sensors

MDPI

Article

SELAMAT: A New Secure and Lightweight Multi-Factor Authentication Scheme for Cross-Platform Industrial IoT Systems

Haqi Khalid [1,*], Shaiful Jahari Hashim [1,*], Sharifah Mumtazah Syed Ahmad [1], Fazirulhisyam Hashim [1] and Muhammad Akmal Chaudhary [2]

[1] Department of Computer and Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia, Serdang 43400, Malaysia; s_mumtazah@upm.edu.my (S.M.S.A.); fazirul@upm.edu.my (F.H.)
[2] Department of Electrical and Computer Engineering, College of Engineering and Information Technology, Ajman University, Ajman 346, United Arab Emirates; m.akmal@ajman.ac.ae
* Correspondence: haqikhalid1@gmail.com (H.K.); sjh@upm.edu.my (S.J.H.)

Abstract: The development of the industrial Internet of Things (IIoT) promotes the integration of the cross-platform systems in fog computing, which enable users to obtain access to multiple application located in different geographical locations. Fog users at the network's edge communicate with many fog servers in different fogs and newly joined servers that they had never contacted before. This communication complexity brings enormous security challenges and potential vulnerability to malicious threats. The attacker may replace the edge device with a fake one and authenticate it as a legitimate device. Therefore, to prevent unauthorized users from accessing fog servers, we propose a new secure and lightweight multi-factor authentication scheme for cross-platform IoT systems (SELAMAT). The proposed scheme extends the Kerberos workflow and utilizes the AES-ECC algorithm for efficient encryption keys management and secure communication between the edge nodes and fog node servers to establish secure mutual authentication. The scheme was tested for its security analysis using the formal security verification under the widely accepted AVISPA tool. We proved our scheme using Burrows Abdi Needham's logic (BAN logic) to prove secure mutual authentication. The results show that the SELAMAT scheme provides better security, functionality, communication, and computation cost than the existing schemes.

Keywords: multi-factor authentication; fog computing; industrial IoT; fog node; cross-platform

Citation: Khalid, H.; Hashim, S.J.; Syed Ahmad, S.M.; Hashim, F.; Chaudhary, M.A. SELAMAT: A New Secure and Lightweight Multi-Factor Authentication Scheme for Cross-Platform Industrial IoT Systems. *Sensors* **2021**, *21*, 1428. https://doi.org/10.3390/s21041428

Received: 20 November 2020
Accepted: 26 December 2020
Published: 18 February 2021

1. Introduction

The Internet of things (IoT) has gained tremendous popularity in the last decade with the advent of many powerful, low-cost devices such as sensors, RFIDs, etc., coupled with various communication media. Recently, the implementation of IoT in industries with Cyber-Physical System (CPS) as a part of the world of production and network connectivity is known as Industrial IoT (IIoT) [1]. The integration combines industrial devices equipped with communication, sensors, and Internet-connected actuator modules [1]. The devices are responsible for sensory data capture, environmental and industrial conditions tracking, and raw goods transport. However, it is estimated that the industrial IoT market will hit $123.89 billion by 2021 [2]. Industrial IoT (IIoT) can significantly enhance communication, efficiency, scalability, time savings, and cost savings for industrial sectors. Interoperability between devices and machines using different protocols with different architectures and the security of such protocols and data generated with these devices is the primary concern for IIoT [2–4]. As has been stated, industrial devices capture, store, transmit, or exchange large amounts of highly sensitive consumer information. The attacker can intercept and alter this transmitted data. These attacks threaten confidentiality in the information collected

and transmitted, leading to less trust in the entire system [5]. Therefore, it is essential to implement essential security features, such as confidentiality and integrity. Constrained devices, however, are the primary security considerations for IoT and IIoT applications. These devices are typically limited in computing power, storage capacity, and energy consumption. Therefore, it is a challenge to use some high computational cryptographic algorithms, which usually require more computation costs [1,6,7]. The National Institute of Standards and Techniques (NIST) reports that the fog computing architecture consists of three layers of edge devices, fog nodes, and a cloud layer. The edge device is the initial layer of fog architecture [8]. It is used to collect data and environmental monitoring by various industrially smart IoT devices (sensors and actuators). The fog nodes are context-conscious and support a shared data and communication system. The last layer consists of the cloud server, which stores data for potential use [9]. Services can be hosted in fog computing at end devices such as access points, as shown in Figure 1.

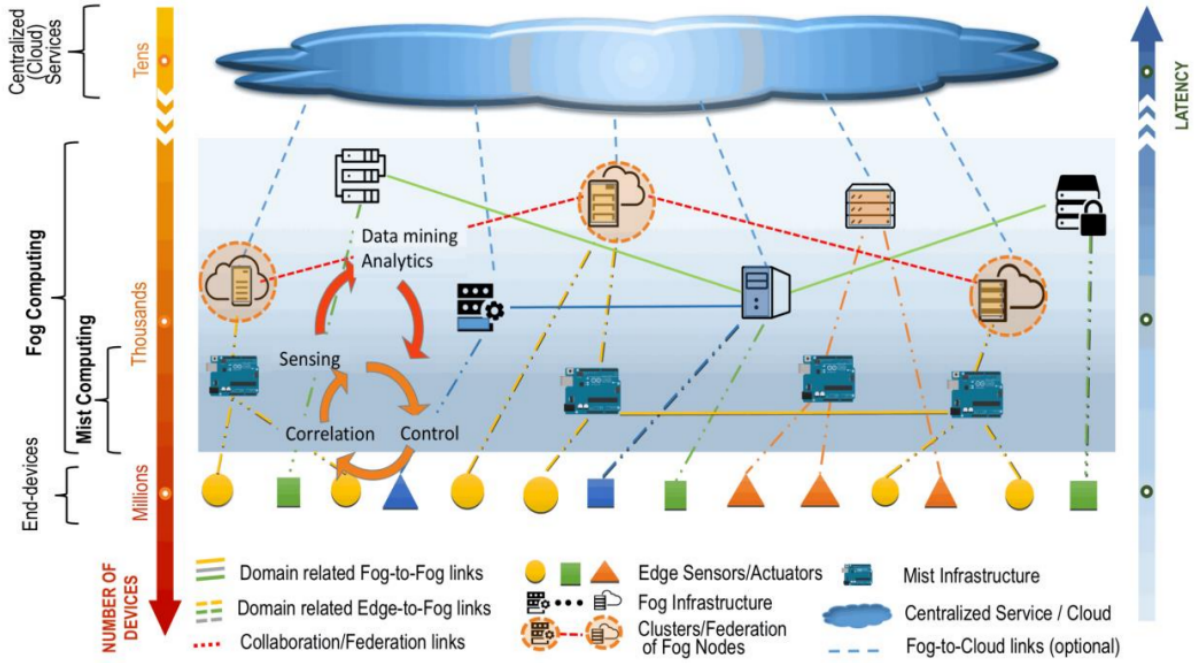


Figure 1. Fog computing supporting a cloud base for smart end-devices [9].

Fog providers may be different parties because of various deployment solutions; the existing infrastructure for fog can be used for wireless carriers (e.g., GSM) that control home or cellular base stations. Fog infrastructure can also be developed by cloud service providers needing their cloud services to the network edge. Because of the lack of authentication services, a rogue fog node/service will be a fog device or fog instance, which claims to be legitimate and attempts to control edge's fog nodes [10]. For example, a fog administrator may permit an insider attack not to instantiate a legitimate attack but to instantiate a rogue fog instance. For example, in an insider attack, a fog administrator can handle fog instances but can instead instantiate a rogue fog instance [9]. After communication, an adversary can manage incoming or outgoing user or cloud requests, capture or control user data stealthily, and initiate further attacks quickly. A fake fog node or server is a serious threat to the security and privacy of user data. Likewise, in 2012, Fire-base Google was launched to allow a front-end application to connect directly to a back-end database. However, they discovered that the Fire-base is vulnerable to the Stuxnet attack due to the absence of authentication and authorization [11]. Thus, due to an exposed interconnection between the edge devices and the fog node, authenticating users/devices and ensuring platform security becomes a huge challenge. The devices

often have a low computational capacity and low power consumption, particularly in the IoT system, which requires reducing authentication and encryption costs while ensuring information security [12]. Therefore, cross-platform authentication in cloud computing has not been considered before, and this gives rise to the problem of trustiness between cross-platforms in fog computing. Cross-platforms are places where multiple fog nodes authenticate mutually and exchange data [13–17]. Hence, developing lightweight cryptographic protocols to protect Industrial IoT devices against vulnerable attacks and satisfying device constraints are needed.

Thus, we propose a secure and lightweight multi-factor authentication scheme for cross-platform Industrial IoT systems (SELAMAT). The scheme intends to improve the security and establish secure communication between the edge devices and fog nodes. The SELAMAT scheme uses the AES-ECC algorithm to design an efficient key management system. AES (Symmetric Key Encryption Scheme) for the ECC Message Encryption (A-Symmetric Key Encryption Scheme) for the Secure Key Management mechanism is used in combination with data hiding to provide strong encryption and decryption requirements by using the advantages of both the cryptographic schemes. With our multi-factor authentication (MFA), three types of factors are used: Username/Password (something you know), smart card (something you have), and biometric (Fingerprint). Our MFA secures the user information from password guessing attacks, session attacks, and impersonation attacks. It provides layered security, making access in the fog node more difficult for unauthorized users to a target such as the physical location, device, network, application, or database.

2. Industrial IoT Security Requirements and Issues

The industrial Internet of things improves the efficiency, scalability, and security of the industrial environment applications. In such sensitive applications, introducing the resource-constrained IoT devices might bring essential security and privacy concerns.

2.1. Security Requirements

Several types of research [18,19] have underlined security requirements that must be considered in IIoT, particularly fog nodes and sensors. We further identify the most critical security and privacy requirements:

- Availability: The network infrastructure, devices (e.g., sensors), and fog nodes that handle the control and optimization queries should be continually available. Besides, unauthorized users should not deny allowing authorized users to handle queries.
- Confidentiality: The data and queries between edge nodes and fog nodes exchanged are confidential and must not be revealed by unauthorized third parties.
- Integrity: The type of data sharing between edge devices and fog nodes improves energy transmission decision making. For better decision-making, the integrity of these data is fundamental. We also need to deal with injection attacks that aim to inject false measures into the fog computing infrastructure that could interrupt decisions.
- Authenticity: Authentication of ubiquitous IoT connectivity is based on the nature of Industrial IoT environments in which communication between equipment and equipment Machine-to-Machine (M2M), between man and device, or between user and other would be possible. The authorizations property allows only authorized entities (any authenticated entity) to carry out certain network operations. It is necessary to design a secure authentication scheme to prevent unauthorized users from accessing the nodes.
- Non-repudiation: Any party in the system between the utilities' fog node servers and the edge nodes must not deny that they subsequently have not received such data or control commands.
- Privacy: Fog computing infrastructure information includes fine-grained data about users and even industrial machines. These data reveal information about the activities of customers in industries and companies. It is compulsory to encrypt and make these data untraceable.

2.2. Security Issues

Fog computing should withstand some security challenges in the Industrial IoT setting. We present the relevant ones as follows:

- Limitations of information system technology: The number of attacks is increasing because many industries are interconnected with cloud computing, which may affect the fog computing network's availability. The integrity of data, confidentiality, and privacy; spoofing servers; injection; DoS/DDoS attacks; impersonation attacks; and replay attacks, among others, are just some examples of attacks.
- Data sensitivity and privacy: The information shared between the network node and other fog nodes involves sensitive customer-specific information such as object tracking, power consumption, real-time data streaming, and performance monitoring. Neighbors should not leak this information while keeping it exploitable across fog nodes.
- Lack of Authentication: Fog computing nodes must be securely designed. They must verify information from a known source and ensure that it was not corrupted to avoid introducing some threats. The weak authentication mechanism for industrial sectors might allow attackers to impersonate a legitimate user. For example, an adversary may execute a password guessing attack, man-in-the-middle, or replay attack to access the targeted node. Therefore, a secure authentication scheme must be designed to prevent such attacks.
- Lack of data transmission encryption: The data exchange between edge devices and fog nodes is typical, not encrypted, and transmitted through a public channel. In this case, the attacker can still capture the network's data by using a simple network sniffer to monitor the connection between the user and the IIoT device. The attacker also can record the message and obtain the edge device information to perform a replay attack. The non-encrypted form allows the attacker to gather information about the targeted node, such as its database used by the node/device.
- Complexity: Researchers have proposed several authentication schemes for the IIoT environment, but those schemes are mostly based on cryptographic techniques requiring a high computational cost. Some of the cryptographic techniques use an extensive operation, such as the identity-based verification and multiplication operation. Since industrial IoT devices are limited resources with low power, a lightweight authentication scheme must be designed for IIoT suitability.

3. Related Works

Many researchers recently focused on providing secure authentication for industrial IoT systems. However, Chen et al. (2020) proposed a fog node Authentication Secure Authenticated Key Exchange Scheme [20]. Moreover, the proposed scheme uses only such basic operations because of limited resources of fog nodes and users, such as multiplication of elliptic curve cryptography point operations, bit-wise exclusive OR, and hash-only functions, instead of other complex algorithms. They argued that their system overcomes the risk of a temporary secret leakage attack. They proposed a three-step authentication system of: (1) the user registration phase, (2) fog server registration phase, and (3) login and authentication phase. However, due to a large amount of computation and communication, it involves heavy calculations due to public cryptography and signature algorithms or other time-consuming calculations (e.g., bilinear pairings). Munir et al. (2018) [21] proposed a biometric smart card authentication scheme based on pin and fingerprint identification in fog computing. In Phase 1, the user enters his information in two phases. Simultaneously, the pin is encrypted with DES and the fingerprint, which uses a robust mathematical algorithm that is not invertible and where both are stored on the fog server and smart card. In Phase 2, the user inserts the card and receives the verified information. The unauthorized individual has access to these data if some government or other source leaks the biometric data. However, privacy issues have been increased because an individual's unique identity is a biometric blueprint. Since the template cannot be decoded back to

biometric data, it can be used to track individuals if a database links the user to a specific biometric prototype, so the user's operation can be tracked unlawfully. Such threats must be tackled, and cancellable biometrics are a potential solution.

Rahman et al. (2019) [22] suggested that an enhanced mutual authentication security scheme be addressed based on an advanced encryption protocol and hashed message authentication code for fog computing. The authors tried to avoid the mid-attack in interactions between the fog user and registration authority. The attacker compromises the user's identity by sending the identity to the registering authority as fog users obtain from the registering authority the master secret key. However, this work still built session keys with a long-term master secret. A different scheme of mutual authentication is proposed for fog-based computer environments with constrained devices [23]. The proposed scheme, called Octopus, needs a long-lived secret key to authenticate with any fog server. However, it is about setting up and resuming the session. Session hijacking on the transport layer will result in a DoS attack. An attacking node may individually identify the victim node to continue between the two nodes. The nodes that are communicating may need to re-transmission messages by altering sequence numbers [22,24,25]. In [26], the authors provided secure key management and user authentication scheme called the SAKA-FC. It is defined as a secure communication protocol that supports fog and uses a one-way hash function and XOR that is bitwise supported by IoT-resource-driven devices. The scheme suffers from controlling the privileged insider attack and cannot provide a secure environment to compromise the attack. It should be noted that the authentication of users and binding agreements are not secured against future attacks. The scheme is not as lightweight as it requires more complex computing and communication. As a result, the scheme proposed for this environment will not correctly authenticate users in cloud-driven IIoT environments [27].

