mathematics

Article

Secure Authentication in the Smart Grid

Mehdi Hosseinzadeh [1,2,3,†], Rizwan Ali Naqvi [4,†], Masoumeh Safkhani [5,6,†], Lilia Tightiz [7,*] and Raja Majid Mehmood [8,*]

1 Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
2 School of Medicine and Pharmacy, Duy Tan University, Da Nang 550000, Vietnam
3 Computer Science, University of Human Development, Sulaymaniyah 0778-6, Iraq
4 School of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, Republic of Korea
5 Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
6 School of Computer Science, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran 16788-15811, Iran
7 School of Computing, Gachon University, 1342 Seongnamdaero, Seongnam 13120, Republic of Korea
8 Information and Communication Technology Department, School of Computing and Data Science, Xiamen University Malaysia, Sepang 43900, Malaysia
* Correspondence: liliatightiz@gachon.ac.kr (L.T.); rmeex07@ieee.org (R.M.M.)
† These authors contributed equally to this work.

Abstract: Authenticated key agreement is a process in which protocol participants communicate over a public channel to share a secret session key, which is then used to encrypt data transferred in subsequent communications. LLAKEP, an authenticated key agreement protocol for Energy Internet of Things (EIoT) applications, was recently proposed by Zhang et al. While the proposed protocol has some interesting features, such as putting less computation on edge devices versus the server side, its exact security level is unclear. As a result, we shed light on its security in this paper through careful security analysis against various attacks. Despite the designers' security claims in the random oracle model and its verification using GNY logic, this study demonstrates that this protocol has security weaknesses. We show that LLAKEP is vulnerable to traceability, dictionary, stolen smart glass, known session-specific temporary information, and key compromise impersonation attacks. Furthermore, we demonstrate that it does not provide perfect forward secrecy. To the best of our knowledge, it is the protocol's first independent security analysis. To overcome the LLAKEP vulnerabilities, we suggested the LLAKEP$^+$ protocol, based on the same set of cryptographic primitives, namely the one-way hash function and ECC point multiplication. Our comprehensive security analysis demonstrates its resistance to different threats, such as impersonation, privileged insider assaults, and stolen smart glass attacks, along with its resistance to sophisticated assaults, such as key compromised impersonation (KCI) and known session-specific temporary information (KSTI). The overhead of the proposed protocol is acceptable compared to the provided security level.

Keywords: authentication; key agreement; energy internet of things; security; key compromised impersonation attack; known session-specific temporary information attack

MSC: 94A62

check for updates

Citation: Hosseinzadeh, M.; Ali Naqvi, R.; Safkhani, M.; Tightiz, L.; Majid Mehmood, R. Secure Authentication in the Smart Grid. *Mathematics* **2023**, *11*, 176. https://doi.org/10.3390/math11010176

Academic Editor: Antanas Cenys

Received: 1 December 2022
Revised: 21 December 2022
Accepted: 23 December 2022
Published: 29 December 2022

1. Introduction

The Internet of Things (IoT) is a new technological concept that aims to provide a communication channel between objects to advance their functionality and enhance the traditional mechanism. It also adapted to different applications with dedicated features, for example, the Internet of Energy or Energy IoT (IoE/EIoT) [1], Internet of Drones (IoD) [2], Industrial Internet of Things (IIoT) [3], Internet of Vehicles (IoV) [4], and Medical Internet of Things (MIoT) [5]. While communication between objects in these technologies provides many advantages, security is a big challenge for all of them, given that different objects have different properties and constraints, and the data should be transferred over the public channel using wireless technologies. Hence, that data could always be accessed by the adversary and used for unauthorized purposes. In addition, the adversary could aim to impersonate any object to just use its capabilities or do malicious activities, e.g., spying, encryption of data for a ransom, total wipeout of disk and data, and abuse for coin mining [6]. Hence, any such application requires an approach to identify friends and foes. Authentication protocols are traditional solutions for providing authorized access to infrastructure while blocking unauthorized access.

The Energy Internet of Things enables the development of new complex communication infrastructures for the modernization and automation of energy infrastructures for producers and manufacturers, as depicted in Figure 1. It results in greater connectivity of these safety-critical systems, allowing for more efficient management of operational processes, service provisioning, and information exchange for various (third-party) actors. EIoT enables more efficient and environmentally friendly energy production with the least amount of waste. This technology, however, raises security concerns, as evidenced by numerous recent cyberattacks on critical infrastructures and utilities [7,8]. The reason for this is that their reliance on information technology (IT) for operation and control is increasing, making them more vulnerable to malicious intent.

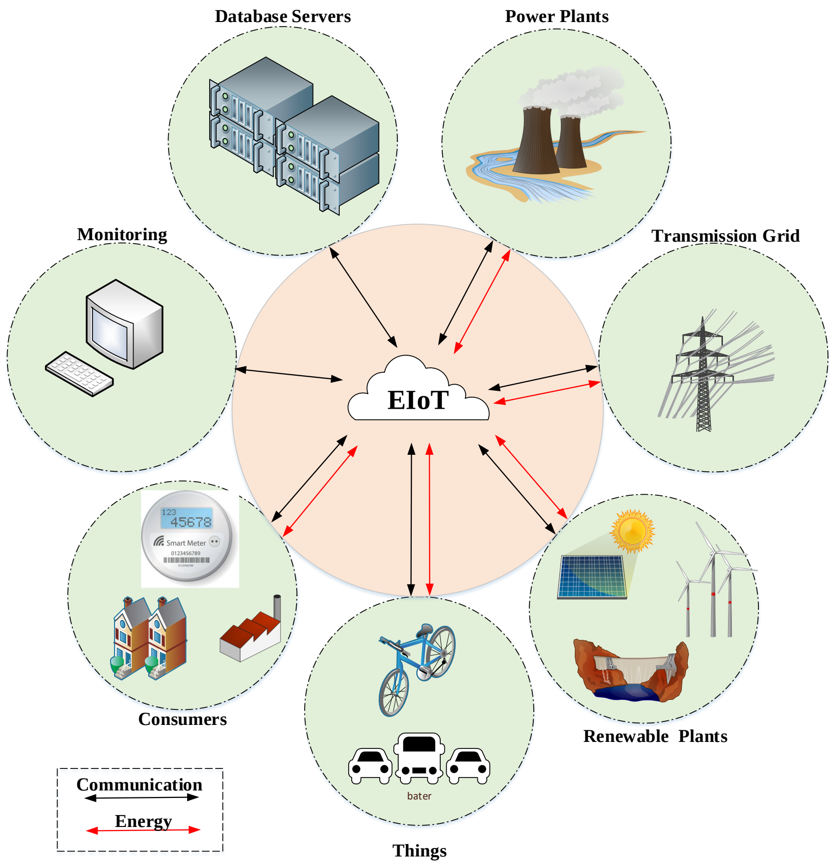


Figure 1. The Energy Internet of Things (EIoT) ecosystem.

Zhang et al. [9] recently proposed an authenticated key agreement (AKE) protocol for EIoT in the metaverse era and named it LLAKEP, which is a protocol for secure authentication and key agreement between an electric bike rider and a battery swap station. In the proposed protocol, the client has fewer computing overheads compared to the server. Hence, the expected latency will be reduced. They also formally analyzed the security of the proposed protocol in the random oracle model and evaluated it using GNY logic. The provided security analysis confirms its security in those models. However, it will not guarantee the protocol's security in a real application because those models do not cover all security concerns. For example, in reality, the adversary can access the device and read its memory or gain the low entropy of passphrases that are used by the users. These types of attacks are not considered in such formal proof because the designers generally consider the protocol environment ideal. Hence, it could be interesting to evaluate the real-world security of this protocol, considering the adversary's advantage in reality, and we aim to address this point in this study.

1.1. Our Contribution

This paper's contributions are as follows:

1. We propose the first independent security analysis of LLAKEP in various scenarios with varying access to the adversary. Our security analysis demonstrates important security issues in this protocol.
2. In terms of efficiency, we show that this protocol could be designed more efficiently and propose a promising solution to address this flaw.
3. We propose the LLAKEP^{+} protocol as an improved protocol that uses the same set of cryptographic primitives as the LLAKEP protocol.
4. We carefully evaluate the security of the proposed protocol both informally and formally using a real or random model and confirm its security using the Scyther tool. We also evaluate the security of the proposed protocol and compare it with the state of the art, which shows that the proposed protocol has a low communication cost and a comparable computation cost.

1.2. Paper Organization

The rest of the paper is organized as follows: in Section 2, we represent the required preliminaries, including notations and some background information. Next, in Section 2.4, we look at the LLAKEP scheme. We demonstrate the security flaws of this scheme in Section 3. LLAKEP^{+}, the enhanced protocol, is proposed in Section 4, and its security and efficiency analysis is provided. Finally, the paper is wrapped up in Section 6.

2. Preliminaries

2.1. Notation

We use the list of notations represented in Table 1 in this paper.

Table 1. List of used notations.

Symbol	Description
EBR	Electric bike riders
BSS	Battery swap station
MC	Microprocessor chip
A	Adversary
ID_{EBR}	Identity of an electric bike rider EBR
PW_{EBR}	Password of an electric bike rider EBR
sk_X	Private key of X
pk_X	Public key of X
sk	Session key
E/F^p	An elliptic curve E over a prime finite field F^p with p being a large prime
n	Order of base point G
Z^n	$1, 2, \ldots, n-1$
$k \cdot G$	Scalar multiplication on elliptic curves and G is a base point in E/Fp
$A\|B$	Concatenation operation between strings A and B
$A \oplus B$	XOR operation between strings A and B
$kdf(\cdot)$	Key derivation function
$H(\cdot)$	A one-way hash function that generates digests

2.2. Elliptic Curve Cryptography

Given a curve $y^2 = x^3 + a \cdot x + b$, a distinguished point at infinity $\mathcal{O}$, and a prime number q, the Elliptic Curve Cryptography (ECC) E_{F_q} is defined as an additive group $\mathcal{G}$ over the finite field F_q including the set of all $(x, y) \in F_q \times F_q$ such that $\lambda^2 = \mu^3 + a\mu + b$, where $a, b \in F_q$ and $4a^3 + 27b^2 \ mod \ q \ \neq 0$, along with $\mathcal{O}$. The order of $\mathcal{G}$ has been defined by the size of the subgroup, which is defined as the smallest positive number n such that $n \cdot G = \mathcal{O}$, G is the generator of this subgroup [10].

Regarding the security of ECC, for large values of n, given any natural scalar $a \in F_q$, it is simple to calculate $y = a \cdot G$ (point multiplication), but given y, E_{F_q}, and G, computing the multiplicand a is computationally impossible. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is a difficult problem whose difficulty is determined by the size of the elliptic curve subgroup, i.e., n. The Elliptic Curve Computational Diffie–Hellman Problem (EC-CDHP) is another difficult problem over ECC that aims to compute $a \cdot b \cdot G$ given $a \cdot G$, $b \cdot G$, E_{F_q}, and G where $a, b \in F_q$ [11].

2.3. System Model

In this paper, we consider a system that includes a battery swap station (BSS), which is used to provide service to registered electric bike riders (EBR), such as battery swapping, using Figure 2. Furthermore, we assume that, with the exception of the registration phase, the remainder of the messages between EBR and BSS are sent over the public channel. Communication consists of two phases: in the first phase, EBR and BSS authenticate each other to share a secret key.

