Probably Secure Lightweight IoT-based Healthcare Services

SK Hafizul Islam, *Senior Member, IEEE*, Krittibus Parai,

***Abstract*—In modern health care system IOT (Internet of Things) based devices are used to take care of the patient's health condition in real-time while in ICU (Intensive Care Unit).Caring health is an important issue in the medical system in a seance of medical report integrity, authenticity and hidden of patient's identities while communications in public channels. Another important issue how to transmit high volume multimedia data securely via insecure channel to resolve this issue we use Interplanetary file system technology which work on distributed environment. In this regards many researchers are proposer their scheme. Unfortunately, none of theme have widely acceptable. So we proposed a smart card based two-factor authentication scheme for health care services ($TASHCS$) based on elliptic curve Cryptography(ECC).**

***Index Terms*—IoT based Medical Care System, anonymous Authentication, Session Key Agreement, Security, ECC**

I. Introduction

Due to the rapid development of Wireless Sensor Network (WSN), several IoT-based applications are possible to implement for versatile field. The IoT-based medical care system is one of them which enable different users (doctors, patients etc.) to communicate between each other for various purposes (healthcare monitoring and services). However, the chance of unauthorized interception to these confidential data still exists due to the nature of wireless media. Hence, the underlying protocol should be designed in such a way so that only authenticate user able to participate in the communication process. Typically, three different agents are involved to deal with a secure user authentication scheme over the wireless environment: User (U_{SER}), Trusted Agent/Server (TA_S), and Local Processing Sensor unit which is Sensor's node around the patient (LPU_i). The entire protocol includes four different phases namely registration phase, log-in and authentication phase, Access Control phase and Password up gradation phase. Initially, User (U_{SER}) registers himself/herself to a Trusted agent/server (TA_S) via a secure channel. The channel is being considered secure as the communication is established in offline mode. The TA_S issue a smart card to that U_{SER} containing all the login information. After that the user uses that smart card for lunching a log-in request to TA_S. TA_s checks the validity of U_{SER} and if the log-in information is correct, then TA_S grants to access sensative data from Local processing Unit(LPU_i) to U_{SER}. The connection between U_{SER} and the LPU_i has been established via public channels after the successful establishment of session key is shared between them. To avoid single point failure, we store all sensitive data in a distributed system by using IPFS(InterPlanetary File System) ,there is no single server all data with out any duplicate, are store in different IPFS node through out the network using uniquely hash code address.

S. H. Islam is with the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India. E-mail: hafi786@gmail.com

K. Parai is with the Siliguri Institute Of Technology (SIT), Siliguri, West Bengal, India. E-mail: krittibas.sit@gmail.com

II. Architecture of patient monitoring system

In our proposed protocol for monitoring patient health condition , three party are involved in communication these are User(U_i), Trusted server(TA) and local processing unit(LPU_j) or smart object which responsible for collect sensitive data(such as Blood presser, Blood sugar, Oxygen level, ECG etc.) from sensor node which placed around the patient body then forward to the TA. To monitor the patient physical condition all user U_i(such as Doctors,Nurses,family member) are responsible. The heart of the system is TA which responsible for register and authenticate both communicating party U_i and LPU_j.
In our proposed scheme two network communication channel have been used (i) Intra-communication channel (ii) Inter-communication channel. The communication network diagram given in fig(1).

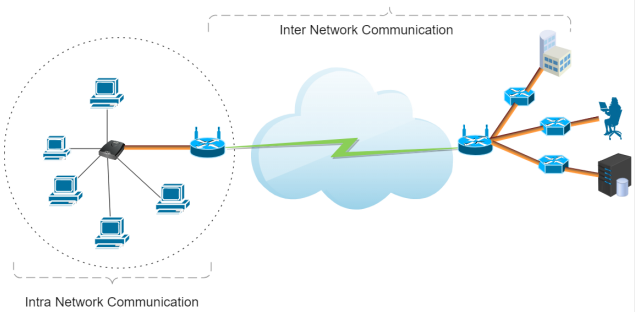

Fig. 1. Intra-Communication Network and Inter-communication Network

- $Intra - CommunicationChannel$: We assume that the distance between sensor node and LPU_j very minimal so that $\mathcal{F}$ can not access or tamper the sensitive data with in real time.That means Intra-communication channel is secure.

- $Inter - communicationChannel$: In this channel communicating objects are placed in various geographical location in the globe. So that $\mathcal{F}$ can launch malicious attack.That means Inter-communication channel is insecure. In our protocol two Inter-network communication channel used (1) Channel between LPU_j and TA (2)

Channel between TA and User(U_i).Architecture diagram given in fig(2).

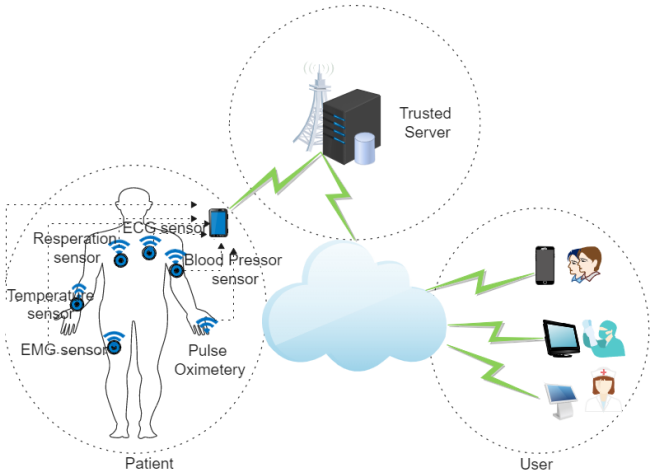

Fig. 2. Architecture diagram

III. Technical Backgrounds

A. Elliptic Curve Cryptography

A promising filed of cryptography is elliptic curve cryptography (ECC) that was firstly conceptualized by the joint effort of Miller [**?**] and Koblitz [**?**]. The security of ECC is based on the discrete logarithm problem. It has been proved that the braking of discrete logarithm problem in elliptic curve, we call it as ECDLP, is computationally hard by a polynomial-time bounded algorithm. Further, a 160-bit key in the elliptic curve has an equal level of security as a 1024-bit key in RSA cryptography [**?**], [**?**]. Moreover, the basic elliptic curve operations, *i.e.*, point addition, point doubling and scalar point multiplication in a elliptic curve group are much faster than those of multiplicative group operations like modular exponentiation. Hence, the protocols based on the elliptic curve are more efficient in terms of communication bandwidth, security, storage space, computation, etc. Therefore, the elliptic curve, along with its operations, is described here.

