ARTICLE IN PRESS

Computers and Electrical Engineering xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A new efficient authenticated multiple-key exchange protocol from bilinear pairings ☆

Mohammad Sabzinejad Farash [a,*], Mahmoud Ahmadian Attari [b], Reza Ebrahimi Atani [c], Mohamadreza Jami [d]

[a] *Department of Mathematics and Computer Sciences, Tarbiat Moallem University, P.O. Box 15815-3587, Tehran, Iran*
[b] *Faculty of Electrical and Computer Engineering, K.N. Toosi University, P.O. Box 16315-1355, Tehran, Iran*
[c] *Department of Computer Engineering, University of Guilan, P.O. Box 3756, Rasht, Iran*
[d] *Faculty of Computer Science, Payame Noor University, P.O. Box 19395-3697, Tehran, Iran*

A R T I C L E   I N F O

Article history:
Received 25 August 2011
Received in revised form 10 September 2012
Accepted 11 September 2012
Available online xxxx

A B S T R A C T

The main goal of this paper is to analyze the security of a multiple-key agreement protocol and to show that this protocol is vulnerable to forgery attack. In order to address the security weakness, a new efficient multiple-key agreement protocol is proposed. The proposed protocol is based on bilinear pairings and utilizes a novel signature scheme without one-way hash functions. In contrast to related protocols which are based on bilinear pairings, in the proposed protocol, these pairings are not utilized for the included signature scheme, but they are only used for session keys. As a result, the computational complexity of the proposed protocol is decreased by 50% compared to that of the existing protocols. Another contribution of the proposed protocol is an increase of the number of the session keys up to $(n^2(n+1)^2/4)$, for $n \geqslant 2$ random numbers, regarding the bilinear property of the pairings. Finally, the simulation results from AVISPA tools confirm the security analysis of the proposed protocol.

1. Introduction

Information systems and the information that they contain are considered to be major assets that require protection. Cryptography is often used to protect information from unauthorized disclosure, to detect modification, and to authenticate the identity of system users. Cryptography is particularly useful when data transmission or authentication occurs over communication networks for which physical means of protection are often cost-prohibitive or even impossible to implement. Cryptography also provides a layer of protection for storing data (in addition to physical and computer security access controls) against insiders who may have physical and possibly logical (e.g., system administrator) access, but not the authorization, to know or modify the information.

Cryptographic techniques use cryptographic keys that are managed and protected throughout their life cycles by the cryptographic key management systems. Effectively implemented cryptography can reduce the scope of the information management problem from the need to protect large amounts of information to the need to protect only the cryptographic keys. One of the significant components of managing cryptographic keys is how to exchange or distribute secret keys between two or more participants who want to establish a secure communication over an insecure channel. For this purpose,

☆ Reviews processed and approved for publication by Editor-in-Chief Dr. Manu Malek.
* Corresponding author.
E-mail addresses: sabzinejad@tmu.ac.ir (M.S. Farash), mahmoud@eetd.kntu.ac.ir (M.A. Attari), rebrahimi@guilan.ac.ir (R.E. Atani), mrjami@pnu.ac.ir (M. Jami).

0045-7906/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2012.09.004

key agreement protocols have been widely proposed and used. Key agreement protocols allow two or more entities to establish a shared secret key to be used in securing subsequent communication over an insecure channel.

Key agreement protocols can be classified in terms of different aspects (e.g., the number of participating entities, exchanged messages, agreed keys, and so on). Based on the number of participating entities, these protocols are divided into two categories: two-party and group. Regarding the number of exchanged messages, they are divided into two-pass, three-pass and so on. Finally, based on the number of agreed keys at the end of an execution, key agreement protocols are divided into two categories: single-key and multiple-key. It should be noted that using any of these types or a combination of them depends on the required applications.

To design key agreement protocols, security and efficiency aspects should be taken into consideration. For the security aspect, the secret key agreed at the end of the protocol should not be revealed to any entity other than the participants in the protocol. For efficiency, it should also be noted that computational and communication costs of the key agreement protocol are optimum. For example, in the context of wireless networks, management of communication resources due to the limited channel bandwidth is very important. Therefore, key agreement protocols to be used in these networks should be efficient in terms of communication costs as well as security.

In multiple-key agreement protocols, several secret keys are shared in each execution of the protocol. Therefore, on the condition that computation and communication resources are restricted, making use of such multiple-key protocols would be very useful and effective. For this reason, design and analysis of such protocols have attracted the attention of many researchers and professionals in the field of network security. Accordingly, the main motivation of this paper is to present a multiple-key agreement protocol which has made innovations in the number of keys as well as in the maintenance of security and efficiency.

1.1. Related works

Diffie and Hellman [1] were the first that proposed an efficient two-party key agreement protocol in 1976. However, the main security flaw of the Diffie–Hellman key agreement protocol is its weak resistance to the man-in-the-middle attack as it does not provide user authentication. A simple solution would be combining a key agreement protocol with a digital signature scheme to obtain an authenticated key agreement protocol. Typical key agreement protocols share one secret key at the end of each run. To establish several shared secret keys, it is necessary to run the protocol several times. Consequently, this leads to imposing high communication and computational costs. To overcome this problem, multiple-key agreement protocols are proposed.

The multiple-key agreement protocols enable two or more entities to establish multiple common secret keys in a single round of message exchange. The first multiple-key agreement protocol, introduced by Harn and Lin in 1998 [2], uses a signature without using a one-way hash function. Since then, many multiple-key agreement protocols [4–9] have been presented using the idea of the signature scheme without one-way hash functions. An application of such protocol is proposed by Wenmin et al. [10] in which a multiple-key agreement protocol is used for mobile networks. The multiple-key agreement protocols are also developed on bilinear pairings.

Bilinear pairings can be used to transport the discrete logarithm problem on a certain class of elliptic curves over a finite field to the discrete logarithm problem on a smaller finite field. They were originally brought to the cryptographic community by Menezes et al. with their MOV attack [11]. Joux [12] used the pairings to propose the first one round tripartite key agreement protocol in 2000. Since then, bilinear pairings due to the merits have become important tools to construct cryptographic schemes. Therefore, many pairing-based authenticated key agreement protocols have been proposed [13–19].

Multiple-key agreement protocols based on bilinear pairings were originally proposed by Lee et al. [20] in 2008. However, Vo et al. [21] showed that it is vulnerable to an impersonation attack and proposed an improved protocol. Next, Farash et al. [22] showed that Vo et al.'s protocol is insecure against a kind of forgery attack and reflection attack. Also, Cheng and Ma [23] showed that the Vo et al.'s protocol is vulnerable to reflection attack and proposed an improved protocol.

1.2. Our contribution

In general, the forgery attack in which an active adversary attempts to make a valid signature on a message instead of a legitimate user is a threat on digital signature schemes. However, it can be extended to the cryptographic application of signature schemes such as authenticated key agreement protocols in which a signature is utilized for user authentication. Since the multiple-key agreement protocols proposed in [20,21,23] are authenticated key agreement and use signatures for authentication, it is necessary to make them robust against the forgery attacks. In this paper, we show that, like Vo et al.'s protocol, Cheng and Ma's protocol is also insecure against a forgery attack which is due to the weakness of its signature scheme. Then, we propose an enhanced and efficient authenticated multiple-key agreement protocol using bilinear parings which satisfies all known security requirements.

