An improved and provably secure privacy preserving authentication protocol for SIP

Shehzad Ashraf Chaudhry[1] · Husnain Naqvi[1] · Muhammad Sher[1] · Mohammad Sabzinejad Farash[2] · Mahmood ul Hassan[1]

Received: 24 October 2014 / Accepted: 4 August 2015 / Published online: 7 September 2015

Abstract Session Initiation Protocol (SIP) has proved to be the integral part and parcel of any multimedia based application or IP-based telephony service that requires signaling. SIP supports HTTP digest based authentication, and is responsible for creating, maintaining and terminating sessions. To guarantee secure SIP based communication, a number of authentication schemes are proposed, typically most of these are based on smart card due to its temper resistance property. Recently Zhang et al. presented an authenticated key agreement scheme for SIP based on elliptic curve cryptography. However Tu et al. (Peer-to-Peer Netw Appl 1–8, 2014) finds their scheme to be insecure against user impersonation attack, furthermore they presented an improved scheme and claimed it to be secure against all known attacks. Very recently Farash (Peer-to-Peer Netw Appl 1–10, 2014) points out that Tu et al.'s scheme is vulnerable to server impersonation attack, Farash also proposed an improvement on Tu et al.'s scheme. However, our analysis in this paper shows that Tu et al.'s scheme is insecure against server impersonation attack. Further both Tu et al.'s scheme and Farash's improvement do not protect user's privacy and are vulnerable to replay and denial of services attacks. In order to cope with these limitations, we have proposed a privacy preserving improved authentication scheme based on ECC. The proposed scheme provides mutual authentication as well as resists all known attacks as mentioned by Tu et al. and Farash.

Keywords Authentication · Authenticated key agreement · Elliptic curve cryptography · Impersonation attack · Provable security · ProVerif

✉ Shehzad Ashraf Chaudhry
shahzad@iiu.edu.pk

Husnain Naqvi
husnain.naqvi@iiu.edu.pk

Muhammad Sher
m.sher@iiu.edu.pk

Mohammad Sabzinejad Farash
m.sabzinejad@gmail.com

Mahmood ul Hassan
mahmoodulhassan@iiu.edu.pk

[1] Department of Computer Science and Software Engineering, International Islamic University, Islamabad, Pakistan

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

1 Introduction

The session initiation protocol (SIP) has got much attractiveness during recent times, as it can achieve sessions including IP callas, multimedia distribution and conferences. SIP works on the standards of hyper text transport protocol (HTTP), which is based on request-response messages between client and server. Authentication is considered as a true vital facet for SIP, because the tangled participants must be validated even before start of the session. In SIP, client initiates the request message, while server asks for the legality of client by sending a challenge message, which also contains built-in server authentication information. The client after authenticating the server, sends a response message. The server validates the client by examining the response message. The SIP authentication makes a use of password based authentication along with other public key cryptography methods. The former, however, is more cost efficient than later, but the later provides more security. So we need a trade off between the two. The first password based authentication scheme was proposed by Chang et al. [12]. Successively a number of password based authentication schemes were proposed [2–9, 13, 14, 16–19, 22–28, 30–34, 37, 39, 42]. In earlier password based schemes, the server needs to store a verifier table having an entry for each client. Such schemes were proved to be vulnerable to stolen verifier attack, scalability issues and having high computational costs, because server has to secure the verifier table from unauthorized access by internal as well as external attackers. Further server has to create a distinct entry for each client, which limits the number of clients and needs extra computation for storing and comparing verifier table entries.

Recently Zhang et al. [40] proposed an efficient authentication scheme, the scheme made an efficient use of elliptic curve cryptography. They introduced the notion of authentication without storing any verifier table on server. Further they claimed their scheme to provide resistance to known attacks. But Irshad et al. [26], Zhang et al. [41] and Tu et al. [35] independently mentioned a number of weaknesses in Zhang et al.'s scheme [40]. Irshad et al. [26] claimed the scheme [40] to be vulnerable to replay and denial of services attack, Further Irshad et al. [26] proposed an improved single round scheme, but their scheme was vulnerable to impersonation attack as mentioned by Arshad and Nikooghadam [8], they also proposed an improved scheme. Unfortunately Arshad and Nikooghadam's scheme [8] once again introduced the verification tables on server side as well as having no provision for user anonymity. Zhang et al. [41] also proposed an improved scheme of [40], but their improved scheme was proved to be vulnerable to server impersonation attack by Farash [21]. Farash [21] then proposed an improved scheme, the scheme of Farash [21] once again does not provide user anonymity and is vulnerable to replay and denial of services attacks.

In 2014, Tu et al. [35] also proposed an improved scheme to improve the security of Zhang et al.'s scheme [40] and claimed it to be secure. However very recently Farash [20] mentioned that Tu et al.'s scheme is vulnerable to server impersonation attack, further Farash [20] proposed an improvement of Tu et al.'s scheme. In this paper we show that Tu et al.'s scheme [35] is vulnerable to server impersonation, replay and denial of services attacks as well as lacking user anonymity. Further, we analyze that Farash's improvement [20] on Tu et al.'s scheme [35] is lacking user anonymity and is also vulnerable to replay attack. Then an anonymous authenticated key agreement is proposed which is more secure and suitable for all lightweight environments. The rest of the paper is organized as follows. In Section 2 the procedure for SIP authentication and background for ECC has been described. Section 3 reviews Tu et al.'s scheme [35] followed by Farash's improvement [20], while cryptanalysis of Tu et al.'s and Farash's schemes is presented in Section 4. Section 5 describes our improved authentication scheme for SIP. In Section 6, we prove the security of the proposed scheme in random oracle model, we have also performed automatic security validation using automated tool ProVerif in same section. Section 7 presents the performance analysis of improved authentication scheme. Finally, we conclude in Section 8.

2 Preliminaries

In this section the SIP architecture [25] and the background for ECC [40] have been described.

2.1 SIP architecture

SIP is based on the request-response messages between client and server like HTTP. In SIP based authentication, a uniform resource identifier (URI) is used to identify users. The SIP design is compromising a number of contributors, including a client agent, redirect, proxy, registration and location servers. The client agent works as a terminal, the proxy server acts as an arbitrator amid the client and server, the caller location is notified by redirect server, while register server posts his new location to location server.

2.2 SIP authentication procedure

To get SIP services, a client initiates registration process with proxy server, the registration process includes a message from client containing his secret information like his identity/user name and password using some secure channel. After registration, the client is allowed to login with proxy server using pre-shared secrets and on some public channel. Then SIP session procedure is performed to locate an other SIP client to establish a session. The login/authentication procedure involves exchange of following messages among client and proxy server:

1. Client → Server: REQUEST
   A connection request is sent to server by client.
2. Server → Client: CHALLENGE (nonce, realm,info)
   For the received request, server sends a challenge message to client. The challenge message must contain some random nonce and realm, further it must also have some built in information to verify the legality of server.
3. Client → Server: RESPONSE (nonce, realm, username, info)

The client after receiving a challenge message, first verifies sender's legality then it spawns a response message.

4. For the received response message, the server using some pre-shared information verifies client's legality, if it proves to be falsify the session is terminated by client. Otherwise, a unique session key is established between both.

2.3 Elliptic curve cryptography

In this subsection the concepts relating to elliptic curve cryptography (ECC) pertinent to the manuscript are illustrated. ECC is based on some chosen real elliptic curve $E_p(a,b): y^2 = x^3 + ax + b \mod p$ where $a, b \in Z_p$ & $4a^3 + 27b^2 \mod p \neq 0$ for a large prime p. The integers a, b both defines the curve. A point (x, y) over $E_p(a,b)$ must verifies the former elliptic curve equation. The scalar multiplication is defined as the recurrent addition $vR = R+R+R.....+R$ (v times), where R is a point over $E_p(a,b)$ and $v \in F_p$. All the field parameters (p, a, b, R, n) are of the field F_p. ECC provides same level of security as of traditional public key cryptography like RSA, DSA and DH with lesser parameters size [36].

3 Tu et al.'s scheme & Farash's improvement

This section reviews Tu et al.'s [35] SIP authentication scheme using ECC and its improvement proposed by Farash [20]. Tu et al.'s scheme as illustrated in Fig. 1 consists of four phases: system initialization phase, registration phase, mutual authentication with key exchange phase and password changing phase. The notation guide for paper is described in Table 1.

