applied sciences

MDPI

Article

On the Security of a Lightweight and Secure Access Authentication Scheme for Both UE and mMTC Devices in 5G Networks

Kisung Park [1] and Youngho Park [2,*]

[1] Blockchain Research Center, Electronics and Telecommunications Research Institute, Daejeon 34129, Korea; ks.park@etri.re.kr
[2] School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea
* Correspondence: parkyh@knu.ac.kr

Abstract: The Internet of Things (IoT) and 5G networks play important roles in the latest systems for managing and monitoring various types of data. These 5G based IoT environments collect various data in real-time using micro-sensors as IoT things devices and sends the collected data to a server for further processing. In this scenario, a secure authentication and key agreement scheme is needed to ensure privacy when exchanging data between IoT nodes and the server. Recently, Cao et al. in "LSAA: A lightweight and secure access authentication scheme for both UE and mMTC devices in 5G networks" presented a new authentication scheme to protect user privacy. They contend that their scheme not only prevents various protocol attacks, but also achieves mutual authentication, session key security, unlinkability, and perfect forward/backward secrecy. This paper demonstrates critical security weaknesses of their scheme using informal and formal (mathemati) analysis: it does not prevent a single point of failure and impersonation attacks. Further, their proposed scheme does not achieve mutual authentication and correctness of security assumptions, and we perform simulation analysis using a formal verification tool to its security flaws. To ensure attack resilience, we put forward some solutions that can assist constructing more secure and efficient access authentication scheme for 5G networks.

Keywords: 5G; authentication; Internet of Things (IoT); key agreement; security weaknesses

Citation: Park, K.; Park, Y. On the Security of a Lightweight and Secure Access Authentication Scheme for Both UE and mMTC Devices in 5G Networks. *Appl. Sci.* **2022**, *12*, 4265. https://doi.org/10.3390/app12094265

Academic Editor: Carla Raffaelli

Received: 24 March 2022
Accepted: 20 April 2022
Published: 23 April 2022

1. Introduction

The radical development of the Internet of Things (IoT) and 5G networks in the present day has made security a demanding requirement for providing various services such as smart-healthcare, smart-home, smart-industries, etc., securely. Many IoT things devices are deployed in IoT environments to make it easy to manage and process huge real-time data to provide convenient services to the users of the 5G network. It is for this reason that 5G and IoT technology have an important role in the life of human beings because it helps in managing real-time data and to improve the quality of life of people [1]. In this situation, the exchange of data must be secure and reliable, made available only to the legitimate entities while keeping them away from the reach of malicious adversaries. IoT and mobile devices generally store secret parameters during the registration phase and then use it to authenticate among legal entities. If these devices are compromised, it can cause serious security problems because the devices have collected various data related with users such as voice, health, location, finance, etc. [2]. Therefore, research for privacy-preserving scheme is needed to ensure user and data privacy, which consider the possibility that user devices are compromised.

The results of several research works have been proposed for ensuring user privacy in IoT [3–8]. In 2016, to enhance user privacy for IoT, Park et al. proposed three factor based authentication scheme using elliptic curve cryptosystem (ECC) [3]. However, in

2017, Moon et al. [4] and Wang et al. [5] demonstrated that Park et al.'s scheme does not prevent impersonation and offilne dictionary attacks, and then they proposed a enhanced authentication and key agreement scheme to ensure secure communications in IoT environments. We et al. [6] also proposed a provable and secure user authentication scheme to resolve the common challenges and ensure the essential security properties of IoT. In 2018, Wazid et al. [7] proposed a secure user authentication with key agreement scheme for generic IoT networks. In 2019, Adavoudi-Jolfaei et al. [8] presented a lightweight three factor authentication scheme for providing access control between different groups. However, all the above-mentioned research works still have security weaknesses and do not consider the practical IoT environments.

Recently, Cao et al. [9] proposed a lightweight and secure access authentication scheme to guarantee security and privacy in 5G based IoT environments. However, this paper points out that Cao et al.'s scheme is not secure against a single point of failure and impersonation attacks. Since the secret parameters are stored as plaintext in devices, an adversary can, not only obtain public parameters but also easily get secret parameters stored in physical devices in their threat model. To resolve these security flaws, several studies [10–12] indicated that storing the secret parameters as plaintext is a major security weakness and it must be masked using a hash function and XOR operation. Further, we suggest a possible solution to ensure attack resilience.

The remainder of this paper is organized as follows. First, we present a review and cryptanalysis of Cao et al.'s scheme in Sections 2 and 3. Afterward, we present a solution to ensure attack resilience and improved security in Section 4. Finally, we present a conclusion of this paper in Section 5.

1.1. Motivation and Contribution

The main purpose of this paper is to demonstrate the major security weaknesses of the LSAA scheme proposed by Cao in et al. [9]. In their scheme, an adversary can easily impersonate a legitimate user and generate a session key among entities. Therefore, we note that Cao et al.'s scheme is not secure against some attacks using informal and formal (mathematical) security analysis and does not meet essential security requirements in their threat model. We also perform the formal verification analysis using automated validation of internet security protocols and applications (AVISPA) [13] to demonstrate its security flaws, and is unsuitable for deployment in a public network. Further, we propose a solution for resolving these security weaknesses and to improve the overall security level.

1.2. Threat Model

In Cao et al.'s scheme, they adopt the Dolev-Yao (DY) threat model [14] to evaluate the security of the protocols. According to this model, an adversary can intercept, eavesdrop, insert, delete, and modify all messages transmitted between the communicating entities including user equipment (UE), machine-type communication (MTC) devices, and serving network (SN) because they communicate over a public (insecure) channel. The key generation center (KGC) is a fully trusted entity because it generates and manages the secret key for UEs and MTC devices (MDs). However, UEs and MDs are not physically protected and an adversary can obtain the data in the memory of UEs and MDs using power analysis attack [10,15,16].

2. Review of Cao et al.'s Scheme

This section succinctly reviews Cao et al.'s [9] scheme and discusses the threat model that can be used to perform cryptanalysis of their scheme. This scheme consists of four phases: system setup, registration, authentication and key agreement between UE and SN, and group access authentication and key agreement between massive MDs and SN. The notations used in this paper are presented in Table 1.

Table 1. Notation used in this paper.

Notation	Description
$H_{1,2}$	Secure one-way hash function, H_1, H_2:$\{1,0\}^* \rightarrow Z_p^*$
UE	A user equiment
MD	A machine-type communication (MTC) device,
SN	A serving network
KGC	A key generation center
$K_{sn,ue,md}$	Parameters for SNs, UEs and MDs, respectively
ID_I	A I's real idenity
GID	A identity of MTC group
s_j	A SN_j's master key
$\{u_i, K_{ue_i}\}$	UE_i's master key and secret parameters
$\{m_i, K_{md_i}\}$	Each MD_i's master key and secret parameters
$\{m_g, K_{md_g}\}$	MTC group's master key and secret parameters
ENC_x	The encrypted value with K_x
MAC_i	The message authentication code
$\|\|$	A concatenation operation

2.1. System Setup Phase

This phase is performed by KGC to setup the system parameters. The KGC generates a large prime number p and three variables (K_sn, K_ue, and $K_{md} \in_R (-\infty, +\infty)$) for registered MDs, UEs, and SNs, where $a \in_R (-\infty, +\infty)$ indicates that a is uniformly random and selected from the range $(-\infty, +\infty)$. Then, KGC selects one-way hash functions H_1 and H_2 and broadcasts public parameters including p, K_{sn}, K_{ue}, K_{md}, H_1, and H_2.

2.2. Registration Phase

In this phase, SN and UE register themselves with the KGC via a secure channel to access the system. SN and UE share the secret parameters with KGC during this phase. A detailed explanation of the process is presented as follows.

2.3. SN Registration

This process is performed by the KGC through a secure channel.

(1) SN_j securely sends a unique identity ID_{SN_j} to KGC.

(2) After receiving ID_{SN_j}, the KGC generates a master key s_j for SN_j and computes $T_{s_j} = (K_{sn}||ID_{SN_j})$ mod p using Chebyshev polynomials. Then, KGC securely sends s_j to SN_j and broadcasts SN_j's public key $T_{s_j}(K_{sn}||ID_{SN_j})$ mod p and the unique identity ID_{SN_j}.

2.4. Device Registration

2.4.1. UE Registration

(1) The KGC generates a master key u_i for UE_i and a variable $K_{ue_i} \in_R (-\infty, +\infty)$, computes $T_{u_i}(K_{ue_i}||ID_{UE_j})$ mod p, and then securely issues the smartcard (SC) to UE_i including ID_{UE_i}, $T_{u_i}(K_{ue_i}||ID_{UE_j})$ mod p, and u_i. These values are secretly shared between UE_i and the KGC.

(2) The KGC computes $H_1(T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod $p||ID_{UE_i})$ and sends it to SN_j for UE_i.

(3) Finally, SN_j stores $H_1(T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod $p||ID_{UE_i})$ into a database for all registered UEs.

2.4.2. MD Registration

(1) The KGC chooses MTC group leader MD_n, a master key m_g, a variable $K_{md_g} \in_R (-\infty, +\infty)$, and then generates a master key m_i, a variable $K_{md_i} \in_R (-\infty, +\infty)$ for MTC group member MD_i.

(2) The KGC computes $T_{m_g} = (K_{md_g}||GID)$ mod p and $T_{m_i} = (K_{md_i}||GID||ID_{MD_i})$ mod p using Chebyshev polynomials. The KGC securely issues SC to MD_i including the unique MD_i's identity ID_{MD_i}, the group identity GID, the shared secret $T_{m_g} = (K_{md_g}||GID)$ mod p and $T_{m_i} = (K_{md_i}||GID||ID_{MD_i})$ mod p between MD_i and KGC.

