

Article

A Novel Authentication Protocol with Strong Security for Roaming Service in Global Mobile Networks

Shanpeng Liu ¹, Xiong Li ^{1,*}, Fan Wu ², Junguo Liao ¹, Jin Wang ³  and Dingbao Lin ²¹ School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China² Department of Computer Science and Engineering, Xiamen Institute of Technology, Xiamen 361021, China³ School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha 410004, China

* Correspondence: lixiongzhq@163.com

Received: 22 July 2019; Accepted: 22 August 2019; Published: 26 August 2019



Abstract: In today's society, Global Mobile Networks (GLOMONETs) have become an important network infrastructure that provides seamless roaming service for mobile users when they leave their home network. Authentication is an essential mechanism for secure communication among the mobile user, home network, and foreign network in GLOMONET. Recently, Madhusudhan and Shashidhara presented a lightweight authentication protocol for roaming application in GLOMONET. However, we found their protocol not only has design flaws, but is also vulnerable to many attacks. To address these weaknesses, this paper proposes a novel authentication protocol with strong security for GLOMONET based on previous work. The fuzzy verifier technique makes the protocol free from smart card breach attack, while achieving the feature of local password change. Moreover, the computational intractability of the Discrete Logarithm Problem (DLP) guarantees the security of the session key. The security of the protocol is verified by the ProVerif tool. Compared with other related protocols, our protocol achieves a higher level of security at the expense of small increases in computational cost and communication cost. Therefore, it is more suitable for securing the roaming application in GLOMONET.

Keywords: global mobile network; authentication; roaming service; session key agreement; password

1. Introduction

The network enriches the way people access information, and technologies such as wireless sensor networks (WSN) [1–4], multi-hop wireless networks [5,6], and the Internet of Things (IoT) [7–9] have greatly advanced the intelligence level of peoples' lives. Benefiting from the development of telecommunication technologies such as 4G, 5G, and Wi-Fi, wireless and mobile communication [10–12] have become the most popular means of communication. Meanwhile, the trend of intelligence and miniaturization of devices makes it a reality for people to deal with daily affairs through mobile terminals for various AI applications [13,14]. The Global Mobile Network (GLOMONET) is an important infrastructure for mobile users to enjoy the network anytime and anywhere, which provides roaming service for the mobile users. In the general architecture of GLOMONET environments, there are three entities, namely the Mobile User (MU), the Foreign Agent (FA) of the roaming network, and the Home Agent (HA) of the registered network. Due to the complexity of the wireless network communication environment, it faces serious security and privacy breach threats [15–18]. For example, in GLOMONET, in order to facilitate billing and secure communication, FA needs to verify the validity of MU with the assistance of the HA. Therefore, authentication is an indispensable mechanism for GLOMONET. Meanwhile, session key agreement is also essential for subsequent secure communication

after the bidirectional authentication between *MU* and *FA*. Besides, the update of the session key is an ideal feature for authentication in GLOMONET if *MU* wants to access *FA* continuously.

As early as 2004, Zhu et al. [19] first proposed an anonymous authentication mechanism for GLOMONET, which allows *MU* to access *FA* after authentication with the assistance of *HA*. Unfortunately, Lee et al. [20] analyzed the drawbacks of the mechanism in [19], i.e., they found that it was vulnerable to forgery attack and could not provide bidirectional authentication and backward secrecy, and they proposed an enhanced scheme. However, the work in [21] and [22] independently pointed out the shortcomings of the scheme in [20], such as it could not ensure user anonymity and could not achieve backward secrecy. In 2011, Xu et al. [23] further analyzed the defects of the scheme in [20] and pointed out that the anonymity feature of the scheme in [20] could be compromised by a malicious insider. Besides, they found that the agreement of the session key between *MU* and *FA* was unfair, and it could be decided unilaterally by *MU*. Based on the Diffie–Hellman Protocol (DHP), Xu et al. [23] devised an authentication scheme preserving anonymity for GLOMONET to achieve desirable security and fair session key agreement. Later, many researchers proposed their special solutions for authentication in GLOMONET. In 2011, the work in [24] proposed a robust authentication scheme for GLOMONET, which used the certificate to achieve the three parties' authentication. Their scheme had many advantages, such as single registration, no verifier table, high efficiency, and resistance to smart card breach attack. However, Li et al. [25] found that the protocol in [24] was still vulnerable to a tracking attack and could not achieve fair key agreement. To fix these drawbacks, Li et al. proposed an improvement scheme [25] relying on DHP. However, their scheme needed more modular exponential operations than the previous scheme [23]. Based on elliptic curve cryptography, Yoon et al. [26] proposed a user-friendly authentication protocol for GLOMONET. However, Niu and Li [27] found that their scheme had the same drawbacks as the scheme in [24]. In 2012, Mun et al. [28] proposed an improved authentication scheme for GLOMONET. Due to it was designed on the basis of elliptic curve DHP, their scheme guaranteed forward secrecy and also could resist man-in-the-middle attack. However, their scheme lacked some ideal functions such as local password change and also suffered from several attacks [29]. Hereafter, many user authentication protocols [30–36] were proposed to improve the efficiency and enhance the security. In 2015, Marimuthu and Saravanan [30] designed an authentication for GLOMONET based on the Discrete Logarithm Protocol (DLP) and DHP. However, their protocol lacked the session key update function and could not achieve perfect forward secrecy [31]. Recently, Madhusudhan and Shashidhara [32] pointed out some other defects of the scheme in [30], such as stolen verifier attack, impersonation attack, and insider attack. They also proposed an improved authentication protocol for GLOMONET [32], which they claimed was secure and lightweight. However, their protocol had a fatal flaw in the design, i.e., some information that should be kept secret was transmitted by plaintext. Therefore, their protocol not only lacked proper bidirectional authentication, but also suffered from stolen verifier attack, mobile device breach attack, session key compromise attack, and user impersonation attack.

Although many solutions for user authentication in GLOMONET have been presented by researchers, most of them have certain defects in security or function. For example, most of the protocols did not address the contradiction between the function of local password change and resistance to smart card loss attack. To achieve the feature of local password change, some verification information should be stored in the mobile device or smart card. Therefore, an adversary $\mathcal{A}$ can use this verification information to guess the corresponding password when $\mathcal{A}$ obtains the mobile device or smart card. Besides, to ensure the security of the session key and secure update of the session key, public key cryptography is essential. In order to reconcile this contradiction and enhance security, we propose a novel authentication protocol with strong security for GLOMONET. The fuzzy verifier technique is used to balance the conflict between local password change and stolen smart card attack. A discrete logarithm based DHP was adopted to keep the session key secure and achieve the session key update.

The arrangement of the remaining sections is as follows: Section 2 reviews the protocol in [32] and points out their design and security flaws; our protocol, the corresponding formal proof, and the formal verification by ProVerif are presented in Sections 3–5, respectively; Section 6 discusses the security properties and compares it with other related protocols; Section 7 summarizes the full paper.

2. Review and Cryptanalysis of Madhusudhan and Shashidhara's Protocol

In this section, Madhusudhan and Shashidhara's scheme [32] for GLOMONET is reviewed, and then, the design and security defects of their scheme are pointed out. For convenience of description, the notations used throughout the paper are listed in Table 1.

Table 1. Notations.

Symbol	Meaning
MU, ID_{MU}, PW_{MU}	a mobile user and the corresponding identity and password
HA, ID_{HA}	MU 's home agent and the identity
FA, ID_{FA}	a foreign agent and the identity
(x, X)	secret and public key pair of HA
K_{HF}	a secret key shared between HA and FA
p, q	two large prime numbers
g	a generator of Z_p^*
n_0	an integer between $[2^6, 2^8]$ for the fuzzy verifier
E, D	a pair of symmetric encryption and decryption algorithms
$h(\cdot)$	the hash function
r_{MU}, r_{FA}	the random integers generated by MU and FA
$\oplus$	the XOR operation
l	secure length
$\mathcal{A}$	an adversary

2.1. Review of Madhusudhan and Shashidhara's Scheme

Their scheme [32] contained four phases, which were initialization, registration, authentication, and password change. Since the last phase was not involved in our analysis, only the first three phases were reviewed.

2.1.1. Initialization Phase

HA generates two large prime numbers p, q and chooses a generator g of Z_p^* . Besides, HA calculates $n = p \times q$ and $\phi(n) = (p - 1) \times (q - 1)$. Then, HA chooses an integer $e \in (1, \phi(n))$ and calculates $d = e^{-1} \bmod \phi(n)$, where $\gcd(e, \phi(n)) = 1$. Next, HA calculates and releases the public key $y = g^d \bmod n$ and retains $\{d, p, q\}$ secretly.

2.1.2. Registration Phase

To register as a legitimate user, MU interacts with HA as follows:

- Step 1. MU chooses identity ID_{MU} , password PW_{MU} , and produces a nonce N . Next, MU calculates $R_1 = h(ID_{MU} \| N)$ and submits it to HA securely.
- Step 2. Upon obtaining the registration request, HA calculates $R = (R_1 \| ID_{HA} \| d)$, $a = h(d)$, $C_{MU} = (g^a \bmod p) \oplus h(R)$. Then, HA initializes the value of the counter K to zero and saves $\{K, R\}$ in the database. Finally, HA submits $\{R, C_{MU}, K, h(\cdot)\}$ to MU via a secret manner.
- Step 3. After getting the reply message from HA , MU computes $K_{MU} = h(ID_{MU} \| PW_{MU} \| R)$ and stores $\{K_{MU}, R, C_{MU}, K, h(\cdot)\}$ in the mobile device.

2.1.3. Authentication Phase

Whenever MU roams to a foreign network, the following steps allow FA and MU to achieve bidirectional authentication with the assistance of HA .