Although the theoretical analysis of cryptographic protocols is normally used to verify the security attributes in the design, it is not sufficient and simulation tools must also be employed to verify all the security requirements of the protocol. AVISPAs (Automated Validation of Internet Security Protocols and Applications) tools [24] are strong simulation engines for automated security analysis of cryptographic protocols. In this paper, the AVISPA tools are also used for verification of the proposed protocol.

1.3. Outline

The remainder of the paper is organized as follows: preliminaries and definitions are discussed in Section 2. Section 3 reviews the Cheng and Ma's protocol. The proposed forgery attack against Cheng and Ma's protocol is performed in Section 4. In Section 5, an improved protocol is proposed. The security analysis of the proposed protocol is discussed in Section 6. Section 7 presents the computational analysis of our protocol and comparisons. Finally, Section 8 concludes the paper.

2. Preliminaries and definitions

Hereafter, we use the notations showed in Table 1 to describe the protocols and our analysis of them.

Definition 1. Bilinear Pairings. Let $\mathbb{G}_1$ be a cyclic additive group generated by P, whose order is a prime q, and let $\mathbb{G}_2$ be a cyclic multiplicative group of the same order q. Let the Discrete Logarithm Problem (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ be hard. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfies the following three properties [22]:

1. *Bilinear:* For $a, b \in \mathbb{Z}_q^*, P, Q \in \mathbb{G}_1$

   $$(aP, bQ) = e(P, Q)^{ab}$$

2. *Non-Degenerate:* There exists $P, Q \in \mathbb{G}_1$ such that $e(P,Q) \neq 1$.
3. *Computable:* There is an efficient algorithm to compute $e(P,Q)$ for all $P, Q \in \mathbb{G}_1$.

Security of pairing-based cryptosystems depend on the intractability of solving Bilinear Diffie–Hellman Problem (BDHP) that was introduced by Boneh and Franklin [25] as follows:

Definition 2. Bilinear Diffie–Hellman Problem (BDHP). For the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, given P, aP, bP and cP, compute $e(P,P)^{abc}$, where a, b, c are randomly chosen from $\mathbb{Z}_q^*$.

3. Review of Cheng and Ma's protocol

In this section, we briefly review Cheng and Ma's protocol. Suppose that $\mathbb{G}$ is a cyclic additive group generated by P of the prime order q. Suppose that, Alice and Bob want to share several secret keys. First, each of them generates a long-term private/public key-pair as follows: Alice chooses a random number $x_A \in \mathbb{Z}_q^*$, as her long-term private key and then computes her long-term public key $Y_A = x_AP$. Similarly, $x_B \in \mathbb{Z}_q^*$ and $Y_B = x_BP$ are also Bob's long-term private and public keys, respectively. Then, Alice and Bob establish several secret keys as follows:

Step 1. Alice $\rightarrow$Bob: $\{T_{A1}, T_{A2}, S_{A1}, S_{A2}, \mathrm{Cert}(Y_A)\}$.
First, Alice chooses two random numbers $r_{A1}, r_{A2} \in_R \mathbb{Z}_q^*$ and computes $T_{A1} = r_{A1}P$ and $T_{A2} = r_{A2}P$, then signs (T_{A1}, T_{A2}) as $S_{A1} = (r_{A1}k_{A1} + r_{A2}k_{A2})T_{A1} + x_AY_B$ and $S_{A2} = r_{A1}r_{A2}Y_B$. Finally, Alice sends the message $\{T_{A1}, T_{A2}, S_{A1}, S_{A2}, \mathrm{Cert}(Y_A)\}$ to Bob, where $\mathrm{Cert}(Y_A)$ is the certificate of Alice's long-term public key signed by a trusted party.

Step 2. Bob $\rightarrow$Alice: $\{T_{B1}, T_{B2}, S_{B1}, S_{B2}, \mathrm{Cert}(Y_B)\}$.
Bob also chooses two random numbers $r_{B1}, r_{B2} \in_R \mathbb{Z}_q^*$ and computes $T_{B1} = r_{B1}P$ and $T_{B2} = r_{B2}P$, then signs (T_{B1}, T_{B2}) as $S_{B1} = (r_{B1}k_{B1} + r_{B2}k_{B2})T_{B1} + x_BY_A$ and $S_{B2} = r_{B1}r_{B2}Y_A$. Finally, he sends the message $\{T_{B1}, T_{B2}, S_{B1}, S_{B2}, \mathrm{Cert}(Y_B)\}$ to Alice, where $\mathrm{Cert}(Y_B)$ is the certificate of Bob's long-term public key signed by a trusted party.

Table 1
The notations.

Notation	Definition
x_A, x_B	Long-term private key of Alice and Bob, respectively
Y_A, Y_B	Long-term public key of Alice and Bob, respectively
$\mathbb{G}$	An algebraic group
q	A random number
$\mathbb{Z}_q$	The residues mod q
$\mathbb{Z}_q^*$	The non-zero residues mod q
E	Elliptic curve
P	Base point of E
$e(.,.)$	The bilinear pairing
r_A, r_B	Short-term private key of Alice and Bob, respectively
T_A, T_B	Short-term public key of Alice and Bob, respectively
k_A, k_B	x-Coordinate of $T_A$and T_B, respectively
S_A, S_B	Digital signature of Alice and Bob, respectively
K	Session key

Step 3. Upon receiving the message $\{T_{B1}, T_{B2}, S_{B1}, S_{B2}, \mathrm{Cert}(Y_B)\}$, Alice verifies the signatures S_{B1} and S_{B2} by the following equations:

$$e(S_{B1}, P) \overset{?}{=} e(\{k_{B1}T_{B1} + k_{B2}T_{B2}\}, T_{B1}) \cdot e(Y_A, Y_B)$$
$$e(S_{B2}, P) \overset{?}{=} e(T_{B1}, T_{B2})^{x_A}$$

If these equations hold, she computes session keys as follows:

$$K_1 = e(r_{A1}T_{B1}, \{x_AT_{B1} + r_{A1}Y_B\})$$
$$K_2 = e(r_{A1}T_{B2}, \{x_AT_{B2} + r_{A1}Y_B\})$$
$$K_3 = e(r_{A2}T_{B1}, \{x_AT_{B1} + r_{A2}Y_B\})$$
$$K_4 = e(r_{A2}T_{B2}, \{x_AT_{B2} + r_{A2}Y_B\})$$

Step 4. Upon receiving the message $\{T_{A1}, T_{A2}, S_{A1}, S_{A2}, \mathrm{Cert}(Y_A)\}$, Bob verifies the signature S_{A1} and S_{A2} by the following equations:

$$e(S_{A1}, P) \overset{?}{=} e(\{k_{A1}T_{A1} + k_{A2}T_{A2}\}, T_{A1}) \cdot e(Y_B, Y_A)$$
$$e(S_{A2}, P) \overset{?}{=} e(T_{A1}, T_{A2})^{x_B}$$

If these equations hold, he computes session keys as follows:

$$K_1 = e(r_{B1}T_{A1}, \{x_BT_{A1} + r_{B1}Y_A\})$$
$$K_2 = e(r_{B2}T_{A1}, \{x_BT_{A1} + r_{B2}Y_A\})$$
$$K_3 = e(r_{B1}T_{A2}, \{x_BT_{A2} + r_{B1}Y_A\})$$
$$K_4 = e(r_{B2}T_{A2}, \{x_BT_{A2} + r_{B2}Y_A\})$$

4. The proposed attack on Cheng and Ma's protocol

4.1. The proposed forgery attack

As mentioned in Section 1, for a forgery attack, an active adversary attempts to make a valid signature on a message instead of a legitimate user without knowing the user's private key. In this section, we aim to propose a forgery attack on Cheng and Ma's protocol. In this attack, the participant Bob who received a signed message from the participant Alice, attempts to forge an Alice's signature as follows:

- Selects two random numbers $r'_{A1}, r'_{A2} \in \mathbb{Z}_q^*$,
- computes $T'_{A1} = r'_{A1}P$ and $T'_{A2} = r'_{A2}P$, and
- computes the signature $S'_{A1} = \{r'_{A1}k'_{A1} + r'_{A2}k'_{A2}\}T'_{A1} + x_BY_A$ and $S'_{A2} = r'_{A1}r'_{A2}Y_B$.