(3) Finally, the KGC computes $H_1(T_{m_g}(K_{md_g}||GID)$ mod $p||GID)$ and $H_1(T_{m_i}(K_{md_i}||GID||ID_{MD_i})$ mod $p||GID||ID_{MD_i})$, and send it to SN_j. Then SN_j stores it into a database for MTC groups.

2.5. Authentication and Key Agreement Phase between UE and SN

This phase is mutual authentication and key agreement process between UE and SN, which is performed through a public channel. A detailed description of the process is presented as follows.

(1) UE_i pre-computes $T_{U_i}(K_{sn}||ID_{SN_j})$ mod p and $T_{u_i}(K_{ue}||ID_{UE_i})$ mod p. Then, UE_i generates x_i and computes $T_{x_i}(K_{sn}||ID_{SN_j})$ mod p, $K_1 = T_{x_i}(T_{S_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $K_2 = T_{u_i}(T_{S_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $MAC_1 = H_1(K_2$,ID_{UE_i}, ID_{SN_j}, $T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod $p||T_{u_i}(K_{ue}||ID_{UE_i})$ mod $p||T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$. UE_i encrypts $E_1 = ENK_{K_1}(ID_{UE_i}||T_{U_i}(K_{sn}||ID_{SN_j})$ mod $p||T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod $p||T_{u_i}(K_{ue}||ID_{UE_i})$ mod $p)$ by the secret parameter K_1 and sends the access request including $\{ID_{SN_j}, T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p, E_1, MAC_1\}$ to SN_j.

(2) After receiving the access request, SN_j computes $K_1' = T_{s_j}(T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p and decrypt $E_1 = ID_{UE_i}, T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod p, $T_{u_i}(K_{ue}||ID_{UE_i})$ mod p. SN_j checks whether ID_{UE_i} exist in a database, If it exist, SN_j verifies that $H_1(T_{u_i}(K_{ue_i}||ID_{UE_i})$ mod $p||ID_{UE_i})$ is correct.

(3) SN_j computes $K_2' = T_{s_j}(T_{u_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p and verifies that MAC_1 is correct. If MAC_1 is correct, SN_j generates y_i and computes $T_{y_i}(K_{sn}||ID_{SN_j})$ mod p, $T_{s_j}(K_{ue}||ID_{UE_i})$ mod p, $K_3 = T_{s_j}(T_{u_i}(K_{ue}||ID_{UE_i})$ mod $p)$ mod $p)$, $MAC_2 = H_1(K_3, ID_{UE_i}, ID_{SN_j}, T_{y_i}(K_{sn}||ID_{SN_j})$ mod $p, T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$.

(4) SN_j computes the session key $SK_{ij} = H_2(T_{y_i}(T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod $p||K_1'||K_2'||K_3||ID_{UE_i}||ID_{SN_j})$ and sends authentication request encrypted with K_1' including $\{ENC_{K_1'}(T_{y_i}(K_{sn}||ID_{SN_j})$ mod $p, T_{s_j}(K_{ue}||ID_{UE_i})$ mod $p, MAC_2)\}$.

(5) On receiving the authentication request, UE_i decrypt $ENC_{K_1'}$ and get the $\{(T_{y_i}(K_{sn}||ID_{SN_j})$ mod $p, T_{s_j}(K_{ue}||ID_{UE_i})$ mod $p, MAC_2)\}$. Then, UE_i computes $K_2' = T_{u_i}(T_{s_j}(K_{ue}||ID_{UE_i})$ mod $p)$ mod p and verify that MAC_2 is correct. If it is correct, UE_i computes the session key $SK_{ij}' = H_2(T_{y_i}(T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod $p||K_1||K_2||K_3'||ID_{UE_i}||ID_{SN_j})$, $MAC_3 = H_1(SK_{ij}'||ID_{UE_i}||ID_{SN_j}||T_{x_i}(K_{sn}||ID_{SN_j})$ mod p, $T_{y_i}(K_{sn}||ID_{SN_j})$ mod $p)$ and sends MAC_3 to SN_j

(6) Finally, SN_j verifies that MAC_3 is correct. If it is correct, UE_i and SN_j authenticate and correctly establish the session key each other.

2.6. Group Access Authentication and Key Agreement Phase between Massive MDs and SN

This phase refers to the group access authentication and key agreement process between MDs and SN, which is performed through a public channel. The MTC group leader MD_n aggregates the group member MD_i's data and sends it to SN to authenticate between group members and SN. A detailed description of the process is presented as follows.

(1) The MTC device MD_i precompute $T_{m_i}(K_{sn}||ID_{SN_j})$ mod p, $T_{m_i}(K_{md}||GID)$ mod p, $K_{M_{2i}} = (T_{m_i}(T_{s_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $K_{G1} = T_{m_g}(T_{s_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p.

(2) MD_i selects x_i, z_i and computes $K_{M_{1i}} = T_{x_i}(T_{s_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $MAC_1 = H_1(K_{M_{1i}}||K_{M_{2i}}||K_{G_1}||GID||ID_{MD_i}||ID_{SN_j}||T_{m_i}(K_{md_i}||GID||ID_{MD_i})$ mod p, $T_{m_g}(K_{md_g}||GID)$ mod $p||T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p||z_i)$. Then, MD_i encrypts $E_{1i} = ENC_{K_{M_{1i}}}(ID_{MD_i}$

$||T_{m_i}(K_{md_i}||GID||ID_{MD_i})$ mod $p||T_{m_i}(K_{md}||GID)$ mod $p||T_{m_i}(K_{sn}||ID_{SN_j}$ mod $p||z_i)$ by the secret parameter $K_{M_{1i}}$ and sends the access request $\{ID_{SN_j}, T_{x_i}(K_{sn}||ID_{SN_j})$ mod p, $E_{1i}, MAC_{1i}\}$ to MD_n.

(3) After receiving the access request from MD_i, MD_n computes $MAC_1 = \oplus_{i=1}^{n} MAC_{1i}$ and $\oplus_{i=1}^{n} E_{1n} = ENC_{K_{M_{1n}}}(GID||ID_{MD_n}||T_{m_n}(K_{md_n}||GID||ID_{MD_n}$ mod $p||T_{m_g}(K_{mg}||GID)$ mod $p||T_{m_n}(K_{md}||GID)$ mod $p||T_{m_n}(K_{sn}||ID_{SN_j}$ mod $p||T_{m_g}(T_{m_g}(K_{sn}||ID_{SN_j})$ mod $p||T_{m_g}(K_{md}||GID)$ mod $p||z_n)$, where $\oplus_{i=1}^{n}$ is function of the aggregating access request for group members, and sends the aggregation request $\{ID_{SN_j}, \oplus_{i=1}^{n} T_{x_i}(K_{sn}||ID_{SN_j})$ mod p, $\oplus_{i=1}^{n} E_{1i}$, MAC_1 to SN_j.

(4) On receiving the aggregation request from MD_n, SN_j computes $K'_{M_{1i}} = T_{s_j}(T_{x_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, decrypts E_{1i} and obtains $ID_{MD_i}, GID, T_{m_i}(K_{md_i}||GID||ID_{MD_i})$ mod p and z_i. Then, SN_j checks whether ID_{MD_i} and GID are exist in a database, If they exist, SN_j verifies that $H_1(T_{m_g}(K_{md_g}||GID)$ mod $p||ID_{MD_i})$ and $H_1(T_{m_i}(K_{md_i}||GID||ID_{MD_i})$ mod $p||GID||ID_{MD_i})$ are correct.

(5) SN_j computes $K'_{2i} = T_{s_j}(T_{m_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $K'_{G1} = T_{s_j}(T_{m_g}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p and verifies that MAC_1 is correct. If MAC_1 is correct, SN_j generates y_j and computes $T_{y_j}(K_{sn}||ID_{SN_j})$ mod p, $T_{s_j}(K_{md}||GID)$ mod p, $K_{G2} = T_{s_j}(T_{m_g}(K_{md}||GID)$ mod $p)$ mod $p)$, $K_{M_{3i}} = T_{s_j}(T_{m_i}(K_{md}||GID)$ mod $p)$ mod $p)$.

(6) SN_j computes $Z = \prod_{i=1}^{n} z_i$, $Z_i = Z/z_i$, $y_i = Z_i^{-1}$ using Chinese remainder theorem (CRT). Then, SN_j get $S = (\sum_{i=1}^{n} H_2(K'_{M_{1i}}, K_{M_{3i}}, K_{G2}, ID_{MD_i}, ID_{SN_j}), y_i, Z_i)$ mod Z. Then, SN_j computes the session key $SK_{ij} = H_2(T_{y_j}(T_{X_i}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $K'_{M_{1i}}$, $K'_{M_{2i}}$, $K_{M_{3i}}$, K'_{G1}, K_{G2}, ID_{MD_i}, GID, $ID_{SN_j})$ and sends the group authentication request $\{ENC_{K'_{G1}}(T_{y_j}(K_{sn}||ID_{SN_j})$ mod $p, T_{s_j}(K_{md}||GID)$ mod $p, S\}$ to MD_i.

(7) MD_i decrypts $ENC_{K'_{G1}}(T_{y_j}(K_{sn}||ID_{SN_j})$ mod p, computes $K'_{G2} = T_{m_g}(T_{s_j}(K_{kmd}||GID)$ mod $p)$ mod p and $K'_{M_{3i}} = T_{m_i}(T_{s_j}(K_{kmd}||GID)$ mod $p)$ mod p, and verifies $H_2(K_{M_{1i}}$, $K'_{M_{3i}}, K'_{G2}, ID_{MD_i}, ID_{SN_j}) \stackrel{?}{=} S$ mod z_i.

