An Anonymous Authenticated Key Agreement Protocol Secure in Partially Trusted Registration Server Scenario for Multi-Server Architectures

Inam ul haq, Jian Wang, Youwen zhu, Sheharyar Nasir

Abstract—The accelerated advances in information communication technologies have made it possible for enterprises to deploy large scale applications in a multi-server architecture (also known as cloud computing environment). In this architecture, a mobile user can remotely obtain desired services over the Internet from multiple servers by initially executing a single registration on a trusted registration server (RS). Due to the hazardous nature of the Internet, to protect user privacy and online communication, a lot of multi-server authenticated-key-agreement (MSAKA) schemes have been furnished. However, all such designs lack in two very vital aspects, i.e., 1) no security under the partially trusted *RS* and 2) *RS* cannot control a user to access only a wanted combination of service-providing servers. To address these shortcomings, we present a new MSAKA protocol using self-certified public-key cryptography (SCPKC). We confirm the security of the proposed scheme by utilizing the well-known automated verification tool AVISPA and also provide a formal security proof in the random oracle model. Moreover, the software implementation of the proposed scheme, and a performance and security metrics comparison shows that it portrays a better security performance trade-off, and hence is more appropriate for real-life applications having resource constraint devices.

Index Terms—Multi-Server architecture, Partially trusted registration server, Mutual authentication, Key agreement, Self-Certified Public Keys.

1 Introduction

The Rapid growth of wired and wireless network communication technologies in the recent era has made the concept of the Internet of things (IoT) a reality. Internet-connected smart devices provide desirable services to their users ranging from daily life matters to more complicated e-commerce transactions. To provide robust and seamless services, service providers host a lot of application servers on the Internet. Historically, this client-server communication was started using single-server architecture, in which a single server was providing services to all registered users. However, widespread of internet-connected devices and an increasing number of online services, cause a computational bottleneck on the server end for providing seamless services. Due to this limitation, multi-server architectures came into place. In this setting, a user can acquire desired services from multiple servers with a single credential set. Fig. 1 depicts a generic multi-server architecture.

The communication over the Internet is susceptible to various security attacks because of malign parties. The activities of these evils may result in a violation of user privacy, financial frauds, and more. Due to this untrusted situation, authentication protocols were in place (single-server architecture) to protect the communication [1]. However, the transition from single-server to multi-server architecture also demands new authentication protocols for later architecture. This demand was inevitable as existing protocols were not flexible enough for use in multi-server architecture. The reasons for this were twofold; first, it is not practical for a user to register on multiple servers and remember complex passwords, and second, in case of two-factor authentication, it is inconvenient for him to carry and manage multiple physical tokens.

To address this problem, authentication protocols for multi-server architecture were started. In this setting, three participants are involved, a registration server *RS*, a small number of service providing servers, and numerous end-users. All the users and servers become part of multi-server architecture after completing a one-time registration process at *RS* end. This new paradigm has its challenges in terms of security and functional requirements, such as mutual authentication with offline *RS* (See Fig. 2), no burden of updating user information to registered servers, key compromise impersonation resistance, and various well-known attacks prevention [2]–[7]. Although every proposed protocol claimed that it fulfills all the security and functional requirements, however, it is evident from the history that the majority of the proposals were found insecure or inefficient, thus forming a break-fix-break chain [8]→ [9]→ [10] → [11] → [12].

The motivation of this research is to identify three very significant shortcomings in the existing MSAKA protocols and furnish a new MSAKA protocol free of these pitfalls. These shortcomings are:

- All the existing MSAKA protocols in the literature assume that *RS* is fully trusted. However, in reality, the registration services are performed by some commercial entities that can not be fully trusted. Therefore, the MSAKA protocol should restrict *RS* from impersonating any user or application server, even if, it maliciously store proscribed information during the registration phase.
- It is assumed that after getting registered, a user can access any service-providing server controlled by

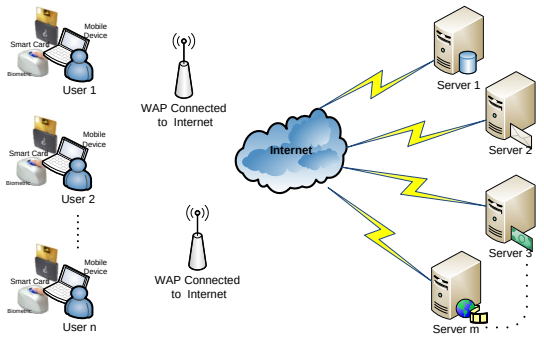


Fig. 1: Generic Architecture of Multi-Server Environment

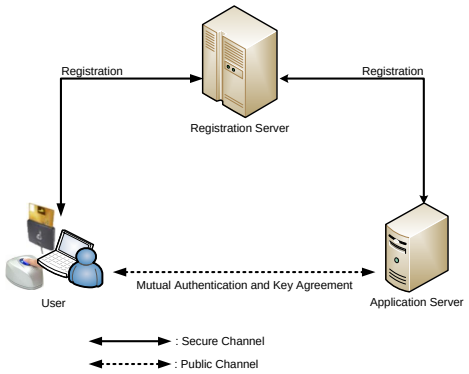


Fig. 2: System Model

RS. Nevertheless, in practical scenarios, users may belong to a different privilege level, thus entitled to access a limited set of servers. For the rest of this paper, we will use the term "flexible access control (FAC)" for this feature.

- The majority of the existing MSAKA protocols with offline RS, which are deemed secure to-date incorporates computationally expensive bilinear map operations. This computational expensiveness makes such schemes unsuitable for battery-limited devices.

Therefore, it is very vital to design an MSAKA protocol that satisfies all the existing security and functionality requirements as well as the above critical features.

1.1 Adversarial Model

- The link between the user and the service-providing server is open to an adversary. Due to this openness, an adversary can read, modify, and store ongoing communication to attack the proposed scheme. However, $\mathcal{A}$ gains no help from underlying cryptography, i.e., the cryptographic primitives used are secure.
- The smart card of the user can be stolen, and an adversary can obtain stored information using the power analysis attack [13].
- Using above mentioned powers an adversary can execute offline identity and password guessing attacks in succession.
- The *RS* is partially trusted, and an internal adversary (administrator of the RS) can store authorized information (during registration time) of any user or service-providing server to launch impersonation attacks at a later stage.
- The long term private keys of *RS* and other servers can not be compromised.

1.2 Related Work

To best of our knowledge, Li *et al.* [14] firstly proposed neural networks based multi-server authentication protocol in 2001. Next, in 2003, Lin *et al.* noticed the computational expensiveness of [14] involved in the training phase. They proposed an authentication protocol based on integer factorization cryptography featuring no verification table at the server end [15]. Later in 2006, Cao and Zhong [16] demonstrated that after observing the single authentication message, it is possible to impersonate any legal user in [15]. Next, a number of symmetric cryptography based MSAKA protocols were presented [17]–[21]. However, conforming strong user anonymity, perfect forward secrecy, and key compromise impersonation resilience in symmetric-key-settings is an arduous job. Due to these restrictions, MSAKA protocols based on public-key cryptography (PKC) remained dominant in this research domain. The security basis of these PKC-based protocols is a hard mathematical problem such as integer factorization problem, discrete logarithm (DL) problem, and Diffie-hellman (DH) problem.

In 2013, Tsai *et al.* presented an MSAKA protocol based on finite field cryptography (FFC) [22]. In this paper, they showed the vulnerability of Yeh-Lo's protocol [23] against undetectable password guessing attack and proposed their amended version. Next, Pippal *et al.* proposed an MSAKA protocol using FFC [24], however later in 2014, Wei *et al.* [25] pointed that attacks like user impersonation, privileged insider, and password guessing are possible in [24]. In 2015, He-Wang [26] presented an MSAKA protocol using ECC that achieves mutual authentication utilizing the registration server *RS*. Later, Odelu *et al.* [27] crypt-analyzed and demonstrated that [26] does not resist impersonation attack and session-specific information attack. [27] presented a new ECC based amended protocol in which a user can re-register himself using his old identity after revocation. In 2016, Tseng *et al.* [28] furnished a pairing-based MSAKA protocol which support dynamic revocation of compromised participants. Recently several ECC based protocols have been proposed, such as [12], [29], [30]. Similarly, several Chebyshev chaotic map cryptography-based MSAKA protocols were proposed during recent years [31]–[34]. All these protocols suffer from critical problems, e.g., [31] requires an online *RS*, [32] shares a pre-shared key to all application servers (no key compromise impersonation resilience), and [33] requires a permanent secure channel between *RS* and application servers for updating the user identity table.

The notion of the Self-certified public key (SCPK) was introduced by Marc Girault in 1991 [35]. The first SCPKC-based MSAKA protocols was proposed by Liao-Hsiao using bilinear maps [4]. Later, Hsieh-Leu found that [4] did not support the feature of unlinkability, and also require a permanent secure channel for updating the newly registered user data on all application servers. Following the cryptanalysis, they presented an improved protocol based on SCPKC

using bilinear maps. Subsequently, Amin-Biswas pointed out that [3] is susceptible to offline-identity guessing attack, offline-password guessing attack, and server masquerading attack [36]. Next, He *et al.* [2] proposed an efficient SCPKC based MSAKA protocol in which computationally expensive bilinear map operation was needed on the server end only. Recently, Ying-Nayak proposed a lightweight SCPKC-based multi-server authentication protocol for 5G networks. However, it is vulnerable to various well-known attack which make it unpractical for real life applications [37].

