ECC-BASED THREE-FACTOR AUTHENTICATION SCHEME FOR MULTI-SERVER ENVIRONMENT

Rahul Kumar*, Mridul K. Gupta and Saru Kuamri

Chaudhry Charan Singh University, Meerut, India

Abstract

Due to advances in computing technology and constraints in the design of the authentication protocols for single-server environment, the authentication protocols for multi-server settings have been a preferred field of research. Recently, Ali and Pal designed a three factor-based authentication protocol for multi-server environment using ECC and they claimed that their protocol is secure against numerous attacks. They also asserted that their protocol is quite efficient. In this paper, we investigate Ali and Pal's protocol and point out that their protocol is not secure against replay attack and session-specific temporary information attack. We also present an improvement of Ali and Pal's protocol.

Keywords

Multi-Server, Authentication, Replay Attack

1. Introduction

In the digital information world, users can easily obtain various kind of services from the distributed networks anywhere and anytime such as online shopping, online bank and pay-TV. Ordinary user authentication protocols are fit to tackle security issues for the single user/server design scenarios. Nowadays, authentication protocols for multi-server architectures play a prime role in the Internet world. The multi-server system contains three participants, including users, servers, and the registration centre. The registration centre as the relied third-party, administers all registered servers and users. A multi-server authentication scheme offers services to be accessed from different servers with one time registration.

Ali and Pal [1] presented a three factor-based authentication scheme in a multi-server environment using ECC. This paper reviews Ali and Pal's protocol [1] and shows its weaknesses, such as session-specific temporary information leakage attack and replay attack. To conquer specific weaknesses, we design an amended protocol.

2. Preliminaries

Table 1 shows symbols and their meaning.

3. Review of Ali and Pal's protocol

Ali and Pal's protocol includes six phases. Beginning from initialization phase, they discussed server enrollment phase, user enrollment phase, login phase, authentication and key agreement phase and password change phase.

3.1. Initialization Phase

To boot up the system, *RC* selects a generator *P* of elliptic curve and chooses a secret key *y* as the system parameter.

3.2. Server Enrollment Phase

The server enrolls itself at the registration center *RC*. Server selects its own identity SID_j

ISIC2021: International Semantic Intelligence Conference, February 25–27, 2021, New Delhi, India

EMAIL: rahulss.rahul1991@gmail.com (A. 1); mkgupta2002@hotmail.com (A. 2); saryusiirohi@gmail.com (A. 3)

ORCID: 0000-0002-2673-2109 (A. 1)

and transfers $\{SID_j\}$ to *RC* through open channel. When the message $\{SID_j\}$ is received by *RC* from the server, then *RC* evaluates $AH = h(SID_j \| y)$. *RC* transmits the information $\{AH\}$ to the server through secure channel.

Table 1

Symbol	Meaning
S_j	j^{th} *server*
RC	*The registration centre*
U_i	*The user*
P	*Generator of elliptic curve*
SID_j	*Server's identity*
UID_i	*User's identity*
B_i	*User's biometric*
$H(\cdot)$	*Bio-hash function*
$h(\cdot)$	*Hash function*
PW_i	*User's password*
J, J_c, J_s, j_{sc}	Random numbers
T_1, T_2, T_3, T_4, T_5	Timestamps
SC	Smart card
SK	Session key
$\|\|$	Concatenation

3.3. User Enrollment Phase

First, user selects his/her identity UID_i, imprints B_i and forwards the message $\{UID_i, H(B_i)\}$ to *RC* through open channel. When a request message is received form the user then *RC* selects a random number J and evaluates $HID_i = Enc_{h(x)}(UID_i \| J)$, $M_i = H(B_i)\cdot P$, $E_i = h(UID_i \| y)\cdot P$, $H_i = M_i + E_i$. Now *RC* inserts all information $\{H_i, HID_i, P, E_k/D_k, h(\cdot), H(\cdot)\}$ into *SC* and forwards $\{SC\}$ to the user. After receiving the message $\{SC\}$ from *RC*, user evaluates $Z_i = h(UID_i \| PW_i \| H(B_i))$ and also inserts Z_i into *SC*.

