Multimed Tools Appl (2016) 75:17215–17245
DOI 10.1007/s11042-015-2988-4

Single round-trip SIP authentication scheme with provable security for Voice over Internet Protocol using smart card

Saru Kumari[1] · **Fan Wu**[2] · **Xiong Li**[3,4] · **Mohammad Sabzinejad Farash**[5] · **Qi Jiang**[6] · **Muhammad Khurram Khan**[7] · **Ashok Kumar Das**[8]

Received: 19 April 2015 / Revised: 30 August 2015 / Accepted: 5 October 2015 /
Published online: 4 November 2015

Abstract In recent years, Voice over Internet Protocol (VoIP) has gained more and more popularity as an application of the Internet technology. For various IP applications including VoIP, the topic of Session Initiation Protocol (SIP) has attracted major concern from researchers. SIP is an advanced signaling protocol operating on Internet Telephony. SIP uses digest authentication protocols such as Simple Mail Transport Protocol (SMTP) and Hyper

✉ Saru Kumari
saryusiirohi@gmail.com

Fan Wu
conjurer1981@gmail.com

Xiong Li
lixiong84@gmail.com

Mohammad Sabzinejad Farash
sabzinejad@khu.ac.ir

Qi Jiang
jiangqixdu@gmail.com

Muhammad Khurram Khan
mkhurram@ksu.edu.sa

Ashok Kumar Das
iitkgp.akdas@gmail.com

[1] Department of Mathematics, Ch. Charan Singh University, Meerut, Uttar Pradesh, India

[2] Department of Computer Science and Engineering, Xiamen Institute of Technology, Huaqiao University, Xiamen 361021, China

[3] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

[4] Nanjing University of Information Science and Technology, Nanjing 210044, China

[5] Department of Mathematics and Computer Sciences, Kharazmi University, Tehran, Iran

[6] School of Computer Science and Technology, Xidian University, Xi'an 710071 Shaanxi, People's Republic of China

Text Transport Protocol (HTTP). When a user seeks SIP services, authentication plays an important role in providing secure access to the server only to the authorized access seekers. Being an insecure-channel-based protocol, a SIP authentication protocol is susceptible to adversarial threats. Therefore, security is a big concern in SIP authentication mechanisms. This paper reveals the security vulnerabilities of two recently proposed SIP authentication schemes for VoIP, Irshad et al.'s scheme [*Multimed. Tools. Appl.* doi:10.1007/s11042-013-1807-z] and Arshad and Nikooghadam's scheme [*Multimed. Tools. Appl.* DOI 10.1007/s11042-014-2282-x], the later scheme is based on the former scheme. Irshad et al.'s scheme suffers from password guessing, user impersonation and server spoofing attacks. Arshad and Nikooghadam's scheme can be threatened with server spoofing and stolen verifier attack. None of these two schemes achieve mutual authentication. It also fails to follow the single round-trip authentication design of Irshad et al.'s scheme. To overcome these weaknesses, we propose a provable secure single round-trip SIP authentication scheme for VoIP using smart card. We formally prove the security of the scheme in random oracle and demonstrate through discussion its resistance to various attacks. The comparative analysis shows that the proposed SIP authentication scheme offers superior performance with a little extra computational cost.

Keywords Session initiation protocol · Voice over internet protocol · Authentication · Smart card · Provable security

1 Introduction

Session Initiation Protocol (SIP) is a text-based signaling protocol which regulates communications over the Internet [6]. SIP is an ideal mechanism to establish, maintain and terminate the multimedia-media sessions carried over the Internet. SIP finds application in controlling multimedia communication sessions such as video calls, conferencing, voice calls, instantaneous messaging via Voice over Internet protocols (VoIP) like Simple Mail Transport Protocol (SMTP) and Hyper Text Transport Protocol (HTTP) [27]. VoIP has revolutionized the concept of conventional telephony of circuit-switch by introducing the notion of internet telephony and SIP is capable to deal with all type of signaling necessities of VoIP. Generally, SIP uses the HTTP-digest-authentication protocol mentioned in RFC2617 [11] to achieve identity authentication. As compared to other signaling protocols such as H.323, SIP is more lightweight and agile [6]. Like SMTP and HTTP, SIP is a request-response procedure carried over insecure network, making request of a server and then awaiting a response. For instance, when a client P (caller) wishes to establish a SIP voice call to the server Q (callee), P should be able to verify that he/she is connected to SIP user agent of Q and not to an adversary. Thus, mutual authentication between the requester and the responder at the opposite ends on the telephone line is very crucial in SIP. However, Internet is subject to various security threats owing to its open nature. Therefore, designing a secure user authentication scheme for SIP-based services is a challenge.

[7] Center of Excellence in Information Assurance, King Saud University, Riyadh, Kingdom of Saudi Arabia

[8] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology (IIIT), Hyderabad 500032, India

1.1 Related work

In 2002, Rosenberg et al. [26], a member in the Internet Engineering Task Force (IETF) of Multi-Party Multimedia Session Control (MMUSIC) Working Group proposed SIP. The Third-Generation Partnership Project (3GPP) has chosen SIP [26] as the protocol for multimedia applications in 3G mobile networks [12, 17]. Steep increase in multimedia services based on Voice over Internet Protocol (VoIP) has made SIP authentication schemes a preferred field of research.

So far, many SIP authentication schemes have been proposed in literature. Many studies such as [13, 27, 28] revealed the applicability of the server spoofing and the off-line password guessing attacks on the original HTTP-digest-authentication [11]. In 2005, Yang et al. designed a SIP authentication scheme [33] using the Diffie–Hellman key exchange algorithm [8]. Although, the security of their scheme relies on the difficulty of Discrete Logarithm Problem (DLP), it is inapt for devices with low computational power due to the high computational cost. But, according to Tang and Liu [29] and Yoon et al. [37], stolen verifier attack is applicable on Yang et al.'s scheme. Besides, Tang et al. [29] pointed out Denning-Sacco attack onYang et al.'s scheme.

Inspired with the design of Yang et al.'s scheme, in the same year Durlanik and Sogukpinar [9] designed a SIP authentication scheme using elliptic curve cryptosystem (ECC) [4, 14, 18, 23, 24]. The security of their scheme relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC offers a security-level comparable to that of the classical cryptosystems which use quite larger key sizes. Thus, Durlanik and Sogukpinar's scheme has considerably low computational cost and occupies less memory space as compared to Yang et al.'s scheme. In 2009, Wu et al. [31] designed an ECC-based SIP authentication scheme to overcome the security problems of various SIP authentication schemes. They claimed that their scheme provides mutual authentication, confidentiality of post authentication messages and perfect forward secrecy simultaneously. They also proved their scheme to be secure in the Canetti–Krawczyk (CK) security model [5]. In contrast to the earlier schemes [9, 33], Wu et al.'s scheme is suitable for low powered devices with limited memory space. Nonetheless, according to Yoon et al. [37], SIP authentication schemes proposed by Durlanik and Sogukpinar and Wu et al., both suffer from Denning-Sacco attack [7], off-line password guessing and stolen-verifier attacks. Moreover, Yoon et al. designed another improved scheme for SIP authentication [37]. However, according to Liu and Koenig [21], Yoon et al.'s scheme is prey to off-line password guessing attack. In 2009, Tsai [30] proposed a lightweight SIP authentication scheme based on nonce. Subsequently, Lee [20] identified password guessing and insider attacks on Tsai's scheme. In 2013, Arshad et al. [2] demonstrated that Tsai's scheme [30] does not resist stolen verifier, off-line password guessing, and Denning-Sacco attacks and fails to provide perfect forward secrecy, key agreement, and known-key secrecy. Further, they presented their own SIP authentication and key agreement scheme [2] based on ECC. In 2012, He et al. [15], and later, in 2013, Tang et al. [29] and Pu et al. [25], independently showed that Arshad et al.'s scheme [2] is vulnerable to off-line password guessing attack. Additionally, they modified the Arshad et al.'s scheme in terms of improved proposals of SIP authentication schemes. In 2010, Yoo et al. [36] proposed an ECC-based SIP authentication scheme to overcome the weaknesses of Tsai et al.'s scheme. In 2012, Xie [32] identified off-line password guessing and stolen-verifier attacks on Yoo et al.'s scheme. In 2013, Farash and Attari [10] exhibited impersonation and off-line password guessing attacks on Xie's scheme [32].

In 2013, Yeh et al. proposed an ECC-based SIP authentication scheme [34] to overcome faults in the contemporary SIP schemes. They claimed their scheme to be secure for applications demanding high security. However, we observe user impersonation attack and insecure password update in the scheme. Besides, their scheme requires one and half round-trip to

accomplish the authentication process. In the same year, Zhang et al. [38] pondered that most of the proposed SIP authentication schemes require the server to store user's verifier(s) in the database which makes these schemes prone to server spoofing, password guessing and stolen verifier attacks. Therefore, they proposed a new password-based SIP authenticated scheme [38] without the need of maintaining a verification table. They claimed their scheme to be free from many known attacks. In the same year, Irshad et al. [16] pointed out that in Zhang et al.'s scheme, the messages transmitted between the user and the server are not fresh and hence can be replayed. Irshad et al. also visualized that one and half round of the Zhang et al.'s scheme could be cut short to a single round. Thus, Irshad et al. proposed a SIP authentication scheme with single round-trip. Recently, Arshad and Nikooghadam [3] showed that in Irshad et al.'s scheme an insidious user of the system can impersonate some other user. Then, they proposed a SIP authentication and key agreement scheme based on Irshad et al.'s scheme. They claimed that their scheme not only provides improved security, but is also quite efficient as compared to the previous SIP authentication schemes.

1.2 Our contributions

This research exposes the weaknesses of Irshad et al.'s [16] and Arshad and Nikooghadam's [3] SIP authentication schemes for VoIP. We show that in addition to the attack mounted by Arshad and Nikooghadam, Irshad et al.'s scheme also suffers from other security pitfalls. We explain that in Irshad et al.'s scheme, the login-authentication phase is erroneous and a secret key of the server can be recovered by any legal user of the system. Further, a sensitive value generated from server's second secret key and which is common for all registered users is not secure in the scheme. These weaknesses pave the way to other vulnerabilities like disclosure of user's random key, password guessing, user impersonation and server spoofing attacks. Besides, we exhibit that in Arshad and Nikooghadam's scheme, an adversary can spoof the legal server to cheat the user without requiring the user's password or the server's secret key. Further, the storage of users' verifiers in server's database makes the scheme open to the stolen verifier attack. In fact, both the schemes lack the attribute of mutual authentication. We observe that the scheme proposed by Arshad and Nikooghadam requires one and half round-trip to complete the authentication process which is contrary to the single round-trip in the original scheme by Irshad et al. To lift the security and to remove the design flaws of these schemes, we then propose a single round-trip SIP authentication scheme for VoIP using smart card, keeping the design originality of Irshad et al.'s scheme intact. We not only discuss conventionally the resistance of the proposed scheme to various attacks but also give formal security proof in random oracle. Further, we compare our scheme with some existent SIP authentication schemes to assess its efficiency. The overall analysis proves the utility of the scheme for SIP-services.