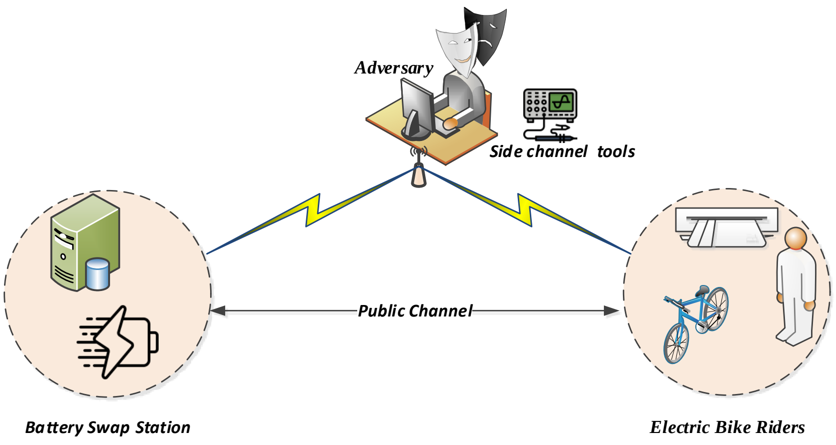


Figure 2. The used system model.

In this paper, depending on the concept, we use the Dolev–Yao (DY) [12] and Canetti and Krawczyk (CK) [13] adversary models. The main difference between these models is the adversary capabilities: in the DY paradigm, the attacker has complete control over messages sent via public channels, or it can take a valid user's smart card and retrieve the stored value from it. On the other hand, in the CK adversary model, the adversary can also compromise the session and access random values generated in each session, for instance. We consider a Probabilistic Polynomial Time Adversary (PPTA) in any model that can only perform a polynomial number of operations, including at most a polynomial call to any cryptographic primitive.

In this paper, following Figure 2, we examine a system that includes a Battery Swap Station (BSS), which is used to provide services such as battery switching to registered Electric Bike Riders (EBRs). Furthermore, with the exception of the registration step, we assume that the remaining communications between EBR and BSS are sent via the public channel. The communication process is divided into two stages. In the first stage, EBR and BSS authenticate each other to exchange a secret key. The shared key is then used to encrypt the remainder of the sent data.

As with the designer model, we assume that each entity has a private and public pair of ECC-based public keys. The public key is known to all protocol entities, including the attacker. The owner is the only one who has access to the private key. In addition to this authentication element, each registered EBR has a username and password as a secondary authentication factor. Because the user is expected to remember the username and password, they are picked from a low-entropy feasible space, and the adversary may execute a dictionary search on each of them independently. However, simultaneously scanning the space of their concatenation is not conceivable.

Although the authentication factors should be kept private, they can be compromised at any time. In this situation, the leak of information should not affect the security of earlier sessions.

To account for all possibilities, we assume the adversary has access to the ephemeral keys, which are the random values created at the end of each session. In actuality, side-channel attacks such as power analysis may do this. The opponent should not gain any more knowledge regarding long-term secrets or the shared session key in this situation.

2.4. LLAKEP Description

LLAKEP's four phases are initialization, user registration, authenticated key agreement, and password update.

2.5. Initialization

The system parameters are chosen by the battery swap station (BSS) during the initialization phase, i.e., $\{E/F_p, G, n, pk_{BSS}, H_1(\cdot), H_2(\cdot)\}$, where $pk_{BSS} = sk_{BSS} \cdot G$.

2.6. User Registration

User registration is required to register an electric bike rider (EBR) in BSS. EBR inputs ID_{EBR} and PW_{EBR} on the smart glass, which includes a microchip MC, during this protocol step. The MC generates random numbers a_{MC} and b_{MC} and computes $HIP = H_2(ID_{EBR} \| PW_{EBR})$, $v = HIP \oplus a_{MC}$, $d = HIP \oplus b_{MC}$ and $C = H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$. Next, using a secure channel, EBR submits $\{pk_{EBR}, ID_{EBR}, d\}$ to BSS.

BSS checks the uniqueness of ID_{EBR} and $H_2(ID_{EBR})$ after receiving the message then it computes $l = H_1(sk_{BSS}) \oplus d \oplus H_2(sk_{BSS} \| ID_{EBR})$, stores $\{H_2(sk_{BSS} \| ID_{EBR}), ID_{EBR}\}$ and sends l to EBR.

On the other hand, EBR calculates $l' = l \oplus b_{MC} = H_1(sk_{BSS}) \oplus HIP \oplus H_2(sk_{BSS} \| ID_{EBR})$ and stores it along v and C in MC.

2.7. Authenticated Key Agreement

To swap batteries, a registered EBR communicates with BSS to share a session key SK as shown in Figure 3:

1. The EBR user enters its ID_{EBR} and PW_{EBR} on the smart glass. MC computes $HIP = H_2(ID_{EBR} \| PW_{EBR})$, $a_{MC} = HIP \oplus v$, and verifies whether $C \overset{?}{=} H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$. Assuming that verification was successful, MC generates a random number r_{MC} and computes $U_{EBR} = r_{MC} + sk_{EBR}, R = r_{MC} \cdot pk_{BSS}, CID_{EBR} = l' \oplus HIP = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR})$ and $Auth_{EBR} = H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{MC})$, where T_{MC} is the current timestamp. Next, using a public channel, EBR sends $\{Auth_{EBR}, CID_{EBR}, U_{EBR}, T_{MC}\}$ to BSS.
2. Once received the message, BSS verifies T_{MC}, calculates $H_2(sk_{BSS} \| ID_{EBR}) = CID_{EBR} \oplus H_1(sk_{BSS})$ and $R_{EBR} = U_{EBR} \cdot G - pk_{EBR} = r_{MC} \cdot G$ and $R^* = sk_{BSS} \cdot R_{EBR}$ and verifies whether $Auth_{EBR} \overset{?}{=} H_2(ID_{EBR} \| R^* \| CID_{EBR} \| T_{MC})$. Next, it generates a random number r_{BSS} and computes $R_{BSS} = r_{BSS} \cdot G$, $SK_{BSS} = r_{BSS} \cdot R_{EBR}$ and $Auth_{BSS} = H_2(ID_{EBR} \| R^* \| SK_{BSS} \| T_{BSS})$, where T_{BSS} is the current timestamp. Next, using a public channel, BSS sends $\{Auth_{BSS}, R_{BSS}, T_{BSS}\}$ to EBR.

3. EBR verifies T_{BSS}, calculates $SK_{EBR} = r_{MC} \cdot R_{BSS}$ and verifies whether $Auth^*_{BSS} \overset{?}{=} H_2(ID_{EBR} \| R \| SK_{EBR} \| T_{BSS})$ to authenticate BSS. After successful authentication, EBR calculates $sk = kdf(ID_{EBR} \| SK_{EBR} \| T_{MC} \| T_{BSS})$ and $Auth_{EB} = H_2(ID_{EBR} \| R \| sk \| T'_{MC})$, where T'_{MC} is the current timestamp. Next, using a public channel, EBR sends $\{Auth_{EB}, T'_{MC}\}$ to BSS.
4. BSS verifies T'_{MC} and computes $sk' = kdf(ID_{EBR} \| SK_{BSS} \| T_{MC} \| T_{BSS})$ and checks whether $Auth_{EB} \overset{?}{=} H_2(ID_{EBR} \| R^* \| sk' \| T'_{MC})$ to authenticate EBR and store the session key sk', which is used for secure communication between EBR and BSS.

EBR ($\{sk_{EBR}, l', v, C\}$)		**BSS** ($\{sk_{BSS}, H_2(sk_{BSS} \| ID_{EBR}), ID_{EBR}\}$)
User enters ID_{EBR} and PW_{EBR} on the smart glass. MC computes $HIP = H_2(ID_{EBR} \| PW_{EBR})$, $a_{MC} = HIP \oplus v$, and verifies $C \overset{?}{=} H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$, generates r_{MC} and computes $U_{EBR} = r_{MC} + sk_{EBR}, R = r_{MC} \cdot pk_{BSS}, CID_{EBR} = l' \oplus HIP = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR})$ and $Auth_{EBR} = H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{MC})$		
	$\xrightarrow{Auth_{EBR}, CID_{EBR}, U_{EBR}, T_{MC}}$	
		Verifies T_{MC}, calculates $H_2(sk_{BSS} \| ID_{EBR}) = CID_{EBR} \oplus H_1(sk_{BSS})$ and $R_{EBR} = U_{EBR} \cdot G - pk_{EBR} = r_{MC} \cdot G$ and $R^* = sk_{BSS} \cdot R_{EBR}$ and verifies $Auth_{EBR} \overset{?}{=} H_2(ID_{EBR} \| R^* \| CID_{EBR} \| T_{MC})$, generates r_{BSS} and computes $R_{BSS} = r_{BSS} \cdot G$, $SK_{BSS} = r_{BSS} \cdot R_{EBR}$ and $Auth_{BSS} = H_2(ID_{EBR} \| R^* \| SK_{BSS} \| T_{BSS})$
	$\xleftarrow{Auth_{BSS}, R_{BSS}, T_{BSS}}$	
Verifies T_{BSS}, calculates $SK_{EBR} = r_{MC} \cdot R_{BSS}$ and verifies $Auth^*_{BSS} \overset{?}{=} H_2(ID_{EBR} \| R \| SK_{EBR} \| T_{BSS})$, calculates $sk = kdf(ID_{EBR} \| SK_{EBR} \| T_{MC} \| T_{BSS})$ and $Auth_{EB} = H_2(ID_{EBR} \| R \| sk \| T'_{MC})$		
	$\xrightarrow{Auth_{EB}, T'_{MC}}$	
		Verifies T'_{MC}, calculates $sk' = kdf(ID_{EBR} \| SK_{BSS} \| T_{MC} \| T_{BSS})$ and verifies $Auth_{EB} \overset{?}{=} H_2(ID_{EBR} \| R^* \| sk' \| T'_{MC})$ to authenticate EBR
Sets $sk = kdf(ID_{EBR} \| SK_{EBR} \| T_{MC} \| T_{BSS})$ as the session key		Sets $sk' = kdf(ID_{EBR} \| SK_{BSS} \| T_{MC} \| T_{BSS})$ as the session key

Figure 3. Mutual authentication phase of LLAKEP scheme to share a session key between *EBR* and *BSS* [9].

2.8. Password Change

To change the current password, EBR enters its current ID_{EBR} and PW_{EBR} on the smart glass. MC computes $HIP = H_2(ID_{EBR} \| PW_{EBR})$, $a_{MC} = HIP \oplus v$, and verifies whether $C \overset{?}{=} H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$ to authenticate EBR. If authentication was successful; MC requests EBR to provide a new password PW_{new}. Then it computes $HIP_{new} = H_2(ID_{EBR} \| PW_{new})$, $v_{new} = HIP_{new} \oplus a_{MC}$, $_{new} = HIP_{new} \oplus b_{MC}$, $C_{new} = H_2(ID_{EBR} \| PW_{new} \| a_{MC})$ and $l_{new} = l' \oplus HIP \oplus HIP_{new} = H_1(sk_{BSS}) \oplus HIP_{new} \oplus H_2(sk_{BSS} \| ID_{EBR})$. At the end, EBR stores l'_{new} along v_{new} and C_{new} in MC.

3. On the Security of LLAKEP

LLAKEP has some interesting features, such as putting less computation on edge devices versus the server side. As a result of the authentication process, this protocol achieves low latency. Furthermore, the designers provided a formal security analysis in the random oracle model and validated its security with GNY logic. The disadvantage of such analysis may be its reliance on security analysis assumptions. Furthermore, in such attacks, primitives are regarded as ideal. For example, a hash function is considered one-way in those models, but if the hash function's input is chosen from a small domain, a dictionary attack on such low entropy input space may be possible. Such an attack is used in password-guessing attacks, such as [14,15]. As a result, it is worthwhile to conduct a third-party analysis of the protocol's exact security, focusing on more detailed attacks. These types of attacks are significant because of Zhang et al. considered the majority of these attacks to be disadvantages of the related protocols, see ([9], Table 1). As a result, it will be interesting to explicitly evaluate its security against such an attack, which we do in this section.

3.1. Insider Adversary

An insider adversary refers to a cyber-security risk that originates from within an organization. Consider a current or former employee, contractor, vendor, or partner with legitimate user credentials that misuse their access to the detriment of the organization's networks, systems, and data as an example of insider. A common privilege of an insider over an ordinary adversary is its access to the private channel. If such an adversary achieves a significant advantage to attack a protocol due to such access, that protocol suffers from insider adversaries.

