Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 4, No. 1, April 2018.

AN EFFICIENT TWO-SERVER AUTHENTICATION AND KEY EXCHANGE PROTOCOL FOR ACCESSING SECURE CLOUD SERVICES

Durbadal Chattaraj[1], Monalisa Sarma[1] and Debasis Samanta[2]

(Received: 14-Dec.-2017, Revised: 13-Feb.-2018, Accepted: 28-Feb.-2018)

ABSTRACT

To avail cloud services; namely, Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), ...etc. via insecure channel, it is necessary to establish a symmetric key between end user and remote Cloud Service Server (CSS). In such a provision, both the end parties demand proper auditing so that resources are legitimately used and privacies are maintained. To achieve this, there is a need for a robust authentication mechanism. Towards the solution, a number of single server authenticated key agreement protocols have been reported recently. However, they are vulnerable to many security threats, such as identity compromization, impersonation, man-in-the-middle, replay, byzantine, offline dictionary and privileged-insider attacks. In addition to this, most of the existing protocols adopt the single server-based authentication strategy, which are prone to single point of vulnerability and single point of failure issues. This work proposes an efficient password-based two-server authentication and key exchange protocol addressing the major limitations in the existing protocols. The formal verification of the proposed protocol using Automated Validation of Internet Security Protocols and Applications (AVISPA) proofs that it is provably secure. The informal security analysis substantiates that the proposed scheme has successfully addressed the existing issues. The performance study contemplates that the overhead of the protocol is reasonable and comparable with those of other schemes. The proposed protocol can be considered as a robust authentication protocol for a secure access to the cloud services.

KEYWORDS

Key agreement, Authentication protocol, User privacy, Cloud data security, Privacy-preserving protocol.

1. INTRODUCTION

With the exponential growth of Cloud service (e.g., SaaS, IaaS, PaaS) accessibility *via* Internet applications, it has been predicted that the annual global data traffic will reach 20.6 Zettabytes (ZB) per annum (approximately 1.7 ZB per month) by the end of 2021, which is approximately three times faster than 6.8 ZB per year (568 Exabytes) in 2016. As a result, the global data center IP traffic will grow 3-fold over the next 5 years. More precisely, data center IP traffic will grow at a Compound Annual Growth Rate (CAGR) of 25 percent from 2016 to 2021[1]. The report indicates a huge success of the cloud technology, but there is a challenge too[2]. In the cloud, a client remotely accesses his service provided by a service provider [52]. This leads to opening up a security problem as the communication takes place over insecure channel for accessing services. Towards this solution, a number of single server-based authentication and key agreement protocols are reported in the recent literature [48]-[51], [53]-[59].

[1] Cisco Global Cloud Index: Forecast and Methodology, 2016-2021 White Paper, [Online], available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[2] Parts of the paper published in ICTCS-2017 and SIN-2017.

This paper is an extended version of two conference papers. The first conference paper entitled "An Efficient Two-Server Authentication and Key Exchange Protocol", appeared in the proceedings of the International Conference "New Trends in Computing Sciences (ICTCS) 2017", 11-13 October 2017, pp. 127-132, Amman, Jordan. The second conference paper entitled "Privacy Preserving Two-Server Diffie-Hellman Key Exchange Protocol" was presented as a student paper at the 10th International Conference on "Security of Information and Networks (SIN) 2017", 13-15 October 2017, ACM SIGSEC, (DOI: 10.1145/3136825.3136871).

1. D. Chattaraj and M. Sarma are with Subir Chowdhury School of Quality and Reliability, Indian Institute of Technology Kharagpur, Kharagpur, India. Emails: dchattaraj@iitkgp.ac.in and monalisa@iitkgp.ac.in.
2. D. Samanta is with Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India. Email: dsamanta@sit.iitkgp.ernet.in.

The existing state-of-the-art authentication protocols [48]-[51], [53]-[59] provide security in the cloud environment and they follow a single server-based authentication mechanism. In order to achieve mutual authentication, all clients must interact with a particular authentication server. The single authentication server stores the credentials of all the clients in its database. Thus, this server is fully reachable for public access and is also vulnerable to a number of attacks, including dictionary, impersonation, password guessing, identity compromization and stolen-verifier attacks [52]. To mitigate these attacks, several schemes have been proposed, which are based on smart card, biometric, RFID (Radio-Frequency Identification) tag-based authentication approach. However, these approaches are economically expensive in terms of extra hardware cost [52].

In addition to this, to ensure a secure communication between the end user and the remote service server, several password-based two-server Diffie-Hellman key exchange protocols [6], [1]-[5], [7], [9], [11] are also reported in the literature. It may be noted that these protocols are not directly associated with the cloud computing domain, but these protocols can be incorporated in the same domain for accessing secure cloud services through web. Since its inception, in [6], both the intended sender and receiver establish a secret key between themselves based on previously shared common domain parameters (also called public parameters). However, this protocol does not have any mechanism to accomplish a mutual authentication between sender and receiver. As a result, an eavesdropper can easily make a man-in-the-middle attack by proxifying himself as a legitimate entity between both sender and receiver. Moreover, identity compromization attacks, impersonation attacks, replay attacks, privileged-insider attacks and offline dictionary attacks are the key security threats that are not properly addressed [10]. To avert these problems, several protocols are reported recently in the literature [24], [26]-[28], [45]-[47]. In these protocols, both parties should verify their identity before the shared secret symmetric key is settled between them. However, these existing state-of-the-art approaches do not properly address the server-side user privacy, protection from malicious insiders (i.e., byzantine attacks), single point of failure and single point of vulnerability issues [8]. In addition to this, most of the existing approaches utilize a single server-based authentication strategy, which is having the single point of failure and single point of vulnerability issues [52]. Note that in lieu of the existing extensive state-of-the-art single server-based authentication and key agreement protocol stack available for the cloud environment, in this paper, we focus only on the limitations and issues related to the password-based two-server Diffie-Hellman authenticated key agreement schemes, which could be incorporated for the same environment and finalize the following objectives.

1.1 Motivation and Research Objectives

To avert the aforesaid discussed issues and limitations of the existing password-based two-server authenticated key agreement protocols, we set the following objectives in the proposed scheme:

1. The proposed scheme should overcome several known attacks, such as password guessing, stolen-verifier, replay, man-in-the-middle, privileged-insider, impersonation, offline dictionary and identity compromization attacks.
2. The proposed scheme should provide a better server-side security and user privacy.
3. The user authentication process in the proposed scheme should be more robust and user-friendly.
4. The proposed scheme should distribute securely the session key between the entities.
5. The proposed scheme should mitigate the existing server-side single point of failure (SOF) and single point of vulnerability (SOV) issues.

The proposed approach *vis-a-vis* the above-mentioned objectives is as follows. An efficient password-based two-server authentication scheme has been proposed. This dual server model is planned in such a way that it is resilient against existing vulnerabilities; namely, man-in-the-middle, offline dictionary, byzantine, server identity compromization, client identity compromization, password guessing, stolen-verifier, privileged-insider and replay attacks. In this two-server model, the server, which is at the front-end, is responsible for interfacing with a client only, while the other server at the back-end accomplishes the authentication task. As the back-end server is hidden from public exposure, it ensures the server-side security by minimizing the risk of both SOF and SOV issues. As a solution to the impersonation attacks, we propose a public key infrastructure (PKI)-enabled password-based two-server Diffie-Hellman authenticated key exchange scheme. To preserve user privacy and support anonymity, an Elliptic Curve Cryptography (ECC)-based digital signature generation and verification strategy has been adopted in the proposed scheme. The research contributions of the proposed approach are as follows.

1.2 Research Contributions

Following are the major research contributions as a realization of the proposed approach.

- We propose an approach to distribute the session keys without time synchronization.
- We introduce a new digital signature-based verification strategy by which each entity would be able to substantiate the other entity along with the issuer of the security credentials (i.e., session key) on which both the entities rely.
- To distribute the session key securely, we propose a pair-wise session key distribution approach using the concept of server-side in memory caching.
- The informal security analysis has been carried out to make evident that the proposed scheme can protect several known active as well as passive attacks.
- The formal security verification using the broadly-accepted AVISPA tool is carried out for the proposed scheme and the simulation results ensure that the proposed scheme is also secure.
- The suggested protocol alleviates the existing SOF and SOV issues by adopting a new dual server-based user authentication strategy.
- To ensure user friendliness of the proposed approach, a user only needs to enter his identity and password in the system for authenticated key agreement process.
- To preserve user privacy during authentication process, the user remains anonymous even if an adversary is eavesdropping the communication messages between user and remote cloud service server.

1.3 Organization of the Paper

The rest of the paper is organized as follows. Section 2 discusses the recent literature related to the work. The basic knowledge throughout the paper is presented in Section 3. Section 4 discusses the proposed protocol. The formal verification of the proposed protocol using AVISPA tool is described in Section 5. Section 6 presents the informal security analysis of the proposed protocol. The performance analyses of the proposed protocol are discussed in Section 7. Finally, Section 8 concludes the paper.

2. Related Works

In this section, we brief about the existing single server-based authenticated key exchange protocols, its issues and challenges.

Recently, two well-known Authenticated Key Agreement Protocol (AKAP) families are widely used in cloud industries. In the first category, several symmetric key-based single server AKAP are reported in the literature, such as, Kerberos, PKINIT, IDFusion, Sesame, …etc. [52]. Second category conveys the asymmetric key-based single server AKAP which are based on Diffie-Hellman key exchange protocol [25] and its variants reported in [24], [26]-[28]. Keeping the eye on the above fact, the literature review of our work is three-fold. First, we discuss the existing known security issues and challenges for the aforesaid two AKAP families and then elaborate the recent issues reported in the area of cloud computing platform by alleviating the state-of-the-art security threats of the existing schemes as follows:

2.1 Issues in Symmetric Key-based Single Server AKAP

In order to access cloud services (e. g. SaaS, PaaS, IaaS, …etc.) over the Internet, it is necessary for a user to enrol himself with the Cloud Service Provider (CSP). After enrolment, the end user can access cloud services remotely over the Web. Usually, according to the symmetric key based single server AKAP scheme, the CSP stores the secret information in the Key Distribution Centre (KDC), where a single point of compromization makes the whole system jeopardized and it is also vulnerable to on/offline dictionary attacks. For example, existing approaches [29], [31]-[32], [51] enrol an end user by asking his "username" and password. This username is used as the primary credential, which is verified at the time of user authentication. In fact, selecting a "username" is not enough to be considered as a strong private entity. As a result, an adversary can easily incorporate different attacks, such as impersonation attacks and identity compromization attacks by sniffing the "username" from the insecure media. Moreover, the existing password-based enrolment strategy is vulnerable to password guessing (on/offline dictionary) attacks and stolen-verifier attacks. Additionally, the existing approaches [29]-[30] derive the client's secret key as the hash value of his password. Therefore, the key will remain the same until client changes the current password. However, changing this password needs updating in enrolled data maintained by the KDC and this, in fact, invites many key rollover problems [1]. According to the current practice, a user makes an authentication request to an authentication server (AS) by means of a plain text containing "username" [29]. In this context, an attacker can eavesdrop the "username" and later expose himself to the AS as a legitimate user. In other words, an attacker can easily determine from the transmitted message which users are currently online. In this situation, an attacker has scope to make man-in-the-middle attacks as well as replay attacks [44]. Further, an eavesdropper can make identity compromization attacks and impersonation attacks by stealing the "username" if the channel is insecure [43], [42]. Moreover, the AS issues an Authentication Ticket (AT) to an end user after verifying only "username" without verifying its password or other security credentials [43]. However, as "username" is not a confidential credential, there is an opportunity for an attacker to get multiple authentication tickets by simply sending a "username" to the AS. As a consequence, a cryptanalyst can decrypt the ciphertexts (i.e., ATs) using some knowledge about underlying user's password. Thus, this scheme is vulnerable to Ciphertext-only Attacks (COAs). In addition to this, the aforesaid discussed protocol is vulnerable to different other known security threats, including SOF and SOA issues reported in [52].