1.3 Arrangement of the paper

Here follows the layout of the remaining paper. Section 2 and 3 give review and cryptanalysis of Irshad et al.'s SIP authentication scheme respectively. Section 4 and 5 give review and cryptanalysis of Arshad and Nikooghadam's SIP authentication scheme respectively. Our proposed SIP authentication is presented in Section 6. Section 7 and 8 respectively focus on the formal and conventional security analysis of the proposed scheme. Section 9 shows the comparisons of the proposed scheme with some related SIP authentication schemes. Finally, the paper is ended with conclusion in Section 10.

1.4 Useful preliminaries

1.4.1 Elliptic Curve Cryptography (ECC)

Here, we give a brief background of an elliptic curve and its computational problems [4, 14, 18, 23, 24]. In elliptic curve cryptography (ECC), the elliptic curve equation is given by $E_p(f, g) : y^2 = x^3 + ax + b \pmod p$ over a finite field F_p of prime order $p>3$, where, $f, g \in F_p$ and $4a^3 + 27b^2 \neq 0 \pmod p$. Given an integer $r \in F_p^*$ and a point $P \in E_p(f, g)$, the elliptic curve point multiplication $r \cdot P$ over $E_p(f, g)$ is defined as $r \cdot P = P + P + \ldots + P$ (r times). Normally, the following two intractable problems form the basis of the security of ECC:

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**: For two given points P and Q over $E_p(f, g)$, ECDLP says to find out an integer $r \in F_p^*$ when $Q = r \cdot P$.
- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**: For three given points P, $r \cdot P$, and $s \cdot P$ over $E_p(f, g)$ for $r, s \in F_p^*$, ECDHP says to find out the point $r \cdot s \cdot P$ over $E_p(f, g)$.

1.4.2 Notations with their description

The notations and their description used throughout the paper are listed in Table 1.

Table 1 The notations with description useful throughout the paper

Notations	Description
S	Server
U	User
A_d	Adversary
$username$	Username of U
Pw	Password of U
e	Randomly generated key of U
SC	Smart card of U
T_l, T_p	Current timestamps at the U side
T_A	Current timestamps at the A_d side
p, n	Large prime numbers
F_p	A finite field of prime order p
$E_p(f, g)$	An elliptic curve with order n defined over F_p
P	A point of $E_p(f, g)$ called the base point, P is of order n
d, d_1, d_2	Secret keys maintained by S
P_{ub}	Public key of the S
G	Additive cyclic group generated by P over $E_p(f, g)$
Z_p	Ring of integers modulo p
Z_p^*	Multiplicative group of Z_p
$E_k(.)/D_k(.)$	En(de)cryption functions with k as key
$h(.)$, $h_1(.)$, $h_2(.)$	Cryptographic one-way hash functions
$\|\|$	Concatenation operator

2 Review of Irshad et al.'s scheme

The details of four phases, system setup phase, registration phase, login-authentication phase and password update phase of Irshad et al.'s scheme are reviewed as follows:

2.1 System setup phase

This phase is about defining the different parameters meant for public use or user's interface with the system. The server S does the following preparations to setup the system. S chooses an elliptic curve equation $E_p(f, g)$ of order n and P as a base point, from security view point n and p are chosen to be two large prime numbers with high entropy. S randomly picks $d_1, d_2 \in Z_p^*$ as two secret keys and computes the public key $P_{ub} = d_2P$. S chooses three one-way hash functions $h(.) : \{0,1\}^* \rightarrow \{0,1\}^k$, $h_1(.) : G \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^k$, $h_2(.) : G \times G \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^k$. Finally, S publishes the information $\{E_p(f, g), P, P_{ub}, h(.), h_1(.), h_2(.)\}$ in a public directory.

2.2 Registration phase

In this phase, S validates U over a secure channel. The steps undergone by U and S for the purpose of registration are as listed below:

1) On affirmative verification at S, U randomly generates a key e, chooses a random integer $i_r \in Z_p^*$ and her password Pw. Next, she computes $h(Pw||i_r)$ and submits $\{(h(Pw||i_r)||username), e\}$ to S.
2) In response to the received information $\{(h(Pw||i_r)||username), e\}$ from U, S computes $R = h(h(Pw||i_r)||username)\, d_2^{-1}P$ and $I = (e||username)d_1^{-1}$. S stores R in a smart card SC and securely provides $SC = \{R\}$ & I to U.
3) U stores i_r in SC and at the closing of the registration phase U owns $SC = \{R, i_r\}$ & I.

2.3 Login-authentication phase

When U wishes to log into S, she inserts her SC into a smart card reader and inputs her *username* and Pw. Subsequently, the procedure that takes place between U and S for the purpose of mutual authentication and session key agreement is listed below as a series of steps:

1) U selects a random integer $a \in Z_p^*$ and acquires the current timestamp T_l. Then U computes $x = aR$, $X = ah(h(Pw||i_r)||username)P_{ub}$ and $m_u = MAC_e(T_l)$. Next, it sends $REQUEST = \{realm, username, x, X, I, T_l, h(m_u)\}$ to S over a public channel. Here MAC_e is message authentication code with e as key, also known as keyed hash function since it takes as input a secret key and an a message of arbitrary length.
2) On receiving the $REQUEST$, S obtains $e' = d_1I$, computes $m_u' = MAC_{e'}(T_l)$ and checks if $h(m_u) \stackrel{?}{=} h(m_u')$. The success of this verification validates the freshness of T_l and the $REQUEST$. After that, S computes $X' = d_2^2(x)$ and checks if $X \stackrel{?}{=} X'$. On the success of this verification, it randomly selects two integers r, $b \in Z_p^*$ and computes the following values: $y = bP$, $K = bd_2x$, $m_s = MAC_{e'}(T_l + 1)$, $sk =$

$h_1(K||r||m_s||username)$ and $Auth_s = h_2(K||X'||r||sk)$. Finally, S answers back with $RESPONSE = \{realm, r, y, Auth_s\}$ to U.

3) On receiving the $RESPONSE$, U computes $K' = ah(h(Pw||i_r)||username)y$, $m_s' = MAC_e(T_l + 1)$, $sk = h_1(K'||r||m_s'||username)$ and checks if $Auth_s \_? h_2(K'||h(h(Pw||i_r)||username)aP_{ub}||r||sk)$. The success of this verification validates S and U can rely on $sk = h_1(K'||r||m_s'||username)$ as the session key to establish confidential communication with S till next login.

2.4 Password update phase

This phase is initiated using a recent session key sk. The details of this phase are as given in the following steps:

1) U chooses a new random integer $i_{rnew} \in Z_p^*$ and a new password Pw_{new}. Next, she computes $h(Pw_{new}||i_{rnew})$ and acquires the current timestamp T_p. Then, U performs the encryption $V = E_{sk}(username||T_p||h(Pw_{new}||i_{rnew})||h(username||T_p||h(Pw_{new}||i_{rnew})))$ using sk and submits $PU\text{-}REQUEST = \{V, T_p\}$ as a password update request to S.
2) On obtaining $PU\text{-}REQUEST$ from U, S decrypts V. Next, S checks the validity of the message by computing the fresh value $h(username||T_p||h(Pw_{new}||i_{rnew}))$ and comparing it with the received value recovered from V. For success of this verification, S computes the new parameter $R_{new} = h(h(Pw_{new}||i_{rnew})||username)d_2^{-1}P$ and performs the encryption $W = E_{sk}(R_{new}||h(username||\ T_p + 1||\ R_{new}))$. S answers back with $PU\text{-}RESPONSE = \{W\}$ to U.
3) On obtaining $PU\text{-}RESPONSE$ from S, U decrypts W and confirms its validity by computing the fresh value $h(username||\ T_p + 1||\ R_{new})$ and comparing it with the received value recovered from W. For success of this verification, U replaces R and i_r in her SC with R_{new} and i_{rnew} respectively.

3 Cryptanalysis of Irshad et al.'s scheme

3.1 Error in login-authentication phase

During the registration phase, S provides $SC = \{R\}$ & $I = (e||username)d_1^{-1}$ to U. Then, U stores the random integer i_r in SC owns $SC = \{R, i_r\}$ & I. At the time of login, U computes $m_u = MAC_e(T_l)$ using her random key e and sends $REQUEST = \{realm, username, x, X, I, T_l, h(m_u)\}$ containing I to S. Our concern is that the values $\{e, I\}$ are neither stored in the memory of SC nor these are like password which can be easily memorized by the user. Besides, if $SC = \{R, i_r, I\}$, U cannot obtain e out of I as it requires the knowledge of S's secret key d_1. Consequently, U cannot initiate the login unless both e and I are stored in SC.

3.2 Server's first secret is at risk

During the registration phase, U submits her registration request containing a randomly generated key e to S. Then S computes $I = (e||username)d_1^{-1}$ and provides it to U. The value I contains U's key e protected with S's secret key d_1. Since U knows her key e and

username, she can obtain d_1^{-1} from I as $d_1^{-1} = (e||username)^{-1}I$ and further obtains the secret key d_1 of S by computing $(d_1^{-1})^{-1}$. In this way, any legal user can get hold of the secret key d_1 of S and can use it to impersonate the other users as will be explained in Subsections 3.6.

3.3 Secret value generated from server's second secret is at risk

It is observable that the value $d_2^{-1}P$ containing S's secret key d_2 is used to compute the parameter $R = h(h(Pw||i_r)||username)d_2^{-1}P$ for U and similar parameters for each of the registered user. Here, we show that any legal user can gain this sensitive value by undergoing password update phase. U chooses a new random integer $i_{rnew} \in Z_p^*$, a new password Pw_{new} and submits her password update request to S. In response, U receives the new parameter $R_{new} = h(h(Pw_{new}||i_{rnew})||username)d_2^{-1}P$ from S. Then she can obtain $d_2^{-1}P$ from R_{new} as $(h(h(Pw_{new}||i_{rnew})||username))^{-1} R_{new}$. In this manner, any legal user can get hold of $d_2^{-1}P$, which is common for all users, without deploying any expensive method. Having $d_2^{-1}P$ in hand, one can mount password guessing and user impersonation threats on Irshad et al.'s scheme as the discussion follows in Subsections 3.5 and 3.6 respectively.

3.4 Attack on user's random key

We have shown in Subsection 3.2 that an adversary A_d, who is a malicious legal user, can obtain the value d_1 which is a secret key of S. Then he can obtain the value $I = (e||username)d_1^{-1}$ from an intercepted $REQUEST = \{realm, username, x, X, I, T_l, h(m_u)\}$ of U. Using d_1, she can gain the random key e of U as $(e||username) = d_1 I$. Thus, A_d can own the random secret key of any user of the system.

