return an integer in $Z_q$, where $Z_q = \{0, 1, \ldots, q-1\}$. Let $a \in_R T$ denote that the element $a$ is chosen randomly from the set $T$. Consider the following experiment, $EXP1^{DLP}_{G,g}(A_1)$ in Algorithm 1.

**Algorithm 1.**

$EXP1^{DLP}_{G,g}(A_1)$

$x \in_R Z_q$
$X \leftarrow g^x (\text{mod } q)$
$x' = A_1(X)$
**if** $g^{x'} = X \ (\text{mod } q)$ **then**
    **return** 1 (TRUE)
**else**
    **return** 0 (FALSE)
**end if**

The DLP advantage of $A_1$ is defined by $Adv^{DLP}_{G,g}(A_1) = Pr[EXP1^{DLP}_{G,g}(A_1) = 1]$, where $Pr[E]$ denotes the probability of an event $E$ in a random experiment. The discrete logarithm problem (DLP) is called a hard problem (computationally infeasible problem) in $G$ if the DLP-advantage of any adversary of reasonable resources is small. Here the resources are measured in terms of the time complexity of the adversary including its code size as usual. In other words, we call the DLP as a hard problem, if $Adv^{DLP}_{G,g}(A_1) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$.

**Theorem 2.** *Under the assumption of the discrete logarithm problem, our improved scheme is provably secure against an adversary for protecting users' privacy and perfect forward security.*

**Proof.** In this proof, we need to construct an adversary $\mathcal{A}$ from a dynamic group $S_n$ consisting of $n$ users, who can obtain the private password of the other members in that group $S_n$ using the available information including his/her own private password. In order to construct such an adversary $\mathcal{A}$, we consider the following random oracle:

- *Reveal*: This oracle unconditionally outputs the value $x \in Z_q$ from the given public value $g^x (\text{mod } q)$, where $g$ is the generator in the cyclic group $G$ of order $q$.

Assume that the adversary $\mathcal{A}$ is a user $U_i$ in the dynamic group $S_n$. The adversary $\mathcal{A}$ needs to run the experimental algorithm $EXP2^{DLP}_{IP,\mathcal{A}}$ for our improved protocol, say IP, given in Algorithm 2.

**Algorithm 2.**

$EXP2^{DLP}_{IP,A}$

$U_i$ (being an adversary $\mathcal{A}$) eavesdrops the message $m_2 = \{g^{y_i} N^{pw_i}, E_{ek_i}[S_n, K_{i-1}, K'_i]\}$ sent from the server $S$.
$U_i$ computes $g^{y_i} = \frac{g^{y_i} N^{pw_i}}{N^{pw_i}}$ using his/her private password $pw_i$ and public $N$.
$U_i$ obtains $K_{i-1}$ and $K'_i$ by decrypting $E_{ek_i}[S_n, K_{i-1}, K'_i]$ using the encryption key $ek_i$, where $ek_i = (g^{y_i})^{x_i} (\text{mod } p)$.
Call *Reveal* oracle on input $g^{y_i}$. Let $y'_i \leftarrow Reveal(g^{y_i})$.
$U_i$ computes $g^{x_{i+1}} = (K'_i)^{(y'_i)^{-2}} (\text{mod } p)$ and then computes $M^{pw_{i+1}} = \frac{g^{x_{i+1}} M^{pw_{i+1}}}{g^{x_{i+1}}}$.
Call *Reveal* oracle on input $M^{pw_{i+1}}$. Let $pw'_{i+1} \leftarrow Reveal(M^{pw_{i+1}})$.
**if** $M^{pw'_{i+1}} = M^{pw_{i+1}} (\text{mod } p)$ **then**
    **return** 1 (TRUE)
**else**
    **return** 0 (FALSE)
**end if**