- Step 1. MU types in ID_{MU} and PW_{MU} . The mobile device computes $K_{MU}^* = h(ID_{MU} || PW_{MU} || R)$ and checks $K_{MU}^* \stackrel{?}{=} K_{MU}$. The request is rejected if they are not equal. Otherwise, MU produces a nonce R_{MU} and computes $U = R \oplus R_{MU}$, $V = ((C_{MU} \oplus h(R)) || ID_{FA}) \oplus R_{MU}$ and $W = (U || K || (C_{MU} \oplus h(R)))$. Finally, MU submits the login request $M_1 = \{U, V, W\}$ to FA .
- Step 2. Upon receiving M_1 , FA produces a nonce R_{FA} , and encrypts M_1 with R_{FA} using K_{FH} . Then, FA submits $M_2 = \{ID_{FA}, E_{K_{FH}}(M_1, R_{FA})\}$ to HA .
- Step 3. After obtaining M_2 , HA checks ID_{FA} and finds the secret key K_{FH} . Next, HA retrieves $\{U, V, W, R_{FA}\}$ by calculating $D_{K_{FH}}(E_{K_{FH}}(M_1, R_{FA}))$ and calculates $a = h(d)$, $R_{MU}^* = V \oplus ((g^a \bmod p) || ID_{FA})$, $R^* = U \oplus R_{MU}^*$. Then, HA checks if R^* is preserved in the database. If so, HA calculates $W^* = (U || K || (g^a \bmod p))$ and checks $W^* \stackrel{?}{=} W$. Non-equality results in the termination of the session, or else HA calculates $SK = h(g^a \bmod p) \oplus R_{MU} \oplus R_{FA}$ and submits $M_3 = \{E_{K_{FH}}(SK)\}$ to FA .
- Step 4. When getting M_3 , FA retrieves SK by calculating $D_{K_{FH}}(E_{K_{FH}}(SK))$ and computes $X = h(SK || R_{FA})$. Then, FA forwards the message $M_4 = \{X, R_{FA}\}$ to MU .
- Step 5. Upon obtaining M_4 , MU calculates $SK^* = h(C_{MU} \oplus h(R)) \oplus R_{MU} \oplus R_{FA}$ (In Madhusudhan and Shashidhara's scheme [32], $SK^* = C_{MU} \oplus h(R) \oplus R_{MU} \oplus R_{FA}$. However, combined with the context, the formula should be $SK^* = h(C_{MU} \oplus h(R)) \oplus R_{MU} \oplus R_{FA}$, and we correct it here.) and $X^* = h(SK^* || R_{FA})$. Then, MU checks $X^* \stackrel{?}{=} X$. The equality means FA is authenticated by MU .

2.2. Cryptanalysis of Madhusudhan and Shashidhara's Scheme

Madhusudhan and Shashidhara's scheme [32] had some design flaws, and these flaws would lead to some security weaknesses. We analyze the design and security defects of Madhusudhan and Shashidhara's scheme [32] in this section.

2.2.1. Transmitting Messages in Plaintext

As we know, the XOR and concatenation operations cannot protect the security of the message, and the different parts of message through these two operations can readily be extracted by an adversary $\mathcal{A}$. In Madhusudhan and Shashidhara's scheme [32], the message $M_1 = \{U, V, W\}$ is directly transmitted after XOR and concatenation operations on the public channel, which means that these messages are transparent to $\mathcal{A}$. We assume that $\mathcal{A}$ gets the messages $M_1 = \{U, V, W\}$ and $M_2 = \{ID_{FA}, E_{K_{FH}}(M_1, R_{FA})\}$ of a session among MU , FA , and HA by eavesdropping attack, where $U = R \oplus R_{MU}$, $V = ((C_{MU} \oplus h(R)) || ID_{FA}) \oplus R_{MU}$ and $W = (U || K || (C_{MU} \oplus h(R)))$. Therefore, $\mathcal{A}$ can extract K and $C_{MU} \oplus h(R) = g^a \bmod p$ from W . Then, $\mathcal{A}$ can calculate $R_{MU} = V \oplus ((C_{MU} \oplus h(R)) || ID_{FA})$ and $R = U \oplus R_{MU}$. Therefore, $\mathcal{A}$ can extract some secret information from M_1 due to it being transmitted in plaintext.

2.2.2. User Impersonation Attack

If $\mathcal{A}$ forges a login request message as the name of MU and the forged message passes the authentication of HA , we say that $\mathcal{A}$ successfully implemented the user impersonation attack. In Madhusudhan and Shashidhara's scheme [32], as shown in Section 2.2.1, an adversary $\mathcal{A}$ can extract MU 's information $\{(C_{MU} \oplus h(R)) = g^a \bmod p, R, K\}$. As we can see from the review of Madhusudhan and Shashidhara's scheme [32], this information was sufficient for $\mathcal{A}$ to impersonate as MU sends the login information to FA . To forge a login request, $\mathcal{A}$ produces a nonce R'_{MU} and calculates

$U' = R \oplus R'_{MU}$, $V' = ((C_{MU} \oplus h(R)) \| ID_{FA}) \oplus R'_{MU}$ and $W' = (U' \| K \| (C_{MU} \oplus h(R)))$. The forged message $M'_1 = \{U', V', W'\}$ is a valid login request in the name of MU by using MU 's corresponding secret information $\{R, (C_{MU} \oplus h(R)), K\}$. The forged login request M'_1 can pass the authentication of FA and HA , and $\mathcal{A}$ can calculate SK by using $(C_{MU} \oplus h(R))$ in Step 5 of the authentication phase. Therefore, Madhusudhan and Shashidhara's scheme [32] is vulnerable to user impersonation attack.

2.2.3. Session Key Compromise Attack

In Madhusudhan and Shashidhara's scheme [32], MU and FA share a session key $SK = h(g^a \bmod p) \oplus R_{MU} \oplus R_{FA}$ after the authentication, where R_{MU} and R_{FA} are two nonces produced by MU and FA , respectively. There are two design drawbacks on the session key. On the one hand, the shared session key should be generally calculated by MU and FA independently. In Madhusudhan and Shashidhara's scheme [32], although both MU and FA contributed the session key, the value of SK was determined by HA , and FA cannot calculate the session key since it cannot get the contribution of MU . On the other hand, the session key was calculated only by XOR operations, and it is easily calculated by an adversary $\mathcal{A}$. Assume that $\mathcal{A}$ has obtained the messages $\{M_1, M_2, M_3, M_4\}$ by eavesdropping on the public channel. Next, $\mathcal{A}$ can extract $g^a \bmod p = C_{MU} \oplus h(R)$ and calculate R_{MU} as shown in Section 2.2.1, as well as can get R_{FA} from M_4 . Then, $\mathcal{A}$ can calculate the session key $SK = h(g^a \bmod p) \oplus R_{MU} \oplus R_{FA}$ of MU , FA , and HA . Therefore, in Madhusudhan and Shashidhara's scheme [32], the session key is easily compromised just by eavesdropping the exchanged messages among MU , FA , and HA on the public channel.

2.2.4. Mobile Device Breach Attack

In the security analysis section of Madhusudhan and Shashidhara's protocol [32], they claimed that their protocol could avoid off-line dictionary attack even if $\mathcal{A}$ obtained the information stored in MU 's mobile device. However, user's identity and password are often easy to remember and have low entropy in real-life scenarios, and the dictionaries of identity and password are very limited. Therefore, Wang et al. [37,38] pointed out that the data pair (ID, PW) can be guessed within polynomial time. We suppose that MU 's mobile device information $\{K_{MU}, R, C_{MU}, K, h(\cdot)\}$ has been obtained by $\mathcal{A}$, then the following offline steps can be used by $\mathcal{A}$ to guess MU 's identity and password:

- Step 1. $\mathcal{A}$ chooses a pair (ID_{MU}^*, PW_{MU}^*) from the identity dictionary and password dictionary.
- Step 2. $\mathcal{A}$ calculates $K_{MU}^* = h(ID_{MU}^* \| PW_{MU}^* \| R)$.
- Step 3. $\mathcal{A}$ tests the correctness of (ID_{MU}^*, PW_{MU}^*) by checking $K_{MU}^* \stackrel{?}{=} K_{MU}$.
- Step 4. $\mathcal{A}$ repeats the above steps until the right pair (ID_{MU}^*, PW_{MU}^*) is found.

2.2.5. Lack of Bidirectional Authentication

In the digital environment, the authentication between two parties often depends on whether the information provided by the two parties matches, i.e., one party verifies if the calculated value is equal to the value received from another party. Generally, a user authentication scheme should allow one participant to authenticate other participants in an explicit or implicit manner. In Madhusudhan and Shashidhara's scheme [32], both FA and MU have not authenticated the validity of HA . When receiving $M_3 = \{E_{k_{FH}}(SK)\}$ from HA in Step 3, FA just decrypts SK , but does not verify the validity of the message and HA . MU also does not authenticate HA . Therefore, Madhusudhan and Shashidhara's scheme [32] is not suitable for practical use due to it lacking the real bidirectional authentication.

2.2.6. Stolen Verifier Attack

In Madhusudhan and Shashidhara's scheme [32], HA stores $\{K, R\}$ as the verification information in the database for the each mobile user. As shown in Section 2.2.2, $\mathcal{A}$ can impersonate a mobile user if he/she knows MU 's information $\{(C_{MU} \oplus h(R)), R, K\}$. Actually, the value $(C_{MU} \oplus h(R))$ for each mobile user is equal, and it is equal to $g^a \bmod p$. As shown in Section 2.2.1, $\mathcal{A}$ can easily get $g^a \bmod p$

by eavesdropping the communication messages from the public channel. If the verifier was stolen by $\mathcal{A}$, he/she can impersonate any mobile user by using $\{(C_{MU} \oplus h(R)) = g^a \bmod p, R, K\}$. Therefore, Madhusudhan and Shashidhara's scheme [32] suffers from stolen verifier attack.