Similarly, Wang et al. [12] designed an anonymous lightweight authentication protocol for multi-level architecture fog-based applications. The protocol dynamically updates both sides of the session keys and ensures anonymous user authentications. The scheme proposed a key group protocol for management. The server can share with a specific attribute the key to desired communication nodes, and a private key between the two fog nodes can be created and updated without having to leak keys on the servers. However, the available group key management systems are not enough for mobile devices and require time-consuming computations. Moreover, He et al. [28] presented a new Mobile Healthcare Social Networks (MHSNs) handshake scheme. The scheme is based on hierarchical identity-based cryptography. The system consists of three tiers, while the highest level is a trusted authority using the Schnorr signature scheme to generate private keys to participating health centers. The second level is registered health centers with the Schnorr signature system, responsible for generating private keys. The third level is machine patients performing symptoms matching the cross-domain handshake. The system's goal was to allow two users registered in separate healthcare centers to conduct a symptom-matching cross-domain handshake. However, the scheme requires more computational effort because of its identity-based cryptographic technologies and lack of reliability due to resource-constrained devices (such as the elliptic curve) being more time consuming as they come with some operations [29,30].

In 2019, Jia et al. [31] proposed an AKE Scheme for an IoT-based fog computing health care system. Compared to conventional medical systems, data from consumer devices and sensors are transmitted via fog nodes in the fog layer instead of via the cloud. Fog nodes process, transfer, and store data to the end-user and return results. However, we found that this AKE device is vulnerable to a temporary secret attack [20]. M. Akram et al. [32] enhanced the security features by proposing an anonymous three-factor authentication scheme for multi-servers. The scheme designed was based on the elliptic curve cryptography, and the biometric information is verified by the user and the server separately. The registration center in their scheme is involved in the authentication phase and has separate responsibilities with the server. Likewise, H. Tan et al. [33] designed a pairing-free homo-

graphic authentication and key management scheme for VANET dynamic cross-platform authentication. The scheme used certificateless cryptography for mutual authentication and homomorphic key management. On each active validation, dynamic updating to anonymous vehicle identity is performed to achieve privacy preservation. It mainly focuses on solving the heavy bandwidth consumption and high latency. However, their scheme is vulnerable to specific passive and active attacks such as server spoofing attacks and DoS attacks.

Venčkauskas et al. (2019) [34] presented the secure Self-Authenticable Data Transfer Protocol to address the issue of secure communication between resource-constrained devices. The authors proposed a new lightweight, secure, and authenticable transfer protocol for communication between the edge nodes and the fog nodes. Instead of UDP (User Datagram Protocol) and DTLS (Datagram Transport Layer Security) protocols, the primary purpose of the proposed protocol is to use CoAP (Constrained Application Protocol) as a secure transport. SSATP only uses primitive symmetric cryptography, allowing small devices with memory and low-end processing capacities to be easily implementable. However, DTLS must be excluded from the critical quality to suit the smart cities' resource-intensive sensor nodes [12,35,36]. From the discussion above, we can see that most of the authentication schemes are still not satisfying the fog computing requirements. Thus, a lightweight and secure authentication scheme for the fog computing architecture is urgently needed. The comparison of the related authentication scheme is illustrated in Table 1.

Table 1. Comparison of the existing authentication scheme in fog computing.

Ref.	Issue	Structure	Implementation	Method	Performance	Limitation
[20]	Vulnerability to an ephemeral secret leakage attack.	Centralized	Prototype (IPhone6S (A9+M9highest 2GBCPU,2GB RAMandiOS10.11+PC).	ECC	Computation time = 85.7388 ms	Costly bilinear pairing operations, and Encryption key management problem.
[21]	Security issues in biometric mis-behavioral	Centralized	Prototype (ACOS and AET60 BioCARDKey kit)	AES	NA	Privacy issues may be used to track the individual and monitor the activities of the user.
[23]	Not preserving user privacy, therefore, exposed to MiTM attack	Centralized	PBC lab	PKC	Computational time = 387.762 ms	Public key revocation and Used a long-term master secret.
[26]	Fog user's smart devices are resource-limited and cannot perform extensive, conventional digital signatures	Centralized	JPBC library and Bouncy Castle	Hash functions and symmetric encryption	Computation cost = 8.745 ms.	Vulnerability denial-of-service attack, replay attack, and password guessing attacks.
[26]	Several protection and privacy issues, including data exposure, session key leakage, replay, MiTM, and impersonation attacks	Centralized	NS2-simulation	ECC	Throughput, End-to-End Delay, Packet Loss Rate, Computation Costs= 54.124 ms	Not suite user authentication in a cloud driven IoT environment.

Table 1. *Cont.*

Ref.	Issue	Structure	Implementation	Method	Performance	Limitation
[12]	The application scenario is single and cannot be expanded to authentication between FC devices	Centralized	Java socket/MYSQL 5	AES	Communication cost = 5760 bits, Computation cost = 76.812 ms	GKM not efficient for mobile devices and require time-consuming computations.
[28]	Not suitable for mobile device deployment due to the requirement for computationally expensive operations	Centralized	JPBC library	HIDS	Computation Overhead = 30.871 ms, Communication Overhead = 2240 bits	The scheme needs more Needs more computational effort due to the identity-based cryptographic technique, Key escrow problem.
[33]	Heavy bandwidth consumption and high latency	Centralized	pbc-0.5.12	Homomorphic encryption	Total Computation time = 9.593ms, Communication cost = 2584 bits	Vulnerable to server spoofing attack and DoS attack.
[34]	limited resources of the edge node devices are requiring less computational resources	Centralized	Prototype (Raspberry Pi computer, and Matlab)	AES, and ECC	Power consumption, Data loss	Vulnerable to replay attacks.
[31]	Unprotected fog nodes in remote cloud data center	Centralized	MIRACL library	ECC	Total Computation time = 20.535 ms, Communication cost = 2752 bits	Vulnerable to man-in-the-middle attack, replay attack
[32]	Vulnerability to some attacks in the authentication between multi-servers	Centralized	Py-crypto library	ECC	Total Computation time = 27.028 ms, Communication cost = 1920 bits	Vulnerable to server spoofing attack, and lack of perfect forward secrecy

4. Preliminaries

This section introduces important cryptographic principles and basic knowledge. The elliptic cryptosystem (ECC) and AES-ECC encryption and decryption are specified. Related notes, system models, security requirements, and network assumptions are then specified.

4.1. The Elliptic Curve Cryptography

Let $p > 3$ be the large prime and F_p be the finite field with order p, where $a, b \in F_p$ satisfy the following equation: $4a^3 + 27b^2(modp) \neq 0$. The elliptic curve $E_p(a, b)$ over the finite field F_p is defined with the following equation:

$$y^2 = x^3 + ax + b \mod p, \quad (1)$$

where $(x, y) \in F_p$. The additional operation on the curve is known as the doubling point of the two points. The addition of point is otherwise specified. All points on the curve and the point at infinity are a group of abelites $E(F_p)$ additives. Please note that $\infty = (-\infty)$ performs as the identity element.

Definition 1 (**Computational Diffie–Hellman Problem (CDHP)**). *Given $P, aP, bP \in G_1$ for $a, b \in Z_q^*$ where P is a generator of G_1, the advantage in computing $ab\,P$ to solve the CDHP problem for any probabilistic polynomial-time (PPT) algorithm A is negligible, which can be defined as:*

$$Adv_{A,G_1}^{CDHP} = Pr\left[A(P, aP, bP) \rightarrow : a, b \in Z_q^*\right] \quad (2)$$

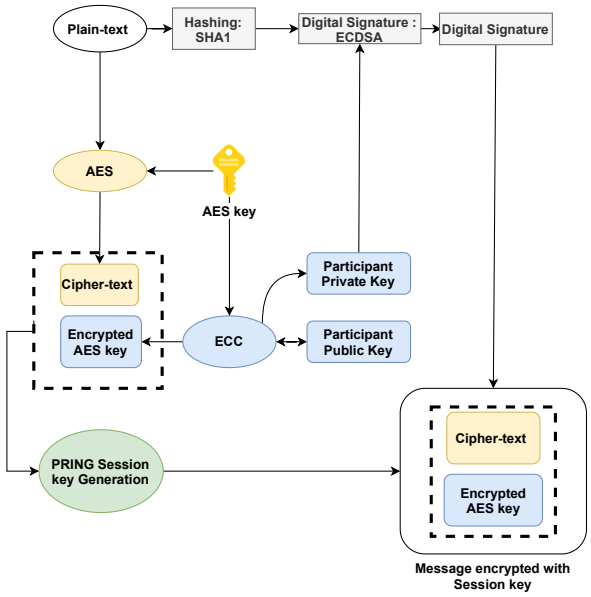
Definition 2 (Elliptic Curve Discrete Logarithm Problem (ECDLP)). *Given $P, Q \in G_1$, where $Q = aP$. The advantage in finding the integer $a \in Z_q^*$ to solve the ECDLP problem for any probabilistic polynomial-time (PPT) algorithm A is negligible, which can be defined as:*

$$Adv_{A,G_1}^{CDHP} Pr\left[(P, aP) \rightarrow a : a \in Z_q^*\right]. \quad (3)$$

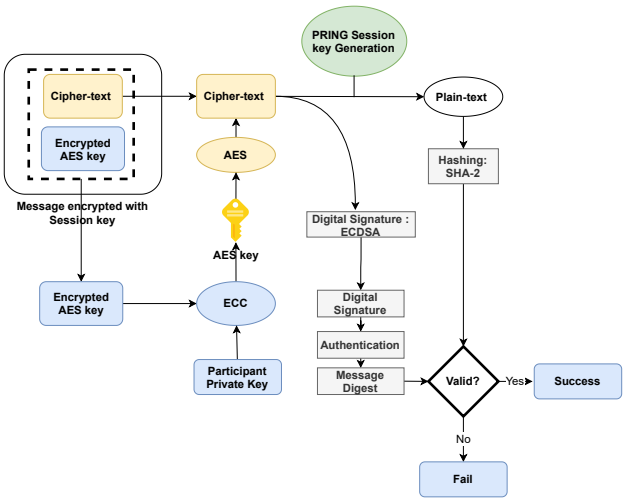
4.2. AES-ECC Encryption/Decryption

In this section, we explain the process of the AES-ECC algorithm for efficient key generation and secure user data transmission. The ECC is utilized to encrypt and transfer the Private Keys as AES Private Keys while AES encrypts the plain text (communication data). The process is applied when an entity needs to encrypt/decrypt the message [37,38]. The encryption/decryption processes are shown in Figure 2, and the steps are explained as follow:

1. Data are the users' information, i.e., their identities, passwords, and biometrics.
2. SHA-2 is used to produce a data summary.
3. ECC-related sender private key and ECDSA module are used to produce a digital signature.
4. According to the AES encryption module (AES private key), digital signature encryption and data to be submitted are encrypted. Then, the ciphertext data and ciphertext signature are encrypted.
5. The AES private key is encrypted by the ECC encryption module, and then the key-ciphertext is generated.
6. All the ciphertexts are packed and sent via the cross-d system network to the receiver.
7. Therefore, the sender uploads the ciphertext to the authentication server.
8. When the receiver receives the ciphertext, the receiver uses its private key to decrypt the AES key, and then decrypts the AES key data-ciphertext and signature-ciphertext. He/she uses the public key to check the signature, digest the message, and then get the plaintext by using the SHA-2 algorithm. If the message digest and plaintext are the same, the data are valid and available; otherwise, they are invalid.



(**a**) Encryption process.



(**b**) Decryption process.

Figure 2. AES-ECC Encryption/Decryption.

5. SELAMAT Scheme

This section proposes a multi-factor authentication scheme for industrial IoT (IIoT) to establish secure communication between the edge devices and the fog node. The system backend architecture is shown in Figure 3; our scheme is based on a smart card, username/password, and a biometric (fingerprint). The scheme adopts the combination of the AES-ECC algorithm for secure key management. It provides a secure mutual authentication among the edge and the fog server; the mutual authentication diagram is illustrated in Figure 4. The proposed scheme comprises five phases, i.e., the setup phase, user registration phase, fog node registration phase, login phase, and the authentication phase. We explain the scheme phases as follows.

5.1. Setup Phase

The Cloud Provider Server (CPS) selects a κ-bit prime number p and an elliptic curve E/F_p. The generated elliptic curve group G has a generator P. Then, CPS selects a random integer $S \in Z_q^*$ as the system master key and calculates the system public key accordingly $PK = S.p$. After that, CPS choose asymmetric encryption/decryption pairs $E\{.\}/D\{.\}$, and cryptographic collision-resistant hash function $H(.)$. Note that the AES shared private key is used to encrypt and decrypt the user information while being transmitted among the entities. CPS later publishes the system public parameters $\{G, p, q, PK, H(.)\}$ and keep the system master key S secretly. Moreover, when the user enters his/her information, it is then encrypted using the AES symmetric algorithm. The AES key is a shared private key between the sender and the receiver, which is used to encrypt the user information, and then it is encrypted with the ECC public key shared earlier with the user. The cipher-text is then transmitted to the AS in the CPS for verification. The used notions in the proposed scheme are shown in Table 2.

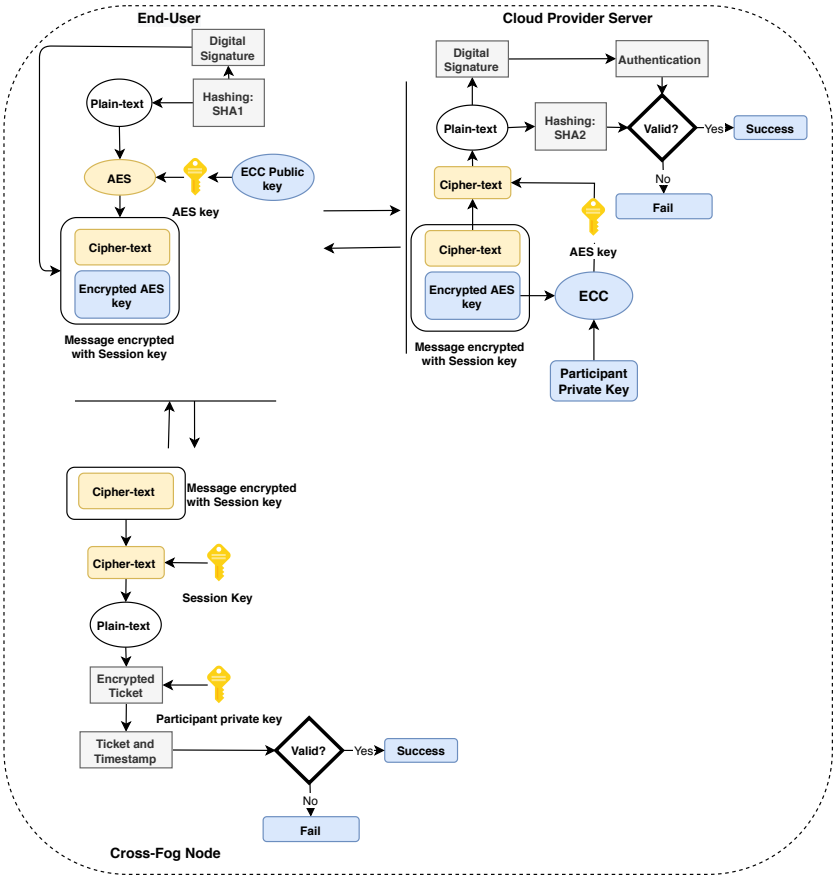


Figure 3. The system architecture for SELAMAT.