1.3 Contributions

- We propose the first MSAKA scheme, which is secure even if the *RS* is partially-trusted. Additionally, our design provides the FAC feature and does not involve expensive bilinear-map and map-to-point operations. The proposed design incorporates only ECC-based operations.
- We present proofs of session key semantic security, and security under partially trusted *RS*.
- We perform formal security verification of the proposed protocol using the well-known AVISPA tool. The output results show the security of the proposed scheme against active attacks.
- We implement the proposed protocol for both, the user-end (android platform) and the server-end (Linux platform).
- The performance analysis and comparison results concerning the security level of 112-bits show that the proposed protocol has good execution efficiency with enhanced features support, as compared with the latest related protocols.

1.4 Roadmap of the Paper

In the remaining paper, cryptographic primitives used in this paper are concisely discussed in Section 2. Section 3 elaborates the proposed MSAKA protocol, whereas, Section 4 provides its security analysis, accompanied by implementation details a performance comparison in Section 5. In the end, Section 6 concludes this paper.

2 Cryptographic Background

Cryptographic primitives and hard mathematical problems used in this paper are explained concisely in this section.

2.1 Fuzzy Extractor (FE)

A FE$(\mathcal{B}, e, l_b, t, \xi)$ function extracts a random key Ψ by taking a biometric template Bio as an input. The extracted secret key can be used for any cryptographic application. Fuzzy extractor [38] comprises of following two procedures:

Gen: A non-deterministic procedure takes input $Bio \in \mathcal{B}$ and yields a secret key $\Psi \in \{0,1\}^{l_b}$ and θ, which is assumed to be public. It is required that $\forall Bio \in \mathcal{B}$ having a minimum entropy e, if $(\Psi, \theta) \leftarrow Gen\,(Bio)$ then we have statistical distance $SD\{(\Psi, \theta), (R_{l_b}, \theta)\} \leq \xi$ where R_{l_b} denotes the random string of length l_b.

Rep: A deterministic procedure that recovers original secret key Ψ by taking a biometric template Bio' and public reproduction parameter θ. The reproduction procedure will be successful if and only if $dis\,(Bio, Bio') \leq t$ where t is a predefined threshold value.

TABLE 1: Nomenclatures

Term	Meaning	Term	Meaning
RS	Registration Server	Bio_i	U_i's biometric template.
x	Private key of *RS*	s_i	Private key of U_i
RS_{pb}	Public key of *RS*	SC_i	Authorized smart card of U_i
AS_j	j^{th} application server	SK_{ij}	Session key
SID_j	Identity of AS_j	G	Elliptic curve points group
ss_j	Private key of AS_j	P	Base point on elliptic curve
U_i	i^{th} user	$\mathcal{A}$	Adversary
ID_i	U_i's identity	$\oplus$	Bitwise xor operation
PW_i	U_i's password	$\|\|$	Concatenation operation

TABLE 2: Computation procedure of $2^m - 1$ unique numbers for $m = 6$

Pattern	Computation
0 0 0 0 0 1	$rn_6 \; mod \; n$
0 0 0 0 1 0	$rn_5 \; mod \; n$
0 0 0 0 1 1	$rn_5 \times rn_6 \; mod \; n$
0 0 0 1 0 0	$rn_4 \; mod \; n$
0 0 0 1 0 1	$rn_4 \times rn_6 \; mod \; n$
0 0 0 1 1 0	$rn_4 \times rn_5 \; mod \; n$
.	.
.	.
1 1 1 1 0 0	$rn_1 \times rn_2 \times rn_3 \times rn_4 \; mod \; n$
1 1 1 1 0 1	$r_1 \times rn_2 \times rn_3 \times rn_4 \times rn_6 \; mod \; n$
1 1 1 1 1 0	$r_1 \times rn_2 \times rn_3 \times rn_4 \times rn_5 \; mod \; n$
1 1 1 1 1 1	$r_1 \times rn_2 \times rn_3 \times rn_4 \times rn_5 \times rn_6 \; mod \; n$

2.2 Elliptic Curve Cryptography (ECC)

A non-singular elliptic curve [39] defined over a finite field F_q and q be a large prime is defined as:

$$E : y^2 = x^3 + ax + b$$

where $a, b \in F_q$ and $4a^3 + 27b^2 \neq 0$ holds.

The points on the curve along with abstract point $\mathcal{O}$ serving as identity element form an additive cyclic group G. A base point P with prime order n serve as the generator of the group. The scalar multiplication is achieved by repeated point addition.

Following are two well-known ECC-based hard problems.

ECDL Problem: Given P and $a \cdot P$ where $a \in [1, n-1]$, it is computationally very hard to find a.

ECDH Problem: Given P, $a \cdot P$ and $b \cdot P$, it is computationally impracticable to find $ab \cdot P$ for some $a, b \in [1, n-1]$.

3 Proposed Protocol

This section elaborates on the proposed protocol, which consists of four phases. These are system setup, user registration, application server registration, mutual authentication & key agreement, and user credentials update. Table 1 summarizes all the terminologies used in this paper.

3.1 System Setup Phase

Registration server *RS* initializes the protocol by executing the following steps:

Setup1: Chooses an elliptic curve $E(F_q)$ over a finite field F_q, where q is a large prime. It further selects a base point P of prime order n.

Setup2: Randomly selects $x \in_R [1, n-1]$ as its private key and computes matching public key $RS_{pb} = x \cdot P$.

Setup3: For registering m service-providing servers, *RS* pre-computes an access control database (ACDB) having $2^m - 1$ unique numbers. For this, 1) it generates m random numbers, i.e., $\{rn_1, rn_2, ...rn_m\}$ (a random number for each server), 2) list down all possible combinations of $m - bits$, and 3) against each pattern it calculates a new number by multiplying the random numbers where the corresponding bit is 1. In this way, a total of $2^m - m - 1$ unique numbers are calculated. These computed numbers, along with generated random numbers, constitute the complete ACDB (see Table 2).

Setup4: *RS* chooses six hash functions $h : \{0,1\}^* \rightarrow Z_n^*$, $h_1 : G \rightarrow \{0,1\}^*$, $h_3 : \{0,1\}^* \times G \times \{0,1\}^* \times G \rightarrow Z_n^*$, $h_4 : \{0,1\}^* \times G \times G \times G \rightarrow Z_n^*$, $h_5 : G \times G \times \{0,1\}^* \times G \rightarrow \{0,1\}^*$, and $h_6 : G \times G \times \{0,1\}^* \times G \rightarrow Z_n^*$.

Setup5: Finally *RS* keeps x and ACDB secret, whereas publishes all other parameters $\{E, q, n, P, RS_{pb}, h, h_1, h_3, h_4, h_5, h_6\}$ publicly.

3.2 User Registration Phase

User Reg1: User selects his unique identity ID_i, password PW_i, random number $h_i \in_R [1, n-1]$ and imprints his biometric template Bio_i on a suitable biometric device.

User Reg2: Computes $F_i = h_i \cdot P$, $Gen(Bio_i) = (\Psi_i, \theta_i)$ and $MPW_i = h\left(PW_i || \Psi_i\right)$. The user U_i transmits the tuple $\langle ID_i, MPW_i, F_i \rangle$ to *RS* using a secure channel.

User Reg3: *RS* first verifies the ID_i to avoid duplicate entries in user registration database (UDB). It asks the user to select a different identity value, if there already exists a registered user with the same identity.

User Reg4: Next, according to the user privilege, *RS* decides the set of allowed server(s) and select the unique number T_i from the ACDB, e.g., if user is only allowed to access first four servers then number against pattern 111100 will be selected as T_i.

User Reg5: Selects $k_i \in_R [1, n-1]$, and computes:

$$R_i = k_i \cdot F_i$$
$$m_i = h(ID_i || T_i) x k_i^{-1} \; mod \; n$$
$$A_i = T_i \oplus MPW_i$$
$$C_i = h(R_i || T_i)$$

User Reg6: Finally, *RS* stores the tuple $\langle ID_i, h_1(R_i) \rangle$ in UDB and writes the information $\langle R_i, A_i, C_i, m_i \rangle$ on a smart card and securely delivers it to the user U_i.

User Reg5: On receiving smart card from *RS*, U_i computes his private key $s_i = [m_i h_i^{-1}] \; mod \; n$. Next, U_i calculates $V_i = h(ID_i) \oplus \theta_i$, $ES_i = s_i \oplus h\left(ID_i || PW_i || \Psi_i\right)$ and $C_i = h(C_i || s_i)$. Finally, user writes V_i and ES_i on SC_i, discards m_i, and replaces the existing C_i value with the new one.

Remark: If a registered user wants to change his privilege level (wants to access more or less servers) after some time, then he has to re-register with the RS by following these steps.

3.3 Application Server Registration Phase

App Srvr Reg1: Application server AS_j chooses its unique identity SID_j and a random number $r_j \in_R [1, n-1]$. It then computes $Y_j = r_j \cdot P$ and sends $\langle SID_j, Y_j \rangle$ to *RS* using a secure channel.

App Srvr Reg2: *RS* first validates the SID_j to avoid duplicate entries in the application server registration database (ADB). If SID_j is fresh, it randomly selects z_j and computes $Q_j = z_j \cdot Y_j$, and $n_j = h(SID_j) x z_j^{-1} \; mod \; n$.

