applied sciences

MDPI

Article

Secure and Efficient Authentication Scheme in IoT Environments

Abhijeet Thakare and Young-Gab Kim *

Department of Computer and Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, Korea; alwaysabhi@sju.ac.kr
* Correspondence: alwaysgabi@sejong.ac.kr

Abstract: Optimization of resource consumption and decreasing the response time of authentication requests is an immense urgent requirement for supporting the scalability of resources in IoT environments. The existing research attempts to design lightweight authentication protocols to address these issues. However, the schemes proposed in the literature are lacking in the creation of a lightweight (i.e., low computing, communication, and storage cost) and secure architecture. IoT devices in existing approaches consume high electricity and computing power, despite the fact that IoT devices have limited power and computing capabilities. Furthermore, the existing approaches lead to an increase in the burden on storage memory and also create heavy traffic on a communication channel, increasing the response time of device authentication requests. To overcome these limitations, we propose a novel lightweight and secure architecture that uses crypto-modules, which optimize the usage of one-way hash functions, elliptic-curve cryptography, and an exclusive-or operation. We demonstrate the proposed scheme's security strength using informal security analysis and verified it by considering the widely used automated validation of internet security protocol application (AVISPA) and the ProVerif tool. The result shows that the proposed scheme is effective against active and passive security attacks and satisfies secure design. Moreover, we calculate the proposed scheme's working cost by implementing it using a widely accepted standard pairing-based cryptography (PBC) library on embedded devices. The implementation proves that the proposed scheme is lightweight and reduces computation time by 0.933 ms, communication cost by 1408 bits, and storage cost by 384 bits, and removes the existing gaps.

Keywords: authentication scheme; elliptic-curve cryptography (ECC); Internet of Things; cloud; pairing-based cryptography (PBC)

Citation: Thakare, A.; Kim, Y.-G. Secure and Efficient Authentication Scheme in IoT Environments. *Appl. Sci.* **2021**, *11*, 1260. https://doi.org/10.3390/app11031260

Academic Editor: Tiago M. Fernández-Caramés
Received: 11 December 2020
Accepted: 25 January 2021
Published: 29 January 2021

1. Introduction

Owing to the fast advancement of cutting-edge innovation in the realm of ubiquitous computing, Internet of Things (IoT) environments have caught significant attention in the field of information technology (IT) business and network communication. The principal motivation behind IoT environments is to obtain data from the environment, following which various other gadgets can share the data for processing. Consequently, IoT becomes a significant and widely utilized innovation with respect to the presently connected world, particularly in health services and critical infrastructure. Therefore, to deal with exponentially growing heterogeneous data and devices, IoT is connected to a standard cloud platform where critical information about individuals is stored. Therefore, the safety, privacy, and security of such data are of paramount information [1]. Hence, unauthorized clients must not be allowed to access data from private cloud servers. In the literature, it was observed that the security of cloud servers could be infringed through the exploitation of security loopholes such as fraudulence, spying, and falsification presented by either interminable attackers or malignant interior entities [2]. Moreover, many kinds of IoT devices (e.g., IP cameras, smart speakers) are vulnerable to malware (e.g., Mirai malware [3]) that might break up the internet infrastructure, including IoT ecosystems [4–6]. Therefore, to ensure that only authenticated parties can access the applications and stakeholders'

assets (e.g., IoT devices), client verification is necessary to verify the electronic transaction by obtaining collective information about the client [7]. The seriousness of data security on the cloud was gauged by the European commission in 2010, as it recognized both security and protection as significant links in the chain of IoT ecosystems in order for IoT to achieve ubiquity. For providing end-to-end security, IoT use-case will have to incorporate strategies to provide end-device security, message authentication, authorization, integrity, confidentiality, denial, availability, and privacy protection [8]. The entire array of security strategies can be supported by designing an efficient authentication process.

Because of this, the authentication process is the principal first layer of security against potential attackers. The authentication process aims to validate the identity of entities or devices that request access to the private cloud platform by performing identity verification; thus, a reliable verification protocol requires performing two functions, namely, mutual verification and session-key agreement [9]. The authentication protocols typically use encryption technology to achieve mutual verification and session-key agreement and counter security attacks. The selection of the appropriate encryption technique relies mainly on device and sensor configurations, determining power, and energy consumption. However, IoT devices and sensors have limited power and computing capabilities because of low memory, low power, low battery, and network confinement. Therefore, the encryption techniques in an appropriate authentication protocol have to be tailored for application to the scenario. Considering the need for IoT devices and sensors, the existing research attempts to design lightweight authentication protocols based on elliptic-curve cryptography (ECC) and profitable open-key cryptography. However, in our literature analysis, we found that the existing schemes provide a secure authentication scheme with high operational costs (i.e., computation, communication, and storage costs). That is, the schemes proposed by existing research unnecessarily consume too much electrical power, storage, and computation energy for computing challenge parameters, verification processes, and key distribution and management, despite IoT devices having a limited capacity.

To overcome the problems mentioned previously, we present a novel, efficient, lightweight authentication protocol for IoT applications. Firstly, we analyzed various recent studies and identified the various factors that include an ECC point multiplication to reduce the operational cost. To ensure security strength that is equivalent or greater than the existing schemes, we efficiently optimized the usage of an ECC point multiplication operation in the proposed scheme, which helped us make the lightweight scheme and improve its performance over existing approaches. The contributions of this study are summarized as follows.

- We propose a novel lightweight authentication scheme with defensive tactics provided by the combination of tamper-proof devices, ECC technology, exclusive OR (XOR), concatenation, working bit, and administrator task involvement.
- To test the proposed scheme's performance, we implemented a standard pairing-based cryptography (PBC) [10] library on embedded devices that compute the primitive timing of the various cryptography operations included in the proposed scheme and the existing schemes. Using this primitive timing, we calculated the proposed scheme's operational cost and compared it with recent techniques. The comparison shows that the proposed scheme is lightweight in design and requires lower operational cost overhead.
- We demonstrate that the proposed scheme is secure under the elliptic curve Diffie Hellman problem (ECDHP) [11] and the computational Diffie Hellman problem (CDHP) [12]. We verify it using mathematical informal security analysis.
- Furthermore, to check the correctness of informal security analysis, we implemented the code for the proposed scheme in the widely accepted automated validation of internet security protocol and application (AVISPA) tool [13] and the ProVerif tool [14]. The result of the analysis and comparison with existing schemes demonstrates that the proposed scheme is more dominant, productive, and secure against all active and passive attacks [15], and it achieves the goal of secure design.

This paper is organized as follows. Section 2 presents the related state-of-the-art works, existing weaknesses, and preliminary mathematics used in the proposed scheme. The proposed ECC-based protocol is introduced in Section 3. Section 4 shows the cryptoanalysis of the proposed protocol by utilizing the informal security analysis and then verifying it using two automated validation of internet security protocol and application tools (i.e., AVISPA and ProVerif). In Section 5, the proposed scheme's performance and efficiency assessments are compared to the existing methods. Finally, Section 6 presents the conclusion and future studies.

2. Background and Related Works

2.1. Related Works

In this study, we analyze numerous current authentication schemes in IoT environments that are reported to have especially secure and lightweight architecture. Zhou et al. [16] presented a lightweight authentication scheme for IoT cloud architecture that is secure against password guessing attacks, insider attacks, and user anonymity attacks. They claimed that their scheme is efficient against computational-limited smart devices. Yu et al. [17] proposed a lightweight authentication scheme based on biometrics protected against replay attacks, embedded device impersonation attacks, forward secrecy, and user anonymity attacks. Xie et al. [18] enhanced Wang et al.'s [19] scheme (i.e., prone to key compromise impersonation (KCI) attacks) and introduced a secure authentication scheme for the Internet of Things. Their method can resist replay attacks, embedded impersonation attacks, and cookie-theft attacks. They also claimed that their scheme provides security against KCI attacks. Chatterjee et al. [20] proposed an ECC-based lightweight three-way authentication scheme that protects against replay attacks, forward secrecy, and known session-specific temporary information attacks. They reported that their scheme provides security with low computation and communication overheads. Yu et al. [21] introduced a secure authentication scheme for the IoT cloud environment. They claimed that their scheme offers good security features that allow their scheme to withstand replay attacks, denial of service attacks (DoS), forward secrecy, insider attacks, and user anonymity attacks, with robust performance. Sengupta et al. [22] presented a biometric-based authentication scheme. They claimed that their scheme is secure against reply attacks, password guessing attacks, DoS attacks, forward secrecy, insider attacks, and user anonymity attacks and has efficient and suitable IoT applications. Yang et al. [23] proposed an authentication scheme for information exchange in a wireless sensor network (WSN) of IoT application. Their scheme provides security against replay attacks, embedded device impersonation attacks, and user anonymity attacks. Wang et al. [24] implemented an innovative authentication system that is resistive to session key attacks, replay attacks, password guessing attacks, forward secrecy attacks, insider attacks, known session-specific temporary attacks, user anonymity attacks, and stealing of cookies. They also reported that their scheme provides optimum security capabilities with reasonable overheads for computation and communication. Wazid et al. [25] showed a biometric-based lightweight authentication scheme focused on biometrics utilizing a framework for fuzzy extractors. Their scheme offers enhanced security against replay attacks, password guessing attacks, embedded device impersonation attacks, DoS attacks, intruder attacks, and user anonymity attacks, with a low computation and communication overhead. However, as mentioned previously, the existing approaches suffer from high computation, communication, or storage costs. A more detailed description will be given in Sections 2.2 and 5.1.

Recently, Wei et al. [26] observed that most of the previous schemes (e.g., [27,28]) only provided heuristic security arguments; little attention has been paid to the formal treatment of protocol security. Most of them are vulnerable to various security loopholes. Thus, they suggested an enhanced scheme for cloud computing without using computation-expensive public-key operations. They employed a new cryptographic primitive, called an authenticated encryption scheme, to guarantee both message confidentiality and integrity and showed that their new scheme could be proven to be secure in the random oracle model.

However, this scheme has several serious defects that are overlooked, such as it cannot achieve the primary goal of "truly two-factor security", it provides no forward secrecy, and it ensures no user untraceability. Limbasiya et al. [29] cryptanalyzed Nikooghadam et al.'s scheme and found it vulnerable to replay attacks, insider attacks, and password guessing attacks. To overcome the security weaknesses, they proposed an improved scheme. Feng et al. [30] presented a replay-attack-resistant authentication scheme based on an improved challenge–response mechanism instead of the timestamp mechanism.

Nikooghadam et al. [31] cryptanalyzed Kumari et al.'s scheme [32] and claimed that the scheme is susceptible to off-line password guessing attacks. Additionally, the scheme failed to ensure user anonymity, thus causing a breach of security. To enhance security and overcome the identified security weaknesses, Nikooghadam et al. proposed an improved scheme that protects user anonymity. Later on, Aikuhlani et al. [33] proposed a secure and lightweight mutual authentication and key agreement scheme, in which the cryptographic functions were proved to be computationally lightweight and to resist some known attacks. Although these schemes were lightweight in the IoT environment, they did not prove that they were applicable to heterogeneous IoT. Dhillon and Kalra et al. [34] proposed an authentication protocol for the medical professional to access patient data for healthcare applications based on a cloud–IoT network. However, the protocol lacks user anonymity and does not provide a strong identity check. Amin et al. [35] pointed out that previous schemes (e.g., [36]) are vulnerable to various security loopholes (e.g., off-line password attacks and user impersonation attacks) and also developed a new scheme for distributed cloud computing environments. Nonetheless, Challa et al. [11] demonstrated that Amin et al.'s scheme is vulnerable to privileged-insider and impersonation attacks.

2.2. Problem Definition

As mentioned in Section 2.1, existing schemes have high computation, communication, or storage costs, despite the fact that the IoT devices and sensors are in limited environments (e.g., low memory, low power, network confinement). Therefore, current methods suffer from the following problems:

- High response time: In IoT network communication, high computing operation is performed in limited-capacity IoT devices, where high communication bit message exchanges happen between node devices, which considerably increase the response time of the message in communication [37].
- High computational overhead: Due to high computation cost, the limited-capacity IoT devices need to process mutual authentication and session key agreement requests using limited memory. In addition, in the real-time scenario of processing a high (i.e., millions or zillions) number of authentication requests between IoT devices, it consumes high processing power, which is limited in IoT devices, making IoT devices suffer from high computational overheads [38].
- Excess memory consumption: The high computational cost of recent schemes causes limited-capacity IoT devices to perform a high level of processing and operations using limited memory, strongly impacting lower down device efficiency and performance (i.e., IoT device processors require considerably increased execution time to execute large content in memory) to a certain extent [26].
- Excess electricity power consumption: The level of computing power consumption is directly proportional to electrical power consumption. The recent schemes make limited-capacity IoT devices suffer from high computational overheads, resulting in high electrical power consumption [38].
- Lack of service availability: Due to the high running cost, it is a burden to handle and process millions or zillions of authentication and communication requests in real-time; this creates a considerable time delay for service in IoT systems [38].
- Lack of performance and costly solutions: Due to high running costs, IoT devices suffer computationally because of their limited processing and storage capacities. When handling a high (e.g., millions or zillions) number of authentication requests in a huge

IoT network, this affects their performance and increases the possibility of damage to IoT devices, including additional charges for replacement and maintenance [39].

