Journal of Grid Computing (2023) 21:35
https://doi.org/10.1007/s10723-023-09672-z

Secure Authentication and Reliable Cloud Storage Scheme for IoT-Edge-Cloud Integration

Ajay Chaudhary · Sateesh K Peddoju · Vikas Chouhan

Received: 14 April 2022 / Accepted: 18 May 2023 / Published online: 27 June 2023

Abstract The Internet of Things (IoT) devices are used in almost every aspect of life to automate routine or critical tasks with great precision. The IoT nodes, users, edge nodes, cloud resources, and the connected network are critical components of IoT-Edge-Cloud integration. Any unauthorized access to these resources may halt or bring down the whole IoT infrastructure leading to a severe impact. Hence, authenticating and authorizing these components is essential. Thus, this paper proposes an authentication scheme to securely integrates users, IoT nodes, Edge node, and the cloud infrastructure. We also proposed a reliable cloud data storage and retrieval mechanism using an Erasure Coding strategy in order to store the data generated by IoT infrastructure. We validate the proposed authentication protocols using the well-known and widely used AVISPA simulator tool. The results demonstrate that the proposed authentication protocols are secure against a wide range of security attacks. Further, a comprehensive security analysis was carried out to demonstrate that our protocols are secure against possible attacks and include essential security features. The proposed scheme provides mutual authentication, accessibility, confidentiality, scalability, secure storage, and a secure communication mechanism in the integrated IoT-Edge-Cloud infrastructure with reliable cloud storage.

Keywords Cloud computing · Edge computing · Internet of things · Secure authentication · Reliability · Cloud storage

A. Chaudhary (✉) · S. K. Peddoju · V. Chouhan
Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee, India
e-mail: achaudhary@cs.iitr.ac.in

S. K. Peddoju
e-mail: sateesh@cs.iitr.ac.in

V. Chouhan
e-mail: vchouhan@cs.iitr.ac.in

1 Introduction

The Internet of Things (IoT) devices are getting deployed in every aspect of life to automate day-to-day activities with high precision [10, 12, 56]. With the increasing deployment of IoT devices and their capabilities to handle critical tasks such as self-driven cars, patient monitoring, traffic management, and industrial automation, if any device gets compromised or is accessed by an unauthorized user, it may lead to adverse conditions. For example, the GE factory in the USA deployed 10000 intelligent sensors in their plant. These sensors are connected to ethernet and accessible via Wi-Fi gateway nodes [6]. The IoT nodes and edge node are accessible via the Internet [11], which brings severe security and privacy threats to the IoT infrastructure. As a result, IoT nodes and users must be authenticated and authorized before accessing the services. The authentication of users and devices must be devised carefully and fail-proof protocols must be deployed.

Storing data locally on IoT and edge nodes are not feasible as both are memory-constrained devices. As a result, the edge node forwarded the historical data to the cloud storage servers. Storing data into the cloud infrastructure provides an extra layer of protection by ensuring data consistency, reliability, and availability [16, 27, 56]. Most cloud storage providers have a diverse network with storage facilities at various locations globally. Often the storage system tries to provide reliable data available to the customers. However, due to system (network, hardware, or software) failures, the data are not readily available to the customers [15]. One of the solutions to this problem is using erasure encoding for data storage. In the erasure encoding scheme, the data is fragmented into multiple coded fragments of data and parity to protect any losses that may occur due to hardware or software failures [1]. The notations used throughout the paper are summarised in Table 1.

1.1 Motivation

Several researchers [13, 17–19, 25, 35, 41, 45, 47, 60] have proposed authentication protocols using the one-way hash and XOR function. Some approaches provide

Table 1 Notations

Type	Symbol	Description	Symbol	Description
Entities	U_i	Users	S_i	Cloud Server
	DVC_i	Device	$\mathcal{A}$	Attacker
	SCN_i	Smartcard Number	ITN_i	IoT Node
	SA	System Administrator	GTW/EN	Gateway/Edge Node
Identities	ID_u	User Identity	ID_{DVC}	Device Identity
	ID_{GTW}	Gateway/Edge Node Identity	ID_{ITN}	IoT Node Identity
Security Symbols	K_{U_i}	Public Key of User	K'_{U_i}	Private Key of User
	K_{DVC_i}	Public Key of Device	K'_{DVC_i}	Private Key of Device
	K_{S_i}	Public Key of Cloud Server	K'_{S_i}	Private Key of Cloud Server
	K_{ITN_i}	Public Key of IoT Node	K'_{ITN_i}	Private Key of IoT Node
	K_{GTW}	Public Key of Gateway/Edge Node	K'_{GTW}	Private Key of Gateway/Edge Node
	sk	Session Key	PW_i	Password of User U_i
	$\{\cdot\}_K$	Encrypt using $key\ k$	$[\cdot]_{K'}$	Decrypt using k'
	MSG_i	i^{th} Message	η_i	i^{th} Encrypted Message
	N_i	Nonce i	T_i	Timestamp i
	σ	Signed Message	σ_i	i^{th} Signed Message
	h	Hash Value	h_i	i^{th} Hash Value
	fng_i	User's biometric template	B_i	Biometric of user U_i
Functions	$B(\cdot,\cdot)$	Bio-Hashing Function	$H(\cdot)$	One-Way Hash Function
	$f(\cdot)$	Combine all the values	$f_x(\cdot)$	Extract Values
	$g_{sk}()$	Generate Session Key Function	$g_{ts}()$	Generate Timestamp Function
Operators	$\|$	Concatenation Operation	$\oplus$	XoR Operation
Storage Notations	α	Number of Data Fragments	τ	Token
	β	Number of Parity Fragments	ψ	Length of Single Fragment
	γ	Total Fragments, $\alpha+\beta$	S_{oh}	Storage Overhead

a certain level of security against several attacks but are still vulnerable to many attacks. However, most of these protocols are devised for Wireless Sensor Networks (WSN) devices and user authentication for standalone setup. The authors in work [2, 26, 46] suggested the protocols for heterogeneous wireless sensor networks with gateway nodes, but their protocols are also susceptible to specific attacks. Ostadsharif et al. [6] and Wu et al. [53] proposed authentication and key agreement protocol for IoT-based WSNs. Besides, Amin et al. [4] proposed the protocol for cloud, fog, and user communication. Recently, Zargar et al. [61] proposed an authentication protocol between the cloud server and user using a trusted third party, and Shahidinejad et al. [39] proposed an authentication protocol for IoT nodes, trust center, and cloud servers integration. Most of these approaches provide either the user or IoT node authentication and authorization to access required services. In the literature, most of the works in this field are initially focused on WSN and user interaction. Some recent works aim to integrate cloud-user, cloud-IoT device, cloud-fog-user, and the user-IoT device with or without a trusted third party. To the best of our knowledge, none of the existing schemes explore providing mutual authentication and authorization for integrated IoT-Edge-Cloud infrastructure. Since both IoT and Edge nodes have limited memory, there is a need for secure and reliable data storage/retrieval in the cloud, and the same is not addressed by the aforementioned schemes. Therefore, we also integrate a secure and reliable cloud data storage and retrieval mechanism. Our scheme can provide data reliability even if a few data fragments are missing or corrupted.

1.2 Contributions

The major contributions of this work are summarized below:

- We propose a secure authentication scheme for IoT-Edge-Cloud integration with reliable cloud storage.
- The proposed protocol's security is validated using the Automated Validation of Internet Security Protocols and Applications (AVISPA) simulation tool. Additionally, we validated through security analysis that the proposed protocols are secure against a wide range of possible attacks and provide essential security features.
- To provide reliable data storage in the cloud, we innovatively devised an Erasure Coding (EC) scheme to meet the Quality-of-Service (QoS) requirements such as storage efficiency, availability, and recoverability.

1.3 Organization

The rest of the paper is organized as follows. Section 2 summarizes the related work for secure IoT and cloud integration. The proposed work and methodology are presented in Section 3. We discuss the experimental setup in Section 4, and the results and analysis are presented in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

This section reviews the related works for secure IoT-Edge-Cloud integration. We classify them into these categories: (i) User/IoT - Edge integration mechanisms, (ii) User/Edge - Cloud integration mechanisms, and (iii) Reliable cloud storage mechanisms.

2.1 User/IoT - Edge Integration Mechanisms

The IoT nodes use the Internet protocol for communication resulting in the threats and vulnerabilities of the Internet now being part of the IoT network. To resolve the security issues of IoT nodes, researchers have focused on providing real deployable solutions for every possible aspect of the security threats. Das [18] proposed a user and node authentication protocol using the Hash and XOR functions where the gateway node handles the compute-intensive authentication task. However, later on, several researchers proved that Das's [18] protocol is vulnerable to several attacks. Huang et al. [25] identified that the protocol is vulnerable to masquerade attacks and cannot achieve user anonymity. The protocol is also susceptible to insider attacks and impersonation attacks [22] as the gateway cannot know the real identity of any user during the

authentication process and the protocol is also suffering from offline password guessing attacks by an insider [35]. Many researchers have proposed an improved version of Das's protocol [18]. Nyang et al. [35] proposed enhanced user authentication using Hash and XOR functions. In their model, they provided patches to Das's protocol [18] to provide security against password-guessing attacks by the insider. This protocol is also prone to the login-id and stolen-verifier attacks [17]. Turkanovic et al. [46] proposed an efficient and novel user authentication and key agreement protocol for heterogeneous Wireless Sensor Networks (WSN). Their protocol provides mutual authentication between the user and sensor password-guessing node for the resource-constrained architecture of the WSN. Amin et al. [2] demonstrated that the protocol proposed by Turkanovic et al. [46] is susceptible to offline identity-password guessing, smartcard theft, user and sensor node impersonation attacks and has inefficient authentication phases. Farash et al. [19] improved three-party password-based authenticated key exchange protocol using extended chaotic maps. Besides, Amin et al. [3] showed that the Farash et al. [19] protocol is susceptible to an offline password guessing attack, lost or stolen smartcard attack, user impersonation attack, session-specific information attack, and new smartcard issue attack. The Farash et al. [19] protocol is also unable to protect user anonymity and the secret key of the gateway node. Amin et al. [3] designed an efficient and robust smartcard-based user authentication and session key protocol for WSN/IoT using Bio-hashing, hash function, and Bitwise XOR operations. Later, Jiang et al. [26] found that the Amin et al. [3] protocol is vulnerable to smartcard loss attacks, known session-specific temporary information attacks, tracking attacks, and fails to fulfill user untraceability. Also, Amin et al. [3] protocol is susceptible to strong replay attacks or unable to provide perfect forward secrecy [6]. Besides Ostadsharif et al. [6] proposed a three-factor authentication and key agreement protocol for IoT-based WSNs. In their protocol, they try to mitigate the security issues of Amin et al. [3] and Jiang et al. [26] approaches based on one-way hash and bio hashing. Subsequently, Chen et al. [14] proposed a three-factor user authentication scheme for wireless medical sensor networks to overcome flaws of smartcard, password, and biometric authentication features. However, Wu et al. [53] found that the Ostadsharif et al. [6] proposed protocol is vulnerable to offline surmising attacks. Similarly, Chen et al. [14] scheme is vulnerable to offline password-guessing attacks. Thus, Wu et al. [53] proposed a fresh authentication scheme for WSNs that employs a three-factor mechanism based on Bio-hashing, One-way hash function, and Bitwise XOR operations.

2.2 User/Edge - Cloud Integration Mechanisms

The main goal of user/edge node and cloud integration is to provide secure cloud access for users and edge node. Li et al. [32] proposed an authentication scheme for multi-servers using the neural network-based technique, but it had high complexity. Xue et al. [58] have proposed a dynamic pseudonym identity-based authentication and key agreement protocol for the multi-server environment. Stergiou et al. [42] surveyed IoT and cloud computing with a focus on the security issues of both technologies. They tested well-known security encryption algorithms for IoT and cloud integration. Besides, Amin et al. [4] proposed an authentication scheme for the integration of IoT nodes and distributed cloud system. They used the private cloud to store confidential information using IoT techniques. Later in 2020, Amin et al. [5] proposed a hierarchical three-tier scheme consisting of the cloud server at the first tier, fog nodes at the second tier, and the users at the last tier. They proposed a protocol for secure communication between all three parties by establishing key management amongst them. In 2021, Shahidinejad et al. [39] proposed an authentication protocol for IoT nodes, trust centers, and cloud server integration. They register the devices with the trust center and assign them a unique identifier. However, the user required authentication and authorization to access the IoT or edge node. Further, Zargar et al. [61] proposed an authentication protocol between the cloud server and users using a trusted third party. The users are registered with a trusted third party to communicate directly with the cloud server in their scheme. However, the user may access the cloud service in this scheme after verifying itself. The cloud servers should also register with a trusted third party. Sometimes the user may want to access IoT nodes directly for some information; in such cases, the IoT node must be accessible to the user. If every piece of information is routed through the cloud, it introduces an additional communication cost to the

system. Also, the user may face some latency in getting the desired information.