Suppose $E(\mathbb{Z}_p)$ denotes a non-singular elliptic curve over the field $\mathbb{Z}_p$ in which p is a prime. This elliptic curve $E(\mathbb{Z}_p)$ is defined by the following modular quadratic equation:

$$y^2 \bmod p \quad = \quad (x^3 + ax + b) \bmod p \tag{1}$$

The Eq. (1) is said to be non-singular elliptic curve if $(4a^3 + 27b^2) \bmod p \neq 0$ holds for $a, b \in \mathbb{Z}_p$. The points on $E(\mathbb{Z}_p)$ form an commutative group over addition, $G = \{(x, y) : x, y \in \mathbb{Z}_p \text{ and } (x, y) \in E(\mathbb{Z}_p)\} \cup \{O\}$. Here, *"O"* is *"point at infinity"*. $E(\mathbb{Z}_p)$ over $\mathbb{Z}_p$ has p points on it. The Hasse theorem assures that the number of points, which is denoted as $\#E$, on $E(\mathbb{Z}_p)$ satisfies the following inequality: $p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}$. The following operation are performed on the points of $E(\mathbb{Z}_p)$.

Definition 1 (Point addition): Let two points on the curve (1) be $X = (x_1, y_1)$ and $Y = (x_2, y_2)$. The addition of these points is another point $Z = (x_3, y_3)$ on the same curve *i.e.*, $Z = X + Y$. The point Z is image of point $-Z$ with respect to *x-axis* and $-Z$ is an intersection point of line joining points X and Y on curve (1). The point $Z = (x_3, y_3)$ is analytically computed as $x_3 = (\lambda^2 - x_1 - x_2) \pmod p$, and $y_3 = (\lambda(x_1 - x_3) - y_1) \pmod p$, where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{(mod p), if } X \neq Y \\ \dfrac{3x_1^2 + a}{2y_1} & \text{(mod p), if } X = Y \end{cases}$$

To make the above definition more clear, we have presented a diagrammatic representation in Fig. 3.

Definition 2 (Point doubling): The point doubling is an operation on the curve (1) where a point X is added to itself to find another point Y, *i.e.*, $Y = 2X = X + X$. For this operation, the tangent to the point X is drawn, which cuts the curve (1) at $-Y$.
This definition is further represented in Fig. 4.

Definition 3 (Point inversion and subtraction): If X is a point on (1), then the inverse of X will be $-X$, which is an image of X with respect to *x-axis*. The point subtraction is adding a point with its inverse *i.e.*, $X + (-X) = X - X = O$. It means that the line joining X and $-X$ meets the curve (1) at O.
This definition is further represented in Fig. 5.

Definition 4 (Scalar point multiplication): The scalar point multiplication of a point X with a scalar t is defined as t times addition of X to itself, *i.e.*, $tX = X + X + \cdots + X$ (t times).

Definition 5 (Order of a point): The order of a point X is a smallest positive integer n for which $nX = O$.

B. Hardness assumption

Definition 6 (Negligible function): Given integer k as security parameter, a negligible function, denoted by $\varepsilon(k)$ is defined as $\varepsilon(k) \leq \frac{1}{k^l}$ if $\forall\ l > 0$, $\exists\ k_0$ such that $\forall\ k \geq k_0$ [**?**].

Definition 7 (Elliptic curve discrete logarithm (ECDL) assumption): Given two points $P, Q \in E(\mathbb{Z}_p)$, finding the value of $x \in \mathbb{Z}_p^*$ from given $Q = xP$ is hard. The success probability of any probabilistic polynomial-time (PPT) forger $\mathcal{F}$ to solve the ECDL problem is defined as follows: $Adv_{\mathcal{F}}^{ECDL} = \mathbf{Pr}[\mathcal{F}(P, Q = xP) = x : P, Q \in E(\mathbb{Z}_p); x \in \mathbb{Z}_p^*]$ [**?**]. For any PPT forger $\mathcal{F}$, inequality $Adv_{\mathcal{F}}^{ECDL} \leq \epsilon$ holds [**?**].

Definition 8 (Computational Diffie-Hellman (CDH) assumption): Given a tuple (P, xP, yP) for $x, y \in \mathbb{Z}_p^*$ and $X \in E(\mathbb{Z}_p)$, computing the value of xyP is hard. The success probability of any PPT forger $\mathcal{F}$ to solve the CDH problem is defined as follows: $Adv_{\mathcal{F}}^{CDH} = \mathbf{Pr}[\mathcal{F}(P, xP, yP) = xyP : P \in E(\mathbb{Z}_p); x, y \in \mathbb{Z}_p^*]$ [**?**]. For any PPT forger $\mathcal{F}$, inequality $Adv_{\mathcal{F}}^{CDH} \leq \epsilon$ holds [**?**].

IV. Initial System Requirement

To built a light-weight protocol, we use one-way hash function [**?**], and encryption/decryption function. To achieve better security, we use elliptic curve [**?**], [**?**]. A long-term secret key of the Trusted Server (TA) is s and a secret key X_{tj} is shared between a TA and a local processing sensor unit LPU_j, where $1 \leq j \leq n$, n is the number of LPU_j.

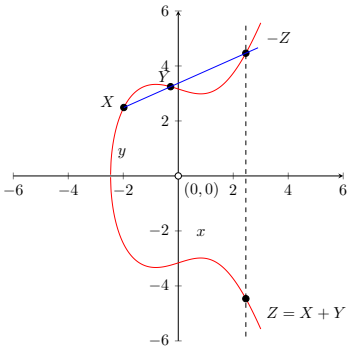

Fig. 3. Point addition operation

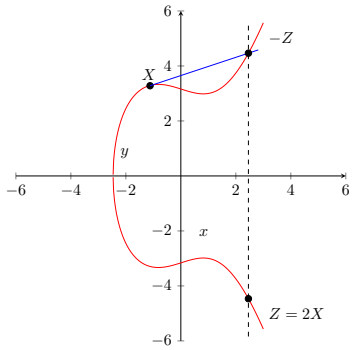

Fig. 4. Point doubling operation

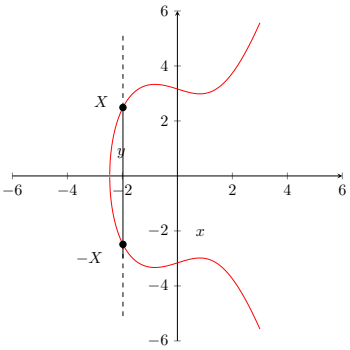

Fig. 5. Point inversion and subtraction operations

TABLE I
NOTATIONS

Notation	Description
p	A large prime number of k-bit ($p \geq 2^k$)
$E(\mathbb{Z}_p)$	Elliptic curve of order p
P	Generator of $E(\mathbb{Z}_p)$
$\mathbb{Z}_p^*$	$\{x : 0 < x < p : \textbf{gcd}(x, p) = 1\}$
TA	Trusted authority
s	Secret key of TA, $s \in \mathbb{Z}_p^*$
ID_t	Identity of TA
U_i	i-th User
ID_i/PW_i	Identity/Password of U_i
LPU_j	j-th Local processing unit
ID_j	Identity of LPU_j
X_{tj}	Secret share key of between TA and LPU_j
SK_{ij}	Session key computed by U_i and LPU_j
a, b, k_i, k_j, s_t, r_i	Numbers selected from $\mathbb{Z}_p^*$ uniformly at random
$a \xleftarrow{r} X$	Choose an element a from X uniformly at random
$H(\cdot)$	Collision resistance one-way hash function
$\|, \oplus$	Concatenation/XOR operation
$\mathcal{F}$	Adversary (an outsider, malicious user, etc.)

