A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment

Amin, R, Kumar, N, Biswas, GP, Iqbal, R & Chang, V

Original citation & hyperlink:
Amin, R, Kumar, N, Biswas, GP, Iqbal, R & Chang, V 2018, 'A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment' Future Generation Computer Systems, vol 78, no. 3, pp. 1005-1019
https://dx.doi.org/10.1016/j.future.2016.12.028

DOI 10.1016/j.future.2016.12.028
ISSN 0167-739X

Publisher: Elsevier

Accepted Manuscript

A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment

Ruhul Amin, Neeraj Kumar, G.P. Biswas, R. Iqbal, Victor Chang

PII: S0167-739X(16)30824-X
DOI: http://dx.doi.org/10.1016/j.future.2016.12.028
Reference: FUTURE 3269

To appear in: *Future Generation Computer Systems*

Received date: 16 August 2016
Revised date: 18 November 2016
Accepted date: 20 December 2016

Please cite this article as: R. Amin, N. Kumar, G.P. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment, *Future Generation Computer Systems* (2016), http://dx.doi.org/10.1016/j.future.2016.12.028

***Highlights (for review)**

Dear FGCS Editor-in-Chief and Lead Guest Editor,

We have made significant improvements and have fully addressed reviewers' requests. We have demonstrated security theory and blended the latest work with our proofs-of-concept. We hope that you can consider our paper. Many thanks.

Yours sincerely,

Dr. Victor Chang on behalf of all co-authors

18 November 2016

Highlights

- We have developed Light Weight Authentication Protocol for IoT-enabled Devices in Distributed Cloud Computing Environment.
- We show security vulnerabilities of the multiserver cloud environment of the protocols proposed by Xue et al. and Chuang et al. and propose an architecture which is applicable for distributed cloud environment and based on it an authentication protocol using smartcard.
- We have used AVISPA tool and BAN logic model and informal cryptanalysis confirms that the protocol is protected against all possible security threats.
- The performance analysis and comparison confirm that the proposed protocol is superior than its counterparts.

Available online at www.sciencedirect.com
ScienceDirect
00 (2016) 1–27

A Light Weight Authentication Protocol for IoT-enabled Devices in Distributed Cloud Computing Environment

Ruhul Amin[1], Neeraj Kumar[1,*], G.P. Biswas[2], R. Iqbal[3], Victor Chang[4]

Abstract

With the widespread popularity and usage of Internet-enabled devices, Internet of things has become popular now a days. However, data generated from various smart devices in IoT is one of the biggest concerns. To process such a large database repository generated from all types of devices in IoT, Cloud Computing (CC) has emerged as a key technology. But, the private information from IoT devices is stored in distributed private cloud server so that only legitimate users are allowed to access the sensitive information from the cloud server. Keeping focus on all these points, this article first shows security vulnerabilities of the multi-server cloud environment of the protocols proposed by Xue et al. and Chuang et al. Then, we propose an architecture which is applicable for distributed cloud environment and based on it, an authentication protocol using smartcard has been proposed, where the registered user can access all private information securely from all the private cloud servers. To proof security strength of our protocol, we have used AVISPA tool and BAN logic model in this article. In addition, informal cryptanalysis confirms that the protocol is protected against all possible security threats. The performance analysis and comparison confirm that the proposed protocol is superior than its counterparts.

Keywords: Authentication, *AVISPA* tool, *BAN* logic, Distributed Cloud Environment, Security Attacks.

1. Introduction

In the year 1999, the concept of the Internet of things (IoT) was introduced by the scientist Ashton. It is the basically set of interconnected things such as sensor devices, tags, and smart objects over the Internet networks. All these devices must have the capability to collect data and communicate the same to all other devices deployed across the globe. The main focus of IoT is to get information from the environment which can be shared among different other devices. Thus, IoT is an important technology in our daily life [1]. For an example, in smart-home environment, people life-style is improved using home energy consumption with the help of a set of home sensor devices. Also this technology is useful in several practical applications such as Control systems Ambient-Assisted Living, Safer Mining Production, Smart Unit and Tracking etc. However, the IoT usually coincides with sensors with low memory, low power and battery and network limitations. Therefore, it is important to compute, store, access and analysis of IoT data. Additionally, there should be a standard platform that can handle efficiently large amount of heterogeneity data and devices, as the data and devices are growing [1] exponentially.

*Corresponding author. (Neeraj Kumar)
Email addresses: amin_ruhul@live.com (Ruhul Amin[1]), neeraj.kumar@thapar.edu (Neeraj Kumar[1,*]), gpbiswas@gmail.com (G.P. Biswas[2]), r.iqbal@coventry.ac.uk (R. Iqbal[3]), ic.victor.chang@gmail.com (Victor Chang[4])
[1]Department of Computer Science and Engineering, Thapar University, Patiala-147004, India
[2]Department of Computer Science and Engineering, Indian School of Mines, Dhanbad, India
[3]Department of Computing , Faculty of Engineering, Coventry University, Coventry, UK
[4]International Business School Suzhou, Xi?an Jiaotong Liverpool University, Suzhou, China

To handle all the above issues discussed, there is a requirement of a unified technology such as cloud using which information can be accessed from anywhere. Presently, a lot of cloud services are available from public and private servers for the internet users. In general, public cloud platforms are open for all user and private cloud services are imperative as it is not accessible without authorization. There are basically several types of services provided by the cloud such as Software as a Service (SaaS) cloud (Ex. IBM LotusLive), Platform as a Service (PaaS) (Ex. Google AppEngine) and Infrastructure as a Service (IaaS)(Ex. Amazon Web Services). Private clouds are owned and used by a single organization or department. In security point of view, accessing data from the private cloud server is not feasible by the internet client without authorization. Therefore, it is very imperative issue to check authorization of the client before accessing. With the rapid development of the Internet and electronic commerce technology, many services are provided to the client/user through the Internet communication such as online shopping, online game, distributed electronic medical records system etc. Among all these applications, cloud security [2] is also an important issues in business perspective. To authorize the client, several authentication protocols are available, but password with hash function based are most acceptable due to easy implementation. In this article, we have designed a distributed environment for the private server where the client could get services on completing authorization process. To complete authorization process, this article designs an authentication protocol which authenticates the client and then agrees upon a common secret session key for secure communication.

1.1. Literature Review

In 1981, Lamport [8] first suggested authentication technique using password over untrusted networks. However, the protocol depends on the password table which leads to stolen-verifier attack at the server end. Thereafter, many [4, 5, 6, 9, 11, 12, 13, 14, 15, 16, 17, 18] user authentication with password and key negotiation techniques have been put forward for client server communication model. In 2001, Li et al. [19] first put forward multi-server authentication protocol using neural network concept and they had shown that client can choose his/her password freely. Then, Lin et al. [20] states that the protocol in [19] is not efficient due to heavy time complexity. Then they utilized Elgamal Digital Signature [21] and geometric properties on the Euclidean plane to design a password based authentication scheme. Cao et al. [22] suggested that Lin et al. [20] protocol is not secure against impersonation attack and takes large memory for storage public parameter into memory of smart card for each user. Thereafter, Juang [23] proposed password and nonce based multi-server authentication protocol. Later on, Ku et al. [24] stated that Juang protocol cannot resist insider attack and forward secrecy is not provided, whereas Cheng et al. [32] suggested better solution of the protocol in [24]. In 2007, Liao et al. [25] suggested a key agreement protocol using the concept of dynamic identity for multi-server environment based on cryptographic hash function and declared that their protocol satisfies all the relevant security aspects of multi-server environment. After long time in 2009, the authors in [26] demonstrated that the protocol in [25] is vulnerable to several security threats and designed an extended protocol and declares that it takes low complexity, higher security and the efficiency is better than previous research. In 2011, Sood et al. [31] criticized that the protocol in [26] is susceptible several imperative attacks and the password change process is not accurate. Then, Sood et al. [31] put forward a dynamic identity based multi-server authentication protocol. In 2012, Li et al. [7] demonstrated that the protocol in [31] is incorrect and not attack protected. To improve security, they developed a counter measure protocol. In 2014, Xue et al. [3] stated that the protocol in [7] is meaningless due to not protecting several security threats and they also suggested a better protocol for security improvement.

1.2. Motivation and Contributions

Our examination on the research for multi-server authentication states that all existing research are not completely protected against security threats. Therefore, Our aim is to develop a security attacks free authentication protocol which can be used in distributed cloud environment. This article contributes the following aspects.

1. We have examined the protocol in [3] and demonstrated that it is not protected against user anonymity problem, off-line password guessing attack, insider attack and user impersonation attack. The same protocol also has incorrect design issues in the authentication phase.
2. We have also demonstrated that the Chuang et al.'s protocol cannot not resist user impersonation and session key discloser attack.
3. For the security and complexity improvement, we design an authentication protocol for distributed cloud system.

4. For the mutual authentication proof, we have used BAN logic model. Further, the entire protocol has been simulated using AVISPA software, whose results ensure safe and sound.
5. It is also our contribution that we offer password and identity change phases in our protocol.

1.3. Road map of the paper

In Section 2, we briefly addresses the Xue et al.'s work. Cryptanalysis of the same scheme is given in Section 3. Section 4, addresses briefly reviews and cryptanalysis of the Chuang et al.'s protocol. We design and present our protocol in Section 5. The *BAN* logic analysis, simulation using *AVISPA* tool and informal cryptanalysis appear in Section 6. The Section 7 evaluates and judges our protocol with previous research works. We present conclusion of this article in Section 8.

2. Brief Review of Xue et al.'s Scheme

This section briefly reviews the Xue et al. [3] scheme which involves three types of entity such as user U_i, service provider server S_j and control server (CS). The CS mainly provides registration procedure to all U_i and S_j. The S_j provides set of services to all the users on demand. The notations used in this article are recorded in Table 1.

Table 1. Notations Table

Symbol	***Description***
CR	Card Reader
S_j	*j-th* service provider server
S_m	*m-th* cloud server
U_i	*i-th* user
CS	The control server
ID_i	Identity of the user U_i
P_i	Password of the user U_i
x	Secret number only known to CS
y	Secret number of CS
d	Random number of S_j
b	Random number of U_i
$h(\cdot)$	Cryptographic one-way hash function $(0,1)^l \rightarrow (0,1)^n$
T	Timestamp
$\triangle T$	Estimated time dealy
SK	Secret session key
$\oplus$	Bit-wise xor operation
$\|\|$	Concatenate operation

2.1. Registration Phase

The U_i choices desired identity ID_i, password P_i, a random number b and calculates $A_i = h(b \parallel P_i)$ and submits registration message $\langle ID_i, A_i, b\rangle$ to the CS. Now the CS first takes two random numbers $\langle x, y\rangle$ and calculates $PID_i = h(ID_i \parallel b)$, $B_i = h(PID_i \parallel x)$ and forwards B_i to the user securely. After receiving B_i, the U_i calculates $C_i = h(ID_i \parallel A_i)$, $D_i = B_i \oplus h(PID_i \parallel A_i)$ and embeds $\langle C_i, D_i, b, h(\cdot)\rangle$ in the smart card.

During the service provider server registration, the S_j choices identity SID_j, a random number d and sends $\langle SID_j, d\rangle$ to the CS. After receiving it, the CS calculates $PSID_j = h(SID_j \parallel d)$, $BS_j = h(PSID_j \parallel y)$ and sends $\langle BS_j\rangle$ to S_j securely. Finally, the S_j records secret parameter $\langle BS_j, d\rangle$ into his/her memory.

2.2. Login Phase

The U_i punches the smart card into the card reader and provides ID_i and P_i. Then, the card reader calculates $A_i^* = h(b \parallel P_i)$, $C_i^* = h(D_i \parallel A_i^*)$ and checks the condition ($C_i^* ? = C_i$). If ($C_i^* == C_i$), the card reader accepts the U_i as a legitimacy user; otherwise, rejects the connection.

2.3. Authentication and Key agreement Phase

This phase describes mutual authentication as well as key agreement among the U_i, S_j and the CS. All operations performed in this phase are given below.