Thus, the currently existing works lack a robust authentication model to ensure the low energy utilization of cryptographic operations, which results in usability issues and strongly impacts the avoidance of the adaption of existing authentication technology in real IoT cloud scenarios. While developing a reliable and secure IoT device, not only privacy and security but also lightweight and efficient mechanisms are the major concerns. In this sense, much work has been done. Nevertheless, existing schemes either have high computation, storage, or communication costs or fail to address many of the security attributes, which results in the existing works failing to address usability problems. Thus, none of them meet the requirement of a lightweight security design. To address all those contexts, we propose a novel lightweight and secure authentication scheme in this research.

2.3. Prerequisite

To develop a deep understanding of the cryptography operations used in constructing the authentication scheme, we gave a quick review of it, as follows:

- ECC definition and property. If there exist prime number $p > 3$ and constant $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$ such that $\boldsymbol{j_1}, \boldsymbol{j_2} \in \boldsymbol{Z_p}$, where $\boldsymbol{Z_p} = \{\boldsymbol{0}, \boldsymbol{1}, \ldots, \boldsymbol{p-1}\}$, then a nonsingular elliptical curve (EC) defined as $\boldsymbol{y^2} = \boldsymbol{x^3} + \boldsymbol{j_1 x} + \boldsymbol{j_2}$ over the finite Galois field (GF(p)) is the set $\boldsymbol{E_p(j_1, j_2)}$ of the solution (x, y) $\in \boldsymbol{Z_p} \times \boldsymbol{Z_p}$ to the congruence $\boldsymbol{y^2} \equiv \boldsymbol{x^3} + \boldsymbol{j_1 x} + \boldsymbol{j_2}$ (mod p), where $\boldsymbol{4}\, j_1^3 + 27\, \boldsymbol{j_2^2} \neq \boldsymbol{0}$ (mod p), and O is called a point at infinity or a zero point. To ensure a nonsingular solution for the equation $\boldsymbol{x^3} + \boldsymbol{j_1 x} + \boldsymbol{j_2} = \boldsymbol{0}$, it is most important that $\boldsymbol{4}\, \boldsymbol{j_1^3} + 27\, \boldsymbol{j_2^2} \neq \boldsymbol{0}$ (mod p). In contrast, $\boldsymbol{4}\, \boldsymbol{j_1^3} + 27\, \boldsymbol{j_2^2} = \boldsymbol{0}$. (mod p) indicates that the elliptical curve is singular. Let us consider two points on the elliptical curve (i.e., $P = (\boldsymbol{x_P}, \boldsymbol{y_P})$, $\boldsymbol{Q} = (\boldsymbol{x_Q}, \boldsymbol{y_Q}) \in \boldsymbol{E_p(j_1, j_2)}$). Then, $\boldsymbol{x_Q} = \boldsymbol{x_P}$ and $\boldsymbol{y_Q} = -\boldsymbol{y_P}$, when $P + Q = O$. Additionally, $P + O = O + P = P$, for all P $\in \boldsymbol{E_p(j_1, j_2)}$. According to Hesse's theorem, if #E denotes the number of points on $\boldsymbol{E_p(j_1, j_2)}$, then it must satisfy the condition $p + 1 - 2\sqrt{\boldsymbol{p}} \leq$ #E $\leq p + 1 + 2\sqrt{\boldsymbol{p}}$. This condition will also be understood as there is point p on $\boldsymbol{E_p(j_1, j_2)}$ over Z_p, where $\boldsymbol{E_p(j_1, j_2)}$ forms a commutative or an abelian group under addition modulo $\boldsymbol{p}$ operation. The ECC operation explanation is as follows: ECC addition. Let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be two points on the $\boldsymbol{E_p(j_1, j_2)}$. Then, we calculate $\mathbf{C} = (\boldsymbol{x_C}, \boldsymbol{y_C}) = $ A + B as $\boldsymbol{x_C} = (\boldsymbol{\lambda^2} - \boldsymbol{x_P} - \boldsymbol{x_Q})$ (mod p), $\boldsymbol{y_C} = (\boldsymbol{\lambda}(\boldsymbol{x_p} - \boldsymbol{x_C}) - \boldsymbol{y_p})$ (mod p), where $\boldsymbol{\lambda} = \frac{\boldsymbol{y_Q} - \boldsymbol{y_P}}{\boldsymbol{x_Q} - \boldsymbol{x_P}}$ (mod p), if P $\neq$ Q or $\boldsymbol{\lambda} = \frac{\boldsymbol{3x_p^2} + \boldsymbol{c_1}}{\boldsymbol{2y_p}}$ (mod p), if P = Q. ECC multiplication ($\times$ **or**). It is also called scalar multiplication, which is done by repeated additions. For example, $\boldsymbol{3P} = \boldsymbol{P} + \boldsymbol{P} + \boldsymbol{P}$ where $P \in \boldsymbol{E_p(j_1, j_2)}$. The ECC multiplication also defines two interesting properties, as follows:
  - Property 1: Elliptic Curve Discrete Logarithm Problem (ECDLP). Let n be a positive integer with two-point P, Q on the curve $\boldsymbol{E_p(j_1, j_2)}$, such that $Q = n \times P$. Then, it is computationally infeasible for a big prime p to derive n in polynomial time, even if P and Q are known. Here, n is called as a scaler and $n \times P = P + P + \ldots + P$ (n times) is known as ECC point or scalar multiplication.
  - Property 2: Computational Diffie Hellman Problem (CDHP). For a, b $\in$ [1,$n-1$], given P, aP, and bP, it is difficult to compute abP.
- A one-way cryptographic hash function. It is a deterministic one-to-one function h:{0, 1}$^* \rightarrow \{\boldsymbol{0},\boldsymbol{1}\}^{\boldsymbol{n}}$, which produces a fixed-length n-bit output string y $\in \{\boldsymbol{0},\boldsymbol{1}\}^{\boldsymbol{n}}$ (i.e., called hash digest) on an arbitrary input string x {0, 1}*, such that $y = h(x)$. The small change in x can give completely different unique results in y (i.e., message digest). One of its examples is the SHA-256 hashing algorithm. It has the following property: Collision-Resistant One-Way Hash Function (h()). Let $\boldsymbol{Adv_{(A)}^{Hash}}$ (t) represent the advantage of an adversary A in finding a collision. Then, $\boldsymbol{Adv_{(A)}^{Hash}}$ (t) = Pr [$(\boldsymbol{a_1}, \boldsymbol{a_2}) \in_R$ A: $\boldsymbol{a_1} \neq \boldsymbol{a_2}$, h($\boldsymbol{a_1}$) = h($\boldsymbol{a_2}$)], where Pr[E] denotes the probability of an event E, and $(\boldsymbol{a_1}, \boldsymbol{a_2}) \in_R$ A indicates that $(\boldsymbol{a_1}, \boldsymbol{a_2})$ is randomly selected by A. By an (ψ, t)-adversary A attacking

the collision resistance of h(·), it means that the runtime of A is at most t and that $\mathbf{Adv}_{(\mathbf{A})}^{\mathbf{Hash}}(t) \leq \psi$.

We used ECDLP, CHDP, and collision-resistant one-way hash function property (h()) in the design of the secure authentication protocol in the proposed section.

3. Proposed ECC-Based Lightweight Authentication Scheme

As mentioned earlier, the existing schemes have limitations because of high operational costs. Therefore, to overcome these issues, we propose a lightweight, secure authentication scheme for IoT cloud- and server-based applications by optimizing the efficient use of multiplication operations (.) (i.e., it includes ECDLP and CDHP), a hash function (h), concatenation (||), and XOR ($\oplus$). The proposed scheme consists of the following two phases: the registration phase and the password-authentication phase, which includes the session-key-distribution step. Note that we propose a scheme for tamper-proof embedded devices and server-based applications. The security of tamper-proof devices ensures that the owners of the devices are unable to access any kind of data stored in the embedded devices [40]. In addition, to launch an attack on successfully embedded devices, an attacker needs special equipment that costs even more than embedded devices; thus, an attacker does not have economic incentives to mount an attack on tamper-proof devices [41]. Therefore, tamper-proof embedded devices improve the security strength of the proposed protocol. The proposed scheme is depicted in Figures 1–3. The notation shown in Table 1 will be used throughout the paper.

Table 1. Notation guide.

Notation	Description
P_{SWi}	Secret password of the embedded device
P_{ke}	Public key of the embedded device
P_{ks}	Public key of the server
I_i	Hashed embedded device ID
CID_i	Server-generated embedded device ID
CK	Session cookie
s	Server private key
E_{ks}	Server database table encryption key
X_i, Y_i, X_j, Y_j, C	Challenges
x_{i1}, x_{j1}	Ephemeral secrets
E_t	Expiration time of the cookie
A, T	Equations
S_k	Session key
ID_{TS}	Unique secret identity of server TS
E_{ke}	Tamper-proof device storage table encryption key

3.1. Initialization Phase

When beginning the registration process, a trusted server (TS) chooses an elliptical curve equation E_p: $y^2 \equiv x^3 + m_1 x + m_2 \pmod{p}$ over Z_p, where Z_p ($p > 2^{160}$) is the finite range of the group. A TS selects two fields, respectively, j_1 and j_2, where m_1 and m_2 must obey condition $4\ m_1^3 + 27\ m_2^2 \pmod{p} \neq 0$. Let G be the E_p, *the* base point with a prime order n ($n > 2^{160}$), and O be the point at infinity, such that $n.G = O$. Then, TS chooses arbitrary nonce (s) as its secret key.

3.2. Registration Phase

In the beginning, to make a user legal, before selling a product to the owner, the company of device ED_i must register itself with the trusted third party through a secure channel. Figure 1 shows the details of the registration phase.

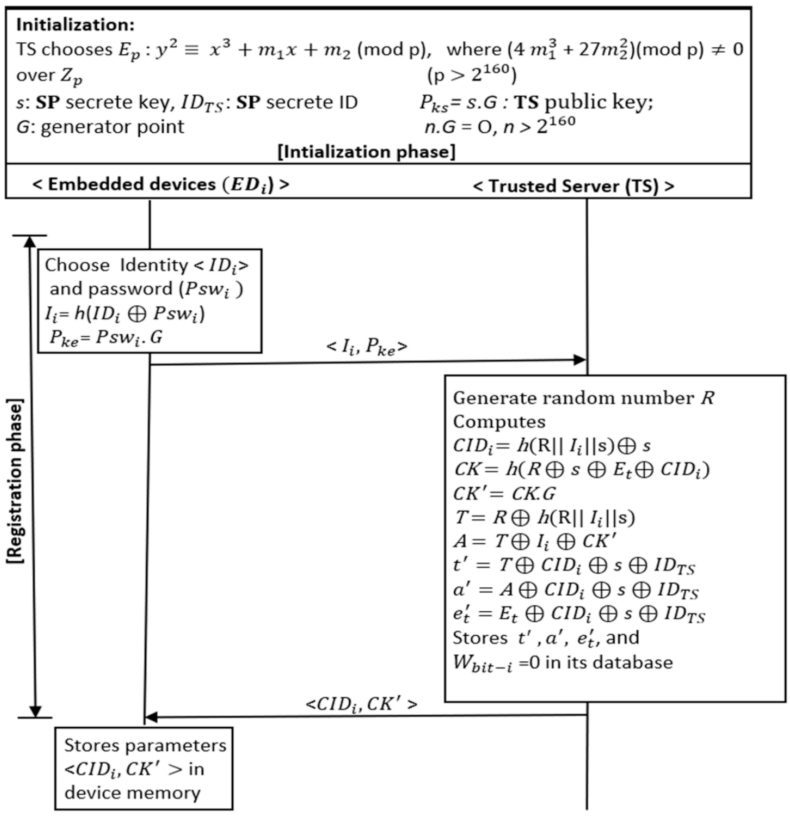

Figure 1. Proposed elliptic-curve cryptography (ECC)-based registration scheme.

