sensors

MDPI

Article

Practical Three-Factor Authentication Protocol Based on Elliptic Curve Cryptography for Industrial Internet of Things

Xingwen Zhao [1,2], Dexin Li [1,2,*] and Hui Li [1,2]

[1] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[2] School of Cyber Engineering, Xidian University, Xi'an 710000, China
* Correspondence: lidexin@stu.xidian.edu.cn

Abstract: Because the majority of information in the industrial Internet of things (IIoT) is transmitted over an open and insecure channel, it is indispensable to design practical and secure authentication and key agreement protocols. Considering the weak computational power of sensors, many scholars have designed lightweight authentication protocols that achieve limited security properties. Moreover, these existing protocols are mostly implemented in a single-gateway scenario, whereas the multigateway scenario is not considered. To deal with these problems, this paper presents a novel three-factor authentication and key agreement protocol based on elliptic curve cryptography for IIoT environments. Based on the elliptic curve Diffie–Hellman problem, we present a protocol achieving desirable forward and backward secrecy. The proposed protocol applies to single-gateway and is also extended to multigateway simultaneously. A formal security analysis is described to prove the security of the proposed scheme. Finally, the comparison results demonstrate that our protocol provides more security attributes at a relatively lower computational cost.

Keywords: industrial Internet of things; wireless sensor network; authentication and key agreement; elliptic curve cryptography; forward secrecy

Citation: Zhao, X.; Li, D.; Li, H. Practical Three-Factor Authentication Protocol Based on Elliptic Curve Cryptography for Industrial Internet of Things. *Sensors* **2022**, *22*, 7510. https://doi.org/10.3390/s22197510

Academic Editors: Christos Xenakis and Thanassis Giannetsos

Received: 26 August 2022
Accepted: 29 September 2022
Published: 3 October 2022

1. Introduction

The emerging industrial Internet of things (IIoT) is a typical application scenario for wireless sensor network (WSN), where the IIoT is dedicated to affording the capacity to construct innovative services and applications within the industrial automation scenario [1]. The IIoT emphasizes extremely low latency, high security, and the ability to handle massive quantities of data [2]. Therefore, efficient authentication and key agreement mechanisms should be designed for the IIoT infrastructure to ensure security and privacy. In this manner, only authorized principals can access the IIoT resource, and these legal entities can interact over the channel using the session key that they have negotiated.

Considering authentication protocols for sensors with a low computing power, the literature [3,4] sacrifices security to build lightweight protocols, resulting in these schemes being vulnerable to certain attacks. It is clearly found that schemes using only a hash function, exclusive OR (XOR), and symmetric cryptography are unable to achieve forward and backward secrecy. Ma et al. [5] claimed that the public key cryptography algorithm was indispensable to achieve forward secrecy. After that, public key cryptography technology was widely implemented in authentication protocols, where using elliptic curve cryptography (ECC) or bilinear pairings was able to help protocols achieve forward and backward secrecy.

Figure 1 illustrates that a representative IIoT architecture usually consists of three categories of entities: industrial IoT sensing devices, an industrial central, and an engineering expert [6], which, respectively, represent sensors, the gateway, and the user in WSNs. IIoT sensing devices are leveraged to monitor the status of objects and gather data, which is subsequently forwarded to a gateway via a wireless channel. A user is able to access the

data collected by the gateway in real time. Sensors, in general, have low processing power, limited computational capabilities, and restricted energy and storage capacity, whereas gateways have a strong capacity for data processing [7].

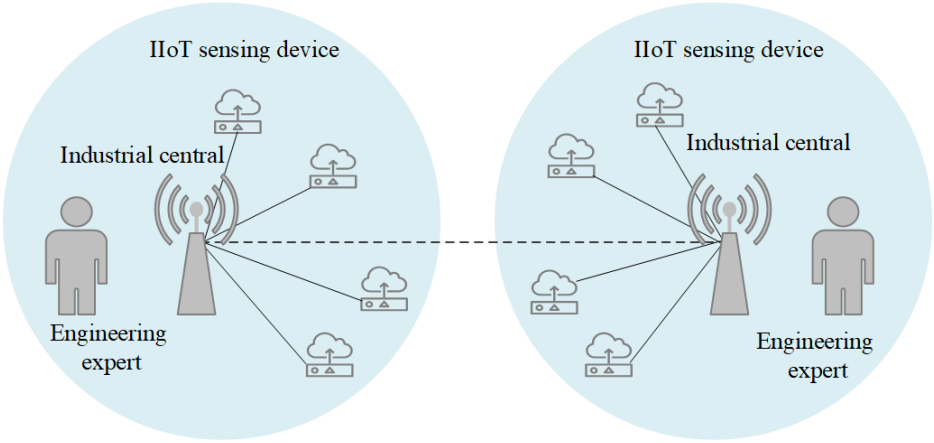

Figure 1. Architecture for an IIoT.