Table 1 Notation guide

Notations	Description
n, p	Two large prime numbers
F_p	The finite prime field
$E_p(a,b)$	Elliptic Curve over F_p
G	Additive group of points over $E_p(a,b)$
P	Generator of G
PW_i	i^{th} client password
d_S	Server Private Key
$K_S = d_S P$	Server Public Key
$\|\|$	Concatenation operation
$\oplus$	XOR operation
$h(.), h_1(.)h_2(.)$	Three One way hash Functions
$\mathscr{U}$	The legal Client
$\mathscr{S}$	The legal Server
$\mathscr{A}$	The Adversary

3.1 System initialization phase

At start Server $\mathscr{S}$ selects an elliptic curve $E_p(a,b)$, then a point P as base point over selected curve. $\mathscr{S}$ chooses three one way hash functions. Then $\mathscr{S}$ selects a random private key $d_S \in Z_n^*$ and calculates public key $K_S = d_S P$. Finally $\mathscr{S}$ publishes $\{E_p(a,b), P, K_S, h(.), h_1(.), h_2(.)\}$ and keeps d_S secret.

3.2 Registration phase

Registration phase consists of two steps firstly client $\mathscr{U}$ choose a password PW_i, selects a random integer $a \in Z_n^*$. Then $\mathscr{U}$ computes $h(PW_i||a)$, and sends $h(PW_i||a)$, $username$ to $\mathscr{S}$ via some secure channel. When server $\mathscr{S}$ receives $h(PW_i||a)$ and $username$, $\mathscr{S}$ computes $R = (h(PW_i||a) + h(username||d_S))$, then stores R in smart card, and delivers the smart card to $\mathscr{U}$ through any secure channel. After receiving R, $\mathscr{U}$ stores a in smart card. Now, smart card contains (R, a).

3.3 Mutual authentication and key exchange phase

Step 1: The client $\mathscr{U}$ initiates authentication process by inserting his smart card in reader and entering the password PW_i, the smart card generate a random number $b \in Z_n^*$, then computes $V = bP$, $V' = b(R - h(PW_i||a)P)$ and $W = h(username|| V||V')$. Further $\mathscr{U}$ requests authentication by sending $username$, V & W in a request message to $\mathscr{S}$.

Step 2: After receiving the request $\mathscr{S}$ calculates $V'' = h(username||d_S)\, V$ and $W = h(username||V|| V'')$. $\mathscr{S}$ verifies $W \overset{?}{=} W'$, if not true $\mathscr{S}$ aborts the session. Otherwise, $\mathscr{S}$ choose two random number $c, r \in Z_n^*$, and calculates $C = cP$, $K = cV$ then $\mathscr{S}$ computes the shared key $SK = h_1(K||r||username)$, and $Auth_S = h_2(K||W||r||SK)$, finally it sends challenge message with $(realm, Auth_S, C, r)$ to client via public channel.

Step 3: $\mathscr{U}$ compute $K = bC$ and $SK = h_1(K||r||username)$ upon receiving the challenge message from $\mathscr{S}$. $\mathscr{U}$ further verifies $Auth_S \overset{?}{=} h_2(K||W||r||SK)$, if the relationship proves to be falsify, the session is aborted by $\mathscr{U}$. Otherwise, $\mathscr{U}$ computes $Auth_U = h_2(K||W||r + 1||SK)$, it further sends the response message $(realm, Auth_S)$ to $\mathscr{S}$. $\mathscr{U}$ keeps SK as shared key with $\mathscr{S}$.

Step 4: When $\mathscr{S}$ receives the response message it checks $h_2(K||W||r + 1||SK) \overset{?}{=} Auth_U$,

Client $\mathscr{U}$	*Server* $\mathscr{S}$
Registration Phase:	
Select *username* and Password PW_i	
Select a a random number $a \in Z_n^*$	
$\xrightarrow{h(PW_i\|\|a), username}$	
	Computes $R = (h(PW_i\|\|a) + h(username\|\|d_S))P$
	Stores R in the smart card
$\xleftarrow{SmartCard}$	
store a in smart card	
Mutual Authentication and Key Exchange Phase:	
Input user name and PW_i	
Generate a random number $b \in Z_n^*$	
$V = bP$	
$V' = b(R - h(PW_i\|a)P)$	
$W = h(username\|V\|V')$	
$\xrightarrow{REQUEST(username, V, W)}$	
	$V'' = h(username\|d_S)V$
	$W' = h(username\|V\|V'')$
	$Check\ W \overset{?}{=} W'$
	Generate two random variables $c, r \in Z_n^*$
	$C = cP$
	$K = cV$
	$SK = h_1(K\|r\|username)$
	$Auth_S = h_2(K\|W'\|r\|SK)$
$\xleftarrow{CHALLENGE(realm, Auth_S, C, r)}$	
$K = bC$	
$SK = h_1(K\|r\|username)$	
$CheckAuth_S \overset{?}{=} h_2(K\|W\|r\|SK)$	
$Auth_U = h_2(K\|W\|r+1\|SK)$	
$\xrightarrow{RESPONSE(realm, Auth_U)}$	
	$CheckAuth_U \overset{?}{=} h_2(K\|W'\|r+1\|SK)$
$\longleftarrow$ $SK = h_1(K\|r\|username)$	$\longrightarrow$

Fig. 1 Tu et al.'s scheme

if relationship does not exist the session is aborted by $\mathscr{S}$. Otherwise, $\mathscr{S}$ stores session key SK.

3.4 Password change phase

A password change request is initiated after generation of a session key. Following steps are performed between $\mathscr{U}$ and $\mathscr{S}$ for successful password update.

Step 1: $\mathscr{U}$ selects a new password PW_n and two random numbers $a_n, N_n \in Z_n^*$, then $\mathscr{U}$ computes, $C_u = E_{SK}(username\|N_n\|h(PW_n\|a_n)\|h(username\|N_n\|h(PW_n\|a_n)))$. Finally $\mathscr{U}$ sends password change request $\{C_u, N_n\}$ to $\mathscr{S}$.

Step 2: For the received password change request $\{C_u, N_n\}$. $\mathscr{S}$ first decrypts C_u, then checks the validity of message tag $h(username\|N_n\|h(PW_n\|a_n))$. If it is valid $\mathscr{S}$ computes $R_n = (h(PW_n\|a_n) + h(username||d_S))P$ and $C_S = E_{SK}(R_n\|h(username\|N_n + 1\|R_n))$. Finally $\mathscr{S}$ sends C_S to $\mathscr{U}$.

Step 3: Upon receiving C_S, $\mathscr{U}$ decrypts it and verifies the tag $h(username\|N_n + 1\|R_n)$, if it is valid. $\mathscr{U}$ stores R_n and a_n in smart card.

3.5 Farash's improvement

This subsection reviews Farsh's improvement on Tu et al.'s scheme. Farash slightly modified the authentication phase of Tu et al.'s scheme. Farash's modification is an alternation in computation of $Auth_S$ shown as follows:

$$Auth_S = h_2(K||V''||r||SK)$$

While there is no change in system initialization, registration and password change phases.

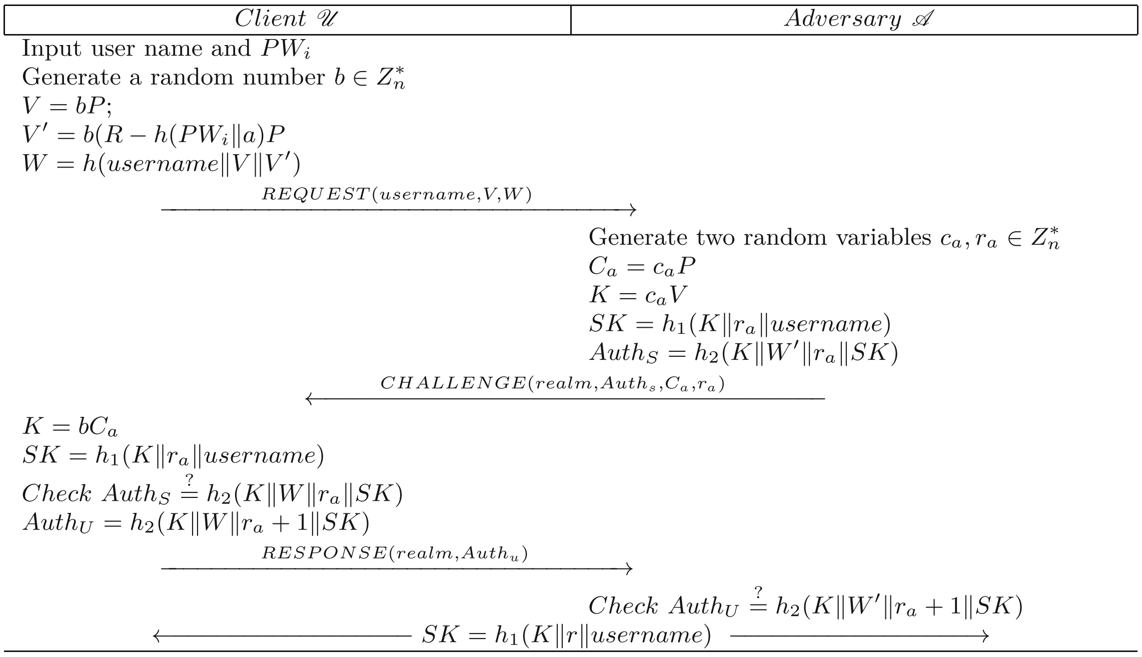
Client $\mathscr{U}$
Adversary $\mathscr{A}$
Input user name and PW_i
Generate a random number $b \in Z_n^*$
$V = bP$;
$V' = b(R - h(PW_i \| a)P$
$W = h(username \| V \| V')$
$\xrightarrow{REQUEST(username, V, W)}$
Generate two random variables $c_a, r_a \in Z_n^*$
$C_a = c_a P$
$K = c_a V$
$SK = h_1(K \| r_a \| username)$
$Auth_S = h_2(K \| W' \| r_a \| SK)$
$\xleftarrow{CHALLENGE(realm, Auth_s, C_a, r_a)}$
$K = bC_a$
$SK = h_1(K \| r_a \| username)$
$Check\ Auth_S \overset{?}{=} h_2(K \| W \| r_a \| SK)$
$Auth_U = h_2(K \| W \| r_a + 1 \| SK)$
$\xrightarrow{RESPONSE(realm, Auth_u)}$
$Check\ Auth_U \overset{?}{=} h_2(K \| W' \| r_a + 1 \| SK)$
$\longleftarrow SK = h_1(K \| r \| username) \longrightarrow$