We now define the success for $EXP2^{DLP}_{IP,\mathcal{A}}$ as $Succ^{DLP}_{IP,\mathcal{A}} = |Pr[EXP2^{DLP}_{IP,\mathcal{A}} = 1] - 1|$ and the advantage function for our improved protocol, IP due to this experiment as $Adv^{DLP}_{IP}(t, q_R) = max_{\mathcal{A}}\{Succ^{DLP}_{IP,\mathcal{A}}\}$, where the maximum is considered over all $\mathcal{A}$ with execution time $t$ and $q_R$ is the number of queries made to the *Reveal* oracle. Consider the experiment $EXP2^{DLP}_{IP,\mathcal{A}}$ for the adversary $\mathcal{A}$ for our improved scheme, IP. If $\mathcal{A}$ has the ability to solve DLP, the adversary wins the game, and thus, the adversary obviously can easily compute the nonce $y_i$ from the public message $g^{y_i} N^{pw_i}$ using his/her private password $pw_i$. Using the derived nonce $y_i$ and $K'_i$, he/she can compute the private password $pw_{i+1}$ of the user $U_{i+1}$. In this case, the adversary can derive the private passwords of all users in the dynamic group $S_n$. However, from Definition 1, DLP is a hard problem, that is, $Adv^{DLP}_{G,g}(A_1) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$. Hence, $Adv^{DLP}_{IP}(t, q_R) \leqslant \epsilon$, since $Adv^{DLP}_{IP}(t, q_R)$ depends on $Adv^{DLP}_{G,g}(A_1)$. As a result, there is no feasible way for the adversary to obtain the private password of any other user in the group $S_n$. Therefore, our proposed protocol provides the users' privacy.

A compromised password does not yield any previous session keys $SK$, because the session key $SK = H(S_n, sk_i)$, where $sk_i = g^{x_1 y_1^2 x_2 + x_2 y_2^2 x_3 + \cdots + x_n y_n^2 x_1} (\text{mod } p)$, depends on the temporarily chosen random nonces $x_1, x_2, \ldots, x_n$ and

$y_1, y_2, \ldots, y_n$, and it is independent of protocol executions. As a result, our improved scheme preserves the perfect security property. □

## 7. Simulation for formal security verification using AVISPA tool

In this section, we examine our scheme through simulation for the formal security verification using the widely-accepted AVISPA tool (Das, 2013; Das et al., 2013), and show that our scheme is secure against active attacks, such as replay and man-in-the-middle attacks.

AVISPA (Automated Validation of Internet Security Protocols and Applications) stands for a push-button tool for the automated validation of Internet security-sensitive protocols and applications (AVISPA, 2013). AVISPA consists of four different back-ends that implement a variety of state-of-the-art automatic analysis techniques, which are called the On-the-fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The protocols to be analyzed under the AVISPA tool require to specify them in a high-level language, called HLPSL (High Level Protocols Specification Language), which is a role-oriented language. The specification written in HLPSL is first translated into a low-level specification by a translator, called HLPSL2IF. This translator generates a specification in an intermediate format, called the Intermediate Format (IF). After that the output format (OF) of AVISPA is generated using one of the four back-ends: OFMC, CL-AtSe, STAMC and TA4SP. The analysis of the OF is made as follows. The first printed section called SUMMARY, indicates whether the protocol is safe, unsafe, or whether the analysis is inconclusive. DETAILS is the second section, which explains under what condition the protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive. Other remaining sections, called PROTOCOL, GOAL and BACKEND, represent the name of the protocol, the goal of the analysis and the name of the back-end used, respectively. Finally, at the end of the analysis, after some possible comments and the statistics, the trace of the attack (if any) is also printed in the usual Alice-Bob format. One can find more details on HLPSL in Advanced Encryption Standard (2010) and AVISPA (2013).

Some basic types available in HLPSL are (Advanced Encryption Standard, 2010; AVISPA, 2013):

- *agent:* Values of type *agent* represent principal names. The intruder is always assumed to have the special identifier *i*.
- *public_key:* These values represent agents' public keys in a public-key cryptosystem. For example, given a public (respectively private) key *pk*, its inverse private (respectively public) key is obtained by *inv_pk*.
- *symmetric_key:* Variables of this type represent keys for a symmetric-key cryptosystem.
- *text:* In HLPSL, *text* values are often used as nonces. These values can be used for messages. If *Na* is of type *text (fresh)*, then *Na'* will be a fresh value which the intruder cannot guess.
- *nat:* The *nat* type represents the natural numbers in non-message contexts.
- *const:* This type represents constants.
- *hash_func:* The base type *hash_func* represents cryptographic hash functions. The base type function also represents functions on the space of messages. It is assumed that the intruder cannot invert hash functions (in essence, that they are one-way).
- *bool:* Boolean values are useful for modeling, for instance, binary flags.