3.4. Login Phase

User embeds *SC* and enters UID_j^*, PW_i^* and imprints B_i^*. Now *SC* evaluates $Z_i^* = h(UID_i^* \| PW_i^* \| H(B_i^*))$ and checks $Z_i^* =? Z_i$. If the equality does not hold then the connection is disrupted by the user. Otherwise, the user chooses a random number J_c and evaluates $A_1 = J_c\cdot P$, $A_2 = H(B_1)\cdot P + A_1$, $A_3 = h(UID_i \| A_1 \| SID_j \| H_i)$ and user transmits the login message $\{HID_i, H_i, A_2, A_3\}$ to *RC* through open channel.

3.5. Authentication and Key agreement Phase

When the login request message $\{HID_i, H_i, A_2, A_3\}$ is received from the user then *RC* evaluates $Dec_{h(x)}(HID_i) = [UID_i, J]$, $M_i^* = H_i - h(UID_i \| y)\cdot P$, $A_1^* = A_2 - M_i^*$, $A_3^* = h(UID_i \| A_1^* \| SID_j \| Hi)$ and checks $A_3^* =? A_3$. If the equality does not hold then the request is dropped by *RC*. Otherwise, *RC* chooses an arbitrary number J_s and evaluates $HID_i^{new} = Enc_{h(x)}(UID_i \| J_{sc})$, $X_i = Enc_{h(SIDj\|y)}[UID_j \| A_1 \| H(B_i)]$, $A_4 = J_{sc}\cdot P$, $A_5 = A_4 + H(B_i)\cdot P$, $A_6 = h(A_4 \| HID_i^{new} \| SID_j)$ and transmits the message$\{A_5, A_6, HID_i^{new}, X_i\}$ to the server through open channel.

After receiving the message $\{A_5, A_6, HID_i^{new}, X_i\}$ from *RC*, server evaluates $Dec_{h(SIDj\|y)}(X_i) = [UID_i \| A_1 \| H(B_i)]$, $A_4^* = A_5 - H(B_i)\cdot P$, $A_6^* = h(A_4^* \| HID_i^{new} \| UID_i \| SID_j)$ and checks $A_6^* =? A_6$. If the equality does not hold then the connection is disrupted by the server. Otherwise, server chooses J_s and evaluates $A_7 = J_s\cdot P$, $A_8 = A_7 + A_1$, $A_9 = h(HID_i^{new} \| A_7 \| A_4 \| H(B_i)\cdot P)$ and transmits the message $\{HID_i^{new}, A_9, A_8, A_5\}$ to the user.

After receiving the message $\{HID_i^{new}, A_9, A_8, A_5\}$ from the server, user evaluates $A_7^* = A_8 - A_1$, $A_4^* = A_5 - H(B_i)\cdot P$, $A_9^* = h(HID_i^{new} \| A_7^* \| A_4^* \| H(B_i)\cdot P)$ and checks $A_9^* =? A_9$. If the equality does not hold then the connection is disrupted by the user. Otherwise, user evaluates $SK = h(J_c\cdot P \| J_{sc}\cdot P \| J_s\cdot P)$, $N_i = SK\cdot P + h(ID_i \| H(B_i))\cdot P$. Now the user transmits message $\{N_i\}$ to the server through open channel. Note that the user changes HID_i^{new} with HID_i into *SC* to avoid user untraceability attack.

After receiving the message $\{N_i\}$ from the user, the server evaluates $SK^* = h(J_c\cdot P \| J_{sc}\cdot P \| J_s\cdot P)$, $N_i^* = SK^*\cdot P + h(UID_i \| H(B_i)\cdot P)$ and checks $N_i^* =? N_i$. If the equality does not hold then the connection is disrupted by the server. Otherwise, the connection is created.