2.2 Issues in Asymmetric Key-based Single Server AKAP

Intuitively, according to the well-known Diffie-Hellman key exchange protocol [25], both the intended sender and receiver establish a secret key between themselves based on previously shared common domain parameters (also called public parameters). However, this protocol does not have any mechanism to accomplish a mutual authentication between sender and receiver. As a result, an eavesdropper can easily make a man-in-the-middle attacks by proxifying himself as a legitimate entity between both sender and receiver. In addition to this, identity compromization attacks, impersonation attacks and replay attacks are the key security threats that are not properly addressed. To avert these problems, several protocols are reported in the literature [24], [26]-[28], [45]-[47]. In these protocols, both parties should verify their identity before the shared secret symmetric key is settled between them. However, these existing state-of-the-art approaches do not properly address the server-side user privacy, protection from malicious insiders (i.e., byzantine attacks), single point of failure and single point of vulnerability issues [8].

2.3 Issues in Recent Single Server AKAP for Cloud Platform

Yang et al. [33] proposed an authentication scheme in a cloud environment setting. However, Chen et al. [34] pointed out the security pitfalls in Yang et al.'s scheme [33] that it is vulnerable to insider and impersonation attacks. To withstand these security loopholes in Yang et al.'s scheme, Chen et al. then designed a dynamic ID-based authentication scheme for cloud computing environment, which is based on the elliptic curve cryptography (ECC). Wang et al. [35] reviewed Chen et al.'s scheme [34] and proved that their scheme is vulnerable to offline password guessing as well as impersonation attacks. In addition, it was found that Chen et al.'s scheme does not provide user anonymity and also has a clock synchronization problem. Later, Hao et al. [36] presented a time-bound ticket-based mutual authentication scheme for cloud computing. The purpose of using the time bound tickets is to reduce the server's processing overhead. Unfortunately, Jaidhar [37] identified that Hao et al.'s scheme [36] is insecure against denial-of-service attack during the password change phase. Wazid et al. [38] also proposed a provably secure user authentication and key agreement scheme for cloud computing environment. Their scheme withstands the weaknesses of the existing schemes and also supports extra functionality features, such as user anonymity, efficient password and biometric update phase in multi-server environment. Recently, Gope and Das [39] proposed an anonymous mutual authentication scheme for ubiquitous mobile cloud computing services, in which a legitimate mobile cloud user is allowed to enjoy n times all the ubiquitous services in a secure and efficient way, where the value of n may differ based on the principal he/she has paid for. In addition, Odelu et al. [40] reviewed Tsai-Lo's scheme [41] and pointed out that their scheme does not provide the session-key security and strong user credentials' privacy. To remove the security weaknesses found in Tsai-Lo's scheme, Odelu et al. designed a provably secure authentication scheme for distributed mobile cloud computing services. Since its inception, S. Kumari et al. [50] proposed a provably secure biometrics-based multi-cloud-server authentication scheme for accessing secure cloud services *via* insecure channel. In this scheme, authors have used a biometrics-based authentication scheme. M. H. Ibrahim et al. [51] proposed an attribute-based authentication protocol on the cloud for thin clients. In this scheme, authors have introduced two new authentication schemes for resource constraint client in cloud environment which support private attribute-based access to remote cloud servers. In this work, authors claimed that unlike existing attribute-based encryption and signature schemes, their scheme requires only a little amount of elliptic curve bilinear pairings and modular exponentiations. In 2015, Kalra and Sood [53] reported an authentication scheme to connect resource-constrained devices (tiny devices) to the cloud server using Elliptic Curve Cryptography (ECC) for Internet of Things (IoT). However, in 2017, S. Kumari et al. [48] have shown that Kalra and Sood's scheme [53] is vulnerable to offline password guessing and privileged insider attacks and does not achieve device anonymity, session key agreement and mutual authentication. To mitigate these vulnerabilities, S. Kumari et al. [48] proposed a secure authentication scheme based on ECC for IoT and cloud. In this scheme, authors have claimed that their scheme achieves all security requirements and is resistant to various known attacks as compared to the Kalra and Sood's scheme. In 2017, Wu et al. [49] proposed a lightweight and anonymous RFID tag authentication protocol with cloud assistance for e-healthcare application. Although these existing state-of-the-art approaches address the server-side user privacy, user anonymity, protection from malicious insiders and other known attacks, most of the schemes utilize extra hardware devices like biometric scanner, RFID-tag, smart card, …etc. and do not resolve the existing ciphertext-only attacks, single point of failure issues and single point of vulnerability issues.

Intuitively, to alleviate several known attacks and issues existing for the evolving cloud computing paradigm of the aforesaid extensive literature, in this work, we convey a new and efficient two server-based authenticated key agreement protocol. More specifically, in this paper, we address security threats, like identity compromization, server-side impersonation, offline dictionary, privileged-insider, man-in-the-middle, replay, byzantine, password guessing, stolen-verifier and COA attacks, as well as two important issues; namely, SOF and SOV issues, respectively. The basic mathematical knowledge required to understand the proposed protocol is discussed as follows.

3. PRELIMINARIES

In this section, we brief the basic security knowledge throughout the paper. In this regard, we discuss elliptic curve cryptography and its two underlying security assumptions; namely, Elliptic Curve Decisional Diffie-Hellman Problem (ECCDHP) and Elliptic Curve Discrete Logarithm Problem (ECDLP), as well as one-way cryptographic hash function as follows.

3.1 Elliptic Curve

Suppose $m, n \in Z_p$, where $Z_p = \{0,1, \cdots p-1\}$ and $p > 3$ is a prime. A non-singular elliptic curve $y^2 = x^3 + mx + n$ over the finite field Z_p is the set $E_p(m,n)$ of solutions $(x,y) \in Z_p \times Z_p$ to the congruence $y^2 \equiv x^3 + mx + n$ (mod p), where $m, n \in Z_p$ such that $4m^3 + 27n^2 \neq 0$ *(mod p)* and a point at infinity or zero point O. Note that $4m^3 + 27n^2 \neq 0$ *(mod p)* is a necessary and sufficient condition to ensure a non-singular solution for the equation $x^3 + mx + n = 0$ [18]. $4m^3 + 27n^2 = 0$ *(mod p)* implies that the elliptic curve is singular [14].

The algebraic formulae for the sum of two same or different points on $y^2 \equiv x^3 + mx + n$ (mod p) are as follows:

1) *$R + O = O + R = R$ for all $R \in E_p(m,n)$.*
2) *If $R = (x, y) \in E_p(m,n)$, then $(x, y) + (x, -y) = O$. (The point $(x, -y)$ is denoted by $-R$).*

3) *Let $R \neq S$, where $R = (x_1, y_1) \in E_p(m,n)$ and $S = (x_2, y_2) \in E_p(m,n)$ with $x_1 \neq x2$, where $R \neq \pm S$. Then $R + S = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2 \ (mod\ p)$ and $y_3 = \lambda(x_1 - x_3) - y_1 \ (mod\ p)$. Here, $\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)$.*
4) *If $x_1 = x_2$, but $y_1 \neq y_2$, then $R_1 + R_2 = O$.*
5) *Let $R = (x_1, y_1) \in E_p(m,n)$ with $y_1 \neq 0$, where $R \neq -R$. Then $2 \cdot R = (x_3, y_3)$, where $x_3 = \lambda^2 - 2 \cdot x_1 \ (mod\ p)$ and $y_3 = \lambda \cdot (x_1 - x_3) - y_1 \ (mod\ p)$. Here, $\lambda = \left(\frac{3 \cdot x_1^2 + m}{2 \cdot y_1}\right)$.*
6) *Let $R = (x_1, y_1) \in E_p(m,n)$ with $y_1 = 0$. Then, $R_1 + R_2 = O$.*

Hasse's theorem states that the number of points on $E_p(m,n)$, denoted as $\#E$, satisfies the following inequality [19]:

$$p + 1 - 2\sqrt{p} \leq p + 1 + 2\sqrt{p} \ .$$

In other words, there are about p points on an elliptic curve $E_p(m,n)$ over Z_p. Also, $E_p(m,n)$ forms a commutative or an abelian group under addition modulo p operation.

Definition 1 (Elliptic Curve Discrete Logarithm Problem). *Given an elliptic curve $E_p(m,n)$ and two points $R, S \in E_p(m,n)$, find an integer x such that $S = x \cdot R$.*

Definition 2 (Elliptic Curve Decisional Diffie-Hellman Problem). *Given a point R on an elliptic curve $\in E_p(m,n)$ and two other points $x \cdot R, y \cdot R \in E_p(m,n)$, find $x \cdot y \cdot R$.*

3.2 One-way Hash Function

A one-way hash function $h: \{0,1\}^* \rightarrow \{0,1\}^l$ takes a binary string of variable length input, say $x \in \{0,1\}^*$ and outputs a binary string $h(x) \in \{0,1\}^l$ as an output of fixed length, say l bits. The formal definition of $h(\cdot)$ is provided as follows [16].

Definition 3 (Collision-resistant one-way hash function). *If an adversary A's advantage in finding collision in hash outputs with the execution time t is denoted by $Adv_A^{HASH}(t)$, it is defined by $Adv_A^{HASH}(t) = \Pr[(x,y) \leftarrow_R A: x \neq y \text{ and } h(x) = h(y)]$, where $\Pr[E]$ is the probability of an event E and $(x,y) \leftarrow_R A$ means that the pair (x, y) is randomly chosen by A. By an (η, t)- adversary A attacking the collision resistance of h(.), it indicates that the execution time of A is at most t and that $Adv_A^{HASH}(t) \leq \eta$.*

Examples of a one-way hash function include the Secure Hash Standard (SHA-1) hashing algorithm and the SHA-256 hashing algorithm [17].

4. PROPOSED PROTOCOL

The system architecture of our proposed protocol is shown in Figure 1. In order to demonstrate our proposed protocol, we break it into four parts: (1) System model. It tells about the system architecture, different entities involvement and individual knowledge about the initial security parameters. (2) Adversary model. It conveys the capabilities of an external entity or a malicious insider to make attacks into the security system. (3) Registration. It discusses the enrolment strategy of different entities into a trusted third-party server. (4) Authenticated key exchange. It conveys the shared session key establishment process between two distinct entities through message passing.

In our discussion, we frequently refer to some notations and symbols. All notations and symbols with their annotations are listed in Table 1.

4.1 System Model

Three types of entities are involved into our system; namely, client (C), service server (SS), back-end server (BS), where SS is the public server in two-server model and BS is the private server. The public server is reachable to everyone, whereas the private server works in the background and is controlled internally by the administrator only. C and SS both enrol themselves with BS during registration phase, but the authenticated key exchange task is carried out by both SS and BS, respectively.

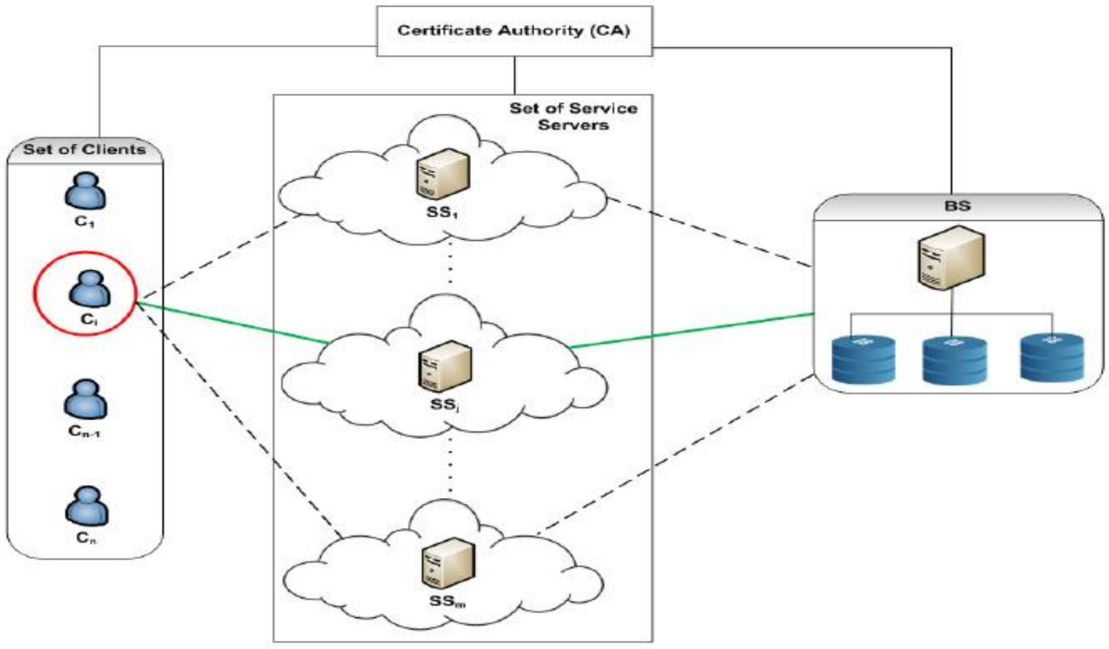

Figure 1. System architecture of the proposed protocol.

