Journal of King Saud University – Computer and Information Sciences (2017) 29, 311–324

King Saud University

Journal of King Saud University – Computer and Information Sciences

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

An improved three party authenticated key exchange protocol using hash function and elliptic curve cryptography for mobile-commerce environments

S.K. Hafizul Islam [a,*], Ruhul Amin [b], G.P. Biswas [b], Mohammad Sabzinejad Farash [c], Xiong Li [d], Saru Kumari [e]

[a] Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani Campus, Rajasthan 333031, India
[b] Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826004, Jharkhand, India
[c] Department of Mathematical Sciences and Computer, University of Kharazmi, Tehran, Iran
[d] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[e] Department of Mathematics, Ch. Charan Singh University, Meerut 250004, Uttar Pradesh, India

Received 12 January 2015; revised 22 April 2015; accepted 27 August 2015
Available online 3 November 2015

KEYWORDS

Elliptic curve cryptography;
Authenticated key exchange protocol;
Man-in-the-middle attack;
Mobile-commerce environments

Abstract In the literature, many three-party authenticated key exchange *(3PAKE)* protocols are put forwarded to established a secure session key between two users with the help of trusted server. The computed session key will ensure secure message exchange between the users over any insecure communication networks. In this paper, we identified some deficiencies in Tan's *3PAKE* protocol and then devised an improved *3PAKE* protocol without symmetric key en/decryption technique for mobile-commerce environments. The proposed protocol is based on the elliptic curve cryptography and one-way cryptographic hash function. In order to prove security validation of the proposed *3PAKE* protocol we have used widely accepted *AVISPA* software whose results confirm that the proposed protocol is secure against active and passive attacks including replay and man-in-the-middle attacks. The proposed protocol is not only secure in the *AVISPA* software, but it also secure

* Corresponding author.
E-mail addresses: hafi786@gmail.com (S.H. Islam), amin_ruhul@live.com (R. Amin), gpbiswas@gmail.com (G.P. Biswas), sabzinejad@khu.ac.ir (M.S. Farash), lixiongzhq@163.com (X. Li), saryusiirohi@gmail.com (S. Kumari).

Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.08.002
1319-1578

against relevant numerous security attacks such as man-in-the-middle attack, impersonation attack, parallel attack, key-compromise impersonation attack, etc. In addition, our protocol is designed with lower computation cost than other relevant protocols. Therefore, the proposed protocol is more efficient and suitable for practical use than other protocols in mobile-commerce environments.

1. Introduction

The authentication of the communicating clients and the confidentiality of the transmitted message are the primary objectives of network security, when the communication media is a public network. Thus, to achieve these two security goals simultaneously, many *3PAKE* protocols have been introduced. *3PAKE* protocol allows two clients to authenticate each other with the assistance of a trusted server and then computes a secret session key via any public network. The session key can subsequently be used to establish a secure channel between the clients. *3PAKE* protocol is divided into following categories: password-based *3PAKE* (Lin et al., 2000, 2001, 2004; Chang and Chang, 2004; Lu and Cao, 2006; Chen et al., 2008b; Yoon and Yoo, 2008; Sun et al., 2005; Lee and Hwang, 2010; Yang et al., 2007; Reddy and Padmavathamma, 2007) and *3PAKE* protocol using server's public key (Chen et al., 2008a; Yang and Chang, 2009; Pu et al., 2009; Tan, 2010a). In password-based *3PAKE* protocol, two clients share an easy-memorable password with the trusted server and then generate the session key securely between them with the help of the server. However, most of these protocols are susceptible to undetectable off-line password guessing attack (Lin et al., 2000, 2001), on-line password guessing attack (Chen et al., 2008b; Yoon and Yoo, 2008; Sun et al., 2005; Nam et al., 2006; Phan et al., 2008), impersonation attack (Chung and Ku, 2008), unknown key-share attack (Phan et al., 2008; Guo et al., 2008), etc. In addition, the computation cost and communication load of these protocols are heavy because they have employed the modular exponentiation (Lin et al., 2001; Lee et al., 2004; Chang and Chang, 2004; Chen et al., 2008b; Sun et al., 2005), public/symmetric key encryption/decryption (Lin et al., 2000, 2001; Chang and Chang, 2004; Yoon and Yoo, 2008; Sun et al., 2005) and the transmitted message size is large in each round (Lin et al., 2000; Lee et al., 2004; Chang and Chang, 2004; Sun et al., 2005). Due to the limitations of bandwidth, computation ability and storage space of the low-power mobile devices, the above mentioned protocols are not suitable for mobile-commerce environments. Another type of *3PAKE* protocol used the server's public key and public/symmetric key cryptosystem. In Fig. 1, we have made a tree structure to show the *3PAKE* protocol division categories and their differences.

1.1. Literature review

In 2008, Chen et al. (2008a) proposed a round and computation-efficient *3PAKE* protocol using smartcard, but the protocol is later shown to be vulnerable to stolen-verifier attack as claimed by Yang and Chang (2009). If the adversary steals the pre-shared secret from the smartcard, then he/she can impersonate the legal client and share the session key with other clients. Moreover, the protocol has the high computation cost and communication loads. Therefore, Chen et al.'s *3PAKE* protocol is not suitable for mobile-commerce environments. To overcome the weaknesses of Chen et al., Yang and Chang (2009) proposed an efficient *3PAKE* protocol using elliptic curve cryptography *(ECC)* and without sharing any pre-shared secrete between client and server in which computation and communication overheads for establishing a session key are significantly reduced. However, Pu et al. (2009) demonstrated that the protocol is potentially vulnerable to unknown key-share attack, man-in-the-middle attack and impersonation attack.

1.2. Motivation and contribution

In 2010, Tan (2010a) independently pointed out that Yang and Chang's protocol is still susceptible to impersonation-of-initiator attack, impersonation-of-responder attack and parallel attack, and further proposed an improved *3PAKE* protocol based on *ECC*. In 2011, Nose et al. (2011) demonstrated that Tan's *3PAKE* protocol still suffers from the impersonation-of-initiator attack, impersonation-of-responder attack and man-in-the-middle attack. Nose et al. also claimed that these three attacks can be mounted on Yang and Chang's protocol (Yang and Chang, 2009), and Pu et al.'s protocol (Pu et al., 2009). Furthermore, this paper shows that Tan's protocol cannot resist the known session-specific temporary attack and the clock synchronization problem. In addition, Tan's protocol has high computation cost due to additional elliptic curve scalar point multiplication and symmetric en/decryption process. In this paper, we proposed an improved *3PAKE* protocol based on *ECC* for mobile-commerce environments. The proposed protocol employs the simple hash function (Message Digest Algorithm, 1992) but no en/decryption (Advanced Encryption Standard, 2001) process is needed. The proposed protocol is secure under known attacks and has lower computation cost, and thus it will be suitable for mobile-commerce environments.

1.3. Outline of the paper

We presented the basic concept of elliptic curve cryptography and the related computational problems in Section 2. Section 3 addressed Tan's *3PAKE* protocol and the security analysis of it is given in Section 4. We then proposed our improved protocol in Section 5. The formal security validation of our protocol in AVISPA software is explained in Section 6. The informal security analysis of our protocol appears in Section 7. Section 8 discussed the performance analysis and the conclusion of this paper in Section 9.

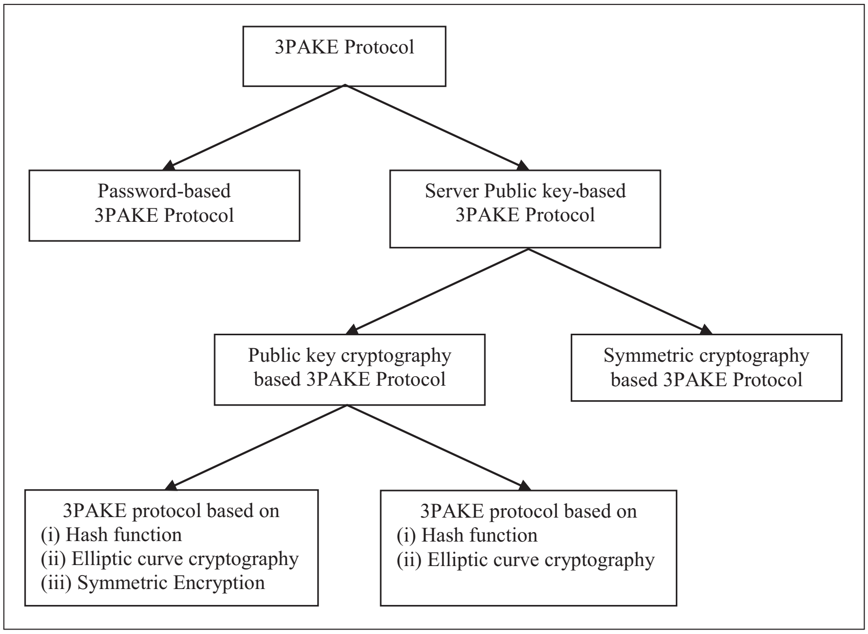

Fig. 1 Different types of *3PAKE* protocols.