Thus, Bob can claim that the forged message $\{T'_{A1}, T'_{A2}, S'_{A1}, S'_{A2}, \mathrm{Cert}(Y_A)\}$ is sent by Alice. He can convince a judge or an arbiter of the system that the forged signature $\{S'_{A1}, S'_{A2}\}$ is computed by Alice. The arbiter checks whether $e(S'_{A1}, P) = e(\{k'_{A1}T'_{A1} + k'_{A2}T'_{A2}\}, T'_{A1}) \cdot e(Y_A, Y_B)$ and $e(S'_{A2}, P) = e(T'_{A1}, T'_{A2})^{x_B}$ hold or not. These equations are justified as follows:

$$\begin{aligned} e(\{k'_{A1}T'_{A1} + k'_{A2}T'_{A2}\}, T'_{A1}) \cdot e(Y_A, Y_B) &= e(\{k'_{A1}r'_{A1}P + k'_{A2}r'_{A2}P\}, r'_{A1}P) \cdot e(Y_A, x_BP) \\ &= e(\{k'_{A1}r'_{A1}P + k'_{A2}r'_{A2}P\}r'_{A1}, P) \cdot e(x_BY_A, P) \\ &= e(\{\{k'_{A1}r'_{A1} + k'_{A2}r'_{A2}\}r'_{A1}P + x_BY_A\}, P) = e(\{\{k'_{A1}r'_{A1} + k'_{A2}r'_{A2}\}T'_{A1} + x_BY_A\}, P) \\ &= e(S'_{A1}, P) \end{aligned} \tag{1}$$

and

$$e(S'_{A2}, P) = e(r'_{A1}r'_{A2}Y_B, P) = e(r'_{A1}r'_{A2}x_BP,\ P) = e(r'_{A1}x_BP,\ r'_{A2}P) = e(T'_{A1}, T'_{A2})^{x_B} \tag{2}$$

Thus, the arbiter would believe that the forged message is actually signed by Alice. In the same way, Alice also can forge a signature instead of Bob. In what follows, we propose a practical mode of the proposed forgery attack.

4.2. The other mode of the proposed attack

As reported in the literature [26], the users' private keys are commonly stored in smart cards, tokens and other security modules. The vulnerability of these storage media is shown repeatedly and several attacks such as side channel attacks [27–30] were reported for revealing the secret stored in them. Therefore, disclosing the users' private key is a reasonable and practicable assumption.

Table 2
Cryptographic operation time (in ms).

T_M	T_A	T_S	T_P
0.23	1.14	6.41	20.37

Suppose that, Bob's private key is compromised. Of course, an adversary who has obtained Bob's private key can easily impersonate him. However, it is important that the adversary cannot impersonate another participant (e.g., Alice) to Bob. Nevertheless, for the Cheng and Ma's protocol, an adversary who has Bob's private key can impersonate not only Bob, but also other users to Bob. To do so, he can easily compute the message $\{T'_{A1}, T'_{A2}, S'_{A1}, S'_{A2}, \mathsf{Cert}(Y_A)\}$, on behalf of Alice, as indicated in the later subsection, and send it to Bob. Upon receiving this message, Bob verifies the validity of S'_{A1} and S'_{A2} which are acceptable as shown in (1) and (2). Consequently, Bob would believe that the message is generated and sent by Alice.

4.3. The complexity of the proposed attack

This subsection aims to compute the complexity of the proposed attack by obtaining the running time of the computational components of the attack. Thus, we obtain the computational complexity using MIRACL (Multiprecision Integer and Rational Arithmetic C/C++Library) [31] which is a standard cryptographic library. The hardware platform is a PIV 3 GHz processor with 512 MB memory and the Windows XP operating system.

For the pairing-based protocols, to achieve 1024-bit level security, we used the pairing defined over the supersingular elliptic curve $E/F_p{:}y^2 = x^3 + x$ with embedding degree $r = 2$. q is a 160-bit prime $q = 2^{159} + 2^{17} + 1$ and p a 512-bit prime satisfying $p + 1 = 12qr$. The running times are listed in Table 2 where T_S,T_A,T_M and T_P stand for running times of scalar multiplication, point addition, modular multiplication and pairing, respectively. As can be clearly seen in Section 4.1, to perform the forgery attack on Cheng and Ma's protocol, the computations needed for an adversary are as little as the computations needed for a legal participant in a session of the protocol. Therefore, the complexity of the attack is equal to the computational cost of the protocol which is $2T_M + 1T_A + 4T_S$. Thus, as a result of Table 2, the time complexity of the forgery attack is 27.24 ms which is easily achievable.

Note that, the time consumed to choose random numbers $r'_{A1}, r'_{A2} \in \mathbb{Z}_q^*$ as compared to other computations is negligible and ignored for the time complexity of the proposed attack.

5. The proposed multiple-key agreement protocol

As mentioned earlier, many attempts have been made to design multiple-key agreement protocols without using one-way hash functions. However, each one of them is shown to be insecure or inefficient. Therefore, designing a secure and efficient multiple-key protocol without using one-way hash functions is a good motivation. Thus, this section aims to propose such a protocol to achieve desirable security and efficiency.

Since computing bilinear pairings imposes a greater computational cost on a protocol, to improve the efficiency, we do not utilize bilinear pairings in the signature scheme as a part of our protocol. However, to remove some security weaknesses which appeared in some previous protocols and to establish more session keys, we use the bilinear pairings just for deriving session keys. The proposed protocol has two phases: the setup phase and the key agreement phase which are described in detail in what follows:

5.1. The setup phase

The proposed protocol is a certificate based protocol in which each user possesses a public/private key-pair and a certificate which includes identity and public key of the user signed by a certificate authority (CA) as a trusted third party. Therefore, the CA determines the following cases with respect to the security parameter κ:

1. An additive group $\mathbb{G}$ of large prime order q, including points of an elliptic curve E over a finite field, (An element $Q \in \mathbb{G}$ has x-coordinate showed by k_Q, in this paper and y-coordinate, in Cartesian coordinates.),
2. the base point P of E which is a generator of $\mathbb{G}$, consequently,
3. the bilinear pairing $e(.,.)$ and
4. CA's private key (sk_{CA}) and the corresponding public key (pk_{CA}).

Next, the CA publishes the public parameters $\{E, \mathbb{G}, P, q, e(.,.), pk_{CA}\}$. Then, each user (e.g., Alice) chooses a random number $x_A \in_R \mathbb{Z}_q^*$ as a private key and computes $Y_A = x_A P$ as a public key. Finally, the user makes a request to the CA to allocate a certificate; after authenticating the user, the CA issues a certificate $\mathsf{Cert}(Y_A)$ and delivers it to the user.