Step 1: User U_i generates a current timestamp TS_i, a random number N_{i1} and computes $\langle B_i, F_i, CID_i, G_i, P_{ij}\rangle$ as follows:

$$B_i = D_i \oplus C_i$$
$$F_i = B_i \oplus N_{i1}$$
$$CID_i = ID_i \oplus h(B_i \parallel N_{i1} \parallel TS_i \parallel \text{"00"})$$
$$G_i = b \oplus h(B_i \parallel N_{i1} \parallel TS_i \parallel \text{"11"})$$
$$P_{ij} = h(Bi \oplus h(N_{i1} \parallel SID_j \parallel PID_i \parallel TS_i))$$

where "00" is a 2 bit binary "0" and "11" are 2 bit binary "1". Then, U_i forwards $\langle F_i, P_{ij}, CID_i, PID_i, G_i, TS_i\rangle$ to S_j publicly.

Step 2: After getting messages from U_i, S_j first checks the time interval condition ($TS_j - TS_i < \triangle T$), where TS_j, $\triangle T$ is the S_j's current timestamp and expected time interval during message transmission respectively. If the condition is not false, S_j proceeds; otherwise, stops this session. Then, the S_j produces a random number N_{i2} and calculates the following operations:

$$J_i = BS_j \oplus N_{i2}$$
$$K_i = h(N_{i2} \parallel BS_j \parallel P_{ij} \parallel TS_i)$$
$$L_i = SID_j \oplus h(BS_j \parallel N_{i2} \parallel TS_i \parallel \text{"00"})$$
$$M_i = d \oplus h(BS_j \parallel N_{i2} \parallel TS_i \parallel \text{"11"})$$

The S_j then sends $\langle F_i, P_{ij}, CID_i, G_i, PID_i, TS_i, J_i, K_i, L_i, M_i, PSID_j\rangle$ to the CS publicly.

Step 3: After getting messages from S_j, CS first checks the condition ($TS_{cs} - TS_i < \triangle T$), where TS_{cs} is the current timestamp of the CS. Stops the connection if the condition is false; otherwise, the CS performs the following operations:

$$BS_j = h(PSID_j \parallel y)$$
$$N_{i2} = J_i \oplus BS_j$$
$$K_i^* = h(N_{i2} \parallel BS_j \parallel P_{ij} \parallel TS_i)$$

The CS checks the condition ($K_i^* ? = K_i$). If ($K_i^* == K_i$), it further calculates:

$$B_i = h(PID_i \parallel x)$$
$$N_{i1} = B_i \oplus F_i$$
$$ID_i = CID_i \oplus h(B_i \parallel N_{i1} \parallel TS_i \parallel \text{"00"})$$
$$SID_j = L_i \oplus h(BS_j \parallel N_{i2} \parallel TS_i \parallel \text{"11"})$$
$$P_{ij}^* = h(B_i \oplus h(N_{i1} \parallel SID_j \parallel PID_i \parallel TS_i))$$

Then, the CS checks the condition whether ($P_{ij}^* ? = P_{ij}$) or not. If ($P_{ij}^* \neq P_{ij}$), stops this session; otherwise, calculates the following operations:

$$b = G_i \oplus h(B_i \parallel N_{i1} \parallel TS_i \parallel \text{"11"})$$
$$d = M_i \oplus h(BS_j \parallel N_{i2} \parallel TS_i \parallel \text{"00"})$$
$$PID_i^* = h(ID_i \parallel b)$$
$$PSID_j^* = h(SID_j \parallel d)$$

The CS checks whether ($PID_i^* = PID_i$) and ($PSID_j^* = PSID_j$) are correct or not. If these condition is not false, the CS takes a random number N_{i3} and calculates the following operations:

$$P_i = N_{i1} \oplus N_{i3} \oplus h(SID_j \parallel N_{i2} \parallel BS_j)$$
$$Q_i = h(N_{i1} \oplus N_{i3})$$
$$R_i = N_{i2} \oplus N_{i3} \oplus h(ID_i \parallel N_{i1} \parallel B_i)$$
$$V_i = h(N_{i2} \oplus N_{i3})$$

Then, the CS sends $\langle P_i, Q_i, R_i, V_i \rangle$ to the S_j.

Step 4: On the receipt of reply message from CS, the S_j calculates the following operations:

$$N_{i1} \oplus N_{i3} = P_i \oplus h(SID_j \parallel N_{i2} \parallel BS_j)$$
$$Q_i^* = h(N_{i1} \oplus N_{i3}).$$

Then, the S_j verifies whether ($Q_i^* ? = Q_i$). If ($Q_i^* == Q_i$), it implies that the CS and U_i are authentic and sends reply messages $\langle R_i, V_i \rangle$ to the user U_i.

Step 5: On the receipt of reply message from S_j, the U_i calculates,

$$N_{i2} \oplus N_{i3} = R_i \oplus h(ID_i \| N_{i1} \| B_i)$$
$$V_i^* = h(N_{i2} \oplus N_{i3})$$

Then, the U_i checks the condition ($V_i^* ? = V_i$). If ($V_i^* == V_i$), the U_i confirms that CS and S_j are authentic. Finally, the U_i, S_j and CS agree upon a common secret key $SK = h((N_{i1} \oplus N_{i2} \oplus N_{i3}) \parallel TS_i)$.

2.4. Password Update phase

After password authentication in the registration phase, the user U_i's password P_i does not appear in B_i. Thus, password renewal procedure can execute in anytime without $CS's$ helps. U_i can renew the parameters in smart card.

$$C_i' = h(ID_i \parallel A_i')\ D_i' = B_i \oplus h(PID_i \oplus A_i')$$

In order to keep password consistency between U_i and CS, U_i needs to submit his/her ID_i and A_i' with a new password P_i' to CS via secure channel. CS renew U_i's password in the verification table. However, the submission process does not have to happen after the password changing immediately.

2.5. Identity Update phase

In order to update the identity of the U_i, the U_i re-choices a random number b^* and then calculates $A_i^* = h(b^* \parallel P_i)$ and submits $\langle ID_i, b^*, A_i^* \rangle$ to CS. After verifying user's legitimacy, the CS re-computes $PID_i^* = h(ID_i \parallel b^*)$, $B_i^* = h(PID_i^* \parallel x)$ and submits B_i^* to the U_i through any private communication. Then, the U_i calculates $C_i^* = h(ID_i \parallel A_i^*)$, $D_i^* = B_i^* \oplus h(PID_i^* \oplus A_i^*)$. At the end, the smart card updates $\langle C_i^*, D_i^*, b^*, h(\cdot) \rangle$. Now, the U_i's protected pseudonym identity PID_i is dynamically changed to PID_i^*.

In the case of service provider server, the S_j re-choices a random number d^* and uses identity S_j to register with the CS. Then, the CS computes $PSID_j^* = h(SID_j^* \parallel d^*)$, $BS_j^* = h(PSID_j^* \parallel y)$ and sends BS_j^* to S_j through private communication. Finally, the S_j updates BS_j^*, d^* in his/her memory and completes the identity updates phase.

3. Cryptanalysis of Xue et al.'s Scheme

This section cryptanalyses the authentication protocol proposed by Xue et al. We assume some valid assumptions which are recorded in [10, 36, 35, 9] for making cryptanalysis of the protocol in [3].

3.1. User Anonymity

The authors in [3] stated that their protocol is user anonymous that means no one can know the legal user's identity. However, we have observed that the attacker can easily compute the legal user identity based on the smart card information and protocol description.

The attacker computes the following operations:

$$B_i^a = C_i \oplus D_i,$$
$$N_{i1}^a = F_i \oplus B_i^a$$
$$ID_i^a = CID_i \oplus h(B_i^a \parallel N_{i1}^a \parallel TS_i \parallel \text{"00"})$$

It can be easily shown that $ID_i^a = ID_i$. Hence, the protocol is not user anonymous.

Correctness: $ID_i^a = ID_i$

$ID_i^a = CID_i \oplus h(B_i^a \parallel N_{i1}^a \parallel TS_i \parallel \text{"00"})$.

$= CID_i \oplus h((C_i \oplus D_i) \parallel (C_i \oplus D_i \oplus Fi) \parallel TS_i \parallel \text{"00"})$. $[\, Since B_i^a = C_i \oplus D_i]$

$= CID_i \oplus h(B_i \parallel N_{i1} \parallel TS_i \parallel \text{"00"})$

$= ID_i$

3.2. Off-line Password Guessing Attack

In general, the user always takes password which is low-entropy and can guess it in off-line approach. The protocol proposed in [3] is not protected against the above attack. The Algorithm 1 presents execution of the above attack.

Algorithm 1

```
1: Input: ⟨C_i, D_i, b, h(·), ID_i⟩, Where ID_i is obtained from the user anonymity problem.
2: Output: P_i.
3: Adversary computes C_i = h(ID_i || A_i) = h(ID_i || h(P_i || b)).
4: Adversary takes word as a password P_i^g from the small dictionary (D).
5: Computes C_i^g = h(ID_i || h(P_i^g || b)).
6: if (C_i^g == C_i) then
7:     Return(pw_A^g);
8: else
9:     Go to step 3 until correct password is obtained
10: end if
```

The description in Algorithm 1 clearly states that after extracting all the smart card information, the attacker can easily guess legal user's password.

3.3. Privileged Insider Attack at the Server End

In the registration phase, the U_i puts in $\langle A_i, b\rangle$ to the CS through secure channel, where $A_i = h(b \parallel P_i)$ and b is the random number. Hence, the insider attacker of the system can verify the guessed password using the P_i parameters. As a good number of users use low entropy and identical password to login into remote system, the insider attacker of the system may access others account of the others server. Therefore, we may claim that the protocol suggested by Xue et al. is not protected against insider attack.

3.4. Session Key Discloser Attack

The protocol suggested in [3] is vulnerable to session key discloser attack as the attacker can easily calculate it. The technique to calculate the session key is as follows.

Step 1: The attacker calculates the following operations.

$$B_i = C_i \oplus D_i$$
$$N_{i1} = F_i \oplus B_i$$
$$N_{i2}^a \oplus N_{i3}^a = R_i \oplus h(ID_i \parallel N_{i1} \parallel B_i)$$

where, ID_i is the legal user's identity obtained from the user anonymity description.

Step2: Now, the attacker computes the session key $SK^a = h((N_{i1} \oplus N_{i2} \oplus N_{i3}) \parallel TS_i)$. It is correct information that the session key computed by the attacker is same with the protocol in [3]. Therefore, the protocol in [3] is not protecting the above attack.

3.5. User Impersonation Attack

It is practical that a legal user may be leaked server's private information to the attacker. The legal user also can act as an attacker. It can be assumed that the legal user provides the identity of S_j to the attacker. The execution procedure of the above attack is described below:

Step 1: The attacker produces a random number N_i^a and computes the following operations:

$$B_i^a = C_i \oplus D_i$$
$$F_i^a = B_i \oplus N_i^a$$
$$P_{ij}^a = h(B_i^a \oplus h(N_i^a \parallel SID_j \parallel PID_i \parallel TS_a))$$
$$CID_i = ID_i \oplus h(B_i^a \parallel N_i^a \parallel TS_a \parallel \text{"00"})$$
$$G_i^a = b \oplus h(B_i^a \parallel N_i^a \parallel TS_a \parallel \text{"11"})$$

where, TS_a is the current timestamp generated by the attacker.

Step 2: Attacker then sends $\langle F_i^a, P_{ij}^a, CID_i, PID_i, G_i^a, TS_a \rangle$ to the S_j. As the timestamp TS_a is valid, it is confirmed that time interval at the S_j end should correct. The S_j now sends $\langle F_i^a, P_{ij}^a, CID_i, PID_i, G_i^a, TS_a, J_i, K_i, L_i, M_i, PSID_j \rangle$ to the control server CS.

Step 3: The CS calculates the following operations:

$$B_i = h(PID_i \parallel x)$$
$$N_{i1}^a = B_i \oplus F_i^a$$
$$ID_i = CID_i \oplus h(B_i \parallel N_{i1}^a \parallel TS_a \parallel \text{"00"})$$
$$b = G_i^a \oplus h(B_i \parallel N_{i1}^a \parallel TS_a \parallel \text{"11"})$$
$$PID_i^* = h(ID_i \parallel b)$$

Now, the CS checks $(PID_i^* ? = PID_i)$. If $(PID_i^* == PID_i)$, the attacker can impersonate as an authorized user to CS. Thus, the protocol is not protecting against the above attack.