We assume the adversary's target is to retrieve the user's credentials, i.e., ID_{EBR} and PW_{EBR}. Let us consider an adversary with access to the transferred messages over the public channel and also the stored values in the smart glass, i.e., $l' = H_1(sk_{BSS} \oplus HIP \oplus H_2(sk_{BSS} \| ID_{EBR}))$, $v = HIP \oplus a_{MC}$, and $C = H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$, where $HIP = H_2(ID_{EBR} \| PW_{EBR})$.

The transferred messages over the public channel are as follows, besides timestamps:

$$\begin{aligned} Auth_{EBR} =& H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{MC}) \\ CID_{EBR} =& l' \oplus HIP = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR}) \\ U_{EBR} =& r_{MC} + sk_{EBR} \\ Auth_{BSS} =& H_2(ID_{EBR} \| R \| SK_{BSS} \| T_{BSS}) \\ R_{BSS} =& r_{BSS} \cdot G \\ Auth_{EB} =& H_2(ID_{EBR} \| R \| sk \| T'_{MC}) \end{aligned}$$

where $R = sk_{BSS} \cdot r_{MC} \cdot G$ and $SK_{BSS} = r_{BSS} \cdot r_{MC} \cdot G$. Since a_{MC} is a random number, the adversary cannot receive low-entropy information from v and C. On the other hand, sk_{BSS} is the private key of BSS, and, due to that, l' does not help the adversary achieve the desired information to extract secret parameters using dictionary attacks. However, combining that information with the transfused messages provides the adversary with some gains. Let's consider the case where the adversary has l' from reading the smart glass's memory and also eavesdrops on $CID_{EBR} = l' \oplus HIP$ from the public channel. Hence, the adversary achieves $HIP = CID_{EBR} \oplus l'$. Assuming that the entropy of ID_{EBR} and PW_{EBR} is, respectively, $\mathcal{H}_{ID}$ and $\mathcal{H}_{PW}$, then the expected complexity to drive ID_{EBR} and PW_{EBR} using dictionary attack is $2^{\mathcal{H}_{ID}+\mathcal{H}_{PW}}$.

Now, let us consider an insider adversary with access to the transferred messages over the public channel during the user registration phase, i.e., $\{pk_{EBR}, ID_{EBR}, d\}$ to BSS, where $d = HIP \oplus b_{MC}$. Let's assume the adversary also eavesdrops on a session and steals the smart glass. Given the transferred messages over the public channel, $HIP = H_2(ID_{EBR} \| PW_{EBR})$ is achieved, and the expected complexity of extracting PW_{EBR} using a dictionary attack is $2^{\mathcal{H}_{PW}}$. Given ID_{EBR} and PW_{EBR}, the adversary can use the stolen smart glass at any time to impersonate the target EBR. Compared with a generic adversary whose cost was $2^{\mathcal{H}_{ID}+\mathcal{H}_{PW}}$, an insider adversary has a significant advantage.

3.1.1. Traceability and Anonymity

Assume a protocol party participated in two different protocol sessions, say at times T and T′. In such a protocol, the target party is traceable whenever the adversary can link the transferred messages over those sessions with a high probability. If a party is transferred over the public channel, its constant identifier is a common way to track it down. Hiding such information, however, does not ensure the protocol's security against this attack. In more detail, during the authentication phase of LLAKEP, the EBR sends $\{Auth_{EBR}, CID_{EBR}, U_{EBR}, T_{MC}\}$ to BSS, where $CID_{EBR} = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR})$. This information is consistent for any EBR user and, with a high probability, unique. Assuming that the adversary eavesdropped on a session between the *i*th user, EBR_i, and BSS and saved CID_{EBR_i}, assuming that it monitors another session between the *i*th user, EBR_i, and BSS, if $CID_{EBR_i} \neq CID_{EBR_j}$ then $EBR_i \neq EBR_j$ and if $CID_{EBR_i} = CID_{EBR_j}$ then $EBR_i = EBR_j$. This means that any EBR can be traced across multiple sessions in LLAKEP.

3.1.2. Known Session-Specific Temporary Information Attack

Assume the adversary is aware of the session-specific temporary values of LLAKEP, i.e., r_{MC} and r_{BSS}, as well as the messages transferred over the public channel. The message transferred from EBR to BSS includes $U_{EBR} = r_{MC} + sk_{EBR}$, and given r_{MC}, the adversary achieves $sk_{EBR} = U_{EBR} - r_{MC}$, which is EBR's secret key. Given r_{MC}, the adversary also computes $R = r_{MC} \cdot G$. $Auth_{EBR} = H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{MC})$, on the other hand, is also available from the public channel, and ID_{EBR} has low entropy, e.g., $\mathcal{H}_{ID}$. As a result, the adversary can extract ID_{EBR} with a complexity of $2^{\mathcal{H}_{ID}}$. The session key is computed as $sk = kdf(ID_{EBR} \| SK_{EBR} \| T_{MC} \| T_{BSS})$ and $SK_{EBR} = r_{MC} \cdot R_{BSS}$ and R_{BSS} is transferred over the public channel. As a result, the first known session-specific temporary information (KSTI) attack on LLAKEP has a complexity of $2^{\mathcal{H}_{ID}}$, while the latter KSTI attacks have negligible complexities.

3.1.3. Impersonation Attack after a Successful KSTI Attack

Assume the opponent successfully launched a KSTI attack on LLAKEP. Following the attack, the adversary has ID_{EBR}, sk_{EBR} and $CID_{EBR} = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR})$. It is obvious that the adversary can do the following:

1. The adversary generates a random number r_{adv} and computes $U_{EBR} = r_{adv} + sk_{EBR}$, $R_{adv} = r_{adv} \cdot pk_{BSS}$ and $Auth_{EBR} = H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{adv})$, where T_{adv} is the current timestamp. Next, using a public channel, the adversary sends $\{Auth_{EBR}, CID_{EBR}, U_{EBR}, T_{MC}\}$ to BSS.
2. Obviously T_{MC} and $Auth_{EBR}$ are accepted by BSS and $R_{adv} = r_{adv} \cdot pk_{BSS}$ is extracted by BSS from the received $U_{EBR} = r_{adv} + sk_{EBR}$. Next, it generates a random number r_{BSS} and computes $R_{BSS} = r_{BSS} \cdot G$, $SK_{BSS} = r_{BSS} \cdot R_{avd}$ and $Auth_{BSS} = H_2(ID_{EBR} \| R_{adv} \| SK_{BSS} \| T_{BSS})$ and sends $\{Auth_{BSS}, R_{BSS}, T_{BSS}\}$ to EBR (impersonated by the adversary).
3. The adversary calculates $SK_{adv} = r_{adv} \cdot R_{BSS}$, $sk = kdf(ID_{EBR} \| SK_{adv} \| T_{adv} \| T_{BSS})$ and $Auth_{ad} = H_2(ID_{EBR} \| R_{adv} \| SK_{adv} \| T'_{adv})$, where T'_{adv} is the current timestamp. Next, using a public channel, the adversary sends $\{Auth_{ad}, T'_{adv}\}$ to BSS.
4. BSS verifies T'_{adv}, calculates $sk' = kdf(ID_{EBR} \| SK_{BSS} \| T_{MC} \| T_{BSS})$ and verifies whether $Auth_{ad} \stackrel{?}{=} H_2(ID_{EBR} \| R_{adv} \| sk' \| T'_{MC})$ to authenticate EBR/adversary and store the session key sk', which is used for the secure communication between EBR/adversary and BSS.

Since $SK_{BSS} = r_{BSS} \cdot R_{adv} = r_{BSS} \cdot r_{adv} \cdot G = r_{adv} \cdot R_{BSS} = SK_{adv}$, the authentication is completed successfully which confirms the impersonation attack on LLAKEP.

3.2. Key Compromised Impersonation Attack

Assume a client $\mathcal{C}$ is in contact with a server $\mathcal{S}$. Assuming that all the secret parameters of $\mathcal{C}$ (resp. $\mathcal{S}$) are given to the adversary, the adversary should not be able to impersonate $\mathcal{S}$ (resp. $\mathcal{C}$) toward $\mathcal{C}$ (resp. $\mathcal{S}$); otherwise, the protocol is vulnerable to key compromise impersonation (KCI) attacks [16–18]. This attack is a variant of the Man in the Middle (MitM) attack and has been successfully applied to TLS. For instance, in the KCI-based MitM attack against TLS [19], an attacker with a client certificate's private key installed on a victim can impersonate any server. Hence, in this section, we investigate LLAKEP's security against this type of attack.

Assume the adversary is given all of the user's information, i.e., the stored information in the smart glass: $l' = H_1(sk_{BSS}) \oplus HIP \oplus H_2(sk_{BSS} \| ID_{EBR})$, $v = HIP \oplus a_{MC}$, $C = H_2(ID_{EBR} \| PW_{EBR} \| a_{MC})$, where $HIP = H_2(ID_{EBR} \| PW_{EBR})$, the user's credentials which are ID_{EBR} and PW_{EBR} and the stored information in the MC's memory which is sk_{EBR}. The adversary does the following to impersonate BSS:

1. The EBR user enters its ID_{EBR} and PW_{EBR} on the smart glass. MC verifies them, generates a random number r_{MC} and computes $U_{EBR} = r_{MC} + sk_{EBR}, R = r_{MC} \cdot pk_{BSS}, CID_{EBR} = l' \oplus HIP = H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR})$ and $Auth_{EBR} = H_2(ID_{EBR} \| R \| CID_{EBR} \| T_{MC})$ and sends $\{Auth_{EBR}, CID_{EBR}, U_{EBR}, T_{MC}\}$ to BSS.
2. The adversary extracts $r_{MC} = U_{EBR} - sk_{EBR}$, computes $R_{EBR} = r_{MC} \cdot G$ and $R^* = r_{MC} \cdot pk_{BSS}$, generates a random number r_{adv}, computes $R_{adv} = r_{adv} \cdot G$, $SK_{adv} = r_{adv} \cdot R_{EBR}$ and $Auth_{adv} = H_2(ID_{EBR} \| R \| SK_{adv} \| T_{adv})$ and sends $\{Auth_{adv}, R_{adv}, T_{adv}\}$ to EBR.
3. EBR verifies T_{adv}, calculates $SK_{EBR} = r_{MC} \cdot R_{adv}$ and verifies whether $Auth^*_{adv} \stackrel{?}{=} H_2(ID_{EBR} \| R \| SK_{EBR} \| T_{adv})$ to authenticate BSS/adversary which authenticates.

Following the aforementioned attack, the adversary could impersonate BSS toward EBR, assuming it has access to EBR's secret parameters but no knowledge of BSS secrets.