V. PROPOSED SCHEME

This section demonstrates our protocol that includes the following phases: (i) User registration, (ii) Local processing unit registration, (iii) Login and authentication, and (iv) Password update. In this protocol, the following entities are involved: (i) a user U_i, which is an instance of a doctor, nurse, family member of a patient, etc., (ii) a Trusted server TA, and (iii) a local processing unit LPU_j. The main role of U_i is to monitor continuously or access health information from remote place. The role of the TA is to validate all user and local processing units. The usage of a LPU_j is to collect data continuously from sensor node embedded in U_i's body, process it, and transmit process information securely as per request from U_i. The notations used in this paper are given in Table I.

A. Setup phase

In this phase, the TA setup all the global parameters required to run the proposed protocol. Assume that the security parameter 1^k is the input of this algorithm, TA creates the list of system parameters Δ as follows:

(1). TA chooses a k bit prime number p, an elliptic curve $E(\mathbb{Z}_p)$ over finite field $\mathbb{Z}_p$, and a generator P of $E(\mathbb{Z}_p)$.

(2). TA selects $s \in \mathbb{Z}_p^*$ as secret key.

(3). TA selects a collision-resistance hash function $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^l$.

(4). Finally, TA publishes system parameter $\Delta = \{E(\mathbb{Z}_p), p, P, H(\cdot)\}$. The Δ is public to all the users associated with the system.

TA Choose a long term secret key s for User U_i and choose another secret key for X_{tj} for Local processing unit.

B. User registration phase

In this phase, a user U_i registered himself/herself with TA through a secure channel (in offline mode). The detailed steps of this phase are stated as follows and further explained in Fig. 6.

(1). U_i choose an unique identity ID_i, a number $r_i \xleftarrow{r} \mathbb{Z}_p^*$, and computes $R_i = H(ID_i \| r_i)P$. U_i sends the registration request message $\{R_i\}$ to TA through a secure channel.

(2). After receiving $\{R_i\}$, TA issue a fresh local processing unit LPU_j and a new smartcard SC_i for U_i. TA computes $V_t = H(R_i \| s)$, and $K_t = sP + R_i$. We assume that the ID_j is the unique identity of LPU_j. Afterwards, TA stores the parameters $\{V_t, K_t, ID_j\}$ in to SC_i and $\{R_i\}$ and into LPU_j. Next, TA sends the personalized smartcard SC_i to U_i over a secure channel.

(3). After receiving the SC_i, U_i choose a fresh password PW_i from a small password dictionary $\mathcal{D}$, and computes $PWD_i = H(ID_i \| PW_i)$ and $K_t' = K_t - R_i$. U_i further computes $R_{i+1} = \textbf{Enc}_{PWD_i}(r_i \| K_t')$, $V_{t+1} = H(V_t \| r_i \| K_t{}')$, and replaces K_t by the value V_{t+1} into the memory of SC_i. Finally, SC_i contains the parameters $\{V_t, V_{t+1}, ID_j, R_{i+1}, H(\cdot)\}$.

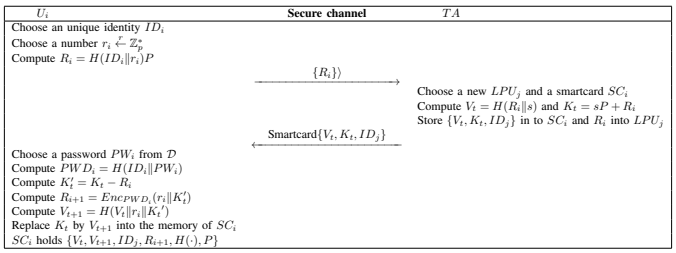

Fig. 6. User registration

C. Local processing unit registration phase

In this phase, a new local processing unit, say LPU_j, is registered to TA through a secure channel. We assume that TA selects a shared secret key X_{tj} and embeds it into the memory of LPU_j. It is also to be noted that LPU_j is placed near the body of the patient to monitor the health conditions. The following steps are performed to complete this phase.

(1). TA choose a number $b \xleftarrow{r} \mathbb{Z}_p^*$, and compute $IDR_t = H(R_i \| b)$ and $IDP_t = X_{tj} \oplus IDR_t$. TA writes $\{ID_t, IDP_t\}$ into the memory of LPU_j.

(2). Next, LPU_j compute $IDR_t = IDP_t \oplus X_{tj}$, and stores it into the memory. Finally, LPU_j hold the information $\{IDR_t, IDP_t, ID_t\}$

TA	**Secure channel**	LPU_j
Choose a number $b \xleftarrow{r} \mathbb{Z}_p^*$		
Compute $IDR_t = H(R_i \| b)$		
Compute $IDP_t = X_{tj} \oplus IDR_t$		
	$\{ID_t, IDP_t\} \longrightarrow$	
		Compute $IDR_t = IDP_t \oplus X_{tj}$
		Store $\{IDR_t, IDP_t, ID_t\}$ into the memory

Fig. 7. Local processing unit registration

D. Login and authentication phase

In this phase, U_i is allowed to access sensitive medical information from LPU_j via TA.

(1). U_i inserts his/her SC_i into the smartcard-reader, and input the login credentials, ID_i and PW_i. At first, SC_i of U_i computes $PWD_i = H(ID_i \| PW_i)$ and $(r_i \| K_t') = \mathbf{Dec}_{PWD_i}(R_{i+1})$. SC_i further computes $R_i = H(ID_i \| r_i)P$, $V_{t+1}' = H(V_t \| r_i \| K_t')$, and verifies whether $V_{t+1}' \stackrel{?}{=} V_{t+1}$. If it is false, SC_i aborts the process; otherwise, chooses a number $a \xleftarrow{r} \mathbb{Z}_p^*$, and computes $A_{i+1} = aP$, $K_{it}' = aK_t' = asP$ and $V_{t+2} = \{(R_i \| ID_j \| K_{it})\} \oplus K_{it}'$. SC_i sends $\{V_{t+2}, A_{i+1}\}$ to TA via an insecure channel.