2.2.7. Other Weaknesses

In addition to the above security flaw, there are some other weaknesses in Madhusudhan and Shashidhara's scheme [32]. First, the counter mechanism is used in their scheme to resist the replay attack. Both MU and HA maintain a count value K synchronously, and HA judges the replay message by checking whether the received count value K equals the stored counter value. However, the authors did not explain how the two parties update the count value synchronously. Besides, the synchronization-based mechanism is vulnerable to asynchronous attack, and MU cannot access the server anymore if the updated K in MU does not match HA 's. Second, there are some errors in their protocol's password change phase. In the description of their protocol, the mobile device MU computes $K_{MU}^* = h(ID_{MU} \| PW_{MU})$ to check if the identity and password are correct, and $K_{MU}^{NEW} = h(ID_{MU} \| PW_{MU}^{NEW})$ to update the password. However, the correct formulas should be $K_{MU}^* = h(ID_{MU} \| PW_{MU} \| K)$ and $K_{MU}^{NEW} = h(ID_{MU} \| PW_{MU}^{NEW} \| K)$.

3. Our Proposed Scheme

This section proposes a novel authentication protocol with strong security for GLOMONET based on the previous relevant work. In the protocol, the fuzzy verifier technique [39] is adopted to resolve the contradiction of local password change and stolen smart card attack. Besides, the Diffie–Hellman agreement method is used in the new protocol to secure the session key. Except the phases shown in Madhusudhan and Shashidhara's scheme [32], our new protocol contains a session key update phase, which makes the protocol more practical.

3.1. Initialization Phase

HA initializes the system by selecting some parameters. HA chooses a large prime number p and a generator g of Z_p^* , where the DLP on Z_p^* is intractable. Next, HA selects a hash function $h(\cdot) : (0,1)^* \rightarrow (0,1)^l$, which transforms the arbitrary binary into l bit (such as 128 bits) data. Furthermore, HA chooses an integer $2^6 \leq n_0 \leq 2^8$, which is used for the fuzzy verifier. Besides, HA generates the secret and public key pair (x, X) , where $x \in Z_p^*$ and $X = g^x \bmod p$. Finally, HA chooses a symmetric encryption algorithm E and the corresponding decryption algorithm D .

In addition to the generation of the above parameters, HA and FA have their identities ID_{HA} and ID_{FA} , respectively, and they share a secret key K_{HF} by using the key exchange protocol.

3.2. Registration Phase

In this phase, MU interacts with the HA as follows to register as a legitimate user.

Step R1. $MU \Rightarrow HA : \{ID_{MU}, HPW_{MU}\}$:

MU first chooses ID_{MU} , PW_{MU} , and a nonce b . Then, MU calculates $HPW_{MU} = h(PW_{MU} \| b)$ and transmits the registration request $\{ID_{MU}, HPW_{MU}\}$ to HA with a secure method.

Step R2. $HA \Rightarrow MU : \text{A Smart Card } SC$:

HA checks if ID_{MU} exists in the system. If so, MU is asked to send a new one. Otherwise, HA computes $A_{MU} = h((h(ID_{MU}) \oplus HPW_{MU}) \bmod n)$, $K_{MH} = h(ID_{MU} \| x)$, and $C_{MU} = HPW_{MU} \oplus K_{MH}$. Then, HA stores the parameters $\{A_{MU}, C_{MU}, ID_{HA}, p, g, X, E, D, h(\cdot)\}$ in a smart card (SC) and forwards it to MU over a secure manner.

Step R3. When obtaining the SC, MU activates it and stores b into it.

3.3. Login and Authentication Phase

Whenever *MU* roams to *FA*, the following steps allow *FA* and *MU* to achieve bidirectional authentication and share a session key with the assistance of *HA*. These procedures can also be found in Figure 1.

Step V1. $MU \rightarrow FA: M_1 = \{ID_{HA}, CID_{MU}, R_{MU}, C_3\}$.

MU inserts the SC into the mobile terminal and types in ID_{MU} and PW_{MU} . *SC* computes $A'_{MU} = h((h(ID_{MU}) \oplus h(PW_{MU}||b)) \bmod n)$ and verifies $A'_{MU} \stackrel{?}{=} A_{MU}$. Unequal means that *MU* entered incorrect ID_{MU} or PW_{MU} , and the session is terminated by *SC*. The card will be locked if there are three consecutive failures on the password. Otherwise, *SC* produces a nonce $r_{MU} \in Z_p^*$ and calculates $R_{MU} = g^{r_{MU}} \bmod p$, $C_1 = X^{r_{MU}} \bmod p$, $CID_{MU} = ID_{MU} \oplus h(R_{MU}||C_1)$, $K'_{MH} = C_{MU} \oplus h(PW_{MU}||b)$, $C_2 = h(ID_{HA}||ID_{FA}||C_1)$, and $C_3 = E_{K'_{MH}}(C_2||ID_{FA})$. Then, *MU* submits the login request $M_1 = \{ID_{HA}, CID_{MU}, R_{MU}, C_3\}$ to *FA*.

Step V2. $FA \rightarrow HA: M_2 = \{CID_{MU}, R_{MU}, C_3, R_{FA}, C_4\}$.

When receiving M_1 from a roaming user, *FA* produces a nonce $r_{FA} \in Z_p^*$ and computes $R_{FA} = g^{r_{FA}} \bmod p$ and $C_4 = h(ID_{FA}||K_{HF}||R_{MU}||C_3||R_{FA})$. Then, *FA* forwards the message $M_2 = \{CID_{MU}, R_{MU}, C_3, R_{FA}, C_4\}$ to *HA*.

Step V3. $HA \rightarrow FA: M_3 = \{C_5, C_7\}$.

Upon obtaining the message from *FA*, *HA* calculates $C'_1 = R_{MU}^x \bmod p$, $ID'_{MU} = CID_{MU} \oplus h(R_{MU}||C'_1)$, $K''_{MH} = h(ID'_{MU}||x)$, and $D_{K''_{MH}}(C_3) = (C'_2||ID'_{FA})$. Then, *HA* calculates $C'_2 = h(ID_{HA}||ID'_{FA}||C'_1)$ and checks $C'_2 \stackrel{?}{=} C_2$. Unequal will lead to the rejection of the session. Otherwise, *MU* is authenticated by *HA*. Next, *FA* retrieves the K_{HF} according to ID'_{FA} . Then, *HA* calculates $C'_4 = h(ID'_{FA}||K_{HF}||R_{MU}||C_3||R_{FA})$ and authenticates *HA* by checking $C'_4 \stackrel{?}{=} C_4$. If the verification passes, *HA* calculates $C_5 = h(ID_{HA}||ID'_{FA}||K_{HF}||R_{MU}||R_{FA})$, $C_6 = h(ID'_{MU}||ID_{FA}||C_1||R_{FA})$, $C_7 = E_{K''_{MH}}(R_{FA}||C_6)$, and forwards $M_3 = \{C_5, C_7\}$ to *FA*.

Step V4. $FA \rightarrow MU: M_4 = \{R_{FA}, C_7, C_9\}$.

When receiving the message from *HA*, *FA* calculates $C'_5 = h(ID_{HA}||ID_{FA}||K_{HF}||R_{MU}||R_{FA})$, and the validity of *HA* is verified if $C'_5 = C_5$. Then, *FA* calculates $C_8 = R_{MU}^{r_{FA}} \bmod p$, the session key $SK = h(R_{MU}||R_{FA}||C_8)$, and $C_9 = h(SK||C_7)$. Then, *FA* submits the message $M_4 = \{R_{FA}, C_7, C_9\}$ to *MU*.

Step V5. When receiving M_4 from *FA*, *MU* calculates $D_{K'_{MH}}(C_7) = (R'_{FA}||C'_6)$ and checks if R'_{FA} is equal to the received R_{FA} . Then, *MU* calculates $C'_6 = h(ID_{MU}||ID_{FA}||C_1||R_{FA})$, and the validity of *HA* is verified if $C'_6 = C_6$. Next, *MU* calculates $C'_8 = R_{FA}^{r_{MU}} \bmod p$, $SK' = h(R_{MU}||R_{FA}||C'_8)$, $C'_9 = h(SK'||C_7)$, and the validity of *FA* is verified if $C'_9 = C_9$. Finally, as a shared session key between *MU* and *FA*, *SK* is used by them for further security communication.