(8) If it is correct, MD_i computes $SK'_{ij} = H_2(T_{x_i}(T_{y_j}(K_{sn}||ID_{SN_j})$ mod $p)$ mod p, $K_{M_{1i}}$, $K_{M_{2i}}$, $K'_{M_{3i}}$, K_{G1}, K'_{G2}, $ID_{MD_i}, GID, ID_{SN_j})$, $MAC_{3i} = H_1(SK'_{ij}, ID_{MD_i}, GID, ID_{SN_j}$, $T_{x_i}(K_{sn}||ID_{SN_j})$ mod p, $T_{y_j}(K_{sn}||ID_{SN_j})$ mod p and sends MAC_{3i} to MD_n.

(9) On receiving the MAC_{3i} from the group members, MD_n computes $MAC_3 = \prod_{i=1}^{n} MAC_{3i}$ and sends it to SN_j.

(10) Finally, SN_j checks correctness of MAC_3 and authenticates with MD_i.

3. Security Weaknesses of Cao et al.'s Scheme

In this section, we demonstrate that Cao et al.'s scheme is vulnerable to MD and UE impersonation attacks as well as a single point of failure. Further, we also show that Cao et al.'s scheme does not achieve secure mutual authentication and session key security, which is a necessary security requirement for authentication and key agreement scheme.

3.1. Formal Security Analysis

We prove that Cao et al.'s scheme does not achieve the session key security using Real-or-Random (ROR) model [17] which is broadly accepted formal proof [18–20]. We first present the basic concept of ROR model, and then perform the formal security analysis through this proof.

- Participants We denote $\Pi_{UE}^{inst_1}$ and $\Pi_{SN}^{inst_2}$ as the instance $inst_1$ and $inst_2$ of UE and SN, respectively.
- Accepted state After exchanging the last messages, the oracle $\Pi^{i}nst$ moves to an accepted state. When all the messages are concatenated in order, a current session identifier *csid* of Π^{inst} is defined.

- Partnering When $\Pi_{UE}^{inst_1}$ and $\Pi_{SN}^{inst_2}$ are in the shared same *sid* and the accepted state, and then complete mutual authentication and key agreement, $\Pi_{UED}^{inst_1}$ and $\Pi_{SN}^{inst_2}$ are defined as partners.
- Freshness To perform the ROR proof, the instances ($\Pi_{UE}^{inst_1}$, $\Pi_{SN}^{inst_2}$) are considered fresh if the session key between UE and SN is not compromised to attacker A at present.
- Attacker Under the threat model of Cao et al. [9], an A has a complete control over the communication network. A also access to the queries presented in Table 2 to break the security of Cao et al.'s scheme.
- Semantic Security Under the this model, A attempt to find an instance's correct session key from a random nonce. A has to utilize the ROR queries, and then guesses a bit c. When A correctly find a bit c, A win the game ans destroy the semantic security of protocol. We define that Win is event of winning the game by A and $Adv_P = |2Pr[Win] - 1|$ is advantage in breaking the session key of Cao et al.'s scheme P.
- Random Oracle In Cao et al.'s scheme, all participants can utilize a random oracle which is a one-way hash function H.

Table 2. Queries and descriptions.

Queries	Descriptions
$Execute(\Pi_{UE}^{inst_1}, \Pi_{SN}^{inst_2})$	This query is an eavesdropping attack that A can control the exchanged messages over the public network.
$CorruptUE(\Pi_{UE}^{inst_1})$	This query is device stolen attacks that A can retrieve the data stored in device UE_i using this query.
$Send(\Pi^{inst}, Msg)$	This query is an attack that A can send a message and obtain a response from the oracle P^{inst}.
$Test(\Pi^{inst})$	This query is an attack to guess the probabilistic result for an unbiased coin c. When the P^{inst} and A establish the session key SK which is fresh, A sends this query. If its result is $c = 0$ or $c = 1$, A get a random number or the SK, respectively. Otherwise, *Tset* query returns the NULL ($\perp$).

Here, we prove that Cao et al.'s scheme does not achieve the session key security by the following Definition 1 and Theorem 1.

Definition 1. *Chaotic Map-based Discrete Logarithm Problem (CMDLP): Given x and y, it is computationally hard to find integer i such that $T_i(x) = y(mod\ p)$.*

Theorem 1. *Suppose that A is an adversary running in a polynomial time t against LASS and Adv_P^A is the advantage of A in breaking the session key security of Cao et al.'s scheme. Then,*

$$Adv_{\mathcal{P}}^{A} \geq \frac{q_h^2}{|Hash|} + 2Adv^{CMDLP}(t), \tag{1}$$

where q_h, $Hash$, and $Adv^{CMDLP}(t)$ denote the number of $Hash$ queries, $Hash$ is a one-way hash function H, and $Adv^{CMDLP}(t)$ is the breaking advantage of CMDLP by A, respectively.

We define the following games $G_i(i = 0, 1, 2)$ with the event $Succ_i$ in which A wins the game G_i. The formal proofs using ROR model are below:

- Game G_0: This game is a direct attack by A against the protocol. The c is first randomly selected at the beginning of this game and its winning advantage is:

$$Adv_{\mathcal{P}}^{A} = |2.Pr[Succ_0] - 1| \tag{2}$$

- Game G_1: This game is an eavesdropping attack by A which A can control the all the exchanged messages using $Execute(\Pi_{UE}^{inst_1}, \Pi_{SN}^{inst_2})$ query. After that, A executes the $Test(\Pi^t)$ query to find whether its output is a correct SK or a random value. In Cao et al.'s scheme, UE and SN exchange the session key SK which is computed by $SK_{ij} = H_2(T_{y_i}(T_{x_i}(K_{sn}||ID_{SN_j}) \bmod p) \bmod p||K_1'||K_2'||K_3||ID_{UE_i}||ID_{SN_j})$ and $SK_{ij}' = H_2(T_{y_i}(T_{x_i}(K_{sn}||ID_{SN_j}) \bmod p) \bmod p||K_1||K_2||K_3'||ID_{UE_i}||ID_{SN_j})$. If A want to correctly guess it, A must break the difficulty of solving CMDLP. However, A should get the temporary private key of UE and SN from the SK. It is computationally hard to find the temporary private key because the SK's security is based on the difficulty of solving CMDLP. Thus, G_0 and G_1 are indistinguishable. Then,

$$Pr[Succ_1] = Pr[Succ_0] \tag{3}$$

- Game G_2: Finally, A performs the final attack and tries to impersonate the legal UE and SN using $Send(\Pi^{inst}, Msg)$, $CorruptUE(\Pi_{UE}^{inst_1})$ and some $Hash$ queries. A execute the $CorruptUE(\Pi_{UE}^{inst_1})$ query, and then extract the values u_i and s_j stored in the memory of UE and SN. A successfully break the session key security using the obtained private key because A can properly proceed the authentication and key agreement phase without solving the CMDLP. Therefore, G_1 and G_2 are distinguishable. Then,

$$|Pr[Succ_1] - Pr[Succ_2| \geq \frac{q_h^2}{2|Hash|} \tag{4}$$

After finishing all the games (G_0, G_1, G_2), A tries to correctly find the c using $Test$ query. Therefore,

$$Adv_{\mathcal{P},G_2}^{A} = \frac{1}{2} \tag{5}$$

We can obtain the following result using the Equations (2), (3) and (5).

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{P}}^{A} &= |Pr[Succ_0] - \frac{1}{2}| \\ &= |Pr[Succ_1] - \frac{1}{2}| \\ &= |Pr[Succ_1] - Pr[Succ_2]| \end{aligned} \tag{6}$$

After that, we can obtain the following result with (4), (5) and (6):

$$|Pr[Succ_1] - Pr[Succ_2]| \geq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{P}}^{CMDLP}(t) \tag{7}$$

Then, we obtain the final result by multiplying 2 both sides of (7):

$$Adv_{\mathcal{P}}^{A} \geq \frac{q_h^2}{|Hash|} + 2Adv^{CMDLP}(t) \tag{8}$$

Finally, we can remove the probability $2Adv^{CMDLP}(t)$ in Equation (8) because we break the session key security without solving the CMDLP. Therefore, we prove that Cao et al.'s scheme does not achieves the session key security using this formal proof.

3.2. Informal Security Analysis

We demonstrate that Cao et al.'s scheme does not resist impersonation attacks and single point of failure, and also does not ensure secure mutual authentication using informal analysis.

3.2.1. UE Impersonation Attack

During UE registration phase, the UE receives SC= $\{ID_{UE_i}, T_{u_i}(K_{ue_i}||ID_{UE_j}) \bmod p, u_i\}$ from KGC. According to Section 1.2, Cao et al. present the threat model and analyze security of the proposed scheme using their threat model. However, if a malicious attacker A compromise the UE and extracts the data stored in the UE's memory, A can successfully generate the access request $\{ID_{SN_j}, T_{x_i}(K_{sn}||ID_{SN_j}) \bmod p, E_1, MAC_1\}$ and the response message $\{MAC_3\}$ because the secret data of the UE's memory is directly stored without employing any cryptographic method. Further, A can also generate the session key SK_{ij}. Therefore, their scheme is vulnerable to UE impersonation attacks and a detailed description of the processed involved in this attack is shown in Figure 1.