(2). After receiving the message $\{V_{t+2}, A_{i+1}\}$, TA computes $K_{it}' = sA_{i+1} = saP$, and $\{(R_i \| ID_j \| K_{it})\} = V_{t+2} \oplus K_{it}$. Next, TA verifies whether $K_{it}' \stackrel{?}{=} K_{it}$. If the condition is false, TA terminate the session immediately. If it is true, TA accept the request from U_i, and retrieves R_i from the database. Thereafter, TA choose $s_t \xleftarrow{r} \mathbb{Z}_p^*$, and computes $S = s_t P$, $K_{tl} = X_{tj} S = X_{tj} s_t P$, $T_{sk} = H(K_{it} \| K_{tl})$, $V_{t+3} = (T_{sk} \| K_{it}) \oplus K_{tl}$ and $V_{t+4} = \mathbf{Enc}_{K_{it}}\{T_{sk} \| K_{tl}\}$. TA sends $\{V_{t+3}, S\}$ to LPU_j, and $\{V_{t+4}\}$ to U_i, respectively over a public channel.

(3). On receiving $\{V_{t+3}, S\}$ from TA, LPU_j computes $K_{tl} = X_{tj} S = X_{tj} s_t P$, and extracts T_{sk} as (T_{sk}, K_{it})=$V_{t+3} \oplus K_{tl}$. LPU_j verifies whether $T_{sk} \stackrel{?}{=} H(K_{it} \| K_{tl})$ holds. If the condition is false, LPU_j aborts the session. else, proceeds further.

(4). On receipt of V_{t+4}, from TA, SC_i decrypts it and obtain $\{T_{sk}, K_{tl}\}$ as $T_{sk} \| K_{tl} = \mathbf{Dec}_{K_{it}}\{V_{t+4}\}$. U_i then verifies whether $T_{sk} \stackrel{?}{=} H(K_{it} \| K_{tl})$ holds. If it is false, U_i aborts the session; otherwise, chooses $k_i \xleftarrow{r} \mathbb{Z}_p^*$, and computes $V_{t+5} = \mathbf{Enc}_{T_{sk}}\{ID_i \| k_i \| ID_j\}$. SC_i sends $\{V_{t+5}\}$ to LPU_j over a public channel.

(5). On receiving $\{V_{t+5}\}$ from SC_i, LPU_j decrypts it and obtains $\{ID_i \| k_i \| ID_j\} = \mathbf{Dec}_{T_{sk}}\{V_{t+5}\}$. LPU_j. Next, LPU_j verifies whether $ID_j' \stackrel{?}{=} ID_j$ holds. If the condition is false, LPU_j aborts the session; otherwise, chooses a number $k_j \xleftarrow{r} Z_p^*$, computes $SK = H(k_i \| k_j)$, $V_{t+6} = H(SK \| k_i \| k_j)$ and $V_{t+7} = \mathbf{Enc}_{T_{sk}}\{k_j \| V_{t+6}\}$. Thereafter, LPU_j sends $\{V_{t+7}\}$ to U_i over a public channel.

(6). After receiving $\{V_{t+7}\}$, U_i decrypts V_{t+7} and obtains $\{k_j, V_{t+6}\} = \mathbf{Dec}_{T_{sk}}\{V_{t+7}\}$, computes $SK = H(k_i \| k_j)$, and verifies whether $V_{t+6} \stackrel{?}{=} H(SK \| k_i \| k_j)$ holds. If the condition is false, U_i aborts the session; otherwise, accepts SK as correct common session key.

E. Password update phase

Frequently changes of password is the vital role for achieving a high-level security system; hence we provide a technique so that a genuine user can change their password without the involvement of the TA. The following steps are involved to changes password.

(1). Initially, U_i insert his/her SC_i into the smartcard-reader, and provide his/her credential, i.e, ID_i and PW_i. Next, SC_i computes $PWD_i = H(ID_i \| PW_i)$, $(r_i \| K_t') = \mathbf{Dec}_{PWD_i}(R_{i+1})$, $R_i = H(ID_i \| r_i)P$, and $V_{t+1}' = H(V_t \| r_i \| K_t')$. Thereafter, SC_i verifies whether $V_{t+1}' \stackrel{?}{=} V_{t+1}$ hlds. If it is false, SC_i aborts the process; otherwise, asks U_i to insert a new password.

(2). U_i inserts a new password PW_{new}, SC_i computes $PWD_i^* = H(ID_i \| PW_{new})$, $R_{i+1}^* = \mathbf{Enc}_{PWD_i^*}\{r_i \| K_T{}'\}$. Finally, SC_i replaces old tuple $\{V_{t+1}, V_t, ID_j, R_{i+1}, H(\cdot)\}$ by a new tuple $\{V_{t+1}, V_t, ID_j, R_{i+1}^*, H(\cdot)\}$ into the memory.

VI. Formal Verification

We denoted our protocol as $\mathcal{P}$. In this section, we have done the formal verification of $\mathcal{P}$ under active and passive attack in the Random Oracle Model (ROM). This model is configured based on the Oracle instances of U_i, LPU_j, and TA in which a forger $\mathcal{F}$ can run various queries to break the security of $\mathcal{P}$.

A. Formal Model

Three participants U_i, LPU_j, and TA are involved during the execution of $\mathcal{P}$. Let π_U^k, π_L^k and π_T^k denotes k^{th} instance of U_i, LPU_j and TA. We also denote the oracle $\Pi_{U,L,T}^k$ is the kth instance of U_i involved with its partner LPU_j via TA in a session k. The session identifier $sid_{U,L,T}^k$ is considered as the concatenation of the messages exchanged among U_i, LPU_j and TA. The partner identifier $pid_{U,L,T}^k$ of $\pi_{U,L,T}^k$ is a set containing the identity U_i, LPU_j and TA for a session k.

Definition 9 (Freshness): An oracle $\pi_{U,L,T}^k$ is *fresh* if (i) the instances π_U^k, π_L^k and π_T^k enter into an *accepted* state, (ii) No **Reveal**(π_U^k, π_L^k, π_T^k) query is executed, (iii) No **Send**(π_U^k, m) or **Send**(π_L^k, m) query is executed, and (iv) only $\langle$**Corrupt**(U_i), **Corrupt**(TA)$\rangle$ or $\langle$**Corrupt**(LPU_j), **Corrupt**(TA)$\rangle$ can be executed.