The space of legal messages is defined as the closure of the basic types. For a given message $M$ and encryption key $K$, we denote $\{M\}\_K$ as the symmetric/public-key encryption using the key $K$. For concatenations, HLPSL uses the associative "." operator. The "*played_by A*" declaration indicates that the agent named in variable $A$ plays in a particular role. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. The immediate reaction transitions have the form $X = | > Y$, which relate an event $X$ and an action $Y$. If a variable $V$ remains permanently secret, it is expressed by the goal *secrecy_of V*. If $V$ is ever obtained or derived by the intruder, a security violation will result. The intruder, always denoted by ($i$), will have the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. $witness(A, B, id, E)$ declares for a (weak) authentication property of $A$ by $B$ on $E$, declares that agent $A$ is witness for the information $E$; this goal will be identified by the constant *id* in the goal section (AVISPA, 2013). This expresses that the agent named in variable $B$ has freshly generated the value $E$ for the agent named in variable $A$. The *id* term is a new constant that identifies the message term upon which the goal should authenticate. $request(B, A, id, E)$ indicates for a strong authentication property of $A$ by $B$ on $E$, declares that agent $B$ requests a check of the value $E$; this goal will be identified by the constant *id* in the goal section (AVISPA, 2013). This formalizes $A$'s acceptance of the value $E$ as having generated for him/her by the agent named in $B$.

### 7.1. Specifying our scheme

The specification in HLPSL language for the role of the user $U_i$ in a group $S_n$ is shown in Fig. 1. At first, during the registration phase of our scheme, a user $U_i$ sends the registration message $\langle ID_i, x_i, pw_i \rangle$ to the server $S$ via a secure channel using the $Snd()$ operation. The channel declaration $channel(dy)$ means that the channel is insecure, which is based the Dolev–Yao threat model (as used in our threat model in Section 5.2) (Dolev and Yao, 1983). Note that secret(Xi, PWi, subs1, Ui) declares that both $x_i$ and $pw_i$ are kept secret to $U_i$ only using the protocol id, subs1. During the authenticated key exchange phase, $U_i$ sends the message $\langle m_1 = \{g^{x_i} M^{pw_i}\} \rangle$ via a public channel. After receiving the message $\langle m_2 = \{g^{y_i} N^{pw_i}, E_{ek_i}[S_n, K_{i-1}, K_i']\} \rangle$ from the server $S$ by the $Rcv()$ operation, $U_i$ broadcasts the messages $\langle m_3 = \{X_i\} \rangle$ and $\langle m_4 = \{Auth_{i1}, Auth_{i2}\} \rangle$. witness(Ui, S, alice_bob_xi, Xi) indicates that $U_i$ has freshly generated the value $x_i$ for the server $S$. By request(S, Ui, bob_alice_yi, Yi), $U_i$ accepts of the random nonce $y_i$ generated for $U_i$ by the server $S$. In a similar way, we have also implemented the specification in HLPSL language for the roles of the server $S$ and another user $U_j$ in a group $S_n$ in Figs. 2 and 3, respectively.

```
role useri (Ui, Uj, S   : agent,
          SKus : symmetric_key,
          % H is hash function
          H : hash_func,
          Snd, Rcv: channel(dy))
played_by Ui
def=
 local State  : nat,
      % F is the other function (such as
      % multiplication, squaring, etc.)
      F : hash_func,
      P, M, N, G : text,
      IDi, Sn, PWi, Xi, Yi, EKi: text,
      SKi, K1, K2, Xi1, Xi2, Yi1: text
 const alice_bob_xi,  bob_alice_yi,
      subs1, subs2, subs3, subs4 : protocol_id
init  State := 0
transition
% User registration phase
1. State = 0 ∧ Rcv(start) =|>
  State' := 1 ∧ Xi' := new()
% Send the registration request message
          ∧ Snd({IDi.Xi'.PWi}_SKus)
          ∧ secret({Xi', PWi}, subs1, Ui)
          ∧ secret({SKi}, subs3, {Ui,Uj,S})
% Authenticated key exchange phase
% Send the message <m1> to the server S via a
% public channel
2. State = 1 ∧ Snd(F(exp(G,Xi').exp(M,PWi)))
% Receive the message <m2> from server S via
% a public channel
          ∧ Rcv(F(exp(G,Yi').exp(N,PWi)).
            {Sn.exp(G,F(Xi1.Yi1)).
            exp(G,F(Xi2,Yi))}_(exp(exp(G,Xi'),Yi'))) =|>
% Send the message <m3>
  State' := 2 % Ui has freshly generated the value Xi' for S
           ∧ witness(Ui, S, alice_bob_xi, Xi')
           ∧ Snd(F(exp(G,F(Xi'.Xi2.Yi')).
               exp(G, F(Xi1.Xi'.Yi1))))
            ∧ secret({Yi'}, subs2, S)
% Send the message <m4>
           ∧ K2' := exp(G,F(Xi2,Yi'))
           ∧ Snd(H(Sn.SKi.Ui).H(Sn.K2'))
 % Ui's acceptance of the value Yi' generated for Ui by S
           ∧ request(S, Ui, bob_alice_yi, Yi')
end role
```