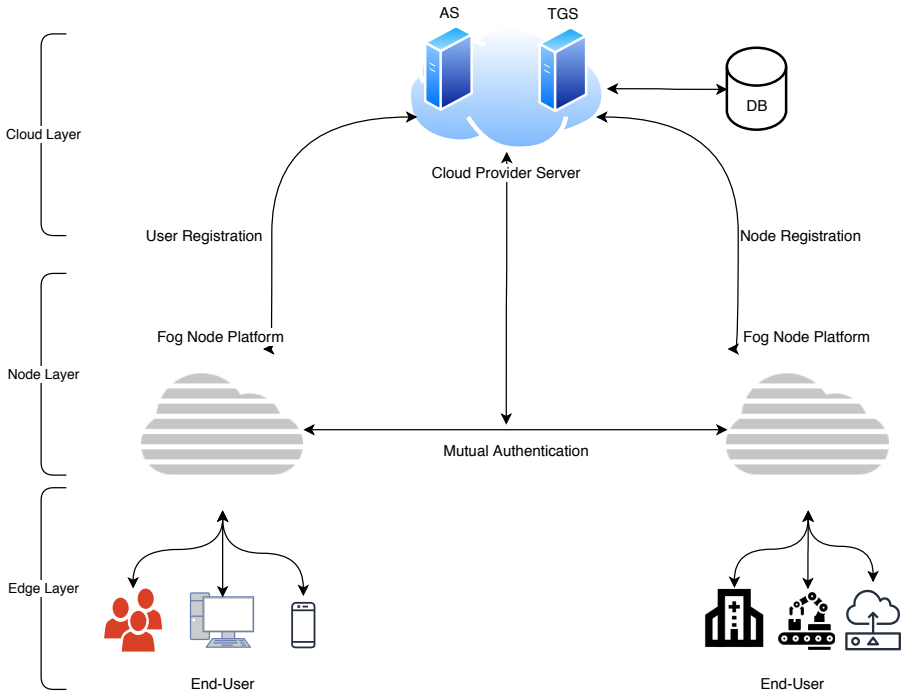


Figure 4. The mutual authentication scheme in fog computing architecture.

Table 2. Notation and abbreviations.

Notations	Description
SC	Smart Card.
U_i	User.
u_{id}	Identity of the User U_i
u_{pw}	The password of the user U_i.
u_{bi}	Biometrics imprint of the user U_i.
p, q	Two prime numbers.
r_i	Random number.
$H(.)$	One-way hash function.
$E\{.\}/D\{.\}$	A pair of symmetric encryption/decryption.
Z_q^*	The non-zero integers modulus p.
PK	The public key of the server.
S	The secret key of the server.
$Auth_u^{'}$	The authenticator of the user U_i.
TS	Timestamp.
$ks_{u \to tgs}$	A key session between User and TGS.
tgs_{sk}	Secret key of the TGS.
tgs_{id}	Ticket granting server identity.
fn_{id}	Identity of the Fog node.
tgs_{tkt}	Ticket granting server ticket.
$ks_{u \to fn_S}$	Key session between User and FN.
fn_{tkt}	Fog Node ticket.
$\|\|$	Concatenation operation.
$\oplus$	XOR operation.

5.2. User Registration Phase

In this phase, the user must register himself/herself at the cloud provider server to access the data that he wants to use. After that, the user Ui is issued a smart card; the authentication server (AS) in the CPS stores the user has protected biometric information Ui in its database. The registration is transmitted securely to obtain the smart card SC.

The user is not required to send his/her information in plaintext because the proposed scheme utilizes asymmetric encryption/decryption pair in which the information will be encrypted. In addition, that transmitted information is sensitive and will be handed to the server in a masked manner using the hash function for other security matters to prevent an insider attack. The user will firstly select his/her unique identity and password and input his/her biometric information. Upon receiving the request, the authentication server will decrypt the message and verify the given information whether he/she already exists in the server database or not. If U_i already exists at the server, it will inform the user that the identity exists and choose another. Otherwise, the AS will start user registration by performing the following steps, as shown in Figure 5:

- The user inserts his/her smart card and then selects a unique user identity u_{id} and a user password u_{pw} and inputs his biometric information u_{bi}. Then, the user randomly chooses an integer $u_{sk} \in Z_q^*$ as user private key and calculates his/her public key $u_{pk} = u_{sk}.p$.
- In addition, user Ui generates a random number r1 and a process to compute $bio_i = H(U_{Bi} \oplus r_1)$ and calculates $m_i = H(u_{id} \oplus u_{pw} \oplus bio_i \oplus r_1)$. The user Ui consequently encrypts the message with the AES private key $\epsilon_k\{u_{id} \parallel u_{pw} \parallel bio_i \parallel m_i\}$, and it encrypts the AES private key with the ECC shared public key Msg.1. $E_{pk}\{k_p\{u_{id} \parallel u_{pw} \parallel bio_i \parallel m_i\}\}$ and sends it to the AS in the CPS.
- Upon receiving the message, AS will use his private key to decrypt the AES key, $d_s\{\epsilon_k\{u_{id} \oplus u_{pw} \oplus bio_i \oplus m_i\}\}$, and then it uses the AES key to obtain the user information $\mu_k\{u_{id} \parallel u_{pw} \parallel bio_i \parallel m_i\}$. The AS verifies the received information with the one in the database. If the user exists, the AS will notify the U_i to choose another identity; otherwise, the AS computes $a_i = H(u_{id} \parallel s)$, $F_i = H(bio_i \parallel s)$ and calculates $R_i = a_i \oplus H(bio_i \oplus m_i)$, $n_i = f_i \oplus H(u_{id} \oplus m_i)$ and $x_i = H(a_i \parallel f_i \parallel m_i)$.
- Next, the authentication server AS will embed the calculated parameters $\{x_i, n_i, r_i, H(.), q, p\}$ onto the smart card SC and will send it to the U_i. Those parameters will also be stored in the database and recorded as an enrolled user. Now, AS sends the parameter to the U_i via a secure channel.
- The user U_i receives the embedded smart card SC and writes the parameters Msg.2: $\{x_i, n_i, r_i, H(.), q, p\}$ into the smart card and stores r_1 in the memory.

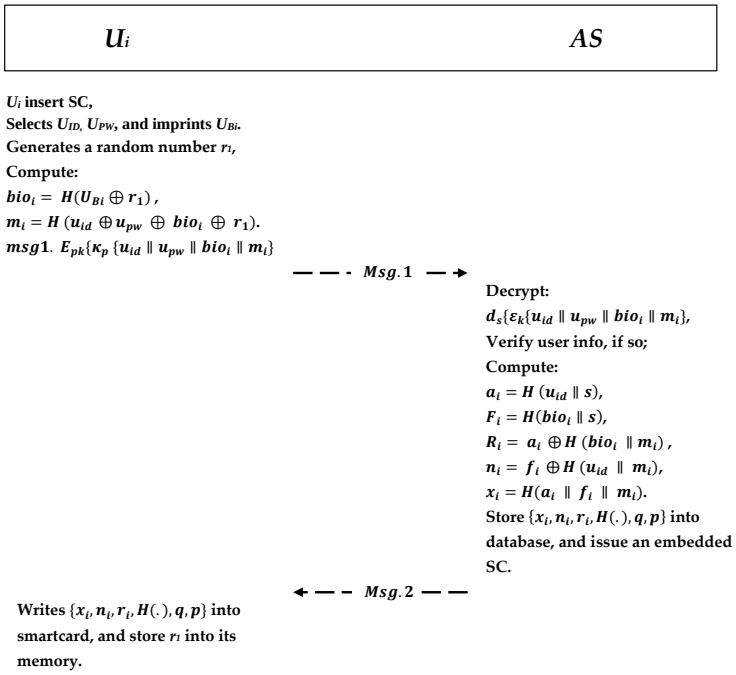


Figure 5. User registration phase.

5.3. Fog Node Registration Phase

This phase requires the fog node Fn_i to register itself into the CPS. As shown in Figure 6, a fog node Fn_i performs the following steps:

- The Fn_i firstly selects an identity fn_{id} and sends fn_{id} it to the AS in the CPS via a secure channel.
- AS receives it and checks whether the identity exists or not by comparing $fn_{id} = fn_{id}$ stored in the database; after verification, AS generates its own random number r_f; computes $\alpha_{fn} = h(fn_{id} \parallel s \parallel r_f)$, $\gamma_{fn} = h(fn_{id} \parallel s)$, $\rho_{fn} = h(fn_{id} \parallel r_f)$, and $C_{fn} = \gamma_{fn} \oplus \rho_{fn}$; stores $\{fn_{id}, r_f\}$ into a database; and sends $\{\alpha_{fn}, C_{fn}\}$ back to Fn_i securely.
- Fn_i receives α_{fn}, C_{fn} and stores it in its database.

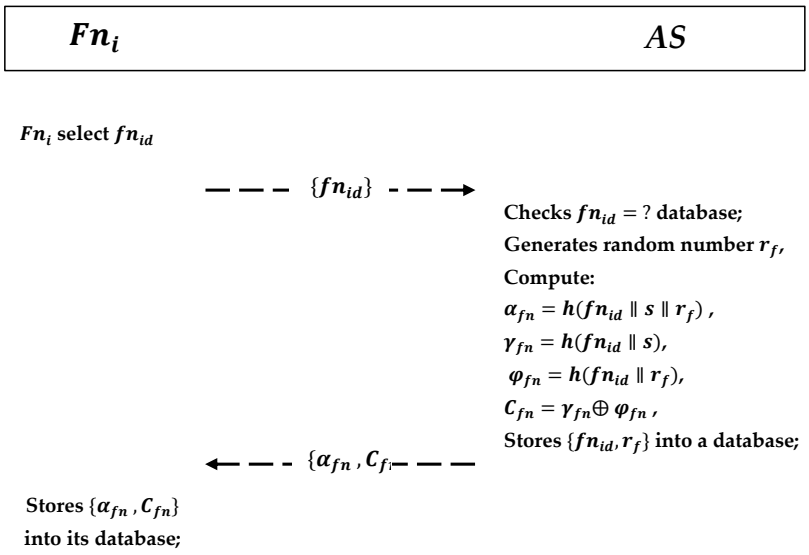


Figure 6. Fog node registration phase.

5.4. Login Phase

When user U_i wants to access the data stored on the cross-platform fog server, she/he inserts her/his smart card into the terminal and performs the following steps to log into the system. Figure 7 shows the process of the login and authentication phases.

- The user U_i inserts her/his identity u_{id} and password u_{pw}; inputs his/her biometric information; and extracts the random number and the information stored in the smart card SC. U_i computes $bio_i = h(u_B^{'} \oplus r_{2i})$, $m_i^{'} = h(u_{id}^{'} \oplus u_{pw}^{'} \oplus bio_i^{'} \oplus r_i)$.
- Then, based on the stored parameters, it will calculate $a_i^{'} = r_i^{'} \oplus h(bio_i^{'} \oplus m_i^{'})$, $f_i^{'} = n_i^{'} \oplus h(u_{id}^{'} \parallel m_i^{'})$, and $x_i^{'} = h(a_i^{'} \parallel f_i^{'} \parallel m_i^{'})$.
- Next, the smart cart computes an authentication message encrypted with the AES private key and the ECC public key $Auth_u^{'} = E_{PK}\{\xi_k\{u_{id}^{'} \parallel x_i^{'} \parallel bio_i^{'} \parallel tgs_{id} \parallel TS_1\}\}$, where the TS is the current user timestamp. Then, U sends the Msg.1 to the AS.
- Upon receiving the authentication request message Msg.1, the server decrypts the message utilizing its private key to decrypt the message $Auth_u^{'} = d_{PK}\{\epsilon_k\{u_{id}^{'} \parallel x_i^{'} \parallel bio_i^{'} \parallel tgs_{id} \parallel TS_1\}\}$, and then the encrypted message with the AES key decrypts the parameters with the same key to obtain the information $\mu_k\{u_{id}^{'} \parallel x_i^{'} \parallel bio_i^{'} \parallel tgs_{id} \parallel TS_1\}$.
- After that, AS checks the timestamp to see if it is similar to the server timestamp, extracts the $u_{id}^{'}$, and verifies $x_i^{'} = x_i$ whether it is valid or not; if not, the session is terminated. Otherwise, AS proceeds generating a random integer number r_2 and

computes the key sessions known to protect the communication between the user and the ticket-granting service. It is only known to the U_i and AS.

- Next, AS computes the key session $ks_{u \to tgs} = H(u_{id}^{'} \parallel a_i^{'} \parallel r_2)$ and generated another random number $tgs_{sk} \in Z_q^*$ as the TGS secret key is known to AS and TGS only. AS then forwardd the TGS secret key to the TGS along with $\langle X_i^{'}, Bio_i^{'} \rangle$ to the TGS.
- Then, AS prepares the message Msg.2: $u_{id}^{'} \parallel tgs_{id} \parallel \epsilon_k\{ks_{u \to tgs} \parallel TS_2 \parallel tgs_{tkt}\}$, where $tgs_{tkt} = E_{tgs_{sk}}\{u_{id}^{'} \parallel ks_{u \to tgs} \parallel TS_2 \parallel x_i \parallel bio_i^{'}\}$ is the Ticket. The tgs_{tkt} cannot be decrypted but the TGS only. It then sends the messages Msg.2 to the user to enable her/him to authenticate the TGS.

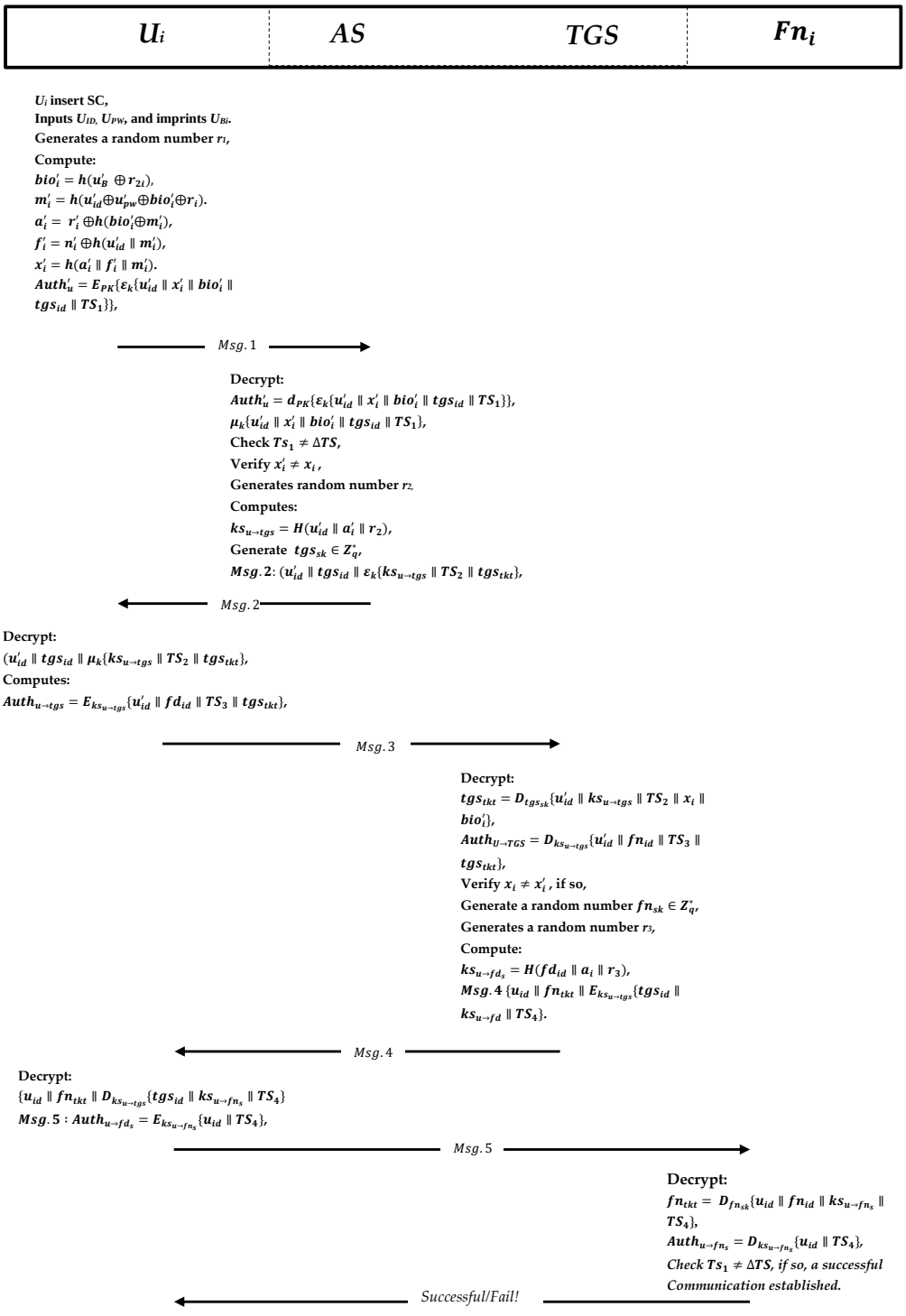


Figure 7. Login and authentication phase.