Adversary (A/UE_i)		**Serving Network** (SN_j)
Precompute $T_{U_i}(K_{sn}\|\|ID_{SN_j}) \bmod p$, $T_{u_i}(K_{ue}\|\|ID_{UE_i}) \bmod p$, Generates x_i, Computes $T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p$, $K_1 = T_{x_i}(T_{S_j}(K_{sn}\|\|ID_{SN_j}) \bmod p) \bmod p$, $K_2 = T_{u_i}(T_{S_j}(K_{sn}\|\|ID_{SN_j}) \bmod p) \bmod p$, $MAC_1 = H_1(K_2, ID_{UE_i}, ID_{SN_j}, T_{u_i}(K_{ue_i}\|\|ID_{UE_i}) \bmod p$ $\|\|T_{u_i}(K_{ue}\|\|ID_{UE_i}) \bmod p \|\|T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p)$, Encrypts $E_1 = ENK_{K_1}(ID_{UE_i}\|\|T_{U_i}(K_{sn}\|\|ID_{SN_j}) \bmod p$ $\|\|T_{u_i}(K_{ue_i}\|\|ID_{UE_i}) \bmod p \|\|T_{u_i}(K_{ue}\|\|ID_{UE_i}) \bmod p)$,		
	$\{ID_{SN_j}, T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p, E_1, MAC_1\}$ $\dashrightarrow$	
		Computes $K_1' = T_{s_j}(T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p) \bmod p$, Decrypts $E_1 = ID_{UE_i}, T_{u_i}(K_{ue_i}\|\|ID_{UE_i}) \bmod p$, $T_{u_i}(K_{ue}\|\|ID_{UE_i}) \bmod p$, Finds ID_{UE_i} in a database Verfies $H_1(T_{u_i}(K_{ue_i}\|\|ID_{UE_i}) \bmod p\|\|ID_{UE_i})$ Computes $K_2' = T_{s_j}(T_{u_i}(K_{sn}\|\|ID_{SN_j}) \bmod p) \bmod p$ Verifies MAC_1, Generates y_i, Computes $T_{y_i}(K_{sn}\|\|ID_{SN_j}) \bmod p$ $T_{s_j}(K_{ue}\|\|ID_{UE_i}) \bmod p$, $K_3 = T_{s_j}(T_{u_i}(K_{ue}\|\|ID_{UE_i})$ $\bmod p) \bmod p)$, $MAC_2 = H_1(K_3, ID_{UE_i}, ID_{SN_j}, T_{y_i}(K_{sn}$ $\|\|ID_{SN_j}) \bmod p, T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p)$, The session key $SK_{ij} = H_2(T_{y_i}(T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p) \bmod p$ $\|\|K_1'\|\|K_2'\|\|K_3\|\|ID_{UE_i}\|\|ID_{SN_j})$
	$\{ENC_{K_1'}(T_{y_i}(K_{sn}\|\|ID_{SN_j}) \bmod p$ $T_{s_j}(K_{ue}\|\|ID_{UE_i}) \bmod p, MAC_2)\}$ $\dashleftarrow$	
Decrypts $ENC_{K_1'}$ Computes $K_2' = T_{u_i}(T_{s_j}(K_{ue}\|\|ID_{UE_i}) \bmod p) \bmod p$, Verify$MAC_2$, Computes the session key $SK_{ij}' = H_2(T_{y_i}(T_{x_i}(K_{sn}\|\|ID_{SN_j}) \bmod p)$ $\bmod p\|\|K_1\|\|K_2\|\|K_3'\|\|ID_{UE_i}\|\|ID_{SN_j})$, $MAC_3 = H_1(SK_{ij}'\|\|ID_{UE_i}\|\|ID_{SN_j}\|\|T_{x_i}(K_{sn}\|\|ID_{SN_j})$ $\bmod p, T_{y_i}(K_{sn}\|\|ID_{SN_j}) \bmod p)$		
	$\{MAC_3\}$ $\dashrightarrow$	

Figure 1. UE impersonation attack in Cao et al.'s scheme.

3.2.2. MD Impersonation Attack

In the MTC device (MD) registration phase, the MD received SC=$\{ID_{MD_i}, GID, T_{m_g} = (K_{md_g}||GID) \bmod p, T_{m_i} = (K_{md_i}||GID||ID_{MD_i}) \bmod p\}$ from KGC, where ID_{MD_i}, GID,

$T_{m_g} = (K_{md_g}||GID) \bmod p$, $T_{m_i} = (K_{md_i}||GID||ID_{MD_i}) \bmod p$ are the unique MD_i's identity, the group identity GID, the shared secret between MD_i and KGC, respectively. When an adversary A obtains the MD_i and extracts these secret parameters, A can not only access the serving network but also generate the session key between MD_i and SN_j. Hence, Cao et al.'s scheme does not prevent MD impersonation attack and for a detailed description of the processes involved in this phase, please refer to [9].

3.2.3. Secure Mutual Authentication

According to Sections 3.2.1 and 3.2.2, an adversary A can easily access the system proposed by Cao et al.'s scheme and authenticate among entities. Additionally, A can generate the session key between UE/SN and MTD devices/SN. Thus, their scheme does not achieve secure mutual authentication.

3.2.4. Single Point of Failure

In Cao et al.'s scheme, the MTC group leader MD_n collects the access request of the group member MD_i and aggregates it. Afterward, MD_n sends the aggregation messages of an access request to serving network SN_j. However, if MD_n node is compromised, off-line or break down, the access request of massive MTC nodes cannot be delivered to SN_j. It limits the security and the performance of the proposed system. Therefore, Cao et al.'s scheme does not offer resistance against a single point of failure attack because the massive MTC nodes cannot be able to access the service when MD_n does not work.

3.2.5. Correctness of Security Assumption

Cao et al. presented a threat model to analyze the security of the scheme, and then claimed that their scheme is secure against various attacks on the presented threat model. However, we demonstrate that Cao et al.'s scheme is vulnerable to the above-mentioned attacks using their threat model and that they did not consider all potential attacks. Thus, we suggest a solution to alleviate the said security flaws in Section 4.

3.3. Simulation Analysis Using AVISPA Tool

This section perform the formal simulation analyis uisng AVISPA tool which is a widely-accepted validation tool for proving security of cryptographic protocols [13,21]. AVISPA verifies that cryptograhpic protocols is secure against replay and man-in-the-middle attacks. It uses a high-level protocols specification language (HLPSL) [22] to construct the security features of the protocols. There are four back-ends models [23]: "constraint logic-based attack searcher (CL-AtSE)", "on the fly model checker (OFMC)", "SAT-based model checker (SATMC)", and "tree automata based on protocol analyzer (TA4SP)". The constructed HLPSL code is converted to a intermediate format (IF) using a translator "HLPSL2IF", and then it is utilized for four back-ends to prove security. Finally, the output presents results of security analysis. This process is presented in Figure 2 and the detailed description of HLPLS can be found in [21,22].

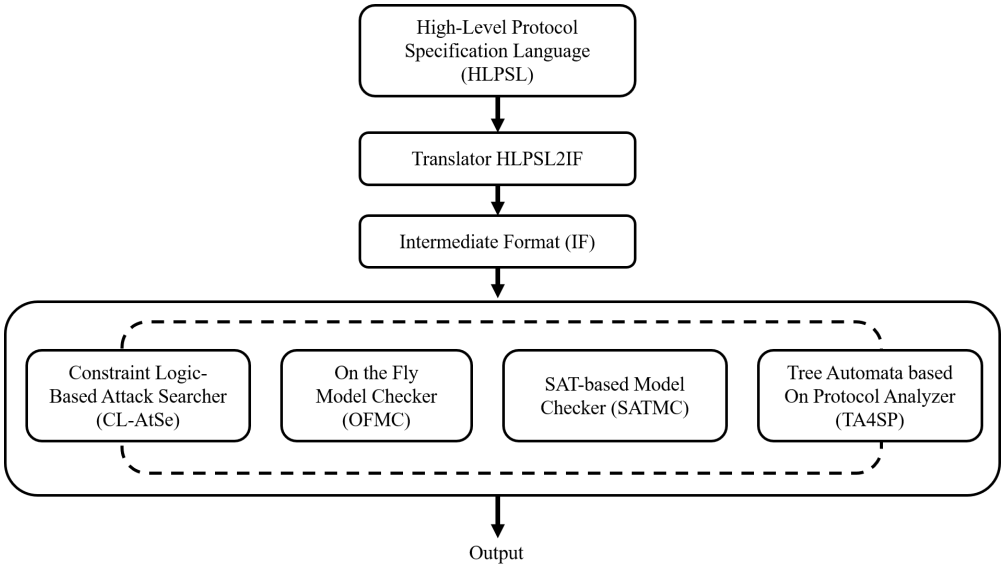

Figure 2. The Process of AVISPA Simulation.

3.3.1. HLPLS Specifications

Before the beginning of simulation proof, all the phases of Cao et al.'s scheme are defined through the HLPLS. We then have tested it under two scenarios (UE-SN and MD-SN), considering UE-SN and MD-SN authentication phases.

Scenario 1. UE-SN Authentication: In scenario 1, there are three basic roles (SN, UE, KGC) and the HLPSL descriptions of each role are shown in Figures 3–5, respectively. The session and environment are defind in Figure 6.