Further, there exists a certificate authority (CA), which issues a certificate C_P for each principal P. The CA selects a generator G on the elliptic curve $E_p(m,n)$ of order k and also selects two hash functions $H_1(\cdot)$ and $H_2(\cdot)$. Further, the principal P can randomly select $s_P \in \mathbb{Z}_k^*$ as secret key and computes the corresponding public key as $Q_P = s_P \cdot G$. Suppose that there exists an application B_{app} provided by BS with which the principal P can transform its secret information (i.e., password and server secret identity) offline (i.e., through a smart phone application). P registers these security credentials (after transformation) into BS *via* out-of-band channel [8]. It may be noted that the enrolments of SS are carried out by individual server administrator. At the time of authenticated key exchange, C or SS verifies the legitimacy of both SS and BS or both C and BS, respectively, without disclosing its secret credentials over the insecure channel. In addition to this, an application A_i is running into client workstation in order to access the service servers.

4.2 Adversary Model

Our adversary model is as follows: C and SS are controlled by an active adversary and BS is controlled by a passive adversary in terms of different attacks, such as offline dictionary, replay, man-in-the-middle, byzantine, identity compromization, impersonation and privileged-insider attacks. The active adversary can behave arbitrarily in order to make the above discussed attacks feasible. In contrast, we also consider the outside intruder as an active adversary. In [8], a passive adversary pursues an honest-but-curious activity; that is, it honestly executes the protocol according to the specification and does not modify any data in server's secure database. But, this adversary listens the communication channels to derive the security credentials between C and SS. Further, the passive adversary also knows all the shared secret keys and the internal state of the server.

Here, BS is trusted for both C and SS. We follow the well-accepted Dolev-Yao threat model (DY model) [20] in the proposed scheme. Under the DY model, any two parties in the network communicate over an insecure (public) channel, in which the end-point communicating parties, such as C and SS, are not considered as trustworthy entities. Therefore, under the DY model, an adversary (passive as well as active) A can then eavesdrop, modify or delete the exchanged messages during communication. Before going to the detail phases of the proposed protocol, we present a brief summary of it in Figure 2.

4.3 User and Service Server Registration Phase

Initially, both C and SS need to register themselves with BS *via* out-of-band channel [8] or postal network. Suppose that C_i wants to enrol his secret credentials with BS. Then, the detail enrolment step is as follows:

Step 1: C_i transforms his user identity and password offline using B_{app} as $T_{C_{ID}} = H_b(C^i_{ID})$ and $T_{C_{pwd}} = H_{b_1}(\text{PWD}||r_1)$. We may note that B_{app} has two hash functions as $H_b(.)$ and $H_{b_1}(.)$. For example, suppose that C_i executes offline B_{app} application in his smart phone and transforms his user identity using $H_{b_1}(.)$. Then, C_i chooses one random phone number from its contact information and transforms his password using $H_{b_1}(.)$. C_i needs to keep both PWD and r_1 secret.

Table 1. Notations and their descriptions.

Notations	Descriptions	Notations	Descriptions
A_i	An application running on user C_i's workstation to access service server	SS^j_{ID}	Public identity of SS_j
BS	Back End Authentication Server	SI^j_{ss}	Secret identity of SS_j
BS_{ID}	BS's public identity	SS^{id}_{sec}	Transformed identity of SS_j
B_{app}	An application computes $H_b(X_P)$ offline	s_{ss}	SS_j 's private key
C_i	i^{th} Client	s_c	C_i's private key
CA	Certificate Authority	s_b	BS's private key
C^i_{ID}	Public identity of C_i	U	A set of all points on the elliptic curve $E_p(m,n)$
C_P	Certificate for a principal P issued by CA	H_1	One-way cryptographic hash function as $\{0,1\}^* \rightarrow U$ [13]
$E_p(m,n)$	Elliptic curve $y^2 \equiv x^3 + mx + n$ (mod p) on the field Z_p	H_2	One-way cryptographic hash function as $U \rightarrow \{0,1\}^*$ [13]
ECC	Elliptic curve cryptography	H_b	One-way hash function as h(·)
ECDLP	Elliptic curve discrete logarithm problem	H_{b_1}	One-way hash function as h(J ‖ δ)
ECDDHP	Elliptic curve decisional Diffie-Hellman problem	$T_{C_{ID}}$	C_i's transformed password
F_k	A field containing the elements $0, 1, \cdots, (k-1)$	$T_{C_{pwd}}$	C_i's transformed user identity
G	A generator point $\in E_p(m,n)$ of some order (say, k)	X_P	Security credential X of a principal P
$h(\cdot)$	One way hash function, such as SHA-1, SHA-2 …etc.	$\mathbb{Z}_k$	A set containing the elements $0, 1, \ldots, (k-1)$
k	A 160 to 256-bit prime	(x/y)	P computes x number of exponentiations in real time and y number of exponentiations in offline mode
L	The size of a group element [1]	|q|	Bit length of q
l	The size of the hash value [1]	J	A secret information
PWD	Password of C_i	δ	Any random secret
P	A principal such as C_i, SS_j , BS	Q_{ss}	SS_j 's public key
Q_b	BS's public key	SS	Service Server
Q_c	C_i's public key	SS_j	j^{th} SS

Step 2: C_i sends C^i_{ID}, $T_{C_{ID}}$ and $T_{C_{pwd}}$ to BS via postal network.

Step 3: BS administrator creates C_i's account and updates user table with C^i_{ID}, $T_{C_{ID}}$ and $T_{C_{pwd}}$.

Step 4: BS broadcasts $T_{C_{ID}}$ to all SS's.

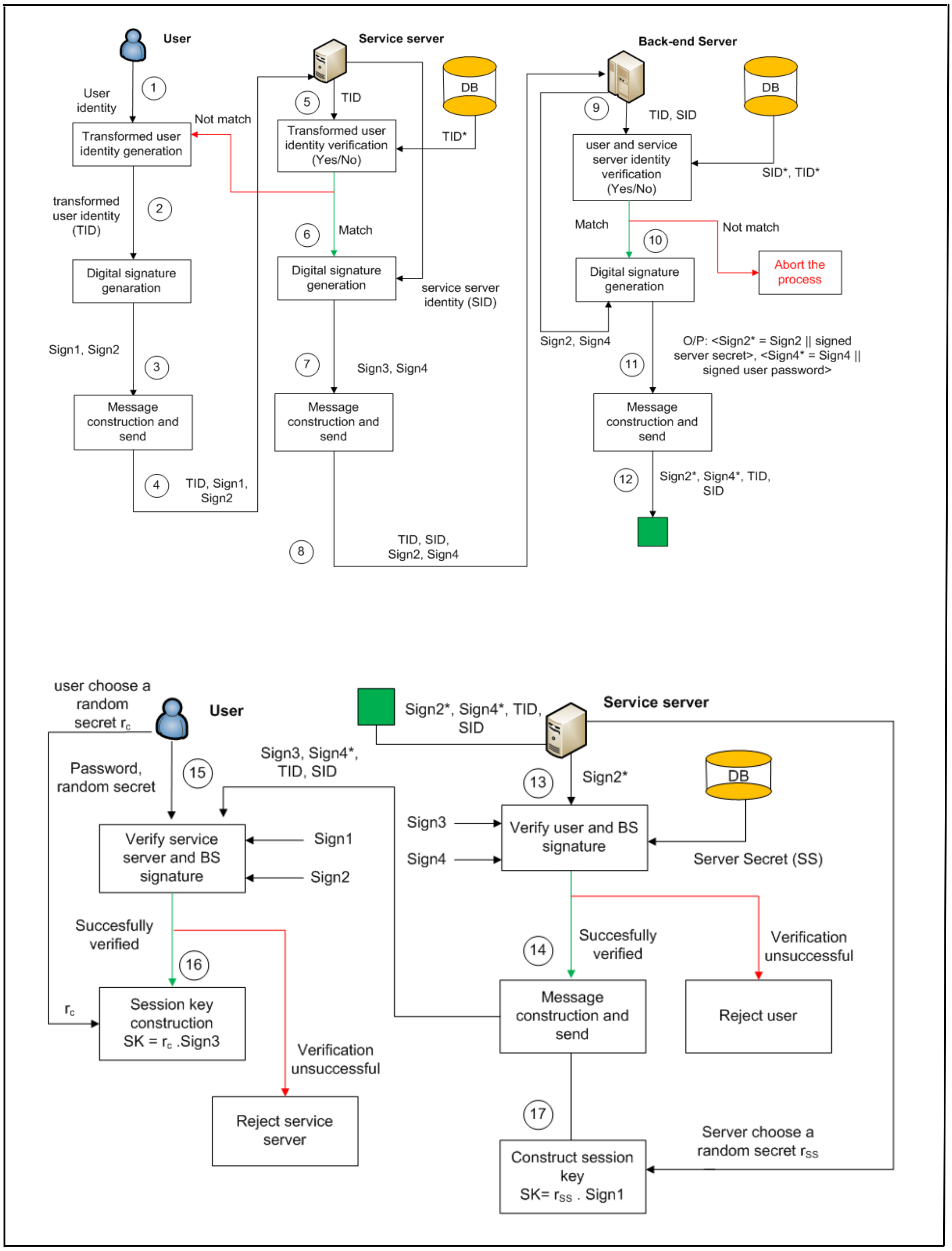
User
Service server
Back-end Server
DB
DB
User identity
1
Transformed user identity generation
Not match
5
TID
Transformed user identity verification (Yes/No)
TID*
9
TID, SID
user and service server identity verification (Yes/No)
SID*, TID*
transformed user identity (TID)
2
6
Match
Match
Not match
10
Abort the process
Digital signature genaration
Digital signature generation
service server identity (SID)
Digital signature generation
Sign2, Sign4
O/P: <Sign2* = Sign2 || signed server secret>, <Sign4* = Sign4 || signed user password>
Sign1, Sign2
3
7
Sign3, Sign4
11
Message construction and send
Message construction and send
Message construction and send
4
TID, Sign1, Sign2
12
Sign2*, Sign4*, TID, SID
8
TID, SID, Sign2, Sign4
user choose a random secret r_c
User
Sign2*, Sign4*, TID, SID
Service server
Password, random secret
15
Sign3, Sign4*, TID, SID
13
Sign2*
DB
Verify service server and BS signature
Sign1
Sign2
Sign3
Sign4
Verify user and BS signature
Server Secret (SS)
Succesfully verified
16
Succesfully verified
14
Verification unsuccessful
r_c
Session key construction SK = r_c .Sign3
Message construction and send
Reject user
Verification unsuccessful
Reject service server
17
Server choose a random secret r_SS
Construct session key SK= r_SS . Sign1

Figure 2. Summary of the proposed protocol.