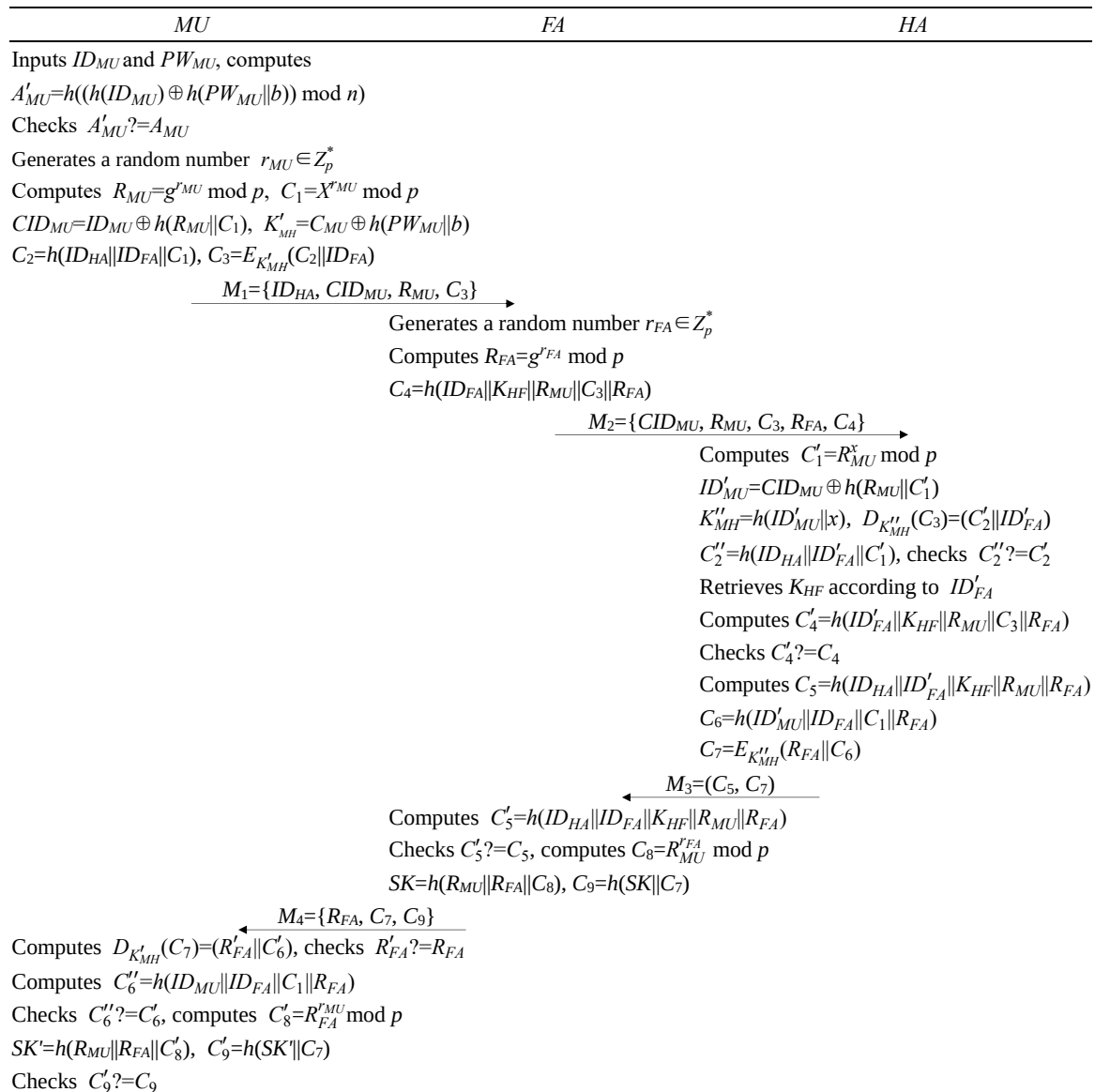


Figure 1. Login and verification phases.

3.4. Session Key Update Phase

Since the session key is only valid for a certain period, MU should update the session key with FA if he/she wants continuous access to the roaming network. The following procedures allow him/her to update a new session key, and it also can be found in Figure 2.

- Step S1. MU produces a new nonce $r'_{MU} \in Z_p^*$ and calculates $R'_{MU} = g^{r'_{MU}} \bmod p$, $C_{MU} = E_{SK}(R'_{MU})$. Then, MU submits $\{C_{MU}\}$ to FA for session key update.
- Step S2. FA retrieves R'_{MU} by decrypting C_{MU} using SK. Next, FA produces a new nonce $r'_{FA} \in Z_p^*$ and calculates $R'_{FA} = g^{r'_{FA}} \bmod p$, $C_{FA} = E_{SK}(R'_{FA})$, $SK_{new} = h(R'_{MU}||R'_{FA}||((R'_{MU})^{r'_{FA}} \bmod p))$, and $V_{FA} = h(SK_{new}||SK)$. Then, FA sends the message $\{C_{FA}, V_{FA}\}$ to MU.
- Step S3. When receiving the message, MU first retrieves R'_{FA} by decrypting C_{FA} using SK. Then, MU calculates $SK'_{new} = h(R'_{MU}||R'_{FA}||((R'_{FA})^{r'_{MU}} \bmod p))$, $V'_{FA} = h(SK'_{new}||SK)$ and checks $V'_{FA} \stackrel{?}{=} V_{FA}$. Unequal means MU and FA fail to agree on a new session key. Otherwise, they share a new session key SK_{new} .

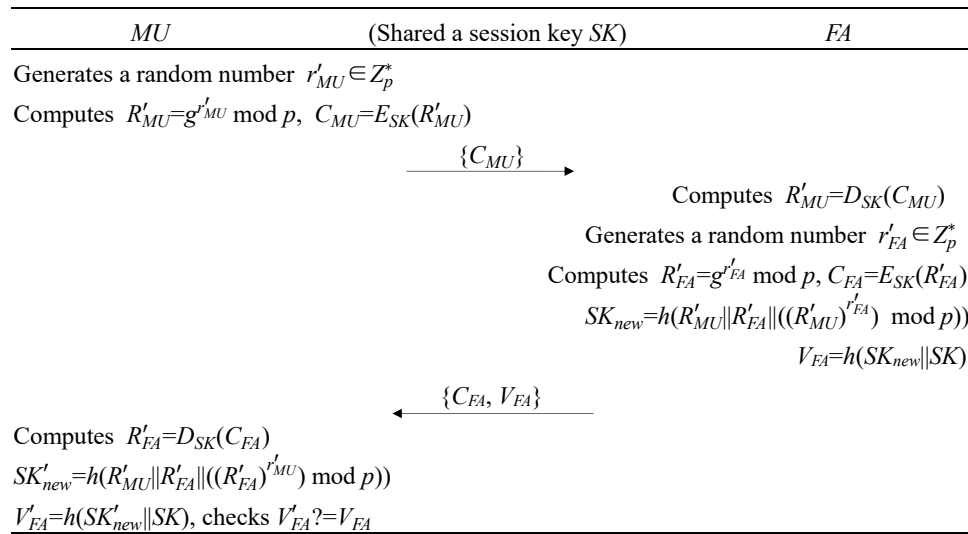


Figure 2. Session key update phase.

3.5. Password Change Phase

The following steps allow MU to change the password without the assistance of the home agent HA.

- Step P1. MU inserts the SC into the mobile terminal and types ID_{MU} and PW_{MU} .
- Step P2. SC computes $A'_{MU} = h((h(ID_{MU}) \oplus h(PW_{MU} || b)) \bmod n)$ and verifies $A'_{MU} \stackrel{?}{=} A_{MU}$. If they are not equal, which means that MU entered incorrect ID_{MU} or PW_{MU} , the request is rejected by SC. On the contrary, MU is asked to type a new password.
- Step P3. MU inputs a new password PW_{MU}^* . SC calculates $HPW_{MU}^* = h(PW_{MU}^* || b)$, $A_{MU}^* = h((h(ID_{MU}) \oplus HPW_{MU}^*) \bmod n)$, $C_{MU}^* = C_{MU} \oplus HPW_{MU}^* \oplus h(PW_{MU} || b)$. Finally, SC replaces A_{MU} and C_{MU} with A_{MU}^* and C_{MU}^* , respectively.

4. Formal Proof

4.1. Basic Knowledge

G is a multiplicative group with a large prime order p , and g is its generator.

- DLP (Discrete Logarithm Problem): Given $g^a \in G$ and $a \in Z_p^*$, it is hard to compute a .
- CDHP (Computation Diffie–Hellman Problem): If $a, b \in Z_p^*$ and $g^a, g^b \in G$ are known, it is hard to compute g^{ab} , and $Adv_{\mathcal{A}}^{CDH}$ denotes the probability for $\mathcal{A}$ to break the CDHP.

4.2. Basic Knowledge for the Proof

The formal proof given in here is based on [40–42].

To make it easy for $\mathcal{A}$ to perform attacks, we consider only three entities: one home agent HA, one foreign agent FA, and one mobile user MU. They make sessions in scheme $\mathcal{S}$. $\mathcal{I}$ is for any entity that is not required to be differentiated. The entities have many instances, and each of them is numbered, as an oracle, e.g., MU^i is the i th instance of MU. Similarly, we can understand T^t , FA^j and HA^k . *accept*, *reject*, and $\perp$ are three states for the result of the instance. *accept* occurs when the oracle receives the correct message; *reject* occurs when an incorrect message comes; and $\perp$ occurs when no answer appears in the oracle. Each instance of MU or FA has a session number, like SN_{MU^i} or SN_{FA^j} , and a partner PN_{MU^i} or PN_{FA^j} . If $SN_{MU^i} = SN_{FA^j}$ and they build the same session key sk , or $sk_{MU^i} = sk_{FA^j}$, we say that MU^i and FA^j are partnering, and $PN_{MU^i} = FA^j$ and $PN_{FA^j} = MU^i$. g, p, ID_{MU}, ID_{FA} , and ID_{HA} are public elements. The secret parameters, such as PW_i , K_{FH} , and x , are owned by the entities

according to the scheme $\mathcal{S}$. All passwords can be retrieved in a finite P set, which has $|P|$ elements in total.

$\mathcal{A}$ can make the following queries on a simulator $\mathbb{S}$ to crack the login and authentication process, especially the session key:

- $Send(\mathcal{I}, \mathcal{I}_r^t, m)$: $\mathcal{I}$ sends message m to $\mathcal{I}_r^t$. If m is correct, the normal operations in $\mathcal{S}$ will be done on $\mathcal{I}_r^t$; or the query will be stopped.
- $Execute(MU^i, FA^j, HA^k)$: All messages in the authentication phase will be eavesdropped by $\mathcal{A}$.
- $Reveal(\mathcal{I}^t)$: If MU^i or FA^j generates a session key, the key will be returned to $\mathcal{A}$.
- $Corrupt(MU^i, SC)$: $\mathcal{A}$ can get all information stored in SC of MU .
- $Corrupt(\mathcal{I}^t)$: $\mathcal{A}$ gets all long-term secret data of $\mathcal{I}$, and this is for strong forward security [41].
- $Test(\mathcal{I}^t)$: $\mathcal{I}$ is for MU or FA . Finally, $\mathcal{A}$ has to choose a session to make a challenge. If $\mathcal{I}$ does not reach *accept* or *sfs – security*, which will be explained later, the result is $\perp$. Else, a bit μ is chosen. If $\mu = 1$, the session key is returned. Otherwise, a random string $\{0, 1\}^l$ is returned.

Some demonstrations must be listed below to complement the above content.