**Figure 1** Role specification in HLPSL for a user $U_i$ in a group $S_n$ of our scheme.

```
role  server (Ui, Uj, S   : agent,
          SKus : symmetric_key,
          % H is hash function
          H : hash_func,
          Snd, Rcv: channel(dy))
played_by S
def=
 local State  : nat,
      % F is the other function (such as
      % multiplication, squaring, etc.)
      F : hash_func,
      P, M, N, G : text,
      IDi, IDj, Sn, PWi, PWj, Xi, Xj, Yi, EKi: text,
      SKi, K1, K2, Xi1, Xi2, Yi1: text
 const alice_bob_xj,  bob_alice_yi,
      subs1, subs2, subs3, subs4 : protocol_id
init  State := 0
 transition
% User registration phase
% Receive the registration request messages from Ui and Uj
1. State  = 0 ∧ Rcv({IDi.Xi'.PWi}_SKus)  =|>
  State' := 1 ∧ secret({Xi', PWi}, subs1, Ui)
         ∧ secret({SKi}, subs3, {Ui,Uj,S})
2. State = 1 ∧ Rcv({IDj.Xj'.PWj}_SKus) =|>
  State' := 2 ∧ secret({Xj', PWj}, subs4, Uj)
% Authenticated key exchange phase
% Receive the message <m1> to Ui via a
% public channel
2. State = 2 ∧ Rcv(F(exp(G,Xi').exp(M,PWi))) =|>
% S has freshly generated the value Yi' for Ui
  State' := 3
3. State = 3  ∧ Rcv(F(exp(G,Xj').exp(M,PWj))) =|>
% Send the message <m2> to Ui via
% a public channel
  State' := 4 ∧ Yi' := new()
           ∧ Snd(F(exp(G,Yi').exp(N,PWi)).
           {Sn.exp(G,F(Xi1.Yi1)).
           exp(G,F(Xi2,Yi'))}_(exp(exp(G,Xi),Yi')))
% S has freshly generated the value Yi' for Ui
          ∧ witness(S, Ui, bob_alice_yi, Yi')
% Receive the message <m3> from Ui
4. State = 4 ∧ Rcv(F(exp(G,F(Xi'.Xi2.Yi')).
              exp(G, F(Xi1.Xi'.Yi1)))) =|>
  State' := 5 ∧ secret({Yi'}, subs2, S)
% Receive the message <m4>
5. State = 5 ∧ Rcv(H(Sn.SKi.Ui).H(Sn.exp(G,F(Xi2,Yi')))) =|>
 % S's acceptance of the value Xi' generated for S by Ui
 State' := 6 ∧ request(Ui, S, alice_bob_xi, Xi)
 % S's acceptance of the value Xj' generated for S by Uj
          ∧ request(Uj, S, alice_bob_xj, Xj)
end role
```

**Figure 2** Role specification in HLPSL for the server $S$ of our scheme.

We have then specified the roles for the session, and the goal and environment of our scheme in Figs. 3 and 4. In the session segment, all the basic roles: useri, userj and server are instanced with concrete arguments. The top-level role, called the environment, is always defined in the specification of HLPSL language, which contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session during the simulation. Goals are given in their own sections, which generally come at the end of a HLPSL specification. Goal is defined with the keyword *goal* and ends with *end goal*. Between the two, multiple security goals may be listed in HLPSL.