1.1. Literature Review

Das [8] first presented a password and smart-card-based two-factor user authentication protocol for WSNs using merely the hash function in 2009. Since then, some drawbacks to this scheme have been discovered by scholars. The presented schemes [9–11] identified some vulnerabilities in Das's scheme [8], and they suggested various countermeasures to overcome these flaws. In 2014, Turkanvoic et al. [12] proposed a novel user and mutual authentication scheme for WSNs using only a hash function and XOR. These lightweight schemes consumed relatively fewer resources but sacrificed security.

In order to achieve more security attributes, a public-key infrastructure was considered in some schemes. In 2011, Yeh et al. [13] performed a cryptanalysis of Das's scheme [8], and they discovered that there was no mutual authentication and no protection against an insider attack or forgery attack. As a result, they first implemented ECC to build the authentication protocol to address the current existing weaknesses. Shi and Gong [14] proposed a new ECC-based authentication protocol for WSNs in 2013, which addressed the shortcomings of the scheme in [13] that lacked a key agreement and forward secrecy. In 2016, Chang and Le [15] stated briefly that the scheme from Turkanovic et al. [12] suffered from an impersonation attack, stolen smart card attack, stolen-verifier attack, and failed to ensure backward secrecy, and they proposed an advanced scheme that used ECC to overcome these flaws. In 2018, Li et al. [16] indicated that the protocol in [15] lacked a proper mutual authentication and had other functionality defects. They [16] presented a three-factor user authentication protocol for the IIoT that addressed the protocol's [15] shortcomings by utilizing ECC and symmetric cryptography. A majority of protocols, however, are designed for a single-gateway scenario, ignoring how to implement them in a multigateway scenario.

In 2016, Aim and Biwas [17] solved some security flaws in the scheme from Turkanvoic et al. [12] and designed the first authentication protocols for a multigateway scenario. Later, Das et al. [18] indicated that there were no efficient online sensor node registration and password change phases in the literature [17], and they presented a new three-factor user authentication scheme applied to the multigateway WSN architecture using AES (Advanced Encryption Standard). In 2017, Wu et al. [19] demonstrated that the scheme in [17] suffered from tracking attacks due to the constant pseudo-identity and previously established session key that adversaries could calculate and presented a novel authentication scheme for multigateway WSNs. Srinivas et al. [20] showed that the protocol in [17] suffered from a stolen smart card attack, password guessing attack, and impersonation attack. They proposed an authentication scheme for multigateway WSNs that could

withstand all the above-mentioned attacks. In 2018, Wang et al. [21] discovered that the scheme in [20] was still subject to offline password guessing attacks and node capture attacks and could not protect the user's anonymity. Therefore, they described efficient countermeasures for these attacks. Since all the above-mentioned multigateway schemes use lightweight cryptographic primitives, it is impossible to achieve forward and backward secrecy. Accordingly, our scheme will solve this problem.

1.2. Network Model

Figure 2 demonstrates how the single-gateway model is implemented in our presented IIoT protocol. After the user logs in, they send the message to the home gateway node (HGWN). If the user can pass the authentication of the HGWN, the HGWN sends the message to the sensor. After the sensor authenticates, it computes the session key and sends a message to the HGWN. Finally, the HGWN sends a message to the user, who calculates the session key to communicate with the sensor. Through two rounds of complete information exchange, the user, HGWN, and sensor can realize mutual authentication.

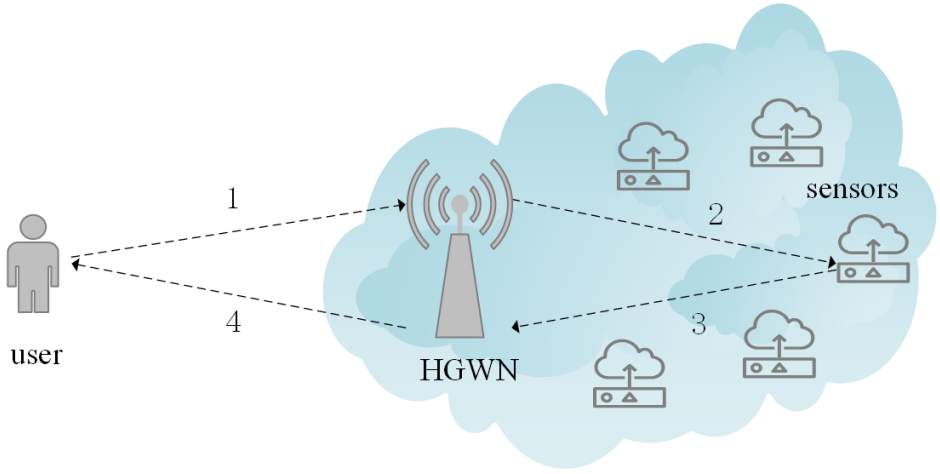

Figure 2. Single-gateway model.