2.3 Reliable Cloud Storage Mechanisms

One of the primary goals of cloud storage approaches is to make data readily available to the end user irrespective of hardware, software, or network failure. One of the easiest ways to maximize user data availability is by providing replication redundancy. However, these techniques introduce storage and bandwidth overhead, as large amounts of data are stored as redundant replicated data. In contrast, an erasure code provides redundancy without replication overhead for the systems [49]. The research works on the erasure techniques is mainly focused on fault tolerance and reliability, Schnjakin et al. [38] and Chouhan et al. [15] in their papers presented an in-depth study of various erasure coding techniques such as Reed-Solomon codes [37] and Cauchy-Reed-Solomon [9].

Many researchers are focused on achieving the fault tolerance and reliability of the system, Woitaszek et al. [50] deployed massive arrays to achieve fault tolerance in the storage system. While Wylie et al. [57] and Greenan et al. [21] used XOR-based erasure codes for their models. Later, Li et al. [31] suggested a technique by introducing the Beehive codes to reduce the network and disk I/O transfers significantly. While Rashmi et al. [36] deployed a piggyback code to reduce the amount of data reads. On the other hand, Huang et al. [23] suggested a model that reduced network overhead and latency. The study and comparison of erasure techniques are discussed in depth by Chouhan et al. [15]. The existing literature has not explored IoT-Edge-Cloud integration with reliability. Hence, we integrated reliable storage using the Erasure Coding mechanism in the proposed work.

2.4 Summary of the Related Works

We reviewed the most relevant literature for authenticating and authorizing the users, IoT nodes, edge nodes, and cloud. Also, some of the most recent works are summarized in Table 2. Most of the previous research on IoT and user authentications focuses on the two-party or three-party based mutual authentication scheme for authenticating the users, IoT nodes, and edge node or gateway. The hash function, bio hash function, and XoR operations are used for authentication and authorization purpose. The authentication-related information is stored safely in the smartcard for the users and in the device memory. As IoT nodes generate massive amounts of data, storing them in an edge node or gateway is not a viable solution due to the storage memory constraint of the edge node. The researchers are focused on the authentication of the users or IoT nodes to access server storage or cloud storage. However, the data stored by the devices in the cloud/server are prone to hardware, software, or network failures. One of the easiest ways to maximize user data availability is by providing redundancy using replication techniques, leading to storage and bandwidth overhead. In contrast, an erasure code provides redundancy without replication overhead. The user is interested in directly accessing IoT nodes or historical data stored in the cloud. The user authentication mechanism provides access to the required IoT nodes' historical cloud storage data to the user. We securely merged these three techniques in our work, i.e., User/IoT-Edge integration mechanisms, User/Edge-Cloud integration mechanisms, and reliable cloud storage mechanisms to provide a user-friendly IoT-Edge-Cloud integration.

3 Proposed Work

In this section, we discuss the proposed scheme for IoT-Edge-Cloud integration along with its use cases. The proposed deployment framework is shown in Fig. 1. The System Administrator (SA), Edge Node (EN), IoT nodes, users, and cloud infrastructure are the main stakeholders of our proposed model and are explained below:

- ***System administrator***: The system administrator is an authorized user having complete control over the infrastructure. The SA registers and authorizes users, IoT nodes, and the cloud to access the deployed infrastructure by verifying each entity. Once registered, the SA provides them (users, IoT nodes, and cloud) means to authenticate themselves whenever they need to access the infrastructural resources and securely communicate between themselves. The system administrator may grant and revoke access to any stakeholder.

Table 2 Summary of the recent works

Author	Year	Protocol/Scheme	Weaknesses
Ostadsharif et al. [6]	2019	Proposed a three-factor authentication along with key agreement protocol for IoT-based WSN. In their protocol, they try to mitigate the security issues of Amin et al. [3] and Jiang et al. [26] approach based on one-way hash and bio hashing.	Wu et al. [53] found that the proposed protocol is vulnerable to offline surmising attacks.
Amin et al. [5]	2020	Proposed a hierarchical three-tier scheme consisting of the cloud server at the first tier, fog nodes at the second tier, and the users at the last tier.	There is no provision for exchanging data from IoT Nodes. A separate mechanism is required to access data from IoT nodes, which adds extra overhead.
Wu et al. [53]	2020	Proposed an authentication scheme for WSNs using a three-factor mechanism, Bio-hashing and One-way hash functions, and Bitwise XOR operations.	Wu et al. [55] observed that the protocol does not provide anonymity, is vulnerable to key compromise impersonation and session-specific temporary information attacks, and also violates perfect forward security.
Chouhan et al. [15]	2020	Compares the encoding parameters of the erasure technique and the proposed a mechanism for storing data on the cloud platform using Erasure Coding.	No provision is made for the integration of IoT nodes and users.
Wu et al. [55]	2021	Proposed a secure three-factor authentication protocol based on the user's password, smartcard, Bio-hashing, One-way hash functions, Fuzzy generator, and reproduction function.	Jiaqing et al. [34] revealed that the proposed scheme is vulnerable to information leakage attacks and fails to provide three-factor security and lacks anonymity.
Kumar [29]	2021	Proposed a secure three-factor authentication protocol based on the smartcard, session key, and fuzzy verifier technique.	Jiaqing et al. [34] found that the proposed protocol is susceptible to the information leakage attack as session key calculation involves dependent random numbers.
Wu et al. [55]	2021	Proposed authentication along with key exchange protocol for cloud-based smart healthcare environment using wireless body area network using Fuzzy generator/reproduction function, one-way hash function, and session key.	The patients' information is stored on the cloud healthcare center, but no integration of the edge node, which might improve overall latency compared to accessing data directly from the cloud.
Zargar et al. [61]	2021	Proposed an authentication protocol between the cloud server and users using a trusted third party. In their scheme, the users are registered with a trusted third party to directly communicate with the cloud server.	In some cases, the user may want to access IoT nodes directly; in such situations, the IoT node must be directly accessible to the user. If every piece of information is routed through the cloud, introduces an additional communication cost to the system. Also, the user may face some latency in getting the desired information.
Shahidinejad et al. [39]	2021	Proposed authentication protocol for IoT nodes, trust center, and cloud servers integration. In their scheme, they register the devices with the trust center and assign them a unique identifier.	The scheme proposed an authentication mechanism for IoT nodes for cloud access, but as IoT, edge and cloud are accessible by the users. Thus, the authentication and authorization of the users to access the IoT node or edge node are also required.

Table 2 continued

Shukla et al. [40]	2022	The authors proposed ECC(Elliptic Curve Cryptography) based multi-factor authentication protocol for a cloud environment. It provides authentication between the trusted third party and the user.	The proposed authentication scheme required a trusted third party, therefore, it cannot be used in situations where neither party can be trusted. Also, the protocol is susceptible to privileged insider and stolen verifier attacks [30].
Wu et al. [54]	2022	The authors proposed an Authentication and Key Agreement (AKA) protocol for IoT and Cloud environments. The User, IoT device, Cloud service provider and control server participate in the authentication process, where the User, Cloud service provider and control server are key entities of the proposed scheme.	The proposed model is able to provide authentication between the user and the cloud server using a trusted control, server, but as IoT node, edge node and cloud are accessible by the users. Thus, the authentication and authorization of the users to access the IoT node or edge node are also required.

– ***Edge node***: It acts as a gateway for the IoT nodes and cloud infrastructure. The access to the edge node is highly secure, and only the system administrator may directly access it. The system administrator registers the IoT nodes, users, and cloud infrastructure and gives limited access to the edge node through an authorization mechanism.

– ***IoT nodes***: The IoT nodes are smart actuators or sensing devices that can sense or act on environmental conditions. They can communicate with the edge node or authorized users. If any unauthorized user can access the IoT node, it may cause severe damage to the infrastructure. Before deployment, the system administrator registers the IoT

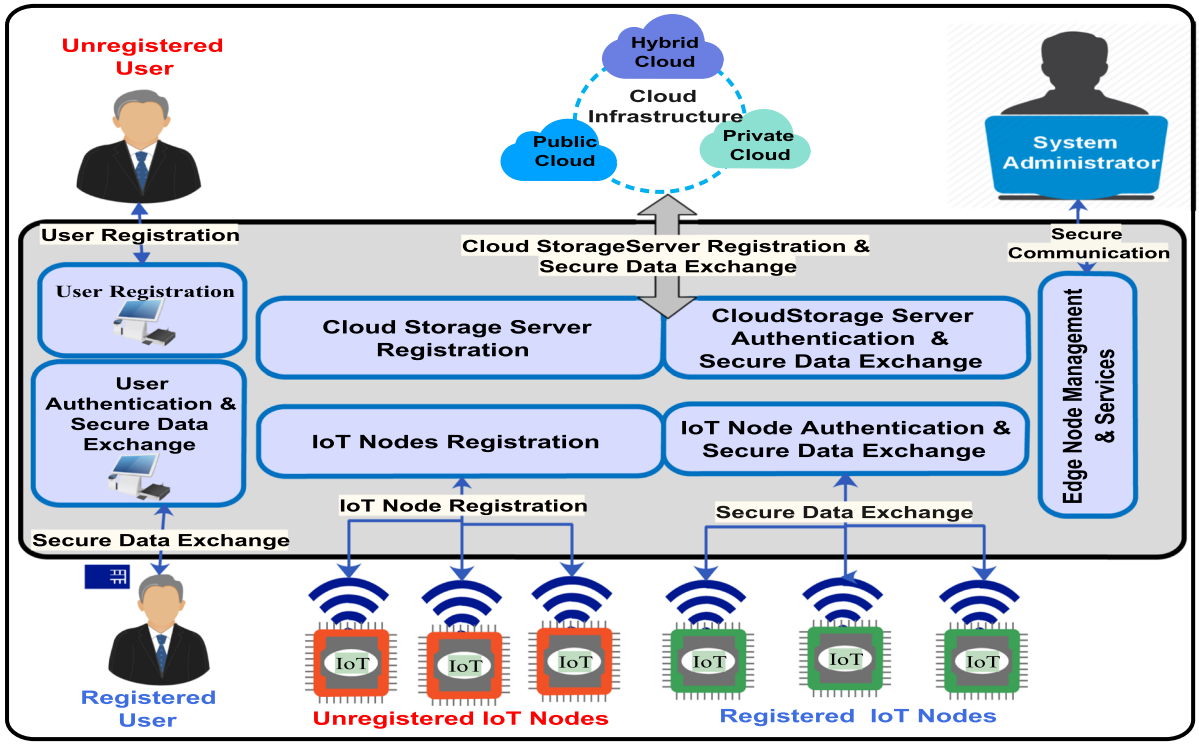


Fig. 1 Proposed deployment framework

nodes with the edge node. The SA stores required credentials and parameters in the tamper-proof memory of the IoT nodes. The IoT nodes use these credentials to authenticate themselves while communicating with the edge node or the authorized users. The node can verify the authenticity of the user using the user's credentials provided by the edge node through a secure message.

– ***Users***: As IoT nodes are competent enough to fulfill users' queries, authorized users may directly access individual IoT nodes. The user must register with the SA before using infrastructure resources. The SA issues a smartcard to the user containing the required information for login into the terminal to access the information from the target node. Registered users can query the IoT node for real-time data or the edge node for historical data. To protect against the lost smartcard attack or user password guessing attack, additional security is enabled using biometrics and password.
– ***Cloud infrastructure***: The SA registers with the cloud infrastructure used to store the historical data generated by the IoT nodes. Once the SA is registered with the cloud service provider, both the cloud service provider and edge node store the required API and credentials.

The deployment infrastructure consists of system setup, authentications and key agreement, and cloud storage integration. The system setup is discussed in Section 3.1, which includes identifying and registering the users, cloud, and IoT nodes with the edge node. The authentications and the key agreement include secure message exchange between registered users, IoT nodes, and the edge node; for the same, we propose authentication protocol along with key agreement as discussed in Section 3.2. Finally, in Section 3.3, we discuss the edge and cloud storage integration for secure data storage in the cloud infrastructure.

3.1 System Setup

The system setup employs a secure channel to safely register users, IoT nodes, and the cloud, as shown in Fig. 2.