The following four secrecy goals and three authentications are verified in our scheme for formal security verification:

```
role userj (Ui, Uj, S   : agent,
          SKus : symmetric_key,
          % H is hash function
          H : hash_func,
          Snd, Rcv: channel(dy))
played_by Uj
def=
 local State  : nat,
      % F is the other function (such as
      % multiplication, squaring, etc.)
      F : hash_func,
      P, M, N, G : text,
      IDj, Sn, PWj, Xi, Xj, Yi, EKi: text,
      SKi, K1, K2, Xi1, Xi2, Yi1: text
 const alice_bob_xj,  bob_alice_yi,
      subs1, subs2, subs3, subs4 : protocol_id
init  State := 0
transition
% User registration phase
1. State = 0 /\ Rcv(start) =|>
  State' := 1 /\ Xj' := new()
% Send the registration request message
          /\ Snd({IDj.Xj'.PWj}_SKus)
          /\ secret({Xj', PWj}, subs4, Uj)
          /\ secret({SKi}, subs3, {Ui,Uj,S})
% Authenticated key exchange phase
% Send the message <m1> to the server S via a
% public channel
2. State = 1 /\ Snd(F(exp(G,Xj').exp(M,PWj)))
% Receive the message <m3>
           /\ Rcv(F(exp(G,F(Xi'.Xi2.Yi')).
                exp(G, F(Xi1.Xi'.Yi1)))) =|>
% Receive the message <m4>
State' := 2  % Uj has freshly generated the value Xj' for S
           /\ witness(Uj, S, alice_bob_xj, Xj')
3. State = 2 /\ Rcv(H(Sn.SKi.Ui).H(Sn.exp(G,F(Xi2,Yi')))) =|>
 % Uj's acceptance of the value Yi' generated for Uj by S
State' := 3  /\ request(S, Uj, bob_alice_yi, Yi)
end role
```

**Figure 3** Role specification in HLPSL for another user $U_j$ in a group $S_n$ of our scheme.

- secrecy_of subs1: It represents that the generated random nonce $x_i$ and the password $pw_i$ are kept secret to the user $U_i$ only.
- secrecy_of subs2: It represents that the generated random nonce $y_i$ is kept secret to the server $S$ only.
- secrecy_of subs3: It represents that the secret value $sk_i$ is kept secret to the users $U_i, U_j$ and the server $S$.
- secrecy_of subs4: It represents that the generated random nonce $x_j$ and the password $pw_j$ are kept secret to another user $U_j$ in a group only.
- authentication_on alice_bob_xi: $U_i$ generates a random nonce $x_i$, where $x_i$ is only known to $U_i$. When the server $S$ receives $x_i$ from the messages from $U_i, S$ authenticates $U_i$.
- authentication_on alice_bob_xj: $U_j$ generates a random nonce $x_j$, where $x_j$ is only known to $U_j$. When the server $S$ receives $x_j$ from the messages from $U_j, S$ authenticates $U_j$.
- authentication_on bob_alice_yi: $S_j$ generates a random nonce $y_i$, where $y_i$ is only known to $S$. When the user $U_i$ receives $y_i$ from the messages from $S, U_i$ authenticates $S$.

```
role session(Ui, Uj, S: agent,
          SKus : symmetric_key,
          H : hash_func)
def=
  local  S1, S2, S3, R1, R2, R3: channel (dy)
composition
          useri(Ui, Uj, S, SKus, H, S1, R1)
       /\  userj(Ui, Uj, S, SKus, H, S2, R2)
       /\  server(Ui, Uj, S, SKus, H, S3, R3)
end role


role environment()
def=
  const ui, uj, s: agent,
       skus : symmetric_key,
       h   : hash_func,
       alice_bob_xi, alice_bob_xj,
       bob_alice_yi, subs1, subs2,
       subs3, subs4 : protocol_id
  intruder_knowledge = {ui, uj, s, h}
  composition
   session(ui, uj, s, skus, h)
/\ session(ui, uj, s, skus, h)
       /\ session(ui, uj, s, skus, h)
end role

goal
  secrecy_of subs1
  secrecy_of subs2
  secrecy_of subs3
  secrecy_of subs4
  authentication_on alice_bob_xi
  authentication_on alice_bob_xj
  authentication_on bob_alice_yi
end goal
environment()
```

**Figure 4** Role specification in HLPSL for the session and environment of our scheme.

### *7.2. Analysis of results*

We have simulated our improved scheme using the AVISPA web tool (AVISPA, 2014) for CL-AtSe back-end. For the replay attack checking, the back-end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that the back-end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev–Yao model check, the back-end checks whether there is any man-in-the-middle attack possible by the intruder. The simulation results for the formal security verification of our scheme using this back-end are shown in Fig. 5. The summary of the results under this back-end clearly shows that our scheme is safe. As a result, our scheme is secure against the passive attacks and the active attacks, such as the replay and man-in-the-middle attacks.