App Srvr Reg3: Next, it assigns a pre-generated random number rn_j (generated during setup phase) to AS_j and construct a server-specific access control database (SACDB) by only picking the entries of ACDB where rn_j was used in computation. So in this way a total of 2^{m-1} entries will be selected for each server.

App Srvr Reg4: Finally, *RS* transmits $\langle Q_j, n_j \rangle$ and SACDB to AS_j over a secure channel. Furthermore, it stores the tuple $\langle SID_j, h_1(Q_j) \rangle$ in the ADB and publish the pair $\{SID_j, Q_j\}$ in a public repository for all registered users.

App Srvr Reg5: Upon receiving the message from *RS*, application server AS_j computes $ss_j = r_j^{-1} n_j \; mod \; n$. Furthermore, it securely stores the ss_j and SACDB.

Fig. 3 illustrates the registration process of both user and application server.

3.4 Mutual Authentication & Key Agreement Phase

MAKA1: Whenever U_i wants to access an application server AS_j, he plugs the SC_i into a smart card reader and keys in $\langle ID_i', PW_i', Bio_i' \rangle$. Smart card SC_i computes $\theta_i' = V_i \oplus h(ID_i')$, $\Psi_i' = Rep(Bio_i', \theta_i')$, $T_i' = h\left(PW_i' || \Psi_i'\right) \oplus A_i$, $s_i' = h\left(ID_i' || PW_i' || \Psi_i'\right) \oplus ES_i$ and $C_i' = h\left(h\left(h(R_i || T_i')\right) || s_i'\right)$. It then checks whether $C_i' = C_i$ and rejects the user if both are not equal.

MAKA2: In case of equal, it selects a random number $\alpha_i \in_R [1, n-1]$ and computes $\gamma_{ij} = \alpha_i h(SID_j) \cdot RS_{pb}$, $M_i = (ID_i' || R_i || T_i) \oplus h_1(\gamma_{ij})$, $D_{ij} = \alpha_i \cdot Q_j$ and $N_i = h_3\left(ID_i' || R_i || T_i' || \gamma_{ij}\right)$. Finally, SC_i sends $Msg_1 = \langle M_i, D_{ij}, N_i \rangle$ to AS_j over a public channel.

MAKA3: On receipt of Msg_1, AS_j computes $\gamma_{ij}' = ss_j \cdot D_{ij}$, $(ID_i' || R_i || T_i') = h_1(\gamma_{ij}') \oplus M_i$. It then computes $N_i' = h_3\left(ID_i' || R_i || T_i' || \gamma_{ij}'\right)$ and compares it with the received N_i and decline the request, if both are different. Next, it verifies whether received T_i value exist in its SACDB, and refuses the user if not.

MAKA4: After successful verification, AS_j generates a random number $\delta_j \in_R [1, n-1]$ and computes $W_{ji} = \delta_j \cdot R_i$, $\Omega_j = \delta_j \cdot P$ and $N_j = h_4\left(T_i' || W_{ji} || \Omega_j || \gamma_{ij}'\right)$. Finally, AS_j transmits $Msg_2 = \langle W_{ji}, N_j, \Omega_j \rangle$ to U_i via a public channel.

MAKA5: Upon receiving Msg_2, U_i calculates $N_j' = h_4\left(T_i' || W_{ji} || \Omega_j || \gamma_{ij}\right)$, compares it with the received N_j and terminates the session if the comparison is not equal. If both are equal, U_i considers the application server AS_j is legitimate. Next, U_i calculates $Auth_i = s_i' \cdot W_{ji}$, $KM_{ij} = \alpha_i \cdot \Omega_j$, $\beta_i = \alpha_i \cdot P$, $SK_{ij} = h_5\left(\gamma_{ij} || KM_{ij} || T_i' || Auth_i\right)$ and $AT_i = h_6\left(\gamma_{ij} || Auth_i || SK_{ij} || \beta_i\right)$. At the end, U_i sends $Msg_3 = \langle AT_i, \beta_i \rangle$ to AS_j over a public channel.

MAKA6: Upon receiving Msg_3, AS_j computes $Auth_i' = [\delta_j h(ID_i' || T_i)] \cdot RS_{pb}$, $KM_{ij} = \delta_j \cdot \beta_i$, $SK_{ij} = h_5\left(\gamma_{ij}' || KM_{ij} || T_i' || Auth_i'\right)$ and $AT_i' = h_6\left(\gamma_{ij}' || Auth_i' || SK_{ij} || \beta_i\right)$. It then checks the equivalence of AT_i and AT_i' and aborts the session if the result is unequal. Otherwise, AS_j believes U_i is authentic and uses SK_{ij} for future correspondence.

This phase has been elaborated in Fig. 4.

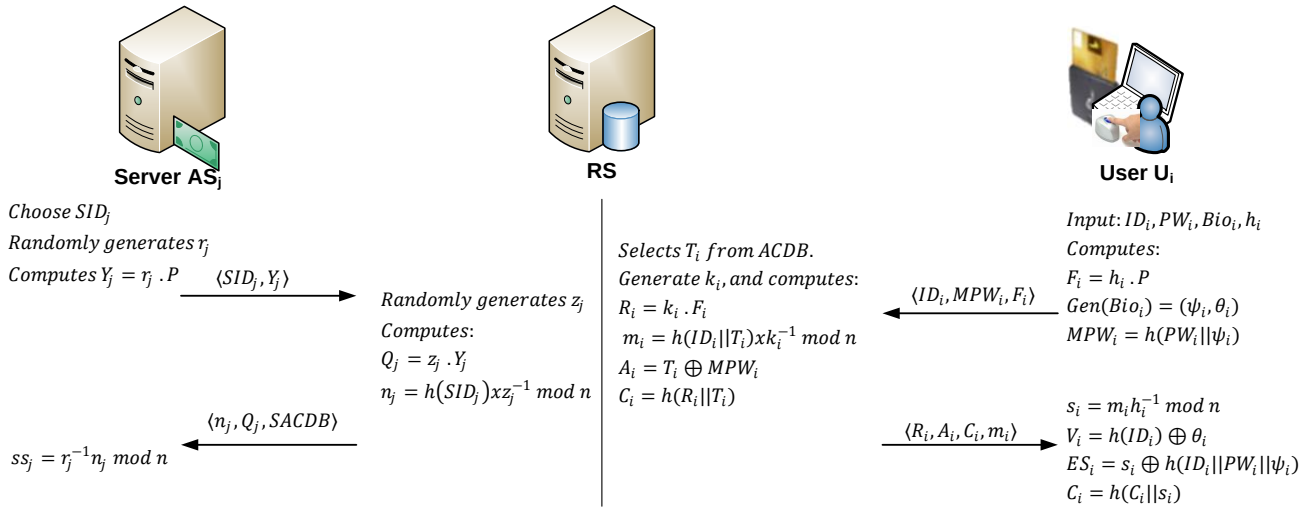


Fig. 3: Registration process of User U_i and server AS_j

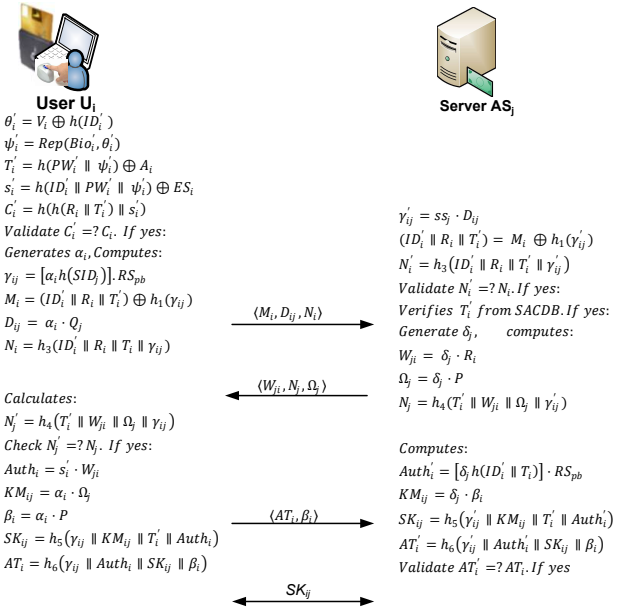


Fig. 4: Mutual Authentication & Key Agreement phase

3.5 User Credential Update Phase

At whatever time user U_i requires to change his password or/and biometric template, he can do it without involving *RS*, by performing the following steps:

Cred Upd1: The user plugs the SC_i into a suitable card reader and keys in $\langle ID_i^{old}, PW_i^{old}, Bio_i^{old} \rangle$. SC_i validates the legitimacy of credentials, as detailed in Section 3.4 (MAKA1).

Cred Upd2: If the result of the verification is successful, SC_i asks the user to provide a new PW_i^{new} and Bio_i^{new}. Next, it computes $Gen(Bio_i^{new}) = (\Psi_i^{new}, \theta_i^{new})$, $A_i^{new} = T_i' \oplus h\left(PW_i^{new} || \Psi_i^{new}\right)$, $V_i^{new} = h(ID_i^{old}) \oplus \theta_i^{new}$, and $ES_i^{new} = s_i \oplus h\left(ID_i^{old} || PW_i^{new} || \Psi_i^{new}\right)$. Finally, SC_i replaces V_i, A_i, and ES_i with V_i^{new}, A_i^{new}, and ES_i^{new}, respectively.

Remark: A user can change his password or biometric template independently.