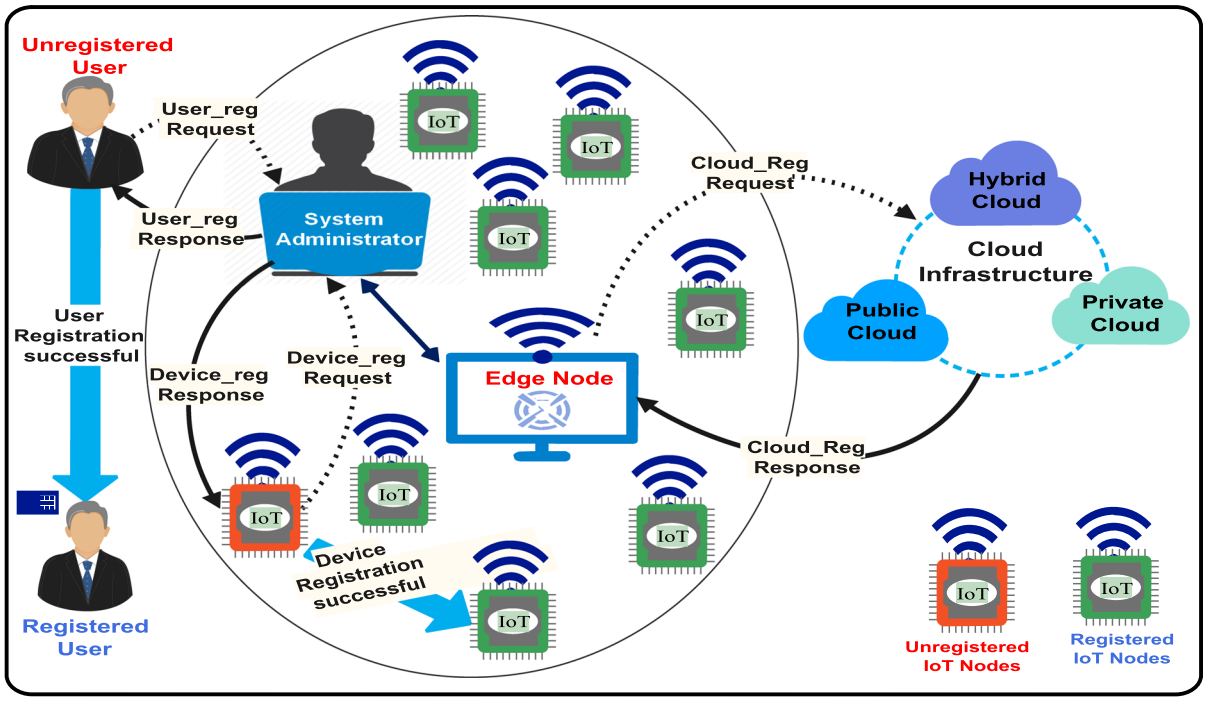


Fig. 2 User, IoT node and cloud registration process

3.1.1 IoT Node Registration

For each IoT node, the SA assigns a unique node identity ITN_i where $1 < i < n$. If any IoT node wants to store data to the cloud, the IoT must go through the edge node using Public-Key Authentication Protocol via Gateway (PKAPG) as explained in Section 3.2.3. Periodically the IoT nodes push data to the edge node. Sometimes edge node pulls data from IoT nodes if enquired by the user. For this purpose, both parties, i.e., IoT node and edge node, may use the Direct Public-Key Authentication Protocol (DPKAP) as explained in Section 3.2.2. To support the execution of both PKAPG and DPKAP authentication protocols, the SA computes private key $K^{'}_{ITN_i}$ of the IoT node and stores the tuple $< K^{'}_{ITN_i}, K_{GTW}, ITN_i, g_{ts}(), H(\cdot) >$ into tamper-proof memory of the IoT node. SA also creates a secure log entry of the IoT node ITN_i with the tuple $< ITN_i, K_{ITN_i} >$ into tamper-proof memory of the edge node.

3.1.2 User Registration

The user conveys the required information to the SA using the secure channel. After successfully verifying the user and receiving information, the SA issues a tamper-proof smartcard and delivers it to the user via the same channel. Once the registration process is completed, whenever a registered user wants to access or retrieve the data from the cloud, a user must go through the PKAPG authentication protocol as explained in Section 3.2.3 to access real-time data or monitor the real-time performance of the infrastructure via the edge node or gateway. On the other hand, a user must go through the DPKAP authentication protocol as explained in Section 3.2.2. In order to access the required information, the user has to login into the terminal using its smartcard and biometric verification. The user registration process completes in the following steps:

(1) User U_i selects a random user identity ID_u, a random nonce r, and password PW_i.
(2) The User U_i computes $TPW_i = H(PW_i \parallel r)$ and conveys $< ID_u, TPW_i, r, K_{u_i} >$ to SA through a secure channel.
(3) SA verifies whether ID_u is already allotted or not, if already allotted then SA requests the user to select new ID_u. If ID_u does not exist in the database, SA allocates the ID_u to the user. SA chooses a random tamper-proof smartcard SCN_i and selects a random nonce N_i. Then, SA computes login instance $L_i = H(SCN_i \oplus N_i \oplus TPW_i)$.
(4) SA delivers the smartcard SCN_i to the user U_i through secure channel after storing $< ID_u, r, L_i, TPW_i, SCN_i, N_i, H(), B(\cdot,\cdot) >$ in it's the tamper-proof memory. Besides, SA creates a secure log entry of the user U_i with parameter $< U_i, ID_u, SCN_i, K^{'}_{GTW}, K_{u_i} >$ into tamper-proof memory of the edge node.
(5) The user U_i inserts the smartcard into smartcard reader. Then user enters the $< ID_u, PW_i >$, smartcard number $SCN^{'}_i$ and scans finger fng_i. The smartcard computes $TPW^{'}_i = H(PW_i \parallel r)$ and $L^{'}_i = H(SCN^{'}_i \oplus N_i \oplus TPW^{'}_i)$ to verify $TPW^{'}_i = TPW_i \ \& \ L^{'}_i = L_i$. The registration process is aborted if the verification fails.
(6) Afterthat, smartcard selects a random number r_i and computes bio authentication instance $B_i = B(r_i, fng_i)$, password authentication instance $auth_i = H(H(B_i \parallel PW_i) \oplus ID_u)$, and $key = (K^{'}_{u_i} \oplus N_i \oplus PW_i \oplus B_i)$ and then deletes $< TPW_i, r >$. Finally, smartcard stores $< ID_u, r_i, auth_i, SCN_i, N_i, H(), B(\cdot,\cdot), K_{GTW}, key >$ into its tamper-proof memory.

3.1.3 Cloud Registration

The SA assigns a unique storage server id (S_i where $0 < i < n$) to each cloud storage server. The EN uses the access token τ to access the secure cloud services. However, the access tokens are string keys issued via the developer console to access the cloud API. Having a token τ enables access to a cloud account A_i, i.e., access to the account's information and files. Multiple Dropbox cloud storage accounts are used in our work; each Dropbox App with storage access has a set of tokens to access the stored data. The API uses HTTP POST requests with JSON arguments and JSON responses. The authorization request header or the authorization URL parameters authenticate requests using OAuth 2.0.

3.2 Authentication Protocols

The User and IoT nodes communicate with each other through the edge node. A user may access required information or data from the IoT node after authenticating itself. The IoT nodes send data to the edge node, where the data is preliminarily processed and utilized. Later on, the sensed data is uploaded to the cloud storage via Edge-Cloud Integration. The Edge node and its cloud integration is discussed in Section 3.3. This section presents a mutual authentication scheme based on public-key cryptography and digital signatures. Public key cryptography protects the message's confidentiality by preventing unwanted access; however, digital signatures confirm that the message is sent by a legitimate user and has not been altered. In our proposed authentication protocol, we use the timestamp concept to keep data fresh and nonce to avoid various forms of response assaults. Our authentication systems use public-key cryptography and digital signatures to verify the identity of participants and exchange the session key among the participants. We use the *AVISPA* simulation to verify our protocols formally and discover that they are secure against various attacks. Further, the proposed scheme is deployed in the following phases:

3.2.1 User Login

Whenever the user U_i wants to access the services, the user U_i has to log in on the terminal using the following steps:

(1) The user U_i inserts the smartcard into the card reader and enters identity $ID_u^{'}$, password $PW_i^{'}$, and scans the registered finger $fng_i^{'}$. The card computes $B_i^{'} = B(r_i, fng_i^{'})$ and $auth_i^{'} = H(H(B_i^{'} \parallel PW_i^{'}) \oplus ID_u^{'})$ and then verifies the computed $auth_i^{'}$ and stored $auth_i$ authentication instances; the login request of U_i is approved if authentication instances are verified.

(2) Once the login is approved, the smartcard fetches the private key $K_{u_i}^{'} = (key \oplus N_i \oplus PW_i^{'} \oplus B_i^{'})$. The tuple $< ID_u, K_{u_i}^{'}, K_{GTW} >$ is now used in authentication phase.

3.2.2 DPKAP Authentication Phase

Direct Public-Key Authentication Protocol (DPKAP) for two-party mutual authentication is described in this phase, which allows a device (user or IoT node) to

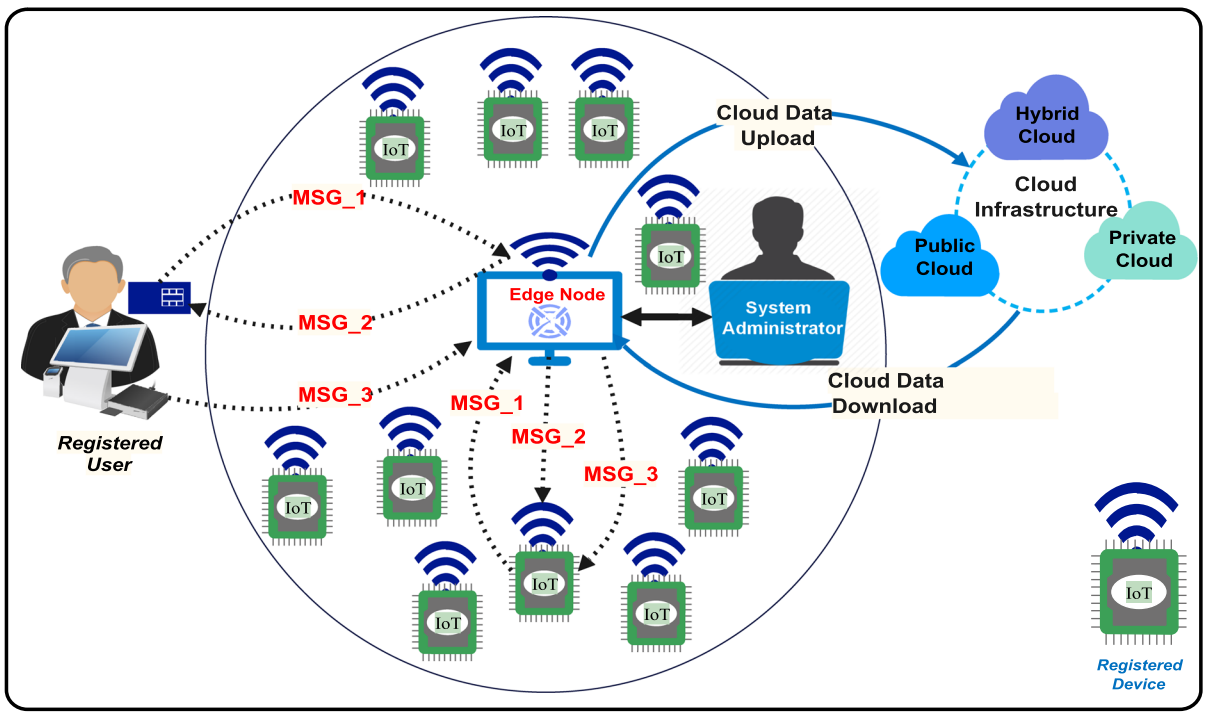


Fig. 3 DPKAP Authentication Protocol – message flow diagram

establish mutual authentication and share session keys for communication. In DPKAP Authentication protocol, for user, we consider $DVC_i = U_i$, $ID_{DVC} = ID_u$, $K_{DVC_i} = K_{U_i}$, and $K'_{DVC_i} = K'_{U_i}$. Similarly, for IoT node, we consider $DVC_i = ITN_i$, $ID_{DVC} = ID_{ITN}$, $K_{DVC_i} = K_{ITN_i}$, and $K'_{DVC_i} = K'_{ITN_i}$. The DPKAP protocol flow is shown in Fig. 3 and the detailed protocol to achieve mutual authentication and session key agreement between the device (user or IoT node) and the edge node EN is shown in Fig. 4. The algorithm performs the following operations:

(1) The device (user or IoT node) first fetches the identity $ID_{DVC} = ID_u/ID_{ITN}$, key pair $K^{'}_{DVC_i} = K^{'}_{u_i}/K^{'}_{ITN_i}$ and K_{GTW} from user smartcard or IoT node's memory for communication. The timestamp T_1 and the session key sk are then generated by invoking the functions $g_{ts}()$ and $g_{sk}()$ accordingly.
(2) Instantly, it computes the message signature σ by encrypting $H(T_1 \oplus sk)$ using the device's private key K'_{DVC}.
(3) The message η is created by calling the function $f(T_1 || sk)$. Afterward, the message MSG_1 is constructed, encrypted with the edge node's public key K_{GTW}, and forwarded to the EN.
(4) EN gets MSG_1 and decrypts it with the EN's private key K'_{GTW}. Then the values of T_1 and sk are retrieved by calling the function $f_x(\eta)$.
(5) EN checks the validity of the received timestamp T_1. If T_1 is invalid, the session is terminated. Otherwise, $h_1 \leftarrow [\sigma]_{K_{DVC}}$ and $h_2 \leftarrow H(T_1 \oplus sk)$ are computed.
(6) The function $Verify(h_1, h_2)$ is then used to verify the equality of both hash values. If the validation test fails, the session will be terminated.
(7) The timestamp T_2 is then generated, and $\sigma \leftarrow \{H(T_1 \oplus T_2 \oplus sk)\}_{K'_{GTW}}$ and $\eta \leftarrow f(T_1 || T_2 || sk)$

$\mathbf{DVC_i}$		$\mathbf{EN}$
Fetch ID_{DVC}, K_{GTW} and k'_{DVC}		
Generate $T_1 \leftarrow g_{ts}()$, $sk \leftarrow g_{sk}()$		
Compute $\sigma \leftarrow \{H(T_1 \oplus sk)\}_{k'_{DVC}}$		
Compute $\eta \leftarrow f(T_1 \parallel sk)$		
	$\xrightarrow{\mathbf{MSG_1}=\{\mathbf{ID_{DVC}},\mathbf{ID_{GTW}},\eta,\sigma\}_{\mathbf{K_{GTW}}}}$	
		$ID_{DVC}, ID_{GTW}, \sigma, \eta \leftarrow [MSG_1]_{K'_{GTW}}$
		Compute $T_1, sk \leftarrow f_x(\eta)$
		$if\ T_1\ is\ invalid,\ abort\ the\ session$
		Compute $h_1 \leftarrow [\sigma]_{K_{DVC}}$
		Compute $h_2 \leftarrow H(T_1 \oplus sk)$
		$Verify(h_1, h_2)\ if\ fail,\ abort\ the\ session$
		Generate $T_2 \leftarrow g_{ts}()$
		Compute $\sigma \leftarrow \{H(T_1 \oplus T_2 \oplus sk)\}_{K'_{GTW}}$
		Compute $\eta \leftarrow f(T_1 \parallel T_2 \parallel sk)$
	$\xleftarrow{\mathbf{MSG_2}=\{\mathbf{ID_{GTW}},\mathbf{ID_{DVC}},\eta,\sigma\}_{\mathbf{K_{DVC}}}}$	
$ID_{GTW}, ID_{DVC}, \eta, \sigma \leftarrow [MSG_2]_{k'_{DVC}}$		
Compute $T_1, T_2, sk \leftarrow f_x(\eta)$		
$if\ T_1, T_2, sk\ are\ invalid,\ abort\ the\ session$		
Compute $h_1 \leftarrow [\sigma]_{K_{GTW}}$		
Compute $h_2 \leftarrow H(T_1 \oplus T_2 \oplus sk)$		
$Verify(h_1, h_2)\ if\ fail,\ abort\ the\ session$		
Compute $\sigma \leftarrow \{H(T_2 \oplus sk)\}_{k'_{DVC}}$		
Compute $\eta \leftarrow f(T_2 \parallel \sigma)$		
	$\xrightarrow{\mathbf{MSG_3}=\{\mathbf{ID_{DVC}},\mathbf{ID_{GTW}},\eta\}_{\mathbf{K_{GTW}}}}$	
		$ID_{DVC}, ID_{GTW}, \eta \leftarrow [MSG_3]_{K'_{GTW}}$
		Compute $T_2, \sigma \leftarrow f_x(\eta)$
		$if\ T_2\ is\ invalid,\ abort\ the\ session$
		Compute $h_1 \leftarrow [\sigma]_{K_{DVC}}$
		Compute $h_2 \leftarrow H(T_2 \oplus sk)$
		$Verify(h_1, h_2)\ if\ fail,\ abort\ the\ session$

Fig. 4 DPKAP Authentication Protocol: direct public-key authentication protocol for two-party mutual authentication

are computed. Thereafter, the message MSG_2 is constructed and then encrypted with the device's public key K_{DVC} before being routed back to the DVC_i device.

(8) DVC_i decrypts the received message MSG_2 with k'_{DVC}. It also extracts T_1,T_2,sk by calling the function $f_x(\eta)$. It now verifies the validity of both T_1 and T_2 timestamps. If any of the timestamps fails the validation test, the session will be terminated.

(9) Otherwise, $h_1 \leftarrow [\sigma]_{K_{GTW}}$ and $h_2 \leftarrow H(T_1 \oplus T_2 \oplus sk)$ are computed. The function $Verify(h_1, h_2)$ is then used to verify the equality of both hash values. If the validation test fails, the session will be terminated. Then $\sigma \leftarrow \{H(T_2 \oplus sk)\}_{k'_{DVC}}$ and $\eta \leftarrow f(T_2 \,||\, \sigma)$ are computed.

(10) The message is then constructed as MSG_3 and encrypted with the EN public key K_{GTW} before being forwarded back to the EN.

(11) EN gets MSG_3 and decrypts it with EN's private key K'_{GTW}. Then the values of T_2 and σ are retrieved by calling the function $f_x(\eta)$.

(12) EN checks the validity of the received timestamp T_2. If T_2 is invalid, the session is terminated. Otherwise, h_1 and h_2 are computed.

(13) The function $Verify(h_1, h_2)$ is then used to verify the equality of both hash values. If the validation test fails, the session will be terminated.

3.2.3 PKAPG Authentication Phase

In this section, we discuss the public-key authentication protocol for mutual authentication via an edge node. In PKAPG Authentication protocol, for user, we consider $DVC_i = U_i$, $ID_{DVC} = ID_u$, $K_{DVC_i} = K_{U_i}$, and $K'_{DVC_i} = K'_{U_i}$. Similarly, for IoT node, we consider $DVC_i = ITN_i$, $ID_{DVC} = ID_{ITN}$, $K_{DVC_i} = K_{ITN_i}$, and $K'_{DVC_i} = K'_{ITN_i}$. The PKAPG protocol flow is shown in Fig. 5 and the detailed PKAPG protocol for the mutual authentication between the device (user or IoT node), the edge node EN, and the cloud server S_i is accomplished using the procedures indicated in Fig. 6. The algorithm performs the following operations:

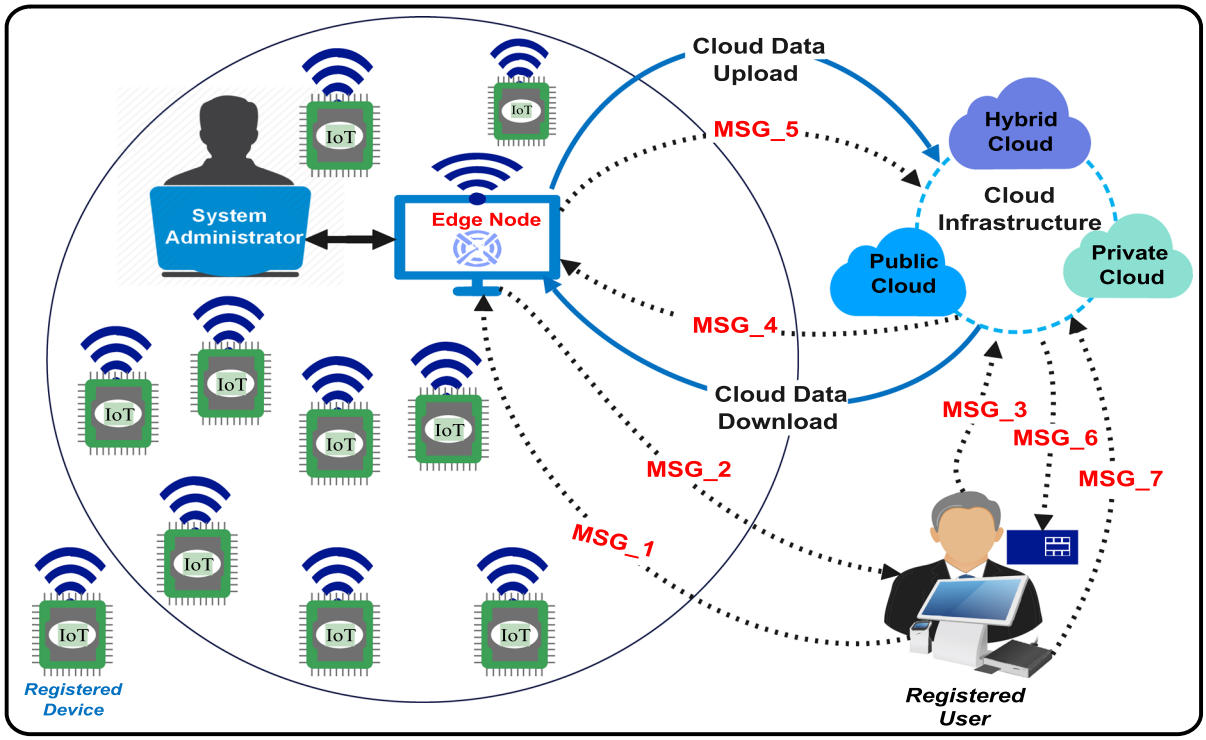


Fig. 5 PKAPG Authentication Protocol – message flow diagram.

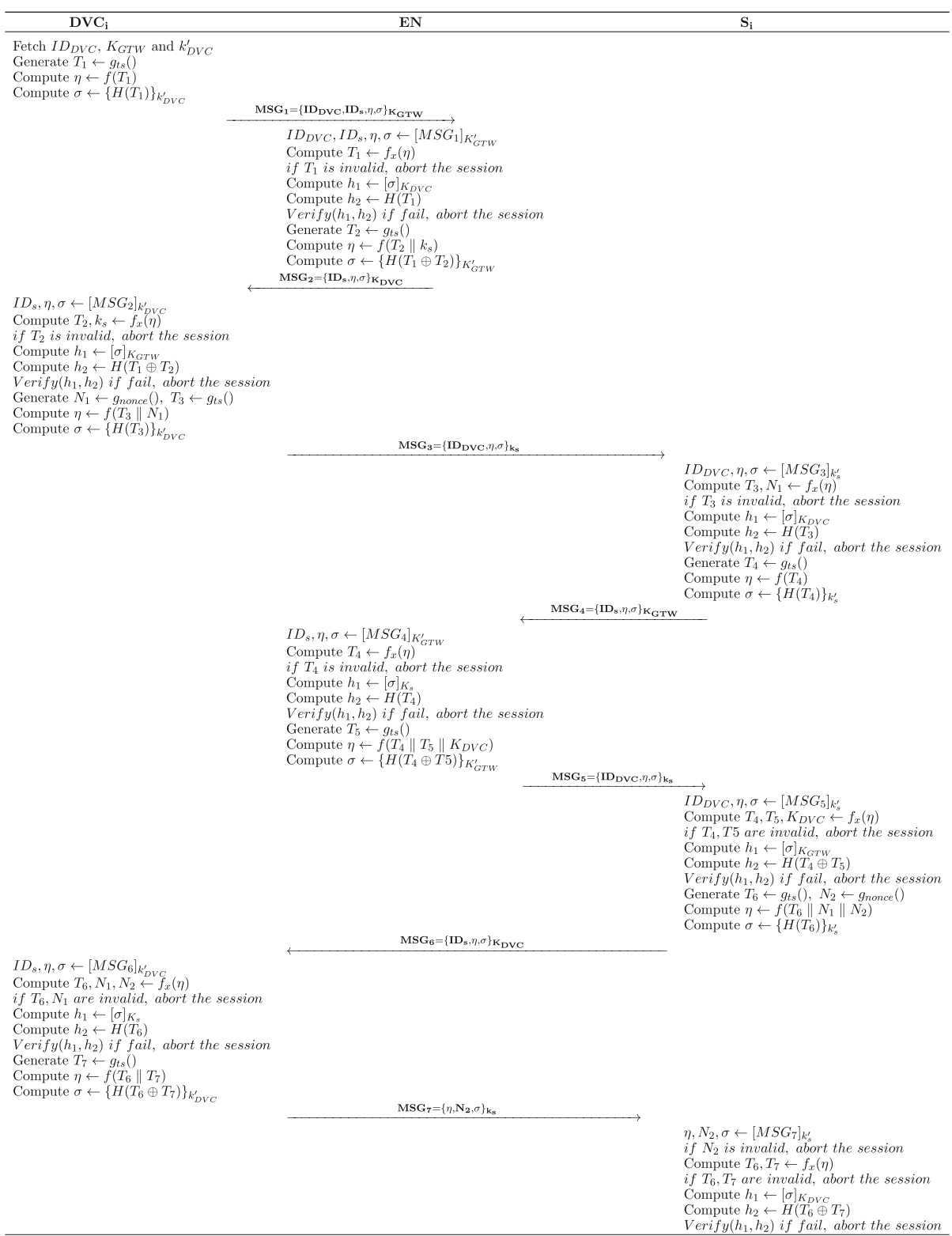


Fig. 6 PKAPG Authentication Protocol: public-key protocol for two-party mutual authentication

(1) The device (user or IoT node) first fetches the identity $ID_{DVC} = ID_u/ID_{ITN}$, keys pair $K'_{DVC_i} = K'_{u_i}/K'_{ITN_i}$ and K_{GTW} from smartcard/IoT device memory for communication. Now, the timestamp T_1 is then generated by executing the function $g_{ts}()$. Afterward the encrypted message η is created by calling the function $f(T_1)$. It then computes the message signature σ by encrypting $H(T_1)$ using the device's private key K'_{DVC}.