Scenario 2. MD-SN Authentication: In scenario 2, there are four basic roles (SN, MD_i, MD_n, KGC) and the HLPSL descriptions of each role are shown in Figures 7–10, respectively. The session and environment are defined in Figure 11.

```
%%%%%%% Role SN %%%%%%
role servingnetwork(SN,UE,KGC: agent, SKsnc: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by SN
def=
local State: nat,
                Ksn,Kue,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,MAC2j,SKij:text,
                IDuei,Uii,Kuei,Xii,CP5i,CP6i,CP7i,CP8i,CP9i,MAC1i,MAC3i : text,
                CP1k,CP2k,HCP2k,CP3k,CP4k,HCP3k,HCP4k:text

const sp1,sp2,sp3,sp4,ue_sn_xii,sn_ue_yjj,ue_sn_skij: protocol_id
init State:=0
transition

%%%%%% Registration phase %%%%%%%%
1. State=0 /\RCV(start)=|>
State':=1 /\SND({IDsnj}_SKsnc)
%%%%%% SN receives msg from KGC %%%%%%
2. State=1 /\RCV({exp(H(Ksn.IDsnj),Sjj').Sjj'.Ksn.Kue}_SKsnc)=|>
State':=2 /\SND(IDsnj.exp(H(Ksn.IDsnj),Sjj').Ksn.Kue)%send public parameter to ue
3. State=2 /\RCV({H(exp(H(Kuei'.IDuei'),Uii').IDuei)}_SKsnc)=|>
State':=3

%%%%%% Authentication phase ue-sn%%%%%%%----------------------------Here!
%-----2.user authentication request
4. State=3
/\RCV({IDuei.exp(H(Ksn.IDsnj),Uii').exp(H(Kuei'.IDuei'),Uii').exp(H(Kue.IDuei),Uii')}_exp(exp(H(Ksn.IDsnj),Sjj),Xii').ID
snj.exp(H(Ksn.IDsnj),Xii').MAC1i')=|>
State':=4 /\Yjj':=new()                                % receive : %Enc_k1(m){iduei, cp5i, cp2k,cp6i}_k1(cp8i), idsnj,
cp7i,mac1i
                /\CP8i':=exp(exp(H(Ksn.IDsnj),Xii'),Sjj)%K1'
                /\CP9i':=exp(exp(H(Ksn.IDsnj),Uii'),Sjj)%k2
                /\CP10j':=exp(H(Ksn.IDsnj),Yjj')%T_yj(ksn||idsnj)
                /\CP11j':=exp(H(Kue.IDuei),Sjj)%T_sj(kue||iduei)
                /\CP12j':=exp(exp(H(Kue.IDuei),Uii'),Sjj)%k3 , T_sj(T_yj(kue||iduei))
                /\MAC2j':=H(CP12j'.IDuei.IDsnj.CP10j'.exp(H(Ksn.IDsnj),Xii'))
                /\SKij':=H(exp(exp(H(Ksn.IDsnj),Xii'),Yjj').CP8i'.CP9i'.CP12j'.IDuei.IDsnj)
                /\SND({CP10j'.CP11j'.MAC2j'}_CP8i')
                /\request(SN,UE,ue_sn_xii,Xii')
                /\witness(SN,UE,sn_ue_yjj,MAC2j')
5. State=4 /\RCV(H(SKij.IDuei.IDsnj.exp(H(Ksn.IDsnj),Xii').exp(H(Ksn.IDsnj),Yjj)))=|>
State':=5 /\MAC3i':=H(SKij.IDuei.IDsnj.exp(H(Ksn.IDsnj),Xii').CP10j)
                /\request(SN,UE,ue_sn_skij,MAC3i')

end role
```

Figure 3. Scenario 1: HLPSL description of SN role.

```
%%%%%%% Role UE %%%%%
role userequipment(SN,UE,KGC: agent, SKunc,SKuni: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by UE
def=
local State: nat,
                Ksn,Kue,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,MAC2j,SKij:text,
                IDuei,Uii,Kuei,Xii,CP5i,CP6i,CP7i,CP8i,CP9i,MAC1i,MAC3i : text,
                CP1k,CP2k,HCP2k,CP3k,CP4k,HCP3k,HCP4k:text

const sp1,sp2,sp3,sp4 : protocol_id,
ue_sn_xii,sn_ue_yjj,ue_sn_skij: protocol_id
init State:=1
transition

%%%%% Registration phase %%%%%%%%
2. State=1 /\RCV(IDsnj.exp(H(Ksn.IDsnj),Sjj').Ksn.Kue)=|>
State':=2 /\IDuei':=new()
                /\SND({IDuei'}_SKunc)
                /\secret(IDuei',sp4,{UE,KGC})
%%%%% receive smartcard %%%%%%%%
3. State=2 /\RCV({exp(H(Kuei'.IDuei'),Uii').Uii'}_SKunc) =|>
State':=3 /\Xii':=new()
                /\SND({Xii'}_SKuni)
%%%%% Authentication phase ue-sn%%%%%%%%
%-----1.access request
                /\CP5i':=exp(H(Ksn.IDsnj),Uii')%T_ui(ksn||idsnj)
                /\CP6i':=exp(H(Kue.IDuei),Uii')%T_ui(kue||iduei)
                /\CP7i':=exp(H(Ksn.IDsnj),Xii')%T_xi(ksn||idsnj)
                /\CP8i':=exp(exp(H(Ksn.IDsnj),Sjj),Xii')%k1
                /\CP9i':=exp(exp(H(Ksn.IDsnj),Sjj),Uii')%k2
                /\MAC1i':=H(CP9i'.IDuei.IDsnj.exp(H(Kuei'.IDuei'),Uii').CP6i'.CP7i')%MAC(k2,iduei,idsnj,cp2k,cp6i,cp7i)
                /\SND({IDuei.CP5i'.exp(H(Kuei'.IDuei'),Uii').CP6i'}_CP8i'.IDsnj.CP7i'.MAC1i')%Enc_k1(m), idsnj,
cp7i,mac1i
                /\witness(UE,SN,ue_sn_xii,Xii')
%-----3.user authentication response
4. State=3
/\RCV({exp(H(Ksn.IDsnj),Yjj').exp(H(Kue.IDuei),Sjj).H(exp(exp(H(Kue.IDuei),Uii),Sjj).IDuei.IDsnj.CP10j'.exp(H(Ksn.ID
snj),Xii'))}_CP8i) =|>
State':=4 /\CP12j':=exp(exp(H(Kue.IDuei),Sjj),Uii)%k3 T_ui(T_sj(Kue||iduei))
                /\MAC2j':=H(CP12j'.IDuei.IDsnj.exp(H(Ksn.IDsnj),Yjj').exp(H(Ksn.IDsnj),Xii))
                /\SKij':=H(exp(exp(H(Ksn.IDsnj),Yjj),Xii).CP8i.CP9i.CP12j'.IDuei.IDsnj)
                /\request(UE,SN,sn_ue_yjj,MAC2j')
                /\MAC3i':=H(SKij'.IDuei.IDsnj.CP7i.exp(H(Ksn.IDsnj),Yjj'))
                /\SND(MAC3i')
                /\witness(UE,SN,ue_sn_skij,MAC3i')

end role
```

Figure 4. Scenario 1: HLPSL description of UE role.

```
%%%%%%% Role KGC %%%%%%
role kgcenter(SN,UE,KGC: agent, SKsnc,SKunc: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by KGC
def=
local State: nat,
                Ksn,Kue,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,MAC2j,SKij:text,
                IDuei,Uii,Kuei,Xii,CP5i,CP6i,CP7i,CP8i,CP9i,MAC1i,MAC3i : text,
                CP1k,CP2k,HCP2k,CP3k,CP4k,HCP3k,HCP4k:text

const sp1,sp2,sp3,sp4,ue_sn_xii,sn_ue_yjj,ue_sn_skij: protocol_id
init State:=0
transition

%%%%%% Registration of SN %%%%%%%%
1. State=0 /\RCV({IDsnj}_SKsnc)=|>
State':=1 /\Sjj':=new()
                /\CP1k':=exp(H(Ksn.IDsnj),Sjj')
                /\SND({CP1k'.Sjj'.Ksn.Kue}_SKsnc)
                /\secret({Sjj'},sp1,{SN,KGC})
%%%%%% Registration of UE %%%%%%%%
2. State=1 /\RCV({IDuei'}_SKunc)=|>
State':=2 /\Uii':=new() /\Kuei':=new()
                /\CP2k':=exp(H(Kuei'.IDuei'),Uii')
                /\SND({CP2k'.Uii'}_SKunc)
                /\secret({CP2k',Uii'},sp2,{UE,KGC})
                /\HCP2k':=H(exp(H(Kuei'.IDuei'),Uii').IDuei)
                /\SND({HCP2k'}_SKsnc)
                /\secret({HCP2k'},sp3,{SN,KGC})


end role
```

Figure 5. Scenario 1: HLPSL description of KGC role.

```
%%%%%%%%%%   session  %%%%%%%
role session(SN,UE,KGC : agent, SKsnc,SKunc,SKuni :symmetric_key, H: hash_func)

def=
local SN1, SN2, SN3, RV1, RV2, RV3 : channel(dy)
composition
servingnetwork(SN,UE,KGC,SKsnc,H,SN1,RV1)
/\userequipment(SN,UE,KGC,SKunc,SKuni,H,SN2,RV2)
/\kgcenter(SN,UE,KGC,SKsnc,SKunc,H,SN3,RV3)
end role

%%%%%  environments and goals  %%%%%%%
role environment()

def=
const sn,ue,kgc : agent,
sksnc, skunc, skuni: symmetric_key,
h: hash_func,
ksn,kue,idsnj,iduei,cp1k,cp2k,cp3k,cp4k,uii,kuei,xii: text,
ue_sn_xii,sn_ue_yjj,ue_sn_skij: protocol_id,
sp1,sp2,sp3,sp4: protocol_id

intruder_knowledge = {sn,ue,kgc,ksn,kue,idsnj,iduei,skunc,skuni,cp1k,cp2k,cp3k,cp4k,uii,xii,kuei,h}%intruder knows the
information in UE
composition
session(sn,ue,kgc,sksnc,skunc,skuni,h)
%/\session(i,ue,kgc,sksnc,skunc,skuni,h)
%/\session(sn,i,kgc,sksnc,skunc,skuni,h)
end role

goal
secrecy_of sp1,sp2,sp3,sp4
authentication_on ue_sn_xii
authentication_on sn_ue_yjj
authentication_on ue_sn_skij
end goal

environment()
```