5.2. The key agreement phase

In this phase, Alice and Bob aim to share several secret keys via an insecure network. Alice and Bob possess the triple $\{x_A,Y_A,\mathrm{Cert}(Y_A)\}$ and $\{x_B, Y_B, \mathrm{Cert}(Y_B)\}$, respectively. The details of the execution of the protocol between them shown in the Fig. 1 are as follows:

Step1. Alice → Bob: $\{T_{A1},T_{A2},S_A,\mathrm{Cert}(Y_A)\}$.
Alice chooses two random numbers $r_{A1}, r_{A2} \in_R \mathbb{Z}_q^*$ and computes $T_{A1} = r_{A1}P$ and $T_{A2} = r_{A2}P$. If $k_{A1} \cdot k_{A2} \neq 0 \bmod q$, she computes $S_A = (k_{A1} \cdot k_{A2})x_A - (r_{A1}k_{A1} + r_{A2}k_{A2}) \bmod q$, and otherwise, she chooses another random number and follows the protocol. Then, she sends $(T_{A1},T_{A2},S_A,\mathrm{Cert}(Y_A))$ to Bob.

Step2. Bob →Alice: $\{T_{B1},T_{B2},S_B,\mathrm{Cert}(Y_B)\}$.
Bob, also chooses two random numbers $r_{B1}, r_{B2} \in_R \mathbb{Z}_q^*$ and computes $T_{B1} = r_{B1}P$ and $T_{B2} = r_{B2}P$. If $k_{B1} \cdot k_{B2} \neq 0 \bmod q$, he computes $S_B = (k_{B1} \cdot k_{B2})x_B - (r_{B1}k_{B1} + r_{B2}k_{B2}) \bmod q$, and otherwise, he chooses another random number and follows the protocol. Then, he sends $(T_{B1},T_{B2},S_B,\mathrm{Cert}(Y_B))$ to Alice.

Step3. Upon receiving the message from Bob, Alice verifies the $\mathrm{Cert}(Y_B)$. If it failed, she terminates the protocol; otherwise, she verifies Bob's signature S_B by checking the following equation:

$$(k_{B1} \cdot k_{B2})Y_B \overset{?}{=} S_BP + (k_{B1}T_{B1} + k_{B2}T_{B2})$$

If the above verification failed, Alice terminates the execution. Otherwise, she computes the session keys as follows:

$$\begin{aligned}
K_1 &= e(T_{B1},T_{B1})^{r_{A1}r_{A1}} = e(P,P)^{r_{A1}r_{A1}r_{B1}r_{B1}}\\
K_2 &= e(T_{B1},T_{B2})^{r_{A1}r_{A1}} = e(P,P)^{r_{A1}r_{A1}r_{B1}r_{B2}}\\
K_3 &= e(T_{B2},T_{B2})^{r_{A1}r_{A1}} = e(P,P)^{r_{A1}r_{A1}r_{B2}r_{B2}}\\
K_4 &= e(T_{B1},T_{B1})^{r_{A1}r_{A2}} = e(P,P)^{r_{A1}r_{A2}r_{B1}r_{B1}}\\
K_5 &= e(T_{B1},T_{B2})^{r_{A1}r_{A2}} = e(P,P)^{r_{A1}r_{A2}r_{B1}r_{B2}}\\
K_6 &= e(T_{B2},T_{B2})^{r_{A1}r_{A2}} = e(P,P)^{r_{A1}r_{A2}r_{B2}r_{B2}}\\
K_7 &= e(T_{B1},T_{B1})^{r_{A2}r_{A2}} = e(P,P)^{r_{A2}r_{A2}r_{B1}r_{B1}}\\
K_8 &= e(T_{B1},T_{B2})^{r_{A2}r_{A2}} = e(P,P)^{r_{A2}r_{A2}r_{B1}r_{B2}}\\
K_9 &= e(T_{B2},T_{B2})^{r_{A2}r_{A2}} = e(P,P)^{r_{A2}r_{A2}r_{B2}r_{B2}}
\end{aligned}$$

Alice $\{x_A, Y_A, \mathrm{Cert}(Y_A)\}$	**Bob** $\{x_B, Y_B, \mathrm{Cert}(Y_B)\}$
Step1. Alice → Bob: $\{T_{A1}, T_{A2}, S_A, \mathrm{Cert}(Y_A)\}$	**Step2.** Bob → Alice: $\{T_{B1}, T_{B2}, S_B, \mathrm{Cert}(Y_B)\}$
$r_{A1}, r_{A2} \in_R \mathbb{Z}_q^*$ $T_{A1} = r_{A1}P$ $T_{A2} = r_{A2}P$ $S_A = (k_{A1} \cdot k_{A2})x_A - (r_{A1}k_{A1} + r_{A2}k_{A2}) \bmod q$	$r_{B1}, r_{B2} \in_R \mathbb{Z}_q^*$ $T_{B1} = r_{B1}P$ $T_{B2} = r_{B2}P$ $S_B = (k_{B1} \cdot k_{B2})x_B - (r_{B1}k_{B1} + r_{B2}k_{B2}) \bmod q$
Step3. Verification $(k_{B1} \cdot k_{B2})Y_B \overset{?}{=} S_BP + (k_{B1}T_{B1} + k_{B2}T_{B2})$	**Step4.** Verification $(k_{A1} \cdot k_{A2})Y_A \overset{?}{=} S_AP + (k_{A1}T_{A1} + k_{A2}T_{A2})$
Session Keys	**Session Keys**
$K_1 = e(T_{B1}, T_{B1})^{r_{A1}r_{A1}}$ $K_2 = e(T_{B1}, T_{B2})^{r_{A1}r_{A1}}$ $K_3 = e(T_{B2}, T_{B2})^{r_{A1}r_{A1}}$ $K_4 = e(T_{B1}, T_{B1})^{r_{A1}r_{A2}}$ $K_5 = e(T_{B1}, T_{B2})^{r_{A1}r_{A2}}$ $K_6 = e(T_{B2}, T_{B2})^{r_{A1}r_{A2}}$ $K_7 = e(T_{B1}, T_{B1})^{r_{A2}r_{A2}}$ $K_8 = e(T_{B1}, T_{B2})^{r_{A2}r_{A2}}$ $K_9 = e(T_{B2}, T_{B2})^{r_{A2}r_{A2}}$	$K_1 = e(T_{A1}, T_{A1})^{r_{B1}r_{B1}}$ $K_1 = e(T_{A1}, T_{A1})^{r_{B1}r_{B1}}$ $K_3 = e(T_{A1}, T_{A1})^{r_{B2}r_{B2}}$ $K_4 = e(T_{A1}, T_{A2})^{r_{B1}r_{B1}}$ $K_5 = e(T_{A1}, T_{A2})^{r_{B1}r_{B2}}$ $K_6 = e(T_{A1}, T_{A2})^{r_{B2}r_{B2}}$ $K_7 = e(T_{A2}, T_{A2})^{r_{B1}r_{B1}}$ $K_8 = e(T_{A2}, T_{A2})^{r_{B1}r_{B2}}$ $K_9 = e(T_{A2}, T_{A2})^{r_{B2}r_{B2}}$

Fig. 1. The proposed multiple-key agreement protocol.