2. Preliminaries

2.1. Elliptic curve cryptography

The *ECC* was initially proposed by Koblitz (1987), and its security was based on the difficulty of *ECDLP*. Later on, it is widely accepted in designing different cryptographic protocols for its effectiveness in security, communication and computation and a number of efficient *ECC*-based *PKCs* have been proposed. For the sake of clarity, the basics of the elliptic curve cryptography and some related computationally hard problems are given below.

Let E/F_q be a set of elliptic curve points over a prime field F_q, defined by the following non-singular elliptic curve equation:

$$y^2 \bmod q = (x^3 + ax + b) \bmod q \quad (1)$$

where $x, y, a, b \in F_q$ and $(4a^3 + 27b^2) \bmod q \neq 0$. The additive elliptic curve group defined as $G_q = \{(x, y) : x, y \in F_q$ and $(x, y) \in E/F_q\} \cup \{O\}$, where the point *"O"* is known as *"point at infinity"* or *"zero point"*. A brief discussion about the elliptic curve group properties is given below:

- **Point addition**. Let P, Q be two points on the curve (1), then $P + Q = R$, where the line joining P and Q intersects the curve (1) at $-R$, and the reflection of it with respect to *x-axis* is R.
- **Point subtraction**. If $Q = -P$, then $P + Q = P - P = O$ i.e., the line joining of P and $-P$ intersects the curve (1) at O.
- **Point doubling**. Point doubling is the addition of a point P on the curve (1) to itself to obtain another point Q on the same curve. Let $2P = Q$, the tangent line at P intersects the curve (1) at $-Q$ and the reflection of it with respect to *x-axis* is Q.
- **Scalar point multiplication**. The scalar point multiplication in G_q is defined as $cP = P + P + \cdots + P$ (c times), where $c \in Z_q^*$ is a scalar.
- **Order of a point**. A point P has order d if d is the smallest integer such that $dP = O$ and $d > 0$.

2.2. Computational problem

Definition 1 (*Elliptic Curve Discrete Logarithm Problem (ECDLP)*). Given $Q, R \in E_q(a, b)$, where $R = a \cdot Q$ and $a \in Z_q^*$. It is hard to compute a from R.

Definition 2 (*Computational Diffie–Hellman (CDH) Problem*). Given $(Q, a \cdot Q, b \cdot Q) \in E_q(a, b)$ for any $a, b \in Z_q^*$, computation of $a \cdot b \cdot Q$ is hard.

3. Review of Tan's *3PAKE* protocol

In this section, we reviewed and analyzed Tan's *3PAKE* protocol (Tan, 2010a) based on *ECC* (Koblitz, 1987; Menezes et al., 1996). The protocol is composed of two phases: system initialization phase and authenticated key exchange phase.

3.1. System initialization phase

In this phase, S initializes and select system's parameters as follows:

Step 1 Select a finite field F_q over $q > 2^{160}$.
Step 2 Select an elliptic curve $E_q(a, b)$: $y^2 \bmod q = (x^3 + ax + b) \bmod q$ with order n over F_q, where $a, b \in F_q$ and $(4a^3 + 27b^2) \neq 0 \bmod q$.
Step 3 Select a symmetric en/decryption algorithm $E_k()/D_k()$ (e.g., *AES* (Advanced Encryption Standard, 2001)), where k denotes the symmetric key.

Step 4 Select a base point Q of order n over $E_q(a, b)$.

Step 5 Publish $E_q(a, b), E_k()/D_k()$ and Q.

Step 6 The clients A and B must register to S to generate their private/public key pair (d_A/U_A) and (d_B/U_B). The private/public key pair of S is (d_S/U_S), where $U_A = d_A \cdot Q, U_B = d_B \cdot Q$ and $U_S = d_S \cdot Q$.

3.2. Authenticated key exchange phase

This phase is divided into three rounds as described below.

Round 1. In this round, A performs the following operations:

Step 1 Select an integer $r_A \in Z_q^*$ randomly and compute $R_A = r_A \cdot U_A$ and $K_A = r_A \cdot d_A \cdot U_S = (K_{Ax}, K_{Ay})$.

Step 2 Randomly select $w_A \in Z_q^*$ and then compute $W_A = w_A \cdot Q$.

Step 3 Select a time stamp T_A and compute $C_{AS} = E_{K_{Ax}}(R_A, W_A, ID_A, ID_B, T_A)$ using the encryption key K_{Ax}.

Step 4 Send the messages $(ID_A, Request)$ and (ID_A, C_{SA}, R_A) to B and S, respectively. Here, the message *"Request"* denotes a request that A asks B to share a session key with him.

Round 2. In this round, B performs the following operations after receiving the initiation request $(ID_A, Request)$ from A:

Step 1 Select an integer $r_B \in Z_q^*$ randomly and compute $R_B = r_B \cdot U_B$ and $K_B = r_B \cdot d_B \cdot U_S = (K_{Bx}, K_{By})$.

Step 2 Randomly select $w_B \in Z_q^*$ and then compute $W_B = w_B \cdot Q$.

Step 3 Select a time stamp T_B and compute $C_{BS} = E_{K_{Bx}}(R_B, W_B, ID_B, ID_A, T_B)$ using the encryption key K_{Bx}.

Step 4 Send the messages $(ID_B, Response)$ and (ID_B, C_{BS}, R_B) to A and S, respectively. Here, the message *"Response"* means that B accepts A's request.

Round 3. S executes the following operations after receiving the messages (ID_A, C_{SA}, R_A) and (ID_B, C_{BS}, R_B) from A and B:

Step 1 S first validates the time stamp $\langle T_A, T_B \rangle$ and then computes the symmetric keys $K_A = d_S \cdot R_A = (K_{Ax}, K_{Ay})$ and $K_B = d_S \cdot R_B = (K_{Bx}, K_{By})$.

Step 2 Retrieve $(R_A, W_A, ID_A, ID_B, T_A) = D_{K_{Ax}}(C_{AS})$ and $(R_B, W_B, ID_B, ID_A, T_B) = D_{K_{Bx}}(C_{BS})$ using K_{Ax} and K_{Bx} as the decryption key, respectively.

Step 3 S checks if the decrypted timestamps T_A and T_B are same as received T_A and T_B, and then compares decrypted ID_A and ID_B are same as received ID_A and ID_B, respectively.

Step 4 Furthermore, S checks if the decrypted R_A and the received R_A are same. If the result is negative, then S sends an *authentication-failed* message to B and also checks if the decrypted R_B and the received R_B are same. If the condition violates then S sends an *authentication-failed* message to A. After validating A and B, S selects a timestamp T_S and computes $C_{SA} = E_{K_{Ax}}(R_A, W_B, ID_A, T_S, ID_S)$ and $C_{SB} = E_{K_{Bx}}(R_B, W_A, ID_B, T_S, ID_S)$.

Step 5 S sends $\langle ID_S, C_{SA}, T_S \rangle$ and $\langle ID_S, C_{SB}, T_S \rangle$ to A and B, respectively.On receiving $\langle ID_S, C_{SA}, T_S \rangle$ from S, A performs the following operations to accomplish the session key exchange.

Step 6 A, validates ID_S and T_S, and then decrypts C_{SA} using K_{Ax} and retrieves $(R_A, W_B, ID_A, T_S, ID_S) = D_{K_{Ax}}(C_{SA})$. A then checks if ID_S and T_S are valid and the decrypted R_A is same as his own R_A selected in *Round 1*. If both the condition hold, then A confirms that B is authenticated by S. Then A computes the session key $SK = w_A \cdot W_B = w_A \cdot w_B \cdot Q$. Otherwise, A rejects the transaction.In the same way, B executes the following operations after receiving $\langle ID_S, C_{SB}, T_S \rangle$ from S.

Step 7 B validates ID_S and T_S, and checks that the decrypted R_B is same as his own R_B, selected in *Round 2*. If they are same, B confirms that A has been authenticated by S and generates the session key by calculating $SK = w_B \cdot W_A = w_A \cdot w_B \cdot Q$. Otherwise, B rejects the transaction.

4. Security vulnerabilities of the Tan's *3PAKE* protocol

Although, Tan's *3PAKE* protocol renovates the weaknesses of Yang and Chang's protocol, we found that Tan's protocol is not suitable for real environments as it has the following drawbacks:

4.1. Known session-specific temporary information attack

In 2001, Canetti and Krawczyk (2001) investigated the known session-specific temporary information attack. Later on, Cheng et al. (2005) pointed out that if the adversary ($\mathcal{A}$) gained the knowledge about the ephemeral secrets (selected by A and B) of a session, however, he should not be able to determine the resulting session key. The following conditions encouraged us to protect this kind of attack and it may happen in real environments due to the following reasons (Mandt, 2006):

- The clients and the server must trust on the internal/external source of random number generator that may be controlled by $\mathcal{A}$ (Islam, 2014a,b,c; Islam et al., 2015; Islam and Khan, 2014).
- The random numbers are generally stored in an insecure device. If the random numbers (ephemeral secrets) are not erased properly in each session, then $\mathcal{A}$ may hijack users' computer and learns the random numbers (Islam, 2014a, b,c; Islam et al., 2015; Islam and Khan, 2014).