## 8. Performance comparison with related schemes

In this section, we compare the performance of our scheme with Lee et al.'s SGPAKE scheme and other related existing

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/avispa/web-interface–computation/
 ./tempdir/workfileGQXn6b.if

GOAL
 As Specified

BACKEND
 CL–AtSe

STATISTICS

 Analysed   : 9 states
 Reachable  : 1 states
 Translation: 0.33 seconds
 Computation: 0.01 seconds
```

**Figure 5** The result of the analysis using CL-AtSe backend of our scheme.

schemes (Abdalla et al., 2006; Boyd and Nieto, 2003; Dutta and Barua, 2006; Kim et al., 2004; Lee et al., 2013).

Table 4 shows the performance comparison of our scheme with Lee et al.'s SGPAKE scheme (Lee et al., 2013), Abdalla et al.'s scheme (Abdalla et al., 2006), Dutta and Barua's scheme (Dutta and Barua, 2006), Kim et al.'s scheme (Kim et al., 2004) and Boyd and Nieto's scheme (Boyd and Nieto, 2003). Boyd and Nieto's scheme (Boyd and Nieto, 2003) and Kim et al.'s scheme (Kim et al., 2004) are based on the public key infrastructure (PKI) and they provide higher security. However, they are required to maintain the complex and heavy public key systems, and also the users must hold extra storage for keeping public/private key pairs. Moreover, Boyd and Nieto's scheme (Boyd and Nieto, 2003) lacks forward security, but Kim et al.'s scheme (Kim et al., 2004) protocol can provide the forward security. Abdalla et al.'s scheme (Abdalla et al., 2006), Dutta and Barua's scheme (Dutta and Barua, 2006), Lee et al.'s scheme (Lee et al., 2013) and our proposed improved scheme are not required to maintain the public key system. Each user only needs to remember his/her weak password without any extra equipment for storing long-term secret keys. Lee et al.'s scheme (Lee et al., 2013) requires expensive exponential computations than Dutta and Barua's scheme and Abdalla et al.'s scheme. However, Zhang et al. (2012) showed that Dutta and Barua's scheme (Dutta and Barua, 2006) is insecure. In addition, Dutta and Barua's scheme and Abdalla et al.'s scheme do not provide users' privacy. In Lee et al.'s SGPAKE scheme (Lee et al., 2013), it needs to compute four exponents $g^{x_i}M^{pw_i}$, $K_i = \left(\frac{g^{y_i}N^{pw_i}}{N^{pw_i}}\right)^{x_i} = g^{x_i y_i}$, $Z_i = (K_{i-1})^{x_i}$ and $Z_{i+1} = (K_i')^{x_i}$, whereas the values $M^{pw_i}$ and $N^{pw_i}$ are pre-computed. Instead of computing the parameters $K_i = g^{x_i y_i}$, $K_{i-1} = g^{x_{i-1}y_{i-1}}$ and $K_i' = g^{x_{i+1}y_i}$ in Step 2 of the authentication and key exchange phase in Lee et al.'s SGPAKE scheme, we have computed $ek_i = g^{x_i y_i}$, $K_{i-1} = g^{x_{i-1}y_{i-1}^2}$ and $K_i' = g^{x_{i+1}y_i^2}$. As a result, our improved scheme takes only one extra field exponentiation to compute the encryption key $ek_i$ (at the server side) to enhance the security of Lee et al.'s SGPAKE scheme. As in Abdalla et al.'s scheme, Dutta–Barua's scheme, Kim et al.'s scheme, Boyd–Nieto's scheme and Lee et al.'s SGPAKE scheme, in our scheme we have also omitted the exponents required to compute the secret value $sk_i = Z_i^n \times X_{i+1}^{n-1} \times X_{i+2}^{n-2} \times \cdots \times X_{i+n-1}^1$. In addition, our scheme does not require any public key encryption and decryptions, whereas Boyd–Nieto's scheme requires a total of $2(n-1)$ public key encryption and decryptions, where $n$ is the number of members in a dynamic group $S_n$. Our scheme, Abdalla et al.'s scheme, Dutta–Barua's scheme and Lee et al.'s SGPAKE scheme do not require any cost for generating and verifying signatures. However, in Kim et al.'s scheme and Boyd–Nieto's scheme the number of signature generation and verification is $2n$ and $n$, respectively. From this table, it is clear that our scheme is efficient as compared to other schemes. Furthermore, our scheme provides formal security analysis and verification using AVISPA tool. Considering better security as compared to other related schemes, our scheme is much more applicable for practical applications.