3.6. Password Change Phase

User can modify his/her password easily without interfering with the server. First, user inserts his/her smartcard into a card reader and enters UID_j^*, PW_i^* and also imprints B_i^*. Now, the smartcard reader evaluates $Z_i^* = h(UID_j^* \| PW_i^* \| H(B_i^*))$ and verifies $V_i^* =? V_i$. If the equality does not hold then the connection is ended. Otherwise, the user selects new password PW_i^{new} and evaluates $Z_i^{new} = h(UID_i \| PW_i^{new} \| H(B_i^*))$. Finally, it interchanges Z_i with Z_i^{new} in memory of the smartcard.

4. Cryptanalysis of Ali and Pal's Protocol

In this phase, we describe the weaknesses of Ali and Pal's protocol.

4.1. Session-Specific Temporary Information Leakage Attack

Ali and Pal's protocol suffers from session-specific temporary information attack as the explanation follows. Suppose, if random number J_c is compromised by any attacker $\mathcal{H}$, then $\mathcal{H}$ can harm the valid user. $\mathcal{H}$ can compute easily other random numbers J_{sc}, J_s by some mechanism. By using the information about $J_c \cdot P$, $J_{sc} \cdot P$, and $J_s \cdot P$, $\mathcal{H}$ can compute $SK = h(J_c \cdot P \| J_{sc} \cdot P \| J_s \cdot P)$. It is a very serious flaw in their protocol.

Suppose temporary random number J_c is leaked somehow, then $\mathcal{H}$ can evaluate session key easily from the known temporary random number in the following steps (also shown in Fig.1).

Step 1: First, $\mathcal{H}$ eavesdrops the login request message $\{HID_i, H_i, A_2, A_3\}$ and also other request messages $\{A_5, A_6, HID_i^{new}, X_i\}$, $\{HID_i^{new}, A_9, A_8, A_5\}$ at the time of authentication and key agreement phase.

Step 2: After this $\mathcal{H}$ computes $A_1 = J_c \cdot P$ and $H(B_i)^* \cdot P = A_1 - A_2$. A_2 is taken from eavesdropped message $\{HID_i, H_i, A_2, A_3\}$.

Step 3: $\mathcal{H}$ also uses eavesdropped message $\{A_5, A_6, HID_i^{new}, X_i\}$ to get A_5, which is sent by registration center to the server. Now $\mathcal{H}$ evaluates $A_4^* = H(B_i)^* \cdot P - A_5 = J_{sc} \cdot P$. Here, J_{sc} is a random number, chosen by *RC*.

Step 4: Now $\mathcal{H}$ wants to compute J_s. A_8 is retrieved from eavesdropped message $\{HID_i^{new}, A_9, A_8, A_5\}$, which is transmitted by the server to the user. After getting the information A_8, $\mathcal{H}$ can compute $A_7^* = A_1^* - A_8 = J_s \cdot P$.

Step 5: After evaluating all the temporary random numbers J_{sc} and J_s from J_c, $\mathcal{H}$ can find out $SK = h(J_c \cdot P \| J_{sc} \cdot P \| J_s \cdot P)$ by using these information. Therefore, Ali and Pal's protocol is suffers from session-specific temporary information attack.

If J_c is compromised and becomes known to the attacker, then the attacker eavesdrops the messages $\{HID_i, H_i, A_2, A_3\}$, $\{A_5, A_6, HID_i^{new}, X_i\}$ and $\{HID_i^{new}, A_9, A_8, A_5\}$ transmitted via insecure channel. Now, the attacker has J_c, A_2, A_5 and A_8.

Attacker computes

$A_1^* = J_c \cdot P$ ……..……..(1)

$H(B_i)^* \cdot P = A_1^* - A_2$

$A_4^* = H(B_i)^* \cdot P - A_5 = J_{sc} \cdot P$ ……...……(2)

$A_7^* = A_1^* - A_8 = J_s \cdot P$ ……..……..……(3)

From (1), (2) and (3), the attacker has the information $J_c \cdot P$, $J_{sc} \cdot P$, $J_s \cdot P$ and he can easily compute the session key, as