3.6. Design Flaws in the Authentication Phase

Step 1: In step 1 of this phase, the U_i calculates $B_i = D_i \oplus C_i = B_i \oplus h(PID_i \parallel A_i) \oplus h(ID_i \parallel A_i) = h(PID_i \parallel x) \oplus h(PID_i \parallel A_i) \oplus h(ID_i \parallel A_i)$, $F_i = B_i \oplus N_{i1}$ and uses $\langle B_i, F_i \rangle$ as login message of the protocol. Finally, control server CS has got $\langle B_i, F_i \rangle$ parameters from the login message.

Step 2: In step 3, CS computes $B_i^* = h(PID_i \parallel x)$ and extracts $N_{i1}^* = B_i^* \oplus F_i$. Now, it is confirm that $N_{i1}^* \neq N_{i1}$, as $B_i^* \neq B_i$. however, it must be $N_{i1}^* = N_{i1}$.

Correctness of $B_i^* \neq B_i$

$B_i = D_i \oplus C_i$
$= B_i \oplus h(PID_i \parallel A_i) \oplus h(ID_i \parallel A_i)$
$= h(PID_i \parallel x) \oplus h(PID_i \parallel A_i) \oplus h(ID_i \parallel A_i)$
$\neq h(PID_i \parallel x)$ since, $h(PID_i \parallel A_i) \neq h(ID_i \parallel A_i)$
$\neq B_i^*$

Step 3: The above description confirms that the protocol rejects user in each authentication cycle, though the user inputs valid information during the login phase. Therefore, it can be strongly concluded that the suggested protocol in [3] is not applicable for practical application.

3.7. AVISPA Simulation of Xue et al. Protocol

We have demonstrated that the Xue et al.'s protocol has several security loopholes. This section shows through simulation results that the same protocol [3] is not secure against replay and man-in-the-middle attacks using AVISPA tool. We have included OFMC and Cl-AtSe results in Figure 1 and Figure 2 respectively.

```
% OFMC
SUMMARY
UNSAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileoDDeHN.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.16s
VisitedNodes: 14 nodes
depth: 4 plies
```

Figure 1. OFMC result

```
SUMMARY
UNSAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileoDDeHN.if
GOAL
As Specified
BACKEND
CL-AtSe
STATISTICS
Analysed : 6 states
Reachable : 4 states
Translation: 0.06 seconds
Computation: 0.00 seconds
```

Figure 2. CL-AtSe result

4. Brief Review and Cryptanalysis of Chuang et al. Protocol

We briefly present the Chuang et al.'s [30] protocol first and then discuss some security weaknesses. we refer to the reader for details information about the Chuang et al.'s protocol in [30]. All phases of the protocol in [30] is given below.

4.1. Registration Phase

The S_j forwards the message to the CS for becoming an authorized server, and on receiving the message the CS replies shared secret key PSK to the S_j.

Step 1: $U_i \rightarrow CS : ID_i, h(P_i \oplus BIO_i)$ in person, where BIO_i is the user's biometric template.

Step 2: The CS computes the following operations:

$$A_i = h(ID_i \parallel x)$$
$$B_i = h(A_i)$$
$$C_i = h(P_i \oplus BIO_i) \oplus B_i$$
$$D_i = PSK \oplus A_i$$

Then, the CS transports a smart card after storing $\langle ID_i, B_i, C_i, D_i, h(\cdot)\rangle$ in the smart card.

4.2. Login Phase

The user provides $\langle ID_i, P_i\rangle$ to the card reader and BIO_i at the sensor devices. Then, the card reader checks the format of ID_i and verifies whether $h(P_i \oplus BIO_i) \oplus C_i$ matches with B_i or not. If both the condition matches, the smart card computes the following operations:

$$M_1 = h(B_i) \oplus N_1,$$
$$AID_i = h(N_1) \oplus ID_i,$$
$$M_2 = h(N_1 \parallel AID_i \parallel D_i)$$

The smart card then sends $\langle AID_i, M_1, M_2, D_i\rangle$ to the S_j, where N_1 is the random number produced by the smart card.

4.3. Authentication Phase

Step 1: The S_j calculates $A_i = D_i \oplus PSK$, $N_1 = M_1 \oplus h^2(A_i)$ and makes sure whether $h(N_1 \parallel AID_i \parallel D_i)$ matches with M_2 or not. If it matches, the S_j produces a random nonce N_2 and calculates the following operations and sends $\langle SID_j, M_3, M_4\rangle$ to the smart card.

$$SK_{ij} = h(N_1 \parallel N_2),$$
$$M_3 = N_2 \oplus h^2(N_1),$$
$$M_4 = h(SID_j \parallel N_2)$$

Step 2: After receiving $\langle SID_j, M_3, M_4\rangle$, the smart card computes $h^2(N_1)$, retrieves random nonce $N_2 = M_3 \oplus h^2(N_1))$ and checks the condition whether $h(SID_j \parallel N_2)$ matches with M_4 or not. After that, the smart card computes $SK_{ij} = h(N_1 \parallel N_2)$, $SK_{ij} \oplus h(N_2)$ and sends $SK_{ij} \oplus h(N_2)$ to the S_j

Step 3: The S_j uses SK_{ij} to retrieves $h(N_2)$ and then verifies the value $h(N_2)$ whether it is correct or not.

4.4. User Impersonation Attack

The protocol proposed in [30] is not provided security against the above attack. The execution procedure for launching the above attack is as follows.

Step 1: The attacker produces a random nonce N_1^a and calculates the following operations:

$$M_1^a = h(h(A_i)) \oplus N_1^a$$
$$AID_i^a = h(N_1^a) \oplus ID_i$$
$$M_2^a = h(N_1^a \parallel AID_i^a \parallel D_i).$$

Then, the attacker sends $\langle M_1^a, M_2^a, AID_i^a, D_i\rangle$ to the S_j through public channel.

Step 2: After getting the login request $\langle M_1^a, M_2^a, AID_i^a, D_i\rangle$, the S_j extracts and computes the following operations:

$$A_i = PSK \oplus D_i$$
$$N_1^a = M_1^a \oplus h(h(A_i))$$
$$M_2^{'} = h(N_1^a \parallel AID_i^a \parallel D_i)$$

Then, the S_j compares the correctness whether $M_2^a = M_2'$ is correct or not. It can be confirmed that the condition $M_2^a = M_2'$ is true, as both parameters $\langle M_2^a, M_2' \rangle$ depend on the common parameters $\langle N_1^a, AID_i^a, D_i \rangle$. Then, the S_j produces a random number N_2 and calculates $SK = h(N_1^a \parallel N_2)$, $M_3 = N_2 \oplus h(h(N_1^a))$, $M_4 = h(SID_j \parallel N_2)$. Finally, the S_j sends $\langle SID_j, M_3, M_4 \rangle$ to the attacker.

Step 3: After getting message from S_j, the attacker calculates $N_2 = M_3 \oplus h(h(N_1^a))$, $M_4' = h(SID_j \parallel N_2)$ and compares the correctness of $(M_4 ? = M_4')$. If $(M_4 == M_4')$, the attacker calculates $SK = h(N_1^a \parallel N_2)$ as session key of the protocol. The above description ensures that the protocol is not protected.

4.5. Session key Discloser Attack

Our following demonstration states that the protocol in [30] suffers from the above attack.

Step 1: Firstly, the attacker calculates $N_1 = h(B_i) \oplus M_1$ from the login message and $N_2 = h(h(N_1)) \oplus M_3$ from the reply message of the protocol.

Step 2: The computation of the session key of the protocol in [30] relies upon the difficulties of the cryptographic hash function and two random numbers N_1 and N_2.

Now, the attacker easily calculates the session key $SK = h(N_1 \parallel N_2)$, as he/she knows the random number $\langle N_1, N_2 \rangle$. Thus, the Chuang et al.'s protocol fails to resist session key discloser attack.

4.6. AVISPA Simulation of Chuang et al. Protocol

We have simulated the published works of Chuang et al. protocol using AVISPA online web-software and its results show that it is UNSAFE under OFMC and CL-AtSe Models. According the information available in the literature [10], the Chuang et al.'s protocol is not secure against replay and man-in-the-middle attacks. We have shown the simulation results in Figure 3 and in Figure 4 of OFMC and CL-Atse models respectively.

```
% OFMC
SUMMARY
UNSAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileoDDeHN.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.22s
VisitedNodes: 18 nodes
depth: 4 plies
```

Figure 3. OFMC result

5. Our Protocol

A private cloud server basically stores confidential information from the environment using the concept of Internet of Things (IoT). Now, the problem is to access stored confidential information from the private cloud. To solve this

SUMMARY
UNSAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileoDDeHN.if
GOAL
As Specified
BACKEND
CL-AtSe
STATISTICS
Analysed : 12 states
Reachable : 6 states
Translation: 0.02 seconds
Computation: 0.00 seconds

Figure 4. CL-AtSe result

problem, this article put forwards a smartcard based authentication protocol for distributed cloud environment, where the registered user or client can access desired private cloud server securely. To design it, our protocol uses six phases namely (1) cloud server registration, (2) user registration (3) login, (4) authentication, (5) password change and (6) identity update. We present our proposed cloud architecture in Figure 5. In our cloud architecture, there are several private cloud servers which are controlled by the control server and all the private cloud servers are located in distributed manner. On executing our protocol, a valid user or client can access all the private cloud servers. The explanation of our all phases is as follows.

5.1. Registration Phase

Our proposed protocol divides registration phase into two sections i.e. (1) cloud server registration and (2) user registration.

5.2. Cloud Server Registration Phase

During cloud server registration, the S_m chooses an identity SID_m, a random number d and sends $\langle SID_m, d\rangle$ to CS. After receiving it, the CS computes $PSID_m = h(SID_m \parallel d)$, $BS_m = h(PSID_m \parallel y)$ and sends $\langle BS_m\rangle$ to S_m securely. Finally, the S_m stores secret parameter $\langle BS_m, d\rangle$ into his/her memory.

5.2.1. User Registration Phase

During registration in CS, user first chooses desired identity ID_i, password P_i and two random numbers $\langle b_1, b_2\rangle$. Then, the U_i computes $A_i = h(P_i \parallel b_1)$, $PID_i = h(ID_i \parallel b_2)$, $bb_i = b_2 \oplus A_i$ and sends $\langle A_i, PID_i\rangle$ to the CS securely. On getting $\langle A_i, PID_i\rangle$, the CS calculates the following operations:

$$C_i = h(A_i \parallel PID_i)$$
$$D_i = h(PID_i \parallel x)$$
$$E_i = D_i \oplus A_i.$$

Finally, the CS prepares and delivers a new smartcard for each U_i after recording $\langle C_i, E_i, h(\cdot)\rangle$ in the smartcard and transports it to U_i through private communication. After getting it, the U_i records $\langle DP, bb_i\rangle$ in the smartcard, where $DP = h(ID_i \parallel P_i) \oplus b_1$. Finally, the smartcard holds $\langle C_i, E_i, bb_i, DP, h(\cdot)\rangle$.

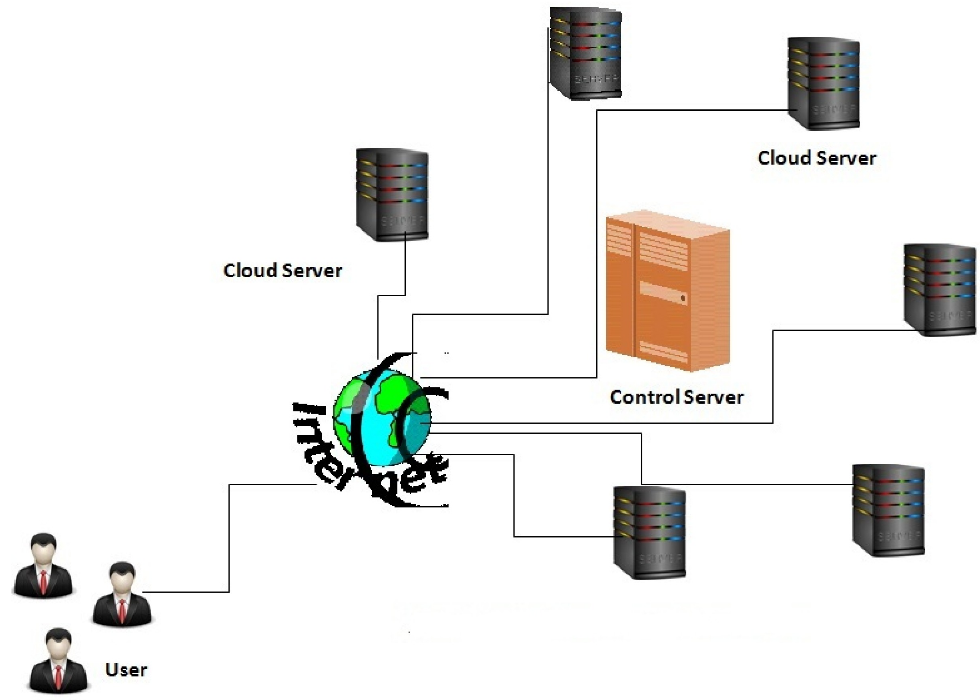

Figure 5. Proposed distributed cloud environment system

***Remark1*:** During the registration phase, we have used two random numbers $\langle b_1, b_2 \rangle$ for resisting insider attack. It is our great approach that the user does not need to remember random numbers $\langle b_1, b_2 \rangle$ and also the smartcard does not store the random numbers.