Fig. 2 Server impersonation attack on Tu et al.'s scheme

4 Cryptanalysis of Tu et al.'s scheme & Farash's improvement

This section shows that an adversary can easily launch impersonation attack on Tu et al.'s scheme. We show that the adversary can easily masquerade as a legitimate server to share a session key. Further, we show that Tu et al.'s scheme and Farash's improvement both are lacking user anonymity and are vulnerable to replay attack and denial of services attacks.

4.1 Weaknesses of Tu et al.'s scheme

4.1.1 Server Impersonation Attack

By impersonation attack, an active adversary $\mathscr{A}$ can easily badge itself as a legal server without knowing the private key of server. As illustrated in Fig. 2, adversary $\mathscr{A}$ do following steps in order to masquerade the legal server $\mathscr{S}$ to share the session key with the client $\mathscr{U}$.

Step 1: Initially when a legal client $\mathscr{U}$ sends $REQUEST$ $(username, V, W)$ to the server $\mathscr{S}$, the attacker $\mathscr{A}$ intercept the message and selects two random numbers $c_a, r_a \in Z_n^*$. $\mathscr{A}$ further calculates $C_a = c_a P$, $K = c_a V$, $SK = h_1(K||r||username)$ and $Auth_S = h_2(K||W'||r||SK)$

Step 2: $\mathscr{A}$ sends $CHALLENGE(realm, Auth_S, C_a, r_a)$ to $\mathscr{U}$.

Step 3: Upon receiving the message $\mathscr{U}$ calculates $K = bC$ and $SK = h_1(K||r||username)$, then $\mathscr{U}$ checks $Auth_S \overset{?}{=} h_2(K||W||r||SK)$, it is obvious that $Auth_S$ hold. $\mathscr{U}$ further computes $Auth_U = h_2(K||W||r+1||SK)$.

Step 4: $\mathscr{U}$ sends $RESPONSE(realm, Auth_U)$ to $\mathscr{S}$.

Step 5: $\mathscr{A}$ intercepts the response message, the shared key between $\mathscr{U}$ and $\mathscr{A}$ is $SK = h_1\ (K||r||username)$.

Therefore, $\mathscr{A}$ successfully launched server impersonation attack and exchanged the session key $SK = h_1(K||r||username)$ with legal user $\mathscr{U}$.

4.1.2 No provision for user anonymity

Along with traditional security, user anonymity and privacy has emerged as an extremely important factor to be considered. Without privacy and anonymity user's sensitive personal information can be accessed by an adversary by just analyzing the session information. Specially in mobile communication, the attacker may become able to identify $\mathscr{U}$'s login history, his movement patterns, current location and so on. Furthermore, such sensitive information may be misused by the adversary. Tu et al.'s scheme did not consider these loopholes hence lacking user anonymity.

4.1.3 Replay attack and denial of service attack

In Tu et al.'s scheme, an active attacker $\mathscr{A}$ after intercepting a login request $REQUEST(username, V, W)$ can replay it later on, because the request does not contain any time stamp. Off course $\mathscr{A}$ will not be able to stake the session key as such replay will be fixed in response message $RESPONSE(realm, Auth_U)$ by the attacker, but such attack can hoax $\mathscr{S}$ and $\mathscr{U}$ to perform step 2 and 3 of authentication phase resulting into a counterfeit utilization of computation power as well as communication and storage resources. A simultaneous execution of a large number of such attacks can even lead to denial of services attacks causing access prevention to the legal client.

4.2 Weaknesses of Farash's scheme

4.2.1 No provision for user anonymity

Farash presented an improvement of Tu et al.'s scheme. Unfortunately in his improvement, Farash did not consider the the importance of user anonymity and just change the computation of $Auth_S$, while $username$ is sent in plain text to server. Therefore Farash's improvement is also lacking user anonymity, which can cause serious threats as discussed earlier in Section 4.1.2.

4.2.2 Replay attack and denial of service attack

Similar to Tu et al.'s scheme, in Farash's scheme an active attacker $\mathscr{A}$ after intercepting a login request $REQUEST(username, V, W)$ can replay it later on, forcing $\mathscr{S}$ to process the request and send the challenge message to $\mathscr{U}$, because the request does not contain any time stamp. Which not only burdens the system but can also cause denial of services to client.

5 Proposed scheme

The security breaches of Tu et al.'s and Farash's schemes are because the security of their schemes relies on public parameters V, W and $username$ transmitted on an insecure channel. In Tu et al.'s scheme V and W are also involved in the computation of SK and $Auth_S$. So an adversary can easily generate SK and $Auth_S$ in order to masquerade itself as the legal server. Similarly, the absence of time stamp in both Tu et al.'s and Farash's schemes resulted into burdening the system and replay as well as denial of service attacks. To improve Tu et al.'s scheme, we alternated the transmission of W and $username$ by $\overline{W}$ and $\overline{username}$, which provides resistance to impersonation and replay attacks as well as provides proper user anonymity. We have amended only registration and mutual authenticated key exchange phases, proposed scheme works as follows:

5.1 Registration phase

Registration phase consists of two steps firstly client $\mathscr{U}$ choose a password PW_i, selects a random integer $a \in Z_n^*$. Then $\mathscr{U}$ computes $h(PW_i\|a)$, and sends $h(PW_i\|a)$, $username$ to $\mathscr{S}$ via some secure channel. Upon reception of registration request message $h(PW_i\|a), username$. Server $\mathscr{S}$ selects random $r \in Z_n^*$ and computes $\overline{username} = Enc_{d_S}(username\|r)$, $R = (h(PW_i\|a) + h(username\|d_S))$. Further $\mathscr{S}$ stores R and $\overline{username}$ in smart card, and deliver the smart card to $\mathscr{U}$ through any secure channel. After receiving smart card, $\mathscr{U}$ stores a in it. Finally, smart card contains $(R, \overline{username}, a)$.

5.2 Mutual authentication & key exchange phase

Step 1: $\mathscr{U} \rightarrow \mathscr{S}$: $\{\overline{username}, V, \overline{W}, t_i\}$
The client $\mathscr{U}$ initiates authentication process by inserting his smart card (SC) in the reader and entering the password PW_i. SC then generates a random number $b \in Z_n^*$, and computes:

$$V = bP \tag{1}$$
$$V' = b(R - h(PW_i||a)P) \tag{2}$$
$$W = h(username||V||V') \tag{3}$$
$$\overline{W} = h_1(W \oplus V \oplus t_i) \tag{4}$$

where t_i is freshly generated time stamp. Further $\mathscr{U}$ requests authentication by sending $\overline{username}$, V and $\overline{W}$, t_i in request message to $\mathscr{S}$.