- *SFS – fresh*: As shown in [41], it is Strong Forward Security-fresh for MU and FA . $\mathcal{I}^t$ is *SFS – fresh* if any of the following conditions do not appear:
 - $Reveal(\mathcal{I}^t)$ or $Reveal(PN_{\mathcal{I}^t})$ is asked.
 - $Corrupt(\mathcal{I}^t)$ or $Corrupt(PN_{\mathcal{I}^t})$ is queried before $Test(\mathcal{I}^t)$.
- *SFS – secure*: The advantage for $\mathcal{A}$ breaking $\mathcal{S}$ is the probability of correctly guessing μ generated in $Test(\mathcal{I}^t)$ over $\frac{1}{2}$, or $Adv_S^{SFS}(\mathcal{A}) = 2|Pr[\mu = \mu'] - \frac{1}{2}|$, where μ' is guessed by $\mathcal{A}$. If q_s is the number of *Send* queries, $Adv_S^{SFS}(\mathcal{A})$ is negligibly bigger than $\frac{O(q_s)}{|P|}$ and $\mathcal{S}$ is *SFS – secure*.

4.3. Process of the Proof

Theorem 1. A multiplicative group G with a large order p , a finite password set with size P , and the security length l are basic parameters for the proof. The adversary $\mathcal{A}$ has chances for q_s *Send* queries, q_e *Execute* queries, and q_h *hash* queries to crack $\mathcal{S}$, with the upper bound polynomial time t . The advantage of breaking the *SFS – secure* scheme $\mathcal{S}$ is:

$$Adv_S^{SFS}(\mathcal{A}) \leq \frac{(q_s + q_e)^2}{p - 1} + \frac{q_h^2 + 6q_s + 12q_h}{2^l} + \frac{2q_s}{|P|} + 4q_h(1 + (q_s + q_e)^2)Adv_A^{CDH}(t')$$

T_m represents the time of a multiplication in G and $t' = t + (6q_e + 2q_s)T_m$.

Proof. The proof contains a sequence of games from G_0 to G_5 . $Succ_i$ denotes $\mathcal{A}$'s successful probability to guess μ correctly. Since there is only one MU , no guessing is required for the identity.

- Game G_0 : The random oracles are added in the real scheme. If more queries are used by $\mathcal{A}$ or the game is finished, but without answer, a new bit μ is produced. We know that $Adv_S^{SFS}(\mathcal{A}) = 2|Pr[Succ_0] - \frac{1}{2}|$.
- Game G_1 : All queries in Section 4.2 are brought in. However, there are five *Send* queries: $Send(start, MU^i, \phi)$, $Send(MU^i, FA^j, M_1)$, $Send(FA^j, HA^k, M_2)$, $Send(HA^k, FA^j, M_3)$, and $Send(FA^j, MU^i, M_4)$. The operations for five *Send* queries correspond to the five steps in Section 3.3. Furthermore, three lists are required: $\mathcal{L}_h$ is to store the results of hash queries from the simulator $\mathbb{S}$; $\mathcal{L}_A$ is to store the hash queries from $\mathcal{A}$; $\mathcal{L}_S$ is to store the transcripts of the sessions. When a string str is asked for the hash result, the simulator $\mathbb{S}$ returns the result r if the tuple (str, r) can be found in the list. However, if there is no such result, the simulator $\mathbb{S}$ picks up a random string $r \in \{0, 1\}^l$ as the result and stores the tuple (str, r) in the corresponding list. $\mathcal{A}$ cannot know the difference between G_1 and G_0 , and $Pr[Succ_1] = Pr[Succ_0]$.

- Game G_2 : The collisions between different sessions are discussed in this game. There are two cases below:

1. The probability for collision of random numbers r_{MU} and r_{FA} is at most $\frac{(q_s+q_e)^2}{2(p-1)}$.
2. The probability for collision of hash results is $\frac{q_h^2}{2^{l+1}}$.

Therefore, we know $|Pr[Succ_2] - Pr[Succ_1]| \leq \frac{(q_s+q_e)^2}{2(p-1)} + \frac{q_h^2}{2^{l+1}}$.

- Game G_3 : We consider the case that $\mathcal{A}$ can impersonate the entities via forging hash results without hash query.

- $Send(MU^i, FA^j, M_1)$: $\mathbb{S}$ checks if $M_1 \in \mathcal{L}_S$, and $(R_{MU}||*,*)$, $(ID_{HA}||ID_{FA}||*,*) \in \mathcal{L}_A$. However, $(PW_{MU}||b,*)$ cannot be checked. The probability is $\frac{2q_h}{2^l}$ for this case.
- $Send(FA^j, HA^k, M_2)$: $\mathbb{S}$ checks if $M_1, M_2 \in \mathcal{L}_S$, and $(R_{MU}||*,*)$, $(ID_{HA}||ID_{FA}||*,*)$, $(ID_{FA}||*||R_{MU}||C_3||*, C_4) \in \mathcal{L}_A$. Furthermore, $(PW_{MU}||b,*)$ cannot be checked. The probability is $\frac{q_h}{2^l}$ for the first and second hash result and $\frac{q_s}{2^l}$ for the last.
- $Send(HA^k, FA^j, M_3)$: $\mathbb{S}$ checks if $M_1, M_2, M_3 \in \mathcal{L}_S$, and $(ID_{HA}||ID_{FA}||*||R_{MU}||R_{FA}, C_5)$, $(ID_{MU}||ID_{FA}||*||R_{FA}, *) \in \mathcal{L}_A$. The probability for the first result is $\frac{q_s}{2^l}$ and for the second is $\frac{q_h}{2^l}$.
- $Send(FA^j, MU^i, M_4)$: $\mathbb{S}$ checks if $M_1, M_2, M_3, M_4 \in \mathcal{L}_S$, and $(ID_{MU}||ID_{FA}||*||R_{FA}, *)$, $(*||C_7, C_9) \in \mathcal{L}_A$. The probability for the first result is $\frac{q_h}{2^l}$ and for the second is $\frac{q_s}{2^l}$.

Therefore, G_3 is the same as G_2 , and the probability $|Pr[Succ_3] - Pr[Succ_2]| \leq \frac{3q_s+6q_h}{2^l}$.

- Game G_4 : We inject the CDH problem in this game, with random oracles added again. Based on [41,42], $Corrupt(MU^i, SC)$ must be asked first to crack the scheme. Two cases can be listed:

1. With the most number of chance q_s , $\mathcal{A}$ can select one password from P to start a session. Such an active attack has the success probability $\frac{q_s}{|P|}$.
2. For passive attacks, two subcases are demonstrated below:
 - (a) *Execute* queries are used. Finally, $(g^{r_{MU}}||g^{r_{FA}}||g^{r_{MU}r_{FA}}, sk)$ should be found in $\mathcal{L}_A$, and the probability is $\frac{1}{q_h}$. Therefore, the probability for this subcase is $q_h Adv_A^{CDH}(t + 6q_e T_m)$.
 - (b) *Execute* can be replaced with the combination of all *Send* queries. Similar to the last subcase, the probability is $q_h Adv_A^{CDH}(t + 2q_s T_m)$.

We make $t' = t + (6q_e + 2q_s)T_m$ and get $|Pr[Succ_4] - Pr[Succ_3]| \leq \frac{q_s}{|P|} + 2q_h Adv_A^{CDH}(t')$.

- Game G_5 : This game is for strong forward security. With the concept of *SFS – secure*, *Corrupt* should be asked after *Test*, so we can only consider old messages here. Like G_4 , the probability for $g^{r_{MU}}$ and $g^{r_{FA}}$ occurring in the same session is $\frac{1}{(q_s+q_e)^2}$, and $|Pr[Succ_5] - Pr[Succ_4]| \leq 2q_h(q_s + q_e)^2 Adv_A^{CDH}(t')$. Till now, $Pr[Succ_5] = \frac{1}{2}$, and there is no advantage for $\mathcal{A}$.

Via the above games, Theorem 1 can be deduced. $\square$

5. Formal Verification by ProVerif

In this section, the emulator tool ProVerif is utilized by us to test the security of the proposed protocol, which is a new and efficient tool to detect the confidentiality of an information security protocol. The tool supports unlimited rounds of protocol execution.

The normalized mode description about our protocol is presented as follows. There are three entities, which are MU , FA , and HA , in the protocol. Firstly, MU sends the registration request to HA . Secondly, HA transmits the information to MU in a secure manner. Thirdly, MU sends the login

request to FA. Fourthly, FA forwards the message to HA. Then, HA tightly transmits the response to FA. Lastly, FA returns the message to HA.

- $MU \rightarrow HA : \{ID_{MU}, HPW_{MU}\}$
- $HA \rightarrow MU : \{A_{MU}, C_{MU}, ID_{HA}, p, g, X, E, D, h(\cdot)\}$
- $MU \rightarrow FA: M_1 = \{ID_{HA}, CID_{MU}, R_{MU}, C_3\}$.
- $FA \rightarrow HA: M_2 = \{CID_{MU}, R_{MU}, C_3, R_{FA}, C_4\}$.
- $HA \rightarrow FA: M_3 = \{C_5, C_7\}$.
- $FA \rightarrow MU: M_4 = \{R_{FA}, C_7, C_9\}$.