Nevertheless, in traditional single-gateway WSNs, high-speed data streams are prone to conflict during data aggregation, because the distance between edge sensors and the gateway node is too far, which may cause an increased communication cost and reduced performance. In this case, multigateway protocols are required, and Figure 3 shows the model we used. This architecture is an extension of Figure 2. The user sends the authentication message to the HGWN. Following that, the HGWN checks the validity of the received message. In the event that this procedure is successful, the HGWN sends a message to the FGWN. The FGWN transmits a message to the HGWN after confirming the message's availability. Then, the HGWN checks the received message and delivers a message to the user. Following steps 1–4, the mutual authentication is achieved between the user and the FWGN. After that, user sends a message to the FGWN for further authentication. After the verification is successful, the FGWN transmits a message to the sensor. Subsequently, the sensor computes the session key and delivers a message to the FGWN. Finally, the user figures out the session key used for subsequent communication after confirming the message that the FGWN sent to it.

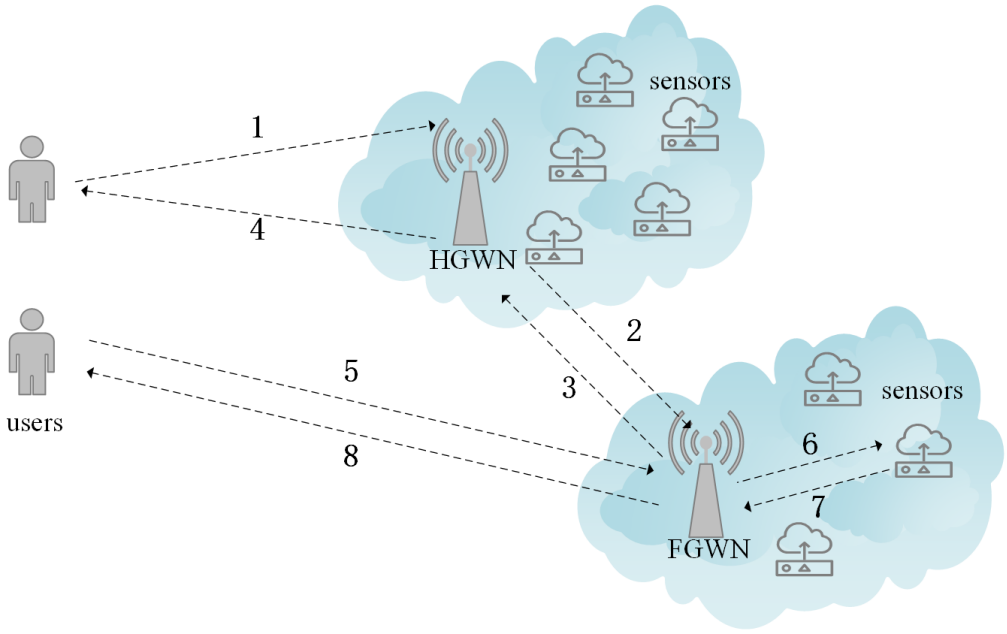

Figure 3. Multigateway model.

1.3. Motivations and Contributions

1. Intractable elliptic curve Diffie–Hellman problem (ECDHP) is applied to our protocol to guarantee the security of the session key. We extend our scheme to multigateway WNSs while considering the limitations of single-gateway WSNs.

2. The random oracle model (ROM) [22] helps us get the formal proof of the presented scheme. The result indicates that the probability of an adversary who can break the proposed protocol is negligible.

3. Scyther, an automated security protocol verification tool [23], is used to simulate and analyze the proposed protocol. The result demonstrates that the scheme is correct and secure against many adversary models.