3.6 Proof of Correctness

The proof of correctness for the computations, i.e., 1) $ss_j \cdot D_{ij} = \gamma_{ij}$ and 2)$Auth_i' = [\delta_j h(ID_i' || T_i)] \cdot RS_{pb}$ done by AS_j is as follows:

$$
\begin{array}{l|l}
ss_j \cdot D_{ij} & Auth_i \\
= r_j^{-1} n_j \alpha_i \cdot Q_j & = s_i \cdot W_{ji} \\
= n_j \alpha_i z_j \cdot P & = s_i \delta_j \cdot R_i \\
= h(SID_j) x z_j^{-1} \alpha_i z_j \cdot P & = m_i h_i^{-1} \delta_j k_i h_i \cdot P \\
= h(SID_j) x \alpha_i \cdot P & = h(ID_i || T_i) x k_i^{-1} \delta_j k_i \cdot P \\
= \alpha_i h(SID_j) \cdot RS_{pb} & = [\delta_j h(ID_i || T_i)] \cdot RS_{pb} \\
= \gamma_{ij} & = Auth_i'
\end{array}
$$

4 Security Analysis

This section presents a security model of the proposed protocol mainly adopted from the work of [40]–[42] and a formal security proof using a sequence of games technique [43]. Moreover, this section also elaborates the automated formal verification using AVISPA tool.

4.1 Security Model

Participants

During registration phase, the application server stores the parameters $\{SID_j, ss_j, SACDB\}$, whereas the smart card of U_i contains $\{R_i, A_i, C_i, E_i, V_i, ES_i\}$. The parameters on smart card are protected with password PW_i (selected from a Zipf distributed dictionary D_t with size $|D_t|$ and Zipf parameters C' and s' [44], [45], biometric template Bio_i and identity ID_i. However, during mutual authentication & key agreement phase of the proposed protocol, only U_i and AS_j engage in communication to establish a session key. Let $\Pi_{U_i}^u$ and $\Pi_{AS_j}^s$ be the instances u and s of $U_i \in Users$ and $AS_j \in Servers$ respectively. These are also known as oracles. Furthermore, we denote any kind of instance with $\Pi^t \in Users \cup Servers$.

Partnering

Instances Π^u and Π^s are said to be partner, if they agreed on the same session id (sid) where $sid \neq NULL$ (This mean

both partners have mutually authenticated each other). The session id is built by arranging all sent/received messages between instances Π^u and Π^s before they go into an accepted state.

Freshness

As long as, session key SK_{ij} is secure, instance Π^u or Π^s remains fresh. This means that both participants are in accept state and *Reveal*(Π^t) query has never been asked to any of the participants.

Adversary

Adversary $\mathcal{A}$ has full control of communication channel [46], i.e., he can intercept, store, modify and delete any message. Furthermore, $\mathcal{A}$ is capable of extracting stored information from user smart card (after stealing) using side channel attacks [13], [47].

$\mathcal{A}$ can ask different oracle queries by interacting with any participant instance Π^u or Π^s. There are three possible outcomes of any oracle query. These are **accept** (oracle receive correct message), **reject** (oracle receive wrong message) and no conclusion (oracle receive no message). To attack the proposed multi-server authenticated key agreement protocol say $\mathcal{P}$, adversary $\mathcal{A}$ can ask following queries:

Execute(Π^u, Π^s): $\mathcal{A}$ uses this query to intercept ongoing communication between U_i and AS_j. So in a practical scenario, $\mathcal{A}$ uses this query to launch an eavesdropping attack on $\mathcal{P}$.

Send(Π^t, MSG)*:* Using this query, $\mathcal{A}$ sends a fabricated message MSG to a legitimate participant Π^t and gets a reply as per the specification of the proposed protocol.

Reveal(Π^t): $\mathcal{A}$ uses this query to acquire knowledge of session key SK_{ij} established during the current session among Π^t and its partner. In case no session key is defined for instance Π^t or a test query was asked to (Π^t) or its partner, then oracle returns undefined symbol $\perp$ to $\mathcal{A}$.

Corruptuser $(\Pi^u_{U_i}, z)$: This query model the three-factor security of $\mathcal{P}$. The adversary runs this query for three different values of z. For $z = 0$ and $z = 1$, oracle responds with $\Pi^u_{U_i}$'s password PW_i and biometric template Bio_i respectively. For $z = 2$, oracle answered with all the information stored in $\Pi^u_{U_i}$'s smart card.

Test(Π^t)*:* Following the indistinguishability of the random oracle model, this query model the semantic security of SK_{ij} established between U_i and AS_j. The output of this query is used to measure the strength of the session key SK_{ij}.

Semantic Security of $\mathbf{SK_{ij}}$

We say that the adversary is successful in breaking the semantic security of the session key if it successfully distinguishes between a real session key and a random key (of the same length) during an experiment. For this purpose, $\mathcal{A}$ can ask a single *Test* query to any participant instance. At the end of the experiment, if the $\mathcal{A}$ guessed bit C'_n is equal to C_n then, $\mathcal{A}$ wins the game. Suppose EV represents an event when $\mathcal{A}$ is successful in a game. Then the advantage of $\mathcal{A}$ in breaking the session key security of the proposed protocol $\mathcal{P}$ can be defined as:

$$Adv^{maka}_{\mathcal{P}}(\mathcal{A}) = |2.Pr[EV] - 1| = |2.Pr[C_n = C'_n] - 1|$$

Where, $Pr[EV]$ represents probability of event EV. Proposed protocol $\mathcal{P}$ is considered to be secure in proposed security model, if the advantage of $\mathcal{A}$ is $\leq \epsilon$ where $\epsilon > 0$ and negligible.

Random Oracle

A cryptographic hash function has been modeled as random oracle HO. All participants including $\mathcal{A}$ can ask queries to HO.

Definition: A three-factor multi-server authenticated key agreement protocol is semantically secure, if the advantage $Adv^{maka}_{\mathcal{P}}(\mathcal{A})$ of an adversary is larger than $max\{C' \cdot q^{s'}_{snd}, q_{snd}(\frac{1}{2^{b_k}}, \epsilon_{fp})\}$ and negligible, where q_{snd} represents number of send queries, bk represent the length of biometric key, and ϵ_{fp} denotes the probability of biometric false positive, respectively.

4.2 Formal Security Proof

Theorem 1. Assume a polynomial-time adversary $\mathcal{A}$ running alongside the $\mathcal{P}$ in random oracle with consistent password dictionary D_t and $|H_l|$ represents the range space of hash function. The adversary is allowed to make at most q_{snd} times Send queries, q_{ex} times Execute queries, and q_H times hash queries, respectively. Then $\mathcal{A}$ has approximately the following advantage in violating the session key security.

$$Adv^{maka}_{\mathcal{P}}(\mathcal{A}) \leq \frac{q_H^2+8q_H}{|H_l|} + 2max\{C' \cdot q^{s'}_{snd}, q_{snd}(\frac{1}{2^{b_k}}, \epsilon_{fp})\} + \frac{(q_{snd}+q_{ex})^2+6q_{snd}}{2^{l_r}} + 2q_H Adv^{ECDL}_{\mathcal{A}}(t) + 2q_H(q_{snd}+q_{ex})^2 Adv^{ECDH}_{\mathcal{A}}(t_1)$$

Where $Adv^{ECDL}_{\mathcal{A}}(t)$ and $Adv^{ECDH}_{\mathcal{A}}(t_1)$ denote adversary advantage in solving the ECDL and ECDH problems in time t and t_1, respectively.

Proof: The proof follows a sequence of six games $Game_i$ where $i \geq 0 \leq 5$. Let EV_i is an event, where $\mathcal{A}$ successfully guesses the random bit C_n in Test query for $Game_i$.

$\mathbf{Game_0}$: This is a real attack launched by $\mathcal{A}$ against the proposed protocol $\mathcal{P}$. Since bit C_n has been chosen randomly at the start of the $Game_0$ so by definition it follows:

$$Adv^{maka}_{\mathcal{P}}(\mathcal{A}) = |2Pr[EV_0] - 1|. \quad (1)$$

$\mathbf{Game_1}$: In this game, random oracle HO is simulated by maintaining a list L_H. All other oracle queries are simulated as real attacks. Clearly, the real protocol execution in $Game_0$ and the simulation in this game are indistinguishable. Hence, we got:

$$Pr[EV_1] = Pr[EV_0]. \quad (2)$$

$\mathbf{Game_2}$: During this game, we simulate all the oracles as in $Game_1$, but we halt the execution in case of collision. There are two possible cases in which collision can occur. These are:

- The collision can happen in the hash oracle queries.
- A collision can occur in the transcript $(\langle M_i, D_{ij}, N_i\rangle, \langle W_{ji}, N_j, \Omega_j\rangle, \langle AT_i, \beta_i\rangle)$, which denotes collision in random numbers α_i and δ_j.

in case of such collisions, $\mathcal{A}$ can win the game by launching a replay attack. According to birthday paradox, the maximum probabilities of case1 and case 2 are $\frac{q_H^2}{2|H_l|}$ and $\frac{(q_{snd}+q_{ex})^2}{2^{l_r+1}}$. So, $Game_1$ and this game are indistinguishable unless these cases of collisions occur. This gives us:

$$|Pr[EV_2] - Pr[EV_1]| \leq \frac{q_H^2}{2|H_l|} + \frac{(q_{snd}+q_{ex})^2}{2^{l_r+1}}. \quad (3)$$

Game$_3$: In this game all the oracles in $Game_2$ are simulated, however we abort the execution, wherein $\mathcal{A}$ luckily guess the correct message transcript (without making the corresponding hash query). Since in the proposed protocol $\mathcal{P}$, three messages are exchanged among U_i and AS_j, following three cases are possible:

a Considering $Send(AS_j, Msg_1)$ query, the hash value, i.e., $N_i = h_3(ID_i'||R_i||T_u'||\gamma_{ij})$ must match; otherwise AS_j will terminate the session. The maximum probability of this event is $\frac{q_H}{|H_l|}$. The probability of successful guessing of message Msg_1 is at most $\frac{q_{snd}}{2^{l_r}}$.

b For $Send(U_i, Msg_2)$ oracle query, the hash value $N_j = h_5(T_u||W_{ji}||\Omega_j||\gamma_{ij}')$ must hold and for this maximum probability is $\frac{q_H}{|H_l|}$. Accordingly, the probability of successful guessing of message Msg_2 is at most $\frac{q_{snd}}{2^{l_r}}$.

c For authentication response, i.e., $Send(AS_j, Msg_3)$ query, the hash value $AT_i = h_6(\gamma_{ij}||Authi_i||SK_{ij}||\beta_i)$ must hold with at most probability $\frac{2q_H}{|H_l|}$.