The detail of the code is shown in the following. The predefinition code in Figure 3 consists by three parts, and we firstly declare some statements (parameters, space and functions) and channels (secure and unsecure channel). Then, we define some equations (hash function, XOR operation, and the inverse operation of $\parallel$). We also make some queries to satisfy the security requirements. The processes of MU, FA, and HA were operated simultaneously, and the corresponding codes are shown in Figures 4–6, respectively. In summary, As shown in Figure 7, we state the protocol by using events BeginUser(muID) and EndUser(muID), and the verification results are “RESULT not attacker(SK[]) is true.”, “RESULT not attacker(KHF[]) is true.”, “RESULT not attacker(KMH'[]) is true.”, and “RESULT inj-event(EndUser(id)) ==> inj-event(BeginUser(id)) is true.”. Thus, the session key SK, the shared key K_{HF} , and the shared key K_{MH} could withstand the attacks, and the proposed protocol could pass the verification of ProVerif.

```

free sch: channel [private].(*—securty channel—*)
free ch: channel. (*—unsecurty channel—*)
const muID: bitstring. (*———muid*)
const pw: bitstring[private].
const FAID:bitstring.
const HAID:bitstring.
const p:bitstring.
const g:bitstring.
free nouceb:bitstring[private].
free x:bitstring[private].
free pubX:bitstring.
free key:bitstring[private].
free rFA:bitstring[private].
free Rmu:bitstring[private].
free KHF:bitstring[private].
free CIDmu:bitstring.
free SK:bitstring[private].
free KMH':bitstring[private].
fun senc(bitstring, bitstring): bitstring.
fun H(bitstring):bitstring.
fun xor(bitstring, bitstring): bitstring.
fun Concat(bitstring, bitstring): bitstring.
fun pow(bitstring,bitstring):bitstring.
fun sub(bitstring,bitstring):bitstring.

```

```

reduc forall m:bitstring, n:bitstring; sdec(senc(m,n),n)=m.
equation forall m:bitstring,n:bitstring; xor(xor(m,n),n)=m.
equation forall m:bitstring,n:bitstring; sub(Concat(m,n),n)=m.

```

```

event BeginUser(bitstring).
event EndUser(bitstring).
(*———query———*) query attacker(SK).
query attacker(KHF).
query attacker(KMH').
query id:bitstring; inj-event(EndUser(id))==>inj-event(BeginUser(id)).

```

Figure 3. Predefinition code.

```

(*-----MU process-----*)
let mu=
new N: bitstring;
new Breg: bitstring;
let muPW = H(Concat(pw, nouceb)) in
out(sch, (muID,muPW));
!
(
event BeginUser(muID);
in(sch,(AMU:bitstring,CMU:bitstring,HAID:bitstring,p:bitstring,g:bitstring,pubX:bitstring,key:bitstring));
let AMU'=H(xor(H(muID),H(Concat(pw,nouceb)))) in
if AMU'=AMU then
new Rmu:bitstring ;
let RMU=pow(g,Rmu) in
let pubX=pow(g,x) in
let C1=pow(pubX,Rmu) in
let CIDmu=xor(muID,H(Concat(Rmu,C1))) in
let Kmh'=xor(CMU,H(Concat(pw,nouceb))) in
let C2=H(Concat(HAID,Concat(FAID,C1))) in
let C3=senc(C2,key) in
out(ch,(HAID,CIDmu,RMU,C3));
event EndUser(muID);
in(ch,(RFA:bitstring,C7:bitstring,C9:bitstring));
let C6'=H(Concat(Concat(xor(CIDmu,H(Concat(RMU,C1))),FAID),Concat(C1,RFA))) in
let Rfa'=sub(sdec(C7,key),C6') in
if Rfa'=RFA then
let C6''=H(Concat(Concat(muID,FAID),Concat(C1,RFA))) in
if C6''=C6' then
let C8'=pow(RFA,Rmu) in
let SK'=H(Concat(Concat(Rmu,RFA),C8')) in
let C9'=H(Concat(SK',C7)) in
if C9=C9' then
0
).

```

Figure 4. Verification code for MU.

```

(*-----FA process-----*)
let FA=
out(sch,(KHF));
!
(
in(ch,(HAID:bitstring,CIDmu:bitstring,RMU:bitstring,C3:bitstring));
new rFA:bitstring;
let RFA=pow(g,rFA) in
let C4=H(Concat(Concat(FAID,KHF),Concat(RMU,Concat(C3,RFA)))) in
out(ch,(CIDmu,RMU,C3,RFA,C4,FAID));
in(ch,(C5:bitstring,C7:bitstring));
let C5'=H(Concat(Concat(HAID,FAID),Concat(KHF,Concat(RMU,Concat(RFA,HAID))))) in
if C5=C5' then
let C8=pow(RMU,rFA) in
let SK=H(Concat(RMU,Concat(RFA,C8))) in
let C9=H(Concat(SK,C7)) in
out(ch,(RFA,C7,C9));
0
).

```

Figure 5. Verification code for FA.

```

(*-----HA process-----*)
let HA=
!
(
in(sch, (muID:bitstring,muPW:bitstring));
let AMU=H(xor(H(muID),muPW)) in
let key=H(Concat(muID,x)) in
let CMU=xor(muPW,key) in
out(sch,(AMU,CMU,HAID,p,g,pubX,key));
in(ch,(CIDmu:bitstring,RMU:bitstring,C3:bitstring,RFA:bitstring,C4:bitstring,FAID:bitstring));
let C1'=pow(RMU,x) in
let MUID'=xor(CIDmu,H(Concat(RMU,C1'))) in
let KMH'=H(Concat(MUID',x)) in
let DKMH'=sdec(C3,KMH') in
let C2'=sub(DKMH',FAID) in
let C2'=H(Concat(HAID,Concat(FAID,C1'))) in
if C2'=C2' then
in(sch,(KHF:bitstring));
let C4'=H(Concat(Concat(FAID,KHF),Concat(RMU,Concat(C3,RFA)))) in
if C4'=C4' then
let C5=H(Concat(Concat(Concat(HAID,FAID),KHF),Concat(RMU,RFA))) in
let C6=H(Concat(Concat(muID,FAID),Concat(C1',RFA))) in
let C7=senc(Concat(RFA,C6),KMH') in
out(ch,(C5,C7));
0
).
process ( mu|FA|HA)

```

Figure 6. Verification code for HA.

```

- Query not attacker(SK[])
Completing...
200 rules inserted. The rule base contains 171 rules. 8 rules in the queue.
Starting query not attacker(SK[])
RESULT not attacker(SK[]) is true.
- Query not attacker(KHF[])
Completing...
200 rules inserted. The rule base contains 171 rules. 8 rules in the queue.
Starting query not attacker(KHF[])
RESULT not attacker(KHF[]) is true.
- Query not attacker(KMH'[])
Completing...
200 rules inserted. The rule base contains 171 rules. 8 rules in the queue.
Starting query not attacker(KMH'[])
RESULT not attacker(KMH'[]) is true.
- Query inj-event(EndUser(id)) ==> inj-event(BeginUser(id))
Completing...
200 rules inserted. The rule base contains 178 rules. 8 rules in the queue.
RESULT inj-event(EndUser(id)) ==> inj-event(BeginUser(id)) is true.

```

Figure 7. Verification result.

6. Security Analysis and Comparisons

To show the advantages of our protocol, security features of our protocol are discussed in this part. Besides, the comparisons of our protocol and other related protocols are also given in this section.

6.1. Resist Smart Card Loss (Off-Line Password Guessing) Attack

To achieve the feature of freely local password update for users, the smart card must store verification information of user's identity and password. However, this verification information can be used to guess a password if the smart card is lost. Our scheme uses the fuzzy verifier technique [39] to resist smart card loss attack while realizing local password change. If MU 's SC is lost, $\mathcal{A}$ can retrieve the information in $SC \{A_{MU}, C_{MU}, ID_{HA}, p, g, X, E, D, b, h(\cdot)\}$. However, $\mathcal{A}$ cannot guess ID_{MU} and PW_{MU} from C_{MU} without knowing HA 's secret key x . $\mathcal{A}$ may also want to guess ID and PW from $A_{MU} = h((h(ID_{MU}) \oplus h(PW_{MU} \| b)) \bmod n)$. However, for 10^6 sized identity/password

dictionaries, and $n_0 \approx 2^8$, there are more than 2^{30} pairs of (ID_{MU}, PW_{MU}) such that the value of $h((h(ID_{MU}) \oplus h(PW_{MU} \| b)) \bmod n)$ equals A_{MU} . Therefore, it is impossible for $\mathcal{A}$ to obtain the correct (ID_{MU}, PW_{MU}) . According to the above analysis, it can be seen that our protocol can resist smart card loss attack, and $\mathcal{A}$ cannot guess the user's correct (ID_{MU}, PW_{MU}) even if $\mathcal{A}$ obtained the smart card information.

6.2. User Anonymity

In our protocol, ID_{MU} is not included in any of communication messages M_1, M_2, M_3 , and M_4 . Among these messages transmitted on the public channel, just CID_{MU} is related to ID_{MU} . Only HA can reveal ID_{MU} from CID_{MU} by using the secret key x , and $\mathcal{A}$ cannot know ID_{MU} from the messages obtained on the public channel. Meanwhile, in each session, the messages M_1, M_2, M_3 , and M_4 exchanged among the three parties, all elements except ID_{HA} are dynamically changed with the random numbers. Therefore, $\mathcal{A}$ cannot track different sessions from the same mobile user. Therefore, our protocol can protect the user's anonymity.

6.3. Proper Bidirectional Authentication

Our protocol achieves the bidirectional authentication among MU, FA , and HA with the assistance of HA . The bidirectional authentication between MU and HA relies on the shared secret information $K_{MH} = h(ID_{MU} \| x)$, which can be retrieved by MU with the PW_{MU} from C_{MU} and also can be calculated by HA when he/she retrieves the correct ID_{MU} from CID_{MU} using x . With the shared K_{MH} , HA can authenticate MU by checking $C_2'' \stackrel{?}{=} C_2'$ in Step V3, and MU can authenticate HA by checking $C_6'' \stackrel{?}{=} C_6'$ in Step V5. Meanwhile, the bidirectional authentication between FA and HA can be achieved due to the pre-shared secret key K_{HF} . With this secret key, HA can authenticate FA by verifying $C_4' \stackrel{?}{=} C_4$ in Step V3, and FA can also verify if HA is valid by checking $C_5' \stackrel{?}{=} C_5$ in Step V5. For MU and FA , MU is a legal foreign user for FA if MU passes the verification of HA . On the contrary, FA is valid for MU if they generate the same SK and $C_9' = C_9$.