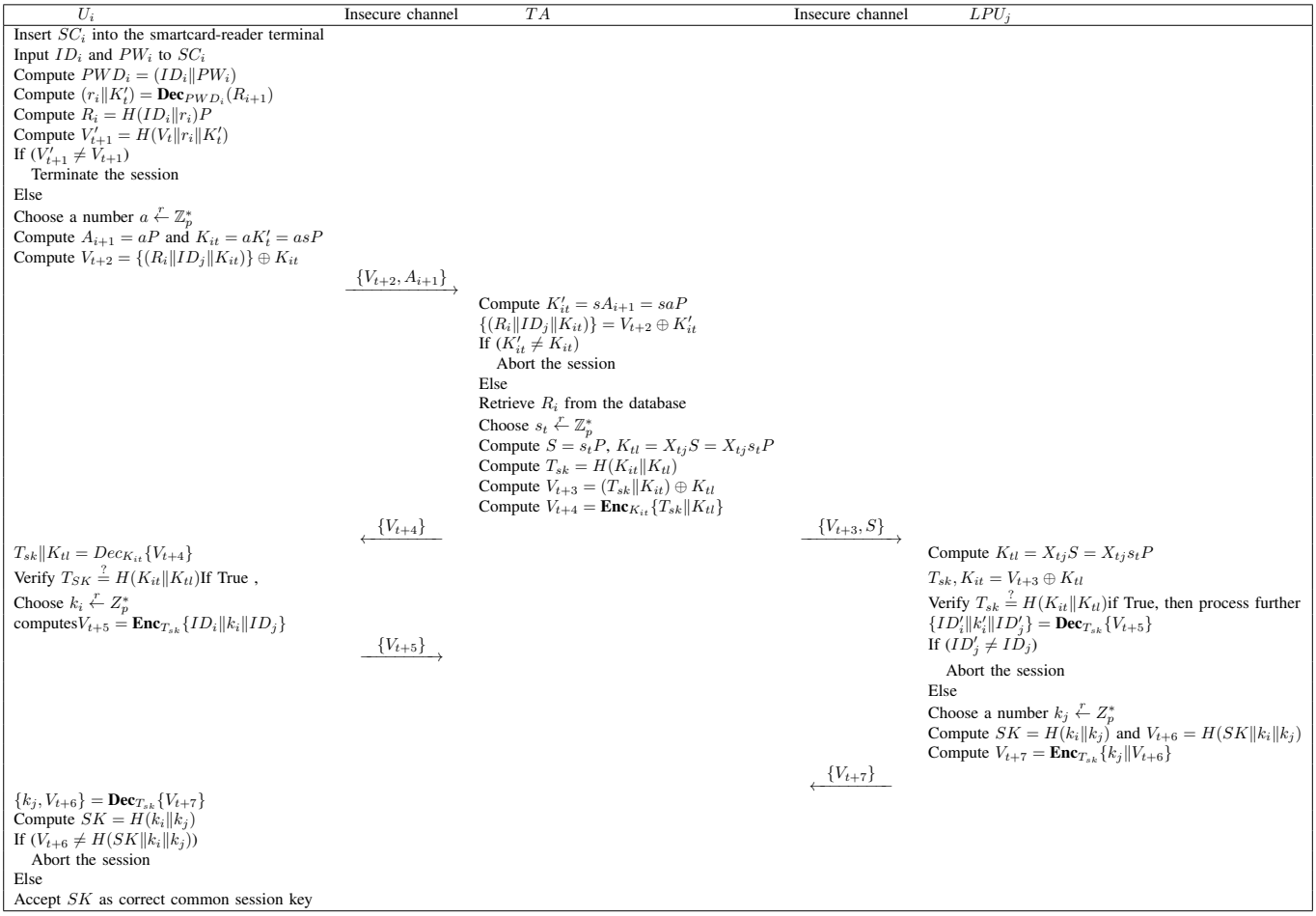

Fig. 8. Login and authentication with session key agreement

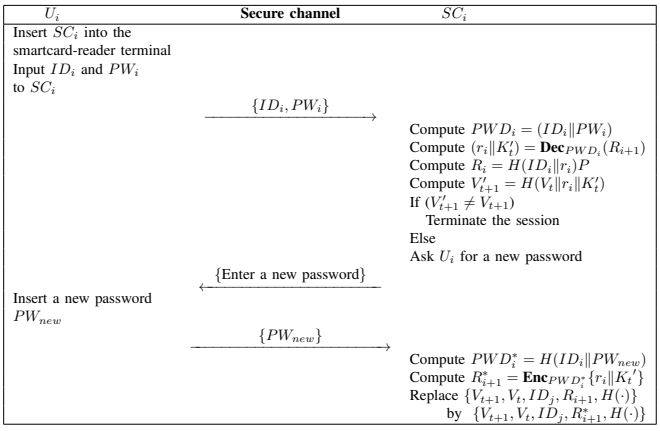

Fig. 9. Password Update Phase

Definition 10 (Accepted State): An instance π_u^i (or π_l^j or π_t^k) is in the accepted state if it has successfully received the last expected protocol message. The order of all communicated messages by π_u^i in a session are concatenated and compute as a session identifier sid of π_u^i for the current session.

Definition 11 (Partnering): Two instances π_U^i and π_L^j are said to be partners if

(i) both π_U^i and π_L^j are in accepted state.

(ii) both π_U^i and π_L^j share the same session identifier.

(iii) both π_U^i and π_L^j mutually authenticate each other.

B. Adversary Model

In our analysis, we consider the following assumptions [**?**], [**?**], [**?**] Give proper citation for each point.

(1) A forger $\mathcal{F}$ can not access any information transmitted via a secure channel, but it possible in any public medium.

(2) U_i may lost his/her SC_i and $\mathcal{F}$ got it. Then $\mathcal{F}$ can access all information hold by SC_i.

(3) $\mathcal{F}$ has complete control over the public communication channel, i.e., he/she can obtain the messages from the public medium and modify them.

To breach the security of $\mathcal{P}$, $\mathcal{F}$ will play a challenge-response game with a challenger $\mathcal{C}$. In this game, $\mathcal{C}$ is allowed to simulate the proposed protocol, and $\mathcal{F}$ is authorized to make any sequence of the queries defined below.

- $Execute(\pi_{U,L,T}^k)$**:** $\mathcal{F}$ can ask $\mathcal{C}$ to simulate this query to obtain the message transmitted among π_U^k, π_L^k and π_T^k to perform an eavesdropping attack.
- $Send(\pi_P^k, m)$**:** $\mathcal{F}$ can ask $\mathcal{C}$ to excuse this query to lunch an active attack against π_P^k. Using this queries, $\mathcal{F}$ can to send a modified message m to π_P^k, and receipt a reply according to the definition of $\mathcal{P}$. Here P signifies a participant (U_i or LPU_j or TA).