(2) Message MSG_1 is then constructed and encrypted with the EN's public key K_{GTW} before being forwarded to the EN.

(3) EN retrieves MSG_1 and decrypts it with the EN's private key K'_{GTW}. Afterward the value of T_1 is retrieved by calling the function $f_x(\eta)$. Then the validity of the received timestamp T_1 is checked by EN.

(4) If T_1 is invalid, the session is terminated. Otherwise, $h_1 \leftarrow [\sigma]_{K_{DVC}}$ and $h_2 \leftarrow H(T_1)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check for equality in both hash values. If the validation test fails, the session is terminated.

(5) The timestamp T_2 is then generated, and $\eta \leftarrow f(T_2 \,||\, k_s)$ and $\sigma \leftarrow \{H(T_1 \oplus T_2)\}_{K'_{GTW}}$ are computed. The message MSG_2 is then constructed and then encrypted with the device's public key K_{DVC} before being forwarded to the device DVC_i.

(6) DVC_i uses k'_{DVC} to decrypt the received message MSG_2. It then extracts η and retrieve T_2 and k_s. Then the validity of the timestamp T_2 is checked. If the T_2 is invalid, the session is terminated.

(7) Otherwise, $h_1 \leftarrow [\sigma]_{K_{GTW}}$ and $h_2 \leftarrow H(T_1 \oplus T_2)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check the equality of both hash values. If the validation test fails, the session is terminated.

(8) Subsequently, it generates both nonce N_1 and timestamp T_3 by calling the function $g_{nonce}()$ and $g_{ts}()$, correspondingly. Further, it constructs $\sigma \leftarrow \{H(T_3)\}_{k'_{DVC}}$ and $\eta \leftarrow f(T_3 \,||\, N_1)$. After that, message MSG_3 is constructed and encrypted using the cloud server's public key k_s, then forwarded to the cloud server S_i.

(9) The cloud server S_i retrieves MSG_3 and decrypts using its own private key k'_s. Now, T_3 and N_1 are retrieved after extracting the η. S_i checks the validity of the received timestamp T_3. It aborts the session if T_3 is invalid.

(10) Otherwise, $h_1 \leftarrow [\sigma]_{K_{DVC}}$ and $h_2 \leftarrow H(T_3)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check the equality of both hash values. If the validation test fails, the session is terminated. It now generates the timestamp T_4 by invoking the function $g_{ts}()$.

(11) It also constructs $\sigma \leftarrow \{H(T_4)\}_{k'_s}$ and $\eta \leftarrow f(T_4)$. Following that, message MSG_4 is constructed and encrypted with EN public key K_{GTW} before being forwarded to the EN.

(12) EN uses K'_{GTW} to decrypts the received Message MSG_4. It then extracts η in order to retrieve T_4. It now verifies the validity of the timestamp T_4. If the T_4 is invalid, the session is terminated.

(13) Otherwise, $h_1 \leftarrow [\sigma]_{K_s}$ and $h_2 \leftarrow H(T_4)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check the equality of both hash values. If the validation test fails, the session is terminated. It now generates the timestamp T_5 by invoking the function $g_{ts}()$.

(14) It also constructs $\eta \leftarrow f(T_4 \,||\, T_5 \,||\, K_{DVC})$ and $\sigma \leftarrow \{H(T_4 \oplus T5)\}_{K'_{GTW}}$. The message MSG_5 is then prepared and encrypted using the cloud server's public key k_s before being forwarded back to the cloud server S_i.

(15) The cloud server S_i retrieves MSG_5 and decrypts with its private key k'_s. After extracting the η, the values of T_4, T_5 and K_{DVC} are retrieved by calling the function $f_x(\eta)$. Then the S_i validates the received timestamps T_4 and T_5. If any of the timestamps is invalid, the session is terminated.

(16) Otherwise, it computes $h_1 \leftarrow [\sigma]_{K_{GTW}}$ and $h_2 \leftarrow H(T_4 \oplus T_5)$. Then it calls the function $Verify(h_1, h_2)$ to check the equality of both hash values. If the validation test fails, the session is terminated.

(17) Following that, it generates a timestamp $T_6 \leftarrow g_{ts}()$ and a nonce $N_2 \leftarrow g_{nonce}()$ by calling the functions $g_{ts}()$ and $g_{nonce}()$, respectively. It also constructs $\sigma \leftarrow \{H(T_6)\}_{k'_s}$ and $\eta \leftarrow f(T_6 \,||\, N_1 \,||\, N_2)$. Following that, message MSG_6 is constructed and encrypted with the device's public key K_{DVC} before being forwarded to the device DVC_i.

(18) DVC_i uses K'_{DVC} to decrypt the received Message MSG_6. It then extracts η in order to extract the T_6, N_1, and N_2. It now verifies the validity of

the T_6, N_1, and N_2. If any of T_6 or N_1 is invalid, the session is terminated.

(19) Otherwise, $h_1 \leftarrow [\sigma]_{K_s}$ and $h_2 \leftarrow H(T_6)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check the equality of both hash values. If the validation test fails, the session is terminated. It now generates the timestamp T_7 by invoking the function $g_{ts}()$.

(20) It also constructs $\eta \leftarrow f(T_6 \mid\mid T_7)$ and $\sigma \leftarrow \{H(T_6 \oplus T_7)\}_{k'_{DVC}}$. Following that, message MSG_7 is constructed and encrypted with the cloud server's public key k_s before being forwarded back to the cloud server S_i.

(21) MSG_7 is retrieved by cloud server S_i and decrypted with its private key k'_s. It now verifies the validity of the N_2. If N_2 is invalid, the session is terminated. Otherwise, η is extracted in order to retrieve both T_6 and T_7.

(22) Following that, S_i validates the received timestamps T_6 and T_7. If any of the timestamps is invalid, the session is terminated. Otherwise, $h_1 \leftarrow [\sigma]_{K_{DVC}}$ and $h_2 \leftarrow H(T_6 \oplus T_7)$ are computed. Then it calls the function $Verify(h_1, h_2)$ to check for equality in both hash values. If the validation test fails, the session is terminated.

In this section, we have discussed steps to authenticate and access information by the registered user and the registered IoT node. The registered user first login into the IoT infrastructure as explained in Section 3.2.1 by completing the authentication process. Once login is approved, the user may either access the edge node using DPKAP Authentication Protocol or the IoT node using PKAPG Authentication Protocol. The IoT node may access the edge node directly using DPKAP Authentication Protocol and may communicate with the user using PKAPG Authentication Protocol.

3.3 Cloud Storage Integration

Intelligence moves to the edge node in IoT with the help of edge computing leads to a data-intensive context where data is aggregated at the edge node. However, due to the huge volume of sensor data, it is not feasible to store and manage it at the edge node. As a result, the cloud module has been incorporated to provide secure and reliable data storage across cloud data centers. Before contacting the storage APIs, Edge Node (EN) performs several computations to maximize data availability and privacy in a distributed setting. Encoding and decoding operations are performed well before uploading the data to the cloud using the erasure encoding mechanism. The beauty of Erasure Coding (EC) is that it supports efficient data recovery and fault tolerance. Thus, even if a certain percentage of storage servers are unavailable, cloud datacenters ensure the reliability of stored data and provide the least affected services. EN encodes the traced data into several fragments after encryption using an error-correcting approach that includes both data and parity fragments. These extra parity fragments aid data recovery even if a few data fragments are lost. The data is traced hourly (up to 12 hours), resulting in a kilobytes or megabytes file. These fragments are spread among cloud datacenters' storage servers. Chouhan et al. [15] claim that the pairs ($\alpha = 5, \beta = 3$) and ($\alpha = 6, \beta = 4$) are preferable for data up to a few megabytes. With the least storage cost, these encoding pairs bring maximum availability at peak times or with few server unavailabilities. As a result, we used these pairings in our work for encoding and decoding. We may accept up to 3 and 4 fragments unavailability/loss using the pair ($\alpha = 5, \beta = 3$) and ($\alpha = 6, \beta = 4$), respectively. Our module only needs $\alpha = 5$ fragments with encoding pair (5, 3) and $\alpha = 6$ fragments with encoding pair (6, 4) for downloading and decoding operations to regenerate the original data. Even if some servers are unavailable or a few fragments are corrupted, it achieves reliability by reconstructing the original data.

3.4 Use Cases

The applications of IoT and edge nodes are widely adopted in many day-to-day solutions due to their communication capabilities over the Internet. However, security attributes such as integrity, confidentiality, authentication, and authorization are key challenges for real-world deployed applications. Therefore, the proposed authentication protocols can be employed to mitigate these challenges in several applications, such as precision agriculture, container monitoring, industry or factory automation, home automation, health or elderly monitoring, commercial building automation, smart metering system for water and electricity, sports and fitness management, connected vehicles, and smart city management.

4 Experimental Setup

This section provides the simulation and hardware-based experimental setup details.

4.1 AVISPA Simulation Setup

The experimental simulation has been carried out on the well-known and widely accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) [7] security analyzer simulation tool. The AVISPA simulator enables verification of the proposed protocols by checking the validity and secrecy during real-time simulated environment execution. The AVISPA simulation tool provides a medium for researchers to inspect whether the proposed security protocol is able to provide the required secrecy to the communicating entities during communication over a public network. It checks the proposed protocol against

Table 3 Experimental specification and parameters settings

Device/Tool	Parameters	Specifications
Raspberry Pi - IV Model B - 8GB RAM [20]	Processor	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
	RAM	8GB LPDDR4-3200 SDRAM
	Bluetooth	Bluetooth 5.0, BLE
	Wifi	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
	USB	2 USB 3.0 ports; 2 USB 2.0 ports
	Ethernet	Gigabit ethernet
	HDMI	2 × micro-HDMI ports
	Storage	microSD card slot
	Power Supply	5V DC minimum 3A
	Operating Temperature	0 - 50 degrees C ambient
Espressif ESP-WROOM-32 Development Board [44]	Processor	Tensilica Xtensa Dual-Core 32-bit LX6 microprocessor, running at 160 or 240 MHz
	RAM	SRAM - 520 KB
	Bluetooth	Bluetooth V4.2 BR/EDR, Bluetooth LE specification
	Wifi	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless 802.11 b/g/n (802.11n up to 150 Mbps), 2.4 GHz
	Pins	38
	Interfaces	UART, SPI, SDIO, I^2C,I^2S, LED PWM, GPIO, ADC, DAC
	Storage	2 MiB (ESP32-D2WD chip) & SD card support
	Power Supply	3.7 V DC
	Operating Temperature	-40°C +85/105°C
AVISPA Tool Configuration [7]	Virtualization Software	VirtualBox
	Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
	Core	1
	Operating System	Ubuntu (32-bit)
	RAM	2048 MB
	Chipset	PIIX3
	Graphic Controller	VBoxVGA
	Protocol Specification Language	HLPSL
	Protocol Analysers	CL-Atse and OFMC

vulnerabilities and determines whether it is safe or unsafe. The proposed protocol is marked as safe if it is able to provide the desired level of security if it fails, then it is marked as unsafe. AVISPA, the cryptographic protocols are specified using the High-Level Protocol Specification Language (HLPSL). Along with AVISPA, the SPAN, a security protocol animator tool, is bundled in an Oracle Virtual Box package with the ubuntu image and is freely downloadable from its website [8]. The virtual box image is installed on Ubuntu with 4 GB RAM. The proposed cryptographic protocols are designed using the HLPSL and simulated in AVISPA. The parameter settings of AVISPA are shown in Table 3. The proposed protocols are discussed in the Proposed Work (Section 3). The details of the Direct Public-Key Authentication Protocol (DPKAP) have been discussed in the DPKAP Authentication Phase (Section 3.2.2). While the details of the public-key authentication protocol for mutual authentication(PKAPG) have been discussed in the PKAPG Authentication Phase (Section 3.2.3). The outcome of the proposed protocols are discussed in the result and analysis section (Section 5).