5.3. Login Phase

For accessing server resources, a legal user U_i first punches the smartcard into card reader CR and inputs ID_i^* and P_i^* to the terminal. Then, the card reader calculates,

$$b_1^* = DP \oplus h(ID_i^* \parallel P_i^*)$$
$$A_i^* = h(P_i^* \parallel b_1^*)$$
$$b_2^* = bb_i \oplus A_i^*$$
$$PID_i^* = h(ID_i^* \parallel b_2^*)$$
$$C_i^* = h(A_i^* \parallel PID_i^*)$$

Then, the CR checks the condition $(C_i^* ? = C_i)$. If $(C_i^* == C_i)$, it means that $(ID_i^* = ID_i)$ and $(P_i^* = P_i)$. The CR produces a 128 bit random number N_i and computes the following operations:

$$D_i = E_i \oplus A_i$$
$$G_i = h(PID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i)$$
$$F_i = D_i \oplus N_i$$
$$Z_i = SID_m \oplus h(D_i \parallel N_i)$$

where SID_m is the cloud server's identity chosen by the user U_i. Then, the CR transmits the login messages $\langle G_i, F_i, Z_i, PID_i, TS_i \rangle$ to the S_m publicly.

5.4. Authentication Phase

This phase is necessary for performing mutual authentication as well as key agreement among U_i, S_m and CS. The details explanation of this phase are as follows.

Step 1: The S_m first checks the condition whether $(TS_m - TS_i < \triangle T)$ holds or not on receiving the login message, where TS_m, $\triangle T$ are the cloud server's current timestamp and expected valid time interval for transmission delay respectively. If the condition is not true, the S_m terminates the connection; otherwise, the S_m produces a 128 bit random number N_m and computes the following operations:

$$J_i = BS_m \oplus N_m,$$
$$K_i = h(N_m \parallel BS_m \parallel G_i \parallel TS_j)$$

Finally, the S_m sends $\langle J_i, K_i, PSID_m, G_i, F_i, Z_i, PID_i, TS_i, TS_m \rangle$ to the CS publicly.

Step 2: On getting messages from S_m, CS first checks the time interval i.e. $(TS_{cs} - TS_m < \triangle T^*)$, where $TS_{cs}, \triangle T^*$ are the CS's current timestamp and expected valid time interval for transmission delay respectively. If the verification holds, CS executes the following operations; otherwise, terminates the session.

$$D_i = h(PID_i \parallel x)$$
$$N_i^* = F_i \oplus D_i$$
$$SID_m^* = Z_i \oplus h(D_i \parallel N_i^*)$$
$$G_i^* = h(PID_i \parallel SID_m^* \parallel N_i^* \parallel D_i \parallel TS_i)$$

After that, the CS checks the condition $(G_i^* ?= G_i)$. If $(G_i^* ?= G_i)$, the CS thinks that the U_i is legal; otherwise, terminates the procedures. After that, the CS computes the following operations:

$$BS_m^* = h(PSID_m \parallel y)$$
$$N_m^* = BS_m^* \oplus J_i$$
$$K_i^* = h(BS_m^* \parallel N_m^* \parallel G_i \parallel TS_m)$$

Again, the CS checks the condition $(K_i^* ?= K_i)$. If $(K_i^* ?= K_i)$, the CS thinks that S_m is legal; otherwise, terminates the procedure.

After that, the CS chooses a 128 bit random number N_{cs} and computes the following operations:

$$P_{cs} = N_m \oplus N_{cs} \oplus h(N_i \parallel D_i)$$
$$R_{cs} = N_i \oplus N_{cs} \oplus h(BS_m^* \parallel N_m^*)$$
$$SK_{cs} = h(N_i \oplus N_m \oplus N_{cs})$$
$$Q_{cs} = h((N_m \oplus N_{cs}) \parallel SK_{cs})$$
$$V_{cs} = h((N_i \parallel N_{cs}) \parallel SK_{cs})$$

where SK_{cs} is the secret session key. Finally, the CS sends $\langle P_{cs}, R_{cs}, Q_{cs}, V_{cs} \rangle$ to the S_m for achieving mutual authentication of the protocol through public communication.

Step 3: On getting reply messages from CS, the S_m computes the following operations:

$$W_m = h(BS_m \parallel N_m)$$
$$N_i \oplus N_{cs} = R_{cs} \oplus W_m$$
$$SK_m = h(N_i \oplus N_{cs} \oplus N_m)$$
$$V_{cs}^* = h((N_i \oplus N_{cs}) \parallel SK_m)$$

Then, the S_m checks the condition $(V_{cs}^* ?= V_{cs})$ or not. If $(V_{cs}^* \neq V_{cs})$, terminates the session; otherwise, sends messages $\langle P_{cs}, Q_{cs} \rangle$ to the U_i publicly.

Step 4: On obtaining messages from S_m, the U_i calculates the following operations:

$$L_i = h(N_i \parallel D_i)$$
$$N_m \oplus N_{cs} = P_{cs} \oplus L_i$$
$$SK_i = h(N_m \oplus N_{cs} \oplus N_i)$$
$$Q_{cs}^* = h((N_m \oplus N_{cs}) \parallel SK_i)$$

Then, the U_i checks the condition $(Q_{cs}^* ?= Q_{cs})$ and if $(Q_{cs}^* == Q_{cs})$, it proves the authenticity of S_m and CS. Finally, the proposed protocol achieves mutual authentication among U_i, S_m and CS. Now, the U_i and the S_m can exchange their secret information securely using $SK_m = SK_i$.

5.5. Password Change Phase

Whenever an existing U_i wishes to renew password, first he/she provides ID_i and P_i after punching the smartcard. Then, the CR executes the following operations:

$$b_1^* = DP \oplus h(ID_i^* \parallel P_i^*)$$
$$A_i^* = h(P_i^* \parallel b_1^*)$$
$$b_2^* = bb_i \oplus A_i^*$$
$$PID_i^* = h(ID_i^* \parallel b_2^*),$$
$$C_i^* = h(A_i^* \parallel PID_i^*)$$

The smartcard checks the condition $(C_i^* ? = C_i)$. If $(C_i^* == C_i)$, the card reader requests to enter a new password P_i^{new} to the U_i and calculates the following operations:

$$A_i^{new} = h(P_i^{new} \parallel b_1)$$
$$C_i^{new} = h(A_i^{new} \parallel PID_i^*)$$
$$D_i^= E_i \oplus A_i = h(PID_i^* \parallel x),$$
$$bb_i^* = b_2^* \oplus A_i^{new}$$
$$E_i^{new} = D_i \oplus A_i^{new}$$
$$DP^{new} = h(ID_i \parallel P_i^{new}) \oplus b_1^*$$

.

Finally, the CR substitutes $\langle C_i^{new}, E_i^{new}, bb_i^*, DP^{new} \rangle$ in the place of $\langle C_i, E_i, bb_i, DP^{new} \rangle$ respectively in the smartcard. Thus, a user can renew password without facing any difficulty.

5.6. Identity Update Phase

It is also essential to update the identity of the legal U_i and for updating the identity ID_i, the U_i punches the card into card reader devices and provides old ID_i and P_i. Then, the card reader calculates the following operations:

$$b_1^* = DP \oplus h(ID_i^* \parallel P_i^*)$$
$$A_i^* = h(P_i^* \parallel b_1)$$
$$b_2^* = bb_i \oplus A_i^*$$
$$PID_i^* = h(ID_i^* \parallel b_2^*)$$
$$C_i^* = h(A_i* \parallel PID_i*)$$

The CR checks the condition $(C_i^* ? = C_i)$. If $(C_i^* == C_i)$, the terminal requests to enter a new identity ID_i^{new} to the U_i. Then, the terminal sends $\langle PID_i^{new}, DD_i, PID_i \rangle$ to the CS through insecure channels, where $PID_i^{new} = h(ID_i^{new} \parallel b_2)$, $D_i = E_i \oplus A_i^*$, $DD_i = h(PID_i^{new} \parallel D_i)$. After getting it, the CS computes $D_i^* = h(PID_i \parallel x)$, $DD_i^* = h(PID_i^{new} \parallel D_i^*)$ and checks the correctness $(DD_i^* ? = DD_i)$. If $(DD_i^* == DD_i)$, the CS thinks that U_i's message is authentic and sends $\langle CSD_i, DD_s \rangle$ to the card reader through insecure channel, where $CSD_i = D_i^* \oplus h(PID_i^{new} \parallel x)$, $DD_s = h(h(PID_i^{new} \parallel x) \parallel PID_i^{new}$.

On getting the message, the card reader calculates $h(PID_i^{new} \parallel x) = CSD_i^{new} \oplus D_i$, $DD_s^* = h(h(PID_i^{new} \parallel x) \parallel PID_i^{new})$ and checks the condition $(DD_s^* ? = DD_s)$. If $(DD_s^* == DD_s)$, the card reader further computes $C_i^* = h(A_i \parallel PID_i^{new})$ $D_i^{new} = CSD_i^{new} \oplus D_i$ $E_i^* = D_i^{new} \oplus A_i$ and replaces $\langle C_i^*, E_i^* \rangle$ instead of $\langle C_i, E_i \rangle$ in the smartcard.

6. Cryptanalysis of the Proposed Protocol

This section makes discussion on security analysis of our protocol. For this purpose, we have used BAN logic for proving authentication and AVISPA tool to ensure whether the protocol is safe or not. Further security analysis is also provided to ensure security protection against relevant security attacks.

6.1. *Authentication proof based on* BAN *logic*

Some prelimaries, notations as well as rules for the BAN logic are given in details in [10]. We only present several goals here to proof that our protocol archives mutual authentication feature.

To proof mutual authentication, our protocol must achieves the following goals.

- **Goal 1:** U_i *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
- **Goal 2:** U_i *believes* S_m *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
- **Goal 3:** S_m *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
- **Goal 4:** S_m *believes* U_i *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
- **Goal 5:** S_m *believes* $S_m \overset{SK}{\leftrightarrow} CS$
- **Goal 6:** S_m *believes* CS *believes* $S_m \overset{SK}{\leftrightarrow} CS$
- **Goal 7:** CS *believes* $S_m \overset{SK}{\leftrightarrow} CS$
- **Goal 8:** CS *believes* $S_m \overset{SK}{\leftrightarrow} CS$

6.1.1. *Idealized form*

Message 1: $U_i \rightarrow S_m : PID_i, TS_i, E_i : \langle A_i \rangle_{(D_i)},\ G_i : \langle (PID_i \parallel SID_m \parallel N_i \parallel TS_i) \rangle_{(Di)},\ F_i : \langle N_i \rangle_{(D_i i)},\ Z_i : \langle N_i \rangle_{(D_i)}$.

Message 2: $S_m \rightarrow CS : Message1,\ PSID_m,\ J_i : \langle N_m \rangle_{BS_m},\ K_i : \langle N_m \rangle_{(BS_m \parallel G_i \parallel TS_m)}$.