Unless the oracle aborts the game with legitimate values, $Game_2$ and this game are indistinguishable to adversary. So, considering all above three cases, we have:

$$|Pr[EV_3] - Pr[EV_2]| \leq \frac{3q_{snd}}{2^{l_r}} + \frac{4q_H}{|H_l|} \quad (4)$$

Game$_4$:In this game, we consider all the possible attacks executed by $\mathcal{A}$ by simulating all the oracles as in $Game_3$. If adversary successfully gets the real session key SK_{ij}, then ECDL problem can be solved in polynomial time. Since, proposed protocol $\mathcal{P}$ provides three-factor security, therefore guessing of both PW_i and Bio_i need to be considered. For this purpose, $\mathcal{A}$ required all the parameters stored in SC_i. Suppose, adversary executes *Corruptuser* $(\Pi_{U_i}^u, 2)$ and gets all the required information. For other values, following are three possible cases:

1 Adversary gets the Bio_i by running the *Corruptuser* $(\Pi_{U_i}^u, 1)$ query and guesses (online) the PW_i by choosing it from D_t with a maximum allowed send queries. The probability of this case is $C' \cdot q_{snd}^{s'}$ [1].

2 Adversary runs *Corruptuser* $(\Pi_{U_i}^u, 0)$ query to get the pasword PW_i. Then, $\mathcal{A}$ can guess the biometric key of the user by adopting any one of the following two approaches:

- $\mathcal{A}$ randomly selects a biometric key ψ_i. The maximum probability for this case with at most q_{snd} *Send* queries is $\frac{q_{snd}}{2^{b_k}}$.
- $\mathcal{A}$ tries with a different biometric template Bio_i^* and a false positive event can happen with probability ϵ_{fp}.

3 In order to break the session key $SK_{ij} = h_5(\gamma_{ij}||KM_{ij}||T_u||Auth_i)$, adversary is required to compute $\gamma_i = ss_j \cdot D_i$ and $Auth_i = s_i \cdot W_j$ which both are instances of ECDL problem. So, in this case to launch an offline guessing attack, $\mathcal{A}$ queries either *Corruptuser* $(\Pi_{U_i}^u, 0)$ or *Corruptuser* $(\Pi_{U_i}^u, 1)$ along with *Corruptuser* $(\Pi_{U_i}^u, 2)$ query. So, we can say that $\mathcal{A}$ asks either pure *Execute*(Π^u, Π^s) or consecutive *Send* queries with hash oracle HO. The advantage of $\mathcal{A}$ in solving ECDL problem in this case is $q_H Adv_{\mathcal{A}}^{ECDL}(t)$.

The simulation of $Game_3$ and this game are indistinguishable with above mentioned guessing attacks. This give us

$$|Pr[EV_4] - Pr[EV_3]| \leq max\{C' \cdot q_{snd}^{s'}, q_{snd}(\frac{1}{2^{b_k}}, \epsilon_{fp})\} + q_H Adv_{\mathcal{A}}^{ECDL}(t) \quad (5)$$

Game$_5$: The purpose of this last game is to verify the forward secrecy of $\mathcal{P}$. Here we assume that adversary has asked *Corrupt*$(U_i^{s_i}/AS_j^{ss_j}/RS^x)$ and simulates *Execute*, *Send* and HO oracle queries on only old transcripts of $\mathcal{P}$ followed by a *Test* query. In this case, $\mathcal{A}$ needs to calculate $\alpha_i \cdot \omega_j \cdot P$ with the available values β_i, Ω_j and P, which is equivalent to solving a ECDH problem. The probability of having correct α_i and ω_j in the same session is $\frac{1}{(q_{snd}+q_{ex})^2}$. Finally, this gives us

$$|Pr[EV_5] - Pr[EV_4]| \leq q_H(q_{snd}+q_{ex})^2 Adv_{\mathcal{A}}^{ECDH}(t_1) \quad (6)$$

Regarding all above games, it is obvious that $\mathcal{A}$ obtains no advantage. So, we can say that this game is identical to real case and $Pr[EV_5] = \frac{1}{2}$.

By applying the triangular inequality, We have the following

$$\begin{aligned} |Pr[EV0] - \tfrac{1}{2}| &= |Pr[EV_1] - Pr[EV_5]| \\ &\leq |Pr[EV_1] - Pr[EV_2]| \\ &\quad + |Pr[EV_2] - Pr[EV_5]| \\ &\leq |Pr[EV_1] - Pr[EV_2]| \\ &\quad + (|Pr[EV_2] - Pr[EV_3]| \\ &\quad + |Pr[EV_3] - Pr[EV_5]|) \\ &\leq [EV_1] - Pr[EV_2]| \\ &\quad + (|Pr[EV_2] - Pr[EV_3]| \\ &\quad + |Pr[EV_3] - Pr[EV_4]|) \\ &\quad + |Pr[EV_4] - Pr[EV_5]|) \end{aligned} \quad (7)$$

By utilizing the equations from (1) to (7) we got.

$$\begin{aligned} \tfrac{1}{2}Adv_{\mathcal{P}}^{maka}(\mathcal{A}) = \; & |Pr[EV0] - \tfrac{1}{2}| \\ \leq \; & \tfrac{q_H^2}{2|H_l|} + \tfrac{(q_{snd}+q_{ex})^2}{2^{l_r+1}} \\ & + \tfrac{3q_{snd}}{2^{l_r}} + \tfrac{4q_H}{|H_l|} \\ & + \; max\{C' \cdot q_{snd}^{s'}, q_{snd}(\tfrac{1}{2^{b_k}}, \epsilon_{fp})\} \\ & + q_H Adv_{\mathcal{A}}^{ECDL}(t) \\ & + q_H(q_{snd}+q_{ex})^2 Adv_{\mathcal{A}}^{ECDH}(t_1) \end{aligned} \quad (8)$$

By simplifying both side of the above equation and rearranging the terms, we got the final result as presented in theorem 1. □

Theorem 2. Assuming that the ECDL problem is intractable in polynomial time, then registration server *RS* can not

impersonate any user or application server in $\mathcal{P}$, even if it has stored all the information during registration phases of $\mathcal{P}$.

Proof: During registration of server AS_j, *RS* receives $\langle SID_j, Y_j \rangle$ from AS_j where $Y_j = r_j \cdot P$. Next, it computes $Q_j = z_j \cdot Y_j$ and $n_j = h(SID_j) x z_j^{-1} \, mod \, n$. Let suppose, *RS* stores all the information, i.e., $\{SID_j, Y_j, Q_j, z_j, n_j\}$ in its memory. Now, registration server acting as an application server AS_j receive login message $\langle M_i, D_{ij}, N_i \rangle$ from user U_i. According to the specification of $\mathcal{P}$, *RS* has to compute $\gamma_{ij} = ss_j \cdot D_i$, where $ss_j = r_j^{-1} n_j \, mod \, n$. To calculate ss_j, *RS* needs to find r_j from given Y_j and P, which is equivalent to solving the ECDL problem. Hence *RS* can not impersonate any application server in the proposed protocol.