Step 2: $\mathscr{S} \rightarrow \mathscr{U}$:$\{realm, Auth_S, C, r, Z\}$
After receiving the request $\mathscr{S}$ first generates a new time stamp t_s and then compares it with received t_i. If the difference between both is with in a threshold time period Δ. $\mathscr{S}$ considers the time stamp fresh and proceeds with the login request. Otherwise, $\mathscr{S}$ aborts the session. For valid time stamp $\mathscr{S}$ proceeds with login request as follows:

$$username\|r = Dec_{d_S}(\overline{username}) \tag{5}$$
$$V'' = h(username||d_S)V \tag{6}$$
$$W' = h(username||V||V'') \tag{7}$$

Further, $\mathscr{S}$ verifies $W \overset{?}{=} h_1(W' \oplus V \oplus t_i)$, if not true $\mathscr{S}$ aborts the session. Otherwise, $\mathscr{S}$ chooses three random numbers $c, r, r_n \in Z_n^*$ and computes:

$$C = cP \tag{8}$$

Client $\mathscr{U}$	*Server* $\mathscr{S}$

Registration Phase:

Select *username* and Password PW_i

Select a a random number $a \in Z_n^*$

$$\xrightarrow{h(PW_i||a), username}$$

select some random $r \in Z_n^*$

$\boxed{\overline{username} = Enc_{d_S}(username\|r)}$

Computes $R = (h(PW_i||a) + h(username||d_S))P$

Stores R and $\boxed{\overline{username}}$ in the smart card

$$\xleftarrow{SmartCard}$$

store a in smart card

Mutual Authentication and Key Exchange Phase:

Input user name and PW_i

Generate a random number $b \in Z_n^*$

$V = bP$

$V' = b(R - h(PW_i\|a)P$

$W = h(username\|V\|V')$

$\boxed{\overline{W} = h_1(W \oplus V \oplus t_i)}$

$$\xrightarrow{REQUEST(\overline{username}, V, \overline{W}, t_i)}$$

For $t_s - t_i \leq \Delta$

$\boxed{username\|r = Dec_{d_S}(\overline{username})}$

$V'' = h(username\|d_S)V$

$W' = h(username\|V\|V'')$

Check $\overline{W} \overset{?}{=} h_1(W' \oplus V \oplus t_i)$

Generate two random variables $c, r \in Z_n^*$

$C = cP$

$K = cV$

$SK = h_1(K\|r\|username\|t_i)$

$Auth_S = h_2(K\|W'\|r\|SK\|t_i)$

choose new random $r_n \in Z_p^*$

$\boxed{Z = Enc_{d_S}(username\|r_n) \oplus W'}$

$$\xleftarrow{CHALLENGE(realm, Auth_S, C, r, Z)}$$

$K = bC$

$SK = h_1(K\|r\|username\|t_i)$

Check $Auth_S \overset{?}{=} h_2(K\|W\|r\|SK\|t_i)$

$\boxed{\overline{username} = Z \oplus W}$

$$\longleftarrow \quad SK = h_1(K\|r\|username\|t_i) \quad \longrightarrow$$

Fig. 3 Proposed scheme

$$K = cV \tag{9}$$

$$SK = h_1(K||r||username\|t_i) \tag{10}$$

$$Auth_S = h_2(K\|W'\|r\|SK\|t_i) \tag{11}$$

$$Z = Enc_{d_S}(username\|r_n) \oplus W' \tag{12}$$

Finally $\mathscr{S}$ sends $\{realm, Auth_S, C, r, Z\}$ to client via public channel.

Step 3: $\mathscr{U}$ Compute $K = bC$ and session key $SK = h_1(K\|r\|username\|t_i)$ upon receiving the challenge message from server, then it verifies $Auth_S \overset{?}{=} h_2(K\|W||r||SK\|t_i)$, if the relationship proves to be falsify, the session is aborted by $\mathscr{U}$. Otherwise $\mathscr{U}$ replaces $\overline{username} = Z \oplus W$. Finally SK is set as shared key with $\mathscr{S}$

6 Security analysis

This section analyzes the security of proposed scheme, the scheme provides mutual authentication, resist user and server impersonation attacks and is secure against stolen verifier, man-in-middle and off line password guessing attack, the scheme also provides perfect forward secrecy. We have proved the security of proposed scheme in random oracle model as well by using automated tool ProVerif. Further we have also performed informal security comparisons with existing schemes.

6.1 Provable security model

To analyze the security of the proposed scheme, we adopted the formal security model introduced in [10, 11].

6.1.1 Security model

There are two participants in the proposed authentication protocol $\mathcal{P}$: a client $\mathscr{U}$ and a server $\mathscr{S}$. During execution of $\mathcal{P}$, there may be several instances of each participant, where each instance is linked with a number z and is termed as an oracle jumbled in a divergent execution of $\mathcal{P}$. We outline U^x as the x^{th} instance of $\mathscr{U}$, similarly S^y is outlined as y^{th} instance of $\mathscr{S}$, we also term I^z for both the instances U^x and S^y with eradication of differences. There can be three possible outcomes of an oracle, accept, reject or $\perp$. An oracle ranges to an accept form, if it receives a righteous message. The wrong message lead to reject form, while $\perp$ state appears if no decision is made or no result returned.

Even before execution of $\mathcal{P}$, $\mathscr{U}$ owns a $username$, PW_i, while the smart card SC contains $R, \overline{username}, a$. $\mathscr{S}$ is having a private and public key pair d_S and $K_S = d_S P$. There are finite number of password, while the password dictionary $\mathcal{D}$ is of size $|\mathcal{D}|$. $\mathscr{S}$ is assumed to be secure.

According to adversary capabilities, the attacker $\mathscr{A}$ is having full control over public communication channel. $\mathscr{A}$ can initiate and arbitrate the session between $\mathscr{U}$ and $\mathscr{S}$. $\mathscr{A}$ aims to violate communication privacy and session key secrecy. $\mathscr{A}$ can make a number of queries in oracles and may get replies. The list of such queries is itemized below:

– $h(s/s1/s2, rec)$: It is a hash oracle and it results into some arbitrary result r. Employment of this query builds a record (rec, r), depending upon the first parameter, it generates three different hash lists h_{slist}, h_{s1list} and h_{s2list}. Dealing of these records is in proof process.
– $Send(U^x/S^y, msg/SCLD)$: This query replicates the active attack on communication, it yields the message that U^x or S^y generates upon reception of message msg, if second argument of $Send$ query is $SCLD$, the output is the message $\{\overline{username}, V, \overline{W}, t_i\}$ in step 1 of authentication phase. The query normally finishes as the steps in mutual authentication phase of $\mathcal{P}$.
– $Execute(U^x, S^y)$: This query enables the attacker to perform a passive attack on the communication channel. By simulating $Execute$, $\mathscr{A}$ can access the messages exchanged over insecure communication channel between $U^x or S^y$.
– $Reveal(I^x)$: This query designates the known session key attack. By this query, $\mathscr{A}$ can acquire the computed session key between U^x and S^y.
– $Corrupt(SC)$: This query enables $\mathscr{A}$ to obtain all the parameters stored in smart card (SC).
– $Test(I^z)$: This query stands for obtaining the session key. The simulation of $Test$ query results into $\perp$, if I^z did not generate a session key. Otherwise, it outputs into flipping of a coin Ω. If $\Omega = 1$, $Test$ query outputs the existent session key, if $\Omega = 0$ uniform random string is returned, whose length is same as the actual session key. $\mathscr{A}$ is allowed to ask $Test$ query only once on the $fresh$ oracle.

Following are some definitions used to prove the security of proposed scheme.

– $Partnering$: Each participating instance U^x S^y is having a partner identity pid_U^x or pid_S^yalog with a session key sk_U^x or sk_S^y an identifier sid_U^x or sid_S^y, which is accepted and agrees a session key. U^x and S^y are termed as partners if and only if $sid_U^x = sid_S^y$, $pid_S^y = U^x$, $pid_U^x = S^y$ and $sk_U^x = sk_S^y$.
– $fresh$: Any instance I^z is believed as $fresh$, if no $Reveal$ query happened on I^z.
– $PAP - security$: The advantage for $\mathscr{A}$ to break the security of $\mathcal{P}$ is defined as the probability that can acceptably guess the result of flipping of coin Ω by $Test(I^z)$, where I^z is $fresh$ as well as accepted. Let $\mathscr{A}$ outputs Ω', the advantage is as follows:

$$Adv_{\mathcal{P}}^{PAP}(\mathscr{A}) = |2Pr[\Omega = \Omega'] - 1| \tag{13}$$

The proposed authentication protocol is designated as $PAP - secure$ if $Adv_{\mathcal{P}}^{PAP}(\mathscr{A})$ is negligible.
– We define the Elliptic curve computational Diffie-Hellman (ECCDH) assumption as follows: Given three point αP, βP and P over an elliptic curve $E_p(a, b)$, where $\alpha, \beta \in Z_n^*$, the probability $\mathscr{A}$ can compute $\alpha\beta P$ in polynomial time t can be defined as $Adv_{\mathscr{A}}^{ECCDH}(t)$. The ECCDH assumption implies that $Adv_{\mathscr{A}}^{ECCDH}(t) \leq \epsilon$.