Figure 6. Scenario 1: HLPSL description of session and environment.

```
%%%%%%% Role SN %%%%%%
role servingnetwork(SN,MDA,MDB,KGC: agent, SKsnc: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by SN
def=
local State: nat,
                Ksn,Kmd,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,SKaj,SKbj:text,
                IDuei,Uii,Kuei,Xii,Xbi,Zbi,Zii,IDmda,IDmdb,GID,Mgg,Kmdg,Kmda,Kmdb,Mga,Mgb,CP13g,CP14g,CP15g,C
P16g,CP17g,CP18g,MAC1a,MAC1b: text,
                CP19g,CP20g,CP21g,CP23g,CP24g,CP25g,CP26g,CP27g,CP28bg,CP28ag: text,
                CP1k,CP3k,HCP3k,CP4ak,CP4bk,HCP4bk,HCP4ak:text

const sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8,mda_SN_xii,mdb_SN_xbi,mda_SN_skaj,mdb_SN_skbj: protocol_id
init State:=0
transition

%%%%% Registration phase %%%%%%%%
1. State=0 /\RCV(start)=|>
State':=1 /\SND({IDsnj}_SKsnc)
%%%%% SN receives msg from KGC %%%%%%
2. State=1 /\RCV({exp(H(Ksn.IDsnj),Sjj').Sjj'.Ksn.Kmd}_SKsnc)=|>
State':=2 /\SND(IDsnj.exp(H(Ksn.IDsnj),Sjj').Ksn.Kmd)%send public parameter to md
3. State=2 /\RCV({H(exp(H(Kmdg'.GID),Mgg').GID).H(exp(H(Kmdb'.GID.IDmdb),Mgb').GID.IDmdb')}_SKsnc)=|>
State':=3
4. State=3 /\RCV({H(exp(H(Kmdg'.GID),Mgg').GID).H(exp(H(Kmda'.GID.IDmda),Mga').GID.IDmda')}_SKsnc)=|>
State':=4

%%%%% Authentication phase mda-mdb-sn%%%%%%
%-----3.group authentication request
%----MDa(i)
5. State=4
/\RCV({IDmda.exp(H(Kmda'.GID.IDmda),Mga').exp(H(Kmd.GID),Mga').exp(H(Ksn.IDsnj),Mga').Zii'}_exp(exp(H(Ksn.ID
snj),Sjj'),Xii').IDsnj.exp(H(Ksn.IDsnj),Xii').H(exp(exp(H(Ksn.IDsnj),Sjj'),Xii').exp(exp(H(Ksn.IDsnj),Sjj),Mga').exp(exp(H(
Ksn.IDsnj),Sjj),Mgg').GID.IDmda.IDsnj.exp(H(Kmda'.GID.IDmda),Mga').exp(H(Kmdg'.GID),Mgg').exp(H(Ksn.IDsnj),Xii'
).Zii'))=|>
State':=5 /\CP18g':=exp(exp(H(Ksn.IDsnj),Xii'),Sjj)%K_m1i
                /\CP15g':=exp(exp(H(Ksn.IDsnj),Mga'),Sjj)%k_m2i
                /\request(SN,MDA,mda_SN_xii,Xii')
%----MDb(n)
6. State=5
/\RCV({GID.IDmdb.exp(H(Kmdb'.GID.IDmdb),Mgb').exp(H(Kmdb.GID),Mgg').exp(H(Ksn.IDsnj),Mgb').exp(H(Kmd.GID
),Mgb').exp(H(Ksn.IDsnj),Mgg').exp(H(Kmd.GID),Mgg').Zbi'}_exp(exp(H(Ksn.IDsnj),Sjj'),Xbi').IDsnj.exp(H(Ksn.IDsnj),X
bi').H(exp(exp(H(Ksn.IDsnj),Sjj'),Xbi').exp(exp(H(Ksn.IDsnj),Sjj),Mgb').exp(exp(H(Ksn.IDsnj),Sjj),Mgg').GID.IDmdb.IDs
nj.exp(H(Kmdb'.GID.IDmdb),Mgb').exp(H(Kmdg'.GID),Mgg').exp(H(Ksn.IDsnj),Xbi').Zbi'))=|>
State':=6 /\CP24g':=exp(exp(H(Ksn.IDsnj),Xbi'),Sjj)%K_m1b
                /\CP21g':=exp(exp(H(Ksn.IDsnj),Mgb'),Sjj)%k_m2b
                /\CP16g':=exp(exp(H(Ksn.IDsnj),Mgg'),Sjj)%KG1
                /\Yjj':=new()
                /\CP25g':=exp(H(Ksn.IDsnj),Yjj')%T_yj(ksn||idsnj)
                /\CP26g':=exp(H(Kmd.GID),Sjj)%T_sj(kmd||idsnj)
                /\CP27g':=exp(exp(H(Kmd.GID),Mgg'),Sjj)%KG2 = T_sj(T_mg(kmd||gid))
                /\CP28ag':=exp(exp(H(Kmd.GID),Mga),Sjj)%Km3a = T_sj(T_ma(kmd||gid))
                /\CP28bg':=exp(exp(H(Kmd.GID),Mgb'),Sjj)%Km3b = T_sj(T_mb(kmd||gid))
                /\SKaj':=H(exp(exp(H(Ksn.IDsnj),Xii'),Yjj').CP18g.CP15g.CP28ag'.CP16g'.CP27g'.IDmda.GID.IDsnj)
                /\SKbj':=H(exp(exp(H(Ksn.IDsnj),Xbi'),Yjj').CP24g'.CP21g'.CP28bg'.CP16g'.CP27g'.IDmdb.GID.IDsnj)
                /\request(SN,MDB,mdb_SN_xbi,Xbi')
                /\SND({CP25g'.CP26g'}_CP16g')
                /\witness(SN,MDA,mda_SN_skaj,SKaj')
                /\witness(SN,MDB,mdb_SN_skbj,SKbj')
end role
```

Figure 7. Scenario 2: HLPSL description of SN role.

```
%%%%%%% Role MDA %%%%%%
role mtcdevicesA(SN,MDA,MDB,KGC: agent, SKmanc,SKanb: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by MDA
def=
local State: nat,
                Ksn,Kmd,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,SKaj,SKbj:text,
                IDuei,Uii,Kuei,Xii,Xbi,Zbi,Zii,IDmda,IDmdb,GID,Mgg,Kmdg,Kmda,Kmdb,Mga,Mgb,CP13g,CP14g,CP15g,C
P16g,CP17g,CP18g,MAC1a,MAC1b: text,
                CP19g,CP20g,CP21g,CP23g,CP24g,CP25g,CP26g,CP27g,CP28bg,CP28ag: text,
                CP1k,CP3k,HCP3k,CP4ak,CP4bk,HCP4bk,HCP4ak:text

const sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8,mda_SN_xii,mdb_SN_xbi,mda_SN_skaj,mdb_SN_skbj: protocol_id
init State:=0
transition

%%%%% Registration phase %%%%%%%%
1. State=0 /\RCV(IDsnj.exp(H(Ksn.IDsnj),Sjj').Ksn.Kmd)=|>
State':=1
2. State=1 /\RCV({GID}_SKanb)=|>
State':=2 /\SND({IDmda.GID}_SKmanc)
                /\secret(IDmda,sp8,{MDA,KGC})
%%%%% authentication phase %%%%%%%%
3. State=2 /\RCV({exp(H(Kmdg'.GID),Mgg').exp(H(Kmda'.GID.IDmda),Mga').Mgg'.Mga'}_SKmanc)=|>
State':=3 /\CP13g':=exp(H(Ksn.IDsnj),Mga')%T_ma(ksn||idsnj)
                /\CP14g':=exp(H(Kmd.GID),Mga')%T_ma(kmd||gid)
                /\CP15g':=exp(exp(H(Ksn.IDsnj),Sjj),Mga')%Km2ib  = T_ma(T_sj(ksn||idsnj))
                /\CP16g':=exp(exp(H(Ksn.IDsnj),Sjj),Mgg')%KG1  = T_mg(T_sj(ksn||idsnj))
                /\Xii':=new() /\Zii':=new()
                /\CP17g':=exp(H(Ksn.IDsnj),Xii')%T_xi(ksn||idsnj)
                /\CP18g':=exp(exp(H(Ksn.IDsnj),Sjj),Xii')%K_m1i = T_xi(T_sj(ksn||idsnj))
                /\MAC1a':=H(CP18g'.CP15g'.CP16g'.GID.IDmda.IDsnj.exp(H(Kmda'.GID.IDmda),Mga').exp(H(Kmdg'.GID),
Mgg').CP17g'.Zii')
                %mac=H(cp18g,cp15g,cp16g,gid,idmda,idsng,cp4,cp3,cp17g,zii)
                /\SND({IDmda.exp(H(Kmda'.GID.IDmda),Mga').CP14g'.CP13g'.Zii'}_CP18g'.IDsnj.CP17g'.MAC1a')
                /\witness(MDA,SN,mda_SN_xii,Xii')
4. State=3 /\RCV({exp(H(Ksn.IDsnj),Yjj').exp(H(Kmd.GID),Sjj)}_CP16g)=|>
State':=4 /\CP27g':=exp(exp(H(Kmd.GID),Sjj),Mgg)%KG2
                /\CP28ag':=exp(exp(H(Kmd.GID),Sjj),Mga)%Km3a
                /\SKaj':=H(exp(exp(H(Ksn.IDsnj),Yjj'),Xii).CP18g.CP15g.CP28ag'.CP16g.CP27g'.IDmda.GID.IDsnj)
                /\request(MDA,SN,mda_SN_skaj,SKaj')
end role
```