Step4. Upon receiving the message from Alice, Bob verifies the $\mathrm{Cert}(Y_A)$. If it failed, he terminates the protocol, and otherwise, he verifies Alice's signature S_A by checking the following equation:

$$(k_{A1}\cdot k_{A2})Y_A \stackrel{?}{=} S_AP + (k_{A1}T_{A1} + k_{A2}T_{A2})$$

If the above verification failed, Bob terminates the execution. Otherwise, he computes the session keys as follows:

$$\begin{aligned}
K_1 &= e(T_{A1},T_{A1})^{r_{B1}r_{B1}} = e(P,P)^{r_{A1}r_{A1}r_{B1}r_{B1}}\\
K_2 &= e(T_{A1},T_{A1})^{r_{B1}r_{B2}} = e(P,P)^{r_{A1}r_{A1}r_{B1}r_{B2}}\\
K_3 &= e(T_{A1},T_{A1})^{r_{B2}r_{B2}} = e(P,P)^{r_{A1}r_{A1}r_{B2}r_{B2}}\\
K_4 &= e(T_{A1},T_{A2})^{r_{B1}r_{B1}} = e(P,P)^{r_{A1}r_{A2}r_{B1}r_{B1}}\\
K_5 &= e(T_{A1},T_{A2})^{r_{B1}r_{B2}} = e(P,P)^{r_{A1}r_{A2}r_{B1}r_{B2}}\\
K_6 &= e(T_{A1},T_{A2})^{r_{B2}r_{B2}} = e(P,P)^{r_{A1}r_{A2}r_{B1}r_{B2}}\\
K_7 &= e(T_{A2},T_{A2})^{r_{B1}r_{B1}} = e(P,P)^{r_{A2}r_{A2}r_{B1}r_{B1}}\\
K_8 &= e(T_{A2},T_{A2})^{r_{B1}r_{B2}} = e(P,P)^{r_{A2}r_{A2}r_{B1}r_{B2}}\\
K_9 &= e(T_{A2},T_{A2})^{r_{B2}r_{B2}} = e(P,P)^{r_{A2}r_{A2}r_{B2}r_{B2}}
\end{aligned}$$

6. Security analysis and simulation results of the proposed protocol

In this section, we discuss the security analysis and simulation results of the proposed protocol.

6.1. Theoretical analysis

In this subsection, we evaluate the security attributes of the proposed protocol.

- *Signature correctness:* the correctness of the signature $S_A = (k_{A1}\cdot k_{A2})x_A - (r_{A1}k_{A1} + r_{A2}k_{A2}) \bmod q$ is shown as follows:

$$\begin{aligned}
S_AP + (k_{A1}T_{A1} + k_{A2}T_{A2}) &= \{(k_{A1}\cdot k_{A2})x_A - (r_{A1}k_{A1} + r_{A2}k_{A2})\}P + (k_{A1}T_{A1} + k_{A2}T_{A2})\\
&= (k_{A1}\cdot k_{A2})x_AP - (r_{A1}k_{A1}P + r_{A2}k_{A2}P) + (k_{A1}T_{A1} + k_{A2}T_{A2})\\
&= (k_{A1}\cdot k_{A2})Y_A - (k_{A1}T_{A1} - k_{A2}T_{A2}) + (k_{A1}T_{A1} + k_{A2}T_{A2}) = (k_{A1}\cdot k_{A2})Y_A
\end{aligned}$$

By the same way, a proof can be indicated for S_B.

- *Security against forgery attack:* The weakness of Cheng and Ma's protocol was due to the presence of private keys x_A and x_B for computing the signatures S_{A1} and S_{B1}, simultaneously. That is,

$$S_{A1} = (r_{A1}k_{A1} + r_{A2}k_{A2})T_{A1} + x_AY_B = (r_{A1}k_{A1} + r_{A2}k_{A2})T_{A1} + \underline{x_Ax_B}P$$

and

$$S_{B1} = (r_{B1}k_{B1} + r_{B2}k_{B2})T_{B1} + x_BY_A = (r_{B1}k_{B1} + r_{B2}k_{B2})T_{B1} + \underline{x_Bx_A}P$$

Since this simultaneity is prevented in our protocol, applying the same attack is impossible. In addition, the signature scheme of our protocol is robust against other kinds of forgery attacks, because computing the signature $S_A = (k_{A1}\cdot k_{A2})x_A$ - $_A = (k_{A1}\cdot k_{A2})x_A - (r_{A1}k_{A1} + r_{A2}k_{A2}) \bmod q$ without knowing the private key x_A is not possible. Note that, the values of k_{A2} and k_{A2} must satisfy $k_{A1}\cdot k_{A2} \neq 0 \bmod q$; otherwise (i.e., $k_{A1}\cdot k_{A2} = 0 \bmod q$), the signature may be forged. Thus, this inequality is necessary for the security of the protocol.

- *Known-Key Security:* In this attack, an adversary who has some previous session keys is willing to compute the next session keys. In case in the proposed protocol, an adversary who knows the previous session keys, it does not give him any useful information to compute the next session keys, because short-term private keys r_{A1}, r_{A2} and r_{B1},r_{B2} are only used for computing the session keys and the keys are changed in each session. Thus, the proposed protocol is resilient to the known-key attack. Note that, computing $r_{A/B}$ from $T_{A/B} = r_{A/B}P$ is equal to solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) which, in turn, is intractable to solve.
- *Perfect Forward Secrecy:* This property implies that the revealed long-term private keys of both participants do not cause the previous session keys to be obtained by the adversary. In the proposed protocol, the adversary who knows x_A and x_B cannot compute the previous session keys because long-term private keys do not play any role for computing the session keys. In addition, the adversary can compute none of the short-term private keys r_{A1}, r_{A2}, r_{B1} and r_{B2} by knowing x_A, x_B, S_A and S_B, because it is equal to exhaustive search in $\mathbb{Z}_q^*$. Therefore, the proposed protocol satisfies the perfect forward secrecy property.

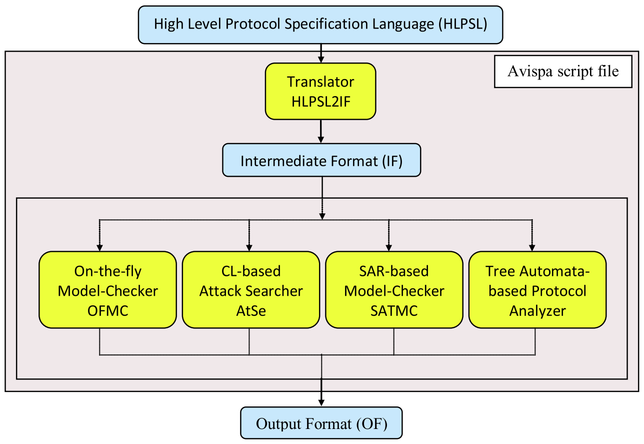
High Level Protocol Specification Language (HLPSL)
Avispa script file
Translator HLPSL2IF
Intermediate Format (IF)
On-the-fly Model-Checker OFMC
CL-based Attack Searcher AtSe
SAR-based Model-Checker SATMC
Tree Automata-based Protocol Analyzer
Output Format (OF)

Fig. 2. The architecture of the AVISPA tool [24].

- *Key-Compromise Impersonation:* This property expresses that, if the long-term private key of one entity (e.g., Alice) is disclosed, the adversary is unable to impersonate the other entity to the compromised entity (e.g., Bob to Alice). Let us suppose the adversary who knows x_A wants to impersonate Bob to Alice in our protocol. To impersonate Bob, the adversary has to generate a valid signature on arbitrary values ($T_{B1} = r_{B1}P$, $T_{B2} = r_{B2}P$) produced by himself as follows:

$$S_B = (k_{B1} \cdot k_{B2})x_B - (r_{B1}k_{B1} + r_{B2}k_{B2}) \bmod q$$

Without knowing x_B, the adversary cannot generate the above signature. Therefore, the proposed protocol is secure against the key-compromise impersonation attack.