3.3. The Lack of Perfect Forward Secrecy

A security protocol provides perfect forward secrecy if compromising a protocol participant's long-term secrets at time T^j does not affect the security of the shared session keys at any time $T^i < T^j$ [10]. It is worth noting that Zhang et al. considered this attack when comparing related protocols ([9], Table 1). As a result, it will be interesting to explicitly evaluate LLAKEP's security against such an attack, which we do in this section. We assume the adversary intercepted the following messages over the public channel at time T^i, along with the timestamps T^i_{MC}, T^i_{BSS} and T'^i_{MC}:

$$\begin{aligned}
Auth_{EBR} &= H_2(ID_{EBR} \| R \| CID_{EBR} \| T^i_{MC}) \\
CID_{EBR} &= H_1(sk_{BSS}) \oplus H_2(sk_{BSS} \| ID_{EBR}) \\
U^i_{EBR} &= r^i_{MC} + sk_{EBR} \\
Auth_{BSS} &= H_2(ID_{EBR} \| R^i \| SK_{BSS} \| T^i_{BSS}) \\
R^i_{BSS} &= r^i_{BSS} \cdot\ G \\
Auth_{EB} &= H_2(ID_{EBR} \| R \| sk^i \| T'^i_{MC})
\end{aligned}$$

where $sk^i = kdf(ID_{EBR} \| SK_{BSS} \| T^i_{MC} \| T^i_{BSS})$ and $SK_{BSS} = r^i_{MC} \cdot R^i_{BSS}$. Obviously, given $U^i_{EBR} = r^i_{MC} + sk_{EBR}$, R^i_{BSS} and sk_{EBR}, the adversary is able to compute $r^i_{MC} = U^i_{EBR} - sk_{EBR}$ and $SK_{BSS} = r^i_{MC} \cdot R^i_{BSS}$. Next, given the revealed ID_{EBR}, the eavesdropped T^i_{MC} and T^i_{BSS} and the computed SK_{BSS} the adversary can extract the session key of the ith session, i.e., $sk = kdf(ID_{EBR} \| SK_{BSS} \| T^i_{MC} \| T^i_{BSS})$. Hence, LLAKEP does not provide perfect forward secrecy.

3.4. A Note on the LLAKEP Efficiency

The efficiency of the protocol should be considered when designing the messages. BSS stores $\{H_2(sk_{BSS}\|ID_{EBR}), ID_{EBR}\}$ during the LLAKEP registration phase, and during the authentication phase, EBR sends $CID_{EBR} = l' \oplus HIP = H_1(sk_{BSS}) \oplus H_2(sk_{BSS}\|ID_{EBR})$ as a part of the message, which is used to identify the target EBR's records. In this regard, BSS first computes $H_1(sk_{BSS}) \oplus CID_{EBR} = H_2(sk_{BSS}\|ID_{EBR})$ and searches its database for related records. This means that BSS should compute this value for all authentications. However, if it stores $H_1(sk_{BSS}) \oplus H_2(sk_{BSS}\|ID_{EBR})$ in the registration phase rather than $\{H_2(sk_{BSS}\|ID_{EBR})\}$, that computation is not required, and the resulted protocol could be more efficient.

4. LLAKEP$^+$ Description

LLAKEP$^+$, the proposed protocol, includes initialization, user registration, authenticated key agreement, and password change. While designing the proposed protocol, we avoided the weaknesses in LLAKEP. To avoid insider attacks, we never send ID_{EBR} to BSS in plain text. Furthermore, we avoid using $U_{EBR} = r_{MC} + sk_{EBR}$ because it could be a source for a KSTI attack. We will include both long-term and ephemeral keys to provide security against KSTI and KCI.

4.1. Initialization

During the startup phase, the battery swap station (BSS) selects the system parameters, which are $\{E/F_p, G, n, pk_{BSS}, H(\cdot)\}$, where $pk_{BSS} = sk_{BSS} \cdot G$. The only difference in this phase is that a single hash function $H(\cdot)$ is used instead of two, which might be any one-way cryptographic hash function. We suppose n is the order of the group $\mathcal{G}$'s basis point G, and the output of $H(\cdot)$ may be transformed to an integer in $0, \ldots, n-1$. In other circumstances, though, we may want lengthier output from the hash function to mask a pattern, in which case we use $H^e(\cdot)$.

4.2. User Registration

During the registration phase, we use the Elliptic Curve Digital Signature Algorithm (ECDSA) [20], which is a version of the Digital Signature Algorithm (DSA). During this phase, EBR chooses its ID_{EBR} and PW_{EBR} as randomly as possible and enters them into the smart glass containing a microchip MC. The embedded MC generates a random number a_{MC} and computes $HID = H(ID_{EBR}\|a_{MC})$. Next, using a secure channel, EBR submits $\{pk_{EBR}, HID\}$ to BSS. It selects a random integer $k \in [1, n-1]$ and calculates $k \cdot G = (x, y)$ and sets $r = x \ mod \ n$. If $r = 0$, BSS selects another random number. Next it computes $s = k^{-1}(HID + r \times sk_{BSS}) \ mod \ n$ and sends (r, s) to EBR. (HID, pk_{EBR}) is stored in a secure Non Volatile Memory (NVM) of BSS also.

Once received (r, s), EBR computes $u_1 = HID \times s^{-1}$ and $u_2 = r \times s^{-1}$ and $(x'_1, y'_1) = u_1 \cdot G + u_2 \cdot pk_{BSS}$. Next, it verifies whether $r \overset{?}{=} x' \ mod \ n$ to confirm the registration. After that, MC computes $d = H^e(ID_{EBR}\|PW_{EBR}) \oplus (a_{MC}\|s\|r)$ and $v = H(s\|r\|H(ID_{EBR}\|PW_{EBR}\|a_{MC})$ and stores (d, v) in the smart glass's NVM.

4.3. Authenticated Key Agreement

A registered EBR connects with BSS to share a session key sk to switch batteries, as shown in Figure 4:

1. The EBR user enters its ID_{EBR} and PW_{EBR} on the smart glass. MC computes $(a_{MC}\|s\|r) = H^e(ID_{EBR}\|PW_{EBR}) \oplus d$, and $HID = H(ID_{EBR}\|a_{MC})$ and verifies whether $v \overset{?}{=} H(s\|r\|H(ID_{EBR}\|PW_{EBR})\|a_{MC})$ to accept the login. If the verification was successful, MC obtains the current time T_{MC}, generates a random number $k_{MC} \in [1, n-1]$ and calculates $R_{EBR} = k_{MC} \cdot G$ and $Auth_{EBR} = HID \oplus H(k_{MC} \cdot pk_{BSS}\|T_{MC})$. Then, it sends $(R_{EBR}, T_{MC}, Auth_{EBR})$ to BSS over a public channel.

2. BSS validates T_{MC} after receiving the message and calculates $HID = Auth_{EBR} \oplus H(sk_{BSS} \cdot R_{EBR} \| T_{MC})$. If it detects the extracted HID in its database, BSS generates a random number $k_{BSS} \in [1, n-1]$ and calculates $R_{BSS} = k_{BSS} \cdot G$, the temporary session key $sk = H(k_{BSS} \cdot R_{EBR} \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$ and $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$. Then, it sends $(R_{BSS}, Auth_{BSS})$ to EBR over a public channel.
3. EBR computes the session key, after receiving the message, $sk = H(k_{MC} \cdot R_{BSS} \| sk_{EBR} \cdot R_{EBR} \| k_{MC} \cdot pk_{BSS})$ and verifies whether $Auth_{BSS} \overset{?}{=} H(R_{BSS} \| T_{MC} \| sk)$ to authenticate BSS. If BSS has been authenticated, EBR returns $V_{EBR} = H(sk \| T_{MC} \| HID)$ to BSS.
4. BSS verifies the received V_{EBR} to authenticate EBR and also confirm the shared session key.

EBR ($\{sk_{EBR}, l', v, C\}$)		**BSS** ($\{sk_{BSS}, H_2(sk_{BSS} \| ID_{EBR}), ID_{EBR}\}$)
The EBR user enters its ID_{EBR} and PW_{EBR} on the smart glass. MC computes $(a_{MC} \| s \| r) = H^e(ID_{EBR} \| PW_{EBR}) \oplus d$, and $HID = H(ID_{EBR} \| a_{MC})$ and verifies $v \overset{?}{=} H(s \| r \| H(ID_{EBR} \| PW_{EBR}) \| a_{MC})$ to accept the login and get T_{MC}, generates a random number $k_{MC} \in [1, n-1]$ to calculate $R_{EBR} = k_{MC} \cdot G$ and $Auth_{EBR} = HID \oplus H(k_{MC} \cdot pk_{BSS} \| T_{MC})$		
	$\xrightarrow{R_{EBR}, T_{MC}, Auth_{EBR}}$	
		Verifies T_{MC}, calculates $HID = Auth_{EBR} \oplus H(sk_{BSS} \cdot R_{EBR} \| T_{MC})$. If HID is valid, generates a random number $k_{BSS} \in [1, n-1]$, computes $R_{BSS} = k_{BSS} \cdot G$ and $sk = H(k_{BSS} \cdot R_{EBR} \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$ and $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$
	$\xleftarrow{R_{BSS}, Auth_{BSS}}$	
Calculates $sk = H(k_{MC} \cdot R_{BSS} \| sk_{EBR} \cdot R_{EBR} \| k_{MC} \cdot pk_{BSS})$, verifies $Auth_{BSS} \overset{?}{=} H(R_{BSS} \| T_{MC} \| sk)$ to authenticate BSS and return $V_{EBR} = H(sk \| T_{MC} \| HID)$		
	$\xrightarrow{V_{EBR}}$	
		Verifies V_{EBR} to authenticate EBR and also confirm sk

Figure 4. Mutual authentication phase of LLAKEP$^+$ scheme to share a session key between EBR and BSS.

4.4. Password Change

To change the current password, EBR enters its current ID_{EBR} and PW_{EBR} on the smart glass and requests a run of the password change phase. MC computes $(a_{MC} \| s \| r) = H^e(ID_{EBR} \| PW_{EBR}) \oplus d$, and $HID = H(ID_{EBR} \| a_{MC})$ and verifies whether $v \overset{?}{=} H(s \| r \| HID \| a_{MC})$ to accept the login. Next, the user inputs the new password PW^{new}_{EBR}. After that, MC computes $d^{new} = H^e(ID_{EBR} \| PW^{new}_{EBR}) \oplus (a_{MC} \| s \| r)$ and $v^{new} = H(s \| r \| H(ID_{EBR} \| PW^{new}_{EBR}) \| a_{MC})$ and stores (d^{new}, v^{new}) in the smart glass's NVM and sends the password successfully updated message.

5. Security and Cost Analysis of LLAKEP$^+$

We argue the security and cost analysis of LLAKEP$^+$ against various attacks and form different aspects in this section.

5.1. Informal Security Analysis

Through the analysis, we consider two types of adversaries: the first is a common adversary with access to the transferred messages over the public channels, i.e., R_{EBR}, T_{MC}, $Auth_{EBR}$, R_{BSS}, $Auth_{BSS}$, V_{EBR}, where:

$$\begin{aligned}
R_{EBR} =& k_{MC} \cdot\ G \\
Auth_{EBR} =& HID \oplus H(k_{MC} \cdot pk_{BSS} \| T_{MC}) \\
R_{BSS} =& k_{BSS} \cdot\ G \\
sk =& H(k_{BSS} \cdot R_{EBR} \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR}) \\
Auth_{BSS} =& H(R_{BSS} \| T_{MC} \| sk) \\
V_{EBR} =& H(sk \| TMC \| HID)
\end{aligned}$$

and $k_{MC} \in [1, n-1]$ and $k_{BSS} \in [1, n-1]$. This adversary's primary goal is to conduct any type of attack, such as tracking an EBR, impersonating it, gaining access to shared secrets, and so on. The second type of adversary, on the other hand, is a privileged adversary, which is more applicable in the CK model. This adversary has access to the secure channel during the registration phase, the ephemeral secrets, the long-term secrets, and so on, depending on the attack type. This adversary's goal, for example, is to gain access to the secret keys.

5.1.1. Replay Attack

To conduct a replay attack, the adversary must be able to reuse the eavesdropped messages transferred in session i in a subsequent session. The use of a timestamp in the calculation of $Auth_{EBR} = HID \oplus H(k_{MC} \cdot pk_{BSS} \| T_{MC})$, $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$ and $V_{EBR} = H(sk \| T_{MC} \| HID)$ ensures the time-freshness of the transferred messages in LLAKEP$^+$. As a result, this protocol is resistant to replay attacks.