5.5. Authentication Phase

In this phase, the user will decrypt the message A to obtain the critical session and the parameters after successfully receiving the messages from the authentication server. The message B will be forwarded to the TGS. The steps of the authentication phase are given below:

- The user decrypts message Msg.2 using the key μ_k to get the critical session $ks_{u \to tgs}$ and other information $(u_{id}^{'} \parallel tgs_{id} \parallel \mu_k\{ks_{(u \to tgs)} \parallel TS_2 \parallel tgs_{tkt}\}$, and contains the Ticket granting service ticket tgs_{tkt} and this message is encrypted by the tgs_{sk} and the user cannot modify the Ticket in private. Therefore, U_i will forward it to the TGS as message Msg.3.
- It then computes an authentication message $Auth_{u \to tgs} = E_{ks_{u \to tgs}}\{u_{id}^{'} \parallel fn_{id} \parallel TS_3 \parallel tgs_{tkt}\}$, which contains the user identity, fog node identity, the current user timestamp, and the TGS ticket. The message is encrypted with the key session $ks_{u \to tgs}$ that is shared to communicate U_i and TGS. The user sends a request to the TGS to get permission to visit the fog node server.
- Upon receiving the request from Ui, TGS decrypts the Ticket $tgs_{tkt} = D_{tgs_{sk}}\{u_{id}^{'} \parallel ks_{u \to tgs} \parallel TS_2 \parallel x_i \parallel bio_i^{'}\}$ to obtain a key session and decrypts the authenticator message as well by using the shared key sessions $Auth_{u \to tgs} = D_{ks_{u \to tgs}}\{u_{id}^{'} \parallel fn_{id} \parallel TS_3 \parallel tgs_{tkt}\}$.
- Next, it verifies the $x_i \neq x_i^{'}$ that was received earlier from AS; if it is not equal, the session will be terminated; if yes, then the TGS will generate a random number $fn_{sk} \in Z_q^*$ as the fog node secret key that will be known to the TGS and the fog node server, and it will then be sent to the fog node along with $< T_4 >$.
- Then, it generates a random number r3 to compute the key session $ks_{u \to fn_s} = H(fn_{id} \parallel a_i \parallel r_3)$ and composes the message Msg.4 $\{u_{id} \parallel fn_{tkt} \parallel E_{ks_u \to tgs}\{tgs_{id} \parallel ks_{u \to fn} \parallel TS_4\}\}$, where the fog node ticket is $fn_{tkt} = E_{fn_{sk}}\{u_{id} \parallel fn_{id} \parallel ks_{u \to fn_s} \parallel TS_4\}$, which contains user the identity, fog node identity, shared key session, and the current timestamp. It then sends the message Msg.4 to the user to enable him/her to authenticate to the fog node.
- The user U_i receives information from the TGS; it will firstly decrypt the message $\{u_{id} \parallel fn_{tkt} \parallel D_{ks_{u \to tgs}}\{tgs_{id} \parallel ks_{u \to fn_s} \parallel TS_4\}\}$ to get the shared session key, and the user cannot decrypt the fog node Ticket.
- Next, the user generates an authenticator message Msg.5: $Auth_{u \to fn_s} = E_{ks_{u \to fn_s}}\{u_{id} \parallel TS_4\}$, and composes the Ticket $fn_{tkt} = E_{fn_{sk}}\{u_{id} \parallel fn_{id} \parallel ks_{u \to fn_s} \parallel TS_4\}$. Then, it will send the messages to the fog node for mutual authentication.
- The fog node server receives the message Msg.5, and it will decrypt the message from the secret key that it shared earlier from the TGS $fn_{tkt} = D_{fn_{sk}}\{u_{id} \parallel fn_{id} \parallel ks_{u \to fn_s} \parallel TS_4\}$ to get a key session.
- Then, the server decrypts the authenticator message using a shared key session $Auth_{u \to fn_s} = D_{ks_{u \to fn_s}}\{u_{id} \parallel TS_4\}$, and checks the authenticator timestamp with a shared timestamp and if it is not equal, the server terminates the session; if yes, then the client can trust the server and can start issuing service requests to the server and send a successful message. The fog node server now provides the requested services to the user.

6. Security Analysis

The security analysis is carried out formally and informally in this section. The formal security analysis of the proposed scheme was conducted with the BAN logic (Burrows–Abadi–Needham), a formal model that aims to see how information exchange can be secured from eavesdropping. Informal security analysis ensures that, e.g., the proposed scheme prevents different kinds of known attacks. The following paragraphs provide details of the BAN logic.

6.1. Mutual Authentication Proof Using BAN Logic

We conducted a BAN logic analysis to verify the proposed scheme with secure mutual authentication. Table 3 specifies the BAN logic notations and postulates and describes the goals, assumptions, idealized version formulas, and confirms secure mutual authentication in the proposed scheme before performing a BAN logic analysis.

Table 3. Notation and abbreviations.

Construct	Explanation
U_i	User.
AS	Authentication Server.
TGS	Ticket Granting Server.
FN	Fog Node.
K_X	X's shared session key with CA.
$K_{X,Y}$	X's shared session key with Y.
$.K$	A message encrypted by Key K.
$Ticket_{X,Y}$	Ticket used to visit X and Y.
TS_1, TS_2, TS_3	Random numbers Timestamp.
$X \xrightarrow{K} Y$	K is the shared session key between X and Y.

6.1.1. Message Exchanges

This section illustrates an optimal way of the message exchanges for our proposed scheme.

1. **The authentication service exchanges:** Firstly, the messages in the scheme are exchanged between the user and the authentication server (AS), also called the ticket exchange. The user applies for the ticket TGT to communicate with TGS and the session key from the Cloud Provider Server (CPS). The user needs to input his/her user identity and password to log into the network. The user sends a request message U_AS_REQ, and AS responds U_AS_REP.
   (a) Ui $\rightarrow$ AS: U_AS_REQ $\{u'_{id}, x'_i, bio'_i, tgs_{id}, ts_1\}$.
   (b) AS $\rightarrow$ Ui: U_AS_REP $(u'_{id}, tgs_{id}, E_{u_{pk}}\{ks_{u \rightarrow tgs}, ts_2, tgs_{tkt}\}$.
2. **The authorization service exchanges:** It is the process of the message's exchanges between the user and the TGS to get a ticket to communicate with the fog server. The user sends a request encrypted with a shared session key and for decryption as well. The TGS exchange consists of two messages: U_TGS_REQ and U_TGS_REP.
   (a) $Ui \rightarrow AS$: U_TGS_REQ $E_{ks_{u ß tgs}}\{u'_{id}, fn_{id}, ts_3, tgs_{tkt}\}$.
   (b) $AS \rightarrow Ui$: $U\_TGS\_REP\{u_{id}, fn_{tkt}, E_{ks_{u \rightarrow tgs}}\{tgs_{id}, ks_{u \rightarrow fn_s}, ts_4, \}\}$.
3. **The user/fog server exchange:** In this process, the user gets the Ticket and shared key session known to the user edge and fog node server fn. The user can now communicate to the fn using the received information and will consist of two messages: U_FN_REQ and U_FN_REP. U_FN_REP is only used when there is a need for two-way authentication, and the server wants to prove its identity to the client.
   (a) Ui $\rightarrow$ FN: U_FN_REQ $(E_{fn_{sk}}\{u_{id}, fn_{id}, ks_{u \rightarrow fn_s}, ts_4\})$.
   (b) FN $\rightarrow$ Ui : U_FN_REP (success/fail).

6.1.2. Goals and Assumptions

Initial assumptions: Hence, the authentication goals are given as follows:

- *Goal 1:* Ui **believes** $Ui \xrightarrow{pk} AS$.
- *Goal 2:* FN **believes** $FN \xrightarrow{ks_{fn,tgs}} TGS$.
- *Goal 3:* Ui **believes** AS **Controls** $Ticket_{ui,fn}$.

- *Goal 4:* Ui **believes** TGS **controls** $ks_{ui,fn}$.
- *Goal 5:* Ui **believes** TGS **controls** $ks_{ui,fn}$.
- *Goal 6:* FN **believes** TGS **controls** $ks_{ui,fn}$.
- *Goal 7:* Ui **believes fresh** (TS_1).
- *Goal 8:* FN **believes fresh** (TS_2).
- *Goal 9:* Ui **believes fresh** (TS_3).

6.1.3. BAN Logic Proof

Based on the logical assumptions, the authentication goals of the proposed scheme can be illustrated according to the BAN logic as follows:

1. According to the message meaning rule (Rule 1): Ui believes the PK is a shared public key between Ui and AS. In addition, the Ui sees that $Tiket_{U,TGS}$ is encrypted with ks_{u-tgs}, and then Ui believes that AS once said $Ticket_{u,tgs}$.

$$\frac{Ui \textbf{ believes } FN \xrightarrow{pk} TGS, Ui \textbf{ sees } \{Ticket_{u,tgs}\}_{ks_{ui \to tgs}}}{Ui \textbf{ believes } AS \textbf{ said } Ticket_{ui,tgs}}$$

2. By timestamp-verification rule (Rule 2): The Ui believes that [TS1] is fresh if Ui believes that AS once said $Ticket_{ui,tgs}$. Meanwhile, Ui believes that AS believes $Ticket_{u,tgs}$.

$$\frac{Ui \textbf{ believes fresh (TS1)}, \ Ui \textbf{ believes } AS \textbf{ said } Ticket_{ui,tgs}}{Ui \textbf{ believes } AS \textbf{ believes } Ticket_{ui,tgs}}$$

3. By Rule 3: If Ui believes that AS controls $Ticket_{ui,tgs}$, then Ui believes that AS believes $Ticket_{ui,tgs}$ and the Ui believes the $Ticket_{ui,tgs}$.

$$\frac{Ui \textbf{ believes } AS \textbf{ controls } Ticket_{ui,tgs}, \ Ui \textbf{ believes } AS \textbf{ believes } Ticket_{ui,tgs}}{Ui \textbf{ believes } Ticket_{ui,tgs}}$$

4. By Rule 4^u,

$$Ui \textbf{ believes } ks_{ui,tgs}$$

5. Again, by message meaning rule: Ui believes that $ks_{ui,tgs}$ is shared session key with TGS, and Ui sees the $Ticket_{ui,fn}$ is encrypted with $KS_{ui,tgs}$, and then Ui believes that TGS once said $Ticket_{ui,fn}$.

$$\frac{Ui \textbf{ believes } Ui \xrightarrow{ks_{ui,tgs}} TGS, Ui \textbf{ sees } \{Ticket_{ui,fn}\}_{ks_{ui \to tgs}}}{Ui \textbf{ believes } TGS \textbf{ believes } Ticket_{ui,fn}}$$

6. By Rule 2: Ui believes that the $(Ticket_{ui,fn})$ is fresh, and Ui believes that TGS once said $Ticket_{ui,fn}$, while Ui believes that TGS believes $Ticket_{ui,fn}$.

$$\frac{Ui \textbf{ believes fresh } (Ticket_{ui,fn}), Ui \textbf{ believes } TGS \textbf{ said } Ticket_{ui,fn}}{Ui \textbf{ believes } TGS \textbf{ believes } Ticket_{ui,fn}}$$

7. Again, by Rule 3, if Ui believes that TGS controls $Ticket_{ui,fn}$, then Ui believes that TGS believes $Ticket_{ui,fn}$ and Ui believes $Ticket_{ui,fn}$.

$$\frac{Ui \textbf{ believes } TGS \textbf{ controls } Ticket_{ui,fn}, Ui \textbf{ believes } TGS \textbf{ believes } Ticket_{ui,fn}}{Ui \textbf{ believes } \ Ticket_{ui,fn}}$$

8. Finally, according to Rule 4*u*

$$Ui \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN$$

9. According to the message meaning rule: FN believes that $ks_{fn,tgs}$ is a shared session key with TGS, and FN sees the $\{Ticket_{ui,fn}\}$ is encrypted with $ks_{fn,tgs}$, and then FN believes that TGS once said $Ticket_{ui,fn}$.

$$\frac{FN \textbf{ believes } FN \xrightarrow{ks_{fn,tgs}} TGS, FN \textbf{ sees } \{Ticket_{ui,fn}\}_{ks_{fn,tgs}}}{FN \textbf{ believes } TGS \textbf{ said } Ticket_{ui,fn}}$$

10. Again, by the timestamp-verification rule: The FN believes that [TS] is fresh if FN believes that TGS once said $Ticket_{ui,fn}$. Then, FN believes that TGS believes $Ticket_{ui,fn}$.

$$\frac{FN \textbf{ believes fresh } (TS), FN \textbf{ believes } TGS \textbf{ said } \{Ticket_{ui,fn}\}_{ks_{fn,tgs}}}{FN \textbf{ believes } TGS \textbf{ believes } Ticket_{ui,fn}}$$

11. By Jurisdiction rule again: If FN believes that TGS has jurisdiction over $Ticket_{ui,fn}$ and FN believes that TGS believes $Ticket_{ui,fn}$, then FN believes $Ticket_{ui,fn}$.

$$\frac{FN \textbf{ believes } TGS \textbf{ controls } Ticket_{ui,fn}, FN \textbf{ believes } TGS \textbf{ believes } Ticket_{ui,fn}}{FN \textbf{ believes } Ticket_{ui,fn}}$$

12. Finally, according to Rule 4^u,

$$FN \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN$$

13. By Rule 1 (message meaning rule): If FN believes that the $(ks_{ui,fn})$ is a shared session key with Ui, then, FN sees that TS_3, $Ui \xrightarrow{ks_{ui,fn}} FN$ is encrypted with $ks_{ui,fn}$, and FN believes that Ui once said $Ui \xrightarrow{ks_{ui,fn}} FN$.

$$\frac{FN \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN, FN \textbf{ sees } \{TS_3, Ui \xrightarrow{ks_{ui,fn}} FN\}_{ks_{ui,fn}}}{FN \textbf{ believes } Ui \textbf{ said } Ui \xrightarrow{ks_{ui,fn}} FN}$$

14. By Rule 2: If FN believes that $Ui \xrightarrow{ks_{ui,fn}} FN$ is fresh and FN believes that Ui once said $Ui \xrightarrow{ks_{ui,fn}} FN$, then FN believes that Ui believes $Ui \xrightarrow{ks_{ui,fn}} FN$.

$$\frac{FN \textbf{ believes fresh } (Ui \xrightarrow{ks_{ui,fn}} FN), FN \textbf{ believes } Ui \textbf{ said } Ui \xrightarrow{ks_{ui,fn}} FN}{FN \textbf{ believes } Ui \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN}$$

Finally, we derive that FN believes that Ui believes.

$$FN \textbf{ believes } Ui \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN.$$

Similarly, we can get,

$$Ui \textbf{ believes } FN \textbf{ believes } Ui \xrightarrow{ks_{ui,fn}} FN.$$

The above demonstrates our authentication goal and proves that this scheme ensures that the user and the fog node are mutually communicated.