Similarly, SS_j's administrator deploys server id (*i.e.*, SS_{ID}^{j}) and transformed secret identity (*i.e.*, $H_b(SI_{SS}^{j})$) to BS. BS updates the service server table in its database. It may be noted that, the secret identity (i.e., SI_{SS}^{j}) has been assigned to each SS by its respective administrator. This completes both C_i and SS_j enrolment process. After completion of registration, C_i can access services from SS_j followed by an authenticated key exchange task as follows:

4.4 Authenticated Key Agreement Phase

This sub-section presents the proposed password-based two-server authentication and key exchange protocol. All the message communications of the proposed protocol are shown in Fig. 3. The detail step in this process is as follows:

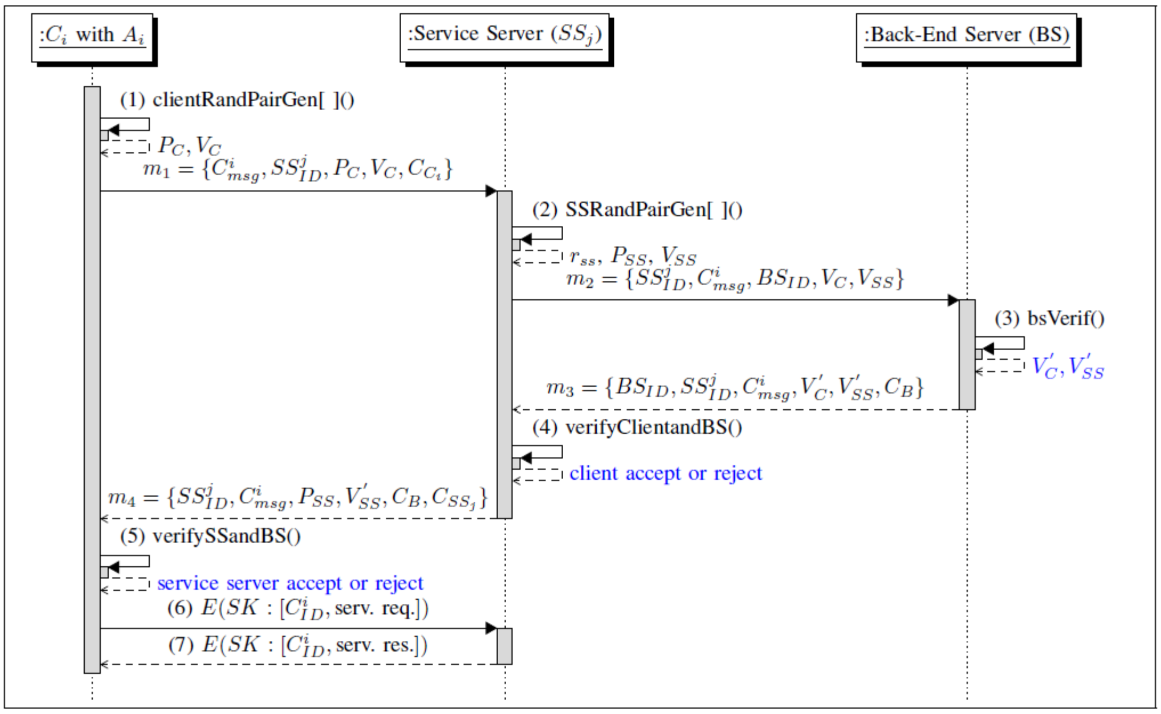
:C_i with A_i
:Service Server (SS_j)
:Back-End Server (BS)
(1) clientRandPairGen[ ]()
P_C, V_C
$m_1 = \{C^i_{msg}, SS^j_{ID}, P_C, V_C, C_{C_i}\}$
(2) SSRandPairGen[ ]()
r_{ss}, P_{SS}, V_{SS}
$m_2 = \{SS^j_{ID}, C^i_{msg}, BS_{ID}, V_C, V_{SS}\}$
(3) bsVerif()
V'_C, V'_{SS}
$m_3 = \{BS_{ID}, SS^j_{ID}, C^i_{msg}, V'_C, V'_{SS}, C_B\}$
(4) verifyClientandBS()
client accept or reject
$m_4 = \{SS^j_{ID}, C^i_{msg}, P_{SS}, V'_{SS}, C_B, C_{SS_j}\}$
(5) verifySSandBS()
service server accept or reject
(6) $E(SK : [C^i_{ID}$, serv. req.$])$
(7) $E(SK : [C^i_{ID}$, serv. res.$])$

Figure 3. Summary of mutual authentication and key exchange process.

Step 1: C_i enters his identity and service server identity in application A_i.

Step 2: A_i computes C^i_{msg} and a signature pair (i.e., P_C and V_C) using clientRandPairGen[ ]() (see Fig. 4).

Step 3: A_i constructs a message m_1 = { $C^i_{msg}, SS^j_{ID}, P_C, V_C$, C_{C_i} } and sends the same to SS_j .

Step 4: After receiving m_1, SS_j checks the transformed user identity (i.e., C^i_{msg}= $T_{C_{ID}}$) from its database.

Step 5: If user identity exists, then SS_j computes a signature pair (i.e., $\{P_{SS}$, $V_{SS}\}$) by taking the input of service server identity as SS^j_{ID} using SSRandPairGen[ ]() (refer to Figure 5). Step 6 is repeated, else the $C_i's$ request is rejected.

Step 6: SS_j constructs a message m_2 = { $SS^j_{ID}, C^i_{msg}, BS_{ID}, V_C, V_{SS}, C_{C_i}$ } and sends the same to BS.

Step 7: BS checks both previously stored transformed user identity and server identity from its database.

Step 8: If both the transformed identities exist, then BS modifies both random values V_C and V_{SS} as V'_C and V'_{SS}, respectively, using bsVerif() (see Figure 6) and go to Step 9, else it rejects SS_j's request.

Step 9: After verifying both C_i and SS_j , BS constructs a message as m_3= { $BS_{ID}, C^i_{msg}, SS^j_{ID}, V'_C, V'_{SS}$, C_B } and sends the same to SS_j.

Step 10: After getting reply from BS, SS_j verifies the legitimacy of both C_i and BS using verifyClientandBS() (refer to Figure 8).

```
function clientRandPairGen[ ]($C^i_{ID}, s_c$)
{
Input: client identity and client private key
Output: two nonnegative random numbers as $P_C$ and $V_C$
Data: $C^i_{ID}, C_{C_i}, PWD$
Compute $C^i_{msg} := H_b(C^i_{ID})$
Choose a pseudo-random number $r_c \in \mathbb{Z}^*_k$
Compute $P_C := r_c \cdot G$
Compute $V_C := r_c \cdot H_2(H_1(C^i_{msg})) + s_c \cdot H_2(P_C) \bmod k$
return $P_C$, $V_C$
}
```

Figure 4. Client-side signature generation process.

```
function SSRandPairGen[ ]($SS^j_{ID}, s_{ss}$)
{
Input: service server identity and service server private key
Output: two nonnegative random numbers $P_{SS}$ and $V_{SS}$
Data: $C^{id}_{reg} := H_b(C^i_{ID})$, $SS^{id}_{sec} := H_b(SI^j_{SS})$, $SS^j_{ID}$, $C_{SS_j}$
if($C^i_{msg} = C^{id}_{reg}$)
  {
      Choose a random number $r_{ss} \in \mathbb{Z}^*_k$
      Compute $P_{SS} := r_{ss} \cdot G$
      Compute $V_{SS} := r_{ss} \cdot H_2(H_1(SS^j_{ID})$
                $+ s_{ss} \cdot H_2(P_{SS}) \bmod k$
      return $r_{ss}$, $P_{SS}$, $V_{SS}$
  }
else
  {
      return false // reject $C_i$ //
  }
}
```

Figure 5. Service server-side signature generation process.

Step 11: If both of them are verified successfully, then SS_j constructs session key as $SK = r_{ss} \cdot r_c \cdot G$ and a message m_4 = { $C^i_{msg}, SS^j_{ID}, P_{SS}, V'_{SS}, C_B, C_{SS_j}$ } and goto Step 12, else it rejects C_i's request.

```
function bsVerif($SS^j_{ID}, C^{id}_{reg}, V_C, V_{SS}$)
{
Input: service server identity, client transform identity and a pair
of random numbers
Output: two nonnegative random numbers as $V'_C$ and $V'_{SS}$
Data: $C_B, C^{id'}_{reg} := H_b(C^i_{ID})$, $SS^j_{rgid} := SS^j_{ID}$, $C^{pwd}_{reg} :=$
$H_{b_1}(PWD||r_1)$, $SS^{id}_{sec} := H_b(SI^j_{SS})$
if($C^{id'}_{reg} = C^{id}_{reg}$ & $SS^j_{ID} = SS^j_{rgid}$)
    {
        Compute $V'_C := V_C + s_b \cdot H_2(H_1(SS^{id}_{sec}))$
        Compute $V'_{SS} := V_{SS} + s_b \cdot H_2(H_1(C^{pwd}_{reg}))$
        return $V'_C$, $V'_{SS}$
    }
else
    {
        break // unauthorized access//
    }
}
```

Figure 6. Client and Service server verification at BS.

```
function verifySSandBS ($SS^j_{ID}, \mathcal{P}_{SS}, \mathcal{V}'_{SS}, C_B, C_{SS_j}$)
{
Input: service server identity, a pair of random numbers, BS certificate, SS certificate
Output: return true or false
Data: $r_c$, $C^{pwd}_{reg} := H_{b_1}(PWD||r_1)$
Compute $temp_2 := V'_{SS} \cdot G$
Compute $temp_3 := P_{SS} \cdot H_2(H_1(SS^j_{ID})) + Q_{ss} \cdot H_2(P_{SS})$
                $+Q_b \cdot H_2(H_1(C^{pwd}_{reg}))$
if($temp_2 = temp_3$)
    {
        Compute $SK := r_c \cdot P_{SS} := r_c \cdot r_{ss} \cdot G$
        return true // accept $SS_j$ and BS //
    }
else
    {
        return false //reject $SS_j$ and BS//
    }
}
```

Figure 7. Client-side verification and session key generation process.

Step 12: SS_j sends the message m_4 to C_i .

Step 13: After receiving m_4 , C_i checks the legitimacy of both SS_j and BS using verifySSandBS() (see Figure 7).

Step 14: If both of them are verified successfully, then C_i constructs session key as $SK = r_c \cdot r_{ss} \cdot G$.

Finally, C_i and SS_j establish a secure channel between themselves and set up a shared secret symmetric key as $SK = r_c \cdot r_{ss} \cdot G = r_{ss} \cdot r_c \cdot G$.

```
function verifyClientandBS($C^i_{msg}, P_C, V'_C, C_{C_i}, C_B$)
{
Input: client transformed identity, a pair of random numbers, client certificate and BS certificate
Output: return true or false
Data: $C^{id}_{reg} := H_b(C^i_{ID}), SS^{id}_{sec} := H_b(SI^j_{SS}), SS^j_{ID}, C_{SS_j}$
Compute $temp := V'_C \cdot G$
Compute $temp_1 := V_C \cdot G + Q_b \cdot H_2(H_1(SS^{id}_{sec}))$
if($temp = temp_1$)
    {
        Compute $SK := r_{ss} \cdot P_C := r_{ss} \cdot r_c \cdot G$
        return true //accept $C_i$ and BS //
    }
else
    {
        return false // reject $C_i$ and BS //
    }
}
```

Figure 8. Service server-side verification and session key generation process.