From the aforementioned discussions, we claimed that Yang and Chang's *3PAKE* protocol (Yang and Chang, 2009), Pu et al.'s *3PAKE* protocol (Pu et al., 2009), Tan's *3PAKE* protocol (Tan, 2010a), Tan's *3PAKE* protocol (Tan, 2010b) and He et al.'s *3PAKE* protocol (He et al., 2013) failed to prevent the known session-specific temporary information attack. Since, in Yang and Chang (2009), Pu et al. (2009), Tan (2010a,b), and He et al. (2013), clients A and B select the ephemeral secrets w_A and w_B, respectively and after the successful authentication, they compute the session key as $SK = w_A \cdot w_B \cdot Q$. If the ephemeral secrets, i.e., w_A and w_B are disclosed to $\mathcal{A}$, then the session key SK can be easily compromised by $\mathcal{A}$. Therefore, Yang and Chang's *3PAKE* protocol (Yang and Chang, 2009), Pu et al.'s *3PAKE* protocol (Pu et al., 2009), Tan's *3PAKE* protocol (Tan, 2010a), Tan's *3PAKE* protocol (Tan, 2010b) and He et al.'s *3PAKE* protocol (He et al., 2013) cannot resist the known session-specific temporary information attack. The detailed explanation of known session-specific temporary information attack is given in Islam and Biswas (2012).

4.2. Clock synchronization problem

In timestamp-based protocols (Tan, 2010a,b; He et al., 2013), system clocks of all the connected devices must be synchronized, otherwise, the clock synchronization problem will hamper the protocol execution. Tan's *3PAKE* protocol (Tan, 2010a) employs the timestamp to detect forced delay to protect the replay attack and man-in-the-middle attack. However, the timestamp raises the problem of clock synchronization in large networks, such as wide area networks, mobile communication networks and satellite communication networks. All the protocols based on the concept of timestamp can withstand the replay attack using systems' timestamp provided the system clock must be synchronized; otherwise the protocol will not work properly. Since the transmission delay is long and unpredictable in a wide area network environment (Gong, 1992), a potential replay attack exists in all timestamp-based protocols. In the communication networks with tightly synchronized system clocks, such as local area networks, the timestamp-based protocol is preferable. On the other hand, the nonce-based protocol is suitable for a large network where clock synchronization is difficult, such as wide area networks, mobile communication networks, and satellite communication networks. Thus, Tan's *3PAKE* protocol (Tan, 2010a) is not suitable for mobile-commerce environments. Accordingly, we confirmed that Tan's *3PAKE* protocol (Tan, 2010b) and He et al.'s *3PAKE* protocol (He et al., 2013) also suffered from the same problem as they employed the timestamp. Note that, in our *3PAKE* protocol we used the random number-based (nonce) solution, instead of timestamp that eliminated the synchronization problem.

4.3. High computation cost

In Table 3, we observed that the computation cost of the protocol proposed in Tan (2010a), Nam et al. (2006), Phan et al. (2008), Tan (2010b), and He et al. (2013) is still high. A *3PAKE* protocol needs higher amount of communication processing time, which means two communicating clients have to spend more time to establish a common session key between them, so the protocol may not be suitable for mobile-commerce environments. Since, the mobile devices have low computation ability, limited power supply and low storage space. Thus, the protocol proposed in Tan (2010a), Nam et al. (2006), Phan et al. (2008), Tan (2010b), and He et al. (2013) cannot be usable in mobile-commerce environments. In order to reduce the computation cost, in our proposed protocol we avoided the use of encryption/decryption technique and used the light weight hash function.

Table 1 Different notations used in this paper.

Notations	Meaning
A	The protocol participant (initiator)
B	The protocol participant (responder)
S	The protocol participant (server)
k	The security parameter
q	A large prime number of k-bit length and $q > 3$
F_q	A field of prime order q
$E_q(a,b)$	A set of elliptic curve points of order n, where $a, b \in F_q$
Q	A base point of order n over $E_q(a,b)$
$E_x()/D_x()$	The symmetric en/decryption algorithm under the key x (e.g., AES Advanced Encryption Standard, 2001)
(d_A, U_A)	The private/public key pair of the entity i, where $i = A, B, S$, where $d_i \in Z_q^*$ and $U_i = d_i \cdot Q$
$H()$	One-way cryptographic hash function (e.g., MD5)
$\Vert$	The message concatenation operator
$(\cdot)$	The elliptic cure scalar point multiplication
$\mathcal{A}$	The Adversary

5. The proposed *3PAKE* protocol

To renovate the drawbacks of Tan (2010a), Nam et al. (2006), Phan et al. (2008), Tan (2010b), and He et al. (2013), we proposed a more efficient and secure *3PAKE* protocol using *ECC* for mobile-commerce environments. The proposed *3PAKE* protocol employs one-way hash function instead of a costly symmetric cryptosystem. The notations used in the proposed *3PAKE* protocol is given in Table 1. Our protocol has two phases: system initialization phase and authenticated key exchange phase.

5.1. System initialization phase

In this phase, S initializes system parameters as done in Tan's *3PAKE* protocol (Tan, 2010a). In our protocol, we used a one-way secure hash function $H(\cdot)$ (i.e., *MD5*) instead of symmetric en/decryption tool.

5.2. Authenticated key exchange phase

In this phase, three entities are involved: two clients A and B that wish to establish a secure session key between them, and a trusted server S, who assists A and B to authenticate each other via a public network. The detailed steps of our protocol are given as follows.

Round 1. In this round, A, executes the following steps:

Step 1 Pick an integer $r_A \in Z_q^*$ randomly and then compute $H_A = H(r_A \Vert d_A)$ and $R_A = H_A \cdot Q$.

Step 2 Then compute $K_A = d_A \cdot U_S = d_A \cdot d_S \cdot Q$ and $C_{AS} = H(ID_A \Vert ID_B \Vert R_A \Vert K_A)$.

Step 3 Send $(ID_A, Request)$ and $(ID_A, ID_B, R_A, C_{AS})$ to B and S, respectively.

Round 2. After receiving A's initiation message $(ID_A, Request)$, following operations are executed by B.

Step 1 Pick an integer $r_B \in Z_q^*$ randomly and then compute $H_B = H(r_B \| d_B)$ and $R_B = H_B \cdot Q$.

Step 2 Compute $K_B = d_B \cdot U_S = d_B \cdot d_S \cdot Q$ and $C_{BS} = H(ID_B \| ID_A \| R_B \| K_B)$.

Step 3 Send $(ID_B, Response)$ and $(ID_B, ID_A, R_B, C_{BS})$ to A and S, respectively.

Round 3. After receiving $(ID_A, ID_B, R_A, C_{SA})$ and $(ID_B, ID_A, R_B, C_{BS})$ from A and B, S performs the following operations.

Step 1 Compute the symmetric keys $K_A = d_S \cdot U_A = d_A \cdot d_S \cdot Q$ and $K_B = d_S \cdot U_B = d_B \cdot d_S \cdot Q$, respectively.

Step 2 Compute $\overline{C}_{AS} = H(ID_A \| ID_B \| R_A \| K_A)$ using received R_A and the computed K_A. S checks the condition $\overline{C}_{AS} = ?C_{AS}$. If it does not hold, S sends an *authentication-failed* message to B. Otherwise, S computes $C_{SA} = H(ID_A \| ID_B \| R_A \| R_B \| K_A)$ and sends the message (R_B, C_{SA}) to A.

Step 3 Compute $\overline{C}_{BS} = H(ID_B \| ID_A \| R_B \| K_B)$ using received R_B and his own K_B. S checks the condition $\overline{C}_{BS} = ?C_{BS}$. If it does not hold, S sends an *authentication-failed* message to A. Otherwise, S computes $C_{SB} = H(ID_B \| ID_A \| R_B \| R_A \| K_B)$ and sends the message (R_A, C_{SB}) to B. Now, A executes following operations after receiving the message (R_B, C_{SA}) from S.

Step 4 On receiving $(R_B, C_{SA}), A$ computes $\overline{C}_{SA} = H(ID_A \| ID_B \| R_A \| K_A)$ using his own R_A and K_A generated in *Round 1* and the received R_B. Now, A checks the condition $\overline{C}_{SA} = ?C_{SA}$. If the result is positive, A computes the session key $SK = H(ID_A \| ID_B \| R_A \| R_B \| K)$, where $K = H_A \cdot R_B = H_A \cdot H_B \cdot Q$. Otherwise, A terminates the session. Now, B takes the following actions after receiving the message (R_A, C_{SB}) from S.

Step 5 Upon receiving $(R_A, C_{SB}), B$ computes $\overline{C}_{Sb} = H(ID_B \| ID_A \| R_B \| R_A \| K_B)$ using the values R_B and K_B generated in *Round 2* and the received R_A. Now, B checks the condition $\overline{C}_{SB} = ?C_{SB}$. If the result is positive then computes the session key $SK = H(ID_A \| ID_B \| R_A \| R_B \| K)$, where $K = H_B \cdot R_A = H_A \cdot H_B \cdot Q$. Otherwise, B terminates the session.

We explained the proposed *3PAKE* protocol in the Fig. 2.

6. Simulation for formal security verification using *AVISPA* tool

This section is provided for formal security verification of the proposed *3PAKE* protocol using *AVISPA* simulator (Amin and Biswas, 2015; Islam and Biswas, 2014, 2013) to ensure that the protocol is secure against the active and passive attacks including replay and man-in-the-middle attacks. We provided the concept and knowledge about *AVISPA* simulator tool and then present *HLPSL* code description along with simulation results of the our protocol.