**Table 4** Comparison of our improved scheme with related schemes.

| | Abdalla et al. (2006) | Dutta and Barua (2006) | Kim et al. (2004) | Boyd and Nieto (2003) | Lee et al. (2013) | Ours |
|---|---|---|---|---|---|---|
| User's public key | No | No | Yes | Yes | No | No |
| Formal security | Yes | Yes | Yes | Yes | No | Yes |
| User's privacy | A password shared with all users | A password shared with all users | PKI based | PKI based | A private password | A private password |
| Asymmetric en/decryption | 0 | 0 | 0 | $2(n-1)$ (total) | 0 | 0 |
| Signing/verifying | 0 | 0 | $2n$ (total) | $n$ (total) | 0 | 0 |
| Exponentiation | 3 | 3 | 2 | 0 | 4(2[a]) | 5(2[a]) |
| Symmetric encryption/decryption | 3 | $n+3$ | 0 | 0 | 1 | 1 |
| Whether provides forward security | Yes | No | Yes | No | Yes | Yes |
| Whether provides users' privacy | No | No | Yes | Yes | No | Yes |

[a] Two modular exponential computations can be pre-computed and thus they are ignored.

## 9. Conclusion

In this paper, we have first reviewed Lee et al.'s SGPAKE scheme. We have then shown that their scheme is vulnerable to off-line password-guessing attack and thus, their scheme fails to provide the users' privacy property. To remedy the security weakness found in Lee et al.'s SGPAKE scheme, we have proposed an effective improvement over Lee et al.'s SGPAKE scheme while retaining the original merits of Lee et al.'s SGPAKE scheme. We have provided both informal and formal security analysis of our scheme, and shown that our improved scheme is secure against various known attacks including off-line weak password guessing attack which is found in Lee et al.'s SGPAKE scheme. Therefore, our scheme provides data as well as users' privacy whereas Lee et al.'s scheme does not provide those properties. Moreover, our improved scheme is also efficient as compared to the other related schemes such as Abdalla et al.'s scheme, Dutta–Barua's scheme, Kim et al.'s scheme, Boyd–Nieto's scheme, and Lee et al.'s scheme. In addition, we have simulated our scheme for the formal security analysis using the widely-accepted AVISPA tool and shown that our scheme is secure against active and passive attacks. As a result, our scheme is much suitable for practical scenarios as compared to Lee et al's SGPAKE scheme and other related schemes.

## Acknowledgment

The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers and the Editor-in-Chief, which have improved the content and the presentation of this paper.

## References

Abdalla, M., Bresson, E.I., Chevassut, O., Pointcheval, D., 2006. Password-based group key exchange in a constant number of rounds. In: Proceedings of 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2006). Lecture Notes in Computer Science, vol. 3958. Springer-Verlag, pp. 427–442.

Abdalla, M., Pointcheval, D., 2005. Simple password-based authenticated key protocols. In: Topics in Cryptology – CT-RSA 2005. Lecture Notes in Computer Science, vol. 3376. Springer-Verlag, pp. 191–208.

Advanced Encryption Standard. FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. Accessed on November 2010.

Ammayappan, K., Negi, A., Sastry, V.N., Das, A.K., 2011. An ECC-based two-party authenticated key agreement protocol for mobile ad hoc networks. J. Comput. 6 (11), 2408–2416.

Aumasson, J.P., Henzen, L., Meier, W., Plasencia, M.N., 2010. Quark: a lightweight hash. In: Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010). Lecture Notes in Computer Science, vol. 6225. Springer-Verlag, pp. 1–15.

AVISPA. Automated Validation of Internet Security Protocols and Applications. <http://www.avispa-project.org/>. Accessed on January 2013.

AVISPA. AVISPA Web Tool. <http://www.avispa-project.org/web-interface/expert.php/>. Accessed on January 2014.

Boyd, C., Nieto, J.M.G., 2003. Round-optimal contributory conference key agreement. In: Proceedings of 6th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2003). Lecture Notes in Computer Science, vol. 2567. Springer-Verlag, pp. 161–174.

Bresson, E., Chevassut, O., Pointcheval, D., 2003. The Group Diffie–Hellman Problems. In: Proceedings of ACM Symposium on Applied Computing (SAC 2002). Lecture Notes in Computer Science, vol. 2595. Springer-Verlag, pp. 325–338.