4.4.1 Proof of Correctness

In order to verify the legitimacy of both sender and receiver along with back-end third party server, each sender or receiver must adhere to the following two cases:

Case 1. *For the purpose of verifying C_i and BS, SS_j needs to check*

$$V'_C \cdot G = P_C \cdot H_2(H_1(C^i_{msg})) + Q_c \cdot H_2(P_C) + Q_b \cdot H_2\left(H_1\left(SS^{id}_{sec}\right)\right). \tag{1}$$

To make the aforesaid check work, the following condition must hold:

$$V_C = r_c \cdot H_2(H_1(C^{i}_{msg})) + s_c \cdot H_2(r_c \cdot G)\ (mod\ k). \tag{2}$$

Case 2. *For the purpose of verifying SS_j and BS, C_i needs to check*

$$V'_{SS} \cdot G = P_{SS} \cdot H_2(H_1(SS^{j}_{ID})) + Q_{ss} \cdot H_2(P_{SS}) + Q_b \cdot H_2\left(H_1(C^{pwd}_{reg})\right). \tag{3}$$

To make the aforesaid check work, the following condition must be satisfied:

$$V_{SS} = r_{ss} \cdot H_2(H_1(SS^{j}_{ID})) + s_{ss} \cdot H_2(r_{ss} \cdot G)\ (mod\ k) \tag{4}$$

5. Formal Verification in AVISPA

In order to check the validity of a security protocol, several formal verification tools and techniques have emerged. In order to analyse our proposed protocol, we utilize AVISPA (Automated Validation of Internet Security Protocols and Applications) tool [21]. This tool is providing a High Level Protocol Specification Language (HLPSL) [22] [23], which is the *de facto* language recommended for AVISPA, to codify a security protocol. After codification, AVISPA internally converts the protocol into an Intermediate Format (IF), in order to assess its formal specification. In addition to this, AVISPA integrates four different model checkers which is actually perform the end-to-end analysis of the protocol. Moreover, the sending/receiving channels utilized for communication among several parties are modelled using the standard Dolev-Yao mechanism [20] to assess perfect secrecy.

According to the specification of HLPSL, each principal is mapped into a basic role, where all the local and global variables are specified. In addition to this, each role needs to declare an initial state and the state transitions. These state transitions represent different interactions; such as send or receive message exchanges among other roles. We code our authentication message exchanges in HLPSL. Moreover, we have specified three different roles: a client (c), a service server (ss) and a back-end authentication server (b). However, by taking an instance of a basic role, we have initiated a composite role called “session” to configure a single protocol run.

In order to specify such security objectives in HLPSL, AVISPA equips some standard commands, for example, “secrecy_of” (used to define the secrecy of keys), “authentication_on” (used to specify strong authentication on nonce), “weak_authentication_on” (used to represent weak authentication on timestamp) …etc. We model the security goals for the proposed protocol as follows:

1) *Origin authentication or man-in-the-middle attacks prevention:* It is done by incorporating the “authentication_on” to every security goal in every data exchange.
2) *Replay attack prevention:* It is achieved by establishing strong origin authentication.
3) *Data confidentiality:* It is ensured by interpreting “secrecy_of” to the secret keys (i.e., PWD and SS^{id}_{sec}) used in the protocol.
4) *Data integrity:* It is assured by applying “secrecy_of” to the secret one time distinct values (i.e., $P_C, V_C, P_{SS}, V_{SS}, V'_C, V'_{SS}, r_c\ and\ r_{ss}$) exchanged by the protocol.

In order to validate our proposed protocol, we have considered seven different cases to find out the attack traces in AVISPA. All the cases and session configurations are presented in Table 2 and illustrated as follows:

- **C1:** It shows a single protocol session implementation with all the legitimate agents (i.e., c, ss and b) involved into authentication.
- **C2, C3 and C4:** They consider a situation, where an adversary (i) is trying to impersonate the client (c), the service principal (ss) and the back end authentication server.
- **C5:** It defines a situation, in which two parallel sessions are being executed and client (c) is interpreting the role of the service server (ss).
- **C6:** It represents a scenario, where two concurrent sessions are being executed simultaneously and client (c) is playing the role of back-end authentication server (b).
- **C7:** This case illustrates the effects of two parallel session executions in a single protocol

run, where an adversary (i) is trying to act same as the client (c).

Table 2. Test cases' execution in AVISPA.

Cases	**Different session configurations**
C1	session (c, ss, b, kc, ks, r1, srs, a, f, h1, h2)
C2	session (c, i, b, kc, ksi, ri, sri, a, f, h1, h2)
C3	session (c, ss, i, kci, ksi, ri, sri, a, f, h1, h2)
C4	session (i, ss,b, kci, srs, ri, sri, a, f, h1, h2)
C5	session (c, ss, b, kc, ks, r1, srs, a, f, h1, h2) session (c, c, b, kc, ksi, r1, sri, a, f, h1, h2)
C6	session (c, ss, b, kc, ks, r1, srs, a, f, h1, h2) session (c, ss, c, kc, ksi, r1, sri, a, f, h1, h2)
C7	session (c, ss, b, kc, ks, r1, srs, a, f, h1, h2) session (i, ss, b, kci, srs, ri, sri, a, f, h1, h2)

[Note: kc – client symmetric key (C_{reg}^{pwd}), ks – service server secret (SS_{sec}^{id}), kci and ksi – intruder knowledge about client and service server key, r1 and srs – shared secret information of client and service server with back-end server, a – hash function, f – hash function, h1 – hash function, h2 – hash function].

After execution of all the above cases in AVISPA, it did not reveal any attack traces. Thus, we can conclude that our proposed protocol is safe from man-in-the-middle attacks, impersonation attacks, parallel-session attacks and replay attacks.

Remark 1: *We consider four hash functions, such as 'a', 'f', 'h1' and 'h2', to codify the proposed protocol in AVISPA utilizing HLPSL. Here, 'a', 'f', 'h1' and 'h2' signify arithmetic addition, ECC point multiplication, encoding of a plaintext to an ECC point (H_1) and decoding of an ECC point to a random text (H_2), respectively. Note that, in HLPSL, there is no provision to code these aforesaid operations directly using mathematical operators. To specify these operations in HPLSL, hash functions are utilized. For example, in role specification and environment configuration, we firmly declare these operations as "a, f, h1, h2: hash_func".*

Remark 2: *In this paper, we use AVISPA tool (version 1.1) to prove that our proposed protocol is safe or not. Currently, the tool supports only four known attacks; namely, man-in-the-middle, replay, impersonation and parallel session attacks. In addition to this, as per the specification of the same tool, it is not possible to trace any arbitrary security attacks. Therefore, in AVISPA version -1.1, there is no scope to implement the recent attacks, like side channel attacks, covert channel attacks, privileged-insider attacks and identity compromization attacks existing in the cloud computing domain. We will finalize this issue in the near future utilizing a later version of the AVISPA tool.*

6. INFORMAL SECURITY ANALYSIS

An informal security analysis of the proposed protocol is carried out with respect to different attacks; namely, man-in-the-middle, replay, offline dictionary, privileged-insider, byzantine, impersonation, identity compromization attacks based on the aforesaid adversary model as follows:

Claim 1: *Resilient against man-in-the-middle attacks as an active adversary without control on servers.*

Proof: Suppose an adversary *Adv* traps the message ⟨ $\mathrm{C}_{\mathrm{msg}}^{\mathrm{i}}$, $\mathrm{SS}_{\mathrm{ID}}^{\mathrm{j}}$, $\mathrm{P_C}$, $\mathrm{V_C}$, $\mathrm{C_{C_i}}$ ⟩ sent by C_i to SS_j. It is noted that:

$$P_C = r_c \cdot G,\ r_c \in \mathbb{Z}_k^* \tag{5}$$

$$V_C = r_c \cdot H_2\left(H_1\left(C_{msg}^i\right)\right) + s_c \cdot H_2(P_C)(mod\ k). \tag{6}$$

It is hard to compute r_c in Eq. 5 (or s_c in Eq. 6) from P_C (or Q_C) due to ECDLP. Further, *Adv* cannot alter either P_C or V_C or both. Suppose that *Adv* alters V_C by V_C'' . Then, to satisfy the verification condition in Eq. 6, *Adv* has to alter from P_C to P_C' . But, due to ECDLP, it is hard to alter V_C by V_C'' and then satisfy the condition in Eq. 6. Analogously, it is also hard to alter P_C by P'_C and then find V''_C so that:

$$V''_C \cdot G = P_C \cdot H_2\left(H_1(C^i_{msg})\right) + Q_c \cdot H_2(P_C) + Q_b \cdot H_2(H_1(SS^{id}_{sec}))$$

holds. Similarly, it holds for the other messages, such as m_2, m_3 and m_4, respectively. Further, suppose that the adversary *Adv* wants to find the session key (i.e., SK) from either the value of $P_C = r_c \cdot$ G or $P_{SS} = r_{ss} \cdot G$ or both by applying man-in-the-middle attacks and guessing the value of r_c and r_{ss}. Then, *Adv* would again end up with the computationally intractable or hard problem (i.e., ECCDHP assumption) to compute $SK = r_c \cdot P_{SS} = r_{ss} \cdot P_C = r_c \cdot r_{ss} \cdot G$. Hence, we can conclude that our scheme is resilient to man-in-the-middle attacks.

Claim 2: *Resilient against server-side identity compromization attacks as a passive adversary controlling SS_j.*

Proof: In the proposed scheme, SS_j is allowed to store only hashed client identities instead of original identities. Therefore, if a malicious insider gets this information, then he cannot trace out the original identity of the client. Suppose an adversary eavesdrops message m_1 = { $C^i_{msg}, SS^j_{ID}, P_C, V_C, C_{C_i}$ } and tries to get the actual identity $\mathrm{C^i_{ID}}$ of $\mathrm{C_i}$ from $\mathrm{C^i_{msg}}$. But, it is computationally hard to extract $\mathrm{C^i_{ID}}$ from $\mathrm{C^i_{msg}}$, as $\mathrm{C^i_{ID}}$ has been transformed using one-way cryptographic hash function at the time of client enrolment phase.

Claim 3: *Resilient against replay attacks.*

Proof: For a particular session, C_i chooses a pseudo-random number r_c acting as a nonce or fresh value. We can easily infer from the four communication messages (i.e., m_1, m_2, m_3 and m_4) that the V_C value is exchanged among three parties and derived from r_c. As r_c is fresh for a single protocol run, thus we can say that V_C is also fresh *vis-a-vis* the messages are also fresh. This ensures our protocol to be free from replay attacks. More precisely, supposed that there is a protection against replay attack during the authentication phase, SS_j can store the values P_C and V_C in its cache memory temporarily. SS_j first checks if $P_C = P'_C$ and $V_C = V'_C$. If this is valid, the message is treated as replay message. Otherwise, SS_j updates P_C and V_C with P'_C and V'_C in its cache. In a similar way, it can address the replay attack issue during the mutual authentication phase between SS_j and BS.

Claim 4: *Resilient against single point of vulnerability and single point of failure.*

Proof: In the proposed dual server model, BS is not directly reachable to the client or an external adversary and it manages the crucial security credentials for both C_i and SS_j. Although SSs' are the front-end interface for malicious external adversary, it is not possible to gain any knowledge about the secret parameters (i.e., PWD, SI^j_{ss}, …etc.) except the hashed identity parameters of the client. This mitigates the single point of vulnerability issue. In addition to this, distribution of secret databases, periodical back-up and auditing of the security credentials from back-end server offline represent an easy task rather doing all these stuffs along with client or external intruder interfacing at front-end. Thus, our solution avoids the single point of failure issue without affecting the service availability of the system.

Claim 5: *Resilient against byzantine attacks.*

Proof: Intuitively, in order to design a full proof system to address byzantine attack, the proposed scheme is being planned in such a way that each entity would be able to substantiate the other entity along with the issuer of the security credentials. Suppose that SS_j wants to check the legitimacy of C_i. For this, SS_j needs to verify

$$V'_C \cdot G = r_C \cdot H_2\left(H_1(C^i_{msg})\right) \cdot G + s_c \cdot H_2(P_C) \cdot G + s_b \cdot H_2\left(H_1(SS^{id}_{sec})\right) \cdot G.$$

From this verification condition, we can easily infer that it associates with both C_i's and BS's private key, as well as C_i 's public identity. That means, implicitly, SS_j verifies both C_i and BS legitimacy using the public keys of themselves (see Eq. 1, Eq. 2, Eq. 3 and Eq. 4, respectively) before establishing the session key. Similarly, C_i checks the legitimacy of SS_j and BS before making the shared symmetric key with SS_j. Thus, it ensures trusted third party verification and is hence free from byzantine attacks.