5.1.2. Impersonation Attack

A passive adversary can perform an impersonation attack, assuming it can perform a replay attack, and Section 5.1.1 demonstrates that LLAKEP$^+$ is secure against replay attacks. As a result, a passive adversary cannot impersonate either EBR or BSS. An active adversary, on the other hand, should forge a valid tuple $R_{EBR}, T_{MC}, Auth_{EBR}, V_{EBR}$ or $R_{BSS}, Auth_{BSS}$ such that related $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$ or $V_{EBR} = H(sk \| T_{MC} \| HID)$ is also verified by the receiver. Assume the adversary is attempting to impersonate EBR at time T_{MC}. The adversary could do this if it can compute a valid $Auth_{EBR} = HID \oplus H(k_{MC} \cdot pk_{BSS} \| T_{MC})$ and return the expected $V_{EBR} = H(sk \| T_{MC} \| HID)$. Assuming the adversary does not have access to HID or sk_{EBR}, it cannot forge those values. As a result, it cannot compute a valid tuple R_{EBR}, T_{MC}, V_{EBR} to impersonate EBR. Similarly, to impersonate BSS, the adversary must be able to return a valid $R_{BSS}, Auth_{BSS}$ given the EBR's challenge tuple $(R_{EBR}, T_{MC}, Auth_{EBR})$. To do so, however, the adversary must either have sk_{BSS} or compromise ECDLP or EC-CDHP, which is not possible. As a result, LLAKEP$^+$ protects against impersonation attacks.

5.1.3. Traceability and Anonymity

To trace an object that is participating in an authentication protocol, it should be feasible to relate the transmitted messages over the public channel with a non-zero probability to the object's identity/unique-parameters. Except for the timestamp, which cannot be used directly to track any object, all parameters in the proposed protocol are either fresh values or distinguished by a one-way hash function or ECC point multiplication. $R_{EBR} = k_{MC} \cdot G$ and $R_{BSS} = k_{BSS} \cdot G$ are fresh values, $Auth_{EBR} = HID \oplus H(k_{MC} \cdot pk_{BSS} \| T_{MC})$ is masked by $H(k_{MC} \cdot pk_{BSS} \| T_{MC})$ and $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$ and a one-way hash function is used to compute $V_{EBR} = H(sk \| T_{MC} \| HID)$. If the adversary can extract HID from $Auth_{EBR}$; it will be able to trace the EBR or BSS. To do so, however, the adversary must either have access to sk_{BSS} or compute $k_{MC} \cdot pk_{BSS}$, as $R_{EBR} = k_{MC} \cdot G$ necessitates compromising ECDLP, which is not feasible for sufficiently large n. As a result, LLAKEP$^+$ protects against traceability attacks.

5.1.4. Secret Disclosure Attack

The EBR secret parameters are ID_{EBR}, PW_{EBR}, HID and sk_{EBR}, whereas the BSS secret parameter is sk_{BSS} and the shared session key is $sk = H(k_{BSS} \cdot k_{MC} \cdot G \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$. The attacker clearly cannot get PW_{EBR} from the parameters transmitted over the public channel. The only parameter that contains ID_{EBR}, on the other hand, is $HID = H_2(ID_{EBR} \| a_{MC})$, which is distinguished by a random value a_{MC}. Furthermore, HID is masked by $H(k_{MC} \cdot pk_{BSS} \| T_{MC})$ and cannot be calculated without knowledge of sk_{BSS} or solving the ECDLP issue. sk_{EBR} and sk_{BSS} are also secret keys that only the owner knows about. To compute sk, the adversary must first compute $k_{BSS} \cdot k_{MC} \cdot G$ given $k_{MC} \cdot G$ and $k_{BSS} \cdot G$, which requires overcoming the ECDLP problem once more. As a result, we conclude that LAKEP$^+$ protects against secret disclosure attacks.

5.1.5. Permanent De-Synchronization Attack

In the suggested protocol, successfully updating the password is the only way to desynchronize a valid EBR from its MC. However, it necessitates a login, the complexity of which is $2^{\mathcal{H}_{ID}+\mathcal{H}_{PW}}$ for an adversary without access to ID_{EBR} and PW_{EBR}. Assuming $\mathcal{H}_{ID} = \mathcal{H}_{PW} = 20$, the total complexity is 2^{40}, which is impractical.

5.1.6. Man-in-the-Middle Attack

To imitate EBR and BSS as a man-in-the-middle adversary, the adversary needs to either perform a replay attack or actively alter the transmitted messages. A PPTA adversary has no possibility of carrying out such assaults if the arguments in Sections 5.1.1 and 5.1.2 are followed. As a result, it ensures the proposed protocol's security against man-in-the-middle attacks.

5.1.7. Stolen Smart Glass Attack

Consider the loss or adversarial access to the smart glass. In this case, the adversary can read the smart glass memory and access the stored values, namely $d = H^e(ID_{EBR} \| PW_{EBR}) \oplus (a_{MC} \| s \| r)$ and $v = H(s \| r \| H(ID_{EBR} \| PW_{EBR} \| a_{MC})$. To obtain any information, it must guess ID_{EBR} and PW_{EBR} at the same time, which costs $2^{\mathcal{H}_{ID}+\mathcal{H}_{PW}}$.

5.1.8. Insider Adversary

The primary advantage of a privileged insider opponent over a naive attacker is access to data exchanged across a secure channel. The single secret channel in the proposed protocol is utilized during the registration phase, and according to Section 4.2, the data exchanged over this channel are (HID, pk_{EBR}) sent by the EBR and (r, s) sent by the BSS. Although HID is valuable information, the adversary cannot use it to extract ID_{EBR} or PW_{EBR} or impersonate EBR since it also requires the secret sk_{EBR}. As a result, the suggested protocol is safe against insider threats.

5.1.9. Perfect Forward Secrecy

If access to the long-term secrets, such as sk_{EBR} and sk_{BSS}, does not compromise the security of the prior session keys, a protocol is assumed to guarantee complete forward secrecy. The session key in LLAKEP$^+$ is computed as $sk = H(k_{BSS} \cdot k_{MC} \cdot G \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$, where k_{BSS} and k_{MC} are ephemeral session dependent fresh values. Although the adversary has access to $k_{MC} \cdot G$ and $k_{BSS} \cdot G$, to compute sk, the adversary must first compute $k_{BSS} \cdot k_{MC} \cdot G$, which requires overcoming the ECDLP issue. As a result, we conclude that LLAKEP$^+$ offers complete forward secrecy.

5.1.10. Known Session-Specific Temporary Information Attack

In LLAKEP$^+$, the session-specific temporary information is the timestamps and k_{BSS} and k_{MC}. However, to compute the session key, the adversary must first compute $sk_{BSS} \cdot R_{EBR}$ and $sk_{EBR} \cdot R_{BSS}$ given $R_{EBR} = k_{MC} \cdot G$, $R_{BSS} = k_{BSS} \cdot G$, pk_{EBR} and pk_{BSS} which again needs to overcome ECDLP problem. Hence we conclude that LLAKEP$^+$ provides security against Known session-specific temporary information attacks.

5.1.11. Key Compromised Impersonation Attack

To mimic BSS in a KCI attack, the adversary must be able to return a valid $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$. However, the adversary must compute valid sk, which is a factor of $sk_{BSS} \cdot R_{EBR}$. Given $R_{EBR} = k_{MC} \cdot G$ and pk_{BSS}, solve the ECDLP issue is required to compute $sk_{BSS} \cdot R_{EBR}$. As a result, we conclude that LLAKEP$^+$ protects against key compromised impersonation attacks.

5.2. Formal Security Evaluation

5.2.1. Scyther

Scyther is a tool for formal analysis of security protocols under the assumption of perfect cryptography, where it is assumed that all functions and cryptographic primitives used are perfect in terms of cryptographic properties. In the sense that the defined symmetric and asymmetric encryption functions are secure enough, that is, an adversary cannot gain anything from a ciphertext unless s/he knows the decryption key. Both hash and one-way functions have all the features of a secure hash function. This means that the adversary cannot reach the input of the one-way functions, such as the hash function and PRNG, from their output.

It should be noted that most of the time, protocol designers use the Scyther tool to find and fix problems caused by the way the protocol is made. In practice, many protocol problems can be found using the Scyther tool and either prove their authenticity or find attacks.

Standard Version and Compromise Version Comparison:

The difference between these two versions is that in the standard model, the adversary model is the Dolo–Yao model, and in the Compromise version, in addition to the Dolo–Yao model is also possible to check forward secrecy scenarios. The meaning of forward secrecy is that if the main and long-term keys of the parties participating in the protocol are revealed to the adversary, s/he cannot obtain the values of the temporary session keys that were used in the previous meetings. There is a concept of backward secrecy in security discussions, which means that if the main and long-term keys of the parties participating in the protocol are revealed to the attacker, s/he will not be able to obtain the values of the temporary session keys used in future meetings. A protocol that has both backward and forward secrecy properties is called a perfect secure protocol.

By using the Compromise version of the Scyther, the security of the discussed protocol can be seen in the following attacks:

1. Suppose that the long-term key is revealed to the adversary, what attack scenarios are the protocol vulnerable to?
2. Assume the session key is exposed, what attack scenarios are the protocol vulnerable to?
3. Suppose that the protocol state is exposed, what attack scenarios are the protocol vulnerable to?

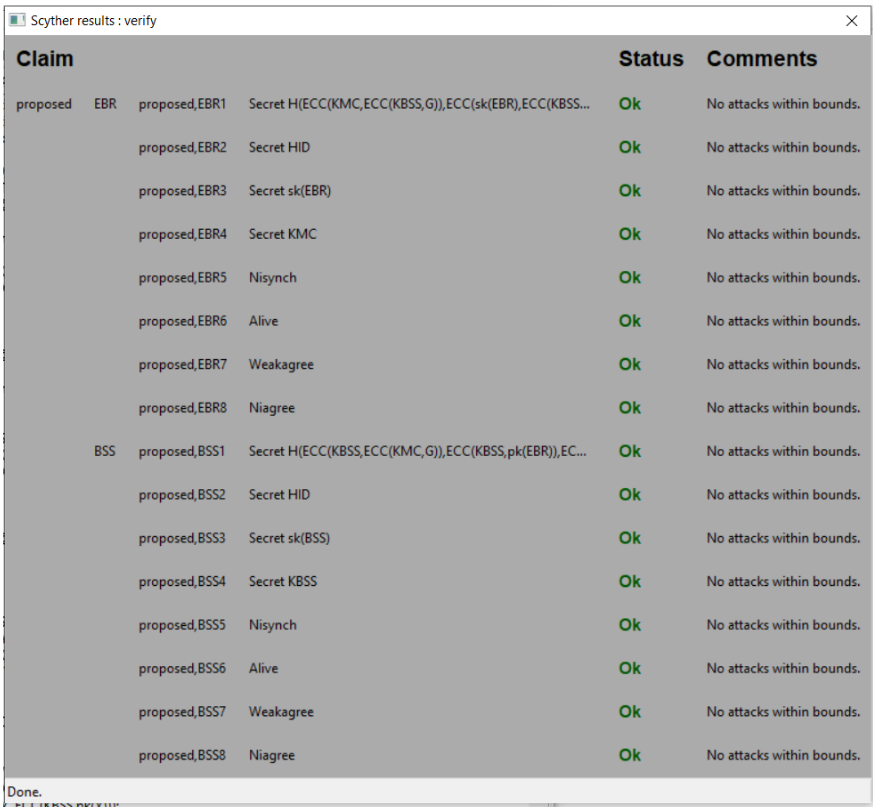
Security Claims of Scyther Tool:

Security goals in each application are defined as three important principles of confidentiality, integrity, and availability. In the Scyther tool, the designers have used these three important principles in the form of two properties, Secrecy and Authentication, with the following definitions:

Secrecy: states that certain confidential and secret information will not be revealed to the adversary. Different forms of secrecy can be defined by subtle differences. In the Scyther tool, a secrecy claim event is written as an example $claim(R, Secret, S)$, which is executed in the role of R, which includes the expression S as a secret parameter. This assertion states whether, for all implementations of the protocol role, the S statement remains secret, i.e., it remains unknown to the adversary or not. There is another security claim related to secrecy which is SKR. If we assume that we want to verify the confidentiality of the S in the role of I with this claim, we should write $Claim(I, SKR, S)$. The purpose of this assertion is to consider the desired parameter as a session key. The result of this issue is that by using the Reveal session-key rule that exists in the Compromise version of the Scyther tool, the secret parameter phrase written in the SKR claim, i.e., S, is considered as the session key. It should be noted that if the Reveal session-key rule is not active in the Compromise version of the Scyther tool, the SKR claim works the same as the $Secret$ claim.