6.1. Brief description of the AVISPA simulation tool

AVISPA is considered as a widely-accepted simulation tool for the formal security verification, which measured whether the security protocol is *SAFE* or *UNSAFE*. *AVISPA* supports High Level Protocol Specification Language *HLPSL*. The structure of *AVISPA* tool is shown in Fig. 3. Currently, *AVISPA* (2015) supports four different back-ends and abstraction based methods, which are integrated through the *HLPSL* code. The First back-end, called On-the-fly Model-Checker *(OFMC)* is responsible for symbolic techniques for exploring the state space in a demand-driven way. The second back-end (*CL-AtSe*) provides a translation from any security protocol specification written as transition relation in intermediate format *(IF)* into a set of constraints, which are effectively used to find whether there are attacks on protocols. The third back-end is *SAT* based Model checker which generates a propositional formulae and then fed to a state-of-the-art *SAT* solver and any model found is translated back into an attack. The Tree Automata based on Automatic Approximations for the Analysis of Security Protocols *(TA4SP)* is the last back-end, which is responsible for approximating the intruder knowledge by using regular tree languages. As mentioned earlier, *HLPSL* specification is translated into the intermediate form *(IF)* using *hlpsl2if* translator. The *(IF)*, which is a lower level language than *HLPSL* is read directly by the back-ends to *AVISPA* tool. It may be noted that this intermediate translation step is transparent to the user.

It is to be noted that *AVISPA* is a role-oriented language that means each participant plays a role during the protocol execution. Each role is independent of the others, getting some initial information by parameters and communicating with the other roles by channels. It is also to be noted that the channel may be secure or insecure. The intruder is modeled using Dolev-Yao model (Dolev and Yao, 1983) with the possibility for the intruder to assume a legitimate role in a protocol run. The role system also described the number of sessions, the number of principals and the roles. Based on the four back-ends, *OUTPUT FORMAT (OF)* is generated and after successful execution, *(OF)* described the result whether the protocol is safe or unsafe or under what condition the output is obtained.

6.2. Brief specification of the proposed protocol

In this section, we discussed all the roles involved in our proposed *3PAKE* protocol in *HPLSL* language. In Fig. 4, we implemented the role for the client A in *HLPSL* language. Initially, S provides the private/prublic key pair to A and then sends $Snd(IDA.IDB.RA'.CAS')$ to A through open channel. It is to be noted that the random number RA' was generated using *new()* operation and A transmits any message with the help of *Snd()* operation. The declaration $secret(\{DA'\}, subs1, \{A, S\})$ indicates that the private key DA' is only known to *(A,S)*. In transition 2, A receives $Rcv(RB.CSA')$ from S through open channel with the help of *Rcv()* operation and then computes the session key.

In Fig. 5, we implemented the role for B in *HLPSL* language. Resembling, A, S provide security parameters and finally sends $Snd(IDB.IDA.RB'.CBS')$ to S through an open channel. It is to be noted that the random number RB' was generated using *new()* operation and B transmits any message

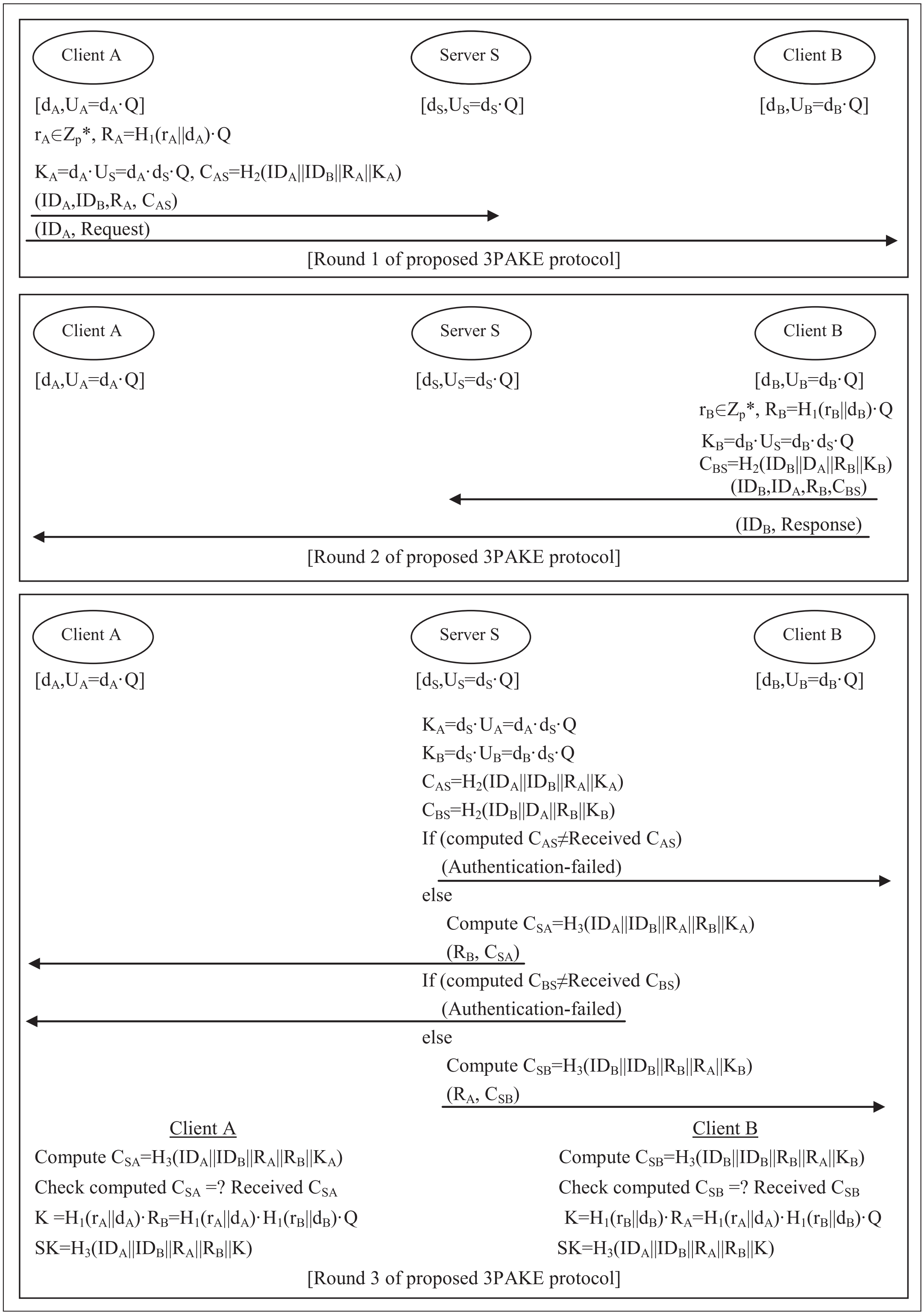

Fig. 2 The proposed *3PAKE* protocol.

with the help of *Snd()* operation. The declaration *secret*$(\{DB'\}, subs2, \{B, S\})$ indicates that the private key *DB'* is only known to *(B,S)*. In transition 2, *B* receives *Rcv*$(RA.CSB')$ from *S* through an open channel with the help of *Rcv()* operation and then computes the session key.

In Fig. 6, we implemented the role for *S* in *HLPSL* language. *S* first received the messages *Rcv*$(IDA.IDB.RA'.KA')$ and *Rcv*$(IDA.IDB.RB'.KB')$ in parallel from *A* and *B*, respectively. Then, *S* sends *Snd*$(RA.CSB')$ and *Snd*$(RA.CSB')$ to *A* and *B*, respectively. The declaration *secret*$(\{DS\}, subs3, \{S\})$

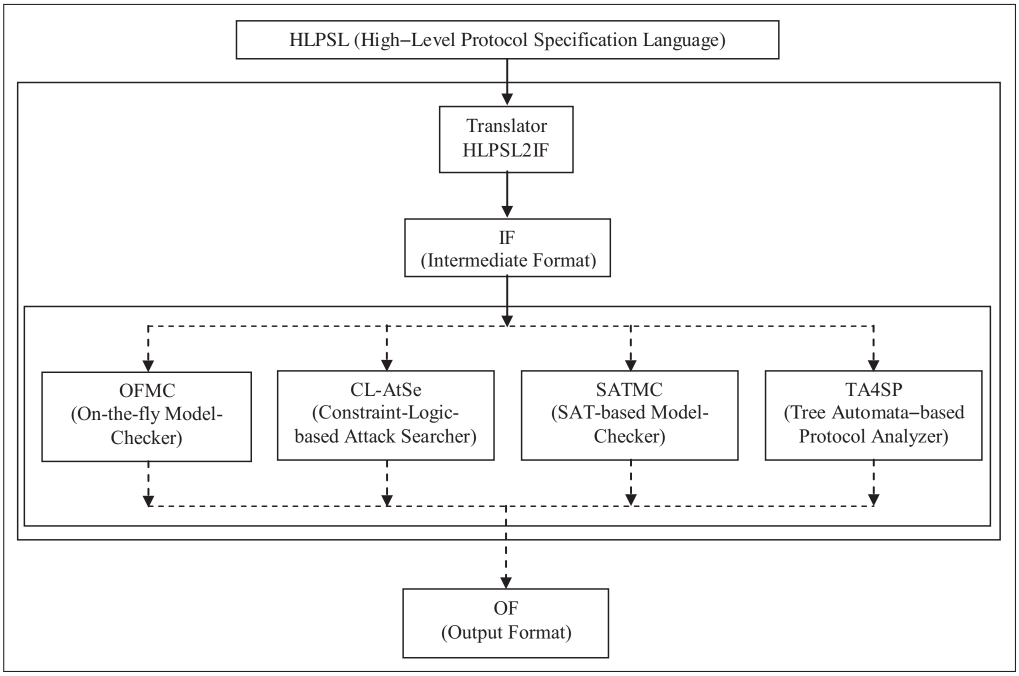

Fig. 3 Architecture of the *AVISPA* Tool.

indicates that the private key *DS* is kept secret permanently and only known to *S*.