Claim 6: *Resilient against offline dictionary attacks as a passive adversary controlling SS_j .*

Proof: Suppose that a malicious insider or a passive adversary *A* controls SS_j. As SS_j is not storing any dictionary of passwords for C_i in its secret database, where $i = 1, 2, \cdots, n$, then it is evident that there is no chance of offline dictionary attacks on C_i's password. Further, assume that *A* eavesdrops all four messages; namely, m_1, m_2, m_3 and m_4, respectively, from the protocol transcripts and tries to guess the C_i's original password. However, it is computationally hard to trace PWD without the knowledge of r_c , s_c *and* s_b, respectively.

Claim 7: *Resilient against offline dictionary attacks as a passive adversary controlling BS.*

Proof: Suppose a malicious insider or a passive adversary *A* controls BS. Then, in order to make offline dictionary attacks on C_i 's password infeasible, we encapsulate a random number (i.e., r_1) with C_i 's original password and make a transformed password as $T_{C_{pwd}} = H_{b_1}(\text{PWD}||r_1)$. Suppose that an insider or a passive adversary *A* compromises BS's secret database and tries to compute the password offline by dictionary attack. Then, he needs the knowledge about the random number r_1 . Further, suppose *A* to eavesdrop all the protocol transcripts and try to make an offline dictionary attack on the password. It is also hard to compute C_i's PWD without the knowledge of r_{ss}, s_{ss} and s_b, respectively.

Claim 8: *Resilient against privileged-insider attacks as a passive adversary controlling BS.*

Proof: During the client enrolment phase, B_{app} asks C_i to give his password. B_{app} transforms the password as $T_{C_{pwd}} = H_{b_1}(\text{PWD}||r_1)$ and C_i needs to send $T_{C_{pwd}}$ to BS *via* postal network. Now, suppose that a privileged-insider user of the BS, being an adversary *A*, knows the information $T_{C_{pwd}}$ by stealing the BS's database. Note that the masked password $H_{b_1}(\text{PWD}||r_1)$ contains the random secret r_1 , which is only known to C_i. Even if *A* guesses a password, he cannot verify it correctly without having r_1. Therefore, it is a computationally infeasible task for *A* to derive the password PWD of C_i . This shows that the privileged-insider attacks are protected in the proposed scheme.

Claim 9: *Resilient against ciphertext-only attacks.*

Proof: Suppose that for a particular session, an adversary *A* eavesdrops all the communication messages (i.e., m_1, m_2, m_3 and m_4) during authenticated key establishment phase. *A* repeats this process for multiple sessions for C_i whose public identity is $T_{C_{ID}} = H_b(C_{ID}^i)$. In this connection, *A* is having with himself a collection of messages. Form those messages, *A* chooses one ciphertext (i.e., V'_{SS}), which is associated with the C_i's password and constructs a set as S = $\{V'_{SS}\}$. After collecting such multiple sets as $S_i = \{V'_{SS_i}\}$, $\forall$ i $= 1, 2, \cdots,$ n, *A* tries to incorporate ciphertext-only attacks to find out the original plaintext (i.e., PWD) of C_i. However, to extract the corresponding plaintext or client's password from the set of ciphertexts, *A* needs the knowledge about r_1, r_c and s_c parameters. Without having these parameters, extraction of the original plaintext (i.e., PWD of C_i) is computationally intractable. Thus, the proof is completed and cipertext-only attacks are prevented

Claim 10: *Resilient against impersonation attacks.*

Proof: In order to mitigate impersonation attacks, the proposed scheme adopts a strong entity authentication principle. Here, SS_j verifies the legitimacy of client C_i using his transformed identity and his digital signature approved by BS and C_i verifies the legitimacy of SS_j utilizing its original identity and its digital signature approved by BS. This digital signature-based verification of each entity makes the protocol free from impersonation attacks by achieving strong authentication (see Cases C5 and C6 in Table 2 of Section 5).

Claim 11: *Resilient against stolen-verifier attacks.*

Proof: In order to mitigate the stolen-verifier attacks, the proposed scheme adopts a strong entity authentication principle. Here, SS_j verifies the legitimacy of client C_i using his transformed identity and his digital signature approved by BS and C_i verifies the legitimacy of service server SS_j utilizing its original identity and its digital signature approved by BS. This digital signature-based verification of each entity makes the protocol free from stolen-verifier attacks. Suppose an attacker steals the hashed verifier (C_{reg}^{pwd}) of C_i from BS and tries to login into the system. But, without knowing the PWD and r_1, it is computationally hard to satisfy the verification condition specified in Eq. 3. Therefore, we can conclude that our scheme is resilient against stolen-verifier attacks.

Claim 12: *Supports forward secrecy.*

Proof: Forward secrecy tells about an analogy that an attacker cannot find the session keys constructed in past sessions even if he discovers the C_i's password PWD and SS_j's secret key SI_{ss}^{j} . In the proposed scheme, both C_i and SS_j compute an incomparable session key SK = $r_c \cdot P_{SS}$ = $r_c \cdot r_{SS} \cdot G = r_{SS} \cdot P_C = r_{SS} \cdot r_c \cdot G = SK^*$ in every protocol run of the proposed scheme. The attacker cannot compute SK (or SK^*) from $P_{SS} = r_{SS} \cdot G$ and $P_C = r_c \cdot G$, even if he finds out C_i 's password PWD and SS_j's secret key SI_{ss}^{j} due to the hardness of large entropy ECC points. Thus, the proposed scheme provides forward secrecy.

7. PERFORMANCE ANALYSIS

This section deals with the performance evaluation of the proposed protocol. The performance analysis in this context is two-fold. Case 1: we compare our proposed protocol with the existing password-based two-server Diffie-Hellman authenticated key agreement protocols. Case 2: we substantiate the proposed methodology with the existing schemes available in cloud computing domain as follows:

(A) Case 1: Here, three major aspects for evaluation have been considered w.r.t. the existing password-based two-server Diffie-Hellman authenticated key agreement protocol as follows:

(1) Computational time (CT): In information theoretic sense, exponentiations govern individual entity's computational overhead [8]. In this synergy, we estimate the number of exponentiations as the evaluation of execution time and outline the performance results in Table 2. Consider C_i with A_i , for instance; it needs to compute a total of 4 exponentiations (see Eq. 1 and Eq. 2); that is, for the calculation of $P_C,\ V_C, Q_b \cdot\ H_2(H_1(SS_{sec}^{id}))$ and $V_C' \cdot\ G$ (refer to Section 3), and 2 exponentiations (i.e., $P_C,\ V_C$) out of them can be performed offline using [12]. We represent the value as "x/y" notation. More precisely, "4/2" signifies out of 4 exponentiations, client needs to perform 2 exponentiations in real time and other 2 exponentiations in offline mode.

(2) Communication overhead (CO): Since $|V_C|$ is equal to $|V_{SS}|$, we cannot distinguish between these two parameters. Thus, for ease of comparison, we fix it as L = $|V_C|$ = $|V_{SS}|$. In addition to this, we have grossly ignored the bandwidth for both principal identities and the public certificates in this aspect of evaluation.

(3) Communication rounds (CR): A single round comprises a one-way transmission of messages.

Table 3 shows that our proposed protocol is quite efficient in terms of both communication and computation. Thus, we can apply our protocol for wireless applications also.

Moreover, the proposed protocol is having several security and functional features (SFFs). Table 4 shows a comparative analysis of the proposed protocol with other existing schemes in terms of the SFFs as follows:

(B) Case 2: In order to evaluate the performance of the proposed protocol w.r.t the existing schemes available in cloud computing domain, here, we consider three major aspects; namely, computation cost (see Table 5 and Table 6), communication overhead and storage cost (see also Figure 9).

Table 3. Comparison with other existing schemes in terms of performance.

Principles	**Yang et al. [8]**	**JWX Protocol [15]**	**Yi et al. [1]**	**DHKEP [6]**	**Our Scheme**
C_i	CT: (4/2)	CT: (6/0)	CT: (4/0)	CT: (2/0)	CT: (4/2)
	CO: 4L + 2l	CO: 6L +2l	CO: 3L + 4l	CO: 2L	CO: 2L
	CR: 4	CR: 3	CR: 3	CR:2	CR: 2
SS_j	CT: (4/1)	CT: (8/0)	CT: (5/0)	CT: (2/0)	CT: (4/2)
	CO: 8L +3l	CO: 11L + 3l	CO: 6L + 3l	CO: 2L	CO: 4L
	CR: 8	CR: 6	CR: 4	CR:2	CR: 4
BS	CT: (3/1)	CT: (4/0)	CT: (5/0)	--	CT: (2/0)
	CO: 4L + 1l	CO: 5L +1l	CO: 6L + 3l	--	CO: 2L
	CR: 4	CR: 3	CR: 4	--	CR: 2

In addition to this, we present several security and functional features (see Table 7) of the proposed protocol. Note that, in this study, we omit device energy consumption during the proposed protocol execution.

The calculation of computation, communication and storage cost respectively, of the proposed protocol is given as follows:

Table 4. Comparison between the proposed protocol and other existing schemes in terms of attacks and other features.

SFFs	**Yang et al. [8]**	**JWX Protocol [15]**	**Yi et al. [1]**	**DHKEP [6]**	**Our Scheme**
SFF1	No	No	No	No	Yes
SFF2	No	No	No	No	Yes
SFF3	Yes	Yes	Yes	No	Yes
SFF4	No	No	No	No	Yes
SFF5	No	No	No	No	Yes
SFF6	No	No	No	No	Yes
SFF7	No	No	No	No	Yes
SFF8	Yes	Yes	Yes	No	Yes
SFF9	No	No	No	No	Yes
SFF10	No	No	No	No	Yes
SFF11	Yes	Yes	Yes	Yes	Yes

[Note: *SFF1*: Resists man-in-the-middle attacks, *SFF2*: Resists replay attacks, *SFF3*: Resists offline dictionary attacks, *SFF4*: Resists identity compromization attacks, *SFF5*: Resists privileged-insider attacks, *SFF6*: Resists single point of failure issue, *SFF7*: Resists single point of vulnerability issue, *SFF8*: Resists impersonation attacks, *SFF9*: Resists byzantine attacks, *SFF10*: Resists ciphertext-only attacks, *SFF11*: Support mutual authentication and key establishment.**]**

Table 5. A comparative summary of existing schemes in cloud considering computational complexity.

Phases	**Principals**	**Karla and Sood [53]**	**S. Kumari et al. [48]**	**Odelu et al. [54]**	**S. Kumari et al. [50]**	**Shen et al. [55]**	**Our scheme**
URP	C_i	NCCI	1T_h	1T_h + 1T_f	2B_H	1T_h	2T_h
	BS/RA	No role	No role	3T_h	2T_h	2T_h+ 1T_m	NCCI
SRP	SS_j	6T_h+ 2T_m	5T_h+ 2T_m	NCCI	NCCI	NCCI	1T_h
	BS/RA	No role	No role	2T_h	1T_h	1T_h	NCCI
AKAP	C_i	4T_h+3T_m	3T_h + 4T_m	8T_h + 1T_f +1T_s+ 3T_m	2B_H + 5T_h + 3T_m	5T_h+3T_m	2T_m (OFL)+ 2T_m (ONL) + 3T_h
	SS_j	5T_h+4T_m	4T_h + 4T_m	6T_h + 2T_s + 2T_m	5T_h + 3T_m	4T_h+1T_m	2T_m (OFL) +2T_m (ONL) + 2T_h
	BS/RA	No role	No role	11T_h+ T_s + 1T_m	6T_h + 2T_m	8T_h+2T_m	4T_h + 2T_m

[**Note:** URP – User Registration Phase, SRP – Service server Registration Phase, AKAP – Authentication and Key Agreement Phase, T_h – Represents the CPU time to execute a one-way hash function, B_H – Represents the CPU time to execute a bio-hashing operation, T_m – Represents the CPU time to execute an elliptic curve scalar point multiplication, T_f – Represents the CPU time to execute a fuzzy extraction operation, T_s – Represents the CPU time to execute a symmetric key en(de)cryption, NCCI – No computational cost involved, OFL – Offline calculation of an elliptic curve scalar point multiplication, ONL – Online calculation of an elliptic curve scalar point multiplication.]