6.4. Resist Impersonation Attack

In our protocol, the secret keys x and K_{HF} are indispensable for HA to authenticate mobile users and FA . Therefore, any adversary cannot imitate HA . For the same reason, the adversary cannot impersonate FA . Besides, based on the description of the authentication phase, ID_{MU} and PW_{MU} are essential for MU to generate a valid login request message. However, as we can see from Section 6.1, an adversary $\mathcal{A}$ cannot guess MU 's ID_{MU} and PW_{MU} even if $\mathcal{A}$ has obtained the user's smart card. Therefore, our protocol can avoid user impersonation attack.

6.5. Session Key Security

After the bidirectional authentication, a session key $SK = h(g^{r_{MU}} \| g^{r_{FA}} \| g^{r_{MU} \cdot r_{FA}})$ (we omit the modular exponential operations here) is shared between MU and FA , which includes contributions from both MU and FA . SK is secure due to the intractability of the DLP; no adversary, even HA , can calculate the SK without knowing r_{MU} or r_{FA} . Besides, since SK is independent of the master secret key x , it is still secure even though x is compromised. Therefore, the forward secrecy feature is achieved. Furthermore, the session key depends on the nonce generated by MU and FA , which dynamically change with different sessions. Therefore, the compromise of one session key would not deduce the compromise of other session keys, and our protocol guarantees the feature of known-key secrecy.

6.6. Resist Insider Attack

In our protocol's registration phase, the mobile user MU submits $HPW_{MU} = h(PW_{MU} \| b)$ to HA for registration. Even if the privileged insider gets the registration information $\{ID_{MU}, HPW_{MU}\}$,

he/she still cannot reveal MU 's real identity ID_{MU} since it is hidden by the random number b . Therefore, our protocol resists insider attack.

6.7. Secure Session Key Update

Session key update is an important function for a mobile user if he/she wants to access the foreign network continuously. Our protocol allows MU and FA to generate securely a new session key based on their current session key. Since our key agreement mechanism is similar to the DHP, we have adjusted the mechanism to avoid the man-in-the-middle attack. MU and FA generate the new shares R_{MU} and R_{FA} and transmit the encrypted share $C_{MU} = E_{SK}(R'_{MU})$ and $C_{FA} = E_{SK}(R'_{FA})$ to each other. An adversary $\mathcal{A}$ cannot obtain their real shares without knowing the current session key SK , and our protocol allows the two parties to update their session key securely.

6.8. Comparison with Other Related Protocols

To demonstrate the robustness and advantages of our protocol, the features of our protocol and other relevant protocols [30,32] are compared in this section.

First, the function and security features of our protocol and the protocols in [30,32] are evaluated, and the result is shown in Table 2. As can be seen from the table, both protocols in [30,32] failed to provide forward secrecy and session key update and were vulnerable to stolen verifier attack. Except for their common defects, the protocol in [30] cannot realize the function of local password change for a mobile user, while the protocol in [32] was also vulnerable to mobile device breach attack and user impersonation attack and cannot provide bidirectional authentication and session key security. Unlike the previous work, our protocol resisted all known attacks and provided some ideal functions. The fuzzy verifier technique resolved the contradiction of local password change and smart card breach attack. Besides, the modular exponential operation based on DLP guaranteed the security of session key. Therefore, our protocol was better than the related protocols on security and functions.

Table 2. Comparison of security and function features.

	Protocol in [30]	Protocol in [32]	Our Protocol
Resist mobile device/smart card breach attack	✓	×	✓
Resist insider attack	✓	✓	✓
Resist user impersonation attack	✓	×	✓
Resist stolen verifier attack	×	×	✓
User anonymity	✓	✓	✓
Local password change	×	✓	✓
Proper bidirectional authentication	✓	×	✓
Session key security	✓	×	✓
Forward secrecy	×	×	✓
Session key update	×	×	✓

✓: satisfies the property ×: fails to satisfy the property.

Next, we evaluate the computation complexity and the storage cost of our protocol and the protocols in [30,32]. In order to facilitate the description, let T_h , T_{me} , and T_e represent the computation cost of a hash function, a modular exponential, and a symmetric encryption/decryption operation, respectively. To evaluate the computation cost, we used the computation costs for related cryptographic operations in [43] on platform Intel(R) T5870 2.00 GHz, which were $T_h = 2.580 \mu s$, $T_{me} = 10.257 ms$, and $T_e = 2.012 \mu s$, respectively. Table 3 shows the result of the performance comparison, from which we can see that our protocol just needed a few more modular exponential operations than the protocols in [30,32]. The total computational cost for a session of our protocol was about 61.6 ms, and it was also at a very efficient level. It was precisely because of these increased computational cost that our protocol surpassed other related work [30,32] in terms of security and function. Our protocol not only guaranteed the security of the session key, but also balanced the contradiction between local password

change and stolen smart card attack. Therefore, the increased computational cost was worthwhile, and it enhanced the security of the protocol. Furthermore, we compared the *MU* side's storage cost of three protocols and omitted the storage cost of *HA* and *FA* since the servers both had adequate storage space. Compared with other protocols [30,32], the storage requirements of *MU* in our protocol was somewhere in between, and it also can be seen in Table 3.

Table 3. Performance comparison.

	Protocol in [30]	Protocol in [32]	Our Protocol
<i>MU</i> 's computational cost	$9T_h + T_e + 2T_{me}$	$4T_h$	$8T_h + 2T_e + 3T_{me}$
<i>FA</i> 's computational cost	$3T_h$	$T_h + 2T_e$	$4T_h + 2T_{me}$
<i>HA</i> 's computational cost	$9T_h + T_e + T_{me}$	$2T_h + 2T_e + 2T_{me}$	$6T_h + 2T_e + T_{me}$
Total computational cost	$21T_h + 2T_e + 3T_{me} \approx 30.83$ ms	$7T_h + 4T_e + 2T_{me} \approx 20.54$ ms	$18T_h + 4T_e + 6T_{me} \approx 61.60$ ms
<i>MU</i> 's storage cost	4496 bits	2448 bits	3472 bits

Finally, we evaluated the communication complexity of our protocol and the protocols in [30, 32]. For comparison purposes, let L_{ID} , L_h , L_r , L_T , L_S , and L_p represent the bit lengths of identity, hash function, random number, timestamps, symmetric cryptography, and large prime number, respectively, and we assumed they were 80 bits, 160 bits, 160 bits, 80 bits, 128 bits, and 1024 bits, respectively. We counted the communication cost of one session for the three protocols, and the comparison result is shown in Table 4. From the table, we can see that our protocol had a lesser increase in communication cost to achieve higher security.

Table 4. Communication cost comparison (bits).

	Protocol in [30]	Protocol in [32]	Our Protocol
<i>MU</i> to <i>FA</i>	$L_p + 10L_S + TL + L_{ID} = 2624$ bits	$2L_{ID} + 4L_p + 2L_h = 4656$ bits	$L_{ID} + L_h + L_p + 2L_S = 1520$ bits
<i>FA</i> to <i>HA</i>	$2L_h + L_{ID} + L_r + 2L_T + L_p + 10L_S = 3024$ bits	$L_{ID} + 20L_S = 2640$ bits	$2L_h + 2L_p + 2L_S = 2624$ bits
<i>HA</i> to <i>FA</i>	$3L_h + L_T = 560$ bits	$2L_S = 256$ bits	$L_h + 10L_S = 1440$ bits
<i>FA</i> to <i>MU</i>	$L_h + L_r + L_T = 400$ bits	$2L_h = 320$ bits	$L_p + L_h + 10L_S = 2464$ bit
Total cost	6308 bits	7872 bits	8048 bits

7. Conclusions

This paper focused on the user authentication mechanism in GLOMONET. Firstly, some design and security weaknesses of the user authentication protocol in [32] were pointed out. Their protocol had a fatal flaw that some important information of the mobile user was transmitted via the public channel in plaintext, and it can be easily obtained by an adversary. Therefore, their protocol suffered from stolen verifier attack, user impersonate attack, session key compromise attack, and mobile device breach attack. Secondly, we proposed a novel user authentication protocol with strong security for GLOMONET. The use of the fuzzy verifier technique ensured that the proposed protocol resisted the smart card loss attack while achieving the feature of local password update. Besides, the proposal was based on the Diffie–Hellman Protocol (DHP) under the discrete logarithm, which not only ensured the security of session key, but also enabled our protocol to achieve the function of secure session key agreement. Compared with other relevant protocols, our protocol achieved a higher level of security with a small amount of increased computation and communication, and it was more suitable for the GLOMONET environment.

Author Contributions: Methodology, S.L., X.L., and F.W.; formal analysis, F.W.; writing, original draft preparation, S.L., X.L., and F.W.; writing, review and editing, J.L., J.W. and D.L.

Funding: This research was funded by the Scientific Research Fund of Hunan Provincial Education Department under Grant No. 18A178, the Natural Science Foundation of Hunan Province, China, under Grant No. 2018JJ3191, and the National Natural Science Foundation of China under Grant No. 61572013. D.L. appreciates the Fujian Education and Scientific Research Program for Young and Middleaged Teachers under Grant No JAT170808 and the University Distinguished Young Research Talent Training Program of Fujian Province (Year 2018).