4.2 Hardware Based Experimental Setup

This section deals with the partial test hardware-based experimental setup, which includes the integration of the IoT node with the edge node and the edge node with the cloud storage server. We deployed IoT nodes using ESP32-WROOM boards and the edge node using Raspberry Pi-IV. The hardware specifications of the ESP32-WROOM board and Raspberry Pi-IV are shown in Table 3.

4.2.1 IoT Nodes and Edge Node Integration

We create a deployment environment to test the proposed protocols using battery-powered ESP32 boards and the Raspberry Pi-IV. The IoT nodes and the edge node communicate wirelessly. The devices are deployed in a heterogeneous infrastructure-less fashion, as shown in Fig. 7. The Edge and IoT node may be deployed in a client-server with end-to-end security and message encryption. The IoT nodes are deployed either in the client-server formation or in the mesh network formation using ESP-NOW [43]. ESP-NOW provides message encryption using Counter Mode with cipher block chaining message authentication code protocol [43]. After authentication, the user may access edge and IoT nodes. The edge node periodically pushes or pulls data from the cloud using API, as discussed in Section 4.2.2. For test deployment, IoT nodes sense the environmental data using BMP280 sensors and securely log the sensed data to the edge node that may later be uploaded to the cloud. The IoT node security was implemented using Arduino Cryptography Library [48] and Mbed TLS library [51].

4.2.2 Edge Node and Cloud Integration

The description of cloud storage integration with edge node is discussed in Section 3.3. The encoding and decoding procedures that assist in achieving reliability during server unavailability are performed using the Python-based PyECLib-1.2.0 library with the liberasurecode-dev package. We use Python's crypto module for cryptographic operations, while Dropbox serves as a datacentre for our implementation. The IoT module is coupled with the cloud storage APIs to perform storing and retrieving activities.

5 Results and Analysis

The proposed protocols are validated using well known AVISPA simulator [7]. The validation results of the AVISPA simulation are presented in Section 5.1. The informal security analysis of the proposed protocols along with the comparative analysis of existing and proposed authentication schemes are discussed in Section 5.2. Further, in Section 5.3, we have examined the performance of cloud storage operation using erasure coding techniques including Encode, Decode, Upload, Download, and the space overhead of tracked data.

5.1 Validation using AVISPA Simulator

The security of the proposed authentication scheme is analyzed using well know and widely used AVISPA simulator. The AVISPA simulator determines whether a protocol is SAFE or UNSAFE against various attacks and analyzes different security attributes related to the protocol by informal security analysis. To check the security aspects, we simulate the proposed protocols

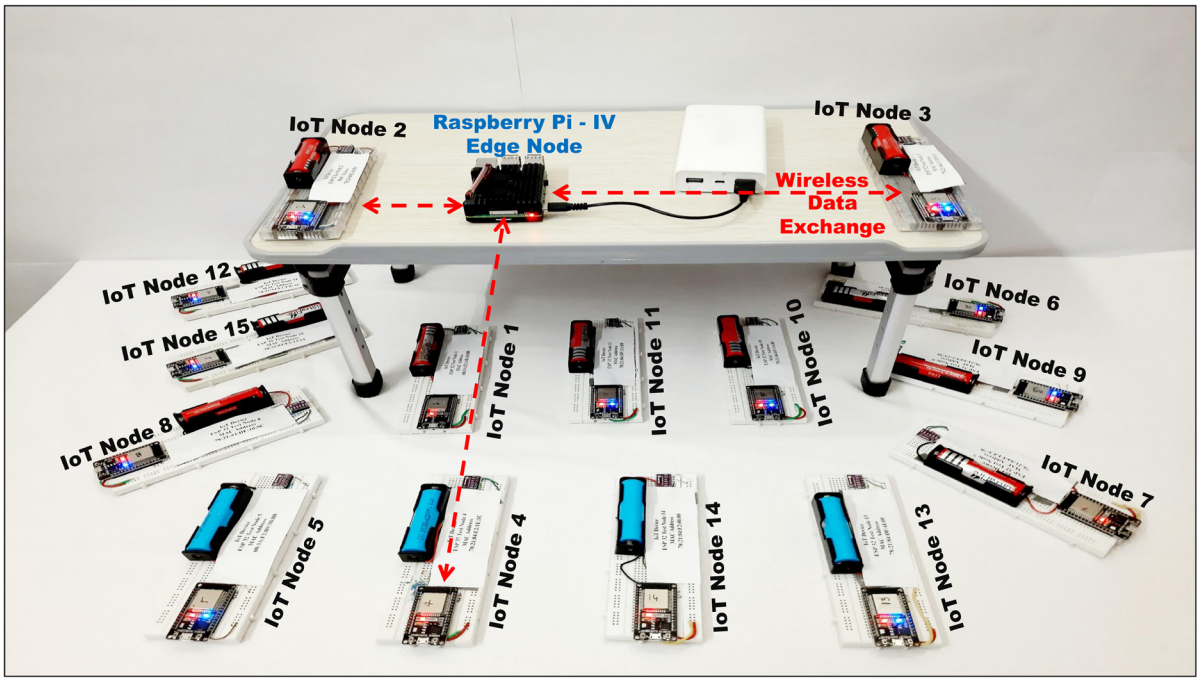


Fig. 7 IoT and edge node integration device setup

under OFMC and CL-AtSe backends [7] to formally assess its security strength against threats. We code our protocol in High-Level Protocol Specification Language (HLPSL), which is then converted to an Intermediate Format (IF) through the HLPSL2IF translator. The verification backends OFMC and CL-AtSe are then provided to IF as input. These backends then perform a security check on the IF and determine whether the protocol is safe or not. The simulation results based on the OFMC backend and CL-AtSe backend for DPKAP Protocol are shown in Fig. 8 and for PKAPG Protocol are shown in Fig. 9. In addition, we investigated the number of hash operations and execution costs (T_h) employed in existing authentication schemes, and through Table 4 we reveal the comparative hash efficiency of existing and proposed authentication mechanisms.

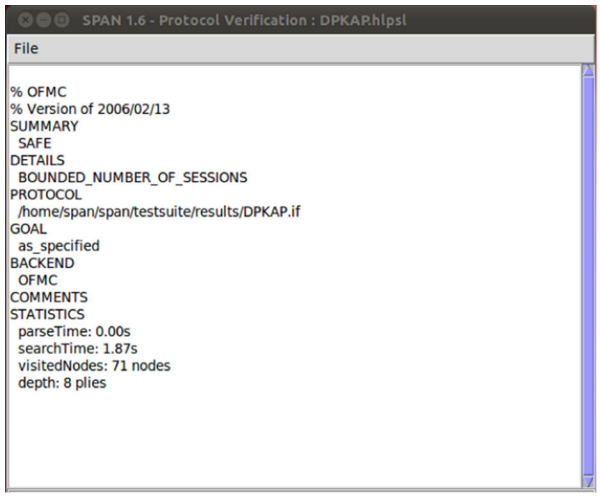
5.2 Analysis of the Proposed Authentication Scheme

The aspects of common security features for authentication analysis of the proposed authentication scheme is described in Section 5.2.2. In Section 5.2.1, we have examined the comprehensive security of the proposed authentication schemes against possible attacks. Further, the comparison of the proposed and existing authentication schemes are presented in Section 5.2.3.

5.2.1 Theoretical Security Analysis

The theoretical security analysis of the proposed authentication protocols against relevant attacks is presented in this section.

1. **Proposition:** The proposed protocol is secure against replay attacks.
   Proof: Our protocol uses a secure channel for transmitting information between entities. So eavesdropping related attacks are not possible. In the worst case, suppose an adversary $\mathcal{A}$ eavesdrop on MSG_i from previous communication during the authentication phase. Then $\mathcal{A}$ may use this forged/eavesdropped message later to target receiving entity (such as User, IoT Node, Edge Node). However, we utilize timestamp and/or nonce in each communication message in our protocols to resist replay attacks where the timestamp is used to keep data fresh and nonce to avoid various forms of response assaults. The receiver entity immediately

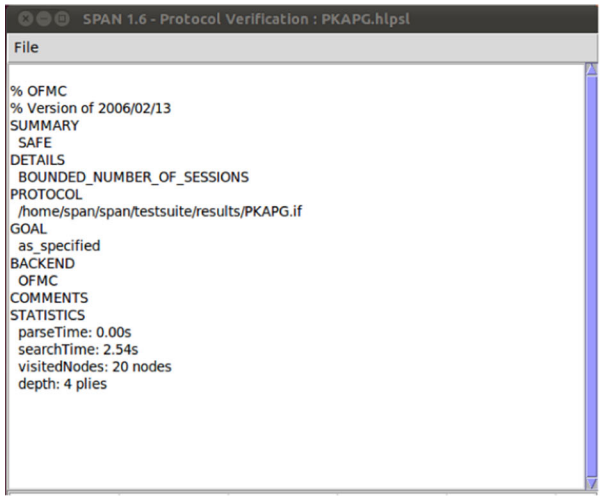


(a) Results based on OFMC backend **(b)** Results based on CL-AtSe backend

Fig. 8 AVISPA Simulation Results for DPKAP Protocol

aborts the session in case of timestamp(s) is invalid.

In DPKAP authentication protocol, we attach a digital signature that verifies the message's authenticity to confirm that the message has been sent by a legitimate user and has not been altered. A signature is constructed by encrypting the computed hash value of a timestamp(s) and Session key. This hash value is protected with a secure one-way hash function. The integrity of the message is verified using digital signature by comparing the received hash value with the derived hash value. The verification is valid only if the integrity of the received session key (sk) and timestamp(s) are preserved; otherwise, the session is aborted immediately. Thus, an adversary will fail in verification checks of the protocol. This is because when receiver calculates the hash $h_1 \leftarrow [\sigma]_k$ and $h_2 \leftarrow H(\sum_{i=1}^{n} T_i \oplus sk)$ they come out to be unequal. Hence, the proposed DPKAP authentication protocol is secure against or resists



(a) Results based on OFMC backend **(b)** Results based on CL-AtSe backend

Fig. 9 AVISPA Simulation Results for PKAPG Protocol

Table 4 The efficiency comparison of hash computations and running times of algorithms

Protocol	DVC_i	S_j	GW/EN	Total computation	Running time
Xue et al. [59]	$7T_h$	$6T_h$	$13T_h$	$26T_h$	≈ 0.0104 ms
Turkanović et al. [46]	$7T_h$	$5T_h$	$7T_h$	$19T_h$	≈ 0.0076 ms
Amin et al. [2]	$8T_h$	$5T_h$	$8T_h$	$21T_h$	≈ 0.0084 ms
Amin et al. [3]	$12T_h$	$5T_h$	$16T_h$	$33T_h$	≈ 0.0132 ms
Mishra et al. [33]	$9T_h$	$6T_h$	$11T_h$	$26T_h$	≈ 0.0104 ms
Wu et al. [52]	$11T_h$	$6T_h$	$17T_h$	$34T_h$	≈ 0.0136 ms
Ostadsharif et al. [6]	$11T_h$	$5T_h$	$17T_h$	$33T_h$	≈ 0.0132 ms
Zhou et al. [62]	$10T_h$	$7T_h$	$19T_h$	$36T_h$	≈ 0.1152 ms
Kang et al. [28]	$8T_h$	$4T_h$	$11T_h$	$23T_h$	≈ 0.0736ms
Huang et al. [24]	$8T_h$	$4T_h$	$10T_h$	$22T_h$	≈ 0.0704 ms
Wu et al. [54]	$10T_h$	$5T_h$	$12T_h$	$27T_h$	≈ 0.0864 ms
DPKAP Scheme	$2T_h$	—	$2T_h$	$4T_h$	≈ 0.0128 ms
PKAPG Scheme	$5T_h$	$5T_h$	$4T_h$	$14T_h$	≈ 0.0448 ms

Abbreviations: *DVC_i : ITN_i-IoT Node/U_i-Users, S_j- Cloud Server/Server, EN - Edge Node, GW - Gateway*

replay attacks.

In PKAPG authentication protocol, we attach a digital signature that uses timestamp(s) and/or nonce using a one-way hash function to verify the received hash value with the derived hash value. The verification is valid only if the integrity of received nonce(s) and timestamp(s) are preserved; otherwise, the session is aborted immediately. Thus, an adversary will fail in verification checks of the protocol. This is because when receiver calculates the hash $h_1 \leftarrow [\sigma]_k$ and $h_2 \leftarrow H(\sum_{i=1}^{n} T_i)$ they come out to be unequal. Hence, the proposed PKAPG authentication protocol is secure against or resists replay attacks.