Message 3: $CS \rightarrow S_m : P_{cs} : \langle N_m \oplus N_{cs} \rangle_{(h(N_i \parallel D_i))},\ R_{cs} : \langle N_i \oplus N_{cs} \rangle_{(h(BS_m \parallel N_m))},\ Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})},\ V_{cs} : \langle N_i \oplus N_{cs} \rangle_{SK_{cs}}$

Message 4: $S_m \rightarrow U_i : P_{cs} :\ \langle N_m \oplus N_{cs} \rangle_{(h(N_i \parallel D_i))},\ Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})}$

Second, the following assumptions about the initial state of the protocol are made to analyze the proposed protocol:

A1: U_i *believes Fresh* (N_i)
A2: S_m *believes Fresh* (N_i)
A3: CS *believes Fresh* (N_i)
A4: S_m *believes Fresh* (N_m)
A5: U_i *believes Fresh* (N_m)
A6: CS *believes Fresh* (N_i)
A7: CS *believes Fresh* (N_{cs})
A8: U_i *believes Fresh* $(N_m \oplus N_{cs})$
A9: S_m *believes Fresh* $(N_i \oplus N_{cs})$
A10: U_i *believes* $U_i \overset{D_i}{\leftrightarrow} S_m$
A11: S_m *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
A12: S_m *believes* $S_m \overset{BS_j}{\leftrightarrow} CS$
A13: CS *believes* $U_i \overset{SK}{\leftrightarrow} S_m$
A14: S_m *believes* U_i *Controls* N_i
A15: CS *believes* S_m *Controls* N_m

6.1.2. *Main proofs using BAN rules and assumptions*

Message 1: $U_i \rightarrow S_m : PID_i, TS_i, E_i : \langle A_i \rangle_{(D_i)}, G_i : \langle (PID_i \parallel SID_m \parallel N_i \parallel TS_i) \rangle_{(D_i)}, F_i : \langle N_i \rangle_{(D_i)}, Z_i : \langle N_i \rangle_{(D_i)}$.

Using seeing rule, we get

S1: S_m *sees* $PID_i, TS_i, E_i : \langle A_i \rangle_{(D_i)}, G_i : \langle (PID_i \parallel SID_m \parallel N_i \parallel TS_i) \rangle_{(D_i)}, F_i : \langle N_i \rangle_{(D_i)}, Z_i : \langle N_i \rangle_{(D_i)}$

Using A11, S1 and message meaning rule, we get

S2: S_m *believes* U_i *said* N_i

Using A2, S2 and freshness-conjuncatenation rule and nonce verification rule is applied, we get

S3: S_m *believes* U_i *believes* N_i, where N_i is the necessary parameter of the session key of the proposed protocol.
Using A14, S3 and the jurisdiction rule is applied, we get
S4: S_m *believes* N_i
Using A2, S3 and the session key rule is applied, we get
S5: S_m *believes* $U_i \overset{SK}{\leftrightarrow} S_m$ **(Goal 3)**
Using A2, S3 and nonce verification rule is applied, we get
S6: S_m *believes* U_i *believes* $U_i \overset{SK}{\leftrightarrow} S_m$ **(Goal 4)**
Message 2: $S_m \rightarrow CS : Message1, PSID_m, J_i : \langle N_j \rangle_{BS_m}, K_i : \langle N_m \rangle_{(BS_m \| G_i \| TS_m)}$.
Using seeing rule, we get
S7: $CS\ sees\ Message\ 1, PSID_m, J_i : \langle N_m \rangle_{BS_m}, K_i : \langle N_m \rangle_{(BS_m \| G_i \| TS_m)}$
Using A13, S7 and the message meaning rule, we get
S8: CS *believes* S_m *said* N_m
Using A6, S7, freshness-conjuncatenation rule and nonce-verification rule, we get
S9: CS *believes* S_m *believes* N_m, where N_m is the necessary session key parameter of the proposed protocol.
Using A15, S9 and jurisdiction rule is applied, we get
S10: CS *believes* N_m
Using A6, S10 and session key rule is applied, we get
S11:CS *believes* $S_m \overset{SK}{\leftrightarrow} CS$ **(Goal 7)**
Using A6, S11 and nonce-verification rule, we get
S12: CS *believes* S_m *believes* $S_m \overset{SK}{\leftrightarrow} CS$ **(Goal 8)**
Message 3: $CS \rightarrow S_m : P_{cs} : \langle N_m \oplus N_{cs} \rangle_{(h(N_i \| D_i))}, R_{cs} : \langle N_i \oplus N_{cs} \rangle_{(h(BS_m \| N_m))}, Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})}, V_{cs} : \langle N_i \oplus N_{cs} \rangle_{SK_{cs}}$
Using seeing rule, we get
S13: $S_m\ sees\ P_{cs} : \langle N_m \oplus N_{cs} \rangle_{(h(N_i \| D_i))}, R_{cs} : \langle N_i \oplus N_{cs} \rangle_{(h(BS_m \| N_m))}, Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})}, V_{cs} : \langle N_i \oplus N_{cs} \rangle_{SK_{cs}}$
Using A9, S13 and message-meaning rule is applied, we get
S14: S_m *believes* CS *said* $(N_i \oplus N_{cs})$
Using A12, S14, freshness-conjuncatenation rule and nonce-verification rule, we get
S15: S_m *believes* CS *believes* $(N_i \oplus N_{cs})$, where $(N_i \oplus N_{cs})$ is the necessary session key parameter of the proposed protocol.
Using A9, S15 and session key rule is applied, we get
S16: S_m *believes* $S_m \overset{SK}{\leftrightarrow} CS$ **(Goal 5)**
Using A9, S16 and nonce-verification rule, we get
S17: S_m *believes* CS *believes* $S_m \overset{SK}{\leftrightarrow} CS$ **(Goal 6)**
Message 4: $S_m \rightarrow U_i : P_{cs} : \langle N_m \oplus N_{cs} \rangle_{(h(N_i \| D_i))}, Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})}$
Using seeing rule, we get
S18: $U_i\ sees\ P_{cs} : \langle N_m \oplus N_{cs} \rangle_{(h(N_i \| D_i))}, Q_{cs} : \langle N_m \oplus N_{cs} \rangle_{(SK_{cs})}$
Using A8, S18 and message-meaning rule is applied, we get
S19: U_i *believes* S_m *said* $(N_m \oplus N_{cs})$
Using A10, S19, freshness-conjunction rule and nonce-verification rule is applied, we get
S20: U_i *believes* S_m *believes* $(N_m \oplus N_{cs})$, where $(N_m \oplus N_{cs})$ is the necessary session key parameter of the proposed protocol.
Using A8, S20 and session key rule is applied, we get
S21: U_i *believes* $U_i \overset{SK}{\leftrightarrow} S_m$ **(Goal 1)**
Using A8, S21 and nonce-verification rule is applied, we get
S22: U_i *believes* S_m *believes* $U_i \overset{SK}{\leftrightarrow} S_m$ **(Goal 2)**

6.2. *Protocol Simulation using* AVISPA *Tool*

This section presents simulation of our protocol using AVISPA software which ensures whether the protocol is protected against security attacks or not. The description and information in details can be found in [37, 38, 10].

```
role alice (Ui, S,Sj : agent,
SK1 : symmetric_key,
SK2 : symmetric_key,
% H is hash function
 H: hash_func, Snd, Rcv: channel(dy))
 played_by Ui
def=
local State : nat,
IDi, SIDj, PIDi, PSIDj, Pi, B1, B2, X, Ai, Ci,
Ei, BBi, BSj, Y, D, Di, Ni, Nj, Ncs, TSi,
TSj, TScs: text,
Gi, Fi, Zi, Li, LLi, Qcs, Vcs, Pcs, Rcs, Ji,
Ki, Wj, WWj, SKi, SKj: message,
Inc : hash_func
const alice_server, server_aserver, aserver_alice,
subs1, subs2, subs3, subs4, subs5,subs6 :
protocol_id
init State :=0
transition
1. State = 0 /\ Rcv(start) =|>
State' := 1 /\ B1' :=new()
/\ B2' := new()
/\ Ai' :=H(Pi.B1')
/\ PIDi' := H(IDi.B2')
/\ BBi' :=xor(B2',Ai')
/\ Snd({Ai'.PIDi}_SK1)
/\ secret({B1',B2',Pi,IDi}, subs1, Ui)
2. State = 1 /\ Rcv({Ci.Ei}_SK1) =|>
State' := 2 /\ Ni' := new()
/\ TSi' := new()
/\ Di' := xor(Ei,Ai)
/\ Gi' := H(PIDi.SIDj.Ni'.TSi'.Di')
/\ Fi' := xor(Di',Ni')
/\ Zi' := xor(SIDj,H(Di'.Ni'))
/\ Snd(Gi'.Fi'.Zi'.PIDi,TSi')
/\ witness(Ui, S, alice_server, Ni')
/\ request(Ui,S,alice_server,Ni')
/\ secret({Ni'}, subs2, {Ui,S,Sj})
3. State = 2 /\ Rcv(Qcs'.Vcs') =|>
State' := 3 /\  Li' := H(Ni.Di)
/\ LLi' := xor(Pcs,Li')
/\ SKi' := H(xor(LLi',Ni))
end role
```

Figure 6. User role in HLPSL

6.2.1. *Brief Specification of the Proposed Protocol*

This section discusses several roles for the U_i, the S, the S_j, the session, the goal and the environment of our protocol. In Fig. 6, we have presented *HLPSL* code for the U_i. In registration phase of user, the U_i generates two random numbers $B1, B2$ using new operation and sends $Snd(Ai'.PIDi\_SK1)$ to the control server CS by utilizing symmetric key $SK1$ and $Snd()$ operation. The symmetric key $SK1$ indicates that the message is transmitted to the server securely. The type declaration channel(dy) means that the channel is for the Dolev-Yao threat model. The information $secret(B1', B2', Pi, IDi, subs1, Ui)$ signifies that the parameters $B1, B2, Pi, IDi$ are only known to the U_i. In the next transition, the U_i receive Ci, Ei parameters securely using Rcv() operation and $SK1$ key. In login phase, the

```
role server (S, Ui, Sj : agent,
SK1 : symmetric_key,
SK2 : symmetric_key,
% H is hash function
H : hash_func,
Snd, Rcv: channel(dy) )
played_by S
def=
local State : nat,
IDi, SIDj, PIDi, PSIDj, Pi, B1, B2, X, Ai, Ci, Ei,
BBi, BSj, Y, D, Di, Ni, Nj, Ncs, TSi, TSj, TScs: text,
Gi, Fi, Zi, Li, LLi, Qcs, Vcs, Pcs, Rcs, Ji, Ki,
Wj, WWj, SKi, SKj, SKcs: message,
Inc : hash_func
const alice_server, server_aserver, aserver_alice,
subs1, subs2, subs3, subs4, subs5, subs6 : protocol_id
init State :=0
transition
1. State = 0 /\ Rcv({Ai.PIDi}_SK1) =|>
State' := 1 /\ Ci' := H(Ai.PIDi)
/\ Di' := H(PIDi.X)
/\ Ei' := xor(Di',Ai)
/\ secret({X}, subs3, {S})
/\ Snd({Ci'.Ei'}_SK1)
2. State =1 /\ Rcv({SIDj'.D'}_SK2) =|>
State' := 2 /\ Y' := new()
/\ PSIDj' := H(SIDj'.D')
/\ BSj' :=H(PSIDj'.Y')
/\ Snd({BSj'}_SK2)
/\ secret({BSj'},subs4,{S,Sj})
3. State = 2 /\ Rcv(Ji.Ki.PSIDj.Gi.Fi.Zi.PIDi.TSi') =|>
State' := 3 /\ Ncs' :=new()
/\ Ni' := xor(Fi,Di)
/\ Nj' := xor(BSj,Ji)
/\ Pcs' := xor(Nj,Ncs',H(Ni.Di))
/\ Rcs' := xor(Ni,Ncs',H(BSj.Nj))
/\ SKcs' := H(xor(Ni,Nj,Ncs'))
/\ Qcs' := H(xor(Nj,Ncs').SKcs)
/\ Vcs' := H(xor(Ni,Ncs').SKcs)
/\ Snd(Pcs.Rcs.Qcs.Vcs)
/\ secret({Ncs'}, subs5, {S,Sj,Ui})
/\ witness(S, Sj, server_aserver, Ncs')
/\ request(S,Sj,server_aserver,Ncs')
end role
```

Figure 7. Server role in HLPSL

U_i produces a random number Ni and a timestamp TSi using new operation and forwards $Snd(Gi'.Fi'.Zi'.PIDi, TSi')$ parameters through open networks. The information $witness(Ui, S, alice\_server, Ni')$ specifies that the Ui has freshly produces the value Ni' for the S and the information $request(Ui, S, alice\_server, Ni')$ specifies that the control server authenticates the Ui. During the authentication phase, the Ui takes delivery of $Rcv(Qcs'.Vcs')$ using Rcv() operation.