Table 6. A comparative summary of existing schemes in cloud considering computational complexity.

Phases	Principals	Yoon and Yoo [58]	Mishra et al. [57]	Wu et al. [56]	He and Wang [59]	Our scheme
URP	C_i	1T_h	1B_H + 3T_h	1T_ h + 1T_f	1T_h	2T_h
	BS/RA	1T_h	3T_h	N/A	2T_h	NCCI
SRP	SS_j	NCCI	NCCI	1T_h	NCCI	1T_h
	BS/RA	1T_h	2T_h	N/A	1T_h	NCCI
AKAP	C_i	5T_h + 2T_m	1B_H + 9T_h	7T_h + 1T_f + 2T_s + 1T_m	3T_m + 7T_h	2T_m (OFL) +2T_m (ONL) + 3T_h
	SS_j	4T_h + 2T_m	7T_h	6T_h + 2T_ s + 1T_m	3T_m + 5T_h	2T_m (OFL)+ 2T_m (ONL) + 2T_h
	BS/RA	7T_h	No role	N/A	2T_m + 9T_h	4T_h + 2T_m

[**Note:** URP – User Registration Phase, SRP – Service server Registration Phase, AKAP – Authentication and Key Agreement Phase, T_h – Represents the CPU time to execute a one-way hash function, B_H – Represents the CPU time to execute a bio-hashing operation, T_m – Represents the CPU time to execute an elliptic curve scalar point multiplication, T_f – Represents the CPU time to execute a fuzzy extraction operation, T_s – Represents the CPU time to execute a symmetric key en(de)cryption, NCCI – No computational cost involved, OFL – Offline calculation of an elliptic curve scalar point multiplication, ONL – Online calculation of an elliptic curve scalar point multiplication.]

In [48], CPU time required to calculate T_h and T_m is 2.3 and 22.26 microseconds. Therefore, in our scheme, the total number of computations required is 9T_h + 6T_m which is equal to 13376.7 microseconds. In addition to this, to calculate the storage cost, in our scheme, we observe that SS_j needs to store the hashed identity (160 bits) of the user and BS needs to keep both hashed identity (160 bits) and hashed password (160 bits) of the user and SS_j's identity (32 bits) and SS_j's secret hashed identity (160 bits), respectively. As a result, the total storage cost required is 672 bits. Further, we calculate the communication overhead in terms of bits for the authentication message exchanges (discussed in Section 4) among different entities as follows:

m_1 = 160 bits + 32 bits + 3 * 160 bits
m_2 = 32 bits + 160 bits + 32 bits + 2 * 160 bits
m_3 = 32 bits + 32 bits + 4 * 160 bits
m_4 = 32 bits + 5 * 160 bits.

Thus, the overall communication overhead by summing up all the messages is equal to 2752 bits. We compare our protocol overhead in terms of computation, storage and communication cost with other existing schemes as follows.

From Table 5 and Table 6, we can conclude that the proposed protocol is efficient in terms of computation cost. In addition to this, from Figure 9, we can easily substantiate the aforesaid claim. Although, it is lagging in terms of communication and storage cost, but still the proposed scheme supports several SFFs mentioned in Table 7.

8. Conclusions

This paper proposes a secure and efficient two-server authentication and key agreement protocol for accessing secure cloud services. For this purpose, we have presented a new password-based two-server Diffie-Hellman authenticated key exchange protocol, in which both the client and the service server can establish a secret symmetric key between themselves after their mutual authentication. In addition to this, the proposed protocol has an efficient verification capability, where during mutual authentication

Table 7. A comparative summary: security and functional features.

Security and functional features	**Karla and Sood [53]**	**S. Kumari et al. [48]**	**Yoon and Yoo [58]**	**Mishra et al. [57]**	**Wu et al. [56]**	**Shen et al. [55]**	**Odelu et al. [54]**	**S. Kumari et al. [50]**	**Our Scheme**
SFF1	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
SFF2	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes
SFF3	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
SFF4	Yes	Yes	No	No	No	Yes	No	Yes	Yes
SFF5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SFF6	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
SFF7	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
SFF8	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
SFF9	No	NA	No	Yes	Yes	Yes	Yes	Yes	Yes
SFF10	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SFF11	No	No	Yes	Yes	No	Yes	Yes	Yes	No
SFF12	No	No	No	No	No	No	No	No	Yes
SFF13	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
SFF14	NA	NA	NA	NA	NA	NA	NA	NA	Yes
SFF15	NA	NA	NA	NA	NA	NA	NA	NA	Yes
SFF16	Yes	Yes	No	No	No	No	No	No	Yes
SFF17	No	Yes	Yes	Yes	Yes	Yes	Yes	NA	Yes
SFF18	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SFF19	No	NA	NA	NA	NA	NA	NA	Yes	Yes
SFF20	NA	NA	NA	NA	NA	NA	NA	NA	Yes

[**Note:** SFF1 – Resists privileged-insider attack, SFF2 – Provides user anonymity, SFF3 – Resists off-line password guessing attack, SFF4 – Resists user impersonation attack, SFF5 – Resists replay attack, SFF6 – Resists cloud-server impersonation attack, SFF7 – Provides mutual authentication, SFF8 – Provides forward secrecy, SFF9 – Resists known session-specific temporary information attack, SFF10 – Provides session key agreement and verification, SFF11 – Provides freely password changing facility, SFF12 – Resists byzantine attacks, SFF13 – Resists identity compromization attacks, SFF14 – Resists single point of failure issue, SFF15 – Resists single point of vulnerability issue, SFF16 – No extra hardware cost required, SFF17 – Resists stolen-verifier attacks, SFF18 – Resists man-in-the-middle attacks, SFF19 – Resists parallel session attacks, SFF20 – Resists ciphertext-only attacks, Yes – SFF achieved, No – SFF not achieved, NA – SFF not addressed in the particular scheme.]

phase both intended parties could verify their trusted third party (i.e., the issuer of security credentials). The security part of the proposed protocol has been thoroughly executed with the help of informal security analysis. In addition, the proposed scheme is also formally verified for security analysis using the broadly-used AVISPA tool. All the security analysis results show that the proposed protocol can protect various well-known attacks against a passive as well as active adversary. Also, it successfully alleviates several limitations in the existing schemes. The performance analyses also substantiate that our protocol is efficient in terms of both computation and communication overhead.

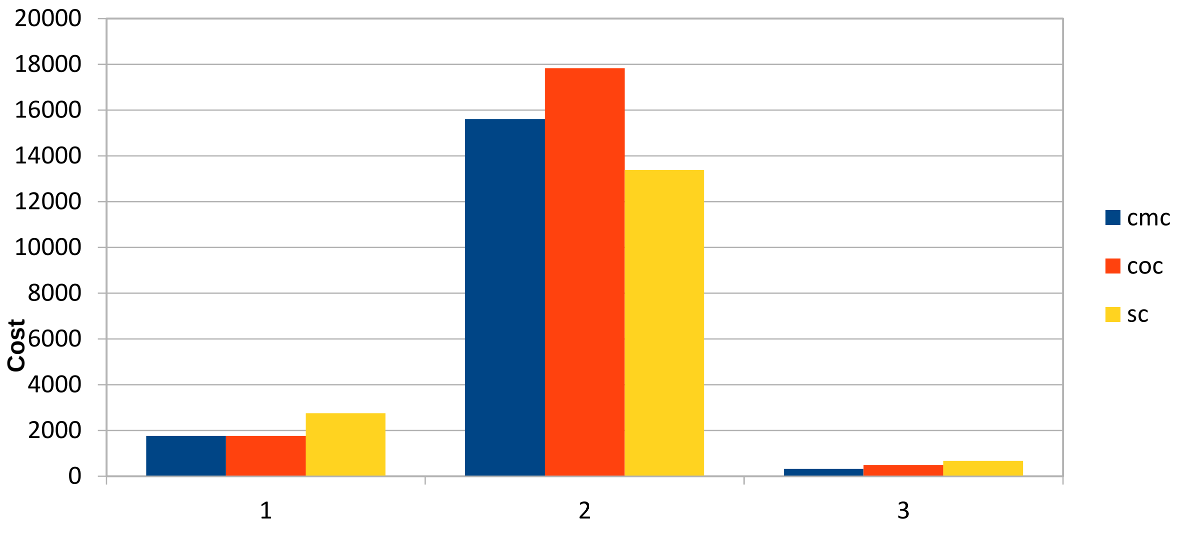

Figure 9. Performance comparison of the proposed protocol with the existing schemes.

[**Note:** Blue, orange and yellow colour plot represents Karla and Sood scheme [53], S. Kumari et al. scheme [48] and the proposed scheme, respectively. cmc – represent the communication cost in terms of bits. coc – represent the computation cost in terms of microseconds. sc – represent the storage cost in terms of bits. Here, 1, 2 and 3 represent the communication overhead, computation cost and storage cost, respectively.]

More significantly, a large-scale service server can be integrated with a single authorization centre hosting our protocol. In the future, we plan to evaluate the proposed protocol in a real-world environment setting. Note that, here, the real-world environment setting signifies an abstract network (wired/sensor/*ad-hoc* network) configuration, wherein different workstations are connected through a communication channel (wired/wireless). After implementing such an environment setting in any network simulator tool (NS2, NS3, …etc.), we can evaluate other performance metrics, such as end-to-end delay and throughput of the proposed scheme. This will allow us to fine-tune the protocol, if necessary, to offer better security and performance in a real-world deployment. The future direction also encompasses a formal treatment of our proposed scheme by utilizing Real-Or-Random model, Canetti-Krawczyk model or extended Canetti-Krawczyk model and analzing more complicated attacks, such as known key, unknown key-share, key-compromise impersonation, …etc.

ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers and the Editor-in Chief for their valuable suggestions, which helped us improve significantly the presentation and quality of the paper.

REFERENCES

[1] X. Yi, S. Ling and H. Wang, "Efficient Two-server Password-only Authenticated Key Exchange," Transactions on Parallel and Distributed systems, IEEE, vol. 24, no. 9, pp. 1773-1782, 2013.

[2] X. Yi, F. Y. Rao, Z. Tari, F. Hao, E. Bertino, I. Khalil and A. Y. Zomaya, "ID2S Password-authenticated Key Exchange Protocols," Transactions on Computers, IEEE, vol. 65, no. 12, pp. 3687-3701, 2016.

[3] V. Boyko, P. MacKenzie and S. Patel, "Provably Secure Password-authenticated Key Exchange Using Diffie-Hellman," in Advances in Cryptology–Eurocrypt, Springer Berlin/Heidelberg, pp. 156-171, 2000.

[4] M. Abdalla and D. Pointcheval, "Simple Password-based Encrypted Key Exchange Protocols," in Cryptographers' Track at the RSA Conference, Springer, pp. 191-208, 2005.

[5] M. Bellare and P. Rogaway, "The AuthA Protocol for Password-based Authenticated Key Exchange," Technical Report, IEEE, vol. 1363, 2000.

[6] W. Diffie and M. Hellman, "New Directions in Cryptography," Transactions on Information Theory, IEEE, vol. 32, no. 2, pp. 644-654, 1976.

[7] X. Yi, F. Hao and E. Bertino, "ID-based Two-server Password Authenticated Key Exchange," in Proceedings of the European Symposium on Research in Computer Security, Springer, pp. 257-276, 2014.

[8] Y. Yang, R. H. Deng and F. Bao, "A Practical Password-based Two-server Authentication and Key Exchange System," Transaction on Dependable and Secure Computing, IEEE, vol. 3, no. 2, pp. 105-114, 2006.

[9] Y. Yang, R. H. Deng and F. Bao, "Fortifying Password Authentication in Integrated Healthcare Delivery Systems," in Proceedings of the ACM Symposium on Information, Computer and Communications Security, ACM, pp. 255-265, 2006.