$SK = h(J_c \cdot P \| J_{sc} \cdot P \| J_s \cdot P)$

Figure 1. Session-Specific Temporary Information Leakage attack

4.2. Replay attack

Assuming that the login request message $\{HID_i, H_i, A_2, A_3\}$ is eavesdropped by $\mathcal{H}$, which was sent by a legal user to the legal server. After some time, $\mathcal{H}$ transmits the same login request message $\{HID_i, H_i, A_2, A_3\}$ to the legal server. When the login request message is received then the server cannot recognize the freshness of the message and evaluates $Dec_{h(y)}(HID_i) = [UID_i, J]$, $M_i^* = H_i - h(UID_i \| y) \cdot P$, $A_1^* = A_2 - M_i^*$, $A_3^* = h(UID_i \| A_1^* \| SID_j \| Hi)$ and checks $A_3^* =? A_3$. Obviously, the equality will hold and the server accepts the login request of the $\mathcal{H}$. The freshness of the login request message is not check by the server in Ali and Pal's protocol. Therefore, Ali and Pal's protocol is suffering from replay attack.

5. The Proposed Protocol

Our proposed protocol includes six phases: initialization phase, server enrollment phase, user enrollment phase, login phase, authentication and key agreement phase and password change phase.

5.1. Initialization Phase

To boot up the system, *RC* selects a generator P of elliptic curve and chooses a secret key y as the system parameter.

5.2. Server Enrollment Phase

In this phase, the server enrolls itself at the registration center *RC*. The server selects its own identity SID_j and transmits the message $\{SID_j\}$ to *RC* through open channel. When SID_j is picked up from the server then *RC* evaluates $AH = h(SID_j \| y)$ and *RC* transmits the message $\{AH\}$ to the server through open channel as shown in Figure 2.

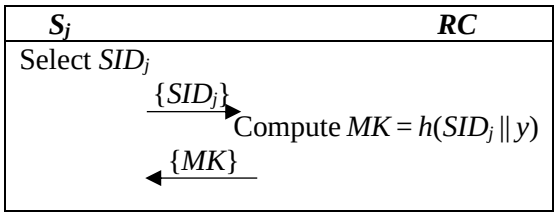

Figure 2. Server Enrollment Phase of the proposed protocol

5.3. User Enrollment Phase

First, user selects his/her identity UID_i, imprints B_i and transmits the message $\{UID_i, H(B_i)\}$ to *RC* through open channel. When the request message is received from the user then *RC* selects a random number J and evaluates $HID_i = Enc_{h(x)}(UID_i \| J)$, $M_i = H(B_i)\cdot P$, $E_i = h(UID_i \| y)\cdot P$, $H_i = M_i + E_i$. Now, *RC* inserts all information $\{H_i, HID_i, P, E_k/D_k, h(\cdot), H(\cdot)\}$ into *SC* and transmits $\{SC\}$ to the user. After receiving $\{SC\}$ from *RC*, user evaluates $Z_i = h(UID_i \| PW_i \| H(B_i))$ and inserts Z_i into *SC* as shown in Figure 3.

5.4. Login Phase

User embeds *SC* into the card reader and enters UID_j^*, PW_i^* and imprints B_i^*. Now *SC* evaluates $Z_i^* = h(UID_i^* \| PW_i^* \| H(B_i^*))$ and checks $Z_i^* =? Z_i$. If the equality does not hold then the connection is stopped. Otherwise, user chooses a random number J_c and evaluates $A_1 = J_c\cdot P$, $A_2 = H(B_1)\cdot P + A_1$, $A_3 = h(UID_i \| A_1 \| SID_j \| Hi)$. Now, user transmits the login request message $\{HID_i, H_i, A_2, A_3, T_1\}$ to *RC* through open channel as shown in Figure 4.