In Fig. 7, we have provided *HLPSL* code for the S. During registration phase of Ui, the server receives $Rcv(Ai.PIDi\_SK1)$ securely using the symmetric key $SK1$ and Rcv() operation. Then, the server sends $Snd(Ci'.Ei'\_SK1)$ securely to the Ui. The information $secret(X, subs3, S)$ specifies that the secret information S is only known to the server. during the application server registration phase, the cloud server takes $Rcv(SIDm'.D'\_SK2)$ using another symmetric key $SK2$

and produces a random number Y using new() operation. After that, the server sends $Snd(BSm'\_SK2)$ to the Sm securely using $SK2$. The declaration $secret(BSm', subs4, S, Sm)$ indicates that the parameter BSm is only known to the control and application server (S, Sm). In transition 3, the server receives $Rcv(Ji.Ki.PSIDm.Gi.Fi.Zi.PIDi.TSi')$ and then produces a random number Ncs' using new() operation. The server now sends $Snd(Pcs.Rcs.Qcs.Vcs)$ to the Sm through open networks. The information $witness(S, Sm, server\_aserver, Ncs')$ specifies that the server freshly produced the value Ncs' for the Sm.

```
role aserver (Sj, Ui, S : agent,
SK1 : symmetric_key,
SK2: symmetric_key,
% H is hash function
H : hash_func,
Snd, Rcv: channel(dy) )
played_by Sj
def=
local State : nat,
IDi, SIDj, PIDi, PSIDj, Pi, B1, B2, X, Ai, Ci, Ei, BBi,
BSj, Y, D, Di, Ni, Nj, Ncs, TSi, TSj, TScs: text,
Gi, Fi, Zi, Li, LLi, Qcs, Vcs, Pcs, Rcs, Ji, Ki, Wj,
WWj, SKi, SKj, SKcs: message,
Inc : hash_func
const alice_server, server_aserver, aserver_alice,
subs1, subs2, subs3, subs4, subs5, subs6 : protocol_id
init State :=0
transition
1. State = 0 ∧ Rcv(start) =|>
State' := 1 ∧ SIDj' := new()
∧ D' := new()
∧ Snd({SIDj'.D'}_SK2)
2. State = 1 ∧ Rcv(Gi'.Fi'.Zi'.PIDi.TSi') =|>
State' := 2 ∧ Nj' := new()
∧ TSj' :=new()
∧ Ji' := xor(BSj,Nj')
∧ Ki' := H(Nj'.BSj.Gi.TSj')
∧ Snd (Ji'.Ki'.PSIDj.Gi.Fi.Zi.PIDi.TSi')
∧ secret({Nj'}, subs6, {S,Sj,Ui})
∧ witness(Sj, Ui, aserver_alice, Nj')
∧ request(Sj, Ui, aserver_alice, Nj')
3. State = 2 ∧ Rcv(Pcs.Rcs.Qcs.Vcs) =|>
State' := 3 ∧ Wj' := H(BSj.Nj)
∧ WWj' := xor(Rcs,Wj')
∧ SKj' := H(xor(WWj,Nj))
∧ Snd(Qcs.Vcs)
end role
```

Figure 8. Cloud server role in HLPSL

In Fig. 8, we have provided HLPSL code for cloud server (Sm). During registration phase of cloud server, the Sm generates an identity $SIDm$ and random number D using new operation and sends $Snd(SIDm'.D'\_SK2)$ securely to the S. In transition 2, the Sm receives $Rcv(Gi'.Fi'.Zi'.PIDi.TSi')$ and generates Nm' using the new() operation. The declaration $secret(Nm', subs6, S, Sm, Ui)$ specifies that the Nm' is only known to S, Sm, Ui and the declaration $request(Sm, Ui, aserver\_alice, Nm')$ tells that the Ui authenticates the Sm. In Fig. 9, we have presented the roles for the session, goal and the environment in *HLPSL* language. After execution of AVISPA tool, six secrecy goals and three authentications are verified.

```
role session(Ui, S, Sj: agent,
SK1 : symmetric_key,
SK2: symmetric_key,
H: hash_func)
def=
local SI, SJ, RI, RJ, TI, TJ, PI, PJ: channel (dy)
composition
alice(Ui, S, Sj, SK1, SK2, H, SI, RI)
/\ server(Ui, S, Sj, SK1, SK2, H, SJ, RJ)
/\ aserver(Ui, S, Sj, SK1, SK2, H, TI, TJ)
end role

role environment()
def=
const ui, s, sj: agent,
sk1: symmetric_key,
sk2: symmetric_key,
h: hash_func,
idi, sidj, pidi, psidj, pi, b1, b2, x, ai, ci, ei,
bbi, bsj, y, d, di, ni, nj, ncs, tsi, tsj, tscs,gi,
fi,zi,pcs,rcs,qcs,vcs,ji,ki: text,
alice_server, server_aserver, aserver_alice,
subs1, subs2, subs3, subs4, subs5, subs6 : protocol_id
intruder_knowledge = {ui, s, sj, h,ci,ei,gi,fi,zi,
pidi,pcs,rcs,qcs,vcs,ji,ki}
composition
session( s, sj, ui, sk1, sk2, h)
/\ session(ui, sj, s, sk1, sk2, h)
/\ session(ui, s, sj, sk1, sk2, h)
end role
goal
secrecy_of subs1
secrecy_of subs2
secrecy_of subs3
secrecy_of subs4
secrecy_of subs5
secrecy_of subs6
authentication_on alice_server_ni
authentication_on server_aserver_ncs
authentication_on aserver_alice_nj
end goal
environment()
```

Figure 9. Roles for session, goal and environment in *HLPSL*.

- ⋆ The $secrecy\_of\,subs1$ signifies that the parameters $\langle B1', B2', Pi, IDi\rangle$ are kept private to only (Ui).
- ⋆ The $secrecy\_of\,subs2$ signifies that the random number (Ni) is only familiar to (Ui, S, Sm).
- ⋆ The $secrecy\_of\,subs3$ signifies that the key (X) is only familiar to the (S).
- ⋆ The $secrecy\_of\,subs4$ signifies that the (BSm) is only familiar to the (S, Sm).
- ⋆ The $secrecy\_of\,subs5$ signifies that the password (Ncs') is only familiar to (Ui, S, Sm).
- ⋆ The $secrecy\_of\,subs6$ signifies that the password (Nm') is only familiar to (Ui, S, Sm).

- ⋆ The *authentication_onalice_server_ni* signifies that the (Ui) produces a random number (ni), where (ni) is well known to (Ui) and if the (S) takes message securely, (S) then corroborates the (Ui).
- ⋆ The *authentication_onserver_aserver_ncs* signifies that the (S) produces a random number (ncs), where (ncs) is well known to (S) and if the (Sm) takes message securely, (Sm) then corroborates the (S).
- ⋆ The *authentication_onaserver_alice_nm* signifies that the (Sm) produces a random number (nm), where (nm) is well known to (Sm) and if the (Ui) takes message securely, (Ui) then corroborates the (Sm).

6.2.2. Simulation Results

This section presents simulation results of the AVISPA tool for the *OFMC* and *CL-AtSe* backends. We have simulated HLSPL code for all the entities in the web-based software available in the link "http://www.avispa-project.org/web-interface/basic.php". Note that, the AVISPA software uses the current version i.e. (2006/02/13). The simulation results are safe under the OFMC and CL-AtSe models and presented in Fig. 10 and Fig. 11 respectively. The protocol is safe under both models indicates that it secured against active and passive attacks including replay and man-in-the-middle attacks. Note that, the protocol is secure under some statistical assumptions for OFMC and CL-AtSe mentioned in Fig. 10 and Fig. 11 respectively.

```
% OFMC
% Version of 2016
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileI0OQEb.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 1.07s
visitedNodes: 64 nodes
depth: 6 plies
```

Figure 10. OFMC result

6.3. Further Security Analysis

This section informally described that our protocol is well security protected against relevant security threats.

6.3.1. User Anonymity

In our protocol, the parameter $PID_i = h(ID_i \parallel b_2)$ is used as a user identity instead of the original identity ID_i. It is noted that the parameter PID_i is saved from harm by the two private values $\langle ID_i, b_2 \rangle$ and hash function. Hence, the attacker cannot extort the original ID_i of a legal user. The attacker is not capable of to verify guessed identity using PID_i, as he/she has to guess two different information at a time. If the attacker attempts to guess the ID_i from PID_i, the probability is very less and is approximately $\frac{1}{2^{6n+128}}$. On the other hand, the attacker cannot determine the original

```
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileI0OQEb.if
GOAL
As Specified
BACKEND
CL-AtSe
STATISTICS
Analysed : 0 states
Reachable : 0 states
Translation: 0.30 seconds
Computation: 0.003 seconds
```

Figure 11. CL-AtSe result

identity SID_m of the S_m without knowing secret parameter D_i and random number N_i from $Z_i = SID_m \oplus h(D_i \parallel N_i)$. Hence, the protocol is user anonymous.

6.3.2. Off-line Password Guessing Attack

It is imperative and a mandatory requirement that always user's password must be kept secret. The following description ensures that the attacker cannot guess legal user's password from the protocol description.

(1) We suppose that the attacker knows all smartcard information $\langle C_i, E_i, DP, bb_i, h()\rangle$, where $C_i = h(A_i \parallel PID_i)$, $E_i = D_i \oplus A_i$, $DP = h(ID_i \parallel P_i) \oplus b_1$ and $bb_i = b_2 \oplus A_i$. As the parameter C_i is non-invertible due to hash function, the attacker cannot extort A_i from C_i. The attacker is not capable of to check guessed password using C_i because of two unknown information $\langle P_i, b_1\rangle$.

(2) It is also clear that the attacker cannot derive A_i from E_i or bb_i, as the parameter is protected by the two unknown parameters. Furthermore, the attacker cannot extract P_i from DP due to non-invertible one-way hash function. On knowing the information ID_i and b_1, the attacker can test the guessed password. In the similar way, the attacker cannot verify the guessed password using the login message parameters $\langle G_i, F_i, Z_i\rangle$.

The above explanation claims that the protocol is protected against password.

6.3.3. Privileged Insider Attack

Insider attack is most crucial in cryptography where the insider person disclose some confidential information to the attacker. Though we assume the server as trusted entity, it is better way out to design a protocol where the server should not know user's credential.

In the registration phase, user U_i sends masked password $A_i = h(P_i \parallel b_1)$ instead of original password P_i to the CS . Therefore, the insider person of the CS is not able to determine password P_i from A_i due to non-invertibility property of hash operation. Additionally, the insider person of the CS cannot verify the guessed password due to unknown parameter b_1. Hence, the protocol is protected.

6.3.4. User Impersonation Attack

Our protocol protects the above attack and it's description is given below.

(1) The attacker traps the login request message $\langle E_i, J_i, F_i, Z_i, PID_i, TS_i \rangle$ of the U_i and attempts to calculate a different but identical login message $\langle E_i^{'}, J_i^{'}, F_i^{'}, Z_i^{'}, PID_i, TS_a \rangle$, which will be authenticated to the CS, where TS_a is the attacker's timestamp.

(2) The attacker can impersonate if he knows secret parameters D_i and as it is not known, impersonation is not feasible.

6.3.5. Replay Attack

In this attack, the attacker forwards previous trapped message to the receiver to proof that he is a legal entity. The proposed protocol uses random number and timestamp to generate fresh login and reply messages. Therefore, if the attacker transmits previous intercepted message, the system denies the request because of invalid timestamp condition. Hence, the above attack is protected.

6.3.6. Session key Discloser Attack

The security of the session key in our protocol is the hardness of hash function and secret random nonces $\langle N_i$, N_m, $N_{cs} \rangle$ generated by the U_i, S_m and CS respectively. As the attacker is not able to derive random nonces $\langle N_i$, N_m, $N_{cs} \rangle$ using open information of the protocol, the protocol is completely protected against the above attack.