3.5 Password guessing attack

Suppose that the smart card of U is lost or stolen and an adversary A_d, who is actually a malicious legal user, gains the information $\{R, i_r,$ etc.$\}$ from its memory using techniques such as power consumption or side channel attacks [19, 22, 35]. Then A_d can obtain the secret $d_2^{-1}P$ using his own smart card (as discussed in subsection 3.3) to guess the password of U. A_d guesses Pw^* as a possible password of U to compute $R^* = h(h(Pw^*||i_r)||username)\, d_2^{-1}P$, where *username* of U is on easy access from some previously intercepted U's *REQUEST*. A_d verifies if $R \_? R^*$, the success of this verification implies the correctness of the guessed value Pw^*, else, A_d repeats the computation and verification process with some another guess. Thus, password guessing is possible in Irshad et al.'s scheme.

3.6 User impersonation attack

In order to impersonate U, a malicious legal user A_d first of all obtains the secrets d_1^{-1} and $d_2^{-1}P$ using her own smart card (as discussed in subsections 3.2 and 3.3) and then continues further in the following manner:

1) A_d selects a random key e^A to compute $I^A = (e^A||username_u)d_1^{-1}$. Then she selects a random integer $a^A \in Z_p^*$ and computes $x^A = a^A d_2^{-1}P$, $X^A = a^A P_{ub}$ and

$m_A = MAC_{eA}(T_A)$ using the current value of the timestamp T_A. Next, it sends $REQUEST = \{realm, username_u, x^A, X^A, I^A, T_A, h(m_A)\}$ to S over a public channel.

2) On receiving the $REQUEST$, S obtains $(e^A)' = d_1 I$, computes $m_A' = MAC_{(eA)'}(T_A)$ and checks if $h(m_A) \_? h(m_A')$. Clearly, this verification will hold and the timestamp T_A will pass the freshness test. Afterwards, S computes $(X^A)' = d_2{}^2(x^A)$ and checks if $X^A \_? (X^A)'$. Clearly, this verification, will hold since $(X^A)' = d_2{}^2(x^A) = d_2{}^2(a^A d_2{}^{-1} P) = a^A d_2 P = a^A P_{ub} = X^A$. Consequently, S randomly selects two integers r, $b \in Z_p{}^*$ and computes the following values: $y = bP$, $K^A = bd_2 x^A$, $m_s = MAC_{(eA)'}(T_A + 1)$, $sk = h_1(K||r||m_s||username_u)$ and $Auth_s = h_2(K||(X^A)'||r||sk)$. Finally, S answers back with $RESPONSE_A = (realm, r, y, Auth_s)$ to U.

3) On receiving the $RESPONSE_A$, A_d computes $K^A = a^A y$, $m_s' = MAC_{eA}(T_A + 1)$, $sk = h_1(K^A||r||m_s'||username_u)$ and checks if $Auth_s \_? h_2(K^A||X^A||r||sk)$. The success of this verification validates S and ensures the correctness of the computed session key sk. Now, A_d can communicate with S using sk.

Thus, A_d impersonates U to cheat S whereas S believes that he is communicating with the registered user U. It is noticeable that A_d can apply a forgery attack by using an arbitrary username $username_A$ instead of $username_u$ in the above process. We can see that the forgery process goes on smoothly if we replace $username_u$ with $username_A$ in the whole process and S has no way to differentiate a registered user from a non- registered user.

3.7 Server spoofing attack

Here, we show the vulnerability of Irshad et al.'s scheme to the server spoofing attack. We show how a malicious legal user A_d, after obtaining the secret d_1 of S using her own smart card (as discussed in subsections 3.2), can prove itself the legal server to the user. For this, A_d performs the following steps:

1) A_d intercepts a current login request $REQUEST = \{realm, username, x, X, I, T_l, h(m_u)\}$ sent by U to S.
2) A_d obtains $(e^A)' = d_1 I = e$ and randomly selects two integers r^A, $b^A \in Z_p{}^*$. Then, she computes $y^A = b^A P_{ub}$, $K^A = b^A X$, $m_A = MAC_{(eA)'}(T_l + 1)$, $sk_A = h_1(K^A||r^A||m_A||username)$ and $Auth_A = h_2(K^A||X||r^A||sk_A)$. A_d answers back with $RESPONSE_A = (realm, r^A, y^A, Auth_A)$ to U.
3) On receiving the $RESPONSE_A$, U computes $K = ah(h(Pw||i_r)||username)y^A$, $m_A' = MAC_{eA}(T_l + 1)$, $sk = h_1(K||r^A||m_A'||username)$ and checks if $Auth_A \_? h_2(K||X||r^A||sk_A)$. Clearly, this verification will hold since $K = ah(h(Pw||i_r)||username)y^A = ah(h(Pw||i_r)||username)b^A P_{ub} = b^A ah(h(Pw||i_r)||username)P_{ub} == b^A X = K^A$. This verification not only ensures the freshness of the extension $T_l + 1$ of the timestamp T_l but also makes U to believe that sk computed by her is equal to the session key sk_A computed at the server side.

In this way, U is contented that the received $RESPONSE_A$ is fresh (free from any replay) and originated from the legal server. Moreover, A_d shares a confidential communication channel with the user by means of the established session key.

3.8 Mutual authentication attack

As discussed in Sections 3.6 and 3.7, an adversary A_d who is a malicious legal user can impersonate the user and can spoof the server, therefore mutual authentication is not achieved in the Irshad et al.'s scheme.

4 Review of Arshad et al.'s scheme

The detail of four phases, system setup phase, registration phase, login-authentication phase and password update phase of Arshad and Nikooghadam's scheme is reviewed as follows:

4.1 System setup phase

This phase is exactly same as in Irshad et al.'s scheme except that S maintains only one secret key $d \in Z_p^*$ with $P_{ub} = dP$ as the corresponding public key. S chooses a one-way hash functions $h(.) : \{0,1\}^* \rightarrow \{0,1\}^k$. Finally, S publishes the information $\{E_p(f, g), n, P, P_{ub}, h(.)\}$ in a public directory. We avoid mentioning the rest of the similar details to avoid the redundancy of the text.

4.2 Registration phase

In this phase, S validates U over a secure channel. The steps undergone by U and S for the purpose of registration are as listed below:

1) U chooses a random integer $i_r \in Z_p^*$ and her password Pw. She computes $v_i = h(ID||Pw||i_r)$ and stores i_r in her memory device (portable HDDs, USB stick, etc.), where ID denotes the identity of U. Next, she submits $\{ID, v_i\}$ to S.
2) After receiving the information $\{ID, v_i\}$ from U, S checks if ID already exists in its database or not. If ID is absent in the database then S computes $R = h(ID||d) \oplus v_i$ and stores $\{ID, R\}$ in its database.

4.3 Login-authentication phase

When U wishes to log into S, the following process takes place between U and S for the purpose of mutual authentication and session key agreement:

1) U chooses a random integer $a \in Z_p^*$ and computes $x = aP_{ub} = adP$. Then, it sends $REQUEST = \{ID, x\}$ to S over a public channel.
2) On receiving $REQUEST$, S checks its database for the existence of ID. S terminates the session in case ID is absent in database. Otherwise, S randomly selects an integer $b \in Z_p^*$ and computes $y = bP$, $K = bd^{-1}x = baP$ and $m_s = h(ID||y||K)$. Then, S answers back with $RESPONSE_S = \{realm, y, m_s\}$ to U.
3) On receiving $RESPONSE_S$, U computes $K' = ay = abP$ and checks if $m_s' \_? h(ID||y||K')$. The success of this verification authenticates S. Thereby, U retrieves i_r from her

memory device to compute $v_i = h(ID||Pw||i_r)$. Next, she computes $m_u = h(ID||y||realm||K'||v_i)$, the session key $sk = h(ID||y||K'||realm)$ and replies with $RESPONSE_U = \{ID, realm, m_u\}$ to S.

4) On receiving $RESPONSE_U$, S computes $v_i = R \oplus h(ID||d)$ and checks if $m_u' \_?$ $h(ID||y||realm||K||v_i)$. The success of this verification authenticates U and thereby S computes the session key $sk = h(ID||y||K||realm)$.

4.4 Password update phase

This phase is initiated using a recent session key sk. The details of this phase are as given in the following steps:

1) U chooses a new random integer $i_{rnew} \in Z_p^*$ and a new password Pw_{new}. She retrieves i_r from her memory device, to computes $v_{inew} = h(ID||Pw_{new}||i_{rnew})$. Next, it computes $z = h(ID||Pw||i_r) \oplus v_{inew} = v_i \oplus v_{inew}$, $Z = z \oplus h(ID||sk) = v_i \oplus v_{inew} \oplus h(ID||sk)$, $V = h(ID||v_{inew}||sk||v_i)$ and submits $PU\text{-}REQUEST = \{ID, Z, V\}$ as a password update request to S.
2) On obtaining $PU\text{-}REQUEST$ from U, S computes $v_i = R \oplus h(ID||d)$, $v_{inew} = Z \oplus v_i \oplus h(ID||sk)$. Next, S checks the validity of the message by computing the fresh value $h(ID||v_{inew}||sk||v_i)$ and comparing it with the received value V. For success of this verification, S computes the new parameter $R_{new} = R \oplus z = h(ID||d) \oplus h(ID||Pw||i_r) \oplus h(ID||Pw||i_r) \oplus h(ID||Pw_{new}||i_{rnew}) = h(ID||d) \oplus h(ID||Pw_{new}||i_{rnew})$ and replaces v_i with v_{inew} in its database. S answers back with $PU\text{-}RESPONSE = \{h(ID||v_i||accept||v_{inew}||sk)\}$ to U.
3) On obtaining $PU\text{-}RESPONSE$ from S, U checks the validity of the message by computing the fresh value $h(ID||v_i||accept||v_{inew}||sk)$ and comparing it with the received value. For success of this verification, U replaces i_r with i_{rnew} in her memory device.

5 Cryptanalysis of Arshad et al.'s scheme

5.1 Server spoofing attack

Now, we show that Arshad et al.'s scheme is susceptible to an impersonation attack as an adversary A_d can spoof the legal server to cheat any registered user of the system. Here follows the description of the attack:

1) When U wishes to log into S, she sends $REQUEST = \{ID, x\}$ to S, where $x = aP_{ub} = adP$ with a as a random integer belonging to Z_p^*.
2) A_d intercepts and blocks the $REQUEST$, and chooses a random integer $b_A \in Z_p^*$. Computes $y_A = b_A P_{ub} = b_A dP$, $K_A = b_A x = b_A aP_{ub} = b_A adP$ and $m_A = h(ID||y_A||K_A)$. Sends $RESPONSE_A = \{realm, y_A, m_A\}$ to U.
3) On receiving $RESPONSE_A$, U computes $K_A' = ay_A = ab_A P_{ub} = ab_A dP$ and checks if $m_A' \_?$ $h(ID||y_A||K_A')$. Clearly, this verification will hold by virtue of the equality of K_A' and K_A. Thus, U believes that the received $RESPONSE_A$ originated from the legal S and proceeds further. Retrieves i_r from her memory device to compute $v_i = h(ID||Pw||i_r)$. Next, she

computes $m_{uA} = h(ID||y_A||realm||K_A'||v_i)$, the session key $sk_A = h(ID||y_A||K_A'||realm)$ and replies with $RESPONSE_{UA} = \{ID, realm, m_A\}$ to S.