Authentication: The most studied security feature in the field of security protocol analysis is authentication. However, contrary to the claim of confidentiality, there is no general consensus on the meaning of authentication. In fact, as Lowe showed in [21], there is a hierarchy of authentication features. Authentication focuses on the fact that the implementation of a protocol role actually guarantees that there is at least one communication partner in the network. In most cases, we want to establish a stronger objective, i.e., that the intended partner is aware of our communication and that a protocol is being implemented, and that messages have been exchanged as expected and according to the protocol. These hierarchies are expressed as $Aliveness$, $Synchronization$, and $Agreement$ properties in the Scyther tool. The interested reader can refer to [21,22] to learn more about these security claims.

To verify the security of LLAKEP$^+$ formally, we used Scyther tool [23]. For this purpose, the protocol is first modeled in SPDL, and then its security is evaluated. The SPDL code and the security verification results are depicted in Appendix A and Figure 5, respectively, which confirms the security of the proposed protocol.


Scyther results : verify

Claim				Status	Comments
proposed	EBR	proposed,EBR1	Secret H(ECC(KMC,ECC(KBSS,G)),ECC(sk(EBR),ECC(KBSS...	Ok	No attacks within bounds.
		proposed,EBR2	Secret HID	Ok	No attacks within bounds.
		proposed,EBR3	Secret sk(EBR)	Ok	No attacks within bounds.
		proposed,EBR4	Secret KMC	Ok	No attacks within bounds.
		proposed,EBR5	Nisynch	Ok	No attacks within bounds.
		proposed,EBR6	Alive	Ok	No attacks within bounds.
		proposed,EBR7	Weakagree	Ok	No attacks within bounds.
		proposed,EBR8	Niagree	Ok	No attacks within bounds.
	BSS	proposed,BSS1	Secret H(ECC(KBSS,ECC(KMC,G)),ECC(KBSS,pk(EBR)),EC...	Ok	No attacks within bounds.
		proposed,BSS2	Secret HID	Ok	No attacks within bounds.
		proposed,BSS3	Secret sk(BSS)	Ok	No attacks within bounds.
		proposed,BSS4	Secret KBSS	Ok	No attacks within bounds.
		proposed,BSS5	Nisynch	Ok	No attacks within bounds.
		proposed,BSS6	Alive	Ok	No attacks within bounds.
		proposed,BSS7	Weakagree	Ok	No attacks within bounds.
		proposed,BSS8	Niagree	Ok	No attacks within bounds.

Done.

Figure 5. Security verification results of LLAKEP$^+$ on Scyther.

5.2.2. Formal Security Analysis in RoR Model

Assume that two parties want to share a session key sk using an authenticated key agreement protocol, as proposed by Abdola et al. [24]. In such a protocol, those parties use long-term secrets along with ephemeral values to genera Sk. In our case, we consider the parties to be EBR and BSS. The adversary $\mathcal{A}$ seeks to distinguish between a real protocol (RP) and a random protocol or world (RW). To determine its ability, on distinguishing a real protocol from a random one, a bit (b) is chosen randomly at the beginning of the experiment, where $b = 0$ defines the random world (RW) and $b = 1$ represents the real protocol or world. Following the proposed adversary's model in Section 2.3, $\mathcal{A}$ can do several query types, also see [24,25], to distinguish the real world from the random world. The first query type, Exe, represents a passive adversary $\mathcal{A}$. The second query is Send, which represents an active opponent $\mathcal{A}$. The last query is Reveal, which outputs the session key held by a party. Finally, if $b = 1$, Test returns the session key; otherwise, it returns a random key of the same size. The adversary returns its decision as b_0 at the end of the game and wins if $b_0 = b$, where b is the hidden bit used in Test. $Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R)$, which defines the semantic security in the real-or-random (RoR) model, is defined as follows:

$$Adv^{RoR}_{RW\to RP} \quad = \quad \big((Pr(\mathcal{A} \to b_0 = 1 : b = 1) - (Pr(\mathcal{A} \to b_0 = 1 : b = 0))\big)$$

If the adversary's advantage to win this game is negligible, the target protocol provides RoR semantic security, i.e.:

$$Adv^{RoR}_{RW\to RP} < \varepsilon(.)$$

and $\varepsilon(.)$ is some negligible function of the protocol's parameters [25].

To bound the security level of LLAKEP$^+$, we use the above-mentioned Real or Random (RoR) model. This model bounds the adversary's advantage to distinguish between two worlds, the random world (RW) in which all protocol messages are randomly generated but respecting the message structure of the target protocol and the real world in which those messages are generated by entities in the real protocol (RP), i.e., LLAKEP$^+$ in this case. Following the given model in Figure 6, two worlds are considered. The left world is LLAKEP$^+$ in which the messages are generated using secure cryptographic primitives, i.e., one-way hash function and ECC, and the right world in which a simulator is negotiating with a random oracle (RO) to adapt the response to the LLAKEP$^+$'s structure. Besides that, the adversary can query directly to a one-way hash function or ECC independently in each world. However, in the real world, the responses are computed by the queried cryptographic primitive, while in the random world, RO returns a random value of proper length.

Theorem 1. *Let q_{EST}, q_H, and q_{ECC}, respectively, represent the number of queries to RP/RW of the form* Exe, Send *and* Test, *$H(\cdot)$ and ECC, then considering a polynomial-time adversary's advantage (Adv) to distinguish these worlds is bounded as follows:*

$$Adv^{RoR}_{RW\to RP} \quad \leq \quad Adv_H^{(q_H+6\times q_{EST})} + Adv_{ECC}^{(q_{ECC}+4\times q_{EST})}$$

where Adv^q_H and Adv^q_{ECC}, respectively, denote the adversary's maximum advantage to distinguish the employed one-way hash function from a random oracle and solve ECDLP or EC-CDHP.

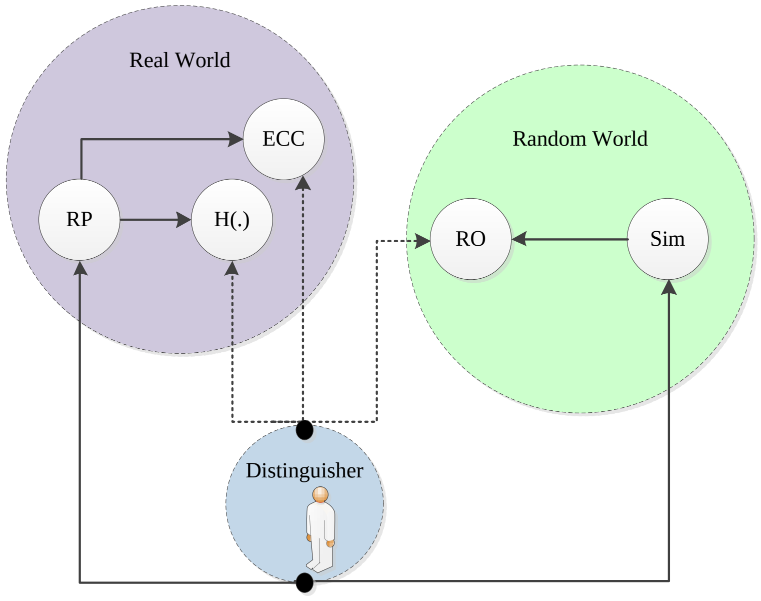


Figure 6. Real or Random (RoR) model; RP: Real Protocol, Sim: Simulator; RO: Random Oracle.

Proof. We consider an EBR that is communicating with BSS to be sharing a session key, sk, and an adversary, $\mathcal{A}$, aims to compromise the semantic security of RP in the real-or-random model. Flowing [25,26], we follow a game-based approach to prove the theorem, and $\mathcal{A}$ wins the game if it can distinguish RW from RP with a non-negligible probability. Through the proof, a series of games are considered, starting with RW and ending with RP. The adversary's advantage while transient from the ith game ($\mathcal{G}_i$) to the $i+1^{th}$ game (G_{i+1}) is computed as the event $Adv^{RoR}_{\mathcal{G}_i \to \mathcal{G}_{i+1}}$ on the term of the number of queries. Since the simulator keeps the structure of the messages identical, it is not possible to trivially distinguish RW from RP, for example, due to timestamps.

Game $\mathcal{G}_0$. It defines the "random world" (RW). Assuming $|H(x)| = n_1$ and $|x \cdot G| = n_2$, in this game, the simulator returns the following values on a run of the protocol: T_{MC} which is selected properly, $R_{EBR} \overset{\$}{\leftarrow} \{0,1\}^{n_2}$, $Auth_{EBR} \overset{\$}{\leftarrow} \{0,1\}^{n_1}$, $R_{BSS} \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $Auth_{BSS} \overset{\$}{\leftarrow} \{0,1\}^{n_1}$ and $V_{EBR} \overset{\$}{\leftarrow} \{0,1\}^{n_1}$. On query x to $H(\cdot)$ the simulator returns $H(x) \overset{\$}{\leftarrow} \{0,1\}^{n_1}$ and on query x to $ECC(\cdot)$ it returns $x \cdot G \overset{\$}{\leftarrow} \{0,1\}^{n_2}$. Then $(R_{EBR}, T_{MC}, Auth_{EBR}, R_{BSS}, Auth_{BSS}, V_{EBR})$ are returned.

Game $\mathcal{G}_1$. In this game, we assume that when querying x to $H(\cdot)$ or $ECC(\cdot)$, the behavior remains identical to G_0. However, on a run of the protocol, the T_{MC} is selected properly, $x_1 \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $R_{EBR} \leftarrow ECC(x_1)$, $y_1 \overset{\$}{\leftarrow} \{0,1\}^*$ and $Auth_{EBR} \leftarrow H(y_1)$, $x_2 \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $R_{BSS} \leftarrow ECC(x_1)$, $y_2 \overset{\$}{\leftarrow} \{0,1\}^*$ and $Auth_{BSS} \leftarrow H(y_2)$ and $y_3 \overset{\$}{\leftarrow} \{0,1\}^*$ and $V_{EBR} \leftarrow H(y_3)$. Then $(R_{EBR}, T_{MC}, Auth_{EBR}, R_{BSS}, Auth_{BSS}, V_{EBR})$ are returned. Given that the inputs of $H(\cdot)$ and $ECC(\cdot)$ are selected randomly and they also return random values on the given input, $Adv^{RoR}_{\mathcal{G}_0 \to \mathcal{G}_1} = 0$.

Game $\mathcal{G}_2$. In this game, $H(\cdot)$ is instantiated by the real one-way hash function. However, the rest of the game remains identical. Therefore, $\mathcal{G}_2$ is identical to $\mathcal{G}_1$ as long as $H(\cdot)$ behaves similarly to a random oracle. However, in reality, most of the hash functions are distinguishable from a random function if there is a collision at their output, for instance. On the other hand, each call to the protocol includes three calls to $H(\cdot)$. If the adversary's advantage to distinguish the instantiated hash function from random oracle after q queries is denoted by Adv^q_H, then

$$Adv^{RoR}_{\mathcal{G}_1 \to \mathcal{G}_2} = Adv^q_H = Adv^{(q_H + 3 \times q_{EST})}_H$$

where q_H and q_{EST}, respectively, denote the total calls to $H(\cdot)$ and a call to the protocol.