6.2. Informal Security Analysis

This section illustrates several security problems and shows that the proposed scheme is secure from various types of malicious attacks as follows:

Theorem 1. *The proposed scheme avoids the key escrow problem inherited by the Identity based cryptography (IBC).*

Proof of Theorem 1. As mentioned above, a distinctive user identity is assigned as u_{id} and biometric u_{bi}, while the assigned secret key is u_{pk}. Please note that u_{pk} is stored in the SC record and shared between the user and the fog server. Subsequently, the user itself randomly generates its secret key $u_{sk} \in Z_q^*$, which will later be kept a secret to AS. The secret key is generated based on the random number r_1, and the server has no access to it. In this way, other entities cannot extract r_1 from the published $bio_i = H(u_{bi} \oplus r_1)$ or $m_i = H(u_{id} \oplus u_{pw} \oplus bio_i \oplus r_1)$. Similarly, the authentication request is encrypted with the user public key. □

Theorem 2. *The proposed scheme is secure from a replay attack.*

Proof of Theorem 2. Assume that an adversary tries replaying the previously captured valid login and authentication messages $< u'_{id}, x'_i, bio'_i, ts_1 >$. The message is encrypted by using the server public key PK, and it takes a fresh timestamp ts_n to validate a legitimate user. After the server decrypts the message to obtain user information, AS will verify the received timestamp TS1 with the server's current stamp $TS\triangle$, $TS_1 \neq TS\triangle$. Likewise, if the adversary tries to replay the authentication message by replaying TS_1 with TS_2, it will not be able to pass because X'_i is encrypted using a one-way hash function $x_i^{'} = H(a_i^{'} \parallel f_i^{'} \parallel m_i^{'})$ so, the AS can detect any changes in the message. Hence, the scheme is resistant to a replay attack. □

Theorem 3. *The proposed scheme is secure from the impersonation attack.*

Proof of Theorem 3 . The adversary in this attack is trying to provide a login message by eavesdropping or computing a message to deceive the AS as a legal user. In the proposed scheme, if the adversary tries to replay the previous message or to impersonate $< u_{id}^{'}, x_i^{'}, bio_i^{'}, ts_1 >$, the AS will validate the message by checking $x_i^{'} \neq x_i$. Moreover, the adversary cannot capture the valid x'_i, due to the lack of the user identity u'_{id}, user password u'_{pw} and user biometric info u'_{bi}. Therefore, a malicious user cannot impersonate a legitimate user to access the fog node. □

Theorem 4. *The proposed scheme is resistant to a man-in-the-middle attack.*

Proof of Theorem 4 . Assume that the adversary intercepts the login and authentication messages successfully $\{u_{id}^{'} \parallel x_i^{'} \parallel bio_i^{'} \parallel tgs_{id} \parallel ts_1\}$, $\{u_{id}^{'} \parallel ks_{u \to tgs} \parallel ts_2 \parallel x_i \parallel bio_i^{'}\}$, and $\{u_{id}^{'} \parallel fn_{id} \parallel ts_3 \parallel tgs_{tkt}\}$. The adversary will fail because there is a crucial session KS established between all the entities and shared after the mutual authentication is generated between them. In addition, the ticket-granting service is sharing the encrypted ticket $tgs_{tkt} = E_{tgs_{sk}}\{u_{id}^{'} \parallel ks_{u \to tgs} \parallel ts_2 \parallel x_i \parallel bio_i^{'}\}$ using the TGS secret key and only can only be decrypted by it, which goes for the communication between TGS and FN. However, for the same reason mentioned above, the attacker cannot pass this process without knowing the patient's uid and the personal values x_i and TS. Therefore, the adversary cannot cheat the user Ui to share a key session and believe that the key is shared with the authentication server AS, and this judgment also works on the FN. Therefore, the adversary cannot launch the man-in-middle attack successfully to cheat either the user or the servers in the proposed scheme. □

Theorem 5. *The proposed scheme withstands the known-key attacks.*

Proof of Theorem 5 . The proposed scheme provides resistance against known-key session attacks according to the unique key session that has been generated between each entity. The key session in the proposed scheme is calculated based on $< a_i^{'} \parallel r_2 >$ that makes it unique because the random integer r is generated randomly and independently by the Ui, AS, and TGS. Since the r_1, r_2, and r_3 are different from each other, the critical session in each run is unique in the proposed scheme. Therefore, the proposed scheme is resisting this attack. Using a unique key session during every communication session allows achieving freshness of the key in the proposed scheme. The session keys are generated in AS $ks_{u \to tgs} = H(u_{id}^{'} \parallel a_i^{'} \parallel r_2)$ and TGS $ks_{u \to fn_S} = H(fn_{id} \parallel a_i \parallel r_3)$, differently and independently. □

Theorem 6. *The proposed scheme withstands privileged insider attacks.*

Proof of Theorem 6. Assume that the adversary attempts to obtain the legal user information u_{id} and u_{pw}, but he/she will fail to impersonate since he/she must provide the correct biometrics u_{bi} of the targeting user. In addition, it will be difficult for the attacker to obtain the legal user information since the proposed scheme is performing a hash function on the user information $H(u_{id} \oplus u_{pw} \oplus bio_i \oplus r_1)$ and contains a randomly generated number in it. The user biometric is protected as well in the hash formatting with the random number r_1. The attacker cannot extract the stored parameters from the stored hash value successfully; thus, the proposed scheme works against any insider attack. □

Theorem 7. *The proposed scheme withstands a stolen smart card attack.*

Proof of Theorem 7. In this attack, the adversary attempts to extract the user information stored in the smart card. He/she will fail since the parameters $\{x_i, n_i, R_i, H(.), q, p\}$ are secured and the attacker cannot successfully compute $H(a_i \parallel f_i \parallel m_i) = x_i, f_i \oplus H(u_{id} \parallel m_i) = n_i$, and $a_i \oplus H(bio_i \parallel m_i) = R_i$ as they are secured using a collision-resistant one-way hash function $H(.)$. Therefore, the attacker is unable to determine the user information u_{id} and u_{pw}. Therefore, the proposed scheme is resistant to the stolen smart card attack. □

Theorem 8. *The proposed scheme is secure from a replay attack.*

Proof of Theorem 8. The proposed scheme is resistant to a server spoofing attack. An adversary exploits a legitimate user's information to counterfeit as a server. To successfully impersonate as an authentication server AS, it cannot compute the $(u_{id}^{'} \parallel tgs_{id} \parallel E_{u_{pk}}\{ks_{u ß tgs} \parallel ts_2 \parallel tgs_{tkt}\}$, and $tgs_{tkt} = E_{tgs_{sk}}\{u_{id}^{'} \parallel ks_{u \to tgs} \parallel ts_2 \parallel x_i \parallel bio_i^{'}\}$; to compute the correct values, a malicious server needs to know the critical session KS, timestamp TS, and the secret value Xi. The values are encrypted and cannot be decrypted by only using the secret server key that is shared earlier. As mentioned above, he/she will need to know the user identity $u - id$ to compute critical sessions. □

Theorem 9. *The proposed scheme withstands a Denial of Service (DoS) Attack.*

Proof of Theorem 9. This attack can suspend services of the server by flooding the network, but the proposed scheme is resistant to this attack. Since the proposed scheme verifies the user identity u_{id} and password u_{pw}, it also verifies the secret value $X_i^{'} \neq X_i$ so he/she fails. Moreover, the server will detect a false message sent to it by the adversary using the timestamp (TS1, TS2, and TS3). The authentication server AS and FN will only proceed if the login message passes the check $(TS_1 - T_{curr}) \leq \triangle TS$, and the Ticket granting server $(TS_2 - T_{curr}) \leq \triangle TS$. The FN will also not process the message only if the shared

timestamp matches the shared timestamp TS3. In addition, the TGS will check the validity of the $X_i^{'} \neq X_i$ after the AS. □

Theorem 10. *The proposed scheme is secure from an offline password guessing attack.*

Proof of Theorem 10. The user identity u_{id}, password u_{pw}, and the biometric u_{bi} are secured using a one-way hash function $H(u_{bi} \oplus R_i) = bio_i, H(u_{id} \oplus u_{pw} \oplus bio_i \oplus r_1) = m_i$ and contains a random number in it. In addition, any alteration in the login message will be detected after the server verification. The attacker can never validate the password with a stolen smart card SC. If the attacker intercepts the login message SC $\{X_i, n_i, R_i, H(.), q, p\}$ as a legitimate user, he/she will need to guess the user identity and password that server will validate the message after decrypting. Thus, the proposed scheme is highly secured against offline password guessing attacks. □

Theorem 11. *The proposed scheme facilitates user anonymity.*

Proof of Theorem 11. Assume an adversary intercepts the message $\{u_{id}^{'} \parallel x_i^{'} \parallel bio_i^{'} \parallel tgs_{id} \parallel ts_1\}$. The attacker cannot obtain the information because the authentication message is encrypted using the authentication server public key $E_{PK}\{\}$. In addition, the server will check the validity of the user by extracting the original user's identity u_{id} and the secret value $x_i^{'} \neq x_i$ as well. The AS and TGS will generate a key session using a unique random number with every communication session. Moreover, the adversary cannot launch a guessing attack to obtain the user information, because with knowledge of x_i, an adversary cannot compute $H(u_{id} \oplus u_{pw} \oplus bio_i \oplus r_1) = m_i$ successfully according to the using of a one-way hash function as well. Therefore, nobody will be able to know the real identity of the user, except the user himself and the server. □

Theorem 12. *The proposed scheme facilitates against user traceability attacks.*

Proof of Theorem 12. The proposed scheme protects the real user identity, and the transmitted message is changed by updating r during every session. The transmitted messages are different from one session to the other since there is a new key session. KS is computed based on a new random number when every new session begins. Therefore, the attacker cannot distinguish whether the intercepted messages belong to the same user or not. Therefore, the proposed scheme provides user untraceability. □

Theorem 13. *The proposed scheme facilitates a mutual authentication property.*

Proof of Theorem 13. The authentication scheme needs to allow all the considered entities in communication to verify the identity of each other mutually. The use of ECC is in providing mutual authentication. The user and the server can authenticate each other by checking session key freshness $ks = H(u_{id}^{'} \parallel a_i^{'} \parallel r)$ in every session, and verifying the $x_i \neq x_i^{'}$ with the timestamp, respectively, in the AS and the TGS. Therefore, the proposed scheme achieves mutual authentication. □

Theorem 14. *The proposed scheme achieves perfect forward secrecy property.*

Proof of Theorem 14 . In the proposed scheme, the adversary cannot generate the key session $ks_{u \to tgs} = H(u_{id}^{'} \parallel a_i^{'} \parallel r_2)$, $ks_{u \to fn_S} = H(fn_{id} \parallel a_i \parallel r_3)$ because the adversary does not know the user identity. Therefore, the attacker cannot obtain the user/server identity. To successfully compute the key session, the attacker needs a secret value $A_i^{'}$, and random number r, but he/she will fail because it is impossible to obtain the random number and the value $A_i^{'}$ is secured using the one-way hash function $a_i^{'} = R_i^{'} \oplus H(bio_i^{'} \parallel m_i^{'})$

In addition, it includes protected biometric. As a result, it is difficult to determine the information, and the proposed scheme is achieving forward secrecy. □

Theorem 15. *The proposed scheme achieves a biometric protection property.*

Proof of Theorem 15. The user biometric $u_{bi}^{'}$ is highly protected by a high entropy random number integer r and one-way hash function $bio_i^{'} = H(u_{bi}^{'} \oplus r_1)$. Assume the adversary obtains the stored information on the smart card; but he/she cannot extract the User biometric $u_{bi}^{'}$ without the knowledge of the user identity $u_{id}{}'$ and password $u_{pw}^{'}$. Therefore, the proposed scheme of protects the user's biometric. □

7. Formal Security Verification Using AVISPA Tool: Simulation Study

Simulations were carried out to test the proposed security framework using AVISPA [39], an extensively used security analysis model. It proves that the scheme avoids replay and man-in-the-middle attacks. This section includes a simple overview of the AVISPA tool [40]. It then shows the implementation code for the User (U), authentication server (AS), ticket-granting server (TGS), fog node (FN), session, goals, and the environment in High-Level Protocol Specification Language (HLPSL). Third, the simulation results are demonstrated.

7.1. AVISPA Tool Basic Explanation

AVISPA is a simulation verification tool to validate authentication schemes. Specification language (HLPSL) is used to implement the simulation code. AVISPA is a participant-related program. Each participant is autonomous and has some knowledge across channels in the form of communication parameters. First, wrote the code into HLPSL in this tool, and then used hlpsl2if to translate it into an intermediate (IF) format. AVISPA is currently being introduced in four back ends: (a) CL-AtSe; (b) OFMC; (c) SATMC; and (d) TA4SP. AVISPA is implemented on a backend basis on-the-fly model checker (OFMC); the output format is generated and then represented based on these back-ends, confirming that the system is safe from active and passive attacks.

7.2. Discussion of Proposed Scheme in HLPSL

The role of user U in HLPSL is shown in Figure 8a. In the registration phase, U sends $\{u_{id}, u_{pw}, bio_i, m_i\}$ to the authentication server AS using $Snd()$ operation via a secure channel. The declaration channel (dy) is made for the Dolev–Yao threat model. Accordingly, two declaration secrets, i.e., $(K\_UG', sec\_c\_K\_UG, AS, U, TGS)$, $(K\_US', sec\_c\_K\_US, TGS, U, S)$ state that Bio, u_{pw} is only known to U, D_S is only known by AS and U_ID is known to U, AS, and TGS. After that, U obtains the smart card having the values, i.e., x_i, N_i, R_i, from AS. In the login phase, the user further creates $N2', T', Mi', TGSid', Ri', Ni', Uid'$ using a new (-) operation and transmitting $(Uid^{'}.N2'.\ Uid'.Bio'.TGSid'.U.T'\_K\_UG')$ to the AS via a public channel. The declaration witness $(U, TGS, t1, T')$ tells that U creates T for AS. In the authentication phase, U gets a reply message $(U.TkT2'.S.K\_US'.Uid'.Bio'.Ts2'.Xi'.Tse2'.N2\_K\_UG)$ from AS by using Rcv () operation. Further, the user creates $T2', TGSid', FNid', Uid'$ and transmits the message $Snd(TkT2'.U.T2'.Uid'.TGSid'.FNid'\_K\_US')$ to the TGS. The declaration request (U, TGS, k_cs1, K_US') states that the user sends a request to the TGS for knowing K_US'. The declaration secret $(K\_US', sec\_c\_K\_US, TGS, U, S)$ states that T is known to U, AS, and TGS. The user later receives a message $(Uid'.TkT1'T2\_K\_US)$ from the TGS. The user then messages to the fog node $(TkT2'.U.T2'.Uid'.FNid'\_K\_US')$. The declaration request (U, S, t2a, T2) states that the user sends a request to the fog node.