2. **Proposition:** The proposed protocol is secure against impersonation attacks.
Proof: In this attack, an adversary $\mathcal{A}$ attempts to create a new login request message after intercepting the login message of a legitimate sender entity (such as User, IoT Node, Edge Node) during protocol execution. However, in our proposed schemes, adversary $\mathcal{A}$ is unable to login because we uses a tamper-proof smartcard along with biometric and password protection. Suppose $\mathcal{A}$ steals a smartcard and retrieves information $ID_u, r, L_i, TPW_i, SCN_i, N_i, H(), B(,)$ from the memory of the smartcard, where $TPW_i = H(PW_i \parallel r)$ and $L_i = H(SCN_i \oplus N_i \oplus TPW_i)$ for authentication and login, the attacker needs $< fng_i, PW_i >$, which are unknown to the attacker, thus the attacker is unable to impersonate a legal user/entity. Moreover, in our protocols, we uses a secure channel between all entities, and all messages are encrypted with a digital signature mechanism, so it is difficult or impossible for attacker $\mathcal{A}$ to retrieve any valid information by forge/intercept communication. Moreover, all forged messages are detected during $Verify(h_1, h_2)$ operation, as mentioned in Figs. 4 and 6. Therefore, $\mathcal{A}$ cannot launch impersonation attacks in our proposed protocols. Similarly, it provides resistance against several attacks, such as eavesdropping, man-in-the-middle attacks, and forging attacks.

3. **Proposition:** The proposed protocol is secure against guessing attacks and user anonymity.
Proof: In proposed protocols, $\mathcal{A}$ is unable to derive or guess ID_i, PW_i, and fng_i from the smartcard, because we use tampered proof memory for storing tuple information. Further, we use a secure channel with public key cryptography for transmitting messages between entities (such as User, IoT Node, and Edge Node). Each message is encrypted with the receiver's public key, so only authorized receivers can decrypt the encrypted message. Therefore, it is not feasible for an attacker to forge the message

and guess the identity of the sender entity. Moreover, in proposed authentication protocols, login and authentication instances are protected with a secure one-way hash function and tamper-proof memory of the smartcard. Besides, the ID is never transmitted over an insecure channel. Hence user identity and anonymity are protected and cannot be retrieved or guessed by an adversary.

4. **Proposition:** The proposed protocol is secure against forward secrecy.
   Proof: Proposed authentication protocols support forward secrecy by involving unpredictable variations using fresh timestamp(s) and/or nonces in every session, which protect forgery and provide randomness. These variables are impossible to predict accurately by an adversary. Therefore, the proposed scheme ensures forward secrecy.
5. **Proposition:** The proposed protocol is secure against a session key for all participant entities.
   Proof: The freshness of the session key is maintained in each session by generating a new random session key. Freshness is the key feature of a session key, which means a distinct random key is used for each new session. We share a common session key among the entities in the proposed direct public key authentication protocol. $\mathcal{A}$ is unable to get sk during transmitting messages between entities because we use a secure channel with public key cryptography. Each message is encrypted with the receiver's public key, so only authorized receivers can decrypt the received encrypted message. Therefore, it is not feasible for attackers to extract session keys from messages during protocol execution.

6. **Proposition:** The proposed protocol is secure against the stolen-smartcard and stolen-verifier attacks.
   Proof: In this attack, $\mathcal{A}$ attempts to retrieve users' and edge nodes' confidential information after extracting information from smartcards. In our work, we uses tamper-proof memory for storing essential information in the smartcard. So $\mathcal{A}$ is unable to retrieve information from tamper-proof memory. Suppose an attacker can retrieve login and authentication instances; however, these instances are even protected with a secure one-way hash function. Moreover, login requires additional information, including secrets (i.e., PW and fng_i) which is impossible for an attacker to guess. Since the attacker cannot extract or guess any confidential information of the user or edge node from the smartcard, our protocol is secure against a stolen-smartcard attack.

In our work, we store required information in tamper-proof memory, which protects it from offline attacks. Public key cryptography is employed during communication with a secure channel, which enables resistance against several attacks of forgery, such as guessing, impersonation, and replay attacks.

5.2.2 Evaluation Parameters for Authentication Analysis

To evaluate the security strength of the proposed protocols, we compared it with existing works in Section 5.2.3 based on the following evaluation parameters:

(1) Mutual Authentications: Both communicating parties must be able to mutually authenticate themselves in order to ensure the integrity of messages.
(2) Accessibility: The IoT nodes, edge node, and cloud infrastructure must be accessible for authenticated users.
(3) Confidentiality: The trusted System Administrator or Trusted Third Party verifies the user, the IoT node, and the cloud infrastructure. Upon receiving a message from such a user or device, the receiver is rest assured that it comes from a legitimate entity.
(4) Scalability: As the number of registered users and IoT nodes keeps changing, the system must scale up or down resources to cater to the increase or decrease in the number of requests from users or IoT nodes.
(5) Anonymity - The identity of entities does not disclose to any unauthorized party or adversary. the receiver's public key, and the receiver decrypts them using their private key.
(6) Secure Cloud Storage- The data is securely stored in the cloud using the erasure coding technique and other cryptographic primitives.
(7) Secure User-Edge communication - The user, as well as the edge node, must authenticate themselves before communicating with each other.
(8) Secure User-IoT node communication - The user, as well as the device node, must authenti-

cate themselves before communicating with each other.

(9) Secure IoT node-Edge communication - The device, as well as the edge node, must authenticate themselves before communicating with each other.

(10) Secure IoT node/Edge node/User-Cloud communication - The user/IoT node/Edge node securely communicates with each other.

(11) Trusted Third Party -The trusted third party is a trusted entity responsible for assuring the identity of all parties (user, IoT node, edge node/gateway, and cloud servers) involve in communication; it validates and verifies the identity of parties involve in communication.

5.2.3 Comparative Analysis of Existing and Proposed Authentication Schemes

As shown in Table 5, our proposed scheme meets all the security parameters, while other approaches partially meet the security parameters. The proposed scheme also provides mutual authentication as stakeholders mutually authenticate each other. The user, upon successful authentication, is able to access resources. The System Administrator verifies the parties involved in the communication, and the confidentiality of the sender or receiver is ensured. Any user or device can join and begin using or providing services after registering with the System Administrator. Also, the user and device may leave the infrastructure; hence the proposed system is scalable. The sender sends a message encrypted with the receiver's public key. Hence, our proposed scheme works securely even on an insecure communication medium. Classical cloud storage is prone to failures of software, hardware, and sometimes the network; therefore, we use the erasure coding mechanism with other cryptographic primitives to provide secure and reliable cloud storage. The proposed scheme supports secure communication between all parties.

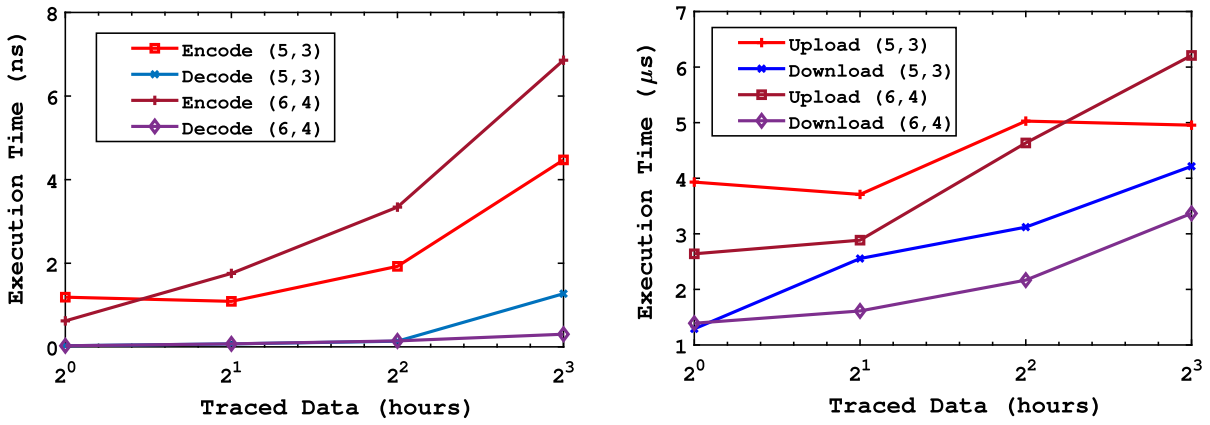
5.3 Analysis of the Cloud Scheme

The generated data are stored in the cloud storage server using the erasure coding scheme. The outcomes of the encoding and decoding parameters of the erasure coding are analyzed in Section 5.3.1. The cloud storage upload and download parameters of erasure coding are analyzed in Section 5.3.2. Finally, in Section 5.3.3, we analyzed the cloud storage overhead for the proposed erasure encoding scheme.

Table 5 Comparison of security requirement parameters

Schemes	MA	Access	Conf	Scale	Anony	SIEC	SUEC	SUIC	SCC	NTTP
Li et al. [32]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Xue et al. [58]	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓
Amin et al. [4]	✓	✗	✓	✗	✓	✗	✗	✗	✓	✓
Amin et al. [5]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Ostadsharif et al. [6]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Wu et al. [53]	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓
Kumar [29]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Wu et al. [55]	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓
Wu et al. [56]	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓
Shahidinejad et al. [39]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗
Zargar et al. [61]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗
Proposed Scheme	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

***Parameters Abbreviations**: MA: Mutual Authentications, Access: Accessibility, Conf: Confidentiality, Scale: scalability, Anony: Anonymity, SIEC: Secure IoT node-Edge communication, SUEC: Secure User-Edge communication, SUIC: Secure User-IoT node communication, SCC: Secure IoT node/Edge node/User-Cloud communication, NTTP: No Trusted Third Party Required*
***Symbols**: ✗: scheme does not support this parameter ✓: scheme supports this parameter*



(a) Encoding and Decoding Operations **(b)** Upload and Download Operations

Fig. 10 Performance of erasure coding scheme on traced data

5.3.1 Analysis of the Cloud Storage Encoding and Decoding Parameters

The encoding and decoding procedures are carried out using the erasure coding mechanism. The streams of $\{2^i; 0 \leq i < 4\}$ hours were evaluated after tracing numerous data streams from the IoT module. The traced data files are tiny to medium in size and measured in kilobyte or megabyte. As a result, we evaluate the encoding pairs (5, 3) and (6, 4) in light of the erasure coding performance requirements indicated in Chouhan et al. [15]. As a result, we have a high level of availability, decent recovery support, and average storage overheads. Figure 10(a) illustrates the average performance for encoding and decoding operations of traced data.

The performance of required operations was measured using Python's *timeit* module. Exponential traces data up to 8 hours is considered, and the related performance is represented in the graph. We see that encoding operations take between 0.5 and 7.0 nanoseconds to complete. On the other hand, decoding processes take between 0.1 and 1.5 nanoseconds to complete. Encode operations are generally more expensive than decode operations because they require extra computations.

5.3.2 Analysis of the Cloud Storage Upload and Download Factors

For uploading data to numerous servers in the cloud, data is broken into fragments. As a result, for each fragment, all procedures are carried out simultaneously based on the encoding values, and the required operations are executed in a short span of time. We capture the performance of the store and retrieve cloud operations with traced data and measure the average performance using the *timeit* module. The storage procedures (i.e., upload and download) for specific encoding pairs are shown in Fig. 10(b). Exponential traces data up to 8 hours is considered, and the related performance is represented in the graph. We can see that uploading activities take between 3.5 and 6.5 microseconds to com-

Table 6 Storage overhead with traced data

Traced Data (hrs)	File Size (KB)	$\psi_{(5,3)}$ (bytes)	$\psi_{(6,4)}$ (bytes)	$S_{oh_{(5,3)}}\%$	$S_{oh_{(6,4)}}\%$
2^0	160	32712	27272	59.72	66.45
2^1	436	89282	74416	59.98	66.67
2^2	822	168340	140296	59.99	66.67
2^3	1502	307604	256350	59.99	66.67

plete. Similarly, downloading processes take between 1.0 and 4.5 microseconds to complete.

5.3.3 Cloud Storage Overhead Analysis

The space overhead of traced data up to 8 hours is summarised in Table 6. First, we look at the size of each data fragment (ψ) in the set. Then, using both encoding pairings, we calculate the average overhead size (S_{oh}) of the required data corresponding to the traced data files. The pair (5,3) requires around 60 percent overhead, while the pair (6,4) requires around 66 percent. Pair (6,4), on the other hand, has better recoverability support than pair (5,3). The parity fragments represent the optimal solution for reliable storage and recoverability, as pair (6, 4) has some additional storage overhead as compared to pair (5, 3). However, at the same time, the rate of recoverability for pair (6, 4) is higher than that of pair (5, 3) since pair (6, 4) uses additional redundant parity fragments than pair (5, 3).