Game $\mathcal{G}_3$. In this game, $ECC(\cdot)$ is instantiated by the ECC point multiplication. For example, $x_1 \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $R_{EBR} \leftarrow ECC(x_1)$. However, the rest of the game remains identical. Therefore, $\mathcal{G}_3$ is identical to $\mathcal{G}_2$ as long as ECDLP or EC-CDHP remains unsolved. In addition, each call to the protocol includes two calls to $ECC(\cdot)$. So, if the adversary's advantage is to solve ECDLP or EC-CDHP, which is used to distinguish the instantiated ECC from a random oracle, after q queries, as denoted by Adv^{q}_{ECC}, then

$$Adv^{RoR}_{\mathcal{G}_2 \to \mathcal{G}_3} = Adv^{q}_{ECC} = Adv^{(q_{ECC}+2\times q_{EST})}_{ECC}$$

where q_{ECC} denotes the total calls to $ECC(\cdot)$.

Game $\mathcal{G}_4$. In this game, a constant value is selected as HID. Then, on each query, $y_1 \overset{\$}{\leftarrow} \{0,1\}^*$ and $Auth_{EBR} \overset{\$}{\leftarrow} HID \oplus H(y_1)$. The rest of the game remains identical to $\mathcal{G}_3$. Since the XOR of a constant value to a random value returns a random value, $\mathcal{G}_4$ behaves identically to $\mathcal{G}_3$, assuming the hash function behaves randomly. Therefore

$$Adv^{RoR}_{\mathcal{G}_3 \to \mathcal{G}_4} = Adv^{(q_{EST})}_{H}$$

Game $\mathcal{G}_5$. In this game $K_{MC} \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $R_{EBR} \leftarrow K_{MC} \cdot G$, $y_1 \overset{\$}{\leftarrow} \{0,1\}^*$ and $Auth_{EBR} \leftarrow H(y_1)$. Clearly, $\mathcal{G}_5$ is indistinguishable from $\mathcal{G}_4$ if $k_{MC} \cdot pk_{BSS} \| T_{MC}$ is indistinguishable from $y_1 \overset{\$}{\leftarrow} \{0,1\}^*$, which happens as long as ECC remains unaffected due to the expected random behavior of $k_{MC} \cdot pk_{BSS}$ and $K_{MC} \cdot G$. Hence

$$Adv^{RoR}_{\mathcal{G}_4 \to \mathcal{G}_5} = Adv^{(q_{EST})}_{ECC}$$

Game $\mathcal{G}_6$. In this game, we change the computation on the BSS side compatible with the real world. More precisely, $k_{BSS} \overset{\$}{\leftarrow} \{0,1\}^{n_2}$ and $R_{BSS} = k_{BSS} \cdot G$ and $sk = H(k_{BSS} \cdot R_{EBR} \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$ and $Auth_{BSS} = H(R_{BSS} \| T_{MC} \| sk)$. Following this modification, $\mathcal{G}_6$ is indistinguishable from $\mathcal{G}_5$ if $y_2 \overset{\$}{\leftarrow} \{0,1\}^*$ and $R_{BSS} \| T_{MC} \| H(k_{BSS} \cdot R_{EBR} \| k_{BSS} \cdot pk_{EBR} \| sk_{BSS} \cdot R_{EBR})$ are indistinguishable. Hence

$$Adv^{RoR}_{\mathcal{G}_5 \to \mathcal{G}_6} = Adv^{(q_{EST})_{ECC}+Adv^(q_{EST})}_{H}$$

Game $\mathcal{G}_7$. In this game, we make the final computation on the EBR's side compatible with the real world. More precisely, $sk = H(k_{MC} \cdot R_{BSS} \| sk_{EBR} \cdot R_{EBR} \| k_{MC} \cdot pk_{BSS})$, and $V_{EBR} = H(sk \| T_{MC} \| HID)$. Following this modification, $\mathcal{G}_7$ is indistinguishable from $\mathcal{G}_6$ if $y_3 \overset{\$}{\leftarrow} \{0,1\}^*$ and $sk \| T_{MC} \| HID$ are indistinguishable. Hence

$$Adv^{RoR}_{\mathcal{G}_6 \to \mathcal{G}_7} = Adv^{(q_{EST})}_{H}$$

Game $\mathcal{G}_8$. This game is identical to the real world, which is identical to $\mathcal{G}_7$ actually, and consequently:

$$Adv^{RoR}_{\mathcal{G}_7 \to \mathcal{G}_8} = 0$$

Finally, it is obvious:

$$\begin{aligned} Adv^{RoR}_{RW \to RP} \leq\ & Adv^{RoR}_{\mathcal{G}_0 \to \mathcal{G}_1} + Adv^{RoR}_{\mathcal{G}_1 \to \mathcal{G}_2} + Adv^{RoR}_{\mathcal{G}_2 \to \mathcal{G}_3} + Adv^{RoR}_{\mathcal{G}_3 \to \mathcal{G}_4} + \\ & Adv^{RoR}_{\mathcal{G}_4 \to \mathcal{G}_5} + Adv^{RoR}_{\mathcal{G}_5 \to \mathcal{G}_6} + Adv^{RoR}_{\mathcal{G}_6 \to \mathcal{G}_7} + Adv^{RoR}_{\mathcal{G}_7 \to \mathcal{G}_8} \end{aligned}$$

which gives

$$\begin{aligned} Adv^{RoR}_{RW\to RP} &\leq 0 + Adv_H^{(q_H+3\times q_{EST})} + Adv_{ECC}^{(q_{ECC}+2\times q_{EST})} + Adv_H^{(q_{EST})} + \\ &\quad Adv_{ECC}^{(q_{EST})} + Adv_{ECC}^{(q_{EST})} + Adv_H^{(q_{EST})} + Adv_H^{(q_{EST})} + 0 \\ &\leq Adv_H^{(q_H+6\times q_{EST})} + Adv_{ECC}^{(q_{ECC}+4\times q_{EST})} \end{aligned}$$

which completes the proof. □

For example, the adversary's advantage to find a collision in a hash function after q-queries is bounded by $\frac{q^2}{2^{n_1}}$ and its advantage to ECDLP or EC-CDHP is $\frac{q^2}{2^{n_2}}$. For $n_1 = n_2 = 256$, the adversary's advantage after q_{EST} queries is at most $\frac{36\times q_{EST}^2+16\times q_{EST}^2}{2^{256}} = \frac{52\times q_{EST}^2}{2^{256}}$. Hence, for these parameters, the provable security bound is almost 122 bits.

5.3. Cost Analysis

Through computational comparison, we designate the cost of ECC point multiplication and the one-way hash function by T_{em} and T_H, respectively. Following [27], on a system with 2.20 GHz processor, Pentium dual core E2200, and 2 GB RAM, $T_{em} \cong 2.226$ ms and $T_H \cong 0.0023$ ms.

The computational expenditures of LLAKEP$^+$ and some related protocols, such as LLAKEP, are compared in Table 2 and depicted in Figure 7. The communication overhead (number of transmitted bits) comparison is also offered in Table 2, where the bit lengths of a timestamp, an identifier, a random number, a hash value, and an ECC point are, respectively, 32, 64, 128, 160, and 320 bits. It should be mentioned that we are considering SHA-256 but limiting its output to 160 bits to avoid the current security issues in SHA-1.

Table 2. Cost comparison of LLAKEP$^+$ and other related protocols.

Protocol	Primitives Call	Computations (ms)	Communication (bit)
[28]	$10 \times T_H + 8 \times T_{em}$	17.831	1440 bits
[29]	$5 \times T_H + 9 \times T_{em}$	20.0455	1632 bits
[30]	$11 \times T_H + 9 \times T_{em}$	20.0593	1600 bits
[31]	$8 \times T_H + 9 \times T_{em}$	20.0524	1344 bits
[9]	$12 \times T_H + 6 \times T_{em}$	13.3836	1187 bits
LLAKEP$^+$	$11 \times T_H + 8 \times T_{em}$	17.8333	1152 bits

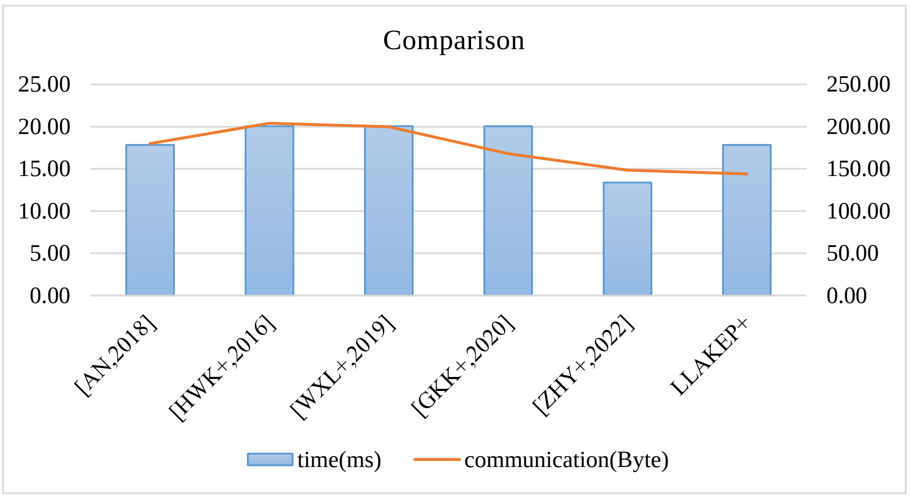


Figure 7. Comparison of LLAKEP$^+$ and related protocols ([AN,2018] [28], [HWK+,2016] [29], [WXL+,2019] [30], [GKK+,2020] [31], and [ZHY+,2022] [9]) in the term of computation and communication cost, the cost of kdf considered equal to a call to hash function.

According to the findings of Table 2 and Figure 7, LLAKEP$^+$ has the lowest communication overhead and is among the fastest in terms of computational cost. Although LLAKEP is 25% faster than LLAKEP$^+$; however, it does not provide perfect security following the provided arguments in Section 3. In addition, LLAKEP$^+$ has the lowest communication cost among those protocols.

6. Conclusions

In this paper, we focused on the security of LLAKEP, an authenticated key agreement protocol for Energy Internet of Things (EIoT) applications. Our security analysis should be viewed as a supplement to the designers' security analysis, with a focus on its provable security. Our findings, however, indicate that this protocol has security weaknesses such as traceability, a dictionary, stolen smart glasses, known session-specific temporary information, key compromise impersonation attacks, and a lack of perfect forward secrecy.

Furthermore, we demonstrated that LLAKEP has some efficiency issues. To overcome the LLAKEP vulnerabilities, we suggested the LLAKEP$^+$ protocol. We employed the same set of cryptographic primitives in the proposed protocol, namely the one-way hash function and ECC point multiplication. Our comprehensive security investigation demonstrates its resistance to different threats such as impersonation, privileged insider assaults, and stolen smart glass attacks. It also provides forward secrecy and user anonymity and is resistant to sophisticated attacks such as KCI and KSTI.

Author Contributions: M.H. and R.A.N.: methodology, designing, validation, supervision, review, and editing; M.S.: conceptualization, methodology, validation, writing—review and editing; L.T. and R.M.M.: conceptualization, validation, writing—review and editing and funding. All authors have read and agreed to the published version of the manuscript.

Funding: The Xiamen University Malaysia Research Fund (XMUMRF) (Grant No.: XMUMRF/2022-C9/IECE/0035).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: For any supplementary material, please contact the corresponding authors.