4) On receiving $RESPONSE_{UA}$, A_d computes the session key $sk = h(ID||y_A||K_A||realm)$ to communicate with U.

Consequently, U believes that she is connected with the legal server whereas it is the adversary A_d at the opposite end who is not only connected with U but has also established a secure communication channel with U. The noticeable thing about this attack is that for this the adversary A_d neither requires U's smart card or password nor needs S's secret key d.

5.2 Stolen verifier attack

S stores $\{ID, R\}$ in its database, where $R = h(ID||d) \oplus v_i = h(ID||d) \oplus h(ID||Pw||i_r)$ is used by S during the authentication to verify U. Assume the leakage of S's secret key d, in such situation, an adversary A_d can obtain U's verifier $h(ID||Pw||i_r)$ from R as $h(ID||Pw||i_r) = R \oplus h(ID||d)$. Further, if A_d happens to access the memory device of U then he can extract the random integer i_r from it to guess U's password. He guess Pw_g as U's possible password and computes $h(ID||Pw_g||i_r)$. Compares the two values $h(ID||Pw||i_r)$ and $h(ID||Pw_g||i_r)$ of the verifier, the equality yields the correct password otherwise he tries with another guess. He can keep guessing continue until success is achieved.

5.3 Lacks mutual authentication

As described in section 5.1, an adversary A_d can cheat a legal user by spoofing as the legal server. Therefore, mutual authentication is shattered in Arshad and Nikooghadam's scheme.

6 The proposed scheme

Here, we propose our improved scheme to conquer the weaknesses of Irshad et al.'s scheme. Similar to Irshad et al.'s scheme, our scheme is also divided into four phases as listed and detailed below along with figure:

6.1 System setup phase

This phase is exactly same as that of Irshad et al.'s scheme except that S maintains only one secret key $d \in Z_p^*$ with $P_{ub} = dP$ as the corresponding public key. We avoid mentioning the rest of the similar details to avoid the redundancy.

6.2 Registration phase

In this phase, S validates U over a secure channel. The steps undergone by U and S for the purpose of registration are as listed below:

1) On affirmative verification at S, U randomly generates a key e, chooses a random integer $i_r \in Z_p^*$ and her password Pw. Next, she computes $h(Pw) + i_r$ and submits $\{username, h(Pw) + i_r, e\}$ to S.

2) In response to the received information $\{username, h(Pw) + i_r, e\}$ from U, S computes $R' = (h(Pw) + i_r) + h(username||d)$ and $I = (e + h(d||username))$. S stores I in a smart card SC and securely provides $SC = \{I\}$ & R' to U.
3) U computes $R = R' - i_r = h(Pw) + h(username||d)$, $Z = (e + h(username||Pw))$ and stores them in her SC. At the closing of the registration phase, U owns $SC = \{I, R, Z\}$.

6.3 Login-authentication phase

When U wishes to log into S, she inserts her SC into a smart card reader and inputs her *username* and *Pw*. Subsequently, the procedure that takes place between U and S for the purpose of mutual authentication and session key agreement is listed below in a series of steps:

1) U selects a random integer $a \in Z_p^*$ and acquires the current timestamp T_l. Then U computes $x = aP$, $X = a(R - h(Pw))P_{ub} = ah(username||d)dP$, $e = Z - h(username||Pw)$ and $m_u = h(username||X||e||T_l)$. Next, it sends $REQUEST = \{realm, username, x, I, T_l, m_u\}$ to S over a public channel.
2) On receiving the $REQUEST$, S obtains $e' = I - h(d||username)$, computes $X' = xh(username||d)d$, $m_u' = h(username||X'||e'||T_l)$ and checks if $m_u \_? m_u'$. The success of this verification validates the freshness of T_l, freshness of $REQUEST$ and hence authenticates U. S randomly selects an integer $b \in Z_p^*$ and computes the following values: $y = bP$, $K = bx = abP$, $sk = h_1(K||X'||e'||username)$ and $Auth_s = h_2(K||sk||T_l + 1)$. Finally, S answers back with $RESPONSE = (realm, y, Auth_s)$ to U.
3) On receiving the $RESPONSE$, U computes $K' = ay = abP$, $sk = h_1(K'||X||e||username)$ and checks if $Auth_s \_? h_2(K'||sk||T_l + 1)$. The success of this verification validates S and U can rely on $sk = h_1(K'||X||e||username)$ as the session key to establish confidential communication with S till next login.

6.4 Password update phase

This phase is initiated using a recently agreed session key sk immediately after the session key is establsihed. The details of this phase are as follows:

1) U chooses a new random integer $i_{rnew} \in Z_p^*$ and a new password Pw_{new}. Next, it computes $h(Pw_{new}) + i_{rnew}$ and acquires the current timestamp T_p. Then, U performs the encryption $V = E_{sk}(username||T_p||h(Pw_{new}) + i_{rnew} \; ||h(username||T_p||h(Pw_{new}) + i_{rnew}))$ using sk and submits $PU\text{-}REQUEST = \{V, T_p\}$ as a password update request to S.
2) On obtaining $PU\text{-}REQUEST$ from U, S decrypts V. Next, S checks the validity of the message by computing the fresh value $h(username||T_p||h(Pw_{new}) + i_{rnew})$ and comparing it with the received value recovered from V. For success of this verification, S computes the new parameter $(R_{new})' = (h(Pw_{new}) + i_{rnew}) + h(username||d)$ and performs the encryption $W = E_{sk}((R_{new})'||h(username||\ T_p + 1||(R_{new})'))$. S answers back with $PU\text{-}RESPONSE = \{W\}$ to U.
3) On obtaining $PU\text{-}RESPONSE$ from S, U decrypts W and confirms its validity by computing the fresh value $h(username||\ T_p + 1||(R_{new})')$ and comparing it with the received value recovered from W. For success of this verification, U computes

$R_{new} = (R_{new})' - i_{rnew} = h(Pw_{new}) + h(username||d)$, $Z_{new} = Z - h(username||Pw) + h(username||Pw_{new})$. Then U replaces R and Z in her SC with R_{new} and Z_{new} respectively.

7 Analysis of security properties of the proposed scheme

7.1 Secret key of the server is safe

During the registration phase, U submits her registration request containing a randomly generated key e to S. Then S computes $R' = (h(Pw) + i_r) + h(username||d)$, $I = (e + h(d||username))$ and provides $SC = \{I\}$ & R' to U. Although a malicious user can obtain $h(username||d)$ from R' as $h(username||d) = R' - (h(Pw) + i_r)$, he cannot obtain S's secret key d from $h(username||d)$ due to the one-way property of the hash function. Similar hindrance is posed by the value I for recovering d from it. Moreover, she needs to extract [19, 22, 35] I from her SC which is a costly method. Even though, an adversary A_d can take I by intercepting a login request $REQUEST = \{realm, username, x, I, T_l, m_u\}$ of U from the network, he cannot gain d from it without knowing U's random secret key e and due to the one-way property of the hash function. In this way, neither a legal user nor an adversary can get hold of the secret key d of S.

7.2 Password guessing attack

Suppose that the smart card of U is lost or stolen and an adversary A_d gains the information $\{I, R, Z\}$ from its memory using techniques such as power consumption or side channel attacks [19, 22, 35]. Then A_d has two choices, R & Z, for guessing U's password. In order to guess user's password from Z, A_d requires the knowledge of U's $username$ and random secret key e. However, $username$ can be taken from an intercepted login request; e is not obtainable by any means. On the other hand, guessing of Pw from R is not feasible as it additionally requires the knowledge of S's secret key d. A_d can take $username$ from an intercepted login request and can guesses Pw^*. Next, he computes $e^* \leftarrow Z - h(username||Pw^*)$, $(h(d||username))^* = I + e^*$. But, the value $(h(d||username))^*$ is of no help to verify the guessed Pw^* from $R = h(Pw) + h(username||d)$ as it contains $h(username||d)$ which is different from $h(d||username)$. Thus, password guessing is not possible in the proposed scheme.

7.3 User impersonation attack

In order to impersonate as U, A_d should be able to compute a workable login request involving U's specific secrets. A_d cannot compute the correct value of X as he does not know S's secret key d. A_d cannot obtain e from I available in login request of U because it involves the secret key d of S. Further, he cannot obtain U's random secret key e from her lost or stolen SC as it requires U's password Pw. Thus, it is not possible to impersonate a legal user.

7.4 Server spoofing attack

To spoof the server, A_d should be able to compute a valid $RESPONSE$ in reply to a current $REQUEST$ sent by U. For this purpose, A_d requires S's secret key d to compute $X' = xh(username||d)d$, needs U's random secret key e to compute $sk = h_1(K||X'||e'||username)$

embedded in the authenticating value $Auth_s = h_2(K||sk||T_l + 1)$. In the absence of d & e, A_d cannot spoof the server.

7.5 Replay attack

The login request $REQUEST = \{realm, username, x, I, T_l, m_u\}$ sent by U to S contains the fresh timestamp T_l. The parameter $m_u = h(username||X||e||T_l)$ responsible for the verification of the $REQUEST$ contains the timestamp T_l. On obtaining the $REQUEST$, S computes $e' = I - h(d||username)$, $X' = xh(username||d)d$, $m_u' = h(username||X'||e'||T_l)$ and checks if $m_u \_? m_u'$. The success of this verification validates the freshness of T_l and ensures that the $REQUEST$ has not been replayed. The response message $RESPONSE = (realm, y, Auth_s)$ sent by S to U contains the extension $T_l + 1$ of T_l. The session key $sk = h_1(K||X'||e'||username)$ is embedded in $Auth_s = h_2(K||sk||T_l + 1)$ which is responsible for the verification of the $RESPONSE = (realm, y, Auth_s)$. On obtaining the $RESPONSE$, U computes $K' = ay = abP$, $sk = h_1(K'||X||e||username)$ and checks if $Auth_s \_? h_2(K'||sk||T_l + 1)$. The success of this verification validates the freshness of $T_l + 1$ and ensures that the $RESPONSE$ has not been replayed. Thus, the contribution of fresh timestamp and its extension in $REQUEST$ and $RESPONSE$ respectively, protects the scheme against replay attack.

7.6 Forward secrecy

Forward secrecy ensures the security of already established session keys even if long term secrets (user's password or server's secret key) of any of the communicating participants are compromised. If U's password Pw or S's secret key d or both are compromised, an adversary A_d has to face the Elliptic Curve Discrete Logarithm Problem ($ECDLP$) to gain a or b from $x = aP$ or $y = bP$ respectively, otherwise, computation of $K = bx = ay = abP$ is not feasible. Since the value of $K = abP$ involved in the session key $sk = h_1(K||X'||e'||username) = h_1(K'||X||e||username)$ is independent of Pw and d, therefore, the proposed scheme provides strong forward secrecy.