Figure 8. Scenario 2: HLPSL description of MD_i role.

```
%%%%%%% Role MDB %%%%%%
role mtcdevicesB(SN,MDA,MDB,KGC: agent, SKmbnc,SKanb: symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by MDB
def=
local State: nat,
                Ksn,Kmd,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,SKaj,SKbj:text,
                IDuei,Uii,Kuei,Xii,Xbi,Zbi,Zii,IDmda,IDmdb,GID,Mgg,Kmdg,Kmda,Kmdb,Mga,Mgb,CP13g,CP14g,CP15g,C
P16g,CP17g,CP18g,MAC1a,MAC1b: text,
                CP19g,CP20g,CP21g,CP23g,CP24g,CP25g,CP26g,CP27g,CP28bg,CP28ag: text,
                CP1k,CP3k,HCP3k,CP4ak,CP4bk,HCP4bk,HCP4ak:text

const sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8,mda_SN_xii,mdb_SN_xbi,mda_SN_skaj,mdb_SN_skbj: protocol_id
init State:=0
transition

%%%%% Registration phase %%%%%%%%
1. State=0 /\RCV(IDsnj.exp(H(Ksn.IDsnj),Sjj').Ksn.Kmd)=|>
State':=1 /\SND({GID}_SKanb) /\SND({IDmdb.GID}_SKmbnc)
                /\secret(IDmdb,sp7,{MDB,KGC})
%%%%% authentication phase %%%%%%%%
%-----2.aggregation access request
2. State=1 /\RCV({exp(H(Kmdg'.GID),Mgg').exp(H(Kmdb'.GID.IDmdb),Mgb').Mgg'.Mgb'}_SKmbnc)=|>
State':=2 /\CP19g':=exp(H(Ksn.IDsnj),Mgb')%T_mb(ksn||idsnj)
                /\CP20g':=exp(H(Kmd.GID),Mgb')%T_mb(kmd||gid)
                /\CP21g':=exp(exp(H(Ksn.IDsnj),Sjj),Mgb')%Km2ib  = T_mb(T_sj(ksn||idsnj))
                /\CP16g':=exp(exp(H(Ksn.IDsnj),Sjj),Mgg')%KG1  = T_mg(T_sj(ksn||idsnj))%equal mda
                /\Xbi':=new() /\Zbi':=new()
                /\CP23g':=exp(H(Ksn.IDsnj),Xbi')%T_xbb(ksn||idsnj)
                /\CP24g':=exp(exp(H(Ksn.IDsnj),Sjj),Xbi')%K_m1b = T_xbi(T_sj(ksn||idsnj))
                /\MAC1b':=H(CP24g'.CP21g'.CP16g'.GID.IDmdb.IDsnj.exp(H(Kmdb'.GID.IDmdb),Mgb').exp(H(Kmdg'.GID),
Mgg').CP23g'.Zbi')%mac=
                /\SND({GID.IDmdb.exp(H(Kmdb'.GID.IDmdb),Mgb').exp(H(Kmdb.GID),Mgg').CP20g'.CP19g'.exp(H(Ksn.ID
snj),Mgg').exp(H(Kmd.GID),Mgg').Zbi'}_CP24g'.IDsnj.CP23g'.MAC1b')
                /\witness(MDB,SN,mdb_SN_xbi,Xbi')
%-----1.aggregation access request
3. State=2 /\RCV({exp(H(Ksn.IDsnj),Yjj').exp(H(Kmd.GID),Sjj)}_CP16g)=|>
State':=3 /\CP27g':=exp(exp(H(Kmd.GID),Sjj),Mgg)%KG2
                /\CP28bg':=exp(exp(H(Kmd.GID),Sjj),Mgb)%Km3a
                /\SKbj':=H(exp(exp(H(Ksn.IDsnj),Yjj'),Xbi).CP24g.CP21g.CP28bg'.CP16g.CP27g'.IDmdb.GID.IDsnj)
                /\request(MDB,SN,mdb_SN_skbj,SKbj')

end role
```

Figure 9. Scenario 2: HLPSL description of MD_n role.

```
%%%%%%%% Role KGC %%%%%%
role kgcenter(SN,MDA,MDB,KGC: agent, SKsnc,SKmbnc,SKmanc: symmetric_key, H: hash_func, SND, RCV :
channel(dy)) %---here!!!
played_by KGC
def=
local State: nat,
                Ksn,Kmd,IDsnj,Sjj,Yjj,CP10j,CP11j,CP12j,SKaj,SKbj:text,
                IDuei,Uii,Kuei,Xii,Xbi,Zbi,Zii,IDmda,IDmdb,GID,Mgg,Kmdg,Kmda,Kmdb,Mga,Mgb,CP13g,CP14g,CP15g,C
P16g,CP17g,CP18g,MAC1a,MAC1b: text,
                CP19g,CP20g,CP21g,CP23g,CP24g,CP25g,CP26g,CP27g,CP28bg,CP28ag: text,
                CP1k,CP3k,HCP3k,CP4ak,CP4bk,HCP4bk,HCP4ak:text

const sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8,mda_SN_xii,mdb_SN_xbi,mda_SN_skaj,mdb_SN_skbj: protocol_id
init State:=0
transition

%%%%%% Registration of SN, UE, MD %%%%%%%%
1. State=0 /\RCV({IDsnj}_SKsnc)=|>
State':=1 /\Sjj':=new()
                /\CP1k':=exp(H(Ksn.IDsnj),Sjj')% T_sj(Ksn||IDsnj)
                /\SND({CP1k'.Sjj'.Ksn.Kmd}_SKsnc)
                /\secret({Sjj'},sp1,{SN,KGC})
%%%%%% Registration of mdb %%%%%%%%
2. State=1 /\RCV({IDmdb.GID}_SKmbnc)=|>
State':=2 /\Mgg':=new() /\Mgb':=new() /\Kmdg':=new() /\Kmdb':=new()
                /\CP3k':=exp(H(Kmdg'.GID),Mgg')%T_mg(kmdg||gid)                                    group
                /\CP4bk':=exp(H(Kmdb'.GID.IDmdb),Mgb')%T_mi(kmdi||gid||idmdi)          individual
                /\HCP3k':=H(CP3k'.GID)
                /\HCP4bk':=H(CP4bk'.GID.IDmdb)
                /\SND({CP3k'.CP4bk'.Mgg'.Mgb'}_SKmbnc)
                /\secret({CP3k',Mgg'},sp2,{MDA,MDB,KGC})
                /\secret({CP4bk',Mgb'},sp3,{MDB,KGC})
                /\SND({HCP3k'.HCP4bk'}_SKsnc)
                /\secret({HCP3k',HCP4bk'},sp4,{SN,KGC})
%%%%%% Registration of mda %%%%%%%%
3. State=2 /\RCV({IDmda.GID}_SKmanc)=|>
State':=3 /\Mga':=new() /\Kmda':=new()
                %/\CP3k':=exp(H(Kmdg'.GID),Mgg')%T_mg(kmdg||gid)
                /\CP4ak':=exp(H(Kmda'.GID.IDmda),Mga')%T_ma(kmda||gid||idmda)
                %/\HCP3k':=H(CP3k'.GID)
                /\HCP4ak':=H(CP4ak'.GID.IDmda)
                /\SND({CP3k.CP4ak'.Mgg.Mga'}_SKmanc)
                /\secret({CP4ak',Mga'},sp5,{MDA,KGC})
                /\SND({HCP3k.HCP4ak'}_SKsnc)
                /\secret({HCP4ak'},sp6,{SN,KGC})

end role
```

Figure 10. Scenario 2: HLPSL description of KGC role.

```
%%%%%%%%%%  session  %%%%%%%
role session(SN,MDA,MDB,KGC : agent, SKsnc,SKmanc,SKmbnc,SKanb :symmetric_key, H: hash_func)

def=
local SN1, SN2, SN3, SN4, RV1, RV2, RV3, RV4 : channel(dy)
composition
servingnetwork(SN,MDA,MDB,KGC,SKsnc,H,SN1,RV1)
/\mtcdevicesA(SN,MDA,MDB,KGC,SKmanc,SKanb,H,SN2,RV2)
/\mtcdevicesB(SN,MDA,MDB,KGC,SKmbnc,SKanb,H,SN3,RV3)
/\kgcenter(SN,MDA,MDB,KGC,SKsnc,SKmbnc,SKmanc,H,SN4,RV4)
end role

%%%%%  environments and goals  %%%%%%%
role environment()

def=
const sn,mda,mdb,kgc : agent,
sksnc, skmanc, skmbnc, skanb: symmetric_key,
h: hash_func,
ksn,kmd,idsnj,idmda,idmdb,gid: text,
mda_SN_xii,mdb_SN_xbi,mda_SN_skaj,mdb_SN_skbj: protocol_id,
sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8: protocol_id

intruder_knowledge = {sn,mda,mdb,kgc,ksn,kmd,idsnj,idmda,idmdb,gid,skmbnc,skmanc,h}%intruder knows the
information in MDs
composition
session(sn,mda,mdb,kgc,sksnc,skmanc,skmbnc,skanb,h)
end role

goal
secrecy_of sp1,sp2,sp3,sp4,sp5,sp6,sp7,sp8
authentication_on mda_SN_xii
authentication_on mdb_SN_xbi
authentication_on mda_SN_skaj
authentication_on mdb_SN_skbj
end goal

environment()
```

Figure 11. Scenario 2: HLPSL description of session and environment.

3.3.2. Simulation Results

To demonstrate that Cao et al.'s scheme is not secure against replay and man-in-the-middle attacks, we simulated the OMFC and CL-AtSe using the pre-defined HLPLS of scenario 1 and 2.

Simulation Result of Scenario 1: Under the OFMC back-ends, the search depth is 3 when 3 nodes have been searched in 0.8 s. Under the CL-AtSe, the translation time is 0.05 s and 2 states are analyzed.

Simulation Result of Scenario 2: Under the OFMC back-ends, the search depth is 3 when 3 nodes have been searched in 0.2 s. Under the CL-AtSe, the translation time is 0.02 s and 2 states are analyzed.