- $Hash(x)$**:** $\mathcal{F}$ can ask $\mathcal{C}$ to execute a Hash query for the input x. Note that $\mathcal{C}$ will maintain a list that includes tuples like (x, y), where y is the output of this query. To return output for x, $\mathcal{C}$ searches the list a return y to $\mathcal{F}$ if a tuple (x, y) is found there; otherwise, choose a value y randomly and returns it as an answer, and inserts the new tuple (x, y) into the list. This query implicitly includes Send a query.
- $Reveal(\pi_U^k)$**:** With this query $\mathcal{F}$ get access session key SK generated by instance π_U^k in a session. This query helps to $\mathcal{F}$ perform the known-key attack against $\mathcal{P}$.
- $CorruptSC(\pi_U^k)$**:** This query helps to $\mathcal{F}$ to corrupt a participant π_U^i, i.e., $\mathcal{F}$ can obtain secret information of π_U^k. The objective of $\mathcal{F}$ is to execute this attack to perform a password guessing attack through a lost/stolen smartcard. If $\mathcal{F}$ asks $\mathcal{C}$ to simulate a $CorruptSC(\pi_U^k)$ query, $\mathcal{C}$ returns the the information stored in the smartcard of π_U^k. $\mathcal{F}$ will use the information obtained from the smartcard to guess the password. It is a passive attack.
- $Test(\pi_{U,L,T}^k)$**:** This query is used to modeled the semantic security of the session key SK. First, $\mathcal{F}$ asks $\mathcal{C}$ to simulate a $Reveal(\pi)$ query to get tSK. If SK has not been established yet, then π_u^i return a null value. Otherwise, $\mathcal{C}$ tosses an unbiased coin c, and returns current SK if $c = 1$; otherwise, returns a bit-string of length l as session key. In this case, $\mathcal{F}$ is allowed to simulate this query only once per session.

Definition 12 (Probability of success): Let **Succ**($\mathcal{F}$) be the event that $\mathcal{F}$ makes a single **Test**($\pi_{U,L,T}^k$) query to a fresh session k that has ended successfully, and eventually returns a guess bit c', where $c' = c$ and c was selected in the **Test**($\pi_{U,L,T}^k$) query. The advantage, which is the probability of success of $\mathcal{F}$ within the polynomial time bound t in violating the semantic security of the session key of $\mathcal{P}$, is defined as $\mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) = |\mathbf{Pr}[\mathbf{Succ}(\mathcal{F})] - \frac{1}{2}|$.

Definition 13 (Semantic security): A protocol $\mathcal{P}$ is semantically secure if: (i) in the presence of $\mathcal{F}$, π_U^k and its partner π_L^k are in *accepted* state and computed a common session key with the help of the instance π_T^k; and (ii) $\mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t)$ is negligible.

Theorem 1: Let q_{send}, q_{hash}, and $|\mathcal{D}|$ denotes the number of **Send**, and **Hash** queries, and the cardinality of uniformly distributed password dictionary $\mathcal{D}$. Therefore, we have:

$$\mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) \leq \frac{q_{hash}^2}{2l} + \frac{q_{send}}{|\mathcal{D}|} + Adv_{\mathcal{P}}^{CDH}(t)$$

Proof 1: In this proof, we define the a sequence of experiments $\mathbf{Exp}_0$, $\mathbf{Exp}_1$, $\cdots$, $\mathbf{Exp}_4$. These experiments are nothing but a challenge-response game played by $\mathcal{F}$ and a simulator $\mathcal{C}$. Here, $\mathcal{C}$ will execute the proposed protocol $\mathcal{P}$ based on the different queries (**Execute**, **Send**, **Hash**, **Reveal**, **CorruptSC**, and **Test**) submitted by $\mathcal{F}$, and returns the output to $\mathcal{F}$ every time. With these outputs obtained from $\mathcal{C}$, $\mathcal{F}$ will try to forge $\mathcal{P}$. In this proof, we will prove that if $\mathcal{F}$ can forge the session key of $\mathcal{P}$, then $\mathcal{F}$ can also solve an instance CDH problem.

The simulation of $\mathbf{Exp}_0$ is equivalent to perform the forgery of the semantic security of the session key of $\mathcal{P}$ by brute-force strategy. For $\mathbf{Exp}_l$, we define an event ψ_l $(0 \leq l \leq 4)$ through which $\mathcal{F}$ tries to violate the semantic security of the session key computed during the simulation of $\mathcal{P}$. Suppose that the event **E** independent of ψ_l, may occur during the simulation $\mathcal{P}$ protocol by $\mathcal{F}$ such that **E** is detectable by $\mathcal{C}$. Noted that $\mathbf{Exp}_l$ and $\mathbf{Exp}_{l+1}$ are indistinguishable unless E occurs. Therefore

$$|\mathbf{Pr}[\psi_{i+1}] - \mathbf{Pr}[\psi_i]| \quad \leq \quad Pr[\mathbf{E}] \tag{2}$$

- $\mathbf{Exp}_0$: This is the real attack against $\mathcal{P}$ performed by $\mathcal{F}$.

$$\mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) = |\mathbf{Pr}[\psi_0] - \frac{1}{2}| \tag{3}$$

- $\mathbf{Exp}_1$: In this game, $\mathcal{F}$ ask $\mathcal{C}$ to simulate the oracle query **Execute**(π_U^k, π_L^k, π_T^k), and obtains the message $\{V_{t+2}$, V_{t+3}, V_{t+4}, V_{t+5}, $V_{t+7}\}$ from the output return by $\mathcal{A}$. ***This scenario is simulating the eavesdropping attack***. Finally, $\mathcal{F}$ issue a **Test**($\pi_{U,L,T}^k$) query to distinguish an actual session key SK from a random string. Note that SK is computed as $SK = H(k_i \| k_j)$ that involves two unknown nonces k_i and k_j. Therefore, we have:

$$|\mathbf{Pr}[\psi_1] = \mathbf{Pr}[\psi_0]| \tag{4}$$

- $\mathbf{Exp}_2$: The execution of this experiment is similar to $\mathbf{Exp}_1$, in addition $\mathcal{F}$ can issue $Hash(m)$ and $Send(\pi_P^k, m)$ queries to any participant by sending a fabricated message. $\mathcal{F}$ repeatedly quires **Hash** oracle to find the collisions. Since all the public messages $\{V_{t+2}, V_{t+3}, V_{t+4}, V_{t+5}, V_{t+7}\}$ is associated with random nonces which can not be obtained by simulating any of the queries. It means that the randomness in the messages ensure no collision in the output of the **Hash** oracle. Therefore, according to the Birthday paradox we get:

$$|\mathbf{Pr}[\psi_2] - \mathbf{Pr}[\psi_1]| \leq \frac{q_{hash}^2}{2l} \tag{5}$$