Table 2. Computation cost and attacks comparison of the proposed scheme with existing related schemes

Schemes ⇒	Yang et al. [27]	Sood et al. [31]	Wang et al. [28]	He et al. [29]	Xue et al. [3]	Li et al. [7]	Proposed
Login Phase	$4T_h$+$1T_e$	$7T_h$	$4T_h$+$2T_{spm}$	$3T_h$+$2T_{spm}$	$3T_h$	$2T_h$	$5T_h$
Authentication Phase	$4T_e$+$4T_h$	$24T_h$	$7T_h$+$4T_{spm}$	$20T_h$+$6T_{spm}$	$24T_h$	$25T_h$	$17T_h$
A1	√	√	×	√	×	×	√
A2	×	×	×	×	×	×	√
A3	√	√	×	√	×	×	√
A4	×	√	×	√	×	×	√
A5	√	×	×	×	√	×	√
Skey	×	×	×	×	√	√	√
MA	×	×	×	√	×	×	√
WPD	×	√	×	√	√	√	√

A1: Resist off-line password guessing attack, A2: Resist Insider attack, A3: User Impersonation Attack, A4: Session key discloser attack, A5: Resist replay attack, Skey: Session key agreement, MA: Satisfy mutual authentication, WPD: Early wrong password detection √: yes, ×: no

7. Performance Study

We compare our protocol's performance with others relevant published protocols such as Xue et al. [3], Yang et al. [27], Sood et al. [31], Wang et al. [28], He et al. [29] and Li et al. [7]. Note that the execution of registration and password change phases happen only once. So we ignore these phases in the comparison table. Besides, Our protocol utilizes mainly hash operation, X-or operation and concatenate operation. It is known information that X-or and concatenate operations are very less computation as compared to other crypto-operations like hash function, exponentiation, integer multiplication, point multiplication, chaotic-maps operations etc.

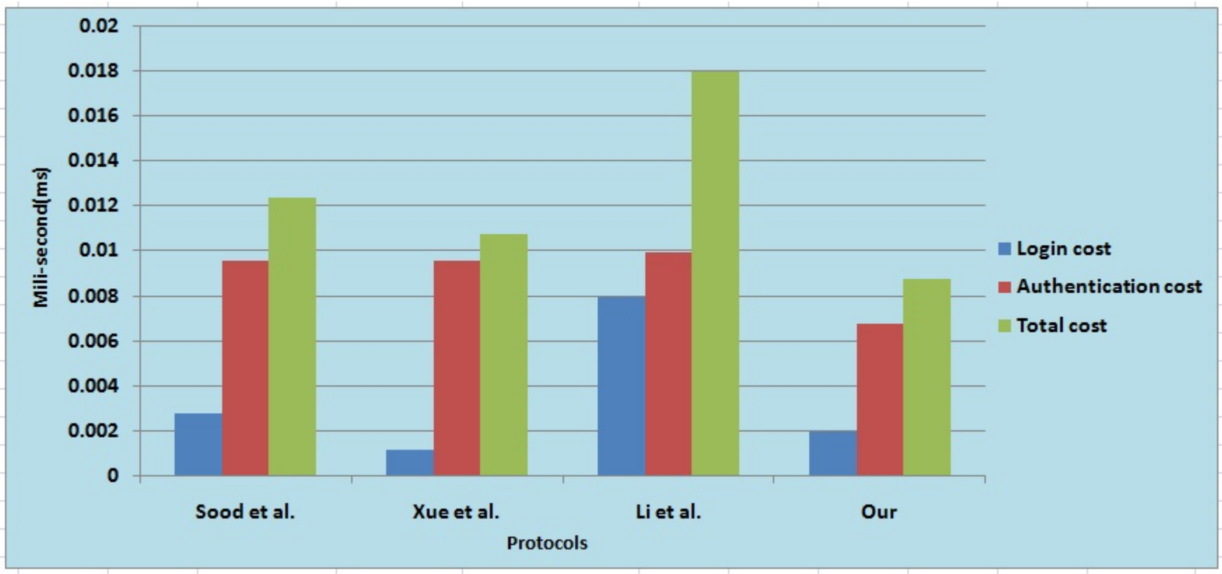

Figure 12. Comparison graph for computation cost

Table 3. Cost complexity comparison of the proposed scheme with existing related schemes

Schemes ⇓	CCL	CCA	Communication mode
Yang et al. [27]	1472	1344	(2) $SC \rightarrow S_j, S_j \rightarrow SC$
Sood et al. [31]	896	1216	(5) $SC \rightarrow S_j, S_j \rightarrow CS, CS \rightarrow S_j, S_j \rightarrow SC, SC \rightarrow S_j$
Wang et al. [28]	320	256	(2) $SC \rightarrow S_j, S_j \rightarrow SC$
He et al. [29]	1408	3584	(5) $SC \rightarrow S_j, S_j \rightarrow CS, CS \rightarrow S_j, S_j \rightarrow SC, SC \rightarrow S_j$
Xue et al. [3]	768	2176	(4) $SC \rightarrow S_j, S_j \rightarrow CS, CS \rightarrow S_j, S_j \rightarrow SC$
Li et al. [7]	512	1664	(4) $SC \rightarrow S_j, S_j \rightarrow CS, CS \rightarrow S_j, S_j \rightarrow SC$
Proposed	768	2048	(4) $SC \rightarrow S_j, S_j \rightarrow CS, CS \rightarrow S_j, S_j \rightarrow SC$

SC: smartcard, S_j: Service provider server, *CS*: Control server, *CCL*: Communication cost in login phase, *CCA*: communication cost in authentication phase

The Table 2 clearly demonstrates that our protocol is efficient than others related existing schemes in terms of computation cost. Therefore, we may claim that the proposed protocol is more light weight than Xue et al.'s protocol. The same table also makes certain that all the security attacks are well protected by our protocol. Hence our protocol is more efficient than protocol in [3].

We have analyzed storage, communication overheads as well as communication mode of our protocol with related works in Table 3. Communication mode in Table 3 states that few schemes cannot hold mutual authentication. For the communication cost analysis, we supposed that the length of the identity (user, server), password, random nonce and message digest takes 128 bits each. The communication cost of our protocol is $(22 \times 128) = 2816$ bits and for the Xue et al. [3] scheme is $(23 \times 128) = 2944$ bits. After achieving all the security requirements and strong security protections, the performance of the proposed protocol is good.

According to the information available in [40], we mentioned some cryptographic operations such as one-way hash function, symmetric key encryption decryption operation and modular exponentiation in mili-second using MIR-

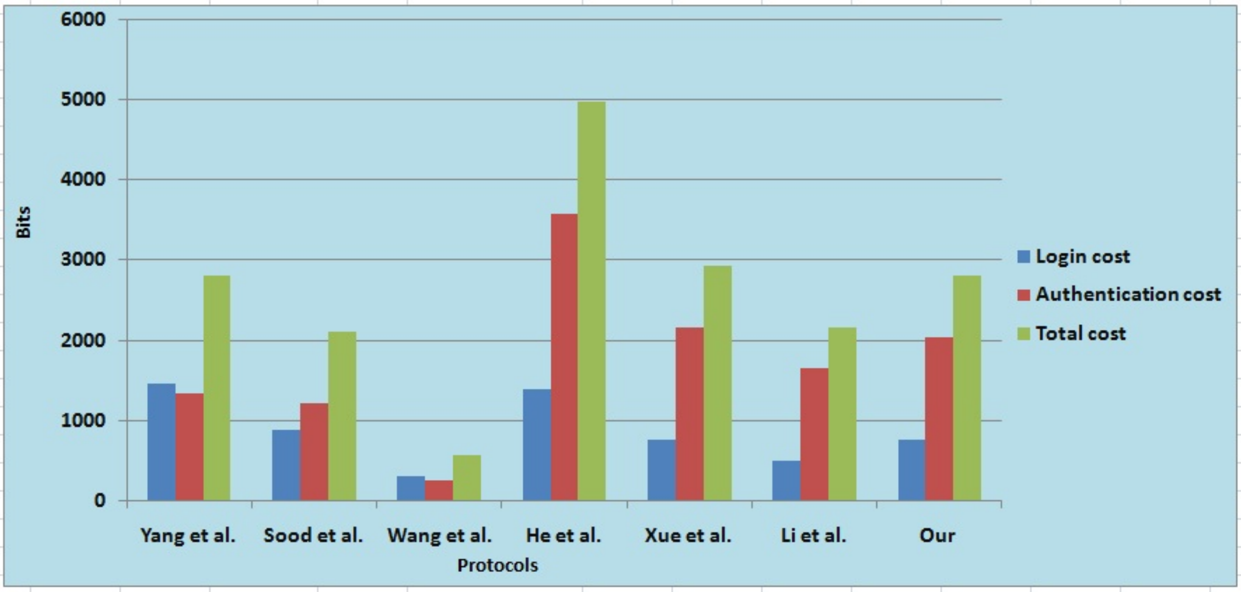

Figure 13. Comparison graph for communication cost

```
query id : bitstring; in j-event (end_UseUi(id))
        ==> inj-event(begin_UserUi(id))
query id : bitstring; in j-event (end_ServerS(id))
        ==> inj-event(begin_ServerS(id))
query id : bitstring; in j-event (end_ApplicationServerAS(id))
        ==> inj-event(begin_ApplicationServerAS(id))
        query attacker(SK)
```

Figure 14. Query of our protocol

ACLE C/C++ library that uses 32-bitWindows 7 operating systems, Visual C++ 2008 Software, AES as symmetric en/decryption technique and SHA-1 as one-way hash function. We separately calculate computation cost in mili-second for the login and authentication phase and the comparison graph for the computation cost is shown in Fig. 12. Similarly, we have also evaluated communication cost in bits and its comparison with other schemes is shown in Fig. 13.

7.1. Pro-Verif Simulation of Our Protocol

Pro-Verif is another important simulation tool to examine security fundamentals such as authentication, secrecy, anonymity and privacy. The description of the Pro-Verif simulation tool can be found in [41, 42, 43]. To examine the security fundamentals, this section only provides some queries and its simulation results. We have mentioned

1. RESULT inj-event(end_serverS(id))
   ==> inj-event(begin_ServerS(id)) is true
2. RESULT inj-event(end_UserUi(id_3117))
   ==> inj-event(begin_UserUi(id_3117)) is true
3. RESULT inj-event(end_ApplicationServerAS(id))
   ==> inj-event(begin_ApplicationServerAS(id)) is true
4. RESULT not attacker (SK[]) is true

Figure 15. Result of Pro-verif simulation

the queries of our protocol in Figure 14 and its simulation results of the Pro-Verif software appears in Figure 15. In Figure 15, Results (1), (2) and (3) make sure that the processes user, application server and server initiated and executed successfully. In addition, Results (4) indicates that the attacker is not able to break session key (SK) of our protocol. Hence, our protocol is secure.

8. Concluding Remarks

In this article, we have described as a contribution that Xue et al.'s and Chuang et al.'s protocols are not protected against numerous security pitfalls. Then, we have designed an architecture for distributed cloud environment where the private cloud stores confidential information using the Internet of Things (IoT) technique. To get secure access of confidential information from any private cloud server of the distributed system, this article designs a standard authentication protocol which resist all kinds of security attacks and provides important features such as user anonymity. Mutual authentication proof has done using *BAN* logic and the protocol simulation using AVSIPA results ensure security safety of the protocol. Furthermore, the informal cryptanalysis of the proposed protocol ensures that the protocol is security attacks protected under hardness assumption of hash function. The performance study of our protocol is better than other works in terms of computation, storage and communication cost. The proposed protocol does not use any password verifier table and gives facility to update password and identity to legal user.