In Fig. 7, we presented the roles for the session, goal and the environment in *HLPSL* language. In session, all the basic roles including the roles for *A*,*S* and *B* are instanced with concrete arguments. The environment section contains the global constant and composition of one or more sessions and the intruder knowledge is also given. The current version (2006/02/2013) of *HLPSL* supports the standard authentication and secrecy goals. In our implementation, the following three secrecy goals and two authentications are verified.

(1) The secrecy_of subs1 represents that the private key of *A* is kept secret to only (S, A).
(2) The secrecy_of subs2 represents that the private key of *B* is kept secret to only (S, B).
(3) The secrecy_of subs3 represents that the private key of server is kept secret to only (S).
(4) The authentication_on alice_server_raa represents that *A* generates a random number *raa*, where *raa* is only known to *A* and if *S* receives it through message securely, *S* then authenticates *A*.
(5) The authentication_on bob_server_rbb represents that *B* generates a random number *rbb*, where *rbb* is only known to *B* and if *S* receives it through message securely, *S* then authenticates *B*.

6.3. Simulation results

In this section, we presented the simulation results of our *3PAKE* protocol on the back-ends *OFMC* and *CL-AtSe* using *AVISPA* web tool. Figs. 8 and 9 ensure that the proposed protocol is *SAFE* under two back-ends *OFMC* and *CL-AtSe*, respectively i.e., the proposed protocol is secure against the active and passive attacks including replay and man-in-the-middle attacks. Therefore, we claimed that the proposed *3PAKE* protocol is secure against security attacks.

7. Further security analysis

In this section, we further demonstrated that the proposed protocol eliminates the security weaknesses of Tan's protocol and also provides resilience against other known attacks.

```
role alice (A, S, B : agent,
% H is hash function
H, Mul: hash_func, Snd, Rcv: channel(dy))
played_by A
def=
local State : nat,
DA, UA, IDA, IDB, RAA, Q, US: text,
HA, RA, RB, KA, CAS, CSA, SKA, K: message,
Inc : hash_func
const alice_server, server_bob, alice_bob, alice_server,
subs1, subs2, subs3: protocol_id
init State :=0
transition
1. State = 0 /\ Rcv(start) =|>
State' := 1  /\ DA'  :=new()
/\ UA' := Mul(DA'.Q)
/\ RAA' := new()
/\ HA' := H(RAA'.DA')
/\ RA' := Mul(HA'.Q)
/\ KA' := Mul(DA.US)
/\ CAS' := H(IDA.IDB.RA'.KA')
/\ Snd(IDA.IDB.RA'.CAS')
/\ secret({DA'}, subs1, {A,S})
2. State = 1 /\ Rcv(RB.CSA') =|>
State' := 2 /\ K' := Mul(HA.RB)
/\ SKA' := H(IDA.IDB.RA.RB.K')
end role
```

Fig. 4 Role specification for the user *A* (initiator) in *HLPSL*.

```
role bob (B, S, A : agent,
% H is hash function
H, Mul: hash_func, Snd, Rcv: channel(dy))
played_by B
def=
local State : nat,
DB, UB, IDA, IDB, RBB, Q, US: text,
HB, RB, RA, KB, CBS, CSB, SKA, K: message,
Inc : hash_func
const alice_server, server_bob, alice_bob, alice_server,
subs1, subs2, subs3: protocol_id
init State :=0
transition
1. State = 0 /\ Rcv(start) =|>
State' := 1  /\ DB' :=new()
/\ UB' :=Mul(DB'.Q)
/\ RBB' := new()
/\ HB' := H(RBB'.DB')
/\ RB' := Mul(HB'.Q)
/\ KB' := Mul(DB.US)
/\ CBS' := H(IDB.IDA.RB'.KB')    %line 50
/\ Snd(IDB.IDA.RB'.CBS')
/\ secret({DB'}, subs2, {B,S})
2. State = 1 /\ Rcv(RA.CSB') =|>
State' := 2 /\ K' := Mul(HB.RA)
/\ SKA' := H(IDA.IDB.RA.RB.K')
end role
```

Fig. 5 Role specification for the user B (responder) in *HLPSL*.

Theorem 1. *Under the assumption that the adversary can eavesdrop all the communicating messages over the public channel, the proposed protocol provides strong security protection on the private key of the user and server.*

Proof. We supposed that the adversary ($\mathcal{A}$) traps all the transmitting messages between the entities involved of the protocol during execution and tries to extract confidential parameter of the user and server such as private key. In order to get success, the $\mathcal{A}$ will face the following problems as follows.

(1) During execution of the authenticated key exchange phase, $\mathcal{A}$ traps $\langle ID_A, ID_B, R_A, C_{AS}\rangle$ from the public channel, where $R_A = H_A \cdot Q, H_A = H(r_A \| d_A), C_{AS} = H(ID_A \| ID_B \| R_A \| K_A)$ and r_A is the random number. It is noticeable that $\mathcal{A}$ cannot extract H_A from $R_A = H_A \cdot Q$ due to *ECDLP*. Additionally, $\mathcal{A}$ cannot extract the private key d_A of A due to non-invertibility property of the cryptographic one-way hash function. The parameter K_A is reliant on the private key of A and S and protected by the *ECDLP*.
(2) Resembling (1), $\mathcal{A}$ can eavesdrop $\langle ID_B, ID_A, R_B, C_{BS}\rangle$, and try to compute private key of B and S. However, the $\mathcal{A}$ will face same problem like (1).
(3) During round 3 of the proposed protocol, the $\mathcal{A}$ traps $\langle R_B, C_{SA}\rangle$ and $\langle R_B, C_{SB}\rangle$ from the public channel and tries to compute the confidential information of A and S, where $C_{SA} = H(ID_A \| ID_B \| R_A \| R_B \| K_A)$ and $C_{SB} = H(ID_B \| ID_A \| R_B \| R_A \| K_B)$. It is worth to note that $\mathcal{A}$ cannot extract any confidential information from $\langle C_{SA}, C_{SB}\rangle$ due to one-way hash function.

```
role server (S, A, B: agent,
% H is hash function
H, Mul: hash_func,
Snd, Rcv: channel(dy) )
played_by S
def=
local State : nat,
DS, UB, UA, IDA, IDB, Q, US: text,
HB, RB, RA, KB, SKB, KA, KAA, KBB, CSA, CSB: message,
Inc : hash_func
const alice_server, server_bob, alice_bob, alice_server,
subs1, subs2, subs3: protocol_id
init State :=0
transition
1. State = 0 /\ Rcv(IDA.IDB.RA'.KA') /\ Rcv(IDA. IDB. RB'.KB') =|>
State' := 1 /\ US' := Mul(DS.Q)
/\ KAA' := Mul(DS.UA)
/\ KBB'  :=Mul(DS.UB)
/\ CSA' := H(IDA.IDB.RA.RB.KAA')
/\ CSB' := H(IDB.IDA.RB.RA.KBB')
/\ Snd(RB.CSA')
/\ Snd(RA.CSB')
/\ secret({DS}, subs3, {S})
end role
```

Fig. 6 Role specification for the server S in *HLPSL*.

```
role session(A, S, B: agent,
H, Mul: hash_func)
def=
local SI, SJ, RI, RJ, TI, TJ: channel (dy)
composition
alice(A, S, B, H, Mul, SI, RI)
/\ server(A, S, B, H, Mul, SJ, RJ)
/\ bob(A, S, B, H, Mul, TI, TJ)
end role
role environment()
def=
const a, s, b: agent,
h,mul: hash_func,
ida, idb, ua, ub, da, db, ra, rb, ds, us, cas, cbs, csa, csb,
kaa, kbb, ha, hb, ka, kb, raa, rbb: text,
alice_server, server_bob, alice_bob, alice_server,
subs1, subs2, subs3: protocol_id
intruder_knowledge = {a, s, b, h, mul, cas, cbs, csa, csb, ra, rb}
composition
session( a, s, b, h, mul)
/\ session(s, a, b, h, mul)
/\ session(b, s, a, h, mul)
end role
goal
secrecy_of subs1
secrecy_of subs2
secrecy_of subs3
authentication_on alice_server_raa
authentication_on bob_server_rbb
end goal
environment()
```

Fig. 7 Role specification for the session, goal and environment in *HLPSL*.

```
% OFMC
 % Version of 2006/02/13
  SUMMARY
  SAFE
 DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
 PROTOCOL
 /home/avispa/web-interface-computation
/./tempdir/workfileEdDMf1.if
 GOAL
  as_specified
 BACKEND
  OFMC
 COMMENTS
 STATISTICS
  parseTime: 0.00s
  searchTime: 0.66s
  visitedNodes: 16 nodes
  depth: 6 plies
```

Fig. 8 Simulation results for the *OFMC* back-end.