6 Conclusions and Future Work

To conclude, this study introduces secure IoT, Edge, and cloud integration with reliable storage through authentication protocols, cryptographic primitives, and the erasure coding mechanism. We validate the proposed authentication protocols using the well-known and widely accepted AVISPA simulator tool, reporting that the proposed authentication scheme is secure against a wide range of attacks. The proposed scheme is able to achieve mutual authentication, accessibility, confidentiality, scalability, message encryption, secure cloud storage, and secure communication mechanisms between the edge node, IoT nodes, users, and cloud infrastructure, which makes the proposed scheme more efficient. On the other hand, in the proposed architecture, we innovatively adopt the encoding parameters for the erasure coding module based on the recent literature to achieve reliability even when a few storage servers are unavailable. Following that, we discover that all of the traced files' encoding operations take extremely little time, ranging from 0.5 to 7.0 nanoseconds. On the other hand, decoding takes anywhere from 0.1 to 1.5 nanoseconds to complete. Similarly, the uploading activities take between 3.5 and 6.5 microseconds to complete with minimal storage overheads, whereas downloading operations take between 1.0 and 4.5 microseconds. In the future, we will explore our work by adopting blockchain technology to establish direct trust between the Edge node, users, IoT nodes, and cloud resources in order to provide more reliable and robust authentication, authorization, and accounting solutions for all involved parties. Further, we intend to deploy our proposed security protocols for precision agriculture that could fulfill the requisite security metrics and desired efficiency simultaneously in the real world.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Alon, N., Edmonds, J., Luby, M.: Linear time erasure codes with nearly optimal recovery. In: Proceedings of IEEE 36th Annual Foundations of Computer Science, pp. 512–519 (1995)
2. Amin, R., Biswas, G.: A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. Ad Hoc Networks **36**, 58–80 (2016)
3. Amin, R., Islam, S.H., et al.: Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks. Compu. Netw. **101**, 42–62 (2016). Industrial Technologies and Applications for the Internet of Things
4. Amin, R., Kumar, N., Biswas, G., Iqbal, R., Chang, V.: A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment. Futur. Gener. Comput. Syst. **78**, 1005–1019 (2018)
5. Amin, R., Kunal, S., Saha, A., Das, D., Alamri, A.: CFSec: Password based secure communication protocol in cloud-fog environment. J. Parallel. Distrib. Comput. **140**, 52–62 (2020)
6. Arezou Ostad-Sharif, H.A., et al.: Three party secure data transmission in IoT networks through design of a lightweight authenticated key agreement scheme. Futur. Gener. Comput. Syst. **100**, 882–892 (2019)
7. Armando, A., Basin, D., et al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: International conference on computer aided verification, pp. 281–285. Springer (2005)
8. Armando, A., Basin, D., et al.: Span plus avispa. (2017). https://people.irisa.fr/Thomas.Genet/span/
9. Blomer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., Zuckerman, D.: An xor-based erasure-resilient coding scheme (1999)
10. Chaudhary, A., Peddoju, S.K.: The role of IoT-based devices for the better world. In: Mishra, D.K., Azar, A.T., Joshi, A. (eds.) Information and Communication Technology, pp. 299–309. Springer Singapore, Singapore (2018)

11. Chaudhary, A., Peddoju, S.K., Kadarla, K.: Study of internet-of-things messaging protocols used for exchanging data with external sources. In: 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 666–671 (2017)
12. Chaudhary, A., Peddoju, S.K., Peddoju, S.K.: Cloud based wireless infrastructure for health monitoring. Virtual and Mobile Healthcare: Breakthroughs in Research and Practice pp. 34–55 (2020)
13. Chen, T.H., Shih, W.K.: A robust mutual authentication protocol for wireless sensor networks. ETRI J. **32**(5), 704–712 (2010)
14. Chen, Y., Ge, Y., Wang, Y., Zeng, Z.: An improved three-factor user authentication and key agreement scheme for wireless medical sensor networks. IEEE Access **7**, 85440–85451 (2019)
15. Chouhan, V., Peddoju, S.K.: Investigation of optimal data encoding parameters based on user preference for cloud storage. IEEE Access **8**, 75105–75118 (2020)
16. Chouhan, V., Peddoju, S.K.: Reliable verification of distributed encoded data fragments in the cloud. J. Ambient Intell. Humanized Comput. 1–17. (2020)
17. Das, A.K., Sharma, P., et al.: A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. J. Netw. Comput. Appl. **35**(5), 1646-1656 (2012)
18. Das, M.L.: Two-factor user authentication in wireless sensor networks. IEEE Trans. Wirel. Commun. **8**(3), 1086–1090 (2009)
19. Farash, M.S., Turkanović, M., Kumari, S., Hölbl, M.: An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. Ad Hoc Networks **36**, 152–176 (2016)
20. Foundation, R.P.: Raspberry pi 4 tech specs. (2022). https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/
21. Greenan, K.M., Miller, E.L., Wylie, J.J.: Reliability of flat xor-based erasure codes on heterogeneous devices. In: 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), pp. 147–156. IEEE (2008)
22. He, D., Gao, Y., Chan, S., Chen, C., Bu, J.: An enhanced two-factor user authentication scheme in wireless sensor networks. Ad-Hoc Sens. Wirel. Netw. **10**(4), 361–371 (2010). Cited By 211
23. Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., Yekhanin, S.: Erasure coding in windows azure storage. In: 2012 USENIX Annual Technical Conference (USENIX ATC 12), pp. 15–26 (2012)
24. Huang, H., Lu, S., Wu, Z., Wei, Q.: An efficient authentication and key agreement protocol for IoT-enabled devices in distributed cloud computing architecture. EURASIP J. Wirel. Commun. Netw. Conf. **2021**(1), 150 (2021). https://doi.org/10.1186/s13638-021-02022-1
25. Huang, H.F., Chang, Y.F., Liu, C.H.: Enhancement of two-factor user authentication in wireless sensor networks. In: 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 27–30 (2010)
26. Jiang, Q., Zeadally, S., Ma, J., He, D.: Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. IEEE Access **5**, 3376–3392 (2017)
27. Kadarla, K., Sharma, S., Bhardwaj, T., Chaudhary, A.: A simulation study of response times in cloud environment for IoT-based healthcare workloads. In: 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 678–683 (2017)
28. Kang, B., Han, Y., Qian, K., Du, J.: Analysis and improvement on an authentication protocol for IoT-enabled devices in distributed cloud computing environment. Math. Probl. Eng. **2020**, 1970798 (2020). https://doi.org/10.1155/2020/1970798
29. Kumar, D.: A secure and efficient user authentication protocol for wireless sensor network. Multimedia Tools Appl. **80**(18), 27131–27154 (2021)
30. Kumar, P., Om, H.: A conditional privacy-preserving and desynchronization-resistant authentication protocol for vehicular ad hoc network. J. Supercomput. 1–32 (2022)
31. Li, J., Li, B.: Beehive: erasure codes for fixing multiple failures in distributed storage systems. IEEE Trans. Parallel Distrib. Syst. **28**(5), 1257–1270 (2017)
32. Li, L.H., Lin, L.C., et al.: A remote password authentication scheme for multiserver architecture using neural networks. IEEE Trans. Neural. Netw. **12**(6), 1498–1504 (2001)
33. Mishra, D., Vijayakumar, P., Sureshkumar, V., Amin, R., Islam, S.H., Gope, P.: Efficient authentication protocol for secure multimedia communications in IoT-enabled wireless sensor networks. Multimedia Tools Appl. **77**(14), 18295–18325 (2018). https://doi.org/10.1007/s11042-017-5376-4
34. Mo, J., Hu, Z., Shen, W.: A provably secure three-factor authentication protocol based on chebyshev chaotic mapping for wireless sensor network. IEEE Access **10**, 12137–12152 (2022)
35. Nyang, D., Lee, M.K.: Improvement of Das's two-factor authentication protocol in wireless sensor networks. IACR Cryptol. ePrint Arch. **2009**, 631 (2009)
36. Rashmi, K., Shah, N.B., et al.: A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers. SIGCOMM Comput. Commun. Rev. **44**(4), 331–342 (2014)
37. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)
38. Schnjakin, M., Metzke, T., Meinel, C.: Applying erasure codes for fault tolerance in Cloud-RAID. In: 2013 IEEE 16th International Conference on Computational Science and Engineering, pp. 66–75. IEEE (2013)
39. Shahidinejad, A., et al.: Light-edge: A lightweight authentication protocol for IoT devices in an Edge-Cloud Environment. IEEE Consum. Electron. Mag. 1–1 (2021)
40. Shukla, S., Patel, S.J.: A novel ecc-based provably secure and privacy-preserving multi-factor authentication protocol for cloud computing. Computing **104**(5), 1173–1202 (2022). https://doi.org/10.1007/s00607-021-01041-6
41. Srinivas, J., Mukhopadhyay, S., Mishra, D.: Secure and efficient user authentication scheme for multi-gateway wireless sensor networks. Ad Hoc Networks **54**, 147–169 (2017)

42. Stergiou, C., Psannis, K.E., Kim, B.G., Gupta, B.: Secure integration of IoT and cloud computing. Futur. Gener. Comput. Syst. **78**, 964–975 (2018)
43. Systems, E.: ESP-NOW (2022). https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html?highlight=esp_now_set_pmk#security
44. Systems, E.: Esp32 series of modules- esp32-wroom series. (2022). https://www.espressif.com/en/products/modules/esp32
45. Turkanovic, M., Holbl, M.: An improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks. Elektronika ir Elektrotechnika **19**(6), 109–116 (2013)
46. Turkanović, M., Brumen, B., Hölbl, M.: A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. Ad Hoc Netw. **20**, 96–112 (2014)
47. Wang, C., Wang, D., et al.: Understanding node capture attacks in user authentication schemes for wireless sensor networks. IEEE Trans. Dependable Secure Comput. 1–1 (2020)
48. Weatherley, R.: Arduino cryptography library. (2020). https://rweather.github.io/arduinolibs/index.html
49. Weatherspoon, H., Kubiatowicz, J.D.: Erasure coding vs. replication: A quantitative comparison. In: International Workshop on Peer-to-Peer Systems, pp. 328–337. Springer (2002)
50. Woitaszek, M., Tufo, H.M.: Tornado codes for maid archival storage. In: 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), pp. 221–226. IEEE (2007)
51. Wolfe, M.: Mbed TLS (2016). https://github.com/wolfeidau/mbedtls. Accessed 2022
52. Wu, F., Li, X., Sangaiah, A.K., Xu, L., Kumari, S., Wu, L., Shen, J.: A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks. Futur. Gener. Comput. Syst. **82**, 727–737 (2018). https://doi.org/10.1016/j.future.2017.08.042. (https://www.sciencedirect.com/science/article/pii/S0167739X1730523X)
53. Wu, F., Li, X., et al.: A novel three-factor authentication protocol for wireless sensor networks with IoT notion. IEEE Syst. J. **15**(1), 1120–1129 (2021)
54. Wu, T.Y., Kong, F., Meng, Q., Kumari, S., Chen, C.M.: Rotating behind security: An enhanced authentication protocol for IoT-enabled devides in distributed cloud computing architecture. (2022). https://doi.org/10.21203/rs.3.rs-1554621/v1
55. Wu, T.Y., Yang, L., Lee, Z., Chu, S.C., Kumari, S., Kumar, S.: A provably secure three-factor authentication protocol for wireless sensor networks. Wirel. Commun. Mob. Comput. **2021** (2021)
56. Wu, T.Y., Yang, L., Luo, J.N., Wu, M.-T.J.: A provably secure authentication and key agreement protocol in cloud-based smart healthcare environments. Secur. Commun, Netw (2021)
57. Wylie, J.J., Swaminathan, R.: Determining fault tolerance of xor-based erasure codes efficiently. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pp. 206–215. IEEE (2007)
58. Xue, K., Hong, P., et al.: A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. J. Comput. Syst. Sci. **80**(1), 195–206 (2014)
59. Xue, K., Ma, C., Hong, P., Ding, R.: A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. J. Netw. Comput. Appl. **36**(1), 316–323 (2013)
60. Yuan, J., Jiang, C., Jiang, Z.: A biometric-based user authentication for wireless sensor networks. Wuhan University J. Natl. Sci. **15**(3), 272–276 (2010)
61. Zargar, S., Shahidinejad, A., Ghobaei-Arani, M.: A lightweight authentication protocol for IoT-based cloud environment. Int. J. Commun. Syst. **34**(11), e4849 (2021)
62. Zhou, L., Li, X., Yeh, K.H., Su, C., Chiu, W.: Lightweight IoT-based authentication scheme in cloud computing circumstance. Futur. Gener. Comput. Syst. **91**, 244–251 (2019). https://doi.org/10.1016/j.future.2018.08.038. (https://www.sciencedirect.com/science/article/pii/S0167739X18307878)