7.7 Mutual authentication

The server S authenticates U by checking the freshness of T_l and hence the validity of the received $REQUEST = \{realm, username, x, I, T_l, m_u\}$ through the equivalence $m_u \_? m_u'$ after computing $e' = I - h(d||username)$, $X' = xh(username||d)d$ and $m_u' = h(username||X'||e'||T_l)$. On the other hand, U authenticates S by checking the freshness of the extension $T_l + 1$ of T_l and hence the validity of received $RESPONSE = (realm, y, Auth_s)$ through the equivalence $Auth_s \_? h_2(K'||sk||T_l + 1)$ after computing $K' = ay = abP$, and $sk = h_1(K'||X||e||username)$. Thus, both user and the server authenticate each other in the proposed scheme.

7.8 Insider attack

During registration phase, U submits $h(Pw) + i_r$, where i_r is a random integer chosen from Z_p^*. Thus, the insider of the system at the server has no knowledge of users' passwords. Moreover, owing to the random choice of i_r and the one-way property of hash function, he cannot reveal Pw from $h(Pw) + i_r$. Therefore, insider attack is not applicable on the proposed scheme.

7.9 Stolen verifier attack

The server does not store any information of its registered users and completes the authentication process using the information possessed by it and received from the user. Therefore, an adversary A_d has no way of stealing and misusing the information stored in some database at the server. Thus, the scheme is free from the stolen verifier attack.

7.10 Secure password update facility

To update her password, U inserts his smart card in a card reader to connect with the server. U uses the session key to communicate his encrypted password update request to S. In reply, S encrypts the parameters required to update the password with the same session key and transmits to U. An adversary A_d cannot update the password of U using the lost/stolen smart card of U unless he knows the session key. Thus, the scheme provides a secure password update phase.

7.11 Denning-Sacco attack

In Denning-Sacco attack [7] an adversary A_d attempts to guess the secret keys (such as server's secret key or user's password or a current session key) of any of the communicating participants from some previously disclosed session key. The proposed scheme utilizes the one-way hash function and ECC to compute the session key $sk = h_1(K||X||e||username) = h_1(abP||ah(username||d)dP||e||username)$. Since sk is independent of the user's password and server's secret key is protected under the one-way of hash function, an attacker A_d cannot obtain any secret key used in the protocol using a compromised previous session key. If A_d tries to guess the value d from sk, he requires to possesses the random secret key e of the user along with the random values a and b. However, to reveal a and b from $x = aP$ and $y = bP$ respectively, A_d has to face the intractable ECDLP. Therefore, the proposed scheme resists the Denning-Sacco attacks.

8 Formal security analysis of the proposed scheme

In this section, we show that our protocol is secure in the random oracle model. We start with the formal security model and the algorithm assumption that will be used in our proof.

8.1 Security model

In order to make our scheme resist the known attacks to the authentication protocols, we use the method of provable security. The security proof is based on the model proposed by Abdalla and Pointcheval [1]. The model that we use is as follows:

8.1.1 Participants

An authentication protocol Π runs in a network of a number of interconnected participants where each participant is either a client $U \in \mathcal{U}$ or a trusted server $S \mathcal{T}$. The set $\mathcal{T}$ is assumed to involve only a single server for simplicity. Each of the participants may have several instances

called oracles involved in distinct executions of the protocol Π. We refer to i-th instance of U (respectively S) in a session as Π_U^i(resp. Π_S^j). Every instance Π_U^i(resp. Π_S^j) has a partner ID pid_U^i(resp. pid_S^j), a session ID sid_U^i(resp. sid_S^j), a session key sk_U^i(resp. sk_S^j). pid_U^i(resp. pid_S^j) denotes the set of identities that are involved in this instance. sid_U^i(resp. sid_S^j) denotes the flows that are sent and received by the instance Π_U^i(resp. Π_S^j). An instance Π_U^i(resp. Π_S^j) is said to be accepted if it holds a session key sk_U^i(resp. sk_S^j), a session identifier sid_U^i(resp. sid_S^j), and a partner identifier pid_U^i(resp. pid_S^j). Two instances Π_U^i and Π_S^j are considered *partnered* if and only if (i) both of them have accepted, (ii) $pid_U^i = pid_S^j$, (iii) $sid_U^i = sid_S^j$, (ii) $sk_U^i = sk_S^j$.

8.1.2 Adversary model

The communication network is assumed to be fully controlled by an adversary $\mathcal{A}$, which schedules and mediates the sessions among all the parties. The adversary $\mathcal{A}$ is allowed to issue the following queries in any order:

Execute(Π_U^i, Π_S^j): This query models passive attacks in which the attacker eavesdrops on honest executions among the client instance Π_U^i and trusted server instance Π_S^j. The output of this query consists of messages that were exchanged during the honest execution of the protocol Π.
SendClient(Π_U^i, m): The adversary makes this query to intercept a message and then modify it, create a new one, or simply forward it to the client instance Π_U^i. The output of this query is the message that the client instance Π_U^i would generate upon receipt of message m . Additionally, the adversary is allowed to initiate the protocol by invoking *SendClient*(Π_U^i, *start*).
SendServer(Π_U^i, m): This query models an active attack against a server. The adversary makes this query to obtain the message that the server instance Π_S^j would generate on receipt of the message m.
Reveal(Π_U^i): This query models the known session key attack. The adversary makes this query to obtain the session key of the instance Π_U^i.
Corrupt(U): This query returns to the adversary the long-lived key Pw_U for participant U.
Test(Π_U^i): Only one query of this form is allowed to be made by the adversary to a fresh oracle. To respond to this query, a random bit $b \in \{0,1\}$ is selected. If $b=1$, then the session key held by Π_U^i is returned. Otherwise, a uniformly chosen random value is returned.

8.1.3 Fresh oracle

An oracle Π_U^i is called fresh if and only if the following conditions hold: (i) Π_U^i has accepted, and (ii) Π_U^i or its partner (if exists) has not been asked a *Reveal* query after their acceptance.

8.1.4 Protocol security

The security of an authentication protocol Π is modeled by the game (Π, $\mathcal{A}$). When playing this game, the adversary $\mathcal{A}$ can make many queries mentioned earlier to Π_U^i and Π_S^j. If $\mathcal{A}$ asks a single query, *Test*(Π_U^i), where Π_U^i has *accepted* and is *fresh*, then $\mathcal{A}$ outputs a single bit b'.

The aim of $\mathcal{A}$ is correctly guessing the bit b in the test session. More precisely, we define the advantage of $\mathcal{A}$ as follows:

$$Adv_{\Pi,D}(\mathcal{A}) = |2Pr[b' = b]-1|$$

The protocol Π is said to be secure if $Adv_{\Pi,D}(\mathcal{A})$ is negligible.

8.2 Computational assumption

We define the Decisional Diffie-Hellman (DDH) assumption which we use in the security proof of our scheme.

Definition 1 The DDH assumption can be precisely defined by two experiments, $Exp_{P,p}^{ddh-real}(W)$ and $Exp_{P,p}^{ddh-rand}(W)$. An adversary W is provided with uP, vP and uvP in the experiment $Exp_{P,p}^{ddh-real}(W)$, and uP, vP and wP in the experiment $Exp_{P,p}^{ddh-rand}(W)$, where u, v and w are drawn at random from Z_p^*. Define the advantage of W in violating the DDH assumption, $Adv_{P,p}^{ddh}(W)$, as follows:

$$Adv_{P,p}^{ddh}(W) = \max\left\{\left|Pr\left[Exp_{P,p}^{ddh-real}(W)=1\right]-Pr\left[Exp_{P,p}^{ddh-rand}(W)=1\right]\right|\right\}$$

8.3 Security proof

Theorem 1 Let D be a uniformly distributed dictionary of passible passwords with size |D|. Let Π describes the improved authentication protocol defined in Fig. 1. Suppose that DDH assumption holds, Then,

$$Adv_{\Pi,D}(\mathcal{A}) \le \frac{q_h^2 + q_{h1}^2 + q_{h2}^2}{2^k} + \frac{(q_s+q_e)^2}{p^2} + 2q_e \cdot Adv_{P,p}^{ddh}(W) + 2\max\left\{\frac{q_{h1}}{2^k}, \frac{q_s}{|D|}\right\},$$

where q_s denotes the number of Send queries; q_e denotes the number of Execute queries; q_h, q_{h1} and q_{h2} denote the number of hash queries to h, h_1 and h_2, respectively.

Proof This proof consists of a sequence of hybrid games, starting at the real attack G_0 and ending up at game G_4 where the adversary has no advantage. For each game $G_i(0 \le i \le 4)$, we define $Succ_i$ as the event that $\mathcal{A}$ correctly guesses the bit b in the test session.

Game G_0: This game is the real protocol, in the random-oracle model. In this game, all the instances of U and the trusted server S are modeled as the real execution in the random oracle. By definition of event $Succ_i$, which means that the adversary correctly guesses the bit b involved in the *Test-query*, we have

$$Adv_{\Pi,D}(\mathcal{A}) = 2\left|Pr[Succ_0]-\frac{1}{2}\right| \quad (1)$$

Game G_1: This game is as the same as the game G_0 except that we simulate the hash oracles h, h_1 and h_2 as usual by maintaining hash lists h_{List}, h_{1List} and h_{2List} with entries of the form (*Inp*, *Outp*). On hash query for which there exists a record (*Inp*, *Outp*) in the hash