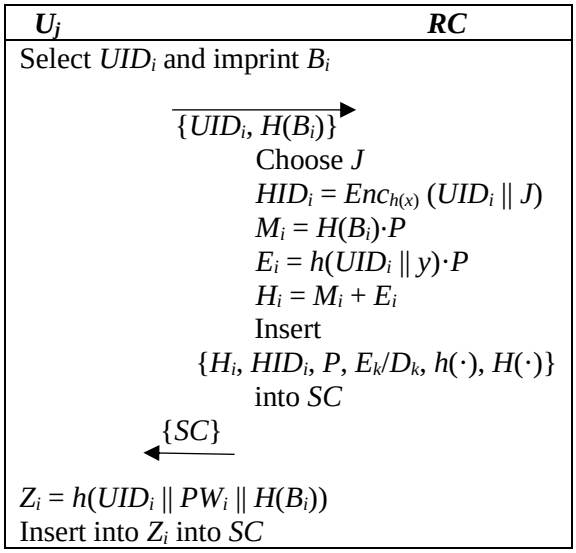

Figure 2. User Enrollment Phase of the proposed protocol

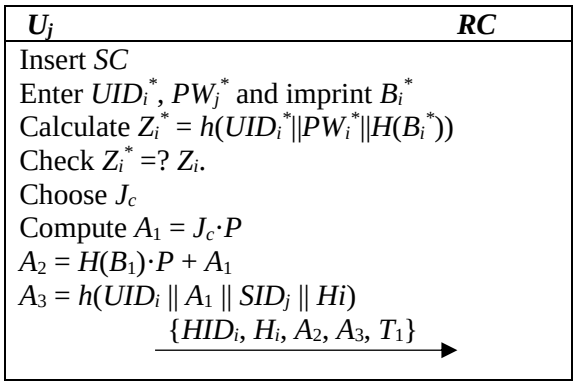

Figure 4. Login Phase of the proposed protocol

5.5. Authentication and Key Agreement Phase

After receiving the login request message $\{HID_i, H_i, A_2, A_3, T_1\}$ from the user, *RC* checks $T_2 - T_1 \le \Delta T$ and evaluates $Dec_{h(x)}(HID_i) = [UID_i, J]$, $M_i^* = H_i - h(UID_i \| y)\cdot P$, $A_1^* = A_2 - M_i^*$, $A_3^* = h(UID_i \| A_1^* \| SID_j \| Hi)$ and checks $A_3^* =? A_3$. If the equality does not hold then the connection is stopped by *RC*. Otherwise, *RC* chooses J_s and evaluates $HID_i^{new} = Enc_{h(x)}(UID_i \| J_{sc})$, $X_i = Enc_{h(SIDj\|y)}[UID_j \| A_1 \| H(B_i)]$, $A_4 = J_{sc}\cdot P$, $A_5 = A_4 + H(B_i)\cdot P$, $A_6 = h(A_4 \| HID_i^{new} \| SID_j)$. *RC* transmits the message $\{A_5, A_6, HID_i^{new}, X_i, T_2\}$ to the server through open channel.

When the message $\{A_5, A_6, HID_i^{new}, X_i, T_2\}$ is picked up from *RC* then server checks $T_3 - T_2 \le$

ΔT and evaluates $Dec_{h(SIDj||y)}(X_i) = [UID_i \| A_1 \| H(B_i)]$, $A_4^* = A_5 – H(B_i)\cdot P$, $A_6^* = h(A_4^* \| HID_i^{new} \| UID_i \| SID_j)$ and checks $A_6^* =? A_6$. If the equality does not hold then the connection is stopped by the server. Otherwise, server chooses J_s and evaluates $A_7 = J_s\cdot P$, $A_8 = A_7 + A_1$, $A_9 = h(HID_i^{new} \| A_7 \| A_4 \| H(B_i)\cdot P)$. Now, server transmits the message $\{HID_i^{new}, A_9, A_8, A_5, T_3\}$ to the user.

After receiving the message $\{HID_i^{new}, A_9, A_8, A_5, T_3\}$ from the server, user checks $T_4 – T_3 \leq \Delta T$ and evaluates $A_7^* = A_8 – A_1$, $A_4^* = A_5 – H(B_i)\cdot P$, $A_9^* = h(HID_i^{new} \| A_7^* \| A_4^* \| H(B_i)\cdot P)$ and checks $A_9^* =? A_9$. If the equality does not hold then the connection is stopped by the user. Otherwise, user evaluates $SK = h(J_c\cdot P \| J_{sc}\cdot P \| J_s\cdot P \| UID_i)$, $N_i = SK\cdot P + h(UID_i \| H(B_i))\cdot P$. Now, user transmits the message $\{N_i, T_4\}$ to the server through open channel. Note that the user replaces HID_i^{new} with HID_i into SC to avoid user untraceability attack.