Similarly, during the registration of the user U_i, *RS* stores the information $\{ID_i, MPW_i, m_i, k_i, F_i, R_i\}$ in its memory, where $F_i = h_i \cdot P$. Assume that *RS* acting as user U_i selects a random number $\alpha_i^{RS} \in_R [1, n-1]$ and computes $\gamma_{ij}^{RS} = \alpha_i^{RS} h(SID_j) \cdot RS_{pb}$, $M_i^{RS} = (ID_i || R_i || T_i) \oplus h_1(\gamma_{ij}^{RS})$, $D_{ij}^{RS} = \alpha_i^{RS} \cdot Q_j$, and $N_i^{RS} = h_3\left(ID_i || R_i || T_i || \gamma_{ij}^{RS}\right)$. Finally, it sends $Msg_1 = \langle M_i^{RS}, D_{ij}^{RS}, N_i^{RS} \rangle$ to AS_j over a public channel. Since, Msg_1 is according to the specification of $\mathcal{P}$, application server AS_j will accept it, and respond with $Msg_2 = \langle W_{ji}, \Omega_j, N_j \rangle$. In order to successfully impersonate the user U_i, *RS* has to compute $Auth_i^{RS} = s_i \cdot W_{ji}$, where $s_i = h_i^{-1} m_i \; mod \; n$. So, to calculate s_i *RS* need to compute h_i from the available information F_i and P, which is an instance of computationally intractable ECDH problem. Hence, *RS* can not impersonate a legal user U_i in the proposed protocol. □

4.3 Automated Verification Using AVISPA Tool

Automated formal verification using any well-known tool proves the safety of AKA protocol against active attacks. The proposed scheme has been verified by utilizing the security protocol animator for AVISPA (SPAN), a platform for well-known automated verification tool AVISPA. In SPAN, we specify our protocol in high-level protocol specification language (HLPSL), and then these specifications are translated in a machine-readable intermediate format. The AVISPA then verifies the machine-readable instruction using any one of its back-ends(OFMC, CL-AtSe, SATMC, and TA4SP). The participants of the protocol are represented as a basic role (role_Ui, role_RS, and role_ASj) in HLPSL implementation.

We have implemented both registration phases and mutual authentication & key agreement phase. The actions of each participant are specified in its corresponding role, and then a composed session role is utilized to execute these basic roles in parallel. Finally, a mandatory top-level role known as environment role is defined, in which we specify all the global constants, session role(s), knowledge of the intruder, and desired security goals. Fig. **??** and Fig . 6 depict the specifications of server role and environment role, respectively.

Among four of the available back-ends, we have used OFMC and CL-AtSe for verification of our scheme. The simulation results, as illustrated in Fig. 7 ensures the security of the proposed protocol against active attacks.

```
role role_ASj(ASj:agent,Ui:agent,RS:agent,H:hash_func,F:hash_func,
              RSpb:public_key,P:text,Qj:public_key,Ri:public_key,
                Key_set_ASj_RS:(symmetric_key)
                set,Key_set_RS_ASj:(symmetric_key) set,SND,RSV:channel(dy))
played_by ASj
def=
  local
        State:nat,Alphai:text,IDi:text,SIDj:text,
        Deltaj:text,Key_2:symmetric_key,Key_1:symmetric_key,Ti:text,
        Rj:text,K:symmetric_key,
        SK:hash(symmetric_key.hash(text.hash(text.text)).text.text)
  init
        State := 0
  transition
      3. State=0   /\ RSV(start) =|>
         State':=1 /\ Key_1':=new()
                   /\ Key_set_ASj_RS':=cons(Key_1',Key_set_ASj_RS)
                   /\ SIDj':=new()  /\Rj':=new()
                   /\ SND({SIDj.F(Rj'.P)}_Key_1')
      4. State=1   /\ in(Key_2',Key_set_RS_ASj)
                   /\ RSV({Qj}_Key_2') =|>
         State':=2 /\ Key_set_RS_ASj':=delete(Key_2',Key_set_RS_ASj)
      5. State=2   /\RSV({K}_Qj.xor(IDi',K).xor(Ri,K).xor(Ti',K).H(IDi'.Ri.Ti'.K))=|>
         State':=3 /\ secret(IDi',sec_uid,{Ui}) /\secret(K,sec_k,{Ui,ASj})
                   /\ Deltaj':=new() /\ witness(ASj,Ui,auth_s_u,Deltaj')
                   /\ SND({Deltaj'}_Ri.F(Deltaj'.P).H(Ti'.Deltaj'.F(Deltaj'.P).K))
      7. State=3   /\ RSV({Alphai'}_Qj.H(K.Deltaj.SK'.F(Alphai'.P)).F(Alphai'.P))=|>
         State':=4 /\ SK':=H(K.F(Alphai'.F(Deltaj.P)).Ti.Deltaj)
                   /\secret(SK',sec_sk,{Ui,ASj})
                   /\ secret(IDi,sec_uid,{Ui})/\secret(K,sec_k,{Ui,ASj})
                   /\ request(ASj,Ui,auth_u_s,Alphai')
end role
```

Fig. 5: The specifications of server role in HLPSL

```
role environment()
def=
  const
        u:agent,hash_0:hash_func,hash_1:hash_func,const_1:text,
        s:agent,r:agent,h:hash_func,f:hash_func,
        const_1:text,sec_uid:protocol_id,sec_upw:protocol_id,
        sec_ubio:protocol_id,ri:public_key,qj:public_key,
        rspb:public_key,p:text,sec_k:protocol_id,
        auth_u_s:protocol_id,auth_s_u:protocol_id,sec_sk:protocol_id
  intruder_knowledge = {u,s,r,h,f,qj,rspb,p}
  composition
        session1(r,u,s,h,f,rspb,p,qj,ri,{},{},{},{})
        /\ session2(r,i,s,h,f,rspb,p,qj,ri,{},{},{},{})
        /\ session3(r,u,i,h,f,rspb,p,qj,ri,{},{},{},{})
end role
goal
        secrecy_of sec_uid
        secrecy_of sec_upw
        secrecy_of sec_ubio
        secrecy_of sec_k
        secrecy_of sec_sk
        authentication_on auth_u_s
        authentication_on auth_s_u
end goal
environment()
```

Fig. 6: The specifications of mandatory environment role of the proposed scheme

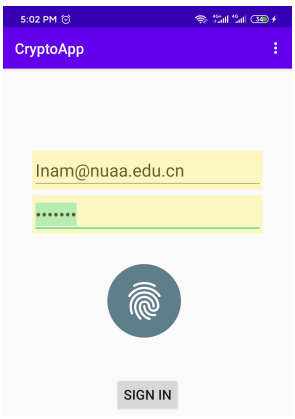
```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/
Finalized.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 1.53s
  visitedNodes: 108 nodes
  depth: 11 plies
```