Acknowledgments: We appreciate the time and effort that were put forth to review the paper by the anonymous reviewers. We sincerely thank you for your insightful comments and recommendations, which allowed us to increase the manuscript's quality.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
EIoT	Energy Internet of Things
IoD	Internet of Drones
IIoT	Industrial Internet of Things
IoV	Internet of Vehicles
MIoT	Medical Internet of Things
NVM	Non Volatile Memory
IT	Information Technology
EBR	Electric Bike Riders
BSS	Battery Swap Station
MC	Microprocessor Chip
ECC	Elliptic Curve Cryptography

KSTI	Known Session-specific Temporary Information
KCI	Key Compromised Impersonation
MitM	Man in the Middle Attack

Appendix A. SPDL Description of the Proposed Protocol

Listing A1: SPDL description of the proposed protocol.

```
hashfunction H;
hashfunction ECC;
const xor : Function;
const G;
secret HID;
usertype Timestamp;
protocol @oracle (X){
        role X {
        var Y:Agent;
        const G;
        recv_!X1(X, X, ECC(X,ECC(Y,G)));
        send_!X2(X, X, ECC(Y,ECC(X,G)));
        }
  }

protocol @oracleE (X){
    role X {
    var Y:Agent;
    const G;
    var KMC;
    fresh KBSS;
    recv_!X1(X, X, ECC(KMC,ECC(KBSS,G)));
    send_!X2(X, X, ECC(KBSS,ECC(KMC,G)));
    }
  }

protocol @oracleZ (X,Y){
    role X {
    const G;
    var KMC;
    fresh KBSS;
    send_1(X,Y,ECC(KBSS,G));
    recv_2(Y,X,ECC(sk(Y),ECC(KBSS,G)));
    send_3(X, Y, ECC(KBSS,pk(Y)));
    }
    role Y{
    fresh KMC;
    const G;
    var KBSS;
    recv_1(X,Y,ECC(KBSS,G));
    send_2(Y,X,ECC(sk(Y),ECC(KBSS,G)));
    recv_3(X, Y, ECC(KBSS,pk(Y)));
    }

  }

protocol @oracleP (X){
    role X {
    var Y:Agent;
    const G;
    var KMC;
    fresh KBSS;
    recv_!X1(X, X, ECC(KMC,pk(X)));
    send_!X2(X, X, ECC(sk(X),ECC(KMC,G)));
    }
  }


protocol @oracleM (X){
    role X {
```

```
    var Y:Agent;
    const G;
    var KBSS;
    macro REBR=ECC(KMC,G);
    macro RBSS=ECC(KBSS,G);
    fresh KMC;
    recv_!X1(X, X, H(ECC(KBSS,REBR),ECC(KBSS,pk(X)),ECC(sk(Y),REBR)));
    send_!X2(X, X, H(ECC(KMC,RBSS),ECC(sk(X),RBSS),ECC(KMC,pk(Y))));
    }
  }

protocol @mad (X){
    role X {
    var Y:Agent;
    const G;
    recv_!X1(X,X,ECC(sk(Y),pk(X)));
    send_!X2(X, X, ECC(sk(X), pk(Y)));
    }
  }

protocol proposed (EBR, BSS){
    role EBR {
    fresh TMC: Timestamp;
    fresh KMC;
    var KBSS;
    var AuthBSS;
    secret HID;
    macro  REBR=ECC(KMC,G);
    macro   AuthEBR=xor(HID,H(ECC(KMC,pk(BSS)),TMC));
    send_1(EBR,BSS,REBR,TMC,AuthEBR);
    recv_2(BSS,EBR,AuthBSS,RBSS);
    macro sk=H(ECC(KMC,RBSS),ECC(sk(EBR),RBSS),ECC(KMC,pk(BSS)));
    match(AuthBSS, H(RBSS,TMC,sk) );
    macro VEBR=H(sk,TMC,HID);
    send_3(EBR, BSS, VEBR);
    claim(EBR, Secret, sk);
    claim(EBR, Secret, HID);
    claim(EBR, Secret, sk(EBR));
    claim(EBR, Secret, KMC);
    claim(EBR, Nisynch );
    claim(EBR, Alive );
    claim(EBR, Weakagree);
    claim(EBR, Niagree);
    };

    role BSS {
    var TMC;
    var KMC;
    fresh KBSS;
    secret HID;
    recv_1(EBR, BSS, REBR,TMC,AuthEBR);
    macro  sk2=H(ECC(KBSS,REBR),ECC(KBSS,pk(EBR)),ECC(sk(BSS),REBR));
    macro tt=xor(AuthEBR,H(ECC(sk(BSS),REBR),TMC));
    match(HID,tt);
    macro  RBSS=ECC(KBSS,G);
    macro   AuthBSS=H(RBSS,TMC,sk2);
    send_2(BSS,EBR,AuthBSS,RBSS);
    recv_3(EBR,BSS,VEBR);
    match(VEBR,H(sk2,TMC,HID));
    claim(BSS, Secret, sk2);
    claim(BSS, Secret, HID);
    claim(BSS, Secret, sk(BSS));
    claim(BSS, Secret, KBSS);
    claim(BSS, Nisynch);
    claim(BSS, Alive);
    claim(BSS, Weakagree);
    claim(BSS, Niagree);
    };
  };
```

References

1. Bolla, R.; Bruschi, R.; Davoli, F.; Cucchietti, F. Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures. *IEEE Commun. Surv. Tutor.* **2011**, *13*, 223–244. [CrossRef]
2. Boccadoro, P.; Striccoli, D.; Grieco, L.A. An extensive survey on the Internet of Drones. *Ad Hoc Netw.* **2021**, *122*, 102600. [CrossRef]
3. Franco, J.; Aris, A.; Canberk, B.; Uluagac, A.S. A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2351–2383. [CrossRef]
4. Ji, B.; Zhang, X.; Mumtaz, S.; Han, C.; Li, C.; Wen, H.; Wang, D. Survey on the Internet of Vehicles: Network Architectures and Applications. *IEEE Commun. Stand. Mag.* **2020**, *4*, 34–41. [CrossRef]
5. Papaioannou, M.; Karageorgou, M.; Mantas, G.; Sucasas, V.; Essop, I.; Rodriguez, J.; Lymberopoulos, D.K. A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT). *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e4049. [CrossRef]
6. Ma, Z.; Ma, J.; Miao, Y.; Liu, X.; Choo, K.R.; Gao, Y.; Deng, R.H. Verifiable Data Mining Against Malicious Adversaries in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2022**, *18*, 953–964. [CrossRef]
7. Gonzalez Granadillo, G.; Zarzosa, S.G.; Diaz, R. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors* **2021**, *21*, 4759. [CrossRef]
8. Zhdanova, M. Security and Trust in Safety Critical Infrastructures. Ph.D. Thesis, Technical University of Darmstadt, Darmstadt, Germany, 2022.
9. Zhang, X.; Huang, X.; Yin, H.; Huang, J.; Chai, S.; Xing, B.; Wu, X.; Zhao, L. LLAKEP: A Low-Latency Authentication and Key Exchange Protocol for Energy Internet of Things in the Metaverse Era. *Mathematics* **2022**, *10*, 2545. [CrossRef]
10. Lansky, J.; Rahmani, A.M.; Ali, S.; Bagheri, N.; Safkhani, M.; Hassan Ahmed, O.; Hosseinzadeh, M. BCmECC: A Lightweight Blockchain-Based Authentication and Key Agreement Protocol for Internet of Things. *Mathematics* **2021**, *9*, 3241. [CrossRef]
11. Rostampour, S.; Safkhani, M.; Bendavid, Y.; Bagheri, N. ECCbAP: A Secure ECC based Authentication Protocol for IoT edge devices. *Pervasive Mob. Comput.* **2020**, *67*, 101194. [CrossRef]
12. Dolev, D.; i-Chih Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–207. [CrossRef]
13. Canetti, R.; Krawczyk, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology—EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, 6–10 May 2001*; Pfitzmann, B., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045, pp. 453–474. [CrossRef]
14. Safkhani, M.; Bagheri, N.; Kumari, S.; Tavakoli, H.; Kumar, S.; Chen, J. RESEAP: An ECC-Based Authentication and Key Agreement Scheme for IoT Applications. *IEEE Access* **2020**, *8*, 200851–200862. [CrossRef]
15. Limbasiya, T.; Das, D. Lightweight Secure Message Broadcasting Protocol for Vehicle-to-Vehicle Communication. *IEEE Syst. J.* **2020**, *14*, 520–529. [CrossRef]
16. Hlauschek, C.; Gruber, M.; Fankhauser, F.; Schanes, C. Prying Open Pandora's Box: KCI Attacks against TLS. In Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT 15), Washington, DC, USA, 10–11 August 2015.
17. Ma, Z.; He, J. Outsider Key Compromise Impersonation Attack on a Multi-factor Authenticated Key Exchange Protocol. In *Applied Cryptography and Network Security Workshops—ACNS 2022 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, SiMLA, Rome, Italy, 20–23 June 2022*; Zhou, J., Adepu, S., Alcaraz, C., Batina, L., Casalicchio, E., Chattopadhyay, S., Jin, C., Lin, J., Losiouk, E., Majumdar, S., et al., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13285, pp. 320–337. [CrossRef]
18. Hosseinzadeh, M.; Ahmed, O.H.; Ahmed, S.H.; Trinh, C.; Bagheri, N.; Kumari, S.; Lansky, J.; Huynh, B. An Enhanced Authentication Protocol for RFID Systems. *IEEE Access* **2020**, *8*, 126977–126987. [CrossRef]
19. RISE GmbH. KCI Attacks against TLS. Available online: https://kcitls.org/ (accessed on 20 December 2022).
20. Johnson, D.; Menezes, A.; Vanstone, S.A. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Sec.* **2001**, *1*, 36–63. [CrossRef]
21. Lowe, G. A hierarchy of authentication specifications. In Proceedings of the 10th Computer Security Foundations Workshop, Rockport, MA, USA, 10–12 June 1997; pp. 31–43.
22. Darbandeh, F.G.; Safkhani, M. SAPWSN: A secure authentication protocol for wireless sensor networks. *Comput. Netw.* **2022**, *220*, 109469. [CrossRef]
23. Cremers, C. CISPA. Available online: https://people.cispa.io/cas.cremers/publications/index.html (accessed on 20 December 2022).
24. Abdalla, M.; Fouque, P.; Pointcheval, D. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *Public Key Cryptography—PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, 23–26 January 2005*; Vaudenay, S., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3386, pp. 65–84.
25. Bagheri, N.; Kumari, S.; Camara, C.; Peris-Lopez, P. Defending Industry 4.0: An Enhanced Authentication Scheme for IoT Devices. *IEEE Syst. J.* **2022**, *16*, 4501–4512. [CrossRef]
26. Rostampour, S.; Bagheri, N.; Bendavid, Y.; Safkhani, M.; Kumari, S.; Rodrigues, J.J.P.C. An Authentication Protocol for Next Generation of Constrained IoT Systems. *IEEE Internet Things J.* **2022**, *9*, 21493–21504. [CrossRef]
27. Khan, A.A.; Kumar, V.; Ahmad, M.; Rana, S.; Mishra, D. PALK: Password-based anonymous lightweight key agreement framework for smart grid. *Int. J. Electr. Power Energy Syst.* **2020**, *121*, 106121. [CrossRef]

28. Abbasinezhad-Mood, D.; Nikooghadam, M. An Anonymous ECC-Based Self-Certified Key Distribution Scheme for the Smart Grid. *IEEE Trans. Ind. Electron.* **2018**, *65*, 7996–8004. [CrossRef]
29. He, D.; Wang, H.; Khan, M.K.; Wang, L. Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography. *IET Commun.* **2016**, *10*, 1795–1802. [CrossRef]
30. Wu, F.; Xu, L.; Li, X.; Kumari, S.; Karuppiah, M.; Obaidat, M.S. A Lightweight and Provably Secure Key Agreement System for a Smart Grid with Elliptic Curve Cryptography. *IEEE Syst. J.* **2019**, *13*, 2830–2838. [CrossRef]
31. Garg, S.; Kaur, K.; Kaddoum, G.; Rodrigues, J.J.P.C.; Guizani, M. Secure and Lightweight Authentication Scheme for Smart Metering Infrastructure in Smart Grid. *IEEE Trans. Ind. Inform.* **2020**, *16*, 3548–3557. [CrossRef]