Figures 12 and 13 and present the results of OFMC and CL-AtSe back-ends, which presents "UNSAFE". Therefore, the scenario 1 and 2 are not secure against replay and man-in-the-middle attacks.

```
SUMMARY
  UNSAFE

DETAILS
  ATTACK_FOUND
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/avispa(ue-sn).if

GOAL
  Secrecy attack on (n9(IDuei))

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 2 states
  Reachable  : 1 states
  Translation: 0.04 seconds
  Computation: 0.00 seconds
```

(a) CL-AtSe result of Scenario 1

```
% OFMC
% Version of 2006/02/13
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  /home/span/span/testsuite/results/avispa(ue-sn).if
GOAL
  secrecy_of_sp4
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.08s
  visitedNodes: 3 nodes
  depth: 3 plies
```

(b) OFMC Result of Scenario 1

Figure 12. Simulation result of Scenario 1.

```
SUMMARY
 UNSAFE

DETAILS
 ATTACK_FOUND
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/avispa(md-sn).if

GOAL
 Secrecy attack on (dummy_nonce)

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 2 states
 Reachable  : 1 states
 Translation: 0.20 seconds
 Computation: 0.00 seconds
```

(a) CL-AtSe result of Scenario 2

```
% OFMC
% Version of 2006/02/13
SUMMARY
 UNSAFE
DETAILS
 ATTACK_FOUND
PROTOCOL
 /home/span/span/testsuite/results/avispa(md-sn).if
GOAL
 secrecy_of_sp7
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.20s
 visitedNodes: 3 nodes
 depth: 3 plies
```

(b) OFMC Result of Scenario 2

Figure 13. Simulation result of Scenario 2.

4. Security Fixes

In Cao et al.'s scheme [9], the major security issues is that the secret parameters are stored in a smartcard (SC) or memory of devices without applying any cryptographic method. An adversary can easily extract and obtain the secret parameters using power analysis because of this problem. It makes the scheme vulnerable to a single point of failure, UE impersonation, MD impersonation attacks, inability to achieve secure mutual authentication and to exhibit a wrong threat model. These major security issues are discussed in Section 3.

In the last decades, several authentication and key agreement protocols have been designed to ensure user and data privacy. These protocols store the secret parameters in the memory of devices such as UE, micro-sensor, smartcard, etc., which use it to authenticate and establish the session key between entities. Like other studies, Cao et al. proposed the LSAA scheme using the same approach. However, according to Cao et al.'s threat model, we assume that an adversary can easily compromise the physical IoT devices and extract the sensitive data stored in the memory of the devices. Although Cao et al. realized that an adversary can easily compromise IoT devices, they did not consider these problems in the design of their proposed protocol.

Herein, we suggest the necessary guidelines to mitigate the security weaknesses of the Cao et al.'s scheme.

Fix 1. In the registration phase of Cao et al.'s scheme, the KGC should not issue the secret parameters as plaintext to prevent stolen device attacks. The UE and MD should store the secret parameters in encrypted form using XOR operation and a hash function.

Fix 2. The LSAA scheme adopts the two-factor authentication technique using smartcard and secret parameters. However, the LSAA scheme does not verify whether UE and MD are from a legitimate entity that has the same security as the one-factor authentication scheme. Thus, the UE and MD need to ensure that the user is a legitimate user using a password or biometrics to improve the security level. We suggest the three-factor authentication with biometrics using a fuzzy extractor [24]

Fix 3. In the Cao et al.'s scheme, the KGC selects only one MTC group leader per group and this leads to a single point of failure attack. The KGC recodes the group leader list and related parameters in a blockchain to ensure that all members can freely access the leader of other groups. This prevents the issue of a single point of failure attack because the group members can freely access other group leaders when there is a problem with its group leader.

In these suggested solutions, the UE, and MD impersonation attacks can be mitigated and we do not assert that our suggested solutions are perfect against the above-mentioned security issues. However, it will definitely improve the security of the system and increase the attack complexity for an adversary.

Cao et al. did well by designing a novel group access authentication scheme in 5G networks. However, they would have looked at their scheme from various angles. Bringing improvements in a field of study is a difference in the individual approaches of researchers. Surely, this paper will bring about awareness of the need to design a secure and efficient authentication scheme for IoT environments.

5. Conclusions

This paper refers to "LSAA: A lightweight and secure access authentication scheme for both UE and mMTC devices in 5G networks". We demonstrated that Cao et al.'s scheme is vulnerable to single point of failure and impersonation attacks, does not provide secure mutual authentication, and does not meet the security requirement of their proposed threat model. Moreover, we prove that their scheme does not achieve the session key security using formal (mathematics) analysis, and perform the simulation test using AVISPA tool to demonstrate their security weakneeses. The above-mentioned security flaws make Cao et al. scheme inappropriate and impractical to utilize. Thus, we suggested ways of improving the security level which can lead to a more secure and efficient scheme for 5G based IoT environments.

Author Contributions: Conceptualization, K.P.; methodology, K.P.; validation, K.P.; formal analysis, K.P.; writing—original draft preparation, K.P.; writing—review and editing, Y.P.; supervision, Y.P.; project administration, Y.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [22ZR1330, Research on Intelligent Cyber Security and Trust Infra].

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

References

1. Chettri, L.; Bera, R. A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems. *IEEE Internet Things J.* **2020**, *7*, 16. [CrossRef]
2. Lopez-Ballester, J.; Pastor-Aparicio, A.; Felici-Castell, S.; Segura-Garcia, J.; Cobos, M. Enabling Real-Time Computation of Psycho-Acoustic Parameters in Acoustic Sensors Using Convolutional Neural Networks. *IEEE Sens. J.* **2020**, *20*, 11429. [CrossRef]
3. Park, Y.; Park, Y. Three-Factor User Authentication and Key Agreement Using Elliptic Curve Cryptosystem in Wireless Sensor Networks. *Sensors* **2016**, *16*, 2123. [CrossRef] [PubMed]

4. Moon, J.; Lee, D.; Lee, Y.; Won, D. Improving Biometric-Based Authentication Schemes with Smart Card Revocation/Reissue for Wireless Sensor Networks. *Sensors* **2017**, *17*, 940. [CrossRef] [PubMed]
5. Wang, C.; Xu, G.; Sun, J. An Enhanced Three-Factor User Authentication Scheme Using Elliptic Curve Cryptosystem for Wireless Sensor Networks. *Sensors* **2017**, *17*, 2946. [CrossRef] [PubMed]
6. Wu, F.; Xu, L.; Kumari, S.; Li, X. A privacy-preserving and provable user authentication scheme for wireless sensor networks based on Internet of Things security. *J. Ambient Intell. Hum. Comput.* **2017**, *8*, 101–116. [CrossRef]
7. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of secure user authenticated key management protocol for generic IoT networks. *IEEE Internet Things J.* **2018**, *5*, 269–282. [CrossRef]
8. Adavoudi-Jolfaei, A.; Ashouri-Talouki, M.; Aghili, S.F. Lightweight and anonymous three-factor authentication and access control scheme for real-time applications in wireless sensor networks. *Peer-Peer Netw. Appl.* **2019**, *12*, 43–59. [CrossRef]
9. Cao, J.; Yan, Z.; Ma, R.; Zhang, Y.; Fu, Y.; Li, H. LSAA: A Lightweight and Secure Access Authentication Scheme for Both UE and mMTC Devices in 5G Networks. *IEEE Internet Things J.* **2020**, *7*, 5329. [CrossRef]
10. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smartcard security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541. [CrossRef]
11. Sureshkumar, V.; Amin, R.; Obaidat, M.S.; Karthikeyan, I. An enhanced mutual authentication and key establishment protocol for TMIS using chaotic map. *J. Inf. Secur. Appl.* **2020**, *53*, 102539. [CrossRef]
12. Xiong, H.; Kang, Z.; Chen, J.; Tao, J.; Yuan, C.; Kumari, S. A novel multiserver authentication scheme using proxy resignature with scalability and strong user anonymity. *IEEE Syst. J.* **2020**, *2*, 2156. [CrossRef]
13. AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: http://people.irisa.fr/Thomas.Ge\net/span/ (accessed on 8 April 2022).
14. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198. [CrossRef]
15. Eisenbarth, T.; Kasper, T.; Moradi, A.; Paar, C.; Salmasizadeh, M.; Shalmani, M.T.M. On the power of power analysis in the real world: A complete break of the KEELOQ code hopping scheme. In *Advances in Cryptology–CRYPTO*; Springer: Heidelberg, Germany, 17–21 August 2008; pp. 203–220.
16. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology–CRYPTO*; Springer: Heidelberg, Germany, 15–19 August 1999; pp. 388–397.
17. Abdalla, M.; Fouque, P.; Pointcheval, D. Password-based authenticated key exchange in the three-party setting. In Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science (LNCS), Les Diablerets, Switzerland, 23–26 January 2005; pp. 65–84.
18. Yu, S.; Lee, J.; Park, K.; Das, A.K.; Park, Y. IoV-SMAP: Secure and Efficient Message Authentication Protocol for IoV in Smart City Environment. *IEEE Access* **2020**, *8*, 167875. [CrossRef]
19. Park, K.; Lee, J.; Das, A.K.; Park, Y. BPPS: Blockchain-Enabled Privacy-Preserving Scheme for Demand-Response Management in Smart Grid Environments. *IEEE Trans. Dependable Secur. Comput.* **2022**. [CrossRef]
20. Srinivas, J.; Das, A.K.; Wazid, M.; Kumar, N.J. Anonymous Lightweight Chaotic Map-based Authenticated Key Agreement Protocol for Industrial Internet of Things. *IEEE Internet Things J.* **2020**, *17*, 1133. [CrossRef]
21. Park, K.; Noh, S.; Lee, H.; Das, A.K.; Kim, M.; Park, Y.; Wazid, M. LAKS-NVT: Provably Secure and Lightweight Authentication and Key Agreement Scheme without Verification Table in Medical Internet of Things. *IEEE Access* **2020**, *8*, 119387. [CrossRef]
22. Von Oheimb, D. The high-level protocol specification language HLPSL developed in the EU project avispa. In Proceedings of the APPSEM 2005 Workshop, Tallinn, Finland, 13–15 September 2005; pp. 1–2.
23. Vigano, L. Automated Security Protocol Analysis with the AVISPA Tool. *Electron. Notes Theor. Comput. Sci.* **2006**, *155*, 61. [CrossRef]
24. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **2008**, *38*, 97. [CrossRef]