6.1.2 Security proof

Theorem 1 *The password engaged by $\mathscr{U}$ is from a password dictionary $\mathcal{D}$ having size $|\mathcal{D}|$. Let l_{hs} be the length of hash value, $\mathcal{P}$ is the proposed authentication protocol. An adversary $\mathscr{A}$ during polynomial time t can make maximum*

q_{snd} Send queries, q_{exe} Execute queries and q_{hs}, q_{hs1}, q_{hs2} hash queries. $\mathscr{A}$'s advantage is as follows:

$$Adv_{\mathscr{P}}^{PAP}(\mathscr{A}) \leq \frac{q_{hs}^2 + q_{hs1}^2 + q_{hs2}^2}{2^{l_{hs}}} + \frac{(q_{snd}+q_{exe})^2}{2(p-1)} + 2q_{exe}\cdot Adv_{\mathscr{A}}^{ECCDH}(\overline{W}) + 2\max\left\{\frac{q_{hs1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\} \quad (14)$$

Proof For proof, we mark a sequence of games ranging from G_0 to G_4, the event $Succ_i$ means that $\mathscr{A}$ correctly gausses Ω during G_i effectively in $Test$. As per the requirements for our model, there is no need for $\mathscr{A}$ to compute identity of the client because there is only one user. The games for our proof are listed below:

– Game G_0: It is the real protocol in random oracle model. Here, we selected random coin flipped value Ω'. We realize that $\mathscr{A}$'s advantage to guess Ω correctly is as follows:

$$Adv_{\mathcal{P}}^{PAP}(\mathscr{A}) = 2Pr[Succ_0] - 1 \quad (15)$$

– Game G_1: We simulate all oracles for queries. Also, three lists are used to store the record (rec, r) formed after query mentioned in the security model. h_{slist}, h_{s1list} and h_{s2list} are used to store answers to h oracle. On hash query, if there exists a record (rec, r) in corresponding hash list, r is returned, otherwise a random value r' is returned to $\mathscr{A}$ and a record is added to corresponding hash list against r'. When h oracle is queried by $\mathscr{A}$ then the record in h_{Alist}.From $\mathscr{A}$'s view point G_0 and G_1 are not distinguishable through the simulation, so

$$Pr[Succ_1] = Pr[Succ_0] \quad (16)$$

– Game G_2:Some of the collisions are avoided during G_2, which is aborted when some collisions ensued on transcripts (V, C) and on hash values. As $b, c \in [1, p-1]$ and the length of each hash value is l_{hs}. Referring the birthday paradox, the the maximum collision probability in result of hash oracles are $q_{hs}^2/2^{l_{hs}+1}$, $q_{hs1}^2/2^{l_{hs}+1}$ and $q_{hs2}^2/2^{l_{hs}}$. Similarly, the maximum collision probability in the transcripts is $(q_{snd}+q_{exe})^2/2(p-1)$. So we have

$$|\Pr[Succ_2] - \Pr[Succ_1]| \leq \frac{q_{hs}^2 + q_{hs1}^2 + q_{hs2}^2}{2^{l_{hs}+1}} + \frac{(q_{snd}+q_{exe})^2}{2(p-1)}. \quad (17)$$

– Game G_3: This game is aborted, if $\mathscr{A}$ computes correct messages with out hash oracles, the game is divided into two cases according to two messages

1. To forge $Send(S^y, (\overline{username}, V, \overline{W}, t_i))$ query, $\mathscr{A}$ must make $(W \oplus V \oplus t_i)$ and V' queries, Or we can say that $(W \oplus V \oplus t_i) \in h_{Alist}$ should be true. If we have not found it as a role of server, the probability is up to $\frac{q_{snd}}{2^{l_{hs}}}$. Note that $\mathscr{S}$ does not know pw_u, so the record $(username_u||pw_u||a_u, *)$ can not be checked. The probability is $\frac{q_{hs}}{2^{l_{hs}}}$.
2. To forge $Send(U^i, (realm, Auth_S, C, r, Z))$, A must make $(K\|W'\|r\|SK\|t_i)$. The probabilities are upper bounded by $\frac{q_{hs1}}{2^{l_{hs}}}$ and $\frac{q_{snd}}{2^{l_{hs}}}$ respectively for the matter that the two records do not exist in h_{Alist}.

Hence games G_3 and G_2 are indistinguishable unless the messages are forged without hash queries. So we have

$$|Pr[Succ_3] - Pr[Succ_2]| \leq \frac{2q_{snd} + 2q_{hs1}}{2^{l_{hs}}} \quad (18)$$

– Game G_4: For this game, ECCDH is brought in, $\mathcal{A}$ is allowed to make oracles normally. $\mathscr{A}$ can acquire session key SK, if he win this game. To win this game $\mathscr{A}$ has to solve ECCDH. To compute SK, $\mathcal{A}$ must ask $(kP\|r\|username_u)$ query. If this record exists in the list h_{Alist}, $\mathscr{A}$ breaks ECCDH problem. The difference between the game G_4 and the game G_3 is as follows:

$$|\Pr[Succ_4] - \Pr[Succ_3]| \leq q_{exe}\cdot Adv_{\mathscr{A}}^{ECCDH}(\overline{W}). \quad (19)$$

There are two possible cases where the adversary distinguishes the real session key SK and the random key as follows:

Case 1. the adversary queries $(K, r, username)$ to h_{s1}. The probability that this event occurs is $\frac{q_{hs1}}{2^{l_{hs}}}$.

Case 2. the adversary asks $Send(U^x$ query and successfully impersonates U to S. The adversary is not allowed to reveal static key PW_i of $\mathscr{U}$. Thus, in order to impersonate $\mathscr{U}$, the adversary has to obtain some information of the password PW_i of $\mathscr{U}$. The probability is $1/|D|$. Since there are at most q_{snd} sessions of this kind, the probability that this event occurs is lower than $q_{snd}/|D|$

As a conclusion,

$$\Pr[Succ_4] = \frac{1}{2} + \max\left\{\frac{q_{hs1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\}. \quad (20)$$

Combining the equations (15), (16), (17), (18), (19) and (20) the announced result as follows:

$$
\begin{aligned}
Adv_{\mathscr{P}}^{PAP}(\mathscr{A}) &= \Pr[Succ_0] - 1| \\
&= 2\left|\Pr[Succ_0] - \Pr[Succ_4] + \max\left\{\frac{q_{h1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\}\right| \\
&\le 2\left(|\Pr[Succ_0] - \Pr[Succ_4]| + \max\left\{\frac{q_{hs1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\}\right) \\
&\le 2\Bigg(|\Pr[Succ_1] - \Pr[Succ_2]| + |\Pr[Succ_3] \\
&\quad - \Pr[Succ_4]| + \max\left\{\frac{q_{hs1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\}\Bigg) \\
&\le \frac{q_{hs}^2 + q_{hs1}^2 + q_{hs2}^2}{2^{l_{hs}}} + \frac{(q_{snd} + q_{exe})^2}{2(p-1)} \\
&\quad + 2q_{exe} \cdot Adv_{\mathscr{A}}^{ECCDH}(\overline{W}) + 2\max\left\{\frac{q_{hs1}}{2^{l_{hs}}}, \frac{q_{snd}}{|D|}\right\}.
\end{aligned}
$$

□

6.2 Automated security verification

In this subsection, we have performed the automated security analysis of the proposed scheme using the widespread automated tool ProVerif [1]. ProVerif can verify privacy and security of authentication schemes [15, 38]. ProVerif is constructed over the well known applied π calculus, which can support many cryptographic primitives including one way functions, encryption, digital signatures, Diffie-Hellman and many more. In-order to prove the security of the proposed scheme, we have imprinted the steps as mentioned in Section 5 and shown in Fig. 3, to model the message exchanged, we have introduced two channels for communication among $\mathscr{U}$ and $\mathscr{S}$, a secure channel $CH1\_Sec$ for registration phase and a public channel $CH2\_Pub$ for login and authentication phase.

```
free CH1_Sec:channel [private].
free CH2_Pub:channel.
```

Constants and variables used in proposed scheme are defined as follows:

```
free username:bitstring.
free PWi:bitstring [private].
const Ks:bitstring.
const ds:bitstring [private].
const p:bitstring.
const q:bitstring.
const P:bitstring.
```