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/testsuite/results/
Finalized.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 755 states
  Reachable  : 240 states
  Translation: 0.76 seconds
  Computation: 0.61 seconds
```

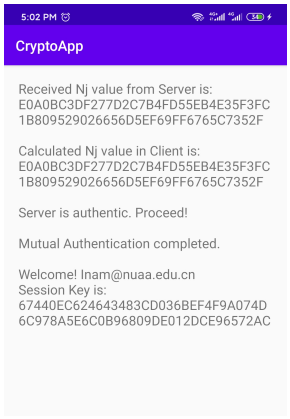
Fig. 7: Simulation Results of AVISPA tool

5 Implementation and Performance Analysis

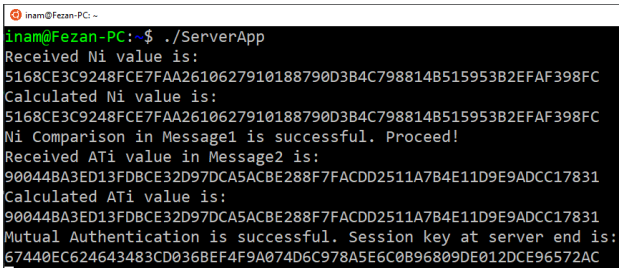
We have developed a sample client server-based application to implement the proposed protocol. The communication



(a) Login Screen of User



(b) MA and Session Key calculated at User End



(c) MA and Session Key calculated at Server End

Fig. 8: Output of a Successful Session key calculation on both ends in real life scenario

protocol is based on user datagram protocol (UDP). All the cryptographic functions in both client and server are implemented using the MIRACL library. A client application is developed for android using Android SDK, whereas, the server application is implemented in C++ on Ubuntu 16.0 LTS operating system using Boost. Asio framework. Login interface of the client application, as well as simulation of a successful mutual authentication & session key agreement, are illustrated in Fig. 8.

The performance analysis of the proposed protocol concerning storage memory required on the smart card, communication overhead, and computational complexity has been carried out in this section. Furthermore, we have compared the proposed protocol with existing SCPKC-based protocols concerning these metrics. For comparison, we select four latest MSAKA protocols, i.e., Wei *et al.* [48], Ying-Nayak [49], He *et al.* [2], and Hsieh-Leu [3].

The analysis and comparison have been done for security strength of 112 bits (security level of 2048 bits RSA algorithm, (2048, 256) bits DSA and 224 bits ECC). To comply with the said security strength we have used, an elliptic curve defined over a Galois field of 224 bits prime number q, and hash functions (sha256) in our experiment. For pairing-base settings, a Tate pairing defined over an elliptic curve with a 192-bit group order, prime modulus of 512 bits with "security multiplier" 4 is used.

5.1 Analysis and Comparison of Communication Overhead and Smart Card Storage Cost

Keeping in mind the cryptographic primitives used in our experiment, the length of q or length of all random numbers, or hash function output (truncated) is 224 bits. The length of elements in both groups of Tate pairing is 2048 bits. The length of each each elliptic curve group element is 448 bits. Using MIRACL library, elliptic curve points (in both pairing and non-pairing settings) can be efficiently compressed to almost half length. Let suppose the length of user identity is 64 bits, then the communication overhead and storage space required on the smart card for each aforementioned protocol is as follows:

Both parties (U_i and AS_j) in Ying-Nayak [49] protocol exchange two messages among them during mutual authentication phase. The communication overhead involve in these two messages, i.e., $U_i \rightarrow AS_j$: $\langle \sigma_{u_i}, DID_{u_i}, A^*_{u_i}, F_{u_i} \rangle$, $AS_j \rightarrow U_i$: $\langle \sigma_{s_j}, ID_{s_j}, B_{s_j}, F_{s_j} \rangle$ is $\langle 224 + 224 + 232 + 232 \rangle$, $\langle 224 + 224 + 232 + 232 \rangle$, respectively. So, the total communication cost of [49] is 1824 bits. Similarly the space required on smart card is $\{\phi_{u_i}, N_{u_i}, M_{u_i}, c_0\} = 224 + 232 + 224 + 224 = 904$ bits.

In He *et al.*'s protocol [2] user sends two messages $(\langle RU_i \rangle, \langle CU_i \rangle)$, whereas AS_j sends only one message $\langle y, a_{s_j} \rangle$. The total communication cost of these messages is $\langle 1032 \rangle + \langle 2048 + 224 \rangle + \langle 2496 \rangle = 5800$ bits. Accordingly, the smart card of the user contains $g_{u_i}, \psi_{u_i}, v_{u_i}$, and b_{u_i}. So, the storage cost of [2] is $2048 + 224 + 224 + 224 = 2720$ bits.

Protocol of Wei *et al.* [48] completes mutual authentication and session key agreement in three messages. Among these, user transmits two messages, i.e., $(\langle F_{ui}, k_{ui}, B_{ui}, d_{tui}, t \rangle$, and $\langle D_{ui} \rangle)$, whereas AS_j sends a single message $\langle D_{sj}, k_{sj} \rangle$. The cumulative communication overhead involved in messages is $\langle 224+2048+2048+224+224 \rangle + \langle 224 \rangle + \langle 224+2048 \rangle = 7264$ bits. Similarly, user's smart card contains $T_{ui}, G_{ui}, V_{ui}, h_1, \theta_{ui}, W_{ui}, B_{ui}, d_{tui}, t$ which require $6(224) + 2048 + 2(224) = 3840$ bits of storage space.

In Hsieh-Leu protocol [3] both participants, i.e., user and application server transmit messages $\langle xAuth_i, C_m, M_i, B_{ij}, R_i \rangle, \langle Auth_{ij} \rangle$ and $\langle Auth_{ji}, K_{ji}, R_j \rangle$ to other end respectively. The communication overhead involves in these messages is $\langle 1032 + 1032 + 1032 + 1032 + 1032 \rangle + \langle 224 \rangle + \langle 224 + 1032 + 1032 \rangle = 7672$ bits. The smart card of U_i stores $\{Auth_i, Reg_{ID_i}, Pub_{RS}, b_i, CID_i\}$, which requires total $1032 + 1032 + 1032 + 224 + 224 = 3544$ bits of memory.

In proposed protocol, three messages are exchanged between U_i and AS_j. The first message,i.e., $U_i \rightarrow AS_j$: contains $\langle M_i, D_{ij}, N_i \rangle$, second message $AS_j \rightarrow U_i$: contains $\langle W_{ji}, N_j, \Omega_j \rangle$ and final message $U_i \rightarrow AS_j$: contains $\langle AT_i, \beta_i \rangle$. So, the total communication overhead calculated as $520 + 232 + 224 + 232 + 224 + 232 + 224 + 232$ is 2120 bits. In the proposed protocol, the SC_i contains values of $\{R_i, A_i, C_i, V_i, ES_i\}$ which require storage space of $232 + 224 + 224 + 224 + 224 = 1128$ bits.

The comparison of communication overhead and space required on the smart card has been shown in Fig. 9.

5.2 Computational Complexity Analysis

This section presents the analysis and comparison in terms of the computational complexity of the proposed

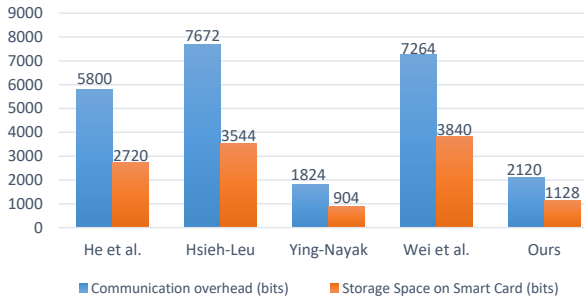


Fig. 9: Communication overhead and storage space required on SC_i

TABLE 3: Running time of Operations

Notation	Computational Time (milliseconds)	
	User End	Server End
T_b	36.13	10.84
T_{mp}	13.22	3.97
T_{px}	7.09	2.13
T_{ex}	4.89	0.89
T_{e+}	0.02	0.005
T_e	13.32	2.96
T_x	0.01	0.005
T_r	1.74	0.11
T_h	0.01	0.003

TABLE 4: Number of Operations executed on both ends

Protocol	No of Operations	
	User End	Server End
Hsieh-Leu [3]	$T_{mp} + 7T_{px} + 8T_h + T_r$	$T_{mp} + 2T_b + 5T_{px} + 3T_h + T_r$
He *et al.* [2]	$2T_e + 2T_{px} + 8T_h + T_x + T_r$	$4T_e + T_b + 5T_h + T_r$
Ying-Nayak [49]	$5T_{ex} + 8T_h + 2T_r$ $T_x + 2T_{e+}$	$5T_{ex} + 4T_h + T_r$ $T_x + 2T_{e+}$
Wei *et al.* [48]	$2T_e + 9T_h + T_r$	$4T_e + T_b + T_{mp} + 5T_h + T_r + T_x$
Ours	$5T_{ex} + 12T_h + T_r + T_x$	$5T_{ex} + T_x + 6T_h + T_r$

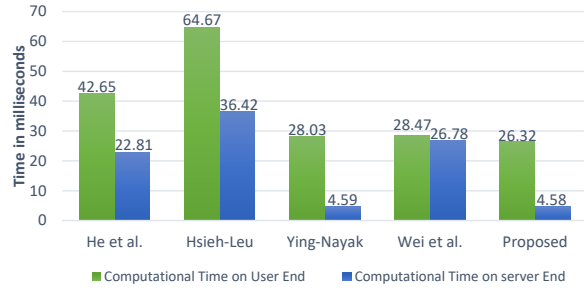


Fig. 10: Computational time elapsed on user and server ends

protocol with four above-mentioned benchmarked protocols. We only consider mutual authentication & key agreement phase for this analysis and comparison. Let $T_b, T_{mtp}, T_{px}, T_{ex}, T_{e+}, T_e, T_x, T_r, T_h$ denote execution time of bilinear pairing operation, map-to-point operation, pairing-based point multiplication operation, elliptic curve point multiplication operation, point addition operation, exponentiation operation, scalar multiplication, random number generation, and one-way hash function function, respectively. The running times of these operations are tabulated in Table 3. Whereas, Table 4 details the number of operations executed on both ends during mutual authentication and key agreement for each protocol.

The computational time for each operation, as illustrated in Table 3 has been calculated on a laptop computer (HP EliteBook 8460p with an i7-2620M 2.70GHz processor, 4G bytes memory, and the Ubuntu 16.04 LTS operating system) using the MIRACL library for server-side. Whereas, for the user side, running time has been calculated on a Xiaomi Redmi Note 7 smartphone having 6G bytes of memory, Snapdragon 660 Octa-core Max 2.20GHz processor, and Android 9 MIUI 11.0.4 development version.

Fig. 10 depicts the total time elapsed on both ends during mutual authentication & key agreement phase.

TABLE 5: The comparison of security and functionality features

Security & Functional Features	[3]	[2]	[49]	[48]	Ours
User Anonymity	✓	✓	✓	✓	✓
Un-traceability	✓	✓	χ	χ	✓
Mutual Authentication	✓	✓	χ	✓	✓
Provides TFS	χ	χ	χ	✓	✓
Provides PFS	✓	✓	✓	✓	✓
Resists UIA	✓	✓	χ	✓	✓
Resists SIA	χ	✓	✓	✓	✓
Resists offline IGA	χ	✓	χ	✓	✓
Resists offline PGA	χ	✓	χ	✓	✓
Resists Replay attack	✓	✓	✓	✓	✓
Resists SVA	✓	✓	✓	✓	✓
Resists PIA	✓	✓	✓	✓	✓
Secure in PTRS	χ	χ	χ	χ	✓
Single Registration	✓	✓	✓	✓	✓
Provides FAC	χ	χ	χ	χ	✓
Mutual Authentication with offline *RS*	✓	✓	✓	✓	✓

SIA: Server Impersonation Attack, UIA: User Impersonation Attack, IGA: Identity Guessing Attack, PGA: Password Guessing Attack, PFS: Perfect Forward Secrecy, TFS: Three-factor Security, SVA: Stolen Verifier Attack, PIA: Privileged Insider Attack, PTRS: Partially Trusted Registration server

5.3 Security and Functionality Features Comparison

This subsection compares the proposed protocol in terms of desired functional and security features with the related protocols. According to Table 5, all the protocols provide single registration and mutual authentication without engaging the *RS*. The protocols of Ying-Nayak [49] and Wei *et al.* do not confirm the un-traceability feature. The protocols and [2], [49], and [3] do not provide three-factor security. Furthermore, [3] is susceptible to offline IGA, offline PGA, and server spoofing attacks, whereas [49] does not withstand offline IGA, offline PGA, and user impersonation attack. Finally, all the compared protocols are insecure in partially trusted *RS* adversary model and do not feature flexible access control in multi-server architecture (MSA). In contrast, the proposed protocol ensures all the functional and security requirements of MSA.