- Step 1. Initially, the device (ED_i) registers an authenticated identity (ID_i) with a trusted server (TS); to that end, the device chooses a password (Psw_i) for computing hashed identity $(I_i = h(ID_i || Psw_i))$ that will protect the device from a user or device anonymity attack.
- Step 2. Then, ED_i sends the registration request $< I_i, P_{ke} >$ to TS through a trusted channel, where $P_{ke} = Psw_i.G$ is a public key of ED_i, which ensures the security of Psw_i by using the concept of ECDLP.
- Step 3. TS checks whether I_i has been registered. If I_i has not been registered, TS selects a random number R and computes a server-generated unique identity ($CID_i = h(R||I_i||s) \oplus s$) for uniquely identifying ED_i. The security of CID_i is well protected in TS by using ECDLP ($CID_i' = CID_i.G$). Then, TS computes a unique session cookie $(CK = h(R||s||E_t||CID))$ for ED_i. The security of CK is protected in TS by using ECDLP ($CK' = CK.G$). Then, TS computes challenge $T = R \oplus h(R||I_i||s)$, hides it inside another challenge ($A = T \oplus I_i \oplus CK'$) using I_i and CK'. Finally, to protect the security of challenges T and A, TS hides them inside $t' = T \oplus CID_i \oplus s \oplus ID_{TS}$, $a' = A \oplus CID_i \oplus s \oplus ID_{TS}$, *and* it also hides expiration time (E_t) of the cookie inside $e_t' = E_t \oplus CID_i \oplus s \oplus ID_{TS}$. Subsequently, TS stores parameters $< t', a', e_t' >$ on the server database, sets the working bit as $W_{bit-i} = 0$, and sends $< CID_i, CK' >$ to ED_i through a secure, trusted channel.

Whenever the device (ED_i) communicates with TS, the working bit W_{bit} is set to one for the corresponding connected communication; otherwise, and after the termination of the communication, it is set to zero. The servers encrypt the table using their personal secret key $E_{ks} = s.ID_{TS}.G$. The security of key (E_{ks}) is well protected by the concept of ECDLP and CDHP. The device security (ED_i) is well protected by its tamper-proof nature. Note that before handing over devices to their owners during the registration process, the company performs the registration process through a secure channel (e.g., wolfSSL [42]). Once the device is successfully registered to TS, the company hands over the registered

tamper-proof product to the owner. Then, every time communication is initiated with *TS*, the owner's device is authenticated by *TS* through an open communication channel, and then, after successful authentication, the device sends an encrypted message to *TS* using their session key (S_k).

3.3. Authentication Phase

Initially, each registered participant client device is required to be authenticated with *TS* before starting communication. Therefore, the authentication of the registered devices with *TS* proceeds as follows (depicted in Figure 2):

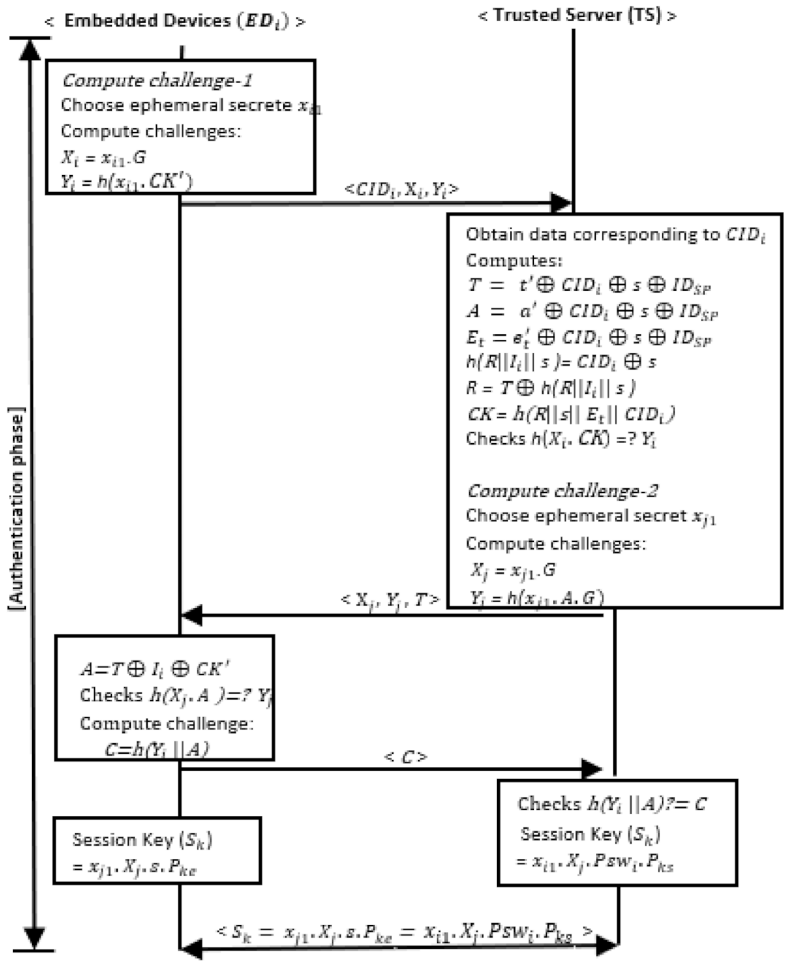

Figure 2. Proposed ECC-based login scheme.

- Step 1. $ED_i \rightarrow TS: < CID_i, X_i, Y_i >$. The legitimately registered device (ED_i) randomly selects an ephemeral secret $x_{i1} \in Z_p$ and computes challenges $X_i = x_{i1}.G$ using an ECC point multiplication, hides x_{i1} inside $Y_i = h(x_{i1}.CK')$, and, finally, sends the challenge $< CID, X_i, Y_i >$ to TS through a secure channel. The security of x_{i1} inside both X_i and Y_i is protected by the concept of ECDLP and CDHP.
- Step 2. $TS \rightarrow ED_i: < X_j, Y_j, T >$. After receiving the challenge, *TS* recomputes and extract parameters $(T, A, E_t, h(R||I_i||s), R, CK)$ from known parameter $< t', e_t', a', CID_i, s, ID_{TS} >$ of the *TS* (i.e., it computes $T = t' \oplus CID_i \oplus s \oplus ID_{TS}$, $A = a' \oplus CID_i \oplus s \oplus ID_{TS}$, $E_t = e_t' \oplus CID_i \oplus s \oplus ID_{TS}$, $h(R||I_i||s) = CID_i \oplus s$, $R = T \oplus h(R||I_i||s)$, $CK = h(R||s||E_t||CID_i)$), and then *TS* computes $h(X_i.CK)$(i.e., $= h(x_{i1}.G.CK) = h(x_{i1}.CK')$) and checks whether $h(X_i \times CK) = ? Y_i$ (i.e., it checks whether the left value is equal to the right value). If the deduction from both sides is unsuccessful, the session is terminated; otherwise, proceed to the further steps. Subsequently, *TS* randomly selects an ephemeral secret $x_{j1} \in Z_p$ and then computes challenges

$X_j = x_{j1}.G$ by using an ECC point multiplication, hides x_{j1} inside $Y_j = h(x_{j1}.A.G)$, and, finally, sends challenges $< X_j, Y_j, T >$ *to* ED_i through an open channel. The security of x_{j1} inside both X_j and Y_j is protected by the concept of ECDLP and CDHP.

- Step 3. Computation of the session key $S_k = x_{i1}.X_j = x_{i1}.x_{j1}.G = x_{j1}.X_i$: After receiving the challenges $< X_j, Y_j, T >$ from TS, ED_i recomputes $A = T \oplus I_i \oplus CK'$ and checks whether $X_j.A$ (i.e., $h(x_{j1}.A.G)$) $=?Y_j$. If the deduction from both sides is unsuccessful, the session is terminated; otherwise, proceed to the further steps. Subsequently, the user ED_i computes the last challenge $C = h(Y_i||A)$ and sends challenges $< C >$ to TS through an open channel. After receiving the challenges, TS recomputes and checks whether $h(Y_i||A\ ) =?C$. If the deduction from both sides is unsuccessful, the session is terminated; otherwise, proceed to the further steps. Finally, after successful deduction, the TS computes the session key $(S_k = x_{j1}.X_i s.P_{ke})$ and ED_i computes the session key $(S_k = x_{i1}.X_j.\ Psw_i.P_{ks})$.

3.4. Password-Change Phase

The proposed scheme allows a legitimated embedded device ED_i to change the password periodically, thereby ensuring security. The embedded devices first perform authentication and authorization to change the password and prove their genuineness. Once device ED_i successfully performs mutual authentication with server TS, device ED_i must obey the following steps for password change, as depicted in Figure 3:

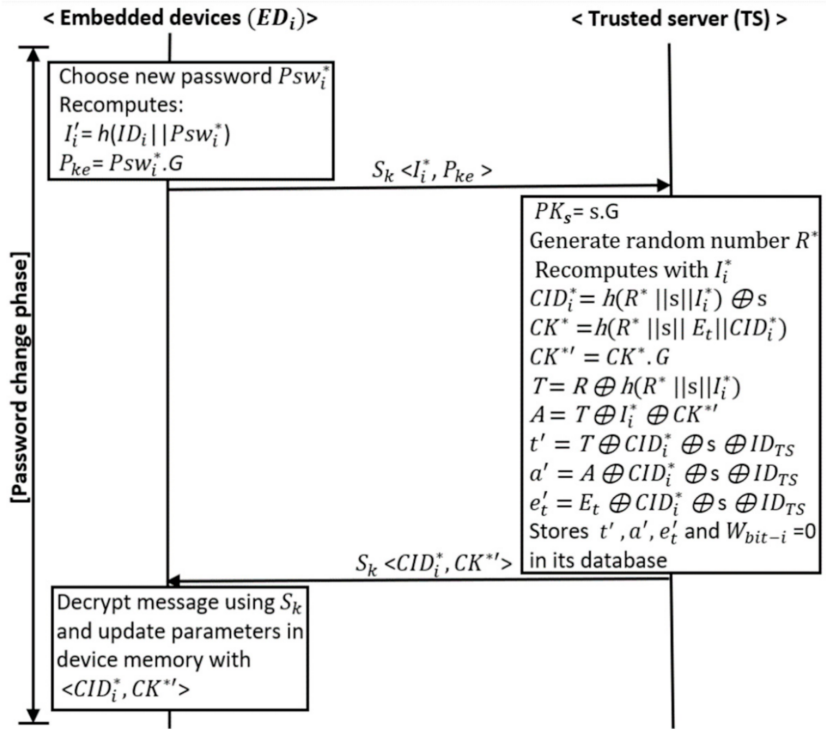

Figure 3. Password-change phase.

- Step 1. $ED_i \rightarrow TS: < I_i^* >$. The authorized legitimated embedded device ED_i selects a new password Psw_i^* and then recomputes the hashed identity $(I_i^* = H(ID_i\ ||Psw_i^*\ ))$ and public key $(P_{ke}' = Psw_i^*.G)$. Subsequently, it sends the updated $< I_i^*,\ P_{ke}' >$ to TS through a trusted public channel by using the session key S_K.
- Step 2. $TS \rightarrow\ ED_i: < CID_i^*, CK'^* >$. TS receives updated parameters $< I_i^*,\ P_{ke}' >$, and then TS selects a random number R^* and then recomputes all parameters $(CID_i{}^* = h(R^*||I_i^*||s) \oplus s, CID_i{}'^* = CID_i{}^*.G,\ CK^* = h(R^*||s||E_t||CID_i{}^*), CK^{*'} = CK^*.G, T = R^* \oplus h(R^*||I_i^*||s), A = T \oplus I_i^* \oplus CK^{*'}, t' = T \oplus CID_i{}^* \oplus s \oplus ID_{TS}\ , a' = A \oplus CID_i{}^* \oplus s \oplus\ ID_{TS},\ and\ e_t'\ =\ E_t \oplus CID_i{}^* \oplus\ s \oplus ID_{TS})$, which are mentioned in Section 3.2.

Subsequently, TS stores the parameters $< t', a',\ e_t'>$ in the server database and sends the updated $< CID_i{}^*,\ CK^{*\prime} >$ to the server TS through an open trusted channel by using the session key (S_K).

4. Cryptoanalysis of the Proposed Protocol

In this section, we present the essential security requirements of the proposed scheme. Then, we analyze the security strength of the proposed scheme using informal security analysis (i.e., mathematical verification). During the analysis, we observed that the proposed scheme prevents various security attacks. Subsequently, we compared the proposed scheme with existing approaches for more clarity, following which we highlighted the weaknesses that the proposed scheme removes from the existing schemes. Furthermore, we verified the correctness of our mathematical explanation by using AVISPA and ProVerif simulation tools.