The above description ensures that the proposed *3PAKE* protocol is secure against $\mathcal{A}$ for deriving the private keys of A, B and S. □

Theorem 2. *The proposed protocol is secured against the impersonation-of-initiator attack.*

Proof. Suppose that $\mathcal{A}$ wishes to impersonate A (initiator) to B. The secret key d_A of A is unknown to $\mathcal{A}$, so he can try to extract it from $U_A = d_A \cdot Q$, but, $\mathcal{A}$ cannot derive d_A from U_A due to the difficulties of *ECDLP*. Now, $\mathcal{A}$ selects two random integers $(r_A, d''_A) \in Z_q^*$, then computes $R''_A = H(r_A \| d''_A)$ and $K''_A = d''_A \cdot U_S$, and further sends the message $(ID_A, Request)$ and $(ID_A, ID_B, R''_A, C''_{SA})$ to B and S, respectively, where $C''_{AS} = H(ID_A \| ID_B \| R''_A \| K''_A)$. Upon receiving $(ID_A, ID_B, R''_A, C''_{SA})$, S computes $K_A = d_S \cdot U_A = d_A \cdot d_S \cdot Q$ and $\overline{C}_{AS} = H(ID_A \| ID_B \| R''_A \| K_A)$ and then verifies with received C''_{SA}. Therefore, S aborts the protocol and sends the *authentication-failed* message to B, because $\overline{C}_{AS} \neq C''_{SA}$. Thus the impersonation-of-initiator attack is infeasible to the proposed protocol. □

Theorem 3. *The proposed protocol is secured against the impersonation-of-responder attack.*

Proof. Assume that $\mathcal{A}$ tries to impersonate B (responder) to A. First, $\mathcal{A}$ selects two random integers $r_B, d''_B \in Z_q^*$ and then computes $R''_B = H(r_B \| d''_B)$ and $K''_B = d''_B \cdot U_S$. Then $\mathcal{A}$ sends the messages $(ID_B, Response)$ and $(ID_B, ID_A, R''_B, C''_{BS})$ to B and S, respectively, where $C''_{BS} = H(ID_B \| ID_A \| R''_B \| K''_B)$. Upon receiving $\mathcal{A}$'s message, S computes $K_B = d_S \cdot U_B$ and $\overline{C}_{BS} = H(ID_B \| ID_A \| R''_B \| K_B)$. Next, S compares the computed $\overline{C}_{BS}$ with received C''_{BS} and confirms that someone is impersonating B, because $\overline{C}_{BS}$ is not equal to C''_{BS}. Therefore, S sends an *authentication-failed* message to A. Thus, the proposed protocol has the ability to protect the impersonation-of-responder attack. □

Theorem 4. *The proposed protocol is secured against the parallel attack.*

Proof. To perform the parallel attack, $\mathcal{A}$ captures the previous protocol run message $(ID_A, ID_B, R_A, C_{AS})$, which was sent by A to S. In the current session, A sends the message $(ID_A, NewRequest)$ and $(ID_A, ID_B, R''_A, C''_{AS})$ to B and S, where $r''_A \in Z_q^*$, $R''_A = H(r''_A \| d_A) \cdot Q$ and $C''_{AS} = H(ID_A \| ID_B \| R''_A \| K_A)$. Now $\mathcal{A}$ captures the current session message $(ID_A, ID_B, R''_A, C''_{AS})$, and replies with the older session message $(ID_A, ID_B, R_A, C_{AS})$ to S. Upon receiving A's request, B sends the message $(ID_B, ID_A, R''_B, C''_{BS})$ to S, where $r''_B \in Z_q^*, R_B = H(r''_B \| d_B) \cdot Q$ and $C''_{BS} = H(ID_B \| ID_A \| R''_B \| K_B)$. Then S replies with the message (R''_B, C''_{SA}) and (R_A, C''_{SB}) to A and B, where $C''_{SA} = H(ID_A \| ID_B \| R_A \| R''_B \| K_A)$ and $C''_{SB} = H(ID_B \| ID_A \| R''_B \| R_A \| K_B)$. Now B computes the session key $SK = H(ID_A \| ID_B \| R_A \| R''_B \| K'')$, where $K'' = H(r_A \| d_A) \cdot H(r''_B \| d_B) \cdot Q$. At the same time, A computes $C^*_{SA} = H(ID_A \| ID_B \| R''_A \| R''_B \| K_A)$ and compares it with received C''_{SA}. Thus, A rejects the protocol transaction immediately because $C^*_{SA} \neq C''_{SA}$. In the same way, if $\mathcal{A}$ replies to B's message when A tries to establish a new communication with B, our proposed protocol has the ability to detect this attack. □

Theorem 5. *The proposed protocol is secured against the man-in-the-middle attack.*

Proof. Assume that an $\mathcal{A}$ wants to learn the session key SK by performing the man-in-the-middle attack (Menezes et al., 1996) to the proposed protocol. However, $\mathcal{A}$ cannot compute $K_A = d_A \cdot d_S \cdot Q$ and $K_B = dB \cdot d_S \cdot Q$ without A's/B's private key or S's private key. Hence, $\mathcal{A}$ selects a random number r''_C from Z_q^* and computes $R_C = H(r_C \| d_C)$ and $K_C = d_C \cdot U_S = d_C \cdot d_S \cdot Q$, where $\mathcal{A}$'s private key is d_C. Further, $\mathcal{A}$ intercepts the message $(ID_A, ID_B, R_A, C_{AS})$ and $(ID_B, ID_A, R_B, C_{BS})$ transmitted form A and B to S and modified to $(ID_A, ID_C, R_A, C_{AS})$ and $(ID_B, ID_C, R_B, C_{BS})$, and then sends them to S. Furthermore, $\mathcal{A}$ sends two concurrent

```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL
PROTOCOL
 /home/avispa/web-interface-computation/
./tempdir/workfileAcJN8I.if
GOAL
 As Specified
BACKEND
 CL-AtSe
STATISTICS
 Analysed   : 0 states
 Reachable  : 0 states
 Translation: 0.06 seconds
 Computation: 0.00 seconds
```

Fig. 9 Simulation results for the *CL-AtSe* back-end.

messages $(ID_C, ID_A, R_C, C_{CS1})$ and $(ID_C, ID_B, R_C, C_{CS2})$ to S, where $C_{CS1} = H(ID_C\|ID_A\|R_C\|K_C)$ and $C_{CS2} = H(ID_C\|ID_B\|R_C\|K_C)$. S assumes that $\mathcal{A}$ tries to establish the session key to A and B simultaneously. Now S performs the validity check on the received messages $(ID_A, ID_C, R_A, C_{AS})$ and $(ID_C, ID_A, R_C, C_{CS1})$ and $(ID_B, ID_C, R_B, C_{BS})$ and $(ID_C, ID_B, R_C, C_{CS2})$. For this purpose, S computes $\overline{C}_{AS} = H(ID_A\|ID_C\|R_A\|K_A)$, $\overline{C}_{BS} = H(ID_B\|ID_C\|R_B\|K_B)$ and checks $\overline{C}_{AS} = ?C_{AS}$ and $\overline{C}_{BS} = ?C_{BS}$. S then sends the *authentication-failed* message to A and B, because $\overline{C}_{AS} \neq C_{AS}$ and $\overline{C}_{BS} \neq C_{BS}$. Thus, the man-in-the-middle attack is impossible in the proposed protocol. □

Theorem 6. *The proposed protocol is secured against the known session-specific temporary information attack.*

Proof. In the proposed protocol, a session key $SK = H(ID_A\|ID_B\|R_A\|R_B\|K)$ is securely established in each session between A and B, where $K = H_A \cdot H_B \cdot Q$ is the partial session key. Assume that $\mathcal{A}$ learns two ephemeral secrets r_A and r_B by some means. However, $\mathcal{A}$ cannot generate $H_A = H(r_A\|d_A)$ and $H_B = H(r_B\|d_B)$ without A's private key d_A and B's private key d_B, so the partial session key K cannot be computed and thus, the resulting session key SK is still unknown to $\mathcal{A}$. In addition, $\mathcal{A}$ tries to derive $K = H_A \cdot H_B \cdot Q$ directly from the pair $(R_A, R_B) = (H_A \cdot Q, H_B \cdot Q)$. Furthermore, the computation of K is also infeasible due to difficulties of CDH problem. Thus, the proposed protocol is robust against the known session-specific temporary information attack. □