We have modeled $\mathscr{U}$'s password PWi and $\mathscr{S}$'s private key as private, while *username*, $\mathcal{S}$'s public key *Ks*, ECC parameters *p,q* and base point *P* are declared as public and accessible to all participants including adversary. ProVerif defines cryptographic primitives as constructors, destructors and equations. We have defined the constructors *H,H1,H2, concat, add, ExcOr, multi, ECMP, subtract* and *syme* for three hash functions, a point addition, exclusive or, integer multiplication, point multiplication, subtract and symmetric key encryption respectively. While symmetric decryption is defined by the destructor *symd*. To exploit the property of exclusive or, $(a \oplus b) \oplus b = a$, we have defined an equation (*ExcOR*).

```
fun H(bitstring):bitstring.
fun H1(bitstring):bitstring.
fun H2(bitstring):bitstring.
fun concat(bitstring,bitstring):bitstring.
fun add(bitstring,bitstring):bitstring.
fun ExcOR(bitstring,bitstring):bitstring.
fun multi(bitstring,bitstring):bitstring.
fun ECMP(bitstring,bitstring):bitstring.
fun subtract(bitstring,bitstring):bitstring.
fun syme(bitstring,bitstring):bitstring.
reduc forall m:bitstring,key:bitstring;
symd(syme(m,key),key)=m.
equation forall a:bitstring,b:bitstring;
ExcOR(ExcOR(a,b),b)=a.
```

Following four events are defined to analyze the security of our proposed scheme.

```
event begin_User(bitstring).
event end_User(bitstring).
event begin_Server(bitstring).
event end_Server(bitstring).
```

We have modeled two events for each $\mathscr{U}$ and $\mathscr{S}$, *begin_User(bitstring).* and *end_User(bitstring).* for start and end of $\mathscr{U}$, similarly two events are defined for $\mathscr{S}$ *begin_User(bitstring).* and *end_User(bitstring).* to start and end of $\mathscr{S}$. Protocol's authenticity can be proved by revealing the corresponding relation ship between each participant's begin and end event.

We have defined two distinct processes to model the participants, the process *pClientU* symbolizes $\mathscr{U}$, while the process *pServerS* models $\mathscr{S}$. The process *pClientU* first registers by selecting *a* and *HUPa* and sends *username* along with *HUPa* to $\mathscr{S}$ on the secure channel *CH1_Sec*. After registration *pClientU* pledges the login & authentication process by computing V, V', W, x_W. Then *pClientU* sends pseudo username *x_username, V,x_W,ti* to *pServerS*. Further *pClientU* computes *K,SK* after receiving response message *xAuths,xC,xr,xZ* from *pServerS*. Then *pClientU* verifies the validity of *xAuths*, if validity holds server is authenticated, then *pClientU* further replaces *X_username* with new value sent by server.

```
let pClientU=
(*Registration*)
new a:bitstring;
let HUPa = H(concat(PWi,a)) in
out(CH1_Sec,(HUPa,username));
in(CH1_Sec,(xR:bitstring,x_username:bitstring));
(*Login and Mutual Authentication*)
event begin_User (username);
new b:bitstring;
let V=ECMP(b,P) in
let V'=multi(b,subtract(xR,ECMP(H(concat(PWi,a)),P)))␣in
let␣W=H(concat(username,(V,V'))) in
new ti: bitstring;
let x_W=H1(ExcOR(W,(V,ti))) in
out(CH2_Pub,(x_username,V,x_W,ti));
in(CH2_Pub,(xAuths:bitstring,xC:bitstring,xr:bitstring, xZ
    :bitstring));
let K=multi(b,xC) in
let SK=H1(concat(K,(xr,username,ti))) in
if(xAuths= H2(concat(K,(W,xr,SK,ti))) ) then
let X_username=ExcOR(xZ,W) in
event end_User(username)
else 0.
```

The server process *pServerS* frights after receiving registration message from *pClientU*, it computes pseudo user name *X_username,R*. Then sends both *X_username,R* to *pClientU*. Then after *pServerS* computes V'', W' and checks the validity of *x_W*, if its valid *pServerS* computes*C,K,SK,Auths* after receiving login request message from *pClientU*. Finally *pServerS* computes *Z* and sends *Auths,C,r,Z* to *pClientU*.

```
let pServerS=
(*Registration*)
in(CH1_Sec,(xHUPa:bitstring,xusername:bitstring));
new r:bitstring;
let X_username = syme(concat(username,r),ds) in
let R = ECMP(add(xHUPa,H(concat(xusername,ds))),P) in
out(CH1_Sec,(R,X_username));
(* Login & Mutual Authentication *)
in(CH2_Pub,(x_username:bitstring,xV:bitstring,x_W:
      bitstring,xti:bitstring));
event begin_Server(ds);
let (xxusername:bitstring,xr:bitstring) =symd(x_username,
      ds) in
let V'' = multi(H(concat(xxusername,ds)),xV) in
let W'=H(concat(username,(xV,V'')))in
if(x_W=H1(ExcOR(W',(xV,xti)))) then
new c:bitstring;
new r1:bitstring;
let C = ECMP (c,P) in
let K = multi (c,xV) in
let SK = H1(concat(K,(r,xxusername,xti))) in
let Auths = H2(concat(K,(W',r,SK,xti)))in
new␣rn:bitstring;
let␣Z=syme(ExcOR(concat(xxusername,rn),W'),ds) in
out(CH2_Pub,(Auths,C,r,Z));
event end_Server(ds)
else 0.
```

The modeled protocol is replicated as the unbounded parallel execution of three processes shown as follows:

```
process ((pClientU)||(!pServerS))
```

We have defined following queries to verify the security and correctness of our protocol.

```
(*Queries*)
free SK:bitstring [private].
query attacker(SK).
query id:bitstring; inj-event(end_User(id)) ==>inj-event(
      begin_User(id)) .
query id:bitstring; inj-event(end_Server(id)) ==>inj-event
      (begin_Server(id)) .
```

The query attacker(Z) models the attacker capabilities, where Z is unknown to attacker, if the predicate *not attacker(Z)* results into false, then *Z* is revealed to attacker hence authenticity and secrecy is not maintained, if it results into true, the protocol is secure.The attacker knows all public parameters. The attacker query is applied on *SK* I (the session key). Further two queries on inj-event verifies that each event started and terminated successfully and the protocol possesses the correctness property.

The results are as follows:

1. inj-event(end_Server(id)) ==> inj-event(begin_Server(id)) is true.
2. inj-event(end_User(id_1780)) ==> inj-event(begin_User(id_1780)) is true.
3. not attacker(SK[]) is true.

The results (1) and (2) verifies that both server and user processes started and terminated successfully, while (3) verifies that SK (session key) is not revealed to adversary and secrecy is maintained.

6.3 Further security discussion

This subsection analyzes the security of proposed scheme. The analysis verifies that proposed scheme resists all known attacks, while ensuring user anonymity and untraceability. Table 2 illustrates the security comparisons of proposed scheme with related existing schemes. It is evident from Table 2 that only proposed scheme provides user anonymity and untraceability, while all other schemes are lacking user anonymity and untraceability. Similarly, only proposed scheme and Irshad et al.'s scheme [26] provides resistance against replay and denial of service attacks. The provable security analysis is provided by proposed and Farash's scheme [20] only, likewise only Farash [20, 21], Arshad et al. and proposed schemes are resistant to impersonation attacks, In short except proposed scheme, all other schemes are lacking at least two security requirements.

6.3.1 Mutual authentication

In proposed scheme, initially the user sends $\{\overline{username}, V, \overline{W}\}$, where $\overline{W}$ involve user's password PW_i, the adversary with out knowing the user password can not generate valid V and $\overline{W}$ pair. Similarly without the knowledge of server secret key d_S the adversary can not generate valid W. Further, $Auth_S$ can be generated after having valid W. So the user is authenticated by checking $\overline{W} = h_1(W \oplus V \oplus t_i)$, while the server by verifying $Auth_S = h_2(K\|W\|r\|SK\|t_i)$. Hence proposed scheme provides mutual authentication.

6.3.2 Impersonation attack

The adversary may impersonate as a legal user if it successfully generate valid V, W pair, but it requires user PW_i and information stored in smart card, so the scheme resist user impersonation attack, similarly the adversary can not impersonate as a legal server, if it become able to generate valid $Auth_S$, but $Auth_S$ involves the computation of $V'' = h(username\|d_S)V$ and $W' = h(username\|V\|V'')$, both of these require the secret key d_S of the server.

6.3.3 Privileged insider attack

Instead of password we just send $h(PW_i||a)$ during registration phase, so privileged insider can not have access to user password PW_i.

6.3.4 Stolen verifier attack

In proposed scheme no verifier table is maintained for user's password, $\mathscr{S}$ makes use of his secret key d_S for authentication. Therefore, the proposed scheme is secure against stolen verifier attack.