Figure 8b depicts the role of the authentication server AS in HLPSL. In the login phase, AS receives $(U.TGS.Lifetime_1^{'}.N1^{'}.Uid^{'}.Bio^{'}.TGSid^{'}.Xi^{'})$ from the user. Then, AS create $(Ts', Tse', K\_UG', Sk', Mi', Bio', Uid')$ and sends $(TkT2'.U.T2'.Uid'.TGSid'\_K\_US')$ to the user. Moreover, the declaration witness (AS, U, k_cg1, K_UG') and (AS, TGS, k_cg2, K_UG') indicates that AS generates a symmetric key for user U and TGS. Furthermore, the decla-

ration secret $(K\_UG', sec\_a\_K\_UG, AS, U, TGS)$ states that AS,U,TGS knows the value of K_UG'.

```
% User
role k_user (U, AS, TGS, FN : agent,
             Snd, Rcv   : channel (dy),
             K_UA       : symmetric_key)
played_by U
def=
 local St   : nat,
    K_UG, K_US : symmetric_key,
    T, T2: text,
    H:hash_func,
    Ts, Tse, Ts2, Tse2    : text,
    TkT1, TkT2 : {agent.agent.symmetric_key.text.text}_symmetric_key,
    N1, N2 ,Bio,UBi,Ri,Uid,Upw,Mi,Xi,Ai,Fi,Ni,TGSid,FNid : text
 const sec_c_K_UG, sec_c_K_US : protocol_id,
      cLifetime_1, cLifetime_2: text
 init  St := 0
 transition
%%%registeration Phase
  1. St = 0  /\ Rcv(start) =|>
    St':= 1 /\ N1' := new()
          /\ Ri':=new()
          /\ Bio':=H(xor(UBi,Ri'))
          /\ Mi':=H(xor(Uid.Upw.Ri'))
          /\ Snd(U.TGS.cLifetime_1.Bio'.Mi'.N1')
%%% recieved message
  2. St = 1  /\ Rcv(U.TkT1'.{TGS.K_UG'.Ts'.Tse'.N1.Xi.Ri}_K_UA) =|>
%%%login and authentication phase
    St':= 2 /\ N2' := new()
          /\ T' := new()
          /\ Mi':= new()
          /\TGSid':=new()
          /\ Bio':= new()
          /\ Ri':= new()
          /\ Ai':= xor(Ri',H(Bio'.Mi'))
          /\ Ni':=new()
          /\ Uid':=new()
          /\ Fi':= xor(Ni',H(Uid'.Mi'))
          /\ Xi':=H(Ai'.Fi'.Mi')
          /\ Snd(FN.cLifetime_2.N2'.Uid'.Bio'.TGSid'.TkT1'.{U.T'}_K_UG')
          /\ witness(U,TGS,t1,T')
          /\ wrequest(U,AS,k_cg1,K_UG')
          /\ secret(K_UG',sec_c_K_UG,{AS,U,TGS})
  3. St = 2  /\ Rcv(U.TkT2'.{FN.K_US'.Ts2'.Tse2'.N2}_K_UG)  =|>
    St':= 3 /\ T2' := new()
           /\ TGSid':=new()
           /\ FDid':=new()
           /\ Uid':=new()
           /\ Snd(TkT2'.{U.T2'.Uid'.TGSid'.FNid'}_K_US')
           /\ witness(U,FN,t2b,T2')
           /\ wrequest(U,TGS,k_cs1,K_US')
           /\ secret(K_US',sec_c_K_US,{TGS,U,FN})
  4. St = 3  /\ Rcv({T2}_K_US) =|>
    St':= 4 /\ wrequest(U,FN,t2a,T2)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(**a**) User role.

```
% Authentication Server
role k_AS (AS, U, TGS : agent,
           Snd, Rcv   : channel (dy),
           K_UA, K_AG : symmetric_key)
played_by AS
def=
 local St           : nat,
     K_UG           : symmetric_key,
     N1, Lifetime_1,Ai,Sk,Uid,Bio,Fi,Ri,Mi,Ni,Xi   : text,
     H:hash_func,
     Ts, Tse : text
 const k_cg1, k_cg2 : protocol_id,
     sec_a_K_UG : protocol_id

 init  St := 0
 transition
  1. St = 0  /\ Rcv(U.TGS.Lifetime_1'.N1') =|>
    St':= 1 /\ Ts' := new()
          /\ Tse' := new()
          /\ K_UG' := new()
          /\ Sk':= new()
          /\ Ai':= H(Uid,Sk')
          /\ Fi':= H(Bio,Sk')
          /\ Mi':= new()
          /\ Bio':=new()
          /\ Ri':= xor(Ai',H(Bio'.Mi'))
          /\ Uid':= new()
          /\ Ni':= xor(Fi',H(Uid'.Mi'))
          /\ Xi':=H(Ai'.Fi'.Mi')
          /\ Snd(U.{U.TGS.K_UG'.Ts'.Tse'}_K_AG.
                   {TGS.K_UG'.Ts'.Tse'.N1'}_K_UA)
          /\ witness(AS,U,k_cg1,K_UG')
          /\ witness(AS,TGS,k_cg2,K_UG')
          /\ secret(K_UG',sec_a_K_UG,{AS,U,TGS})
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(**b**) AS role

Figure 8. The User and AS roles in HLPSL.

Figure 9a presents the role of the Ticket granting server TGS in HLPSL. The role starts from the authentication phase, TGS receives $(S.Lifetime_2'.N2'.U.TGS.K\_UG'.Ts'.Uid', Xi', Bio'\_K\_AG.U.TkT2'\_K\_UG')$. Then, TGS transmits $U.U.S.K\_US'.Ts2'.Uid', FNid', Xi'\_K\_GS.S.K\_US'.Ts2'Uid', FNid', Xi'.Tse2'.N2'\_K\_UG')$ to the user U. The request declaration $(TGS, U, t1, T')$ states that the user U sends a request to TGS for knowing T', where the request (TGS, AS, k_cg2, K_UG') states that AS sends a request to TGS for knowing the K_UG'. The declaration witness (TGS, U, k_cs1, K_US') indicates that the TGS generates K_US' for U, where the declaration witness (TGS, S, k_cs2, K_US') specifies that TGS generates K_US' for the fog node server FN. Furthermore, the declaration secret $(K\_UG', sec\_g\_K\_UG, AS, U, TGS)$ states that AS, U, TGS know the K_UG', while the $(K\_US', sec\_g\_K\_US, TGS, U, S)$ states that TGS, U, S know the value K_US'.

Figure 9b depicts the role of the fog node FN in HLPSL. FN receives the message $(U.FN.K\_US'.Ts2'.Tse2'.Uid', FNid', Xi'\_K\_GS.U.T2'\_K\_US')$ from the user. Then, FN provides the user message $(T2'_K_US')$and transmits it to the user. The declaration witness $(FN, U, t2a, T2')$ indicates that FN generates text T (Fail/Success) for the user. The declaration request (FN, TGS, k_cs2, K_US') states that TGS sends a request to FN for knowing K_US', where the declaration request $(FN, U, t2b, T2')$ states that the user sends a request to S for knowing the feedback of the fog node (Fail/Success). Moreover, the declaration secret $(K\_US\prime, sec\_K\_US, TGS, U, FN)$ states that K_US' is known to TGS, U, and FN. Figure 10 demonstrates the roles of session, goals, and environment in HLPSL.

```
% Ticket Granting Server
role k_TGS (TGS, AS, FN, U  : agent,
           Snd, Rcv    : channel (dy),
           K_AG, K_GS  : symmetric_key)
played_by TGS
def=
 local St                          : nat,
     K_UG                          : symmetric_key,
     K_US                          : symmetric_key,
     Lifetime_2, Ts, Tse, T, N2,Ai,Sk,Uid,Bio,Fi,Ri,Mi,Ni,Xi  : text,
     H:hash_func,
     Ts2, Tse2              : text
 const t1, k_cs1, k_cs2 : protocol_id,
     sec_g_K_UG, sec_g_K_US : protocol_id

 init  St := 0
 transition
  1. St = 0  /\
    Rcv(FN.Lifetime_2'.N2'.{
            U.TGS.K_UG'.Ts'.Tse'}_K_AG.{U.T'}_K_UG') =|>
   St':= 1 /\ K_US' := new()
         /\ Ts2' := new()
         /\ Tse2' := new()
         /\ Sk':= new()
         /\ Ai':= H(Uid,Sk')
         /\ Fi':= H(Bio,Sk')
         /\ Mi':= new()
         /\ Bio':=new()
         /\ Snd(U. {U.FN.K_US'.Ts2'.Tse2'}_K_GS.
                    {FN.K_US'.Ts2'.Tse2'.N2'}_K_UG')
              /\ wrequest(TGS,U,t1,T')
              /\ wrequest(TGS,AS,k_cg2,K_UG')
              /\ witness(TGS,U,k_cs1,K_US')
              /\ witness(TGS,FN,k_cs2,K_US')
              /\ secret(K_UG',sec_g_K_UG,{AS,U,TGS})
              /\ secret(K_US',sec_g_K_US,{TGS,U,FN})
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(**a**) TGS role

```
% Fog node
role k_FS (FN, TGS, U  : agent,
           Snd, Rcv : channel (dy),
           K_GS     : symmetric_key)
played_by FN
def=

 local St                : nat,
     Ts2, Tse2, T2,Ai,Sk,Uid,Bio,Fi,Ri,Mi,Ni,Xi,TGSid,FNid : text,
     H:hash_func,
     K_US                : symmetric_key

 const t2a, t2b : protocol_id,
     sec_s_K_US : protocol_id

 init  St := 0

 transition

  1. St = 0  /\ Rcv({U.FN.K_US'.Ts2'.Tse2'}_K_GS.{U.T2'}_K_US') =|>
   St':= 1 /\ TGSid':=new()
         /\ FDid':=new()
         /\ Uid':=new()
         /\  Snd({T2'}_K_US')
              /\ witness(FN,U,t2a,T2')
              /\ wrequest(FN,TGS,k_cs2,K_US')
              /\ wrequest(FN,U,t2b,T2')
              /\ secret(K_US',sec_s_K_US,{TGS,U,FN})

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(**b**) Fog Node role

Figure 9. The TGS and Fog node roles in HLPSL.

Goals:

- sec_a_K_UG: It tells that the AS, U, and TGS know K_UG′.
- sec_g_K_UG: It states that K_UG′ is shared among TGS, U, and FN.
- sec_c_K_UG: It shows that AS, U, and TGS know the value K_UG′.
- sec_c_K_US: It tells that the TGS, U, and FN know the K_US′.
- sec_g_K_UG: It states that the K_UG′ is shared among AS, U, and TGS.
- sec_g_K_US: It tells that TGS, U, and FN know the value K_US′.

Authentications:

- authentication_on k_cg1: The Ticket is only shared between the user and the TGS.
- authentication_on k_cg2: The timestamp is only valid for the User to the TGS authentication session.
- authentication_on k_cs1: The user sends the second ticket is only known by the User and the FN.

- authentication_on k_cs2: The timestamp is valid only for the authentication session to FN.
- authentication_on t2a: The first TS is replaced with a fresh one between the User and the TGS.
- authentication_on t2b: The User generates a fresh timestamp to authenticate to the FN.
- authentication_on t1: FN replies a text to the user about successful or failed authentication.

```
role session( U, AS, TGS, FN                    : agent,
              K_UA, K_AG, K_GS      |   : symmetric_key)
def=
  local S_C, R_C, S_A, R_A, S_G, R_G, S_S, R_S : channel (dy)

  composition
     k_user(U,AS,TGS,FN,S_C,R_C,K_UA)
   ∧ k_AS(AS,U,TGS,S_A,R_A,K_UA,K_AG)
   ∧ k_TGS(TGS,AS,FN,U,S_G,R_G,K_AG,K_GS)
   ∧ k_FS(FN,TGS,U,S_S,R_S,K_GS)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role environment() def=
  const  u, as, tgs, fn, i        : agent,
              kca, kag, kgs, kia     : symmetric_key

  intruder_knowledge = {u,as,tgs,fn,kia
                        }
  composition
              session(u,as,tgs,fn,kca,kag,kgs)
∧             session(i,as,tgs,fn,kia,kag,kgs)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
  secrecy_of sec_a_K_UG,
          sec_g_K_UG, sec_g_K_US,
          sec_s_K_US,
          sec_c_K_UG, sec_c_K_US

  weak_authentication_on k_cg1
  weak_authentication_on k_cg2
  weak_authentication_on k_cs1
  weak_authentication_on k_cs2
  weak_authentication_on t2a
  weak_authentication_on t2b
  weak_authentication_on t1

end goal
environment()
end role
```

Figure 10. The session, goals, and environment roles in HLPSL.

7.3. Simulation Results

Figures 11 and 12 show our proposed scheme's results with simulation results in OFMC and CL-AtSe back ends. These back-ends show that our scheme secure from active and passive attacks. The sequence diagram of the proposed scheme shown in Figure 13 is represented as user U, authentication server AS, ticket-granting server TGS, and fog node FN.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/3FACD.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.06s
  visitedNodes: 28 nodes
  depth: 5 plies
```

Figure 11. The simulation result using OFMC back-end.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/3FACD.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 48 states
  Reachable  : 13 states
  Translation: 0.02 seconds
  Computation: 0.00 seconds
```

Figure 12. The simulation result using CL-AtSe back-end.

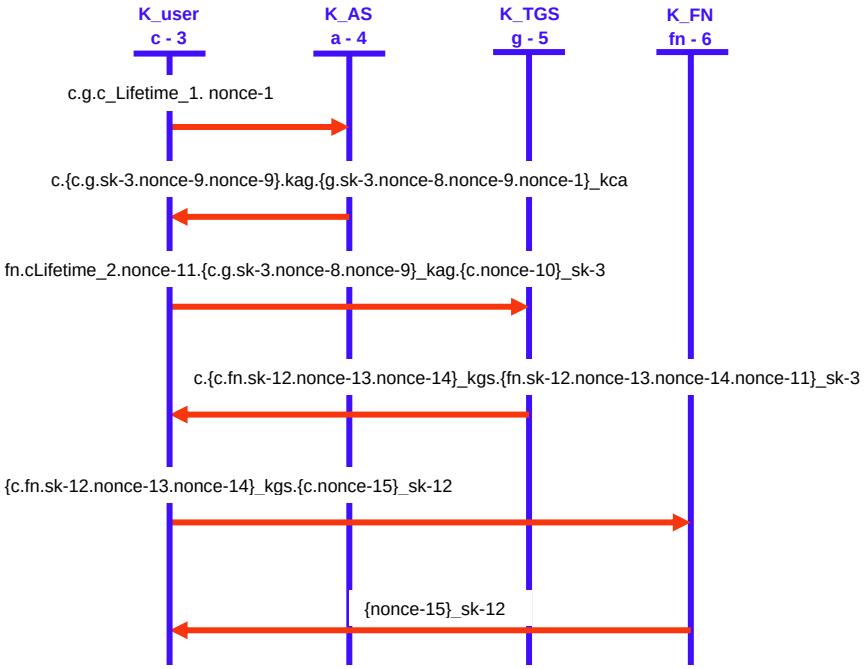


Figure 13. The simulation sequence using AVISPA.