Chatterjee, S., Das, A.K., Sing, J.K., 2014. A novel and efficient user access control scheme for wireless body area sensor networks. J. King Saud Univ. Comput. Inf. Sci. 26 (2), 181–201.

Das, A.K., 2011. Analysis and improvement on an efficient biometric based remote user authentication scheme using smart cards. IET Inf. Secur. 5 (3), 145–151.

Das, A.K., 2013. A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. Netw. Sci. 2 (1-2), 12–27.

Das, A.K., Goswami, A., 2013. A secure and efficient uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. J. Med. Syst. 37 (3), 1–16.

Das, A.K., Massand, A., Patil, S., 2013. A novel proxy signature scheme based on user hierarchical access control policy. J. King Saud Univ. Comput. Inf. Sci. 25 (2), 219–228.

Das, A.K., Paul, N.R., Tripathy, L., 2012. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. Inf. Sci. 209, 80–92.

Dolev, D., Yao, A., 1983. On the security of public key protocols. IEEE Trans. Inf. Theory 29 (2), 198–208.

Dutta, R., Barua, R., 2006. Password-based encrypted group key agreement. Int. J. Netw. Secur. 3 (1), 30–41.

Islam, S.H., Biswas, G.P., 2013. Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings. J. King Saud Univ. Comput. Inf. Sci. 25 (1), 51–61.

Islam, S.H., Biswas, G.P., 2014. A provably secure identity-based strong designated verifier proxy signature scheme from bilinear pairings. J. King Saud Univ. Comput. Inf. Sci. 26 (1), 55–67.

Jeong, I., Lee, D., 2007. Key agreement for key hypergraph. Comput. Secur. 26 (7-8), 452–458.

Kim, H.J., Lee, S.M., Lee, D.H., 2004. Constant-round authenticated group key exchange for dynamic groups. In: Proceedings of Advances in Cryptology – ASIACRYPT 2004. Lecture Notes in Computer Science, vol. 3329. Springer-Verlag, pp. 245–259.

Lee, S.M., Hwang, J.Y., Lee, D.H., 2004. Efficient password-based group key exchange. In: Proceedings of 1st International Conference on Trust and Privacy in Digital Business (TrustBus 2004). Lecture Notes in Computer Science, vol. 3184. Springer-Verlag, pp. 191–199.

Lee, T.F., Chang, I.P., Wang, C.C., 2013. Simple group password-based authenticated key agreements for the integrated EPR information system. J. Med. Syst. 37 (2), 1–6.

Lee, T.F., Wen, H.A., Hwang, T., 2006. A weil pairing-based round-efficient and fault-tolerant group key agreement protocol for sensor networks. In: Sensor Network Operations. IEEE Press, pp. 571–579.

Manuel, S., 2011. Classification and generation of disturbance vectors for collision attacks against SHA-1. Des. Codes Cryptogr. 59 (1-3), 247–263.

Odelu, V., Das, A.K., Goswami, A., 2014. A secure effective key management scheme for dynamic access control in a large leaf class hierarchy. Inf. Sci. 269 (C), 270–285.

Sarkar, P., 2010. A simple and generic construction of authenticated encryption with associated data. ACM Trans. Inf. Syst. Secur. 13 (4), 33.

Stallings, W., 2003. Cryptography and Network Security: Principles and Practices, third ed. Prentice Hall, India.

Secure Hash Standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. Accessed on November 2010.

Steiner, M., Tsudik, G., Waidner, M., 2000. Key agreement in dynamic peer groups. IEEE Trans. Parallel Distrib. Syst. 11 (8), 769–780.

Stinson, D.R., 2006. Some observations on the theory of cryptographic hash functions. Des Codes Cryptogr. 38 (2), 259–277.

Tzeng, W.G., 2002. A secure fault-tolerant conference-key agreement protocol. IEEE Trans. Comput. 51 (4), 373–379.

Tzeng, W.G., Tzeng, Z.J., 2000. Round-efficient conference key agreement protocols with provable security. In: Proceedings of Advances in Cryptology – ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976. Springer-Verlag, pp. 614–627.

Zhang, H., Xu, C., Li, C., Sangim, A.R., 2012. Two attacks on Dutta's dynamic group key agreement protocol. In: Proceedings of International Conference on Wireless Communications and Applications (ICWCA 2011). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), vol. 72. Springer Berlin Heidelberg, pp. 419–425.