4.1. Informal Security Analysis

For ensuring the security of the proposed scheme, we used the collision-free one-way hash function and two hard problems: the ECDHP and the CDHP. We analyze and summarize the main security benefits of our proposed scheme. The proposed scheme targets optimized performance at a lower cost. Therefore, in this subsection, we present how the proposed scheme is secure against various cryptographic attacks that affect the performance of the authentication scheme. Furthermore, we compare the proposed scheme with the existing approaches by utilizing security attributes to explain its efficiency, as follows:

- Replay attacks: In this attack, an adversary may impersonate a legitimate user by reusing the message $< CID_i,\ X_i, Y_i >$, obtained from a previous protocol run, and transferring it to the TS. After receiving the login request, TS computes and verifies whether $h(X_i.CK) =?Y_i$ and sends back new challenges $< X_j, Y_j, T >$ to ED_i. However, after impersonating the messages $< X_j, Y_j, T >$ sent to ED_i , the attacker would be unable to compute $A = T \oplus I_i \oplus CK'$ without knowing I_i and CK'. Notably, both CK' and I_i are neither sent through any messages over public channels nor can they be acquired from the embedded device (ED_i) because of its tamper-proof design. Thus, it cannot verify the challenge $C_i = H(Y_i||A)$. Without the knowledge of the secret key x_{i1} and x_{j1} of the device and server, the attacker cannot compute the valid session key $(S_k = x_{i1}.X_i.Psw_i.P_{ks} = x_{j1}.X_i.\text{s}.P_{ke}$). Additionally, the security of the session key is well protected by ECDLP and CDHP. Therefore, it almost impossible to extract the private key $< Psw_i,\ s >$ of the device and the server in polynomial time. Hence, the proposed scheme is secure against replay attacks.
- Password-guessing attacks (or confidentiality): In the proposed scheme, an embedded device password psw_i is stored in the form of a password generator (i.e., public key $(E_{ke} = psw_i.G)$) and wrapped in the form of $I_i = H(ID_i||psw_i)$. Consequently, the attacker cannot guess the password psw_i without knowing I_i and P_{ke}. Therefore, the proposed scheme maintains the security of the password by using ECDLP and a hash function ($h()$). Notably, I_i of ED_i is neither sent through any messages over an open public channel nor can it be acquired from the embedded device (ED_i) because of its tamper-proof design (i.e., nor is it stored in the ED_i or TS). Therefore, the proposed scheme prevents password-guessing attacks.
- Embedded device impersonation attacks (or key compromise impersonation attacks): In this case, an attacker impersonates real embedded devices (ED_i) by replaying a previously intercept message to the server (TS). However, an attacker will still lack $ED_i{}'s$ secret parameters $< psw_i,\ I_i,\ CK' >$ because the secret parameters are protected due to the tamper-proof design of ED_i. In addition, psw_i of ED_i in $I_i = H(ID_i||psw_i)$ and in $E_{ke} = psw_i.G$ is protected by hash function $h()$ and ECDLP, respectively. Therefore, the attacker unable to compute $A = T \oplus I_i \oplus\ CK'$ correctly without knowing parameters $< I_i, CK' >$. This will result in an incorrect deduction $h(X_j.\text{A}) = ?\ Y_j$, which will lead

to the termination of the session. Furthermore, an attacker will be unsuccessful in extracting Psw_i from the past session key $(S_k = x_{i1}.X_i.Psw_i.P_{ks})$, whose security is protected by ECDLP and CDHP. Therefore, an attacker is unable to compute a session key S_k for a current session without knowing Psw_i. Therefore, for the above reasons, an attacker will fail to launch embedded device impersonation attacks on ED_i.

- DoS attacks: To prevent the proposed scheme from DoS attacks, the server (TS) terminates the login session if the number of incorrect attempts to enter CID_i reaches the maximum limit. However, the login request will be continued as soon as the correct CID_i is entered. Furthermore, in the login phase, assume the adversary replaces message $< CID_i, X_i, Y_i >$ with $< CID_i', X_j', Y_j' >$ by randomly selecting the elliptical curve point x_{i1}' and sending it back to TS; however, TS computes and compares the previous value with the received Y_i' (i.e., $h(X_i.CK)) =? Y_i$. If TS finds a difference between the values, it terminates the protocol with a failure message to the user. Therefore, the proposed scheme is secure for DoS attacks. The resource optimization of the IoT cloud system can be influenced by the success of a major attack (i.e., DoS and replay attacks). The proposed scheme is, however, protected against both attacks. Therefore, the proposed scheme ensures the security of the performance of the proposed scheme.
- Many logged-in users attacks: Suppose an adversary somehow manages to get a legally embedded device's credentials $< CID_i', CK' >$, along with the secret identity I_i. Subsequently, the adversary tries to communicate with the server by impersonating ED_i. However, in the proposed scheme, out of all-knowing the valid credential, only one legal ED_i can communicate with TS at a time as, every time they communicate, TS sets a working bit W_{bit-i} equal to one for the corresponding communication with ED_i after successful authentication and stores the working bit in its database. Every time, before communication, the receiver TS will check W_{bit-i} before establishing a connection with the requested ED_i. Furthermore, the receiver TS can deny all requests if $W_{bit-i} = 1$, representing the existing ED_i is still communicating with it.
- Server impersonation attacks: Assume the scenario where the phase of authentication of ED_i is impersonated by an adversarial server. An adversarial server impersonates and receives the parameters $< CID_i , Xi , Yi >$ from ED_i. Then, the adversary server randomly chooses parameters $< X_j', Y_j', T' >$ and sends it back to the ED_i. After receiving parameters $< X_j', Y_j', T' >$, the ED_i computes the factor and checks $h(X_j'.\mathrm{A}) = ? Y_j'$. However, the equivalence comes out wrong. This is because an attacker randomly computes challenge T as T' and is completely unaware of challenging A. In addition, attackers are unable to compute a session key. This is due to the attacker being unable to extract a private server key (s) or a device password (psw_i) from the past session key due to security protection by ECDLP and CDHP. Furthermore, the security of s is protected in the server $(E_{ks} = \mathrm{s}.ID_{TS}.G)$ by ECDLP and CDHP. Therefore, the proposed scheme is secure against server impersonation attacks.
- Forward secrecy attacks (or key freshness): Even if the private key of both the ED_i and the server is compromised by some other means, the confidentiality of the recently established session keys ought not to be affected. Suppose an adversary somehow discovers ED_i's password (Psw_i) and TS's secret key (s); thus, the adversary determines other components from the message. However, the adversary cannot derive the session key $(S_K = x_{i1}.X_j.\ Psw_i.P_{ks} = x_{j1}.\mathrm{X_i}.s.P_{ke})$. This is because, in order to compute it, the adversary must determine x_{i1} and x_{j1} from X_i and X_j, which seems to be computationally infeasible because of the complexity of ECDLP. Therefore, even if the present session key (S_K) is leaked, the adversary cannot determine all the past session keys, as the session key also depends upon the random secrets (x_i and x_j), whose security is protected by the concept of ECDLP; the security of S_K is protected by the concept of ECDLP and CDHP. Hence, the proposed scheme is secure against forward secrecy attacks.

- Privileged insider attacks: In the proposed scheme, during the registration of a device, ED_i sends $I_i = H(ID_i||psw_i)$, instead of sending the password (psw_i), securely over the trusted channel (e.g., wolfSSL). After successful registration, the product is handed over to the owner by the company. Since the owner cannot extract any data $< psw_i,\ I_i,\ CID_i,\ CK' >$ from device ED_i due to its tamper-proof design, the advisor of TS cannot acquire the secret psw_i because it is ensured by $ED_i's$ identity and $h()$. Thus, the privileged insider from ED_i will fail to impersonate the legitimate ED_i. Hence, the proposed scheme is secure against ED_i insider attacks. However, a TS adversary can steal information from server TS for the device ED_i. Thus, in such a case, the proposed scheme is less secure against privileged insider attacks.
- Known session-specific temporary information attacks: After successful authentication, both the communicating ED_i and TS compute the session key $(S_K = x_{i1}.X_j.\ Psw_i.P_{ks} = x_{j1}.X_i.s.P_{ke})$, whose security is protected by ephemeral secrets x_{i1} and x_{j1}. Suppose that an adversary somehow discovers the ephemeral secret x_{i1} or x_{j1}. In this case, the adversary cannot derive the session key (S_K) only with the knowledge of a single ephemeral secret since the security of S_K still depends on Psw_i or s. Therefore, to derive the session key, the advisor must determine $< Psw_i, s >$ from the past session key (S_k), which, in turn, seems to be computationally infeasible because of the difficulties in solving CDHP and ECDLP for pairs, which are difficult to comprehend using a polynomial-time algorithm. Thus, the proposed scheme is securely infeasible to a known session-specific temporary information attack.
- Attacks on user anonymity: Device anonymity implies that an attacker cannot discover the device's concealed identity (ID_i) by using the transmitted messages during the login and authentication phases. Here, a company that is selling device ED_i does the registration process itself and hides ID_i inside identity (I_i) using the hash function $h()$. In addition, during the registration process, the company does the registration process with the server by sending $< I_i,\ psw_i >$ through a secure channel. Then, the company hands over the tamper-proof product to the owner. Therefore, the security of I_i is well protected by the tamper-proof design of ED_i. Furthermore, the security of I_i is well secured in CID_i using a secret key (s) and a random number (R) with the assistance of the hash function $h()$. Moreover, both s and the R are neither sent through any message nor stored in the ED_i and TS in the plaintext format. Hence, the proposed scheme is securely infeasible to attacks on user anonymity.
- Database attacks: In this case, if an attacker makes a server database attack, the attacker will be unsuccessful in breaking the table of the server database as the security of the server and the database is protected by an encryption key $E_{ks} = s.\ Id_{Ts}.G$, whose security is protected by ECDLP and CDHP. Therefore, an attacker will be unable to extract either s or Id_{Ts} from E_{ks}. In addition, even if the attacker gets it by any means, the attacker still unable to extract Id_{Ts} from E_{ks}. Furthermore, if the encryption key of the server is compromised somehow, the attacker will be unable to extract security parameter $< T,\ A,\ E_t >$ from $< t',\ a',\ e'_t >$. This is due to parameter security is still protected by Id_{Ts} and s. Therefore, the proposed scheme is preventive against database attacks.
- Cookie-theft attacks: In the proposed scheme, the session cookie (C_k) is stored and sent in the form of $CK' = h(R||s||E_t||CID_i).G$, (i.e., an ECC point multiplication) in the embedded device (ED_i). Therefore, it is very difficult to extract CK from CK' because of the complexity of ECDLP. In addition, because of the tamper-proof design of ED_i, it is impossible for an attacker to extract CK' from it [43]. In addition, CK' is sent through a secure, trusted channel during the registration step, which we have explained in Section 3.2. Consequently, the attacker cannot get the cookie (CK'). Therefore, the proposed scheme is secure against cookie-theft attacks.
- Server-spoofing attacks: In these attacks, an adversary may masquerade as a server (TS) to uncover the secret credential of a device (ED_i). $ED_i's$secret credential CID_i comprises the hashing of the random secret (R), the secret identity (I_i), and the

server secret (s). In addition, CID_i is stored in the TS's encrypted database (i.e., $E_{ks} = s.\ Id_{Ts}.G$) in the form of $CID_i' = CID_i.G$. Since the security of E_{ks} is protected by ECDLP and CDHP, it is impossible for an attacker to extract CID_i from $CID_i' = CID_i.G$. Thus, an attacker cannot get $ED_i's$ secret credentials CID_i by any means due to the complexity of solving ECDLP and CDHP and the device's tamper-proof nature. Therefore, the proposed scheme is secure against server-spoofing attacks.

- Stolen-verifier attack: If the attacker somehow discovers a smart card/smart device of server TS, the attacker could launch a power-analysis attack to uncover the secret information stored inside. However, in the proposed scheme, during the registration phase, TS stores $t' = T \oplus CID_i \oplus s \oplus ID_{TS}$, $a' = A_i \oplus CID_i \oplus s \oplus ID_{TS}$ and $e_t' = E_t \oplus CID_i \oplus s \oplus ID_{TS}$ against CID_i. Even if the attacker somehow steals those records, the attacker cannot perform the malicious activity because he/she would be unable to access the plain text (T, A, E_t) as the record is protected using the secret key (s) and ID_{TS} of TS and CID_i. The security of s and ID_{TS} in $E_{ks} = s.\ Id_{Ts}.G$ is well protected by the concept of ECDLP and CDHP. In addition, the attacker cannot create a substantial login solicitation to pass the verification stage without knowing CK' as it is not stored in the server's database. Furthermore, cookie computation, i.e., $CK' = CK.G$, depends on the correct computation of $CK = h(R||s||E_t||CID_i)$. Without knowing the server's secret key (s), the attacker cannot compute a substantial cookie (CK) and, hence, cannot make a legitimate login request. In addition, it is impossible for an attacker to extract information stored in ED_i due to its tamper-proof design. Therefore, the proposed scheme can withstand stolen-verifier attacks.
- Man-in-the-middle attacks: Mutual authentication prevents man-in-the-middle attacks. We verified the mutual authentication of the proposed scheme using the ProVerif tool in Section 4.3. Thus, the proposed scheme successfully supports mutual authentication between ED_i and TS. Consequently, the proposed scheme is secure against man-in-the-middle attacks.
- Brute-force attacks: To launch a brute-force attack, an attacker must extract the security parameters X_i, Y_i, X_j, Y_j, and T from the transmitted messages. However, even if the attacker succeeds in extracting the parameters, he/she cannot determine the password (psw_i) or the server's secret key (s), which is obscure and protected by the concepts of ECDLP and CDHP (which we mentioned earlier). Furthermore, there is no chance to speculate the random numbers (x_{i1} and x_{j1}) because of the protection offered by ECDLP and CDHP (i.e., this has already been explained in the section on known session-specific temporary information attacks). Therefore, the proposed scheme can resist brute-force attacks.