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

References

1. Wang, J.; Gao, Y.; Yin, X.; Li, F.; Kim, H.J. An Enhanced PEGASIS Algorithm with Mobile Sink Support for Wireless Sensor Networks. *Wirel. Commun. Mob. Comput.* **2018**, 1–9. [\[CrossRef\]](#)
2. Nguyen, T.; Pan, J.; Dao, T. An Improved Flower Pollination Algorithm for Optimizing Layouts of Nodes in Wireless Sensor Network. *IEEE Access* **2019**, 7, 75985–75998. [\[CrossRef\]](#)
3. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Hye-Jin, K. Energy Efficient Routing Algorithm with Mobile Sink Support for Wireless Sensor Networks. *Sensors* **2019**, 19, 1494. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Wang, J.; Gao, Y.; Sangaiah, A.K.; Hye-Jin, K. An Improved Routing Schema with Special Clustering using PSO Algorithm for Heterogeneous Wireless Sensor Network. *Sensors* **2019**, 19, 671. [\[CrossRef\]](#) [\[PubMed\]](#)
5. He, S.; Xie, K.; Xie, K.; Xu, C.; Jin, W. Interference-aware Multi-source Transmission in Multi-radio and Multi-channel Wireless Network. *IEEE Syst. J.* **2019**, 13, 2507–2518. [\[CrossRef\]](#)
6. He, S.; Xie, K.; Chen, W.; Zhang, D.; Wen, J. Energy-aware Routing for SWIPT in Multi-hop Energy-constrained Wireless Network. *IEEE Access* **2018**, 6, 17996–18008. [\[CrossRef\]](#)
7. Li, X.; Peng, J.; Niu, J.; Wu, F.; Liao, J.; Choo, K.K.R. A robust and energy efficient authentication protocol for industrial internet of things. *IEEE Internet Things J.* **2018**, 5, 1606–1615. [\[CrossRef\]](#)
8. Li, X.; Liu, S.; Wu, F.; Kumari, S.; Rodrigues, J.J.P.C. Privacy Preserving Data Aggregation Scheme for Mobile Edge Computing Assisted IoT Applications. *IEEE Internet Things J.* **2019**, 6, 4755–4763. [\[CrossRef\]](#)
9. Li, W.; Chen, Z.; Gao, X.; Liu, W.; Wang, J. Multi-Model Framework for Indoor Localization under Mobile Edge Computing Environment. *IEEE Internet Things J.* **2019**, 6, 4844–4853. [\[CrossRef\]](#)
10. Yin, Y.; Xu, Y.; Xu, W.; Min, G.; Pei, Y. Collaborative Service Selection via Ensemble Learning in Mixed Mobile Network Environments. *Entropy* **2017**, 19, 358. [\[CrossRef\]](#)
11. Wang, J.; Gao, Y.; Liu, W.; Wenbing, W.; Se-Jung, L. An Asynchronous Clustering and Mobile Data Gathering Schema based on Timer Mechanism in Wireless Sensor Networks. *Comput. Mater. Contin.* **2019**, 58, 711–725. [\[CrossRef\]](#)
12. Wang, J.; Ju, C.; Gao, Y.; Sangaiah, A.K.; Jun Kim, G. A PSO based Energy Efficient Coverage Control Algorithm for Wireless Sensor Networks. *Comput. Mater. Contin.* **2018**, 56, 433–446.
13. Zhang, J.; Jin, X.; Sum, J.; Wang, J.; Li, K. Dual model learning combined with multiple feature selection for accurate visual tracking. *IEEE Access* **2019**, 7, 43956–43969. [\[CrossRef\]](#)
14. Zhang, J.; Lu, C.; Li, X.; Kim, H.J.; Wang, J. A full convolutional network based on DenseNet for remote sensing scene classification. *Math. Biosci. Eng.* **2019**, 16, 3345–3367. [\[CrossRef\]](#)
15. Liu, Y.N.; Wang, Y.P.; Wang, X.F.; Xia, Z.; Xu, J.F. Privacy-preserving raw data collection without a trusted authority for IoT. *Comput. Netw.* **2019**, 148, 340–348. [\[CrossRef\]](#)
16. Song, J.; Liu, Y.; Shao, J.; Tang, C. A Dynamic Membership Data Aggregation (DMDA) Protocol for Smart Grid. *IEEE Syst. J.* **2019**. [\[CrossRef\]](#)
17. Zhang, S.; Choo, K.K.R.; Liu, Q.; Wang, G. Enhancing privacy through uniform grid and caching in location-based services. *Future Gener. Comput. Syst.* **2018**, 86, 881–892. [\[CrossRef\]](#)
18. Zhang, S.; Mao, X.; Choo, K.K.R.; Peng, T.; Wang, G. A trajectory privacy-preserving scheme based on a dual-K mechanism for continuous location-based services. *Inf. Sci.* **2019**. [\[CrossRef\]](#)
19. Zhu, J.; Ma, J. A new authentication scheme with anonymity for wireless environments. *IEEE Trans. Consum. Electron.* **2004**, 50, 231–235.
20. Lee, C.C.; Hwang, M.S.; Liao, I.E. Security enhancement on a new authentication scheme with anonymity for wireless environments. *IEEE Trans. Ind. Electron.* **2006**, 53, 1683–1687. [\[CrossRef\]](#)
21. Wu, C.C.; Lee, W.B.; Tsaur, W.J. A secure authentication scheme with anonymity for wireless communications. *IEEE Commun. Lett.* **2008**, 12, 722–723.
22. Chang, C.C.; Lee, C.Y.; Chiu, Y.C. Enhanced authentication scheme with anonymity for roaming service in global mobility networks. *Comput. Commun.* **2009**, 32, 611–618. [\[CrossRef\]](#)
23. Xu, J.; Zhu, W.T.; Feng, D.G. An efficient mutual authentication and key agreement protocol preserving user anonymity in mobile networks. *Comput. Commun.* **2011**, 34, 319–325. [\[CrossRef\]](#)
24. He, D.; Ma, M.; Zhang, Y.; Chen, C.; Bu, J. A strong user authentication scheme with smart cards for wireless communications. *Comput. Commun.* **2011**, 34, 367–374. [\[CrossRef\]](#)

25. Li, C.T.; Lee, C.C. A novel user authentication and privacy preserving scheme with smart cards for wireless communications. *Math. Comput. Model.* **2012**, *55*, 35–44. [\[CrossRef\]](#)
26. Yoon, E.J.; Yoo, K.Y.; Ha, K.S. A user friendly authentication scheme with anonymity for wireless communications. *Comput. Electr. Eng.* **2011**, *37*, 356–364. [\[CrossRef\]](#)
27. Niu, J.; Li, X. A novel user authentication scheme with anonymity for wireless communications. *Secur. Commun. Netw.* **2014**, *7*, 1467–1476. [\[CrossRef\]](#)
28. Mun, H.; Han, K.; Lee, Y.S.; Yeun, C.Y.; Choi, H.H. Enhanced secure anonymous authentication scheme for roaming service in global mobility networks. *Math. Comput. Model.* **2012**, *55*, 214–222. [\[CrossRef\]](#)
29. Zhao, D.; Peng, H.; Li, L.; Yang, Y. A secure and effective anonymous authentication scheme for roaming service in global mobility networks. *Wirel. Pers. Commun.* **2014**, *78*, 247–269. [\[CrossRef\]](#)
30. Karuppiyah, M.; Saravanan, R. A secure authentication scheme with user anonymity for roaming service in global mobility networks. *Wirel. Pers. Commun.* **2015**, *84*, 2055–2078. [\[CrossRef\]](#)
31. Li, X.; Sangaiah, A.K.; Kumari, S.; Wu, F.; Shen, J.; Khan, M.K. An efficient authentication and key agreement scheme with user anonymity for roaming service in smart city. *Pers. Ubiquitous Comput.* **2017**, *21*, 791–805. [\[CrossRef\]](#)
32. Madhusudhan, R.; Shashidhara. A secure and lightweight authentication scheme for roaming service in global mobile networks. *J. Inf. Secur. Appl.* **2018**, *38*, 96–110. [\[CrossRef\]](#)
33. Banerjee, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Kumar, N.; Park, Y.; Tanwar, S. Design of an anonymity-preserving group formation based authentication protocol in global mobility networks. *IEEE Access* **2018**, *6*, 20673–20693. [\[CrossRef\]](#)
34. Xu, G.; Liu, J.; Lu, Y.; Zeng, X.; Zhang, Y.; Li, X. A novel efficient MAKa protocol with desynchronization for anonymous roaming service in Global Mobility Networks. *J. Netw. Comput. Appl.* **2018**, *107*, 83–92. [\[CrossRef\]](#)
35. Li, X.; Niu, J.; Kumari, S.; Wu, F.; Choo, K.K.R. A robust biometrics based three-factor authentication scheme for global mobility networks in smart city. *Future Gener. Comput. Syst.* **2018**, *83*, 607–618. [\[CrossRef\]](#)
36. Wei, F.; Vijayakumar, P.; Jiang, Q.; Zhang, R. A mobile intelligent terminal based anonymous authenticated key exchange protocol for roaming service in global mobility networks. *IEEE Trans. Sustain. Comput.* **2018**. [\[CrossRef\]](#)
37. Wang, D.; Wang, P. Understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks. *Ad Hoc Netw.* **2014**, *20*, 1–15. [\[CrossRef\]](#)
38. Wang, D.; Cheng, H.; He, D.; Wang, P. On the challenges in designing identity-based privacy-preserving authentication schemes for mobile devices. *IEEE Syst. J.* **2018**, *12*, 916–925. [\[CrossRef\]](#)
39. Wang, D.; He, D.; Wang, P.; Chu, C. Anonymous Two-Factor Authentication in Distributed Systems: Certain Goals Are Beyond Attainment. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 428–442. [\[CrossRef\]](#)
40. Bresson, E.; Chevassut, O.; Pointcheval, D. Security proofs for an efficient password-based key exchange. In Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003; ACM: New York, NY, USA, 2003; pp. 241–250.
41. Xu, L.; Wu, F. An improved and provable remote user authentication scheme based on elliptic curve cryptosystem with user anonymity. *Secur. Commun. Netw.* **2015**, *8*, 245–260. [\[CrossRef\]](#)
42. 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. Humaniz. Comput.* **2017**, *8*, 101–116. [\[CrossRef\]](#)
43. Wang, D.; Wang, P. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 708–722. [\[CrossRef\]](#)