- *Unknown key security:* This property implies that, the active adversary Eric should not enable to interfere in a key agreement protocol run such that Alice believes that Bob is her participant while Bob believes that he shared the session key with Eric. Let us suppose the adversary C intercepts (T_{A1},T_{A2},S_A) and wants to sign (T_{A1},T_{A2}) with his long-term private key x_C as follows:

$$S_C = (k_{A1} \cdot k_{A2})x_C - (r_{A1}k_{A1} + r_{A2}k_{A2}) \bmod q$$

In computing S_C, he has to know r_{A1} and r_{A2} while by the assumption of intractability of ECDLP, this is equal to solving the hard problem. Thus, the proposed protocol is not vulnerable to unknown key attack.

6.2. Simulation results

In the last decade, we have witnessed the development of a large number of new techniques for the formal analysis of security protocols. Until now, many (semi-)automated security protocol analysis tools have been proposed (e.g., [24,32,33]). One of the analysis tools that has the widest use in the cryptography is the AVISPA. AVISPA is a push-button tool for the automated validation of the Internet security-sensitive protocols and applications. It provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques.

The architecture of AVISPA is shown in Fig. 2. The first step in using the tool is to present the analyzed protocol in a special language called High Level Protocol Specification Language (HLPSL). The HLPSL presentation of the protocol is translated into the lower level language called Intermediate Format (IF). This translation is performed by the translator called HLPSL2IF. This step is totally transparent to the user. IF presentation of the protocol is used as an input to the four different back-ends: On-the-fly Model-Checker (OFMC), CL-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree-Automata-based Protocol Analyzer (TA4SP). These back-ends perform the analysis and output the results in precisely defined output format stating whether there are problems in the protocol or not.

In order to evaluate the security of the proposed protocol by the AVISPA tools, the protocol is first coded in HLPSL. The HLPSL code of the protocol is included in Appendix A. After execution of the code in AVISPA tool, both OFMC and CL-AtSe back-end outputs were generated. According to the summary results of the outputs shown in Fig. 3, the proposed protocol is safe and there are no major attacks to the proposed protocol.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS


PROTOCOL
  C:\SPAN\testsuite\results\protocol proposed.if

GOAL
  as_specified

BACKEND
  OFMC

COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.09s
  visitedNodes: 4 nodes
  depth: 2 plies
```

(i)

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  C:\SPAN\testsuite\results\protocol proposed.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 2 states
  Reachable  : 0 states
  Translation: 0.05 seconds
  Computation: 0.00 seconds
```

(ii)

Fig. 3. (i) Summary of OFMC back-end and (ii) summary of CL-AtSe back-end.

7. Computational analysis and comparisons

In the proposed protocol, each party chooses two random numbers to establish nine session keys. However, it is possible to extend the protocol for making more session keys with a generalized signature scheme and key derivation for n random numbers. The generalized signature for $U \in \{Alice, Bob\}$ and $T_{Ui} = r_{Ui}P$ is

$$s_U = \left(\prod_{i=1}^{n} k_{Ui}\right) x_U - \sum_{i=1}^{n} r_{Ui} k_{Ui} \pmod q$$

where $\prod_{i=1}^{n} k_{Ui} \neq 0 \pmod q$ and the session keys are computed as follows:

$$K = e(T_{Bi}, T_{Bj})^{r_{Ak} r_{Al}} \quad \text{for } i, j, k, l \in \{1, \ldots, n\}$$

To compute the number of the session keys, we use the combination $C(m,r)$, defined as follows:

$$C(m, r) = \frac{m!}{(m-r)!r!}$$

Therefore, we can count the number of different ways to take i and j from $\{1, \ldots, n\}$ with repetition by finding the number of non-negative integer solutions of the equation $i + j = n$, which is $C(\{2 + n - 1\}, 2)$. In the same way, we can also compute the number of different ways for taking k and l which is $C(\{2 + n - 1\}, 2)$. Hence, the number of the session keys is equal to

$$(C(\{2 + n - 1\}, 2))^2 = \left(\frac{(n+1)!}{(n-1)!2!}\right)^2 = \left(\frac{(n+1)n}{2}\right)^2.$$

For example, if $n = 2$ the number of the session keys is

$$(C(\{2 + 2 - 1\}, 2))^2 = \left(\frac{(2+1)!}{(2-1)!2!}\right)^2 = \left(\frac{(3)2}{2}\right)^2 = 9$$

and for $n = 3$ it is

$$(C(\{2 + 3 - 1\}, 2))^2 = \left(\frac{(3+1)!}{(3-1)!2!}\right)^2 = \left(\frac{(4)3}{2}\right)^2 = 36.$$

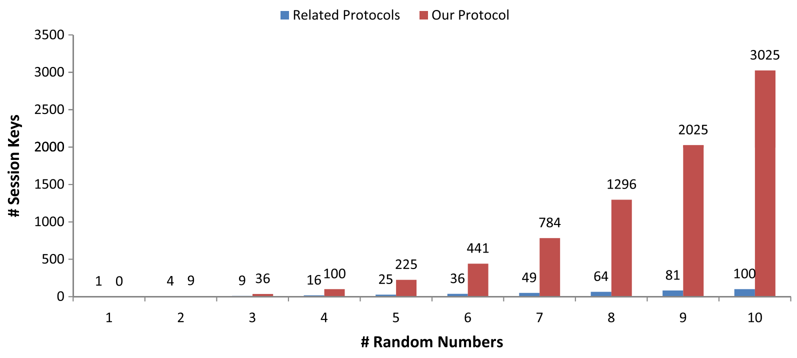

Fig. 4. Comparison of our protocol with three other protocols in the number of session keys.

Table 3
Comparison of the proposed protocol and the related previous protocols.

Protocol	LWW [20]	VLYK [21]	CM [23]	Our protocol
Computations of $T_{(A/B)1,2}$	$2S$	$2S$	$2S$	$2S$
Signing $S_{A \text{ or } B}$	$2M + 1A + 2S$	$2M + 1A + 2S$	$3M + 1A + 3S$	$4M$
Verification	$1A + 2S + 3P$	$1A + 2S + 3P$	$1A + 3S + 5P$	$1M + 2A + 4S$
Computations for each key	$1A + 1S + 1P$	$1A + 3S + 1P$	$1A + 3S + 1P$	$1M + 1S + 1P$
Total computations	$2M + 3A + 7S + 4P$	$2M + 3A + 9S + 4P$	$3M + 3A + 11S + 6P$	$6M + 2A + 7S + 1P$
Running time (ms)	130.23	143.05	186.84	68.9
# Session keys for 2 Rands	4	4	4	9

Fig. 4 illustrates a comparison between the number of session keys of our protocol and three previous protocols [20,21,23]. Generally, those previous protocols make n^2 session keys with n random numbers, while our protocol establishes $(C(\{2 + n - 1\}, 2))^2$ session keys. As can be clearly seen, this number for our protocol exponentially rises, while it is increased slowly for the other protocols. The difference between the two diagrams quickly grows, such that with 10 random numbers, the number of session keys in our protocol, is 30 times more than that of the others.