[10] E. Bresson, O. Chevassut and D. Pointcheval, "Security Proofs for an Efficient Password-based Key Exchange," in Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, pp. 241-250, 2003.

[11] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based Protocol Secure against Dictionary Attack," in Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72-84, 1992.

[12] R. P. Gallant, R. J. Lambert and S. A. Vanstone, "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms," in Annual International Cryptology Conference, Springer, pp. 190-200, 2001.

[13] P. A. Fouque, A. Joux and M. Tibouchi, "Injective Encodings to Elliptic Curves," in Australasian Conference on Information Security and Privacy, Springer, pp. 203-218, 2013.

[14] NIST white paper, "Recommended Elliptic Curves for Federal Government Use," [Online], Available: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, 1999, Last accessed: 2nd April, 2017.

[15] H. Jin, D. S. Wong and Y. Xu, "An Efficient Password-only Two-server Authenticated Key Exchange System," Proceeding of 9th International Conference of Information and Communication Security, Springer, pp. 44-56, 2007.

[16] P. Sarkar, "A Simple and Generic Construction of Authenticated Encryption with Associated Data," ACM Transactions on Information and System Security, vol. 13, no. 4, pp. 33, 2010.

[17] Secure Hash Standard, FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995, [Online], Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. Accessed in September 2017.

[18] R. W. D. Nickalls, "A New Approach to Solving the Cubic: Cardan's Solution Revealed," The Mathematical Gazette, vol. 77, no. 480, pp. 354-359, 1993.

[19] N. Koblitz, "Elliptic Curves Cryptosystems," Mathematics of Computation, vol. 48, pp. 203-209, 1987.

[20] D. Dolev and A. Yao, "On the Security of Public Key Protocols," Transactions on Information Theory, IEEE, vol. 29, no. 2, pp. 198-208, 1983.

[21] A. Armando, D. Basin, Y. Boichut et al., "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications," International Conference on Computer-Aided Verification, Springer, pp. 281-285, 2005.

[22] L. Vigan`o, "Automated Security Protocol Analysis with the AVISPA Tool," Electronic Notes in Theoretical Computer Science, Elsevier, vol. 155, pp. 61-86, 2006.

[23] D. V. Oheimb, "The High-level Protocol Specification Language HLPSL Developed in the EU Project AVISPA," Proceedings of APPSEM Workshop, pp. 1-17, 2005.

[24] B. De Decker, "Unix Security and Kerberos," Computer Security and Industrial Cryptography, Springer, pp. 257-274, 1993.

[25] S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman Key Agreement Protocols, "International Workshop on Selected Areas in Cryptography, Springer, pp. 339-361, 1998.

[26] H. Krawczyk, "HMQV: A High-performance Secure Diffie-Hellman Protocol," Annual International Cryptology Conference, Springer, pp. 546-566, 2005.

[27] L. Harn, W. J. Hsin and M. Mehta, "Authenticated Diffie–Hellman key agreement protocol using a single cryptographic assumption," IEE Proceedings-Communications, vol. 152, no. 4, pp. 404-410, 2005.

[28] B. LaMacchia, K. Lauter and A. Mityagin, "Stronger Security of Authenticated Key Exchange," International Conference on Provable Security, Springer, pp. 1-16, 2007.

[29] C. Neuman, S. Hartman, T. Yu and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120, [Online], Available: https://tools.ietf.org/pdf/rfc4120.pdf, 2005.

[30] B. C. Neuman, T. Ts'o and Kerberos, "An Authentication Service for Computer Networks," IEEE Communications Magazine, vol. 32, no. 9, pp. 33-38, 1994.

[31] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, vol. 21, no. 12, pp. 993-999, 1978.

[32] L. O'gorman, A. Bagga and J. Bentley, "Query-directed passwords," Computers & Security, Elsevier, vol. 24, no. 7, pp. 546-560, 2005.

[33] J. H. Yang and P. Y. Lin, "An ID-based User Authentication Scheme for Cloud Computing," IEEE 10th Inter. Conf. on Intell. Inform. Hiding and Multimedia Signal Processing (IIH-MSP), pp. 98-101, 2014.

[34] T. H. Chen, H. L. Yeh and W. K. Shih, "An Advanced ECC Dynamic ID-based Remote Mutual Authentication Scheme for Cloud Computing," IEEE 5th International Conference on Multimedia and Ubiquitous Engineering (MUE), pp. 155-159, 2011.

[35] D. Wang, Y. Mei, C. G. Ma and Z. S. Cui, "Comments on an Advanced Dynamic ID-based Authentication Scheme for Cloud Computing," International Conference, WISM, vol. 7529, Lecture Notes in Computer Science, Springer, pp. 246-253, 2012.

[36] Z. Hao, S. Zhong and N. Yu, "A Time-bound Ticket-based Mutual Authentication Scheme for Cloud Computing," International Journal of Computers, Communications and Control, vol. 6, no. 2, pp. 227-235, 2011.

[37] C. D. Jaidhar, "Enhanced Mutual Authentication Scheme for Cloud Architecture," IEEE 3rd International Advance Computing Conference (IACC), pp. 70-75, 2013.

[38] M. Wazid, A. K. Das, S. Kumari, X. Li and F. Wu, "Provably Secure Biometric-based User Authentication and Key Agreement Scheme in Cloud Computing," Security and Communication Networks, vol. 9, no. 17, pp. 4103-4119, 2016.

[39] P. Gope and A. K. Das, "Robust Anonymous Mutual Authentication Scheme for n-times Ubiquitous Mobile Cloud Computing Services," Internet of Things Journal, IEEE, pp. 1-9, DOI: 10.1109/JIOT.2017.2723915, 2017.

[40] V. Odelu, A. K. Das, S. Kumari, X. Huang and M. Wazid, "Provably Secure Authenticated Key Agreement Scheme for Distributed Mobile Cloud Computing Services," Future Generation Computer Systems, vol. 68, pp. 74–88, 2017.

[41] J. L. Tsai and N. W. Lo, "A Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services," IEEE Systems Journal, vol. 9, no. 3, pp. 805-815, 2015.

[42] I. E. Liao, C. C. Lee and M. S. Hwang, "A Password Authentication Scheme over Insecure Networks," Journal of Computer and System Sciences, vol. 72, no. 4, pp. 727-740, 2006.

[43] S. M. Bellovin and M. Merritt, "Limitations of the Kerberos Authentication System," ACM SIGCOMM Computer Communication Review, vol. 20, no. 5, pp. 119-132, 1990.

[44] G. S. Sadasivam, K. A. Kumari and S. Rubika, "A Novel Authentication Service for Hadoop in Cloud Environment," IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1-6, 2012.

[45] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp. 453-474, 2001.

[46] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement," Designs, Codes and Cryptography, Springer, vol. 28, no. 2, pp. 119-134, 2003.

[47] S. K. Sood, A. K. Sarje and K. Singh, "A Secure Dynamic Identity-based Authentication Protocol for Multi-server Architecture," Journal of Network and Computer Applications, Elsevier, vol. 34, no. 2, pp. 609-618, 2011.

[48] S. Kumari, M. Karuppiah, A. K. Das, X. Li, F. Wu and N. Kumar, "A Secure Authentication Scheme Based on Elliptic Curve Cryptography for Iot and Cloud Servers," Journal of Supercomputing, Springer, DOI: 10.1007/s11227-017-2048-0, pp. 1-26, 2017.

"An Efficient Two-Server Authentication and Key Exchange Protocol for Accessing Secure Cloud Services" , D. Chattaraj, M. Sarma and D. Samanta.

[49] F. Wu, L. Xu, S. Kumari, X. Li, A. K. Das and J. Shen, "A Lightweight and Anonymous RFID Tag Authentication Protocol with Cloud Assistance for E-Healthcare Applications," Journal of Ambient Intelligence and Humanized Computing, Springer, DOI: 10.1007/s12652-017-0485-5, pp. 1-12, 2017.

[50] S. Kumari, X. Li, F. Wu, A. K. Das, K. R. Choo and J. Shen, "Design of a Provably Secure Biometrics-based Multi-cloud-server Authentication Scheme," Future Generation Computer Systems, Elsevier, vol. 68, pp. 320-330, 2017.

[51] M. H. Ibrahim, S. Kumari, A. K. Das and V. Odelu,"Attribute-based Authentication on the Cloud for Thin Clients," The J. of Supercomputing, Springer, DOI: 10.1007/s11227-016-1948-8, pp. 1-33, 2017.

[52] D. Chattaraj, M. Sarma and A. K. Das, "A New Two-server Authentication and Key Agreement Protocol for Accessing Secure Cloud Services," Computer Networks, Elsevier, DOI: 10.1016/j.comnet.2017.12.007, vol. 131, pp. 144-164, 2018.

[53] S. Kalra and S. K. Sood, "Secure Authentication Scheme for IoT and Cloud Servers," Pervasive and Mobile Computing, vol. 24, pp. 210-223, 2015.

[54] V. Odelu, A. K. Das and A. Goswami, "A Secure Biometrics-based Multi-Server Authentication Protocol Using Smart Cards," IEEE Trans. Inf. Forensics Secur., vol. 10, no. 9, pp. 1953-1966, 2015.

[55] H. Shen, C. Z. Gao, D. D. He and L. B. Wu, "New Biometrics-based Authentication Scheme for Multi-server Environment in Critical Systems," J. Ambient Intell. Hum. Comput., vol. 6, no. 6, pp. 825-834, 2015.

[56] F. Wu, L. Xu, S. Kumari and X. Li, "A Novel and Provably Secure Biometrics-based Three-factor Remote Authentication Scheme for Mobile Client-Server Networks," Comput. Electr. Eng., Elsevier, vol. 45, pp. 274-285, 2015.

[57] D. Mishra, A. K. Das and S. Mukhopadhyay, "A Secure User Anonymity Preserving Biometrics-based Multi-server Authenticated Key Agreement Scheme Using Smart Cards," Expert Syst. Appl., Elsevier, vol. 41, no. 18, pp. 8129-8143, 2014.

[58] E. Yoon and K. Yoo, "Robust Biometrics-based Multi-server Authentication with Key Agreement Scheme for Smart Cards on Elliptic Curve Cryptosystem," J. Supercomput, Springer, vol. 63, no. 1, pp. 235-255, 2013.

[59] D. He and D. Wang, "Robust Biometrics-based Authentication Scheme for Multi-server Environment," IEEE System Journal, vol. 9, no. 3, pp. 816-823, 2015.

ملخص البحث:

من أجل إتاحة الخدمات السحابية [البرمجيات؛ المنصّة؛ البنية التحتية]، من الضروري إنشاء مفتاح متناظر بين المستخدم النهائي والخادم البعيد الخاص بالخدمات السحابية. وعندئذٍ، يتعين على الطرفين النهائيين أن يكون بينهما تدقيق مُحكم. ولتحقيق ذلك، هناك حاجة إلى آلية تصديق متينة. حيث أن البروتوكولات الخاصة بالتصديق بواسطة خادم واحد تتسم بالهشاشة أمام العديد من التهديدات المرتبطة بالأمان.

تقترح هذه الدراسة بروتوكول تصديق وتبادل للمفاتيح يتميز بالفعالية، يرتكز على كلمة السر ويستخدم خادمين. والبروتوكول المقترح يعالج أوجُه القصور في البروتوكولات القائمة. وقد أثبت التحقق الرسمي من البروتوكول المقترح، باستخدام التحقق الآلي من بروتوكولات أمان الإنترنت وتطبيقاتها، أنّ البروتوكول المقترح في هذه الدراسة آمن. كما برهن تحليل الأمان أن البروتوكول المقترح عالج القضايا القائمة بنجاح. وتشير دراسة الأداء أن تكلفة البروتوكول معقولة مقارنة بالبروتوكولات الأخرى المماثلة. و يمكن اعتبار البروتوكول المقترح بروتوكول تصديقٍ متيناً من أجل الوصول إلى الخدمات السحابية بأمان.