5.4 Discussion on Comparison Results

The comparison results evidently show that the proposed protocol is computationally efficient and provides more enhanced security and functionality features among all the compared protocols. The protocol of Ying-Nayak performs better concerning communication overhead and smart card storage costs, but it is vulnerable to IGA, PGA, and UIA [37]. These security pitfalls cause it a less practical scheme for security-critical multi-server architectures.

6 Conclusion

In this paper, we identify three very vital deficiencies in the existing state-of-the-art MSAKA protocols. To alleviate this matter, we propose an MSAKA protocol with offline *RS* using SCPKC. The proposed scheme withstands both UIA and SIA in a partially trusted *RS* scenario, provides flexible access control, and is provably secure. Additionally, the simulation results of the AVISPA tool ensure that the proposed scheme resists active attacks in the Dolev-Yao intruder model. The provision of more satisfying security controls and better computational efficiency makes it very suitable for use in real-life applications with battery-limited mobile devices.

References

[1] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, July 2018.

[2] D. He, S. Zeadally, N. Kumar, and W. Wu, "Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2052–2064, Sep. 2016.

[3] W.-B. Hsieh and J.-S. Leu, "An anonymous mobile user authentication protocol using self-certified public keys based on multi-server architectures," *The Journal of Supercomputing*, vol. 70, no. 1, pp. 133–148, Oct 2014.

[4] Y.-P. Liao and C.-M. Hsiao, "A novel multi-server remote user authentication scheme using self-certified public keys for mobile clients," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 886 – 900, 2013, special Section: Recent Developments in High Performance Computing and Security.

[5] T. Wu, Z. Lee, M. S. Obaidat, S. Kumari, S. Kumar, and C. Chen, "An authenticated key exchange protocol for multi-server architecture in 5g networks," *IEEE Access*, vol. 8, pp. 28 096–28 108, 2020.

[6] H. Yao, X. Fu, C. Wang, C. Meng, B. Hai, and S. Zhu, "Cryptanalysis and improvement of a remote anonymous authentication protocol for mobile multi-server environments," in *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, June 2019, pp. 38–45.

[7] N. M. Lwamo, L. Zhu, C. Xu, K. Sharif, X. Liu, and C. Zhang, "Suaa: A secure user authentication scheme with anonymity for the single & multi-server environments," *Information Sciences*, vol. 477, pp. 369 – 385, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025518308570

[8] M.-C. Chuang and M. C. Chen, "An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics," *Expert Systems with Applications*, vol. 41, no. 4, Part 1, pp. 1411 – 1418, 2014.

[9] D. Mishra, A. K. Das, and S. Mukhopadhyay, "A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8129 – 8143, 2014.

[10] C. Wang, X. Zhang, and Z. Zheng, "Cryptanalysis and Improvement of a Biometric-Based Multi-Server Authentication and Key Agreement Scheme," *PLOS ONE*, vol. 11, no. 2, FEB 11 2016.

[11] A. G. Reddy, E. Yoon, A. K. Das, V. Odelu, and K. Yoo, "Design of mutually authenticated key agreement protocol resistant to impersonation attacks for multi-server environment," *IEEE Access*, vol. 5, pp. 3622–3639, 2017.

[12] D. Xu, J. Chen, and Q. Liu, "Provably secure anonymous three-factor authentication scheme for multi-server environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 2, pp. 611–627, Feb 2019.

[13] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[14] L.-H. Li, L.-C. Lin, and M.-S. Hwang, "A remote password authentication scheme for multiserver architecture using neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1498–1504, Nov 2001.

[15] I.-C. Lin, M.-S. Hwang, and L.-H. Li, "A new remote user authentication scheme for multi-server architecture," *Future Generation Computer Systems*, vol. 19, no. 1, pp. 13 – 22, 2003, selected papers of the 29th SPEEDUP workshop on distributed computing and high-speed networks, 22-23 March 2001, Bern, Switzerland.

[16] X. Cao and S. Zhong, "Breaking a remote user authentication scheme for multi-server architecture," *IEEE Communications Letters*, vol. 10, no. 8, pp. 580–581, Aug 2006.

[17] C.-C. Chang and J.-S. Lee, "An efficient and secure multi-server password authentication scheme using smart cards," in *2004 International Conference on Cyberworlds*, Nov 2004, pp. 417–422.

[18] J.-L. Tsai, "Efficient multi-server authentication scheme based on one-way hash function without verification table," *Computers & Security*, vol. 27, no. 3, pp. 115 – 121, 2008.

[19] E. Yoon and K. Yoo, "Robust multi-server authentication scheme," in *2009 Sixth IFIP International Conference on Network and Parallel Computing*, Oct 2009, pp. 197–203.

[20] 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*, vol. 34, no. 2, pp. 609 – 618, 2011, efficient and Robust Security and Services of Wireless Mesh Networks.

[21] X. Li, Y. Xiong, J. Ma, and W. Wang, "An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 763 – 769, 2012, simulation and Testbeds.

[22] J.-L. Tsai, N.-W. Lo, and T.-C. Wu, "A new password-based multi-server authentication scheme robust to password guessing attacks," *Wireless Personal Communications*, vol. 71, no. 3, pp. 1977–1988, Aug 2013.

[23] K.-H. Yeh and N. W. Lo, "A Novel Remote User Authentication Scheme For Multi-Server Environment Without Using Smart Cards," *International Journal of Innovative Computing Information and Control*, vol. 6, no. 8, pp. 3467–3478, AUG 2010.

[24] R. S. Pippal, C. D. Jaidhar, and S. Tapaswi, "Robust smart card authentication scheme for multi-server architecture," *Wireless Personal Communications*, vol. 72, no. 1, pp. 729–745, Sep 2013.

[25] J. Wei, W. Liu, and X. Hu, "Cryptanalysis and improvement of a robust smart card authentication scheme for multi-server architecture," *Wireless Personal Communications*, vol. 77, no. 3, pp. 2255–2269, Aug 2014.

[26] D. He and D. Wang, "Robust biometrics-based authentication scheme for multiserver environment," *IEEE Systems Journal*, vol. 9, no. 3, pp. 816–823, Sep. 2015.

[27] V. Odelu, A. K. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953–1966, Sep. 2015.

[28] Y. Tseng, S. Huang, T. Tsai, and J. Ke, "List-free id-based mutual authentication and key agreement protocol for multiserver architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 102–112, Jan 2016.

[29] S. Kumari, X. Li, F. Wu, A. K. Das, K.-K. R. Choo, and J. Shen, "Design of a provably secure biometrics-based multi-cloud-server authentication scheme," *Future Generation Computer Systems*, vol. 68, pp. 320 – 330, 2017.

[30] Q. Feng, D. He, S. Zeadally, and H. Wang, "Anonymous biometrics-based authentication scheme with key distribution for mobile multi-server environment," *Future Generation Computer Systems*, vol. 84, pp. 239 – 251, 2018.

[31] A. Irshad, S. A. Chaudhry, Q. Xie, X. Li, M. S. Farash, S. Kumari, and F. Wu, "An enhanced and provably secure chaotic map-based authenticated key agreement in multi-server architecture," *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 811–828, Feb 2018.

[32] S. Kumari, A. K. Das, X. Li, F. Wu, M. K. Khan, Q. Jiang, and S. K. Hafizul Islam, "A provably secure biometrics-based authenticated key agreement scheme for multi-server environments," *Multimedia*

Tools and Applications, vol. 77, no. 2, pp. 2359–2389, Jan 2018.

[33] S. Chatterjee, S. Roy, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "Secure biometric-based authentication scheme using chebyshev chaotic map for multi-server environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 824–839, Sep. 2018.

[34] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018.

[35] M. Girault, "Self-certified public keys," in *Advances in Cryptology — EUROCRYPT '91*, D. W. Davies, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 490–497.

[36] R. Amin and G. P. Biswas, "Design and analysis of bilinear pairing based mutual authentication and key agreement protocol usable in multi-server environment," *Wireless Personal Communications*, vol. 84, no. 1, pp. 439–462, Sep 2015. [Online]. Available: https://doi.org/10.1007/s11277-015-2616-7

[37] I. [ul haq], J. Wang, and Y. Zhu, "Secure two-factor lightweight authentication protocol using self-certified public key cryptography for multi-server 5g networks," *Journal of Network and Computer Applications*, vol. 161, p. 102660, 2020.

[38] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 523–540.

[39] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Comp.*, vol. 48, no. 177, pp. 203–209, 1987.

[40] M. Bellare and P. Rogaway, "Entity authentication and key distribution," *Adv. Cryptology-CRYPTO 1993*, pp. 232–249, 1994.

[41] J. Xu, W.-T. Zhu, and D.-G. Feng, "An improved smart card based password authentication scheme with provable security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723 – 728, 2009.

[42] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018.

[43] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, https://eprint.iacr.org/2004/332.

[44] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[45] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. P. C. Rodrigues, "Provably secure fine-grained data access control over multiple cloud servers in mobile cloud computing based healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 457–468, 2019.

[46] D. Dolev, "On the Security of Public Key Protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[47] 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, May 2002.

[48] W. Li, L. Xuelian, J. Gao, and H. Y. Wang, "Design of secure authenticated key management protocol for cloud computing environments," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[49] B. Ying and A. Nayak, "Lightweight remote user authentication protocol for multi-server 5g networks using self-certified public key cryptography," *Journal of Network and Computer Applications*, vol. 131, pp. 66 – 74, 2019.