  So $\mathcal{F}$ replies to this message to obtain future or past session key $SK = H(k_i \| k_j)$, but he fails to reveal this due to negligible probability of occurring collision of $H(\cdot)$ function. ***By this experiment, $\mathcal{F}$ is executing the known-key attack***.
- $\mathbf{Exp}_3$: This game simulate the forgery attack and forward and backward secrecy of the session key. Hence, $\mathcal{F}$ will execute **Execute**($\pi_{U,L,T}^k$), **Hash**(m), **Send**(π_P^k, m), and **Revel**(π_U^k) queries to compute the session key $SK = H(K_i \| K_j)$ from publicly available messages $\{V_{t+2}$, V_{t+3}, V_{t+4}, V_{t+5}, $V_{t+7}\}$. Now, $\mathcal{F}$ may try to compute $T_{sk} = H(K_{it} \| K_{tl})$, where $K_{it} = saP$ and $K_{tl} = X_{tl} s_t P$. Therefore, $\mathcal{F}$ needs to know the secrets of U_i and LPU_j. To win this experiment, $\mathcal{F}$ need to solve CDH problem to get K_{it} (or K_{tl}) in polynomial time (t). Therefore, the advantage of $\mathcal{F}$ is $Adv_{\mathcal{F}}^{CDH}(t')$, where $a, s_i, k_i, k_j \xleftarrow{r} \mathbb{Z}_p^*$, $(k_j \| V_{t+6}) = Dec_{T_{sk}}(V_{t+7})$, $V_{t+6} = H(SK \| k_i \| k_j)$, and $SK = H(k_i \| k_j)$. Though, $\mathcal{F}$ can send a forged/fabricate message to any oracle instant($\pi_{U,L,T}^k$), however, he/she cannot compute the actual session key SK due to the unknown values a, s_i, k_i, k_j. From this experiment, we can write:

$$|\mathbf{Pr}[\psi_3] - \mathbf{Pr}[\psi_2]| \leq Adv_{\mathcal{F}}^{CDH}(t') \tag{6}$$

- **Exp**$_4$: This experiment simulates the password guessing attack through stolen/lost smartcard. To perform this attack, $\mathcal{F}$ will execute **CorruptSC**(π_U^k) oracle query, and obtains the information $\{V_t,\ V_{t+1},\ R_{i+1},\ ID_j\}$ from the smartcard of U_i, where $V_t = H(R_i \| s)$, $V_{t+1} = H(V_t \| r_i \| K_t{}')$, and $R_{i+1} = \mathbf{Enc}_{PWD_i}(r_i \| K_t')$. To obtain PW_i, $\mathcal{F}$ have to guess it by selecting a password randomly from dictionary $\mathcal{D}$ with limited number of attempted using $Send(\pi_P^k, m)$ query to the instance π_U^k.

$$|\mathbf{Pr}[\psi_4] - \mathbf{Pr}[\psi_3]| \leq \frac{q_{send}}{|\mathcal{D}|} \tag{7}$$

Note that $\mathcal{F}$ has played all the experiment and execute **Test**$(\pi_{U,L,T}^k)$ query. Since, $\mathcal{F}$ have no knowledge about bit c by tossing unbiased coin, as session key is generate freshly each new session between U_i and LPU_j. Therefore, we have:

$$\mathbf{Pr}[\psi_4] = \frac{1}{2} \tag{8}$$

Therefore, from the Equations (3) and (4), we get

$$\begin{aligned} \mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) &= |\mathbf{Pr}[\psi_0] - \frac{1}{2}| \\ &= |\mathbf{Pr}[\psi_1] - \frac{1}{2}| \\ &= |\mathbf{Pr}[\psi_1] - \mathbf{Pr}[\psi_4] + \mathbf{Pr}[\psi_4] - \frac{1}{2}| \\ &= |\mathbf{Pr}[\psi_1] - \mathbf{Pr}[\psi_4]| \end{aligned} \tag{9}$$

Applying the triangle inequality and form equations (5) to (9) we get

$$\begin{aligned} \mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) &= |\mathbf{Pr}[\psi_1] - \mathbf{Pr}[\psi_4]| \\ &= |\sum_{i=1}^{4} \mathbf{Pr}[\psi_i] - [\psi_{i+1}]| \\ &\leq \sum_{i=1}^{4} |\mathbf{Pr}[\psi_i] - [\psi_{i+1}]| \\ &= \frac{q_{hash}^2}{2l} + \frac{q_{send}}{|\mathcal{D}|} + Adv_{\mathcal{P}}^{CDH}(t) \end{aligned}$$

Therefore, we obtained,

$$\mathbf{Adv}_{\mathcal{F}}^{\mathcal{P}}(t) \leq \frac{q_{hash}^2}{2l} + \frac{q_{send}}{|\mathcal{D}|} + Adv_{\mathcal{P}}^{CDH}(t)$$

VII. Simulation of proposed protocol using AVISPA tool

The AVISPA is a simulator for verifying whether a cryptographic protocol is secure or not [**?**], [**?**]. To simulate our protocol, it is implemented using High Level Protocols Specification Language (HLPSL) that supports four different back-ends: (i) OFMC, (ii) CL-AtSe, (iii) SATMC, and (iv) TA4SP [**?**], [**?**] . The HLPSL implementation consists of the roles played by the participants, sending and receiving events of the participants in a bounded number of sessions, immediate actions taken by the participants on the reception of a message, and the intruder's knowledge. The sending and receiving events are denoted by two separate channels: (i) sending channel and (ii) receiving channel. The role played by an intruder is demonstrated by Dolev-Yao threat model [**?**]. The simulation runs based on one or more user-defined secrecy and authentication goals to decide the underlying protocol's safety. The secrecy goal defines a secret parameter of a participant throughout the session. The authentication goal determines the authenticity of the sender of the message. Any violation in these goals (such as intruder reveals the secret parameter or authentication of the participant is not satisfied) will result in an unsafe protocol. In our simulation, we consider three different roles: (i) Ui, (ii) TA, and (ii) LPU. A secure channel between Ui and TA is achieved during registration through the symmetric key encryption/decryption process. All of the participants' sending and receiving events in the key agreement phase are implemented accordingly, considering a public channel. The intruder has been given complete control over public media, i.e., they can obtain the parameters of any (or all) of the access and authentication request/response messages. However, the intruder has no direct access to the security parameters of the participants. The simulation results of our protocol are given in below.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/krittibas/avispa-1.1/testsuite/
 results/hc2.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.72s
  visitedNodes: 130 nodes
  depth: 6 plies
```

Fig. 10. Simulation results of our protocol for OFMC back-end.

VIII. Security Analysis of Proposed Scheme :

A. User Anonymity and untraceability:

Anonymity is an important attribute to prevent vulnerability from forger.The identity of an user never disclose during communication either in reliable or unreliable channel. In our scheme Original Identity of user ID_i never disclose either any phase. An forger can not revile ID_i from message$\{V_{t+2} = (R_i \| ID_j \| K_{it}) \oplus K_{it}\}$where$R_i = H(ID_i \| r) \cdot P$ and $K_{it} = s \cdot a \cdot P$ due to computationally infeasible of one way hash function []. User Identity ID_i is not involve any other message like$\{\ V_{t+3}, V_{t+4}, V_{t+5}, V_{t+7}\}$ and also generated using random nonce for every new session. If the $\mathcal{F}$ gets the transmitted

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/krittibas/avispa-1.1/testsuite/
  results/hc2.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed    : 63 states
  Reachable   : 63 states
  Translation: 0.07 seconds
  Computation: 0.01 seconds
```

Fig. 11. Simulation results of our protocol for CL-AtSe back-end.

message he/she can not link them to trace the User activity. So our protocol achieve anonymity and untraceability.