Table 2 Security comparisons

Schemes → Security Properties ↓	Our	[20]	[35]	[40]	[8]	[26]	[41]	[21]
Resists insider attack	✓	✓	✓	✓	✓	✓	✓	✓
Resists off line guessing attack	✓	✓	✓	✓	✓	✓	✓	✓
Resists user impersonation attack	✓	✓	✓	x	✓	x	✓	✓
Resists server impersonation attack	✓	✓	x	x	✓	✓	x	✓
Resists known key attack	✓	✓	✓	x	✓	✓	✓	✓
Resists Smart card lost attack	✓	✓	✓	x	✓	x	✓	✓
Resists man in middle attack	✓	✓	x	x	✓	x	x	✓
Provided user anonymity	✓	x	x	x	x	x	x	x
Provides forward secrecy	✓	✓	✓	✓	✓	✓	✓	✓
Provable Security	✓	✓	x	x	x	x	x	x
No Verifier stored at server	✓	✓	✓	✓	x	✓	✓	✓
Resists Strong Replay & denial of services attacks	✓	x	x	x	x	✓	x	x

6.3.5 Man-in-middle attack

In proposed scheme, valid V' can only be generated by using user password, while V'' can only be computed by server master key d_S. Therefore, the scheme withstand the man-in-middle attack.

6.3.6 Replay attack

The adversary can easily intercept the request message $\{\overline{username}, V, \overline{W}, t_i\}$. Also the adversary can easily replicates the request message. When such replicated request reaches, the server simply verifies the freshness of t_i, as t_i is old dated, server will know its a replay message. Further Adversary can generate new time stamp t_a and can replay request after changing t_i by t_a, as time stamp is fresh, server after computing V'' and W', checks $\overline{W} \overset{?}{=} h_1(W' \oplus V \oplus t_a)$. The adversary will not pass this test, because $\overline{W}$ contains in built t_i. Similarly adversary will not be able to computeom session key $SK = h_1(K||r||username\|t_i)$ without knowing user password PW_i and either the value of b or c obtaining b from $V = bP$ and c from $C = cP$, the adversary has to solve untraceable elliptic curve discrete logarithm problem. Similarly if the adversary intercept $\{realm, Auth_S, C, r\}$ and sends it to user, the replayed message can not pass the $Auth_S \overset{?}{=} h_2(K\|W\|r\|SK\|t_i)$ test. Therefore the scheme is secure against replay attack.

6.3.7 Off-line password guessing attack

Assuming the adversary get smart card and obtained the secret information (R, a). Further the adversary intercept the message $\{\overline{username}, V, \overline{W}, t_i\}$. In order to guess user password PW_i, the adversary still need server secret key d_S to check password validity from $V'' = h(username||d_S)V$. Therefore the proposed scheme resist off-line password guessing attack.

Table 3 Computational cost analysis

	Client	Server	Total	Running time
Farash [21]	$3T_{ecpm} + 5T_h$	$4T_{ecpm} + 1_{ecpa} + 5T_h$	$7T_{ecpm} + 1_{ecpa} + 10T_h$	≈ 15.8408
Zhang et al. [41]	$3T_{ecpm} + 4T_h$	$4T_{ecpm} + 1_{ecpa} + 4T_h$	$7T_{ecpm} + 1_{ecpa} + 8T_h$	≈ 15.6292
Irshad et al. [26]	$3T_{ecpm} + 6T_h$	$4T_{ecpm} + 5T_h$	$7T_{ecpm} + 11T_h$	≈ 15.6073
Arshad et al. [8]	$2T_{ecpm} + 4T_h$	$2T_{ecpm} + 4T_h$	$4T_{ecpm} + 8T_h$	≈ 8.9224
Zhang et al. [40]	$4T_{ecpm} + 1T_{ecpa} + 6T_h$	$4T_{ecpm} + 1T_{ecpa} + 5T_h$	$8T_{ecpm} + 2T_{ecpa} + 11T_h$	≈ 17.8909
Tu et al. [35]	$3T_{ecpm} + 1T_{ecpa} + 5T_h$	$3T_{ecpm} + 5T_h$	$6T_{ecpm} + 1T_{ecpa} + 10T_h$	≈ 13.4078
Farash [20]	$3T_{ecpm} + 1T_{ecpa} + 5T_h$	$3T_{ecpm} + 5T_h$	$6T_{ecpm} + 1T_{ecpa} + 10T_h$	≈ 13.4078
Proposed	$3T_{ecpm} + 1T_{ecpa} + 5T_h$	$3T_{ecpm} + 5T_h + 2T_{sed}$	$6T_{ecpm} + 1T_{ecpa} + 10T_h + 2T_{sed}$	≈ 13.417

Table 4 Storage & communication cost analysis

Schemes →	Our	[20]	[35]	[40]	[8]	[26]	[41]	[21]
Memory needed in smart Card	480	320	320	320	160	480	320	320
Communication overhead(Bits)	1184	1056	1056	1056	832	1508	1056	1056
Exchanged Messages	2	3	3	3	3	2	3	3

6.3.8 Perfect forward secrecy

Perfect forward secrecy means that if long term secret keys of one or more legal users are compromised, the secrecy of old session keys will not be affected. For estimating an old session key, the attacker needs to guess more than one session parameters, the random number b is separately generated by the client $\mathcal{U}$ for each session, while server generates random number c exclusive for each session. In order to find b from $V = bP$ or c from $C = cP$, the adversary has to solve a hard problem $ECDLP$. Hence, the attacker could not estimate the previous session keys out of compromised current session key and/or the password.

7 Comparative performance analysis

7.1 Computation cost analysis

Following notations are used for computation cost analysis,

- T_{ecpm} : Time for Elliptic curve point multiplication
- T_{ecpa} : Time for Elliptic curve point addition
- T_h : Time for one way hash function
- T_{sed} : Time for a symmetric encryption/decryption operation

According to Kilinc and Yanik [29], T_{ecpm} : takes 2.226 ms, T_{ecpa} takes 0.0288 ms, T_{sed} : takes 0.0046 ms, while T_h : takes 0.0023 to complete their processing on a personal computer with Dual CPU E2200 2.20 GHz processor, 2048 MB of RAM and the Ubuntu Operating system by using PBC Library.

Computation cost of proposed scheme as compared with schemes proposed [8, 20, 21, 26, 35, 40, 41] is summarized in Table 3, the proposed scheme over casted [21, 26, 40, 41]. Arshad et al.'s [8] scheme takes least computation resources because in their scheme the verifier is stored at server. The proposed scheme incurs only $2T_{sed}$ more on server side as compared with Tu et al.'s and Farsh's schemes [20, 35].

7.2 Storage & communication cost analysis

We have also compared the storage and computation costs of proposed scheme with recent related schemes [8, 20, 21, 26, 35, 40, 41]. We selected hash function SHA-1, whose out put is 160 bit long, further we employed AES as symmetric key algorithm of block size 128 bits. We selected 64 bits *username* length, while size of realm is 32 bits. The NIST recommended size for ECC operations is 160 bits. The storage and communication cost analysis is illustrated in Table 4. Proposed scheme incurs some extra storage in smart card and having some more communication overhead as compared with schemes [8, 20, 21, 35, 40, 41], while it is having equal storage and less communication cost as compared with [26]. Further Only proposed scheme and Irshad et al.'s scheme [26] achieves authentication in only 2 messages, while rest of the schemes [8, 20, 21, 35, 40, 41] achieves same in 3 messages. Hence proposed scheme is more suitable for practical environments.

8 Conclusion

This paper analyzed Tu et al.'s authentication and key agreement scheme for SIP and Farash's improvement on Tu et al.'s scheme. We have shown that Tu et al.'s scheme is vulnerable to server impersonation attack. Further, we have also analyzed that both Tu et al.'s scheme and Farash's improvement do not provide user anonymity and are vulnerable to replay as well as denial of services attack. To overcome the weaknesses, we have proposed an improved privacy preserving scheme, which ensures mutual authentication and is secure against all known attacks.

Acknowledgments Authors are thankful to the anonymous reviewers and the editor Prof. Zhenfu Cao for their valuable recommendations to improve the quality, correctness, presentation and readability of the manuscript.