Theorem 7. *The proposed protocol is secured against the key offset attack.*

Proof. The key offset attack is one of the forms of man-in-the-middle attack. Suppose that the active $\mathcal{A}$ monitors the communication channel, e.g., he can modify, delete or delay the message in a session, and enforce the clients to agree upon a wrong session key which is not the one two entities agree on. Although this attack does not allow $\mathcal{A}$ to gain any knowledge about the agreed session key but two entities generates the wrong session key. This violates the key integrity property, which indicates that any accepted session key should depend only on inputs from the clients. $\mathcal{A}$ cannot generate $K_A = d_A \cdot d_S \cdot Q$ and $K_B = d_B \cdot d_S \cdot Q$, so the modification of C_{AS}/C_{BS} and C_{SA}/C_{SB} is not possible. However, if $\mathcal{A}$ modifies R_A and R_B, S can detect it by checking $\overline{C}_{AS} = ?C_{AS}$ and $\overline{C}_{BS} = ?C_{BS}$. Therefore, the proposed protocol can prevent the key offset attack. □

Theorem 8. *The proposed protocol is secured against the key-compromise impersonation attack.*

Proof. The key-compromise impersonation attack indicates that if the private key of A is known to $\mathcal{A}$ then he can impersonate B to A. However, our *3PAKE* protocol does not allow $\mathcal{A}$ to impersonate B to A. Assume that the private key d_A of A is compromised to $\mathcal{A}$ who wishes to impersonate B, then $\mathcal{A}$ must have a valid key $K_B = d_B \cdot U_S$. Otherwise, he cannot authenticate himself to S. It is possible if he knows B's/S's private key, but $\mathcal{A}$ failed to derive d_B/d_S form U_B/U_S due to infeasibility of *ECDLP*. Therefore, the proposed protocol protects the key-compromise impersonation attack. □

Theorem 9. *The proposed protocol is secured against the unknown key-share attack.*

Proof. The unknown key-share attack means, after the completion of the protocol session, A believed he shared a session key with B, but B unfortunately believed that he shared a session key with $\mathcal{A}$. In our protocol the identities are included in C_{AS}, C_{BS}, C_{SA} *and* C_{SB}, and these are validated by A, B and S. Therefore, A and B get confirmation that they share the session key with the original clients not with $\mathcal{A}$. □

Theorem 10. *The proposed protocol provides session key perfect forward security.*

Proof. The perfect forward security states that if the private key of one or more clients happens to be disclosed, the security of previously established session keys should not be compromised. Assume that the private keys d_A and d_B of A and B are revealed to $\mathcal{A}$, but he cannot find out the previous and/or future session keys from this disclosure. $\mathcal{A}$ can generate the session key $SK = H(ID_A\|ID_B\|R_A\|R_B\|K)$ if the partial session key $K = H_A \cdot H_B \cdot Q$ is known to him/her, and only it can be computed from $(R_A, R_B) = (H_A \cdot Q, H_B \cdot Q)$ provided that $\mathcal{A}$ possesses a polynomial time algorithm that can solve the CDH problem. Furthermore, the session key SK can be compromised if r_A and r_B are known to $\mathcal{A}$. To do so, $\mathcal{A}$ tries to derive them from $R_A = H(r_A\|d_A)$ and $R_B = H(r_B\|d_B)$ but, it is also impossible due to difficulties of *ECDLP* and the "one-way" property of the hash function. Thus, the proposed protocol provides the perfect forward security. □

Theorem 11. *The proposed protocol provides the known-key security.*

Proof. A unique and common session key is generated in each session and disclosure of one session key should not compromise other session keys. Assume that the current session key $SK = H(ID_A\|ID_B\|R_A\|R_B\|K)$ is leaked to $\mathcal{A}$, however, he cannot compute all the previously established session keys from this disclosure. Since $\mathcal{A}$ cannot extract r_A/r_B and d_A/d_B from $R_A = H(r_A\|d_A) \cdot Q$ and $R_B = H(r_B\|d_B) \cdot Q$ as this would be equivalent to solving the *ECDLP* and the "one-way" property of the hash function. In addition, a hash function is used to generate the session key therefore, $K = H_A \cdot H_B \cdot Q$ cannot be extracted from the disclosed session key $SK = H(ID_A\|ID_B\|R_A\|R_B\|K)$. Thus, the proposed protocol does not allow the disclosure of past session keys from the compromised session key. □

Theorem 12. *The proposed protocol no key control property of the session key.*

Proof. The session key is generated in each session mutually by both the clients, and therefore, a single party cannot control the outcome of the session key to a pre-selected value or lie within a small set of values. □

Table 2 Computation cost and functionality comparison of proposed protocol with existing related protocols.

Attribute	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
Yang and Chang (2009)	×	×	×	×	√	√	×	×	√	√	√	√
Pu et al. (2009)	×	×	√	×	×	√	×	√	√	√	√	√
Tan (2010a)	×	×	√	×	√	√	×	√	√	√	√	×
Tan (2010b)	√	√	√	√	√	√	×	√	√	√	√	×
He et al. (2013)	√	√	√	√	√	√	×	√	√	√	√	×
Proposed	√	√	√	√	√	√	√	√	√	√	√	√

√: resist the attack or meet the security criteria; ×: attack is possible or violate the security criteria; A_1: impersonation-of-initiator attack; A_2: impersonation-of-responder attack; A_3: parallel session attack; A_4: man-in-the-middle attack; A_5: key offset attack; A_6: key-compromise Impersonation attack; A_7: known session-specific temporary information; A_8: unknown key share attack; A_9: provide key control; A_{10}: provide known key security; A_{11}: provide perfect forward secrecy; A_{12}: free from clock synchronization problem.

Table 3 Execution time (ms) of different cryptographic operation.

Notation	Description and execution time (ms)
T_{PM}	Time complexity for executing the elliptic curve point multiplication $T_{PM} = 17.10$ ms
T_{SE}	Time complexity for executing the symmetric en/decryption $T_{SE} = 5.60$ ms
T_H	Time complexity for executing the hash function, $T_H = 0.32$ ms

Table 2 shows the security comparisons of the proposed protocol and some other relevant protocols. It is obvious that the proposed protocol supports all the related security properties.

8. Performance analysis

In this section, we compared the computation cost efficiency of the proposed protocol with Yang and Chang's protocol (Yang and Chang, 2009), Pu et al.'s protocol (Pu et al., 2009), Tan's protocol (Tan, 2010a), Tan's protocol (Tan, 2010b) and He et al.'s protocol (He et al., 2013). We considered the computational complexity of different cryptographic operations as executed in He et al. (2014) and Schneier (1996). In Table 3, we listed different computational notations and their execution time in milliseconds.

It is proven in Schneier (1996) that one symmetric encryption/decryption operation is at least 100 times faster than one asymmetric encryption/decryption operation, and one hashing operation is at least 10 times faster than a symmetric encryption/decryption in software implementation. The experimental results given in Table 3 are executed on a four-core 3.2 GHz machine with 8 GB memory, and the results were averaged over 300 randomized simulation runs. The experimental evaluations were implemented on the simulator written in *MATLAB*. The Table 4 provides a comparative study of the proposed protocol with other protocols. The computation cost of the proposed protocol is lesser than the protocols in Yang and Chang (2009), Pu et al. (2009), Tan (2010a), Tan (2010b), and He et al. (2013). For better understanding, we have given the computation cost comparison in Fig. 10.

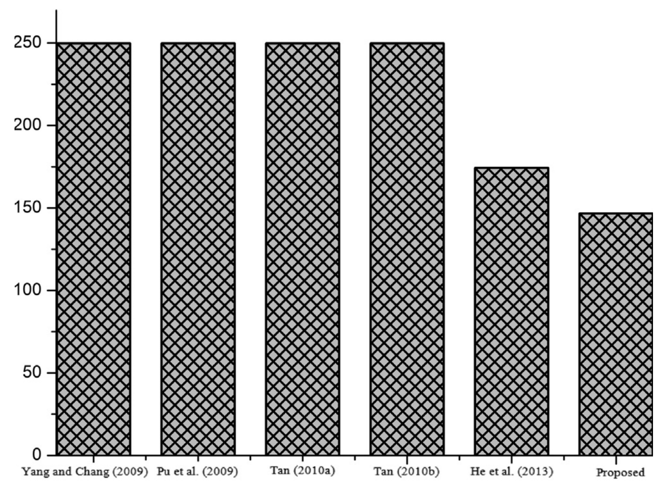

Fig. 10 Execution time (ms) of different *3PAKE* protocols including ours.

9. Conclusions

In order to provide the security enhancement, Pu et al. designed an improved *3PAKE* protocol over Yang and

Table 4 Computation cost (ms) comparison of proposed protocol with existing related protocols.