B. Mutual authentication:

In cryptography Mutual authentication is one of the important security property during sensitive data transmission over public channel. Mutual authentication means each communicating party validate to each other. In our scheme three party user(U_i) , Trusted server(TA) and local processing unit(LPU_i) validate to each other during publicly data communication.

(i) In the log-in and authentication phase user in *step* 1(V-D) , (U_i) verify his/her credential by checking smart card information after provide ID_i and PW_i and computes $PWD_i = H(ID_i \| PW_i)$ then decrypt $r, K_T{}' = Dec_{PWD_i}\{R_1\}$ then verify $V'_{t+1} \stackrel{?}{=} H(V_t \| r_i \| K_T{}')$. If verified then it ensure legitimacy of U_i.

(ii) In *step* 2(V-D) Trusted server(TA) verify the legitimacy of user(U_i) and local processing unit (LPU_i) by computing $K'_{it} = s \cdot A = s \cdot a \cdot P$ and $R_i \| ID_j \| K_{it} = V_{t+2} \oplus K_{it}$ then verify $K'_{it} \stackrel{?}{=} K_{it}$ if true then accept R_i and ID_j for legitimacy of U_i and local processing unit(LPU_j) respectively.

(iii) In *step* 3(V-D)Local processing unit (LPU_j) verify the legitimacy of TA by computing $K_{tl} = X_{tj} \cdot S = X_{tj} \cdot s_t \cdot P$ extract $T_{sk}, K_{it} = V_{t+3} \oplus K_{tl}$ then checks $T_{sk} \stackrel{?}{=} H(K_{it} \| K_{tl})$ if True then TA will authenticate.

(iv)In *step* 4(V-D)User(U_i) verify the legitimacy of Trusted server(TA) by checking $T_{sk} \stackrel{?}{=} H(K_{it} \| K_{tl})$ where $T_{sk}, K_{tl} = Dec_{K_{it}}\{V_{t+4}\}$

(v) In *step* 5(V-D) Local processing Unit (LPU_j) authenticate U_i by decrypting V_{t+5} i.e $ID_i \| k_i \| ID_j = Dec_{T_{sk}}\{V_{t+5}\}$

(vi)In *step* 6(V-D)The User(U_i) further checks $V_{t+6} \stackrel{?}{=} H(SK \| k_i \| k_j)$ to verify the authenticity of local processing unit(LPU_j) by computing $k_j \| V_{t+6} = Dec_{T_{SK}}\{V_{t+7}\}$ and $SK = H(k_i \| k_j)$.

C. Lost smart cards attack :

i.Stolen smart card attack : To obtain user credential ID_i, PW_i from smart card. $\mathcal{F}$ have to compute $R_i = H(ID_i \| r_i) \cdot P$ so, to guess user identity ID_i $\mathcal{F}$ have to solve $ECDL$. So the advantage of solving $ECDL$ is negligible i.e $Adv_{\mathcal{P}}^{ECDL}(t) \leq \epsilon$ and also $\mathcal{F}$ have to solve one way hash function the probability of solving this $\epsilon(k) \leq \frac{1}{k^l}$ this value also negligible. Therefore our scheme protect from privilege insider attack and stolen smart card attack.

ii.Offline password guessing attack : To guess U_i password PW_i in offline, $\mathcal{F}$ have to compute $PWD_i = H(ID_i \| PW_i)$ computing correct PWD_i then compute$(r_i \| K_t^{prime} = Dec_{PWD_i}(R_{i+1})$. So the probability of guessing correct PWD_i is $\frac{1}{k^l}$ which is negligible. So our scheme is prevent Offline password guessing attack.

iii.Online password guessing attack : $\mathcal{F}$ may try online password guessing attack with the help of obtain information from smart card. To lunch valid log-in request i.e V_{i+2}, A the $\mathcal{F}$ have to know valid ID_i and PW_i . $\mathcal{F}$ have to compute $V_{i+2} = (R_i \| D_j \| K_{it}) \oplus K_{it}$ and $A_{i+1} = a.P$ where $K_{it} = s.a.P$ and $R_i = H(ID_i \| r_i).P$ and also successfully passes the verification steps $V_{t+1}^{prime} \stackrel{?}{=} H(V_t \| r_i \| K_t^{'})$. To get r_i, $\mathcal{F}$ have to select original password from password dictionary $\mathcal{D}$ with limited number of attempted by using $Send(\pi_P^k, m)$ query. $(r_i \| K_t^{'}) = Dec_{PWD_i}(R_{i+1})$ and $PWD_i = H(ID_i \| PW_i)$. So success probability guessing password is $\frac{q_{send}}{|\mathcal{D}|}$ which is negligible So, this scheme protect onlinr password guassing attack.

D. Denial of services:

In Our scheme For each step receiver will check the legitimacy of sender's request . Therefore, there is no chance to drop the request which coming from a valid user. So, our scheme is prevent from Denial of services attack.

E. Protect Clock Synchronization Problem:

our scheme never use Time stamp technique during log in authentication phase or session key generation phase . So no need to synchronize the global clock to maintain the freshness of all message by which we can prevent this attack.

F. Impersonation attack:

Any attacker can not impersonate local processing unit(LPU_i),Trusted Agent(TA) and User(U_I) with out knowing valid ID_i, PW_i, K_{it} and K_{tl} forger never generate valid requests message $V_{t+2}, V_{t+3}, V_{t+4}$and V_{t+5} and V_{t+7}.

- $UserImpersonation(U_i)$: To impersonate User, $\mathcal{F}$ have to pass the verification step successfully i.e $V'_{t+1} \stackrel{?}{=} H(V_t \| r_i \| K_T{}')$ which is not feasible.
- $TrustedServerImpersonation(TA)$: Adversary can not impersonate TA due to verification step of $K'_{it} \stackrel{?}{=} K_{it}$ where $K'_{it} = s \cdot A = s \cdot a \cdot P$.

- $LPU_j Impersonation$: To impersonate Local processing unit, $\mathcal{F}$ have to successfully compute $K_{tl} = X_{tj}\ cdots_t\ cdotP$ and successfully verify $T_{sk} \stackrel{?}{=} H(K_{it}\|K_{tl})$. It is infeasible to compute K_{tl} in polynomial time. Therefore, $\mathcal{F}$ can not impersonate LPU_j. So Our scheme protect from Impersonation attack.

G. Man in the middle attack:

In our scheme every phase maintain verification as well as encryption and decryption process. So while sending message one user to another user it will encrypted format and in the receiver end message will be decrypt first then verify the legitimacy of sender. So there is no chance to tamper or alter the message by an forger in public channel that's why our scheme are not vulnerable form man-in the-middle attack.

IX. Conclusion :

This article establishes a shared session key between two legitimate parties they are User and Local processing sensor unit. Also, our scheme achieves privacy, authentication, integrity and free from all kinds of well-known security threats. The performance analysis phase ensures that the protocol is free from several attacks with improved computational complexity. So we can use this scheme for IoT based health care services.