References

1. Abadi M, Blanchet B, Comon-Lundh H (2009) Models and proofs of protocol security: A progress report. In: Computer aided verification. Springer, pp 35–49
2. Abi-Char PE, Mhamed A, El-Hassan B (2007) A fast and secure elliptic curve based authenticated key agreement protocol for low power mobile communications. In: The 2007 international conference on Next generation mobile applications, services and technologies, 2007. NGMAST'07. IEEE, pp 235–240
3. Abi-Char PE, Mhamed A, El-Hassan B (2007) A secure authenticated key agreement protocol based on elliptic curve cryptography. In: 3rd international symposium on information assurance and security, 2007. IAS 2007. IEEE, pp 89–94
4. ul Amin N, Asad M, Din N, Ashraf Ch S (2012) An authenticated key agreement with rekeying for secured body sensor networks based on hybrid cryptosystem. In: 9th IEEE international conference on networking, sensing and control (ICNSC), 2012. IEEE, pp 118–121
5. Amin R, Biswas G (2015) An improved rsa based user authentication and session key agreement protocol usable in tmis. J Med Syst 39(8):1–14
6. Amin R, Biswas G (2015) A secure three-factor user authentication and key agreement protocol for tmis with user anonymity. J Med Syst 39(8):1–19
7. Amin R, Biswas G (2015) A novel user authentication and key agreement protocol for accessing multi-medical server usable in tmis. J Med Syst 39(3):1–17
8. Arshad H, Nikooghadam M (2014) An efficient and secure authentication and key agreement scheme for session initiation protocol using ecc. Multimedia Tools and Applications:1–17. doi:10.1007/s11042-014-2282-x
9. Bala S, Sharma G, Verma AK (2013) An improved forward secure elliptic curve signcryption key management scheme for wireless sensor networks. In: IT convergence and security 2012. Springer, pp 141–149
10. Bellare M, Rogaway P (1994) Entity authentication and key distribution. In: Advances in Cryptology, CRYPTO 93. Springer, pp 232–249
11. Bellare M, Rogaway P (1995) Provably secure session key distribution: the three party case. In: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. ACM, pp 57–66
12. Chang CC, Wu TC (1991) Remote password authentication with smart cards. IEEE Proceedings Computers and Digital Techniques 138(3):165–168
13. Chaudhry SA (2015) Comment on 'robust and efficient password authenticated key agreement with user anonymity for session initiation protocol-based communications'. IET Commun 9(1):1034–1034. doi:10.1049/iet-com.2014.1082
14. Chaudhry SA, Farash MS, Naqvi H, Kumari S, Khan MK (2015) An enhanced privacy preserving remote user authentication scheme with provable security. Security and Communication Networks:1–13. doi:10.1002/sec.1299
15. Chaudhry SA, Farash MS, Naqvi H, Sher M (2015) A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography. Electron Commer Res:1–27. doi:10.1007/s10660-015- 9192-5
16. Chaudhry SA, Naqvi H, Shon T, Sher M, Farash M (2015) Cryptanalysis and improvement of an improved two factor authentication protocol for telecare medical information systems. J Med Syst 39(6):66. doi:10.1007/s10916-015-0244-0
17. Chou CH, Tsai KY, Lu CF (2013) Two id-based authenticated schemes with key agreement for mobile environments. J Supercomput 66(2):973–988
18. Chuang MC, Chen MC (2014) An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. Expert Systems with Applications 41(4):1411–1418
19. Debiao H, Jianhua C, Jin H (2012) An id-based client authentication with key agreement protocol for mobile client–server environment on ecc with provable security. Information Fusion 13(3):223–230
20. Farash M (2014) Security analysis and enhancements of an improved authentication for session initiation protocol with provable security. Peer-to-Peer Netw Appl:1–10. doi:10.1007/s12083-014-0315-x
21. Farash MS (2014) An improved password-based authentication scheme for session initiation protocol using smart cards without verification table. Int J Commun Syst. doi:10.1002/dac.2879
22. Farash MS, Attari MA (2014) A secure and efficient identity-based authenticated key exchange protocol for mobile client–server networks. J Supercomput 69(1):395–411. doi:10.1007/s11227-014-1170-5
23. Farash MS, Chaudhry SA, Heydari M, Sadough SMS, Kumari S, Khan MK (2015) A lightweight anonymous authentication scheme for consumer roaming in ubiquitous networks with provable security. Int J Commun Syst. doi:10.1002/dac.3019
24. Harn L, Lin HY (2001) Authenticated key agreement without using one-way hash functions. Electron Lett 37(10):629–630
25. Irshad A, Sher M, Faisal MS, Ghani A, Ul Hassan M, Ashraf Ch SA (2014) A secure authentication scheme for session initiation protocol by using ecc on the basis of the tang and liu scheme. Security and Communication Networks 7(8):1210–1218. doi:10.1002/sec.834
26. Irshad A, Sher M, Rehman E, Ch SA, Hassan MU, Ghani A (2015) A single round-trip sip authentication scheme for voice over internet protocol using smart card. Multimedia Tools and Applications 74(11):1–18. doi:10.1007/s11042-013-1807-z
27. Islam S, Biswas G (2011) A more efficient and secure id-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem. J Syst Softw 84(11):1892–1898
28. Jiang Q, Ma J, Tian Y (2014) Cryptanalysis of smart-card-based password authenticated key agreement protocol for session initiation protocol of Zhang et al. Int J Commun Syst 28(7):1340–1351
29. Kilinc H, Yanik T (2014) A survey of sip authentication and key agreement schemes. IEEE Commun Surv Tutorials 16(2):1005–1023. doi:10.1109/SURV.2013.091513.00050
30. Liao YP, Wang SS (2010) A new secure password authenticated key agreement scheme for sip using self-certified public keys on elliptic curves. Comput Commun 33(3):372–380
31. Mehmood Z, Nizamuddin N, Ch S, Nasar W, Ghani A (2012) An efficient key agreement with rekeying for secured body sensor networks. In: Second International Conference on digital information processing and communications (ICDIPC), 2012. IEEE, pp 164–167
32. Nicanfar H, Leung VC (2013) Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system. IEEE Trans Smart Grid 4(1):253–264
33. Ryu EK, Yoon EJ, Yoo KY (2004) An efficient id-based authenticated key agreement protocol from pairings. In: Networking technologies, services, and protocols; performance of computer and communication networks; mobile and wireless communications networking 2004. Springer, pp 1458–1463
34. Sharma G, Bala S, Verma AK (2013) Extending certificateless authentication for wireless sensor networks: A novel insight. International Journal of Computer Science Issues (IJCSI) 10(6)

35. Tu H, Kumar N, Chilamkurti N, Rho S (2014) An improved authentication protocol for session initiation protocol using smart card. Peer-to-Peer Netw Appl:1–8. doi:10.1007/s12083-014-0248-4
36. William S, Stallings W (2006) Cryptography and network security, 4/E. Pearson education india
37. Xie Q (2012) A new authenticated key agreement for session initiation protocol. Int J Commun Syst 25(1):47–54
38. Xie Q, Hu B, Dong N, Wong DS (2014) Anonymous three-party password-authenticated key exchange scheme for telecare medical information systems. PloS one 9(7):e102,747
39. Xu X, Zhu P, Wen Q, Jin Z, Zhang H, He L (2014) A secure and efficient authentication and key agreement scheme based on ecc for telecare medicine information systems. J Med Syst 38(1):1–7
40. 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 27(11):2691–2702
41. Zhang L, Tang S, Cai Z (2014) Cryptanalysis and improvement of password-authenticated key agreement for session initiation protocol using smart cards. Security and Communication Networks 7(12):2405–2411. doi:10.1002/sec.951
42. Zhao Z (2014) An efficient anonymous authentication scheme for wireless body area networks using elliptic curve cryptosystem. J Med Syst 38(2):1–7

Shehzad Ashraf Chaudhry received his MS Computer Science with distinction, from International Islamic University Islamabad, Pakistan in 2009 and was awarded *Gold Medal*. Currently he is working as Lecturer at the Department of Computer Science and Software Engineering, International Islamic University, Islamabad. He authored more than 25 scientific publications including 9 in SCI/E journals. His research interests include Lightweight Cryptography, Elliptic/Hyper Elliptic Curve Cryptography, Multimedia Security, E- Payment systems, MANETs, SIP authentication, IP Multimedia sub-system and Next Generation Networks.

Dr. Husnain Naqvi received his Ph.D. from The University of Auckland, New Zealand. Currently he is working as Assistant Professor at the Department of Computer Science, International Islamic University, Islamabad. He authored more than 30 scientific publications published in different international journals and proceedings. His broad research interests include Sensor Networks, Collaborative Communications, Lightweight Cryptography, Beamforming and Space Time Block Codes.

Muhammad Sher is a Professor in international Islamic University, Islamabad, Pakistan. He is having more than 120 scientific publications. He received his Ph.D. Computer Science from TU Berlin, Germany and M. Sc. from Quaid-e-Azam University, Islamabad. His research interests include Next Generation Networks, IP Multimedia Sub-systems and Network Security.

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.

Mahmood Ul Hassan is currently a PhD scholar at International Islamic University Islamabad Pakistan. He has been working as Assistant Director at Department of Computer Science & Software Engineering, International Islamic University, Islamabad, Pakistan. His research interests Next Generation Networks, quality of service, IP Multimedia Subsystem, wireless sensor networks, and mobile ad hoc networks.