After receiving the message $\{N_i, T_4\}$ from the user, server checks $T_5 – T_4 \leq \Delta T$ and evaluates $SK^* = h(J_c\cdot P \| J_{sc}\cdot P \| J_s\cdot P \| UID_i)$, $N_i^* = SK^*\cdot P + h(UID_i \| H(B_i)\cdot P)$. After that, the server checks $N_i^* =? N_i$. If the equality does not hold then the connection is stopped by the server. Otherwise, the connection is established between the user and the server as shown in Figure 5.

5.6. Password Change Phase

User can modify his/her password easily without interacting with the server. First, user injects his/her smartcard into a card reader and chooses UID_i^*, PW_i^* and also imprints B_i^*. Now, the card reader evaluates $V_i^* = h(UID_i^* \| PW_i^* \| H(B_i^*))$ and verifies $V_i^* =? V_i$. If it is not satisfied, then the connection is ended. Otherwise, user selects a new Password PW_i^{new} and evaluates $V_i^{new} = h(UID_i \| PW_i^{new} \| H(B_i^*))$. Finally, it displaces V_i with V_i^{new} in the memory of the smartcard.

6. Security Analysis

6.1. Prevents Session-Specific Temporary Information Leakage attack

Suppose the temporary arbitrary number J_c is leaked by the server to the attacker $\mathcal{H}$ then $\mathcal{H}$ will try to evaluate session key from the known temporary random number in the following manner. First, $\mathcal{H}$ computes $A_1^* = J_c\cdot P$, $H(B_i)^*\cdot P = A_1^* – A_2$, $A_4^* = H(B_i)^*\cdot P – A_5 = J_{sc}\cdot P$ and $A_7^* = A_1^* – A_8 = J_s\cdot P$. After evaluating the all temporary random numbers J_{sc} and J_s from J_c, But the session key $SK = h(J_c\cdot P \| J_{sc}\cdot P \| J_s\cdot P \| UID_j)$ could not calculate by $\mathcal{H}$ without knowing UID_j. Therefore, our proposed protocol prevents session- specific temporary information attack.