Table 2 summarizes the comparison of security attacks on the existing approaches and the proposed scheme. We selected some security attributes (i.e., usually protecting the security of an authentication protocol) for comparison. The comparison demonstrates that the proposed scheme is free from all the shortcomings in the existing schemes.

Table 2. Comparison by types of security attack.

Attacking Scenarios	Zhou et al. [16]	Yu et al. [17]	Xie et al. [18]	Chatterjee et al. [20]	Yu et al. [21]	Sengupta et al. [22]	Yang et al. [23]	Wang et al. [24]	Wazid et al. [25]	Proposed
Reply attack	V	P	P	P	P	P	P	P	P	P
Password guessing attack	P	V	V	V	V	P	V	P	P	P
Embedded device impersonation attack	V	P	P	V	V	V	P	V	P	P
DoS attack	V	V	V	V	P	P	V	V	P	P
Many logged-in user attack	V	V	V	V	V	V	V	V	V	P
Server impersonation attack	V	V	V	V	V	V	V	V	V	P
Forward secrecy	V	P	V	P	P	P	V	P	V	P
Insider attack	P	V	V	V	P	V	V	P	P	P
Known session -specific temporary information attack	V	V	V	P	V	V	V	P	V	P
Attack on user anonymity	P	P	V	V	P	P	P	P	P	P
Database attack	V	V	V	V	V	V	V	V	V	P
Cookie theft attack	V	V	P	V	V	V	V	P	V	P

V: vulnerable; P: protected.

4.2. Formal Security Validation Using the AVISPA Tool

In this section, we perform the simulation of the proposed scheme by utilizing the AVISPA tool and ensuring that the proposed scheme is secure against both man-in-the-middle and replay attacks [44]. The AVISPA tool is a push-button tool for performing the automated validation of internet-security-sensitive protocols and applications. It has become widely accepted for formal security verification in recent years [13]. The AVISPA tool is coded in one of the power languages (i.e., high-level protocol specification language (HLPSL)). This language comprises the role that represents each participating activity. The role in a scenario is separate/different from that in other scenarios. The role receives primary information from a parameter that communicates with other roles through channels. Furthermore, the HLPSL protocol is translated into intermediate format specification using an HLPSL2IF translator. The input is given to one of the four back ends (i.e., on-the-fly model-checker (OFMC), tree automata based on automatic approximations for the examination of security protocols, constraint-logic-based attack searcher (CL–AtSe), and SAT-based model-checker) to create yield.

We implemented the proposed scheme on a SPAN Ubuntu 10.10 virtual machine with RAM = 2048 Gb. This experimental setup, installed on a Windows 10 operating system (OS) with an Intel Core i5–8500 and a 6–core 3.10 GHz CPU, was provided by the service provider. The AVISPA simulation used SPAN software to assess the security strength of the proposed protocol against both active and passive attacks using the AVISPA toolset. The numerous basic types were used to define specific criteria for the roles. For example, agent—the agent determines the principle of the attacker, receives the data, analyses it, and provides recommendations for the protocol being designed by correctly defining the state of whether the protocol is in a secure or dangerous state. Different basic types are utilized for defining the specifications of each role. Some of them are as follows: (1) *agent*: it defines the principal name of the intruder using a special identifier *i*, (2) *public_key*: it represents the public key, (3) *symmetric_key*: it represents the key used for encryption, (4) *const*: it defines the constant declared in the roles, (5) *text*: it represents the nonce that is always fresh and unique, and it secures the message from an attacker, (6) *function*: it represents the irreversible one-way hash cryptography function of the type hash_func used for modeling, and (7) *nat*: it represents the natural number in a no-message context.

The AVISPA tool receives input as a designed protocol, analyzes it, and precisely generates the output by portraying the state of whether the protocol is in a safe or an unsafe state. The channel is used for communication, which is supposed to be controlled by the Dolev–Yao attacker. This means that the intruder is modeled using the Dolev–Yao model, with the possibility that the intruder might assume a legitimate role in a protocol run. The session role defines all the basic roles. The role of environment is a top-level role, and it is the beginning point for the execution; furthermore, it instantiates a session role, utilizing distinctive basic roles to simulate various possible scenarios. Finally, in the goal section, according to our prerequisites of the designed protocol, we define all the necessary and sufficient goals. While writing the code in AVISPA, we wrote two primary roles: the first role was for the user devices and the second for the server. Subsequently, we wrote another three roles: the first role was for the session, the second for the environment, and the third for the goal. The last three roles represent the user devices and the second represents the server. Subsequently, we wrote another three roles: the first role was for the session, the second for the environment, and the third for the goal. The last three roles represent the execution environment of the first two roles. Figure 4 depicts the specific role performed by the agent. Upon receiving the start signal, the device U (i.e., ED_i) updates its state from 0 to 1. This state is expressed ($Pass1'$, Ga') > (i.e., <I_i, P_{ke}>) securely to the server through a secure channel (*SND*); we call this the registration phase. The transmission channel < *SND, RCV* >, which is an unreliable channel, is used for the message transmission of the Dolev–Yao-type threat model; it enables an attacker to modify or delete the contents of transmitted messages. Afterward, in response, U receives the message <Cid', Ck'> (i.e., <CID_i, CK'>) from the server securely with the help of the secure RCV channel. During the

login phase, U sends a message <Cid', $P1'$, $P2'$ > (i.e., <CID_i, X_i, Y_i>) to the server via the public open channel (see Figure 5).

```
role device(U, S: agent,
            K, Ea: symmetric_key,
            Hash : hash_func,
            SND, RCV: channel (dy))

played_by U
def=

local
      State                  :nat,
      Un1, Pass1, Cid, Ck, N1, Ga, Ck2 :text,
      P1, P2, Ti, P3,  P4, Da, Ai, P44, Vi :text

      const e_m, g_m, h_m, a_m, b_m, d_d:protocol_id
      init State :=0
transition
 1. State = 0 /\ RCV(start)  =|>
    State':= 2 /\ Un1':= new()
               /\ Pass1':= new()
               /\ Ga' := new()
               /\ SND({Hash(Un1'.Pass1').exp(Pass1',Ga')}_Ea.S)
               /\ secret(Pass1', g_m, {U,S})
               /\ secret(Un1', h_m, {U,S})
               /\ secret(U, S, seq1, Hash(Un1'.Pass1').exp(Pass1', Ga))
 2. State = 2 /\ RCV({{Cid'.Ck'}_K) =|>
    State' := 4 /\ N1':= new()
                /\ Ga' := new()
                /\ Un1' := new()
                /\ P1' := exp(N1', Ga')
                /\ Ck2':= {{Ck'}_K}_K
                /\ P2':= Hash(exp(N1, Ck2'))
                /\ SND (Cid'.P1'.P2')
                /\ secret(Cid', e_m, {U,S})
                /\ secret(P1', a_m, {U,S})
                /\ witness(U, S, seq2, Cid'.P1'.P2')
                /\ request(U, S, req1, Cid'.Ck')
 3. State = 4 /\ RCV(Ti'.P3'.P4') =|>
    State':= 6 /\ Pass1':= new() /\ N1':= new() /\ Ck':= new() /\ Un1':=
new()
               /\ Ai':= Hash(xor(Ti', xor(Hash(Un1'.Pass1'),Ck')))
               /\ P44':= exp(P3', Ai')
               /\ Vi':= Hash(P4'. Ai')
               /\ SND(Vi')
               /\ secret(Vi', d_d, {U,S})
               /\ witness(U, S, seq3, Vi')
               /\ request (U, S,req2, Ti'.P3'.P4')

 end role
```

Figure 4. Role specification for IoT devices.

```
role server(U, S: agent,
            K, Ea, Sk: symmetric_key,
            Hash : hash_func,
            SND, RCV: channel (dy))

played_by S
def=

local
      State                  :nat,
      Un1, Cid, Pass1, Ra, T, A, Et, Ai, Ai1,Xa, Ga, Ids,Ck, Ck1, Vii
:text,
      Exp_tm, P1, P2, P22, P3, P4, N2, Da, Ti, Vi :text

      const f_n, y_k, z_k, m_k, p_p, s_p:protocol_id
      init State := 1
transition
 1. State = 1 /\ RCV({Hash(Un1'.Pass1').exp(Pass1', Ga)}_Ea.S)  =|>
    State':= 3 /\ Ra':= new()
               /\ Xa':= new()
               /\ Ids':= new()
               /\ Ga':= new()
               /\ Exp_tm':= new()
               /\ Cid' := xor(Hash(Hash(Ra'.Hash(Un1'.Pass1')).Xa'), Xa')
               /\ Ti':= xor(Ra',Hash(Ra'.Hash(Un1'.Pass1').Xa'))
               /\ Ck':= Hash(Hash(Ra'.Xa').Hash(Exp_tm'.Cid'))
               /\ Ck1':= exp(Ck', Ga)
               /\ Ai':= xor(Ti', xor(Hash(Un1'.Pass1').Ck1'))
               /\ T':= xor(Ti', xor(xor(Cid', Xa'), Ids'))
               /\ A':= xor(Ai', xor(xor(Cid', Xa'), Ids'))
               /\ Et':= xor(Exp_tm', xor(xor(Cid', Xa'), Ids'))
               /\ SND(Cid'.Ck1')
               /\ secret(Cid', f_n, {U,S})
               /\ secret(Ck1', f_n, {U,S})
               /\ witness(S, U, seq4, Cid'.Ck1')
               /\ request(S, U, req3, Hash(Un1'.Pass1').exp(Pass1', Ga))
 2. State = 3 /\ RCV(Cid'.P1'.P2') =|>
    State' := 5 /\ Xa':= new()
                /\ Ids':= new()
                /\ T':= new()
                /\ A':= new()
                /\ Et':= new()
                /\ Ti':= xor(T', xor(xor(Cid', Xa'), Ids'))
                /\ Ai':= xor(A', xor(xor(Cid', Xa'), Ids'))
                /\ Exp_tm':= xor(Et', xor(xor(Cid', Xa'), Ids'))
                /\ Ra':= xor(Ti', xor(Cid', Xa'))
                /\ Ck' := Hash(Ra'.Xa'.Exp_tm'.Cid')
                /\ P22' := exp(P1', Ck')
                /\ Ga' := new()
                /\ N2' := new()
                /\ P3' := exp(N2', Ga')
                /\ P4' := exp(N2', exp(Ai', Ga'))
                /\ SND(Ti'.P3'.P4')
                /\ secret(Ti', y_k, {U,S})
                /\ secret(P3', p_p, {U,S})
                /\ secret(P4', s_p, {U,S})
                /\ witness(S, U, seq5, Ti'.P3'.P4')
                /\ request(S, U, req4, Cid'.P1'.P2')
3. State = 5 /\ RCV(Vi'}_K) =|>
   State':= 7 /\ P2':= new() /\ Ai':= new()
              /\ Vii':= Hash(P2', Ai')
end role
```

Figure 5. Role specification for server.