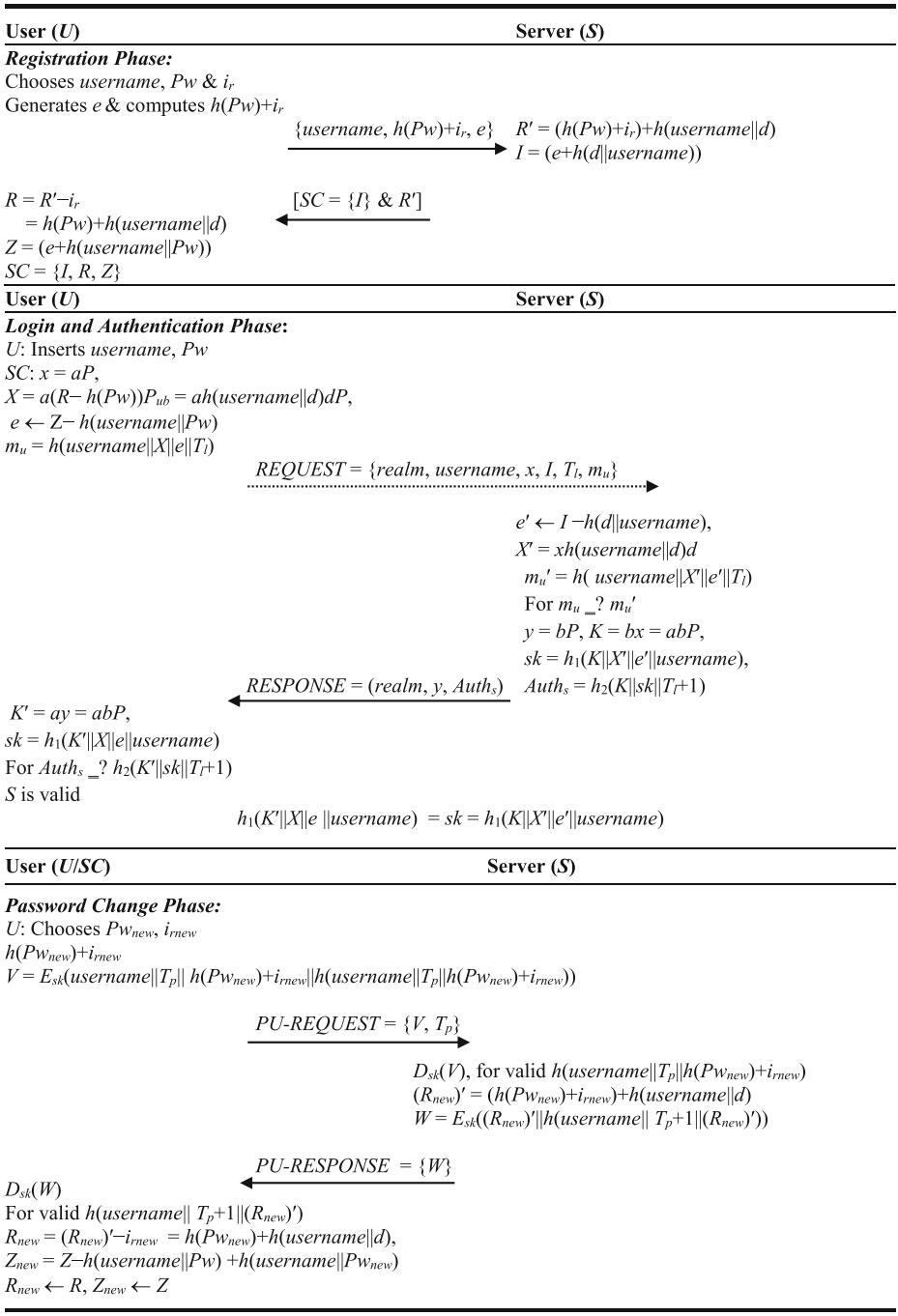


Fig. 1 The proposed scheme

list, return *Outp*. Otherwise, randomly choose $Outp \in \{0,1\}^k$, send it to $\mathcal{A}$ and store the new tuple (*Inp*, *Outp*) into the hash list. We also simulate all the instances, as the real

players would do, for the *Send-query* and for the *Execute*, *SendClient*, *SendServer*, *Reveal*, *Corrupt* and *Test* queries. From the viewpoint of the adversary, we easily see that the game is perfectly indistinguishable from the real attack. Hence

$$Pr[Succ_1] = Pr[Succ_0] \tag{2}$$

Game G_2: In this game, we simulate all the oracles in game G_1, except we cancel the game in which some collisions appear on the partial transcripts (x,y) and on hash values. According to the birthday paradox, the probability of collisions in output of hash oracles are at most $q_h^2/2^{k+1}$, $q_{h1}^2/2^{k+1}$ and $q_{h2}^2/2^{k+1}$ where q_h, q_{h1} and q_{h2} denote the maximum number of hash queries. Similarly, the probability of collisions in the transcripts is at most $(q_s+q_e)^2/2p^2$, where q_s represents the number of queries to the *SendClient* and *SendServer* oracles and q_e represents the number of queries to the *Execute* oracle. So we have

$$Pr[Succ_2]-Pr[Succ_1] \le \frac{q_h^2 + q_{h1}^2 + q_{h2}^2}{2^{k+1}} + \frac{(q_s+q_e)^2}{2p^2} \tag{3}$$

Game G_3: In this game, we consider the probability of the attacker forging all hash results without hash queries. We divide the game to two cases:

Case 1: To forge the message $\{realm, username, x, I, m_u\}$, *h(Pw), h(username||Pw)* and $h(username||X||e||T_l)$ should be queried. The probability for $h(username||X||e||T_l)$ is $\frac{q_s}{2^k}$, while the other two are $\frac{q_h}{2^k}$.
Case 2: To forge the message$\{realm, y, Auth_s\}$, $sk = h_1(K||X'||e'||username)$ and $Auth_s = h_2(K||sk||T_l + 1)$ should be asked. The probability for the former is $\frac{q_{h1}}{2^k}$ and for the latter is $\frac{q_s}{2^k}$ since $Auth_s$is transmitted as part of the message.

So, it can be easily seen that this game is perfectly indistinguishable from the previous game G_2. Hence,

$$|Pr[Succ_3]-Pr[Succ_2]| \le \frac{2q_h + q_{h1} + 2q_s}{2^k} \tag{4}$$

Game G_4: In this game, we consider that the adversary can ask the random oracles. Also we assume $\mathcal{A}$ can solve CDH problem with the advantage $Adv_{P,p}^{cdh}(t)$.Two cases can be divided to analyze this game:

Case 1: It is the online password guessing attack. Since the passwords are retrieved from D and there are q_s times for the attacker to try, the probability for this case is $\frac{q_s}{|D|}$.
Case 2: It is the off-line password guessing attack. $\mathcal{A}$ should query the h_1 query with asking K and that denotes the adversary breaks the CDH problem. We can find K by the bound $\frac{1}{q_{h1}}$. $\mathcal{A}$ can use two ways to finish that attack.

SubCase 1 : The first is querying $Execute(\Pi_U^i, \Pi_S^j)$. Since there are 6 scalar multiplications in one Execute process, the probability for this subcase is $q_{h1}Adv_{P,p}^{cdh}(t+6q_e t_{em})$.
SubCase 2: The second is querying Send queries successively. Since there are 3 scalar multiplications at most in one Send query, the probability for this subCase is $q_{h1}Adv_{P,p}^{cdh}(t+3q_s t_{em})$

So we can see that Game G_4 is indistinguishable from G_3, and

$$\begin{aligned}|\Pr[\mathrm{Succ}_4]-\Pr[\mathrm{Succ}_3]| &\le \frac{q_s}{|D|} + q_{h1}Adv^{cdh}_{P,p}(t+6q_e t_{em}) + q_{h1}Adv^{cdh}_{(P,p)}dh(t+3q_s t_{em}) \\ &\le \frac{q_s}{|D|} + 2q_{h1}Adv^{cdh}_{P,p}(t+(6q_e+3q_s)t_{em})\end{aligned} \tag{5}$$

Game G_5: In this game, we consider the forward security. According to the notion of Fresh, Corrupt queries can only be asked after Test query. So we use the old games. Like the SubCase 2 of Game G_4, the probability for we find *x,y, K* in the same session is bounded by $\frac{1}{(q_s+q_e)}{}^2$. We have

$$|\Pr[\mathrm{Succ}_5]-\Pr[\mathrm{Succ}_4]| \le 2q_{h1}(q_s+q_e)^2 Adv^{cdh}_{P,p}(t+(6q_e+3q_s)t_{em}) \tag{6}$$

$\mathcal{A}$ has no advantage in guessing *b* and $\Pr[\mathrm{Succ}_5] = 1/2$. Combining the above games, the theorem can be obtained.

9 Formal verification with Proverif

We give the formal verification of our scheme via the popular tool Proverif. Following codes can be checked via the reference (http://proverif.rocq.inria.fr/index.php)

```
(*———channels———*)
free ch: channel. (*public channal*)
free sch: channel [private]. (*private channel*)
(*———shared keys———*)
free sku: bitstring [private].(*the user's session key*)
free sks: bitstring [private]. (*the server's session key*)
(*———S's secret key———*)
free d:bitstring [private].
(*———constants———*)
free username:bitstring.
free PW:bitstring [private].
const P:bitstring. (*the base point*)
const realm: bitstring.
const one: bitstring. (*for replacing 1 in the authentication
process*)
table t(bitstring). (*table stored in S for auditing U*)
(*———functions———*)
fun h(bitstring):bitstring. (*hash function*)
```

```
  fun h1(bitstring):bitstring. (*hash function*)
  fun h2(bitstring):bitstring. (*hash function*)
  fun mul(bitstring,bitstring):bitstring. (*scalar
multiplication*)
  fun add(bitstring,bitstring):bitstring. (*addition*)
  fun sub(bitstring,bitstring):bitstring. (*substraction*)
  fun con(bitstring,bitstring):bitstring. (*string
concatenation*)
  (*————equations————————*)
  equation for all m:bitstring,n:bitstring; mul(m,(n,P)) =
mul(n,(m,P)).
  equation for all m:bitstring,n:bitstring; sub(add(m,n),n) = m.
  (*————————aims for verification————————*)
  query attacker(sku).
  query attacker(sks).
  query id:bitstring; inj-event(UserAuth(id))==>inj-
event(UserStart(id)).
  (*————————event————————*)
  event UserStart(bitstring). (*User starts authentication*)
  event UserAuth(bitstring). (*User is authenticated*)
  (*————User's process————*)
  let User=
  new se:bitstring;
  new ir:bitstring;
  out(sch,(username,add(h(PW),ir),se));
  in(sch,(rI:bitstring,rR':bitstring,Pub:bitstring));
  let R = sub(rR',ir) in
  let Z = add(se,h(con(username,PW))) in
  !
  (
  event UserStart(username);
  new a:bitstring;
  new Tl: bitstring;
  let x = mul(a,P) in
  let X = mul(a,mul(sub(R,h(PW)),Pub)) in
  let e = sub(Z, h(con(username,PW))) in
  let mu = h(con(con(con(username,X),e),Tl)) in
  let Request = (realm,username,x,rI,Tl,mu) in
  out(ch,Request);
  in (ch,(rrealm:bitstring,ry:bitstring,rAuths:bitstring));
  let K' = mul(a,ry) in
  let sku = h1(con(con(con(K',X), e), username)) in
  if rAuths = h2(con(con(K',sku),add(Tl,one))) then
  0
  ).
  (*————S's process————*)
  let SReg =
```

```
in(sch,(rusername:bitstring,rr1:bitstring,re:bitstring));
let sPub = mul(d,P) in
insert t(rusername);
let R' = add(rr1,h(con(rusername,d))) in
let I = add(re,h(con(d,rusername))) in
out (sch,(I,R',sPub)).
let SAuth =
in (ch,(xrealm:bitstring,xusername:bitstring,xx:
bitstring,xI:bitstring,xTl:bitstring,xmu:bitstring));
let e' = sub(xI,h(con(d,xusername))) in
let x' = mul(xx,mul(h(con(xusername,d)),d)) in
let mu' = h(con(con(con(username,x'),e'),xTl)) in
if xmu = mu' then
get t(=xusername) in
event UserAuth(xusername);
new b:bitstring;
let y = mul(b,P) in
let K = mul(b,xx) in
let sks = h1((con(con(con(K,x'), e'), xusername))) in
let Auths = h2(con(con(con(K,sks),sks),add(xTl,one))) in
let Response = (xrealm,y,Auths) in
out(ch,Response).
let S = SReg | SAuth.
process !User | !S
```