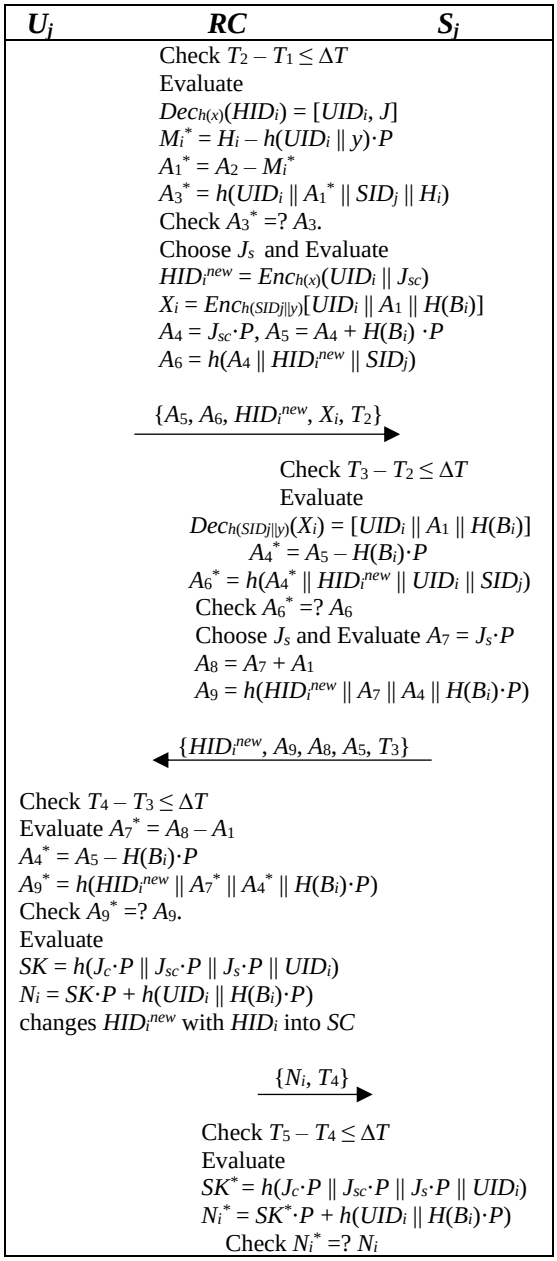

Figure 5. Authentication and Key Agreement Phase

6.2. Replay attack

Assume that the previous login request message $\{HID_i, H_i, A_2, A_3, T_1\}$ is eavesdropped by an adversary, which was sent by a legal user to the legal server. After some time, $\mathscr{H}$ transmits the same login request message $\{HID_i, H_i, A_2, A_3, T_1\}$ to the legal server. When the login request message is received by the server then the server checks the freshness of the timestamp and stops the connection if T_1 is not fresh. Therefore, our proposed protocol prevents the replay attack.

7. Security and Performance Comparison

This section describes the performance and security comparison, along with other related protocols [1]. Some notations are defined as $\boldsymbol{T_H}$ indicates one way hash function, $\boldsymbol{T_{PM}}$ means scalar point multiplication and $\boldsymbol{T_S}$ indicates symmetric decryption/encryption functions, as shown in Table 2.

Table 2 shows the computation cost comparison of the proposed protocol with the protocols in [1]. Ali and Pal's protocol needs to perform total 17 hash functions, 11 scalar multiplication operations and 5 symmetric encryption/decryption functions *i.e.*, $17T_H + 11T_{PM} + 5T_S$. On the other hand, our proposed protocol needs to perform 15 hash functions, 11 scalar multiplication operations and 5 symmetric encryption/decryption functions, *i.e.*, $15T_H + 11T_{PM} + 5T_S$. According to Table 2, our proposed protocol's computation overhead, and Ali and Pal's protocol are identical. The only change is the reduction of 2 hash functions in our proposed protocol. Nevertheless, our protocol is secure against the attacks to which Ali and Pal's protocol is not resistant.

Table 3 compares the proposed protocol's security features with the protocols in [1]. As shown in Table 3, our protocol gives security against replay attack and session-specific temporary information leakage attack. Still, Ali and Pal's protocol doesn't offer protection against the above vulnerabilities. Therefore, our proposed protocol is more effective and secure than the protocol in [1].

Table 2 Comparison of Computation Cost

	Ali and Pal [4]	**Our protocol**
Computation cost of registration phase	$4T_H + 2T_{PM} + 1T_S$	$3T_H + 2T_{PM} + 1T_S$
Computation cost of login and authentication phase	$13T_H + 9T_{PM} + 4T_S$	$12T_H + 9T_{PM} + 4T_S$
Total computation cost	$17T_H + 11T_{PM} + 5T_S$	$15T_H + 11T_{PM} + 5T_S$

Table 3: Comparison of Security Features

Attacks	**Ali and Pal [4]**	**Our protocol**
Prevents session specific temporary information leakage attack	×	✓
Prevents replay attack	×	✓

8. Conclusion

In this paper, we have investigated Ali and Pal's protocol. We have revealed that their protocol is suffering from replay attack and session-specific temporary information leakage attack. To reduce these vulnerabilities, we have proposed an improved protocol. We have used timestamps in our proposed protocol to prevent replay attack. Our proposed protocol is more robust than Pal and Ali's scheme, and there is no extra computation needed in our scheme. We will propose a lightweight scheme for multi-server environment with low computation cost and better security in future work.

9. Acknowledgements

The first author gratefully acknowledges the financial support received from CSIR (India) in the form of Junior Research Fellowship CSIR award no. 09/113(0020)/2018-EMR-I.

References

[1] R. Ali, A. K. Pal, An efficient three factor–based authentication scheme in multi-server environment using ECC, Int. J. Commun. Syst. 31(4) (2017) 1-22.