Subsequently, the server responds with a message <T', $P3'$, $P4'$> (i.e., <T, X_j, Y_j>) to U via the public open channel. Finally, ED_i replies with the message <Vi'> (i.e., < C >) to the server via the public open channel. A knowledge declaration situated at the top of each role is used to specify the initial knowledge of the intruder. The immediate-reaction transition is of the form $X = | > Y$, which relates an event X to an action Y. The declaration witness <U, S, *seq2*, Cid'. $P1'.P2'$> is generated by U, where the *seq2* indicates the message sequence (i.e., message <*alice_bob* _ $Cid'.P1'.P2'$> from Alice to Bob) and <$P1'$, $P2'$> is freshly generated for the server (S). Subsequently, again, another declaration request < S, U, *seq5*, $T'.P3'.P4'$> shows U's acceptance of the random nonce <$P3'$,$P4'$> generated for ED_i by the server, where seq5 indicates the message sequence (i.e., message <*bob_alice* _ $T'.P3'.P4'$> from Bob to Alice). Similarly, we executed the role of the server during the registration phase. Finally, as depicted in Figure 6, we implemented the role of the session, goal, and environment of the proposed scheme. All these roles are the instances with solid arguments in the role session. In the HLPSL protocol, the intruder (i) is likewise interested in the execution of protocol in a concrete session. Furthermore, we defined many secret goals and nine authentication goals. For example, the secret goal (secrecy_of_k1) indicates that ID_i and Psw_i are kept secret from the device U only. The authentication goal: authentication_on seq2 (i.e., authentication_on alice_bob_,$Cid'.P1'.P2'$) means that U generates <$Cid'.P1'.P2'$>, where secret $N1'$ hidden inside <$P1'$,$P2'$> is only known to U. When the server receives <$Cid'.P1'.P2'$> from other messages from the same U, the server performs a strong authentication for the devices based on <$P1'$,$P2'$>. For analyzing the protocol, we selected widely accepted CL–AtSe and OFMC backend for the execution tests. Both CL–AtSe and OFMC backends analyze whether the legal agents can execute the specific scheme by searching for the passive intruder. Subsequently, backend results provide the intruders with the knowledge of some typical sessions among the legitimate agents. The section summary on the CL–AtSe and OFMC backend indicates whether the protocol is SAFE, UNSAFE, or INCONCLUSIVE against all active and passive attacks [13,25]. As depicted in Figure 7, our output summary section shows the safe terminology, meaning that the proposed protocol is safe against significant attacks. Furthermore, the section details specify that the condition under which the proposed protocol is safe or has been used for an attack, and the details also specify why the analysis was inconclusive. Moreover, the protocol section specifies the name of the protocol. The goal section indicates the main objective of the analysis. The backend section represents the name of the backend used. The statistics section specifies why the analysis was inconclusive. The attack-trace section specifies whether an attack is found; the trace of the attack is printed in the standard seq (i.e., Alice–Bob) format. It also represents how the attack has been performed in the protocol. Therefore, on the basis of analyzing the simulation result from Figure 7, we conclude that the proposed protocol is safe.

```
role session (U, S : agent,
     K, K4, K2, P  : symmetric_key,
              Hash  : hash_func)
def=

  local SA, SB, RA, RB: channel (dy)
  composition
  device(U, S, K, P, Hash, SA, RA)/\server(U, S, K, K4, K2, Hash, SB, RB)
  end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment() def=

  const k1, e_m, g_m, h_m, f_n, y_k, z_k, m_k, p_p, s_p, a_m, b_m
:protocol_id,
        seq1, seq2, seq3, seq4, seq5:protocol_id,
        req1, req2, req3, req4 :protocol_id,
        kab, kai, kib, kba, k11, k12, k13, k14, kia, kbi :symmetric_key,
        k15. k16. k17. k18. k19. k20. k21. k22 :symmetric key.
u, s     : agent,
        h     : hash_func

  intruder_knowledge={u,s,kai,kia, kbi, kib}

  composition

session(u,s,kab,k11, k12, k13, h)
     /\ session(u,i, kai, k14, k15, k16, h)
     /\ session(i,s, kib, k17, k18, k19, h)

end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

goal

   secrecy_of k1, e_m, g_m, h_m, f_n, y_k, z_k, m_k, p_p, s_p, a_m, b_m,
d_d
   authentication_on seq1
   authentication_on seq2
   authentication_on seq3
   authentication_on seq4
   authentication_on seq5

   authentication_on req1
   authentication_on req2
   authentication_on req3
   authentication_on req4

end goal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 6. Role specification for IoT devices.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/Paper2_update.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 5 states
 Reachable  : 1 states
 Translation: 0.04 seconds
 Computation: 0.00 seconds
```

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Paper2_update.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.15s
 visitedNodes: 6 nodes
 depth: 3 plies
```

Figure 7. Role specification for server.

- Replay attack: For the replay-attack check, the CL–AtSe and OFMC backend confirms whether the genuine agents can execute the specified protocol by inquiring about an inactive intruder. This backend provides the interloper with the information of some normal sessions among the genuine agents. The outcomes in Figure 7 show that our scheme is secure against replay attacks.
- Active and passive attack check: The outline result for the CL–AtSe and OFMC backend shows that the proposed scheme is SAFE, meaning that the proposed scheme is secure against all active and passive attacks.
- Dolev–Yao model check: For performing the Dolev–Yao model check, the CL–AtSe and OFMC backend additionally confirms whether there is a conceivable man-in-the-middle attack by an intruder. The outcomes show that our scheme satisfies the design properties and is also secure under this backend.

4.3. Formal Security Validation Using ProVerif Tool

ProVerif is widely accepted as an automatic cryptographic protocol verifier tool in the symbolic model (the so-called Dolev–Yao model) [14]. This protocol verifier depends on the portrayal of the protocol by Horn clauses. In this subsection, we implemented the proposed scheme in the Proverif tool to test its correctness of mutual authentication. We performed this experiment on an Intel Core i5–8500 computer, with a 6–core 3.10 GHz CPU and a Windows 10 OS, provided by a service provider. This simulation used the ProVerif version 2.00 binary package [43] to stimulate the registration, mutual authentication, and session key agreement phase between an embedded device and a server. A description of the ProVerif simulation tool can be found in [14].

As depicted in Figure 8, the model comprises three fragments, namely, parameter declaration, principal process, and query execution. The parameter-declaration fragment includes variables, names, and channels other than those with cryptographic capacities. The meanings of the principal process and subprocess are explained in the process segment, although the outline of the assessing scheme is set up in the query-execution segment. A couple of channels, constants, variables, and factors other than those with cryptographic capacities, outlined as constructors and conditions, are displayed in the parameter-declaration fragment. Furthermore, as shown in Figure 9, the principle-process fragment characterizes the commencement and end of the participating clients. The procedures of the execution of the participating clients are kept parallel. Toward the end of the execution, three queries are executed to amend the correctness and secrecy of the proposed scheme. The consequences of the queries are shown in Figure 10. The rightness of the proposed scheme is substantiated because the initial two queries are executed successfully, which indicate the successful beginning of initial interaction between the devices and the server, although its secrecy is affirmed because of an unsuccessful query (i.e., last query in the Queries section in Figure 10) attack on the session key.

```
(***********Declaration of parameters************)
(*************** Channels************************)
free SCh: channel [private].     (*Secure Channel*)
free PCh: channel.               (*Public Channel*)
(************Constants & Varibles***************)
const P: bitstring.
(****Tamper-proof embedded device parameters***)
free IDi: bitstring [private].
free Pswi: bitstring [private].
free PKe: bitstring [private].
(***************Server parameters************)
free PKs: bitstring.
free IDs: bitstring  [private].
free s: bitstring  [private].
(**************Constructor*********************)
fun H1(bitstring, bitstring) : bitstring.
fun H2(bitstring, bitstring, bitstring) : bitstring.
fun H3(bitstring) : bitstring.
fun ECPM(bitstring, bitstring) : bitstring.
fun MULT(bitstring, bitstring) : bitstring.
fun CONCAT(bitstring, bitstring) : bitstring.
fun XOR(bitstring, bitstring) : bitstring.
(***********End of Declaration of parameters*****)
(*********Events and queries executation*********)
(*************** Events************************)
event beginDeviceUi(bitstring).
event endDeviceUi(bitstring).
event beginServerUj(bitstring).
event endServerUj(bitstring).
(*************** Queries************************)
free SK: bitstring [private].
query id : bitstring; inj-event(endDeviceUi(id)) ==>
     inj-event(beginDeviceUi(id)).
query id: bitstring; inj-event(endServerUj(id)) ==>
     inj-event(beginServerUj(id)).
query attacker(SK).
(******End of events and queries executation*****)
```

Figure 8. Declaration of parameters, with event and query execution.

```
(***************Processs************************)
(***************DeviceUi************************)
let DeviceUi=
(***************Registraion*********************)
out(SCh, (H3(CONCAT(IDi, Pswi))));
in(SCh, (CIDi:bitstring, Ckk:bitstring));
(***************Login and Authentication*********)
event beginDeviceUi(IDi);
new xi:bitstring;
let Xi = ECPM(xi,P) in
let Yi = H1(xi,Ckk) in
out(PCh, (CIDi, Xi, Yi));
(*****in(PCh, (xXj:bitstring, xYj:bitstring, xKjp:bitstring,
xIDj:bitstring,   xtj:bitstring)); *******)
in(PCh, (xXj:bitstring, xYj:bitstring, xT:bitstring));
let A = XOR(xT,XOR(H3(CONCAT(IDi, Pswi)), Ckk)) in
let YjP'= H1(xXj, A) in
if (xYj = YjP') then
let C= H3(CONCAT(Yi, A)) in
out(PCh, (C));
let (=SK)= MULT(MULT(xi,xXj), MULT(Pswi,PKs)) in
event endDeviceUi(IDi)
else 0.
(***************Server TS************************)
let ServerUj=
(***************Login and Authentication*********)
event beginServerUj(IDs);
in(PCh, (xXi:bitstring, xYi:bitstring, xCIDi:bitstring));
(**in(PCh, (xXi:bitstring, xYi:bitstring, xCIDi:bitstring,
xxIDi:bitstring,   xti:bitstring));***)
new t:bitstring;
new a:bitstring;
new et:bitstring;
let T = XOR(t,XOR(xCIDi,XOR(IDs, s))) in
let A = XOR(a,XOR(xCIDi,XOR(IDs, s))) in
let Et= XOR(et,XOR(xCIDi,XOR(IDs, s))) in
let R = XOR(T, XOR(xCIDi,s)) in
let CK= H3(CONCAT(CONCAT(R,s), CONCAT(Et,xCIDi))) in
let YiP = H3(ECPM(xXi,CK)) in
if (xYi = YiP) then
new xj:bitstring;
let Xj = ECPM(xj,P) in
let Yj = H3(ECPM(xj,ECPM(A,P))) in
out(PCh, (Xj, Yj, T));
in(PCh,(xC:bitstring));
let C'= H3(CONCAT(xYi,A)) in
if (C' = xC) then
let (=SK)= MULT(MULT(xj,xXi), MULT(s,PKe)) in
event endServerUj(IDs)
else 0.
(*****************TTP************************)
let TTP=
(*************Registration*******************)
in(SCh, xIi:bitstring);
new Ri:bitstring;
new Et:bitstring;
let CIDi = XOR(H3(CONCAT(Ri,CONCAT(xIi,s))), s) in
let CKi = H3(CONCAT(CONCAT(Ri,s), CONCAT(Et,CIDi))) in
let CKi' = ECPM(CKi, P) in
let T = XOR(Ri, H3(CONCAT(Ri,CONCAT(xIi,s)))) in
let A= XOR(T, XOR(xIi, CKi')) in
let t = XOR(T, XOR(CIDi, XOR(IDs, s))) in
let a = XOR(A, XOR(CIDi, XOR(IDs, s)))  in
let et = XOR(Et, XOR(CIDi, XOR(IDs, s))) in
out(SCh, (CIDi, CKi'));
0.
process ((!DeviceUi) | (!TTP) | (!ServerUj))
```

Figure 9. Process specification for embedded device and server.

ProVerif text output:

```
-- Query inj-event(endDeviceUi(id)) ==> inj-event(beginDeviceUi(id))
nounif mess(SCh[],xIi_508)/-5000
Completing...
Starting query inj-event(endDeviceUi(id)) ==> inj-event(beginDeviceUi(id))
RESULT inj-event(endDeviceUi(id)) ==> inj-event(beginDeviceUi(id)) is true.
-- Query inj-event(endServerUj(id_18)) ==> inj-event(beginServerUj(id_18))
nounif mess(SCh[],xIi_1500)/-5000
Completing...
Starting query inj-event(endServerUj(id_18)) ==> inj-event(beginServerUj(id_18))
RESULT inj-event(endServerUj(id_18)) ==> inj-event(beginServerUj(id_18)) is true.
-- Query not attacker(SK[])
nounif mess(SCh[],xIi_2354)/-5000
Completing...
Starting query not attacker(SK[])
RESULT not attacker(SK[]) is true.
```

Figure 10. Output of query execution.

5. Evaluation

In the previous section, we verified the security of our proposed scheme by using the AVISPA tool. We also verified the mutual authentication and session-key agreement of the proposed scheme by using the ProVerif tool. In this section, we present how the proposed scheme is more efficient than the existing schemes in terms of computation, communication, and storage cost, based on a performance analysis, an efficiency study, and a performance study. Note that to evaluate the performance of the proposed scheme, we selected nine recently published authentication schemes (i.e., [16–25]) that have focused on lightweight performance.

5.1. Comparison by Performance

In this section, we implemented the PBC library on embedded devices to calculate the primitive timing needed to measure the operational cost of the proposed scheme. In the PBC library, we implemented a pairing operation on a curve $y^2 = + x^3 + x$ over the field F_q for some prime $q = 3$ mod 4, where both G_1 and G_1 are the group of points belonging to the $E(F_q)$. Note that SHA–256 is much more secure than other hashing algorithms and provides 2256 possible hash values, which makes it nearly impossible for two different attackers to have the exact same hash value coincidentally [45]. Therefore, we used the standard SHA–256 function to compute the cost of the general hash operation. Our experimental setup consisted of an Ubuntu 12.04 virtual machine installed on an Intel Core i5–8500 computer, with a 6–core 3.10 GHz CPU and a Windows 10 OS, provided by a service provider. In the Oracle virtual environment, we stimulated IoT devices and set RAM to 256 MB and execution cap to 26 percent (i.e., it reflects a single-core 798 MHz CPU), which is not very far from a real IoT system configuration [46].

To calculate the primitive timing of different cryptography operations used in the proposed scheme, our simulation used the pbc–0.5.14 library [10] and the GMP library [47]. Finally, we calculated and compared the operational cost of the proposed scheme and existing schemes using the PBC library to testify to the performance of the proposed scheme over existing schemes. Note that existing schemes use heavy operations such as bilinear pairings, modular exponentiation, and symmetric encryption + decryption, which are generally the most computationally expensive cryptographic operations. Therefore, we computed only such heavy operations on IoT devices and noted their respective execution times for the existing schemes. Then, we implemented the proposed scheme's code and recorded the execution time for hash and ECC operation. Table 3 summarizes the outcome of the simulation performance analysis of the PBC code, which we executed, along with the terms used to describe different cryptographic operations. The cost of detail computation is as follows:

Table 3. Primitive timing for cryptography operation.

Operation	Terminology	Execution Time (ms)
Hash operation	T_h	0.039
ECC operation	T_{ecm}	0.110
Modular exponentiation	T_{expo}	0.343
Pairing	T_{pair}	0.992
Symmetric encryption + decryption	T_s	0.686

- Computational cost: Notice that the computing costs of lightweight operations (i.e., XOR, concatenation, and comparison) are overlooked due to their inexpensive computation. At the authentication phase, a considerable computing process is conducted. Hence, we measure the cost of the computation at the authentication phase. As depicted in Table 4, we compute some recent schemes' computation cost along with that of the proposed scheme. To compute cost, we count the number of T_h, T_{ecm}, T_{expo}, T_{pair}, and T_s operations involved in the login phase, where communication happens on a public open channel. Then, for each cryptography operation, we assign the value we obtained in our PBC implementation. We evaluated all computation costs in *m*s; however, this *m*s is the considerable maximum period when processing a high number of authentication requests (i.e., millions and zillions) in a real-time scenario. We also computed the computation cost of some recent schemes. Compared to the computation costs of the existing schemes, the proposed scheme is more productive by virtue of its lightweight computation power.
- Communication cost: We used the parameter < q, r > of each of the size < 256, 224 > bits during the PBC–library-based primitive timing calculation; we chose this parameter since the suggested elliptic curve key length is 256 bit for NIST 2016–2030 and ECRYPR II 2031–2040. An ECC point $X, Y = (x_P, y_P) \in E_p(a, b)$ therefore, requires (128 + 128)

= 256 bits, where each parameter consumes 128 bits. The costs of the other parameters for the communication element, described in Table 5, are $X_i, Y_i, CID_i, X_j, Y_j, T$, and C_i. Thus, the communication cost is $(X_i, Y_i, CID_i) + (X_j, Y_j, T) + C_i)$ (640 + 640 + 128) = 1408 bits. Similarly, we computed the communication costs for the recent schemes of existing approaches and compared the recent schemes with the proposed scheme. In comparison, as depicted in Table 4, we conclude that the communication cost of the proposed protocol is lower than the existing schemes.

- Storage cost: Throughout this subsection, we measured the storage component cost depending on the PBC library applied on our machine, which meets NIST [48] and ECRYPR II [49], stored in the embedded device's memory. In the proposed scheme, the embedded device stores cookies as recommendations. To compute storage cost, we need to identify the number of components is well as their pseudo-identity (i.e., C'_k, CID_i), thereby consuming 256 + 128 = 384 bits of memory. Likewise, we counted a number of components stored in the embedded devices' memory for recent schemes and computed their storage cost. From Table 4, we infer that the proposed scheme consumes a very small amount of memory, 384 bits, compared to other existing schemes.

Thus, from Table 4, we conclude that the proposed scheme has a lower running (i.e., operation) cost for IoT devices relative to other recent schemes. This lower operating cost helps the proposed scheme to eliminate the existing gaps in recent schemes in order to become more efficient and suitable for real-time scenarios.

Table 4. Comparison of computation, communication, and storage costs.

Schemes	Computation Cost (*ms*)	# of Messages	Communication Cost (Bit)	Storage Cost (Bit)
Zhou et al. [16]	$36T_h = 1.404$	4	3072	640
Yu et al. [17]	$34T_h = 1.326$	4	2176	384
Xie et al. [18]	$9T_h + 7T_{ecm} = 1.121$	5	1792	384
Chatterjee et al. [20]	$11T_h + 13T_{ecm} = 1.859$	4	2176	640
Yu et al. [21]	$1T_{pair} + 6T_{expo} + 6T_h = 3.284$	3	1408	768
Sengupta et al. [22]	$11T_h + 9T_{ecm} = 1.419$	3	1408	640
Yang et al. [23]	$28T_h = 1.311$	10	5376	512
Wang et al. [24]	$21T_h + 5T_s = 4.249$	4	2304	768
Wazid et al. [25]	$34T_h = 1.326$	3	1664	640
Proposed	$7T_h + 6T_{ecm} = 0.933$	3	1408	384

Table 5. Comparison of functional requirements.

Schemes	Mutual Authentication	Friendly Password Selection and Its Security	Session-Key Agreement	Secured Password Update
Zhou et al. [16]	Supported	Supported	Supported	Supported
Yu et al. [17]	Supported	Supported	Supported	Supported
Xie et al. [18]	Supported	n/a	Supported	n/a
Chatterjee et al. [20]	Supported	n/a	Supported	n/a
Yu et al. [21]	Supported	Supported	Supported	Supported
Sengupta et al. [22]	Supported	Supported	Supported	Supported
Yang et al. [23]	Supported	Supported	Supported	n/a
Wang et al. [24]	Supported	Supported	Supported	Supported
Wazid et al. [25]	Supported	n/a	Supported	n/a
Proposed	Supported	Supported	Supported	Supported

5.2. Comparison by Functional Requirement

This subsection compares the proposed scheme with those of the existing schemes by satisfying the following crucial functionality requirements:

- Mutual authentication: The proposed scheme provides mutual authentication between embedded devices (ED_i) and server (TS) by using a three-way challenge–response handshake technique. The TS validates ED_i by checking whether Y_i'(i.e., $h(X_i.CK)) = ?Y_i$ and C_i'(i.e., $h(Y_i \| A)) = ?C_i$, while ED_i validates S by verifying whether Y_j'(i.e., $h(X_j.A)) = ?Y_j$. It is impossible to compute Y_j' without knowing CK' and I_i as both CK' and I_i are completely secure due to the tamper-proof nature of the device [43]. Furthermore, in a server, the security of CK depends upon the factors R and E_t. This factor (i.e., R, A and E_t) security is protected by a secret server key (s) and a secret identity. This parameter (i.e., s and ID_{TS}) security is preserved in a server key ($E_{ks} = s.ID_{TS}.G$) by the concept of ECDLP and CDHP. For the above-mentioned reasons, the attacker cannot compute factors $h((Y_i \| A))$ and $h(X_i.CK)$ without knowing A and CK. Thus, only ED_i and TS can mutually authenticate each other. The result shown in the Proverif tool in Section 4.3 verifies the mutual authentication confirmation. Therefore, the proposed scheme achieves mutual authentication.
- Confidentiality (or friendly password selection and its security): In the proposed scheme, for easy remembering, the client can pick a simple password psw_i, which might be of low or high intensity [41]. This password is stored in the password-generator format (i.e., $P_{ke} = psw_i.G$). It is very difficult to extract the password from the password generator because of the difficulty of ECDLP. Even identity $I_i = h(ID_i \| Psw_i)$ contains a hash of the concatenation of device identity and the password generator, which is stored in the server. It is very difficult to extract a Psw_i from I_i because of the property of the collision-resistant one-way hash function ($h()$).
- Session-key agreement: The proposed scheme justifies the session-key agreement in the session-key-computation subsection, ensuring secure sessions with highly confidential data exchange between the embedded devices (ED_i) and the server (TS). In addition, a session key $(S_K = x_{i1}.X_j.\ Psw_i.P_{ks} = x_{j1}.X_i.s.P_{ke})$ is protected by the concept of ECDLP and CDHP. The result from the Proverif tool in Section 4.3 also verifies its session key security.
- Secure password update: The proposed scheme supports password updates for embedded devices. The password update helps ensure the security of the proposed protocol (i.e., it prevents DoS attacks on the scheme). Legitimate embedded devices can change their passwords any time after registration.

Finally, as depicted in Table 5, we compared the proposed scheme with the existing schemes on functional requirements. In the comparison, we observed that the scheme proposed by Zhou et al., Yu et al., Yu et al., Sengupta et al., and Wang et al. support all crucial functional requirements. However, the scheme proposed by Xie et al., Chatterjee et al., and Wang et al. failed to support friendly password selection and its security and to secure password updates. The scheme proposed by Yang et al. failed to support a secure password update policy. However, the proposed scheme removed those limitations and reinforced security requirements.

5.3. Summary

In this section, as depicted in Figure 11, we summarize the proposed scheme's performance in comparison with the existing schemes based on security necessities, cost calculation, and functional requirement factors using the earlier analysis results in Sections 5.1 and 5.2. The first, second, third, and fourth bars in Figure 11 represent computation, communication, storage cost, the types of security attacks prevented, and the number of additional security features (i.e., crucial functional requirements) supported. The analysis results in Figure 11 show that recent authentication schemes have high computation cost or high storage cost or high communication levels. Furthermore, the existing schemes suffer from various security attacks and lack other crucial security requirements, such as friendly password selection and its security and update policy. However, the proposed scheme eliminates all the previously mentioned security-related problems and vulnerabilities of the existing schemes and prevents 12 types of security attacks. Moreover,