Protocol	User *A*	User *B*	Server *S*	Total cost
Yang and Chang (2009)	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$2T_{PM} + 4T_{SE} \approx$ 56.40 ms	249.80 ms
Pu et al. (2009)	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$2T_{PM} + 4T_{SE} \approx$ 56.40 ms	249.80 ms
Tan (2010a)	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$2T_{PM} + 4T_{SE} \approx$ 56.40 ms	249.80 ms
Tan (2010b)	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$5T_{PM} + 2T_{SE} \approx$ 96.70 ms	$2T_{PM} + 4T_{SE} \approx$ 56.40 ms	249.80 ms
He et al. (2013)	$3T_{PM} + 3T_H \approx$ 52.26 ms	$3T_{PM} + 3T_H \approx$ 52.26 ms	$3T_{PM} + 3T_H \approx$ 69.68 ms	174.20 ms
Proposed	$3T_{PM} + 2T_H \approx$ 54.58 ms	$3T_{PM} + 4T_H \approx$ 54.58 ms	$2T_{PM} + 4T_H \approx$ 37.48 ms	146.64 ms

Chang's *3PAKE* protocol and then Tan et al. proposed another improvement over Pu et al.'s *3PAKE* protocol to resist the impersonation-of-initiator attack, impersonation-of-responder attack and parallel attack. We again examined Tan's *3PAKE* protocol and pointed out that still the protocol is not secure against the known session-specific temporary attack and cannot withstand the clock synchronization problem. In addition, Tan et al.'s *3PAKE* protocol has high computation cost due to the involvement of the additional elliptic curve scalar point multiplications and symmetric cryptosystem. We then designed a computation efficient *3PAKE* protocol for mobile commerce environment to resolve the security pitfalls of the Tan's *3PAKE* protocol. The security analysis on our *3PAKE* protocol confirmed that the proposed protocol not only renovates Tan's *3PAKE* protocol, but it also secure against other known attacks. Additionally, the simulation results on *AVISPA* software confirmed that the proposed *3PAKE* protocol is secure under *OFMC* and *CL-AtSe* backends. The performance analysis ensured that the proposed *3PAKE* protocol is computationally efficient than other existing works.

Acknowledgements

SK Hafizul Islam is supported by the Outstanding Potential for Excellence in Research and Academics (OPERA) award, Birla Institute of Technology and Science (BITS) Pilani, Pilani Campus, Rajasthan, India. The authors extend their sincere appreciations to the National Natural Science Foundation of China under Grant No. 61300220, the Research Fund of the State Key Laboratory of Software Development Environment, BUAA under Grant no. SKLSDE-2014KF-02, and the China Postdoctoral Science Foundation Funded Project under Grant No. 2014M550590.

References

Advanced Encryption Standard, 2001. Available at <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

Amin, R., Biswas, G.P., 2015. A novel user authentication and key agreement protocol for accessing multi-medical server usable in TMIS. J. Med. Syst. 39 (3), 1–17.

AVISPA Web Tool. Avaliable at: <http://www.avispa-project.org/web-interface/expert.php/>. Accessed on April, 2015

AVISPA. Automated validation of internet security protocols and applications. Avaliable at: <http://www.avispa-project.org/>

Canetti, R., Krawczyk, H., 2001. Analysis of key exchange protocols and their use for building secure channels. In: Proceedings of the Advances in Cryptology (Eurocrypt'01). Springer-Verlag, LNCS, pp. 453–474.

Chang, C.C., Chang, Y.F., 2004. A novel three-party encrypted key exchange protocol. Comput. Stand. Interfaces 26 (5), 472–476.

Chen, T.H., Lee, W.B., Chen, H.B., 2008a. A round-and computation-efficient three-party authenticated key exchange protocol. J. Syst. Softw. 81 (9), 1581–1590.

Chen, H.B., Chen, T.H., Lee, W.B., Chang, C.C., 2008b. Security enhancement for a three-party encrypted key exchange protocol against undetectable on-line password guessing attacks. Comput. Stand. Interfaces 30, 95–99.

Cheng, Z., Nistazakis, M., Comley, R., Vasiu, L., 2005. On the indistinguishability-based security model of key agreement protocols-simple cases. Cryptology ePrint Archieve, Report 2005/129.

Chung, H.R., Ku, W.C., 2008. Three weaknesses in a simple three-party key exchange protocol. Inf. Sci. 178 (1), 220–229.

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

Gong, L., 1992. A security risk of depending on synchronized clocks. ACM Operating Syst. Rev. 26 (1), 49–53.

Guo, H., Li, Z., Mu, Y., Zhang, X., 2008. Cryptanalysis of simple three-party key exchange protocol. Comput. Secur. 28 (1–2), 16–21.

He, D., Chen, Y., Chen, J., 2013. An Id-based three party authenticated key exchange protocol using elliptic curve cryptography for mobile commerce environments. Arab. J. Sci. Eng. 38 (8), 2055–2061.

He, D., Kumar, N., Lee, J.-H., Sherratt, R.S., 2014. Enhanced Three-factor Security Protocol for Consumer USB Mass Storage Devices. IEEE Trans. Consum. Electron. 60 (1), 30–37.

Islam, S.H., 2014a. Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps. Nonlinear Dyn. 78 (3), 2261–2276.

Islam, S.H., 2014b. Design and analysis of an improved smartcard based remote user password authentication scheme. Int. J. Commun Syst. http://dx.doi.org/10.1002/dac.2793.

Islam, S.H., 2014c. A provably secure ID-based mutual authentication and key agreement scheme for mobile multi-server environment without ESL attack. Wireless Pers. Commun. 79, 1975–1991.

Islam, S.H., Biswas, G.P., 2012. An improved pairing-free identity-based authenticated key agreement protocol based on ECC. Proc. Int. Conf. Commun. Technol. Syst. Des. Procedia Eng. 30, 499–507.

Islam, S.H., Biswas, G.P., 2013. An efficient and secure strong designated verifier signature scheme without bilinear pairings. J. Appl. Math. Inf. 31 (3–4), 425–441.

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.

Islam, S.H., Khan, M.K., 2014. Cryptanalysis and improvement of authentication and key agreement protocols for telecare medicine information systems. J. Med. Syst. 38 (10), 1–16.

Islam, S.H., Khan, M.K., Obaidat, M.S., Muhaya, F.T.B., 2015. Provably secure and anonymous password authentication protocol for roaming service in global mobility networks using extended chaotic maps. Wireless Pers. Commun. http://dx.doi.org/10.1007/s11277-015-2542-8.

Koblitz, N., 1987. Elliptic curve cryptosystem. Math. Comput. 48, 203–209.

Lee, T.F., Hwang, T., 2010. Simple password-based three-party authenticated key exchange without server public keys. Inf. Sci. 180 (9), 1702–1714.

Lee, T.F., Hwang, T., Lin, C.L., 2004. Enhanced three-party encrypted key exchange without server public keys. Comput. Secur. 23 (7), 571–577.

Lin, C., Sun, H.M., Hwang, T., 2000. Three-party encrypted key exchange: attacks and a solution. ACM Operating Syst. Rev. 34 (4), 12–20.

Lin, C., Sun, H.M., Steiner, M., Hwang, T., 2001. Three-party encrypted key exchange without server public-keys. IEEE Commun. Lett. 5 (12), 497–499.

Lu, R., Cao, Z., 2006. Simple three-party key exchange protocol. Comput. Secur. 26, 94–97.

Mandt, T.K., 2006. Certificate less authenticated two-party key agreement protocols (Master's thesis), Gjvik University College, Department of Computer Science and Media Technology.

Menezes, A.J., Orschot, P.C., Vanstone, S.A., 1996. Hand-Book of Applied Cryptography. CRC Press.

Message Digest Algorithm, 1992. Available at <http://people.csail.mit.edu/rivest/Rivest-MD5.txt>

Nam, J., Kim, S., Won, D., 2006. Attack on the Sun-Chen-Hwang's three-party key agreement protocols using passwords. IEICE Trans. Fundam. E89-A (1), 209–212.

Nose, P., 2011. Security weaknesses of authenticated key agreement protocols. Inf. Process. Lett. 11 (14), 687–696.

Phan, R.C.W., Yau, W.C., Goi, B.M., 2008. Cryptanalysis of simple three-party key exchange protocol (S-3PAKE). Inf. Sci. 178, 2849–2856.

Pu, Q., Zhao, X., Ding, J., 2009. Cryptanalysis of a three-party authenticated key exchange protocol using elliptic curve cryptography. In: Proceedings of the International Conference on Research Challenges in Computer Science, pp. 7–10.

Reddy, P.V., Padmavathamma, M., 2007. An authenticated key exchange protocol in elliptic curve cryptography. J. Discrete Math. Sci. Cryptogr. 10 (5), 697–705.

Schneier, B., 1996. Applied Cryptography, Protocols, Algorithms, and Source Code, second ed. Wiley.

Sun, H., Chen, B., Hwang, T., 2005. Secure key agreement protocols for three-party against guessing attacks. J. Syst. Softw. 75, 63–68.

Tan, Z., 2010a. An enhanced three-party authentication key exchange protocol for mobile commerce environments. J. Commun. 5 (5), 436–443.

Tan, Z., 2010b. An Improvement on a three party authentication key exchange protocol using elliptic curve cryptography. J. Convergence Inf. Technol. 5 (4), 120–129.

Yang, J.H., Chang, C.C., 2009. An efficient three-party authenticated key exchange protocol using elliptic curve cryptography for mobile-commerce environments. J. Syst. Softw. 82, 1497–1502.

Yang, J., Seo, C., Cho, J., 2007. A three party authenticated key exchange scheme smartcard using elliptic curve cryptosystem for secure key exchange in wireless sensor network. In: Proceedings of the International Symposium on Consumer Electronics, pp. 1–6.

Yoon, E.J., Yoo, K.Y., 2008. Improving the novel three-party encrypted key exchange protocol. Comput. Stand. Interfaces 30, 309–314.