The results for the codes are listed as follows:

```
- Query inj-event(UserAuth(id)) ==> inj-event(UserStart(id))
nounif mess(sch[],(rusername_4217,rr1_4218,re_4219))/-5000
Completing...
Termination warning: begin(UserStart(username[]), @sid_301 =
@sid_5119, Pub = Pub_5120, rR' = rR'_5121, rI = rI_5118, @sid =
@sid_5122, @occ9 = @occ_cst) && mess(sch[],(rI_5118,rR'_5121,Pub_5120)) -> attacker(rI_5118)
Selecting 1
200 rules inserted. The rule base contains 177 rules. 14 rules in the queue.
Starting query inj-event(UserAuth(id)) ==> inj-event(UserStart(id))
RESULT inj-event(UserAuth(id)) ==> inj-event(UserStart(id)) is true.
- Query not attacker(sks[])
nounif mess(sch[],(rusername_12050,rr1_12051,re_12052))/-5000
Completing...
Termination warning: mess(sch[],(rI_12947,rR'_12948,Pub_12949)) -> attacker(rI_12947)
Selecting 0
200 rules inserted. The rule base contains 142 rules. 11 rules in the queue.
```

```
  Starting query not attacker(sks[])
  RESULT not attacker(sks[]) is true.
  - Query not attacker(sku[])
  nounif mess(sch[],(rusername_18830,rr1_18831,re_18832))/-5000
  Completing...
  Termination warning: mess(sch[],(rI_19727,rR'_19728,Pub_19729))
-> attacker(rI_19727)
  Selecting 0
  200 rules inserted. The rule base contains 142 rules. 11 rules in
the queue.
  Starting query not attacker(sku[])
  RESULT not attacker(sku[]) is true.
```

So our scheme passes the verification.

10 Comparisons of performance with some related SIP schemes

Here, we compare the proposed scheme for its strength with schemes proposed by Zhang et al. [38], Yeh et al. [34], Irshad et al. [16] and Arshad and Nikooghadam [3]. First of all, Table 2 introduces the notations with description for the purpose of comparison. Afterwards Tables 3 and 4 compare these schemes for computational complexity and security features respectively.

We begin the efficiency comparison of these schemes from the very first phase, that is, registration phase. During this phase, our scheme adds only one one-way hash operation at the user as compared to other four schemes [3, 16, 34, 38]. While at the server, our scheme remarkably cuts the computational cost in relation to the schemes [16, 34, 38] and adds only one extra one-way hash operation than in Arshad and Nikooghadam's scheme. Computational complexity/cost at the user during password update phase is least in Yeh et al.'s scheme, higher and quite same in schemes [3, 16, 38], and only two one-way hash operations are more in our scheme than in [3, 16, 38]. For the same phase, the computational complexity/cost at the server is least in Yeh et al.'s scheme, slightly more in Arshad and Nikooghadam's scheme, highest and exactly same in schemes [16, 38], for our scheme it is higher than in schemes [3, 34] but lower than in schemes [16, 38]. Although registration phase is deployed only when a user has to register at the server and password update phase is deployed only when a user wishes to

Table 2 Notations used in comparison

Notation	Description
t_{em}	Time complexity to execute an elliptic curve scalar point multiplication
t_{ea}	Time complexity to execute an elliptic curve point addition
t_{sy}	Time complexity to execute a symmetric key encryption or decryption
t_{mi}	Time complexity to execute a modular inversion
t_{ha}	Time complexity to execute a one-way hash function
t_{kh}	Time complexity to execute a keyed hash function also called message authentication code MAC
Yes	If a scheme resists an attack or offers a feature
No	If a scheme does not resist an attack or does not offer a feature
N/A	If an attack or a feature is not applicable in a scheme due to some missing characteristic

Table 3 Comparison of efficiency: memory space, communication cost and computational cost/complexity

Schemes→ ↓Computational complexity	Zhang et al.'s [38]	Yeh et al.'s [34]	Irshad et al.'s [16]	Arshad and Nikooghadam's [3]	Ours
Registration phase (U_i /SC_i)	$1t_{ha}$	$1t_{ha}$	$1t_{ha}$	$1t_{ha}$	$2t_{ha}$
Registration phase (S_i)	$1t_{em}$ + $1t_{mi}$ +$1t_{ha}$	$1t_{em}$ + $2t_{ha}$	$1t_{em}$ + $2t_{mi}$ +$1t_{ha}$	$1t_{ha}$	$2t_{ha}$
Password update phase (U_i /SC_i)	$2t_{sy}$ + $3t_{ha}$	$1t_{ha}$	$2t_{sy}$ + $3t_{ha}$	$5t_{ha}$	$2t_{sy}$ + $5t_{ha}$
Password update phase (S_i)	$1t_{em}$ + $2t_{sy}$ +$1t_{mi}$ + $3t_{ha}$	$2t_{ha}$	$1t_{em}$ + $2t_{sy}$ +$1t_{mi}$ + $3t_{ha}$	$3t_{ha}$	$2t_{sy}$ + $3t_{ha}$
Login-authentication phase (U_i /SC_i)	$5t_{em}$ + $1t_{ea}$ +$6t_{ha}$	$4t_{em}$ + $3t_{ea}$ +$6t_{ha}$	$3t_{em}$ + $2t_{kh}$ +$5t_{ha}$	$2t_{em}$ + $4t_{ha}$	$3t_{em}$ + $5t_{ha}$
Login-authentication phase (S_i)	$4t_{em}$ + $2t_{ea}$ +$4t_{ha}$	$4t_{em}$ + $3t_{ea}$ +$5t_{ha}$	$4t_{em}$ + $2t_{kh}$ +$3t_{ha}$	$2t_{em}$ + $1t_{mi}$ +$4t_{ha}$	$3t_{em}$ + $5t_{ha}$
Sum of computational complexity of login-authentication phase	$9t_{em}$ + $3t_{ea}$ +$10t_{ha}$	$8t_{em}$ + $6t_{ea}$ +$11t_{ha}$	$7t_{em}$ + $4t_{kh}$ +$8t_{ha}$	$4t_{em}$ + $1t_{mi}$ +$8t_{ha}$	$6t_{em}$ + $10t_{ha}$

update a new password in her smart card or memory device, most frequently used phase is the login-authentication phase. Computational complexity/cost for login-authentication phase at the user is least in Arshad and Nikooghadam's scheme, and noticeably higher in schemes [16, 34, 38]. In our scheme, at the user there is only one elliptic curve scalar point multiplication and one one-way hash operations extra than in Arshad and Nikooghadam's scheme. At the server, computational complexity/cost for this phase in our scheme is quite same to that in Arshad and Nikooghadam's scheme and least among the considered schemes. If we look at the overall computational complexity/cost of these schemes, we observe the least load in Arshad and

Table 4 Comparison of performance: security characteristics

Schemes→ ↓Security Characteristics	Zhang et al.'s [38]	Yeh et al.'s [34]	Irshad et al.'s [16]	Arshad and Nikooghadam's [3]	Ours
Resists password guessing attack	No	Yes	No	Yes	Yes
Resists user impersonation attack	No	No	No	Yes	Yes
Resists server spoofing attack	Yes	Yes	No	No	Yes
Resists replay attack	Yes	Yes	Yes	Yes	Yes
Resists insider attack	Yes	Yes	Yes	Yes	Yes
Resists stolen verifier attack	Yes	Yes	Yes	No	Yes
Denning-Sacco attack	Yes	Yes	Yes	Yes	Yes
Provides security of S's key	No	Yes	No	Yes	Yes
Provides forward secrecy	Yes	Yes	Yes	Yes	Yes
Provides mutual authentication	No	No	No	No	Yes
Provides Pw update facility	Yes	Yes	Yes	Yes	Yes
Provides secure Pw update facility	Yes	No	Yes	Yes	Yes
Offers single round authentication	No	No	Yes	No	Yes
Problem on which security relies	ECDLP	ECDLP	ECDLP	ECDLP	ECDLP

Nikooghadam's scheme. Our scheme uses two elliptic curve scalar point multiplication and two one-way hash operations extra than in [3] but it does not make use of costly modular inversion. Collectively, it can be stated that our scheme adds only little computational load and offers promising security features as the discussion along with Table 4 follows.

The extra computational load at some places at user/server in our scheme is justified as displayed in Table 4. Our scheme is safe from various potential attacks to which the other considered schemes are victim. None of the schemes [3, 16, 34, 38] except our's provides mutual authentication, the reason being the applicability of either the user impersonation attack or the server spoofing attack or both. Arshad and Nikooghadam's scheme has least computational load at most of the places (as shown in Table 3) but it is insecure to server spoofing and stolen verifier attack. Besides, this scheme requires one and half round-trip to complete the authentication process unlike Irshad et al.'s scheme. On the other hand, our scheme follows the single round-trip design of Irshad et al.'s scheme. Like other four schemes in Table 3, the security of our scheme is based on ECDLP and hence it exhibits the virtue of forward secrecy. It is clear from this table that our scheme is superior as compared to other schemes for different security attributes. The increased efficiency and security of our scheme over the other schemes [3, 16, 34, 38] is evident from the results displayed by Tables 3 and 4.

Like predecessor schemes given by Irshad et al.'s and Arshad and Nikooghadam's, in our scheme also, a user cannot freely change his/her password without server's assistance. In order to update a new password in his/her smart card, a user is required to interact with the server. On one hand, this is an additional load on the server and on the other hand this is a limitation for user. Since server has sufficiently large computational capacity and a user has to occasionally change his password, this does not affect our scheme much.

11 Conclusion

In this paper, we have focused on the shortcomings of both Irshad et al.'s and Arshad and Nikooghadam's ECC-based SIP authentication schemes for VoIP. We have revealed that Irshad et al.'s scheme not only suffers from user impersonation attack but threats like password guessing and server spoofing also exist. Further, we have exhibited that some secrets of both user and the server are at the risk of disclosure in their scheme. Next, Arshad and Nikooghadam's scheme is shown to be inflicted with server spoofing and stolen verifier attack. With a view to settle the security weaknesses of these schemes, we have designed an efficient SIP authentication scheme for VoIP based on Irshad et al.'s scheme. It has been displayed that the proposed scheme resists all the attacks and is free from all the weaknesses present in two aforementioned schemes. Like most of the SIP authentication schemes, the security of our scheme is also based on the Eliptic Curve Discrete Logarithm Problem (ECDLP) and offers forward secrecy. Our scheme is simple, high in performance and has provable security; hence it offers a viable authentication mechanism for SIP-services for VoIP.