References

[1] ZaslavskyAB, Perera C, Georgakopoulos D. "Sensing as a service and big data. In: International conference on advances in cloud computing (ACC-2012). pp.219. Bangalore, India; July 2012.
[2] V. Chang, Y. -H. Kuo, M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds", Future Generation Computer Systems, Vol. 57, pp. 24-41, 2016.
[3] K. Xue, P. Hong, C. Ma "A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture" *Journal of Computer and System Sciences* 80 (2014) 195-206.
[4] J. Katz, P. MacKenzie, G. Taban, V. Gligor "Two-server password-only authenticated key exchange" *Journal of Computer and System Sciences* 78 (2) (2012) 651-669.
[5] T. Xiang, K. Wong, X. Liao "Cryptanalysis of a password authentication scheme over insecure networks" *Journal of Computer and System Sciences* 74 (5) (2008) 657 - 661.
[6] H.-M, Sun, H. -T. Yeh "Password-based authentication and key distribution protocols with perfect forward secrecy " *Journal of Computer and System Sciences* 72 (6) (2006) 1002 - 1011.
[7] X.Li, Y.P.Xiong, J.Ma, W.D.Wang "An efficient and security dynamic identity based authentication protocol for multi-server architecture using smartcards", *Journal of Network and Computer Applications* 35(2) (2012)763-769.
[8] L. Lamport "Password authentication with insecure communication", *communication of the ACM*, Vol. 24, No. 11, PP. 770-772, 1981.
[9] C.C.Chang, T.C.Wu "Remote password authentication with smartcards", *IEEProc. Computers and Digital Techniques1* 38 (3) (1999)165-168.
[10] R. Amin, G.P. Biswas, A Secure Light Weight Scheme for User Authentication and Key Agreement in Multi-gateway Based Wireless Sensor Networks, Ad Hoc Networks, doi:10.1016/j.adhoc.2015.05.020.
[11] X.Li, W.Qiu, D.Zheng, K.Chen, J.Li "Anonymity enhancement on robust and efficient password authenticated key agreement using smartcards", *IEEE Transactions on Industrial Electronics* 57(2)(2010)793-800.

[12] H.Chien, J.Jan, Y.Tseng "An efficient and practical solution to remote authentication using smartcard", *Computers and Security* 21(4) (2002)372-375.
[13] A.K.Awashti, S.Lal "An enhanced remote user authentication scheme using smartcards", *IEEE Transactions on Industrial Electronics* 50 (2) (2004) 583-586.
[14] J.Xu, W.T.Zhu, D.G.Feng "An improved smartcard based password authentication scheme with provable security", *Computer Standards and Interfaces* 31 (4) (2009)723-728.
[15] Khan, M. K., Kumari, S., and Gupta, M. K., More efficient keyhash based fingerprint remote authentication scheme using mobile device. Computing, vol. 96(9), 793-816 doi:10.1007/s00607-013-0308-2, 2013.
[16] Kumar, M., Gupta, M. K., and Kumari, S., An Improved efficient remote password authentication scheme with smartcard over insecure networks. Int. J. Netw Secur. 13(3):167177, 2011.
[17] Kumari, S., Khan, M. K., and Kumar, R., Cryptanalysis and improvement of A privacy enhanced scheme for telecare medical information systems. J. Med. Syst. 37(4):9952, 2013.
[18] M.Badra, P.Urien "Introducing smartcards to remote authenticates passwords using public key encryption", *in: Proc. Of 2004 IEEE Symposium on Advances in Wiredand Wireless Communications*, NJ, USA, 2004, pp.123-126.
[19] L. Li, I. Lin, M. S. Hwang "A remote password authentication scheme for multi-server architecture using neural networks", *IEEE Transaction on Neural Networks*, vol. 12, pp. 1498-1504, 2001.
[20] I. C. Lin, M. S. Hwang, L. H. Li "A new remote user authentication scheme for multi-server architecture", *Future Generation Computer Systems*, vol. 19, pp. 13-22,2003.
[21] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, vol. 31(4), pp. 469-472, 1985.
[22] X. Cao, S. Zhong, "Breaking a remote user authentication scheme for multi-server architecture", *IEEE Communications Letters*, vol. 10, pp. 580-581, 2006.
[23] W. S. Juang, "Efficient password authenticated key agreement using smartcards", Computers and Security vol. 23, pp. 167-173, 2004.
[24] W. C. Ku, H. M. Chuang, M. H. Chiang, K. T. Chang, "Weaknesses of a multi-server password authenticated key agreement scheme", *In Proceedings of the 2005 national computer Symposium*, vol. 138, pp. 1-5, 2005.
[25] Y. P. Liao, S. S. Wang, "A secure dynamic ID based remote user authentication scheme for multi-server environment", *Computer Standards and Interfaces*, vol. 31, pp. 24-29, 2007.
[26] C. Hsiang, W. K. Shih, "Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment", *Computer Standards and Interfaces*, vol. 31, pp. 1118-1123, 2009.
[27] D. Yang, and B. Yang, "A Biometric Password-based Multi-server Authentication Scheme with smartcard", *International Conference On Computer Design And Appliations (ICCDA)*, vol. 5, pp. 554-559, 2010.
[28] B.Wang, and M. Ma, "A smartcard Based Efficient and Secured Multi-Server Authentication Scheme", *Wireless Personal Communications*, vol. 68, no. 2, pp. 361-378, 2013.
[29] D. He, and S.Wu, "Security Flaws in a smartcard Based Authentication Scheme for Multi-server Environment", *Wireless Personal Communications*, vol. 70, no. 1, pp. 323-329, 2013.
[30] M. -C. Chuang, and M. C. Chen, "An anonymous multi-server authenticated key agreement scheme based on trust computing using smartcards and biometrics", *Expert Systems with Applications*, vol. 41, no. 4(part 1), pp. 1411-1418, 2014.
[31] S. K. Sood, A. K. Sarje, K. Singh, "A secure dynamic identity based authentication protocol for multiserver architecture", *Journal of Network and Computer Applications*, vol. 34, pp. 609-618, 2011.
[32] C. C. Chang, J. S. Lee , "An efficient and secure multi-server password authentication scheme using smartcards", *In Proceedings of the International Conference on Cyber worlds*, pp. 417-422, 2004.
[33] M. Burrows, , M. Abadi, R. Needham: A logic of authentication, *ACM Trans. Comput. Syst.*, 1990, 8, (1), pp. 1836.
[34] J.L. Tsai, T.C Wu, K.Y Tsai: New dynamic ID authentication scheme using smartcards, *Int. J. Commun. Syst.*, 2010, 23, pp. 14491462.
[35] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks, *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541-552, 2002.
[36] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis", *Proceedings of advances in Cryptology*, pp. 388-397, 1999.
[37] Armando, A. et al, "The AVISPA tool for the automated validation of internet security protocols and applications." 17th *International conference on computer aided verification (CAV05). Lecture notes in computer science* (Vol. 3576, pp. 281285). (2005) Springer-Verlag.
[38] D. Dolev, A. Yao, "On the security of public key protocols." *IEEE Transactions on Information Theory*, 29(2), 198208, (1983).
[39] AVISPA. (2014). AVISPA Web Tool. ¡http://www.avispa-project.org/web-interface/ expert.php/¿. Accessed on january 2015.
[40] Xu, L.,Wu, F., Cryptanalysis and Improvement of a User Authentication Scheme Preserving Uniqueness and Anonymity for Connected Health Care, Journal of Medical SystemS, 39: 10, 2015. DOI 10.1007/s10916-014-0179-x
[41] Abadi M, Blanchet B, Comon-Lundh H. Models and proofs of protocol security: A progress report. Computer Aided Verification; 2009:3549.
[42] Chaudhry SA, Farash MS, Naqvi H, Kumari S, Khan MK. An enhanced privacy preserving remote user authentication scheme with provable security. Security and Communication Networks. 2015;8(18):37823795.
[43] Maitra, T., Obaidat, M. S., Amin, R., Islam, S. H., Chaudhry, S. A., and Giri, D. (2016), A robust ElGamal based password authentication protocol using smart card for client-server communication, Int J Commun Syst, doi:10.1002/dac.3242

***Biographies (Photograph)**

Ruhul Amin received his B.Tech and M.Tech from West Bengal University of Technology in Computer Science and Engineering in 2009 and 2013, respectively. He has earned doctoral degree (PhD) in Computer Science & Engineering from Indian School of mines, Dhanbad, India. Currently, he is working as a lecturer in the Department of Computer Science & Engineering, Thapar University, Punjab, India. He has published many research papers in Journals and Conference proceedings of International reputes. He also served as reviewer in many international journals. His current research interests include cryptographic authentication protocol and security in wireless sensor network.

G. P. Biswas received B.Sc (Engg.) and M.Sc (Engg.) degrees in Electrical & Electronics Engineering and Computer Science & Engineering, respectively. He completed his PhD degree in Computer Science & Engineering from Indian Institute of Technology, Kharagpur, India. He is currently working as a Professor in the Department of Computer Science & Engineering, Indian school of Mines, Dhanbad, Jharkhand, India. He has around 20 years of teaching and research experiences, and published more than 100 research articles in Journals, Conferences and Seminar Proceedings. His main research interests include Cryptography, Computer Network and Security, Cellular Automata, VLSI Design.

Neeraj Kumar received his Ph.D. in CSE from Shri Mata Vaishno Devi University, Katra, India. He is now an Associate Professor in the Department of Computer Science and Engineering, Thapar University, Patiala, Punjab (India). He is a member of IEEE. His research is focused on mobile computing, parallel/distributed computing, multi-agent systems, service oriented computing, routing and security issues in mobile ad hoc, sensor and mesh networks. He has more than 100 technical research papers in leading journals such as-IEEE TII, IEEE TIE, IEEE TDSC, IEEE ITS, IEEE TWPS, IEEE SJ,IEEE ComMag, IEEE WCMag, IEEE NetMag and conferences. His research is supported from DST, TCS and UGC. He has guided many students leading to M.E. and Ph.D.

Dr Rahat Iqbal is a Reader/Associate Professor in the Faculty of Engineering, Environment and Computing at Coventry University. He has a track record of project management and leadership of industrial projects funded by EPSRC, TSB, ERDF and local industries (e.g. Jaguar Land Rover Ltd, Trinity Expert Systems Ltd). He was involved in the project management and development of the EU FP7 project CHIL (Computers in Human Interaction Loop) at the Technical University of Eindhoven , Netherlands. Recently, he has successfully led a project in collaboration with Jaguar Land Rover on self-learning car for predicting driver's behaviour for personalisation of telematics and optimisation of route planning. He has managed many industrial projects, in Intelligent Systems, Predictive Modelling, User Behaviour, Information Retrieval and Fault Detection. He has published more than 100 papers in peer-reviewed journals and reputable conferences and workshops. Dr Iqbal is on the programme committee of several international conferences and workshops. He is also a fellow of the UK Higher Education Academy (HEA). Dr Iqbal has also edited several special issues of international journals within the field of Information Retrieval and User Supportive systems.

Dr. Victor Chang is an Associate Professor in Information Management and Information Systems at International Business School Suzhou (IBSS), Xi'an Jiaotong Liverpool University, China. He is a Director of PhD Program and the 2016 European and Cloud Identity winner of "Best Project in Research". Victor Chang was a Senior Lecturer in the School of Computing, Creative Technologies at Leeds Beckett University, UK and a visiting Researcher at the University of Southampton, UK. He is an expert on Cloud Computing and Big Data in both academia and industry with extensive experience in related areas since 1998. He completed a PGCert (Higher Education) and PhD (Computer Science) within four years while working full-time. He has over 100 peer-reviewed published papers. He won £20,000 funding in 2001 and £81,000 funding in 2009. He was involved in part of the £6.5 million project in 2004, part of the £5.6 million project in 2006 and part of a £300,000 project in 2013. He won a 2011 European Identity Award in Cloud Migration and 2016 award. He was selected to present his research in the House of Commons in 2011 and won the best papers in 2012 and 2015. He has demonstrated ten different services in Cloud Computing and Big Data services in both of his practitioner and academic experience. His proposed frameworks have been adopted by several organizations. He is the founding chair of international workshops in Emerging Software as a Service and Analytics and Enterprise Security. He is a joint Editor-in-Chief (EIC) in International Journal of Organizational and Collective Intelligence and a founding EIC in Open Journal of Big Data. He is the Editor of a highly prestigious journal, Future Generation Computer Systems (FGCS). His security paper is the most popular paper in IEEE Transactions in Services Computing and his FGCS paper has one of the fastest citation rate. He is a reviewer of numerous well-known journals and had published three books on Cloud Computing which are available on Amazon website. He is a keynote speaker for CLOSER 2015/WEBIST2015/ICTforAgeingWell 2015 and has received positive support. He is the founding chair of IoTBD 2016 www.iotbd.org and COMPLEXIS 2016 www.complexis.org conferences.