the proposed scheme also supports prevention against four more security attacks, which we explained in Section 4.1. Therefore, the proposed scheme removes security weakness from the recent existing schemes and prevents 16 types of major security attacks. Furthermore, the proposed scheme not only secures against 16 major attacks but also successfully improves security mechanisms by providing four additional security features (i.e., mutual authentication, session-key agreement, friendly password selection, and updates) at the same time, which is absent in recent existing schemes. In addition, the proposed scheme maintains the lowest computational cost of 9330×10^{-1} µs, the lowest storage cost of 384 bit, and the lowest communication cost of 1408 bit. Due to the lightweight nature of the proposed scheme, the proposed scheme is significantly more efficient in handling a high number (i.e., millions or zillions) of authentication requests in real-time IoT cloud scenarios. This lightweight flexibility helps the proposed scheme to achieve significant optimized usage of memory, electricity power, and computational power and efficiently reduce traffic on the communication channel and improve the response time of authentication requests. This is the significant contribution of this paper, which will play a substantial role while dealing with a huge organizational need.

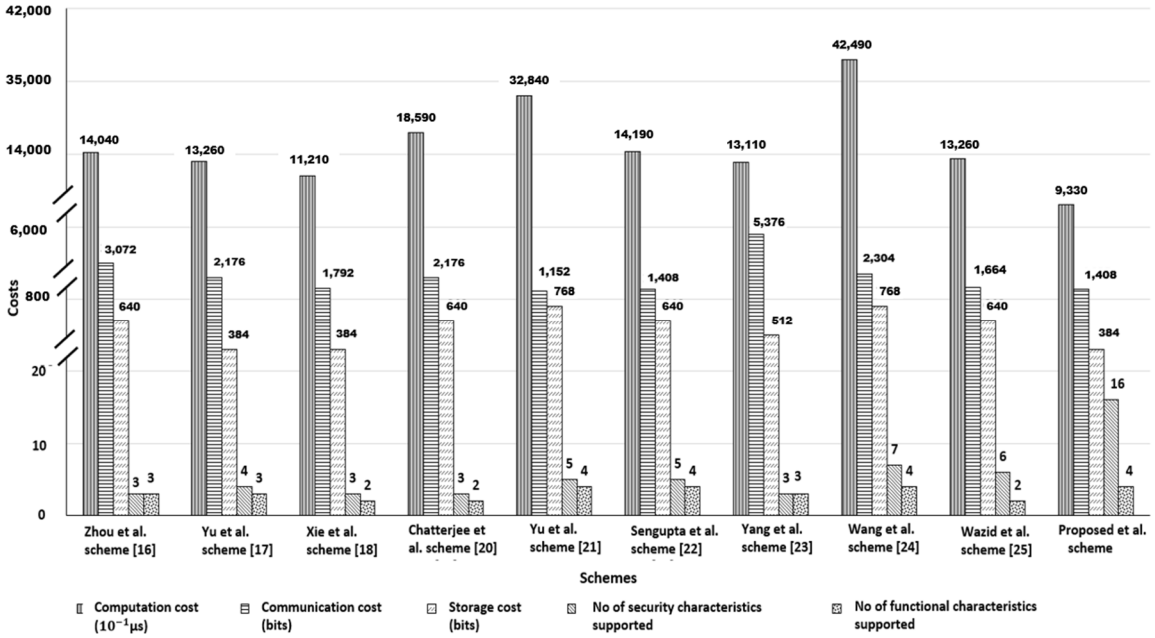

Figure 11. Summary of performance analysis.

6. Conclusions

In this study, we examined current authentication schemes reported to be secure, lightweight architecture in IoT environments. The analysis of the study indicates that the current schemes do not achieve the target of lightweight computing with resilience to all forms of known attack. This would also reduce the performance of embedded devices, which results in the consumption of additional resources (e.g., electricity power, memory). Therefore, this can be a significant loss while satisfying users' needs in a vast organizational scenario. To address this issue, we proposed a novel lightweight-computation authentication scheme. We verified the security strength of the proposed scheme using mathematical formulation (i.e., informal security analysis for 16 security attacks). Then, we verified the mathematical proof using widely accepted standard AVISPA and ProVerif tools. The result confirms that the proposed scheme is preventive against all kinds of active and passive attacks. Moreover, we verified the performance of the proposed scheme by implementing the widely accepted standard PBC library on the embedded devices. The outcome shows that the proposed scheme is lightweight (i.e., lowest computation

(0.933 ms), communication (1408 bit), and storage (384 bit)) compared with other recent authentication protocols, rendering the proposed scheme reasonable and realistic for massive implementation scenarios based on industrial IoT environments. In the future, we would like to build an interoperable architecture to allow devices to use the same kind of lightweight authentication mechanism when accessing different types of cloud network services. The interesting fundamental question that arises from our research analysis is whether the lack of sufficient security and a lightweight goal can explain many of the possible repeated errors in past authentication protocols and the importance of building a more efficient, robust, lightweight authentication scheme in future work.

Author Contributions: Formal analysis, A.T.; funding acquisition, Y.-G.K.; investigation, A.T.; validation, Y.-G.K.; writing—original draft, A.T.; writing—review and editing, Y.-G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT; No.2019- 0-00231, development of artificial intelligence-based video security technology and systems for public infrastructure safety).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Roy, S.; Chatterjee, S.; Das, A.K.; Chattopadhyay, S.; Kumari, S.; Jo, M. Chaotic map-based anonymous user authentication scheme with user biometrics and fuzzy extractor for crowdsourcing internet of things. *IEEE Internet Things J.* **2018**, *5*, 2884–2895.
2. Yang, Y.; Wu, L.; Yin, G.; Li, L.; Zhao, H. A survey on security and privacy issues in internet-of-things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258.
3. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the mirai botnet. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 1093–1110.
4. Kelly, C.; Kelly, N.; McKeown, S.; Lambrinoudakis, C. Testing and hardening IoT devices against the mirai botnet. In Proceedings of the 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Dublin, Ireland, 15–19 June 2020; pp. 1–8.
5. Oh, S.-R.; Kim, Y.-G.; Cho, S. An interoperable access control framework for diverse IoT platforms based on oauth and role. *Sensors* **2019**, *19*, 1884.
6. Park, Y.; Choi, H.; Cho, S.; Kim, Y.-G. Security analysis of smart speaker: Security attacks and mitigation. *Comput. Mater. Contin.* **2019**, *61*, 1075–1090.
7. Dhillon, P.K.; Kalra, S. A secure multi-factor ECC based authentication scheme for cloud-IoT based healthcare services. *J. Ambient Intell. Smart Environ.* **2019**, *11*, 149–164.
8. Punithavathi, P.; Geetha, S.; Karuppiah, M.; Islam, S.H.; Hassan, M.M.; Choo, K.K.R. A lightweight machine learning-based authentication framework for smart IoT devices. *Inf. Sci.* **2019**, *484*, 255–268.
9. Wang, P.; Li, B.; Shi, H.; Shen, Y.; Wang, D. Revisiting anonymous two-factor authentication schemes for IoT-enabled devices in cloud computing environments. *Secur. Commun. Netw.* **2019**, *2019*, 2516963.
10. Pbc Library. Available online: https://crypto.stanford.edu/pbc/ (accessed on 16 December 2020).
11. Challa, S.; Das, A.K.; Gope, P.; Kumar, N.; Wu, F.; Vasilakos, A.V. Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems. *Future Gener. Comput. Syst.* **2020**, *108*, 1267–1286.
12. Panda, P.K.; Chattopadhyay, S. A secure mutual authentication protocol for IoT environment. *J. Reliab. Intell. Environ.* **2020**, *6*, 79–94.
13. Oheimb, D.V. The high-level protocol specification language helps developed in the EU project AVISPA. In Proceedings of the APPSEM 2005, Frauenchiemsee, Germany, 12–15 September 2005; pp. 1–17.
14. Maitra, T.; Obaidat, M.S.; Amin, R.; Islam, S.H.; Chaudhry, S.A.; Giri, D. A robust elgamal-based password-authentication protocol using smart card for client-server communication. *Int. J. Commun. Syst.* **2017**, *30*, e3242.
15. Sowjanya, K.; Dasgupta, M.; Ray, S. An elliptic curve cryptography based enhanced anonymous authentication protocol for wearable health monitoring systems. *Int. J. Inf. Secur.* **2020**, *19*, 129–146.

16. Zhou, L.; Li, X.; Yeh, K.H.; Su, C.; Chiu, W. Lightweight IoT-based authentication scheme in cloud computing circumstance. future generation computer systems. *Future Gener. Comput. Syst.* **2019**, *91*, 244–251.
17. Yu, S.; Park, K.; Park, Y. A secure lightweight three-factor authentication scheme for IoT in cloud computing environment. *Sensors* **2019**, *19*, 3598.
18. Xie, Z.; Jiang, L. An improved authentication scheme for the internet of things. *Mater. Sci. Eng.* **2020**, *715*, 012031.
19. Wang, K.-H.; Chen, C.-M.; Fang, W.; Wu, T.-Y. A secure authentication scheme for the internet of things. *Pervasive Mob. Comput.* **2017**, *42*, 15–26.
20. Chatterjee, S.; Samaddar, S.G. A robust lightweight ECC-based three-way authentication scheme for IoT in the cloud. In Proceedings of the 6th International Conference on Advanced Computing Networking, and Informatics (ICANI 2018) NIT, Silchar, India, 4–6 June 2018; pp. 101–111.
21. Yu, Y.; Hu, L.; Chu, J. A Secure authentication and key agreement scheme for IoT-based cloud computing environment. *Symmetry* **2020**, *12*, 150.
22. Sengupta, S. A secured biometric-based authentication scheme in IoT-based patient monitoring system. In Proceedings of the International Conference on Emerging Technology in Modelling and Graphics (IEMGraph 2018), Kolkata, India, 6–7 September 2018; pp. 501–518.
23. Yang, S.K.; Shiue, Y.M.; Su, Z.Y.; Liu, I.H.; Liu, C.G. An authentication information exchange scheme in WSN for IoT applications. *IEEE Access* **2020**, *8*, 9728–9738.
24. Wang, F.; Xu, G.; Xu, G.; Wang, Y.; Peng, J. A robust IoT-based three-factor authentication scheme for cloud computing resistant to session key exposure. *Wireless Commun. Mobil. Comput.* **2020**, *2020*, 3805058.
25. Wazid, M.; Das, A.K.; Bhat, V.; Vasilakos, A.V. LAM-CIoT: Lightweight authentication mechanism in a cloud-based IoT environment. *J. Netw. Comput. Appl.* **2020**, *150*, 102496.
26. Wei, F.; Zhang, R.; Ma, C. A provably secure anonymous two-factor authenticated key exchange protocol for cloud computing. *Fundam. Inform.* **2018**, *157*, 201–220.
27. Li, X.; Xiong, Y.; Ma, J.; Wang, W. An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards. *J. Netw. Comput. Appl.* **2012**, *35*, 763–769.
28. Wang, B.; Ma, M. A smart card based efficient and secured multi-server authentication scheme. *Wireless Pers. Commun.* **2013**, *68*, 361–378.
29. Limbasiya, T.; Soni, M.; Mishra, S.K. Advanced formal authentication protocol using smart cards for network applicants. *Comput. Electr. Eng.* **2018**, *66*, 50–63.
30. Feng, Y.; Wang, W.; Weng, Y.; Zhang, H. A replay-attack resistant authentication scheme for the internet of things. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (euc), Guangzhou, China, 21–24 July 2017; pp. 541–547.
31. Nikooghadam, M.; Jahantigh, R.; Arshad, H. A lightweight authentication and key agreement protocol preserving User anonymity. *Multimed. Tools Appl.* **2017**, *76*, 13401–13423.
32. Kumari, S.; Khan, M.K.; Li, X. An improved remote user authentication scheme with key agreement. *Comput. Electr. Eng.* **2014**, *40*, 1997–2012.
33. Alkuhlani, A.M.; Thorat, S.B. Lightweight anonymity-preserving authentication and key agreement protocol for the internet of things environment. In Proceedings of the International Conference on Intelligent Information Technologies (ICIIT 2017), Chennai, India, 20–22 December 2018; pp. 108–125.
34. Dhillon, P.K.; Kalra, S. Multi-factor user authentication scheme for IoT-based healthcare services. *J. Reliab. Intell. Environ.* **2018**, *4*, 141–160.
35. Amin, R.; Kumar, N.; Biswas, G.P.; Iqbal, R.; Chang, V. A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment. *Future Gener. Comput. Syst.* **2018**, *78*, 1005–1019.
36. Xue, K.; Hong, P.; Ma, C. A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. *J. Comput. Syst. Sci.* **2014**, *80*, 195–206.
37. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A survey on the edge computing for the internet of things. *IEEE Access* **2017**, *6*, 6900–6919.
38. Hester, R.E.; Harrison, R.M. *Energy Storage Options and Their Environmental Impact*; Royal Society of Chemistry: London, UK, 2018.
39. Lo, J.W.; Wu, C.Y.; Chiou, S.F. A lightweight authentication and key agreement scheme for telecare medicine information system. *J. Inter. Technol.* **2020**, *21*, 263–272.
40. Kumari, S.; Karuppiah, M.; Das, A.K.; Li, X.; Wu, F.; Kumar, N. A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. *J. Supercomput.* **2018**, *74*, 6428–6453.
41. Bertino, E.; Shang, N.; Wagstaff, S.S., Jr. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Trans. Depend. Secur. Comput.* **2008**, *5*, 65–70.
42. wolfSSL. Available online: http://www.wolfssl.com/ (accessed on 16 December 2020).
43. Proverif: Cryptographic Protocol Verifier in the Formal Model. Available online: http://prosecco.gforge.inria.fr/personal/bblanche/proverif/ (accessed on 16 December 2020).
44. AVISPA Web Tool. Available online: http://www.avispa-project.org/web-interface/basic.php (accessed on 16 December 2020).

45. Chen, Y.; Martínez, J.F.; Castillejo, P.; López, L. A bilinear map pairing based authentication scheme for smart grid communications: Pauth. *IEEE Access* **2019**, *7*, 22633–22643.
46. Smart Connected Secure. Available online: http://www.microchip.com/design-centers/internet-of-things (accessed on 16 December 2020).
47. GMP Source. Library for Arbitrary Precision Arithmetic, Operating on Signed Integers, Rational Numbers, and Floating-Point Numbers. Available online: https://gmplib.org/ (accessed on 16 December 2020).
48. Recommendation for Key Management, Part 1: General, SP 800-57 Part 1 Rev. 4. Available online: https://www.keylength.com/en/compare/ (accessed on 16 December 2020).
49. Algorithms, Key Size and Protocols Report. Document H2020-ICT2014-Project 645421, D5.4. ECRYPT-CSA. 2018. Available online: https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf (accessed on 16 December 2020).