In future, we would try to propose a SIP authentication scheme facilitating user to change his/her password freely without interacting with the server and yet preventing any unauthorized person to update a false password in user's smart card.

Acknowledgments The authors extend their sincere appreciations to the Deanship of Scientific Research at King Saud University for its funding this Prolific Research Group (PRG-1436-16). This research is also supported by the National Natural Science Foundation of China under Grant No. 61300220, and it is also supported by PAPD and CICAEET.

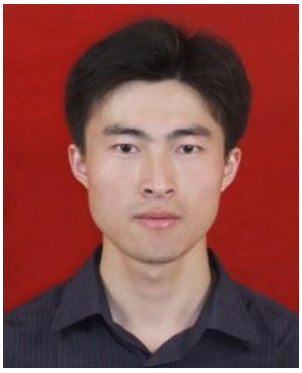
References

1. Abdalla M, Pointcheval D 2005 Interactive Diffie-Hellman assumptions with applications to password-based authentication. In: Proceedings of FC'05, LNCS 3570 341–356.
2. Arshad R, Ikram N (2013) Elliptic curve cryptography based mutual authentication scheme for session initiation protocol. Multimed Tools Appl 66(2):165–178
3. Arshad H, Nikooghadam M (2014) An efficient and secure authentication and key agreement scheme for session initiation protocol using ECC. Multimed Tools Appl. doi:10.1007/s11042-014-2282-x
4. Branovic I, Giorgi R, Martinelli E (2004) A workload characterization of elliptic curve cryptography methods in embedded environments. ACM SIGARCH Comput Archit News 32(3):27–34
5. Canetti R, Krawczyk H (2001) Analysis of key-exchange protocols and their use for building secure channels. In: Proc. Eurocrpt 2001, Lecture Notes in Computer Science, 2045: 453–474
6. Dalgic I, Fang H (1999) Comparison of H.323 and SIP for IP telephony signaling. In: Proc. of photonics East. SPIE, Boston
7. Denning D, Sacco G (1981) Timestamps in key distribution systems. Commun ACM 24:533–536
8. Diffie W, Hellman M (1976) New directions in cryptology. IEEE Trans Inf Theory 22(6):644–654
9. Durlanik A, Sogukpinar I (2005) SIP authentication scheme using ECDH. World Enformatika Soc Trans Eng Comput Technol 8:350–353
10. Farash MS, Attari MA (2013) An enhanced authenticated key agreement for session initiation protocol. Inform Technol Control 42(4):333–342
11. Franks J, Hallam-Baker P, Hostetler J, Lawrence S, Leach P, Luotonen A, Stewart L (1999) RFC2617: HTTP authentication: basic and digest access authentication. IETF
12. Garcia-Martin M, Henrikson E, Mills D (2003) Private header (P-Header) extensions to the session initiation protocol (SIP) for the 3rd-generation partnership project(3GPP). IETF RFC3455
13. Geneiatakis D, Dagiuklas T, Kambourakis G, Lambrinoudakis C (2006) Survey of security vulnerabilities in session initial protocol. IEEE Commun Surv Tutorials 8:68–81
14. Hankerson D, Menezes A, Vanstone S (2004) Guide to elliptic curve cryptography. LNCS, Springer, New York
15. He D, Chen J, Chen Y (2012) A secure mutual authentication scheme for session initiation protocol using elliptic curve cryptography. Secur Commun Netw 5(12):1423–1429
16. Irshad A, Sher M, Rehman E, Ashraf Ch S, Hassan MU, Ghani A (2013) A single round-trip SIP authentication scheme for Voice over Internet Protocol using smart card. Multimed Tools Appl. doi:10.1007/s11042-013-1807-z
17. Jo JH, Cho JS (2008) Cross-layer optimized vertical handover schemes between mobile Wimax and 3G networks. KSII Trans Internet Inf Syst (TIIS) 2(4):171–183
18. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48:203–209
19. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Proceedings of advances in cryptology, Santa Barbara, CA, U.S.A. 388–397
20. Lee CC (2009) On security of an efficient nonce-based authentication scheme for session initiation protocol. Int J Netw Secur 9:201–203
21. Liu FW, Koenig H (2011) Cryptanalysis of a SIP authentication scheme. In: 12th IFIP TC6/TC11 International Conference, CMS 2011, Ghent, Belgium 134–143
22. Messerges TS, Dabbish EA, Sloan RH (2002) Examining smart-card security under the threat of power analysis attacks. IEEE Trans Comput 51(5):541–552
23. Miller V (1986) Uses of elliptic curves in cryptography. In: Advances in cryptology CRYPTO'85, Lecture Notes in Computer Science Springer-Verlag 218: 417–426
24. NIST (1999) Recommended elliptic curves for federal government use Available on csrc.nist.gov
25. Pu Q, Wang J, Wu S (2013) Secure SIP authentication scheme supporting lawful interception. Secur Commun Netw 6:340–350
26. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M, Schooler E (2002) SIP: session initiation protocol. IETF RFC3261
27. Salsano S, Veltri L, Papalilo D (2002) SIP security issues: the SIP authentication procedure and its processing load. IEEE Netw 16(6):38–44

28. Sisalem D, Kuthan J, Ehlerts S (2006) Denial of service attacks targeting a SIPVoIP infrastructure: stack scenarios and prevention mechanisms. IEEE Netw J 20:26–31
29. Tang H, Liu X (2013) Cryptanalysis of Arshad et al'.s ECC-based mutual authentication scheme for session initiation protocol. Multimed Tools Appl 65(3):321–333
30. Tsai JL (2009) Efficient nonce-based authentication scheme for session initiation protocol. Int J Netw Secur 9:12–16
31. Wu L, Zhang Y, Wang F (2009) A new provably secure authentication and key agreement protocol for SIP using ECC. Comput Stand Interfaces 31(2):286–291
32. Xie Q (2012) A new authenticated key agreement for session initiation protocol. Int J Commun Syst 25(1): 47–54
33. Yang CC, Wang RC, Liu WT (2005) Secure authentication scheme for session initiation protocol. Comput Secur 24:381–386
34. Yeh HL, Chen TH, Shih WK (2014) Robust smart card secured authentication scheme on SIP using Elliptic Curve Cryptography. Comput Stand Interfaces 36:397–402
35. Yen SM, Joye M (2002) Checking before output may not be enough against fault-based cryptanalysis. IEEE Trans Comput 49(9):967–970
36. Yoon E, Shin Y, Jeon I, Yoo K (2010) Robust mutual authentication with a key agreement scheme for the session initiation protocol. IETE Tech Rev 27(3):203–213
37. Yoon EJ, Yoo KY, Kim C, Hong YS, Jo M, Chen HH (2010) A secure and efficient SIP authentication scheme for converged VoIP networks. Comput Commun 33:1674–1681
38. Zhang L, Tang S, Cai Z (2013) Efficient and flexible password authenticated key agreement for voice over internet protocol session initiation protocol using smart card. Int J Commun Syst. doi:10.1002/dac.2499

Dr. Saru Kumari is currently an Assistant Professor with the Department of Mathematics, Agra College, Agra, Dr. B. R. A. University, Agra, India. She received Ph.D. degree in Mathematics in 2012 from C.C.S. University, Meerut, Uttar Pradesh, India. She has published 35 papers in international journals and conferences including 22 research publications in SCI indexed journals. She is reviewer of many International journals, including the Journal of Network and Computer Applications (Elsevier), the Journal of Security and Communication Networks (Wiley), Computers and Electrical Engineering (Elsevier), Electronic Commerce Research Journal (Springer), Journal of International Journal of Distributed Sensor Networks (Hindawi) and International Journal of Communication Systems (Wiley). Her current research interests include Information Security, Digital Authentication, Security of Wireless Sensor Networks and Applied Mathematics.

Fan Wu received the Bachelor degree in Computer Science from Shandong University, Jinan, China in 2003, and received Master degree in Computer Software and Theory from Xiamen University, Xiamen, China in 2008. Now he is a lecturer in Xiamen Institute of Technology, Huaqiao University. His current research interests include information security, internet protocols, and network management.



Dr. Xiong Li received his master's degree in mathematics and cryptography from Shaanxi Normal University (SNNU) in 2009 and Ph.D. degree in computer science and technology from Beijing University of Posts and Telecommunications (BUPT) in 2012. Dr. Li now is a lecturer of Hunan University of Science and Technology (HNUST). He has published more than 15 referred journal papers. His research interests include cryptography and information security, etc.

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. He also received the Ph.D. degree in Cryptographic Mathematics at the Department of Mathematics and Computer Sciences of Tarbiat Moallem University in Iran in 2013. His research interests are Security Protocols and Provable Security Models.

Qi Jiang received the B.S. degree in Computer Science from Shaanxi Normal University in 2005 and Ph.D. degree in Computer Science from Xidian University in 2011. He is now an associate professor at School of Computer Science and Technology, Xidian University. His research interests include security protocols and wireless network security, etc.

Dr. Muhammad Khurram Khan is currently working at the Center of Excellence in Information Assurance, King Saud University, Saudi Arabia. He has edited seven books and proceedings published by Springer-Verlag and IEEE. He has published more than 200 papers in international journals and conferences and he is an inventor of 10 U.S./PCT patents. Dr. Khan is the Editor-in-Chief of a well-reputed journal 'Telecommunication Systems' (Springer). He is also on the editorial boards of several International SCI journals, including the Journal of Network and Computer Applications (Elsevier), Journal of Security and Communication Networks (Wiley), PLOSONE (USA), Computers and Electrical Engineering (Elsevier), Electronic Commerce Research (Springer), Scientific World Journal, Journal of Computing & Informatics, the Journal of Information Hiding and Multimedia Signal Processing (JIHMSP), and the International Journal of Biometrics (Inderscience). Dr. Khurram is one of the organizing chairs of several top-class international conferences and he is also on the program committee of dozens of conferences. He is a recipient of several national and international awards for his research contributions. In addition, he has been granted several national and international funding projects in the field of Cybersecurity. His current research interests include Cybersecurity, biometrics, multimedia security, and digital authentication.

Ashok Kumar Das received the Ph.D. degree in Computer Science and Engineering, the M.Tech. degree in Computer Science and Data Processing,and the M.Sc. degree in Mathematics, all from IIT Kharagpur, India.He is currently an Assistant Professor with the Center for Security,Theory and Algorithmic Research of the International Institute of Information Technology (IIIT), Hyderabad, India. He has authored over 75 papers in international journals and conferences in his research areas. His current research interests include cryptography, wireless sensor network security, proxy signature, hierarchical access control,data mining and remote user authentication. He received the Institute Silver Medal from IIT, Kharagpur. For more details, visit http://sites.google.com/site/iitkgpakdas/, http://www.iiit.ac.in/people/faculty/ashokkdas.