8. Security Features Comparison

The security features comparison is shown in Table 4. The schemes SAKA-FC [26] and AKA-FC [31] highly suffer from a key escrow problem and key encryption management. The scheme is also vulnerable to identity and password guessing attacks, replay Attacks, impersonation Attacks, insider Attacks, and DoS. Moreover, the cryptoanalysis shows that the scheme SAKA-FC [26] suffers from user anonymity and untraceability. The schemes SAKE [20] and AKA-FC [31] are vulnerable to stolen smart card attacks, offline password guessing attacks, and missing mutual authentications. The schemes SAKA-FC [26] and SAKE [20] do not facilitate perfect forward secrecy and biometric protection. Since the proposed scheme is validated using BAN logic, this ensures secure preservation of mutual-authentication and key session agreement. The proposed scheme is also simulated by the web tool AVISPA [40], whose simulation results indicate that it is defended against active and passive attacks. The proposed solution is protected against various security threats.

Table 4. Comparison on security properties.

	SAKA-FC [26]	SAKE [20]	AKA-FC [31]	AKA-MS [32]	SELAMAT
Key escrow	×	✓	×	×	✓
Replay attack	×	✓	✓	×	✓
Impersonation attack	×	✓	×	✓	✓
Man-in-the-middle attack	✓	✓	✓	×	✓
Known-key attack	✓	✓	✓	✓	✓
Insider attack	×	✓	×	✓	✓
Stolen smart card attack	✓	×	✓	✓	✓
Server spoofing attack	✓	✓	×	×	✓
Denial of service (dos) attack	×	✓	×	×	✓
Offline password guessing attack	✓	×	×	✓	✓
User anonymity	×	✓	✓	✓	✓
User untraceability	×	✓	✓	✓	✓
Mutual authentication	✓	×	×	✓	✓
Perfect forward secrecy	×	×	✓	×	✓
Biometric protection	×	×	×	×	✓
Cross-platform authentication	×	×	×	×	✓

8.1. Computation and Communication Costs

Here, we explain the comparison of the communication cost and the computation cost of the proposed scheme with other existing schemes [20,26,31,32] which are shown in Table 5. The performance metrics can be explained as follows.

8.1.1. Computation Cost

In this subsection, we analyze the computation cost of the related authentication schemes SAKA-FC [26], SAKE [20], AKA-FC [31], and AKA-MS [32] and our proposed scheme. The number of cryptographic operations involved in this study are counted. To represent the comparison, Table 6 shows the notations, description, and computed their approximate execution time for various cryptographic operations by using the PBC library reported by Jia et al. [41]. Specifically, the study employed a secure hash function, public-key-based encryption, symmetrical encryption, and symmetric decryption, which are, respectively, denoted as T_H, T_{PE}, T_{SE}, and T_{SD}. It is noted that the XOR operation and concatenates operation $\|$ are ignored because their execution time is negligible. The proposed scheme's simulation was carried out on Intel Core™i7-5700HQ, CPU 2.70GHz platform using Java Pairing-Based Cryptography Library (JPBC) library. Figure 14 compares the proposed scheme's computation cost against SAKA-FC, SAKE, and AKA-FC.

Our scheme's computation cost is shown in Table 5 comparing it to other authentication schemes.

Table 5. Performance comparisons.

Scheme		Computational Cost	Communication Cost (bits)	Total
SAKA-FC [26]	Login phase	$2T_{sm} + 26T_h + 1T_{mtp} \approx 35.595$ ms	2240 bits	2816
	Authentication phase	$9T_h \approx 4.005$ ms	376 bits	
AKA-FC [31]	Login phase	$2ET_{sm} + 4T_h + T_P \approx 9.251$ ms	1376 bits	2752
	Authentication phase	$3ET_{sm} + 11T_h + T_P \approx 11.284$ ms	1376 bits	
SAKE [20]	Login phase	$8T_h + 2T_{pm} + 2T_{fe} \approx 40.76$ ms	1376 bits	2584
	Authentication phase	$20T_h + 4T_{pm} \approx 4.295$ ms	1576 bits	
AKA-MS [32]	Login phase	$8H_M \approx 7.792$ ms	928 bits	1920
	Authentication phase	$7H_M + 1H_p \approx 19.236$ ms	992 bits	
SELAMAT	Login phase	$1H_P + 1H_M + 5T_H + 1T_{PE} + 4T_{SE} + 2T_{SD} \approx 17.292$ ms	624 bits	1336
	Authentication phase	$2T_H + T_{AV} + 2T_{SE} + 3T_{SD} \approx 0.232$ ms	712 bits	

Table 6. Computation time consumption.

Description	Time (ms)
Identity-based signature (T_{IDS})	23.866
Identity-based signature verification (T_{IDV})	5.8720
Asymmetric signature (T_{AS})	3.8500
Asymmetric signature verification (T_{AV})	0.1925
Public-key-based encryption (T_{PE})	3.8500
Public key-based decryption (T_{PD})	3.8500
symmetrical encryption (T_{SE})	0.0046
Symmetric decryption (T_{SD})	0.0046
Scalar multiplication $(T_{(sm-ecc)})$ in G_1	0.4420
Scalar multiplication (T_{sm})	20.2300
ECS scalar multiplication (ET_{sm})	1.9700
Exponentiation Operations (T_e)	1.2950
Bilinear pairing (T_P)	4.2110
Map-to-point hash function (T_{mtp})	4.4060
Fuzzy extractor (T_{fe})	0.0023
$H_n : \{0,1\}^* \rightarrow Z_n$	0.0023
$H_P : \{0,1\} \rightarrow G_1$	12.4180
$H_M : \{0,1\}^* \rightarrow G_2$	0.9740
$H_S : \{0,1\}^* \rightarrow \{0,1\}^*$	0.0046

In SAKA-FC [26], three cryptographic operation are scheme, T_{sm}, T_h, and T_{mtp}, respectively, as shown in Table 5. The execution times of these operations are 0.442, 1.709, and 4.406 ms, respectively. In the login phase, the user firstly needs to execute the Scalar multiplication (T_{sm}) six times, Map-to-point hash function (T_{mtp})one, and the hash operation (T_h) twenty-six times related to SAKA-FC to start login into the system, so the execution time of the phase costing nearly $\approx$ 35.595 ms. In the authentication phase, the user needs to execute the hash operations (T_h) nine times related to G_1. Therefore, the execution time of the authentication process in SAKA-FC is $\approx$ 4.005 ms. Therefore, the total computation cost of their scheme is 39.595 ms. The computation of their scheme is computationally high due to the used multiplication operation in the scheme. In AKA-FC [31], there are three cryptographic operations related to ECC used in their scheme, ET_{sm}, T_h, and T_P, respectively. Table 5 shows the estimated execution time of the performed operations individually. However, in the login phase, the user needs to perform the scale

multiplication related Elastic compute service (ECS) $2ET_{sm}$ twice, the hash function $4T_h$ four times, and the bilinear pairing T_P once. Therefore, the execution time in the login phase is $2ET_{sm} + 4T_h + T_P = 9.251$ ms. In the authentication phase, the scheme needs to perform the scale multiplication related Elastic compute service (ECS) $2ET_{sm}$three times, the hash function $4T_h$ eleven times, and the bilinear pairing T_P once; thus, the time cost for this phase is $3ET_{sm} + 11T_h + T_P = 11.284$ ms. Therefore, the total computational cost of AKA-FC is 20.535 ms.

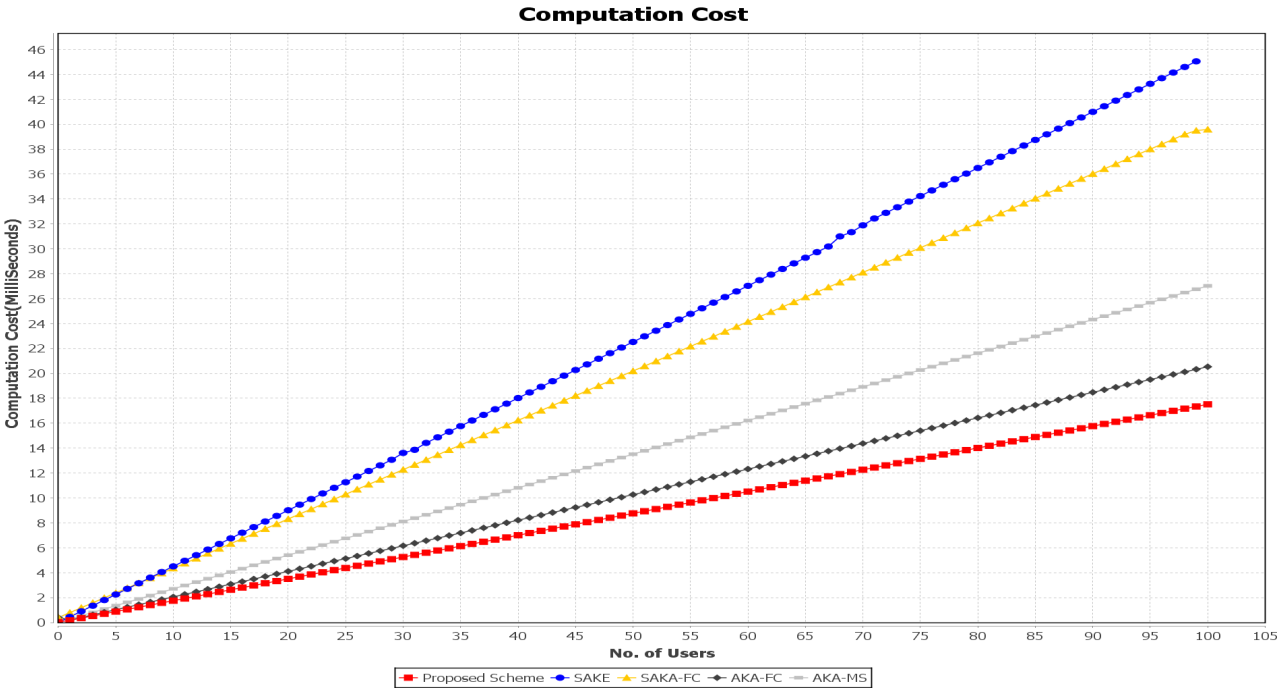


Figure 14. Computation costs comparison of SELAMAT against SAKA-FC, SAKE, AKA-FC, and AKA-MS.

In SAKE [20], the scheme employed secure hash functions, multiplication, and fuzzy extractor operation are, respectively, denoted as T_h, T_{pm}, and T_{fe} and are mainly related to fuzzy extractor algorithm. However, in the login phase, the user needs to execute point multiplication T_{pm} twice, the hash function $8T_h$ eight times, and the fuzzy extractor operations $2T_{fe}$ twice. Consequently, the execution time in this phase is approximately 40.76 ms. In contrast, the authentication phase's execution time is $\approx$ 4.295 ms, since the utilized operations are T_H and T_{pm}. It is noted that the login phase takes longer than the authentication phase in this scheme under the usage of the multiplication operation. Therefore, the total execution time in the SAKE scheme is 45.055 ms. The AKA-MS scheme [32] uses two cryptographic operations: hash operation related to bilinear pairing H_P and hash operation related to the group of ECC H_M. The estimated execution times are 12.418 and 0.974 ms, individually. The user needs to execute the hash operations $8H_P$ eight times in the login phase. In comparison, there is a need to execute the hash operations $7H_P$ seven times and the hash operation $1H_M$ once related to the ECC group in the authentication phase. The total computation cost is $8H_M + 7H_M + 1H_p = 27.028$ ms, while the proposed scheme applied a very lightweight operation T_H, H_M,T_{PE}, H_P, T_{SE}, and T_{SD}. These operations' execution times independently are 12.4180, 0.9740, 0.9740, 3.8500, 0.0046, and 0.0046 ms. However, in the first phase, the user needs to perform the hash function five times, public key encryption one time, the symmetrical encryptions four times, and the symmetrical decryption described as $1H_P + 1H_M + 5T_H + 1T_{PE} + 4T_{SE}$ twice. Therefore, the execution time of the login phase is nearly $\approx$ 17.2920 ms. In the authentication phase, the employed operations are T_H, T_{AV}, T_{SE}, and T_{SD}. The verification operation is included in this phase. However, the computation cost here is $2T_H + T_{AV} + 2T_{SE} + 3T_{SD}$. Therefore, the execution time for user operation is $\approx$ 0.232 ms. Thus, the total execution time of the proposed scheme

is 17.524 ms. Compared to SAKA-FC [26], SAKE [20], AKA-FC [31], and AKA-MS [32], the proposed scheme has less computation cost. According to utilizing the AES-ECC algorithm, this result was achieved due to the fast AES encryption speed that makes it suitable for encryption of long plain-text. The ECC solution also uses a smaller key size and low computational system requirements, making it faster and more efficient cryptographic keys.

8.1.2. Communication Cost

The number of message interactions measures the communication costs. To compute the communication cost, it mainly depends on how many messages are transmitted between the entities multiplied by the (bit) size. We assume that the user's identity can be represented by 32 bits, the secret value represented using 160 bits, the timestamp value is 24 bits, and the ticket value is represented as 128 bits. The communication cost of SAKA-FC [26], SAKE [20], AKA-FC [31], AKA-MS [32], and the proposed scheme are summarized in Table 5. The compassion of the proposed protocol's communication cost against the selected works is shown in Figure 15. In SAKA-FC [26], the scheme exchanging messages $Msg.1 =< RID_i^{'}, R_u, a_u, E_u, F_u, TS_u >$, needs $(160 + 320 + 160 + 160 + 160 + 32) = 992$ bits, while in the login and authentication phase exchange the message $Msg.2 =< RID_i^*, RID_k^{'}, G_i, H_j, P_f, TS_f >$ needs $(160 + 160 + 160 + 160 + 320 + 32) = 992$ bits. The message $Msg.3 =< RID_k^*, M_k, N_k, P_f, TS_k >$ needs $(160 + 160 + 160 + 320 + 32) = 832$ bits. Therefore, the total communication cost of their scheme is $2 \times (992 + 992 + 832) = 2816$ bits. To evaluate the communication cost, the scheme AKA-FC [31] includes the length of the points in the group G_1, which is 1024 bits, the output of the hash function $2|q|$, which has the length of 160 bits, and the length of the timestamp, being 32 bits, which is denoted as $|T|$. Thus, the communication cost in the login phase is 1376 bits. In the authentication phase, the user performs the same length of messages, which is represented as $|G1| + 2|q| + |T|$ and has the length of 1346. Therefore, the total communication cost of AKA-FC is 2752 bits. In SAKE [20], the initialization calculations on the user parameters set $(TS_2^i, ID_{RSU}^i, O_i, R_i, Cert_{RSU}^i)$. At this point, the total size of this message is calculated as 32 × 6 + 256 × 1 + 160 × 3 + 24 × 1 = 952 + 56 = 1008 bits. In the authentication phase, the server finally generates packet $(TS_4^i, ID_1^j, Cert_{RSU}, \phi_j)$, Hence, the total communication cost for an individual user is (32 × 13) + (256 × 3) + (160 × 2) + (24 × 3) = 1576 bits. Therefore, the total communication cost for SAKE is 2584 bits.

In the AKA-MS scheme [32], the user exchanges the information (IDu, M, TW) with the registration center; hence, the total size is computed as (160 × 4)+ (256 × 1) + (32 × 1) = 928 bits. In the authentication phase, the user communicates with the server and finally exchanges $M_1 = PID_u, DID_u$, which needs (160 + 160) =320 bits. Then, the server sends message $M_2 = Q_{uj}, V_j$, which has the size of (256 + 256) = 512 bits. The user later sends $M_3 = Z_{uj}$ that needs 160 bits. Thus, the total communications cost of AKA-MS is 1920 bits.