Other computational fields can also be compared between our protocol and three previous protocols [20,21,23]. For this comparison, we consider basic computation atoms: modular multiplication (M), point addition (A) and scalar multiplication (S) on an elliptic curve, and bilinear pairings (P) for computing the short-term public keys $T_{(A/B)\ 1,2}$, the signing, the verification and the session key steps. As can be seen in Table 3, the maximum difference between the complexity of our protocol and that of the other protocols is in the signature scheme. We can also obtain the running time of the protocols using the results of Table 3. Accordingly, the running time of our protocol, 68.9 ms, is the minimum compared to that of the others.

8. Conclusions

In this paper, we analyzed the security of Chang and Ma's multiple-key agreement protocol. We showed that Chang and Ma's protocol is vulnerable to forgery attack. The computations needed for an adversary to apply this forgery attack are as little as the computations needed for a legal participant in a session of the Chang and Ma's protocol. Therefore, the complexity of the forgery attack is equal to the computational cost of the protocol. To evaluate the feasibility of the proposed attack, we calculated the computational complexity of the attack using MIRACL library. The time complexity of the forgery attack is 27.24 ms which is easily achievable.

Generally, the weakness of Chang and Ma's protocol arises from the signature scheme that it uses. As a result, designing a signature scheme without one-way hash functions which is not only secured against well-known attacks, but also provides the security properties of the multiple-key agreement protocols, is a fundamental principle. Based on the principle, we proposed a novel protocol based on bilinear pairings. The main contribution of the proposed protocol is the use of bilinear pairings in the computation of the session keys and not in the digital signature scheme. As a result, the computational cost of the proposed protocol is less than that of the other related protocols. Another novelty of the proposed protocol is the number of the session keys which is $(n + 1)^2/4$ times more than that of the similar previous protocols. Finally, the simulation results from AVISPA tools confirmed the security analysis of the protocol.

In the proposed protocol, each user possesses a certificate that binds the identity of the user with his/her public-key. In some situations, using the certificates has undeniable restrictions and an identity-based solution is recommended. Therefore, our future work shall be aimed at designing a secure identity-based multiple-key agreement protocol.

Appendix A. HLPSL code of the proposed protocol

```
role alice(
A,B:agent,
SND,RCV : channel(dy),
Minus,Union,Pred,Succ,Expr,Ebilinear:function,
Ka1,Ka2,Kb1,Kb2,P:  symmetric_key)
played_by A
def=
local State :nat,
Ya,Yb: public_key,
Xa,K1,K2,K3,K4,K5,K6,K7,K8,K9: symmetric_key,
Ra1,Ra2,Rb1,Rb2 :text,
CertA,CertB,Ta1,Ta2,Tb1,Tb2,Sa,Sb :message
const key_id1:protocol_id
init State:=0
transition
1.State=0/\RCV(start)=|>State':=2
 /\Ra1':=new()/\Ra2':=new()/\Ta1':=Pred(Ra1,P)/\Ta2':=Pred(Ra2,P)
/\Sa':=Minus(Pred(Pred(Ka1,Ka2),Xa),Union(Pred(Ra1,Ka1),Pred(Ra2,Ka2)))
/\SND(Ta1.Ta2.Sa.CertA)
2.State=2/\RCV(Tb1'.Tb2'.Sb'.CertB')
/\Sb'=minus((Pred(Pred(Kb1,Kb2),Yb)),Union(Pred(Rb1,Kb1),Pred(Rb2,Kb2)))=|>State':=4
/\K1':=Expr(Ebilinear(Tb1,Tb1),Pred(Ra1,Ra1))
/\K2':=Expr(Ebilinear(Tb1,Tb2),Pred(Ra1,Ra1))
/\K3':=Expr(Ebilinear(Tb2,Tb2),Pred(Ra1,Ra1))
/\K4':=Expr(Ebilinear(Tb1,Tb1),Pred(Ra1,Ra2))
/\K5':=Expr(Ebilinear(Tb1,Tb2),Pred(Ra1,Ra2))
/\K6':=Expr(Ebilinear(Tb2,Tb2),Pred(Ra1,Ra2))
/\K7':=Expr(Ebilinear(Tb1,Tb1),Pred(Ra2,Ra2))
/\K8':=Expr(Ebilinear(Tb1,Tb2),Pred(Ra2,Ra2))
/\K9':=Expr(Ebilinear(Tb2,Tb2),Pred(Ra2,Ra2))
/\secret(Xa,key_id1,{a,b})
/\request(A,B,pxa,Xa)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role bob(
B,A:agent,
SND,RCV : channel(dy),
Minus,Union,Pred,Succ,Expr,Ebilinear:function,
Kb1,Kb2,Ka1,Ka2,P:  symmetric_key)
played_by B
def=
local State :nat,
Ya,Yb: public_key,
Xb,K1,K2,K3,K4,K5,K6,K7,K8,K9: symmetric_key,
Ra1,Ra2,Rb1,Rb2:  text,
CertA,CertB,Tb1,Tb2,Ta1,Ta2,Sb,Sa:message
const key_id2:protocol_id
init State:=1
transition
1.State=1/\RCV(Ta1'.Ta2'.Sa'.CertA')
/\Sa'=Minus(Pred(Pred(Ka1,Ka2),Ya),Union(Pred(Ra1,Ka1),Pred(Ra2,Ka2)))=|>State':=3
/\Rb1':=new()/\Rb2':=new()/\Tb1':=Pred(Rb1,P)/\Ta2':=Pred(Rb2,P)
/\Sb':=minus((Pred(Pred(Kb1,Kb2),Xb)),Union(Pred(Rb1,Kb1),Pred(Rb2,Kb2)))
/\K1':=Expr(Ebilinear(Ta1,Ta1),Pred(Rb1,Rb1))
/\K2':=Expr(Ebilinear(Ta1,Ta1),Pred(Rb1,Rb2))
/\K3':=Expr(Ebilinear(Ta1,Ta1),Pred(Rb2,Rb2))
/\K4':=Expr(Ebilinear(Ta1,Ta2),Pred(Rb1,Rb1))
/\K5':=Expr(Ebilinear(Ta1,Ta2),Pred(Rb1,Rb2))
/\K6':=Expr(Ebilinear(Ta1,Ta2),Pred(Rb2,Rb2))
/\K7':=Expr(Ebilinear(Ta2,Ta2),Pred(Rb1,Rb1))
/\K8':=Expr(Ebilinear(Ta2,Ta2),Pred(Rb1,Rb2))
/\K9':=Expr(Ebilinear(Ta2,Ta2),Pred(Rb2,Rb2))
/\SND(Tb1.Tb2.Sb.CertB)
/\secret(Xb,key_id2,{b,a})
/\request(B,A,pxb,Xb)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(
A,B:agent,
Minus,Union,Pred,Succ,Expr,Ebilinear:function,
Ka1,Ka2,Kb1,Kb2,P:  symmetric_key)
def=
local
SendA,ReceiveA:channel(dy),
SendB,ReceiveB:channel(dy)
composition
alice(A,B,SendA,ReceiveA,Minus,Union,Pred,Succ,Expr,Ebilinear,Ka1,Ka2,Kb1,Kb2,P)
/\bob(B,A,SendB,ReceiveB,Minus,Union,Pred,Succ,Expr,Ebilinear,Ka1,Ka2,Kb1,Kb2,P)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
const
pxa:protocol_id,
pxb:protocol_id,
kb1,kb2,ka1,ka2,ki1,ki2,p : symmetric_key,
a,b,i :  agent,
minus,union,pred,succ,expr,ebilinear:function
intruder_knowledge={a,b,ki1,ki2,minus,union,pred,succ,expr,ebilinear,p}
composition
session(a,b,minus,union,pred,succ,expr,ebilinear,ka1,ka2,kb1,kb2,p)
/\session(a,i,minus,union,pred,succ,expr,ebilinear,ka1,ka2,ki1,ki2,p)
/\session(i,b,minus,union,pred,succ,expr,ebilinear,ki1,ki2,kb1,kb2,p)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
secrecy_of  key_id1,key_id2
authentication_on pxa
authentication_on pxb
end goal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
environment()
```

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compeleceng.2012.09.004.