In the proposed scheme, there are two interacting messages between the user and the authentication server in the login phase. The user first sends an initialization authentication request $Msg1.Auth_u^{'} = E_{pk}\{u_{id}^{'} \parallel X_i^{'} \parallel Bio_i^{'} \parallel tgs_{id} \parallel TS_1\}$; the size of the message is calculated as $32 + 160 + 32 + 32 + 24 = 248$ bits. The AS sends the second message to the user as $Msg.2 : (u_{id}^{'} \parallel tgs_{id} \parallel E_{u_{pk}}\{ks_{(u \to tgs)} \parallel TS_2 \parallel tgs_{tkt}\}$; the size of the message is computed as $32 + 32 + 128 + 24 + 160 = 376$ bits. Therefore, the total communication cost in the first phase is $(248 + 376) = 624$ bits. In the authentication phase, there are three messages which are shared between the user and the AS. The user sends a message to the TGS $Msg.3 : Auth_{u \to tgs} = E_{ks_{u \to tgs}}\{u_{id}^{'} \parallel fn_{id} \parallel TS_3 \parallel tgs_{tkt}\}$, with the size of length computed as $(32 + 32 + 24 + 160) = 248$ bits. Then, the TGS response to the user is calculated as $Msg.4 : u_{id} \parallel fn_{tkt} \parallel E_{ks_{u \to tgs}}\{tgs_{id} \parallel ks_{u \to fn_s} \parallel ts_4\}$, with size $(32 + 160 + 32 + 128 + 24) = 376$ bits. The user requests access to the fog node $Msg.5 : Auth_{u \to fn_s} = E_{ks_{u \to fn}}\{u_{id} \parallel TS_4 \parallel fn_{id}\}$; the message size is calculated as $(32 + 24 + 32) = 88$ bits. Therefore, the total communication cost of the proposed scheme is (624 + 248 + 376 + 88) = 1336 bits. In our proposed scheme, the used operations are lightweight compared to the

other schemes. Table 5 shows that the proposed scheme has a less communication cost compared to the others.

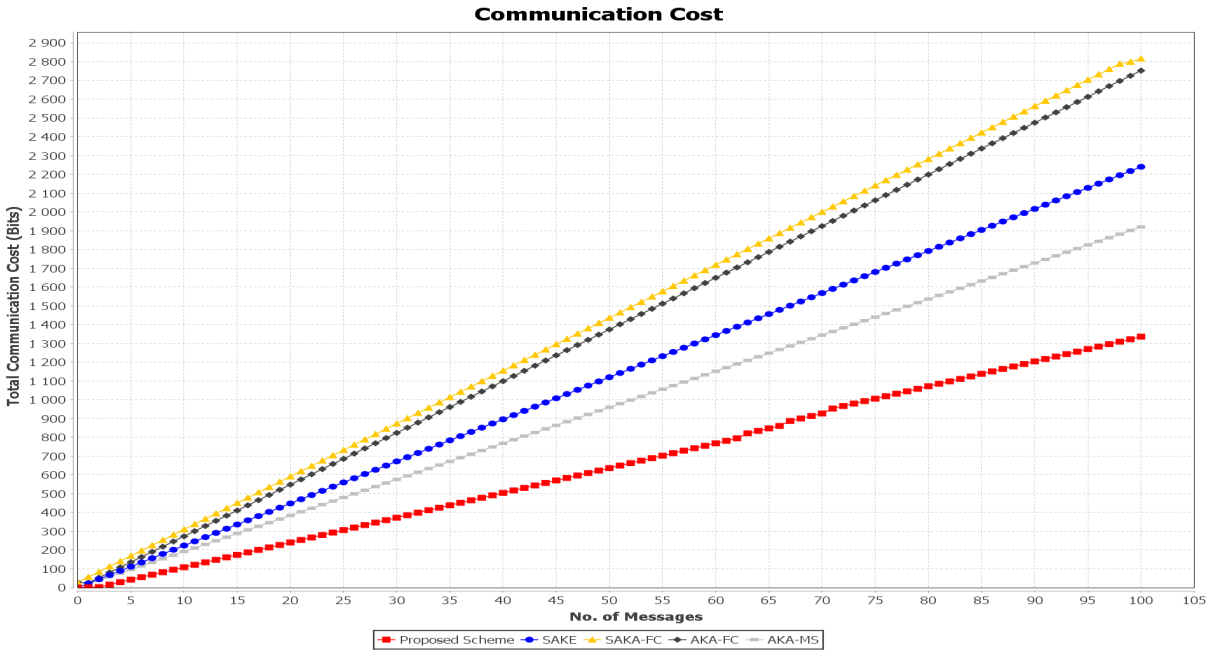


Figure 15. Communication costs comparison of SELAMAT against SAKA-FC, SAKE, AKA-FC, and AKA-MS.

9. Conclusions

This paper proposes a lightweight multi-factor authentication scheme for cross-platform industrial IoT systems, *SELAMAT*. In SELAMAT, we use the AES-ECC algorithm for efficient and secure key management encryption mechanisms in the cloud provider server that acts as a trusted authority. Furthermore, the scheme adopts the Kerberos workflow due to the wide acceptance of the protocol in real-life applications. The designed algorithm offers secure communication between the edge devices and the fog node servers when the messages are transmitted via a public network. The proposed scheme enables edge devices to access any fog server in the fog computing network and improves the efficiency of the system by reducing the computation and communication cost to avoid a network burden.

The results show that the SELAMAT scheme reduces the communication and computation cost compared to the SAKA-FC, SAKE, AKA-FC, and AKA-MS schemes. This extensive comparison of the scheme efficiency shows that the proposed scheme can achieve better performance than the existing scheme. The AVISPA tool is used to verify the security of the scheme. The SELAMAT scheme provides robust security against attacks (replay attack, impersonation attack, man-in-the-middle attack, known-key attack, insider attack, server spoofing attack, etc.), and it was evaluated by using the formal and informal security analyses. In addition, the mutual authentication of the proposed scheme was proven by using the BAN logic. As mentioned above, the advantages pave a path for IIoT usability and suit the IIoT resources-constrained devices. In the future, the proposed scheme can improve the performance and the security of industrial hardware.

Author Contributions: Conceptualization, H.K, and S.J.H.; Methodology, H.K., S.J.H.; Software, H.K.; Validation, H.K.; Results interception, H.K. and S.J.H.; Formal analysis, H.K.; Writing—original draft preparation, H.K.; Writing—review and editing, H.K. and S.J.H.; Supervision, S.J.H., S.M.S.A., F.H., and M.A.C.; and Project administration, S.J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: We would like to thank the reviewers for their careful, constructive and insightful comments in relation to this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. El-hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A survey of internet of things (IoT) Authentication schemes. *Sensors* **2019**, *19*, 1141. [CrossRef]
2. Kwon, S.; Jeong, J.; Shon, T. Toward security enhanced provisioning in industrial IoT systems. *Sensors* **2018**, *18*, 4372. [CrossRef]
3. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
4. Ni, J.; Zhang, K.; Lin, X.; Shen, X.S. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 601–628. [CrossRef]
5. Choudhary, K.; Gaba, G.S.; Butun, I.; Kumar, P. MAKE-IT—A Lightweight Mutual Authentication and Key Exchange Protocol for Industrial Internet of Things. *Sensors* **2020**, *20*, 5166. [CrossRef]
6. Lin, C.; He, D.; Huang, X.; Choo, K.K.R.; Vasilakos, A.V. BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *J. Netw. Comput. Appl.* **2018**, *116*, 42–52. [CrossRef]
7. Lupascu, C.; Lupascu, A.; Bica, I. DLT Based Authentication Framework for Industrial IoT Devices. *Sensors* **2020**, *20*, 2621. [CrossRef] [PubMed]
8. Sari, A.; Lekidis, A.; Butun, I. Industrial Networks and IIoT: Now and Future Trends. In *Industrial IoT*; Springer: Berlin, Germany, 2020; pp. 3–55.
9. Iorga, M.; Feldman, L.; Barton, R.; Martin, M.J.; Goren, N.S.; Mahmoudi, C. *Fog Computing Conceptual Model*; Technical Report; No. Special Publication (NIST SP)-500-325; NIST: Gaithersburg, MD, USA, 2018.
10. Greenberg, A. How 30 Lines of Code Blew Up a 27-Ton Generator. WIRED Security. 2020. Available online: https://www.wired.com/story/how-30-lines-of-code-blew-up-27-ton-generator/ (accessed on 26 December 2020).
11. Evans, B. Firebase: Google Cloud's Evil Twin. SANS Blog, Security Boulevard. 2020. Available online: https://securityboulevard.com/2020/10/firebase-google-clouds-evil-twin-excerpt/ (accessed on 26 December 2020).
12. Wang, L.; An, H.; Chang, Z. Security Enhancement on a Lightweight Authentication Scheme with Anonymity for Fog Computing Architecture. *IEEE Access* **2020**, *8*, 97267–97278. [CrossRef]
13. Cigoj, P.; Blažič, B.J. An authentication and authorization solution for a multiplatform cloud environment. *Inf. Secur. J. Glob. Perspect.* **2015**, *24*, 146–156. [CrossRef]
14. Monteiro, A.C.B.; França, R.P.; Estrela, V.V.; Iano, Y.; Khelassi, A.; Razmjooy, N. Health 4.0 as an Application of Industry 4.0 in Healthcare Services and Management. *Med. Technol. J.* **2018**, *2*, 262–276.
15. Yang, Y.; Hu, M.; Kong, S.; Gong, B.; Liu, X. Scheme on cross-domain identity authentication based on group signature for cloud computing. *Wuhan Univ. J. Nat. Sci.* **2019**, *24*, 134–140. [CrossRef]
16. Wang, W.; Hu, N.; Liu, X. BlockCAM: A blockchain-based cross-domain authentication model. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 896–901.
17. Kaur, H.; Kumar, N.; Batra, S. ClaMPP: A cloud-based multi-party privacy preserving classification scheme for distributed applications. *J. Supercomput.* **2019**, *75*, 3046–3075. [CrossRef]
18. Sengupta, J.; Ruj, S.; Bit, S.D. A Comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. *J. Netw. Comput. Appl.* **2020**, *149*, 102481. [CrossRef]
19. Da Xu, L.; He, W.; Li, S. Internet of things in industries: A survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243.
20. Chen, C.M.; Huang, Y.; Wang, K.H.; Kumari, S.; Wu, M.E. A secure authenticated and key exchange scheme for fog computing. *Enterp. Inf. Syst.* **2020**, 4, 1–16. [CrossRef]
21. Munir, K.; Mohammed, L.A. Biometric smartcard authentication for fog computing. *Int. J. Netw. Secur. Appl. (IJNSA)* **2018**, *10*, 34–42. [CrossRef]
22. Rahman, G.; Wen, C.C. Mutual Authentication Security Scheme in Fog Computing. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 443–451. [CrossRef]
23. Ibrahim, M.H. Octopus: An Edge-fog Mutual Authentication Scheme. *IJ Netw. Secur.* **2016**, *18*, 1089–1101.
24. Zmezm, H.F.; Hashim, S.; Sali, A.; Alezabi, K.A. Pre-authentication design for seamless and secure handover in mobile WiMAX. *Int. Rev. Comput. Softw. (IRECOS)* **2015**, *10*, 764–772. [CrossRef]
25. Alezabi, K.A.; Hashim, F.; Hashim, S.J.; Ali, B.M. An efficient authentication and key agreement protocol for 4G (LTE) networks. In Proceedings of the 2014 IEEE Region 10 Symposium, Kuala Lumpur, Malaysia, 14–16 April 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 502–507.
26. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V. Design of secure key management and user authentication scheme for fog computing services. *Future Gener. Comput. Syst.* **2019**, *91*, 475–492. [CrossRef]

27. Wazid, M.; Das, A.K.; Hussain, R.; Succi, G.; Rodrigues, J.J. Authentication in cloud-driven IoT-based big data environment: Survey and outlook. *J. Syst. Archit.* **2019**, *97*, 185–196. [CrossRef]
28. He, D.; Kumar, N.; Wang, H.; Wang, L.; Choo, K.K.R.; Vinel, A. A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 633–645. [CrossRef]
29. Wazid, M.; Das, A.K.; Lee, J.H. User authentication in a tactile internet based remote surgery environment: Security issues, challenges, and future research directions. *Pervasive Mob. Comput.* **2019**, *54*, 71–85. [CrossRef]
30. Wen, Y.; Zhang, F.; Wang, H.; Gong, Z.; Miao, Y.; Deng, Y. A new secret handshake scheme with multi-symptom intersection for mobile healthcare social networks. *Inf. Sci.* **2020**, *520*, 142–154. [CrossRef]
31. Jia, X.; He, D.; Kumar, N.; Choo, K.K.R. Authenticated key agreement scheme for fog-driven IoT healthcare system. *Wirel. Netw.* **2019**, *25*, 4737–4750. [CrossRef]
32. Akram, M.A.; Ghaffar, Z.; Mahmood, K.; Kumari, S.; Agarwal, K.; Chen, C.M. An anonymous authenticated key-agreement scheme for multi-server infrastructure. *Hum. Centric Comput. Inf. Sci.* **2020**, *10*, 1–18. [CrossRef]
33. Tan, H.; Xuan, S.; Chung, I. HCDA: Efficient Pairing-Free Homographic Key Management for Dynamic Cross-Domain Authentication in VANETs. *Symmetry* **2020**, *12*, 1003. [CrossRef]
34. Venčkauskas, A.; Morkevicius, N.; Jukavičius, V.; Damaševičius, R.; Toldinas, J.; Grigaliūnas, Š. An edge-fog secure self-authenticable data transfer protocol. *Sensors* **2019**, *19*, 3612. [CrossRef]
35. Zhang, H.; Babar, M.; Tariq, M.U.; Jan, M.A.; Menon, V.G.; Li, X. SafeCity: Toward Safe and Secured Data Management Design for IoT-Enabled Smart City Planning. *IEEE Access* **2020**, *8*, 145256–145267. [CrossRef]
36. Katsikas, S.; Gkioulos, V. Security, Privacy, and Trustworthiness of Sensor Networks and Internet of Things. *Sensors* **2020**, *20*, 3846. [CrossRef]
37. Mohamed, N.N.; Yussoff, Y.M.; Saleh, M.A.; Hashim, H. Hybrid Cryptographic Apprach For Internet of Hybrid Applications: A Review. *J. Inf. Commun. Technol.* **2020**, *19*, 279–319.
38. Ganesh, A.R.; Manikandan, P.N.; Sethu, S.P.; Sundararajan, R.; Pargunarajan, K. An improved AES-ECC hybrid encryption scheme for secure communication in cooperative diversity based Wireless Sensor Networks. In Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT), Tamil Nadu, India, 3–5 June 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1209–1214.
39. Viganò, L. Automated security protocol analysis with the AVISPA tool. *Electron. Notes Theor. Comput. Sci.* **2006**, *155*, 61–86. [CrossRef]
40. Chevalier, Y.; Compagna, L.; Cuellar, J.; Drielsma, P.H.; Mantovani, J.; Mödersheim, S.; Vigneron, L. The High Level Protocol Specification Language. Available online: http://avispa-project.org/delivs/2.1/d2-1.pdf (accessed on 26 September 2006).
41. Jia, X.; Hu, N.; Su, S.; Yin, S.; Zhao, Y.; Cheng, X.; Zhang, C. IRBA: An Identity-Based Cross-Domain Authentication Scheme for the Internet of Things. *Electronics* **2020**, *9*, 634. [CrossRef]