References

[1] Diffie W, Hellman ME. New directions in cryptography. IEEE Trans Inform Theory 1976;22(6):644–54.
[2] Harn L, Lin HY. An authenticated key agreement protocol without using one-way function. In: Proceedings of eighth information security conference. Taiwan, May 1998; p. 155–60.
[4] Yen SM, Joye M. Improved authenticated multiple-key agreement protocol. Electron Lett 1998;34(18):1738–9.
[5] Wu TS, He WH, Hsu CL. Security of authenticated multiple-key agreement protocol. Electron Lett 1999;35(5):391–2.
[6] Harn L, Lin HY. Authenticated key agreement without using one-way hash function. Electron Lett 2001;37(10):629–30.
[7] Zhou HS, Fan L, Li JH. Remarks on unknown key-share attack on authenticated multiple-key agreement protocol. Electron Lett 2003;39(17):1248–9.
[8] Farash MS, Attari MA, Bayat M. Security of multiple key agreement protocols and propose an enhanced protocol. Cryptology ePrint archive, report 2011/634. http://eprint.iacr.org/2011/634.
[9] Tan Z. Efficient identity-based authenticated multiple key exchange protocol. Comput Electr Eng 2011;37(2):191–8.
[10] Wenmin L, Qiaoyan W, Qi S, Hua Z, Zhengping J. Password-authenticated multiple key exchange protocol for mobile applications. China Commun 2012;9(1):64–72.
[11] Menezes A, Vanstone S, Okamoto T. Reducing elliptic curve logarithms to logarithms in a finite field. In: STOC '91: proceedings of the twenty-third annual ACM symposium on theory of computing. New York, NY, USA: ACM Press; 1991. p. 80–9.
[12] Joux A. A one round protocol for tripartite Diffie–Hellman. In: 4th International symposium on algorithmic number theory 2000. LNCS, vol. 1838. New York: Springer; 2000. p. 385–94.
[13] Smart NP. An identity based authenticated key agreement protocol based on the Weil pairing. Electron Lett 2002;38(13):630–2.
[14] Chen L, Kudla C. Identity based authenticated key agreement protocols from pairings. In: Proceedings of the 16th IEEE computer security foundations workshop. IEEE Computer Society Press; 2003. p. 219–33.
[15] McCullagh N. Barreto PSLM. A new two-party identity-based authenticated key agreement. In: Proceedings of the CT-RSA 2005. LNCS, vol. 3376. Springer-Verlag; 2005. p. 262–74.
[16] Chen L, Cheng Z, Smart NP. Identity-based key agreement protocols from pairings. Int J Info Secur 2007;6(4):213–41.
[17] Chow SSM, Choo KKR. Strongly-secure identity-based key agreement and anonymous extension. In: Proceedings of the ISC 2007. LNCS, vol. 4779. Springer-Verlag; 2007. p. 203–20.
[18] Huang H, Cao Z. An ID-based authenticated key exchange protocol based on bilinear Diffie–Hellman problem. In: Proceedings of the ACM ASIACCS 2009. ACM; 2009. p. 333–42.
[19] Ni L, Chen G, Li J, Hao Y. Strongly secure identity-based authenticated key agreement protocols. Comput Electr Eng 2011;37(2):205–17.
[20] Lee NY, Wu CN, Wang CC. Authenticated multiple key exchange protocols based on elliptic curves and bilinear pairings. Comput Electr Eng 2008;34(1):12–20.
[21] Vo DL, Lee H, Yeun CY, Kim K. Enhancements of authenticated multiple key exchange protocol based on bilinear pairings. Comput Electr Eng 2009;36(1):155–9.
[22] Farash MS, Bayat M, Attari MA. Vulnerability of two multiple-key agreement protocols. Comput Electr Eng 2011;37(2):199–204.
[23] Cheng Q, Ma C. Analysis and improvement of an authenticated multiple key exchange protocol. Comput Electr Eng 2011;37(2):187–90.
[24] The AVISPA project <http://www.avispa-project.org>.
[25] Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Proceedings of the CRTPTO 2001. LNCS, vol. 2139. Springer-Verlag; 2001. p. 213–29.
[26] Leng X. Smart card applications and security. Information security technical report; 2009:1–10.
[27] Mangard S, Oswald E, Popp T. Power analysis attacks revealing the secrets of smart cards. LLC, New York: Springer Science+Business Media; 2007.
[28] Matthews A, Consultant S, Software NG. Side-channel attacks on smartcards. Network Sec 2006:18–20.
[29] Boswell T. Smart card security evaluation : Community solutions to intractable problems. Information security technical report; 2009;1–13.
[30] Majzoobi M, Koushanfar F, Potkonjak M. Testing techniques for hardware security. In: Proceedings of the international test conference; 2008. p. 1–10.
[31] Shamus Software Ltd., Miracl library <http://www.shamus.ie/index.php?page=home>.
[32] Blanchet B. An efficient cryptographic protocol verifier based on Prolog rules. In: Proceedings 14th IEEE computer security foundations workshop (CSFW); 2001. p. 82–96.
[33] Cremers C. The Scyther tool: verification, falsification, and analysis of security protocols. In: Proceedings of CAV, LNCS, vol. 5123; 2008. p. 414–8.

Mohammad Sabzinejad Farash received the B.Sc. degree in Electronic Engineering from Shahid Chamran College of Kerman in 2006, and the M.Sc. degree in Communication Engineering from I.Hussein University in 2009. Currently, He is a Ph.D. candidate in Applied Mathematics at the Department of Mathematics and Computer Sciences of Tarbiat Moallem University in Iran. His research interests are Security Protocols and Provable Security Models.

Mahmoud Ahmadian Attari received the combined B.Sc. and M.Sc. degree in Electrical Engineering from University of Tehran, Iran in 1977. He also received the Ph.D. degree in Digital Communication Systems from University of Manchester in 1997. He is currently an Associate Professor at Electrical and Computer Engineering Faculty of K.N. Toosi University of Technology. His research interests are Coding Theory and Cryptography.

Reza Ebrahimi Atani is working as an assistant professor in the Computer Engineering Department of the University of Guilan in Rasht, Iran, from April 2010. He obtained a PhD from Iran University of Science and Technology in 2010, while working as a researcher at the Electronic Research Center of IUST. His main research interests are in symmetric cryptography and cryptographic hardware and embedded systems.

Mohamadreza Jami is currently working as a lecturer in the Faculty of Computer Science, Department of Information Technology (IT), Payame Noor University in Tehran, Iran. He obtained a M.Sc from Maleke Ashtar University of Technology in Tehran, Iran. His main research interests are in network security, cryptographic protocols and formal methods in security analysis of the protocols.