2. Preliminaries

2.1. Elliptic Curve Cryptography

ECC was initially proposed by Koblitz [24] and Miller [25] in the 1980s, and an introduction to the basic knowledge of ECC is described in the following. Given a large prime number p and a finite field F_p, let a set of elliptic curve points E over F_p be defined by the equation: $E(F_p) : y^2 = x^3 + a \cdot x + b \bmod p$, where $a, b \in F_p$ and $\Delta = 4a^3 + 27b^2 \neq 0 \bmod p$. All points on $E(F_p)$ and the point O at infinity come from an additive Abelian group G of order q, where P is the generator point of the group and $n \cdot P = P + P + \ldots + P$, where n is an integer and $n \in Z_q^*$. There are two corresponding mathematical problems in ECC defined as follows:

1. The elliptic curve discrete logarithm problem (ECDLP): Figure 4 demonstrates points distributed over an elliptic curve $y^2 = x^3 - x + 2$ in finite field F_{97}. Selecting two points Q and P in Figure 4, where $Q, P \in F_{97}$ satisfy $Q = kP$, where k is between 0 and 96 at random. Given k and P, it is easy to figure out Q by a scalar multiplication and addition rules. Nevertheless, given Q and P, it is difficult to calculate k.
2. The elliptic curve Diffie–Hellman problem (ECDHP): It is scarcely possible to find abP when given $aP \in F_p$ and $bP \in F_p$ in polynomial time, where a and b are both between 0 and $p - 1$ at random.

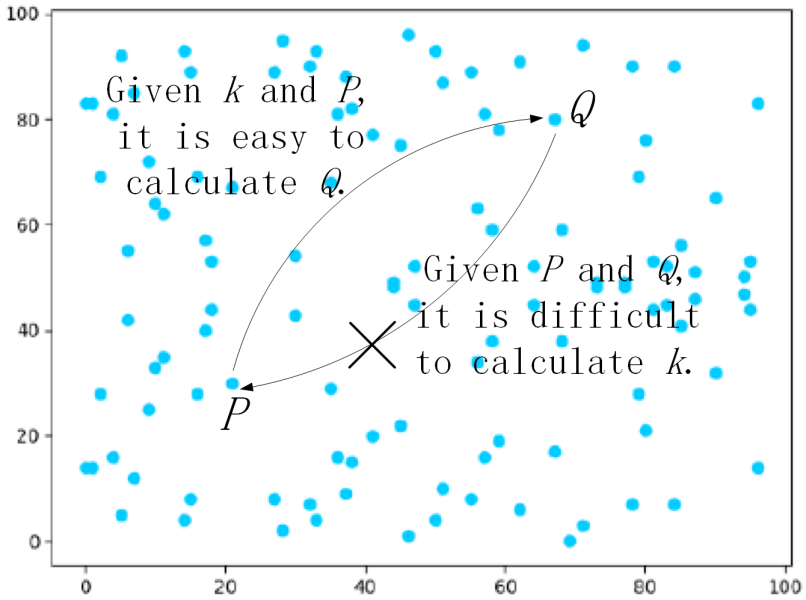

Figure 4. Points over the elliptic curve.

2.2. Threat Model

The proposed authentication and key agreement protocol was formally analyzed taking advantage of the Dolev–Yao threat model [26], which assumes that two communication principals interact over an insecure and open channel. The following are the properties of this model:

1. The used one-way hash function is unbreakable.
2. In a uniform protocol, an identical format is used by each entity that wishes to communicate.
3. An adversary can eavesdrop, intercept, replay, and even modify all the transmitted messages over an open and insecure channel.

2.3. Fuzzy Extractor

Biometric features are adopted to improve security in many schemes. Due to the uniqueness of biometric features, they can be effectively applied to authentication. Compared with low-entropy passwords, biometric features also have the advantages of being difficult to forge and not being easy to lose.

The fuzzy extractor was used to process the original biometric fingerprint, which can eliminate subtle differences between biometric features extracted by the same user at different points in time. A fuzzy extractor comprises two phases as follows Ref. [27]:

1. Probabilistic generation function *Gen*: The original biometric fingerprint BIO_i is the input of *Gen*, and then the process outputs biometric identification key data and public parameter, namely $Gen(BIO_i) \rightarrow (\sigma_i, \theta_i)$.
2. Deterministic reproduction procedure *Rep*: Using the public parameter θ_i and the fingerprint BIO_i reproduces key data σ_i, namely $Rep(BIO_i, \theta_i) \rightarrow \sigma_i$.

3. The Proposed Scheme

In this section, the detailed process of the proposed scheme is demonstrated. The proposed scheme consists of the following phases: initialization phase, registration phase, user login phase, authentication and key agreement phase, and user password update phase.

3.1. Initialization Phase

All the parameters that are used in the proposed protocol are listed in Table 1. During the initialization phase, SA chooses an elliptic curve E over a prime finite field F_p, a point $P \in E(F_p)$ and a subgroup G of $E(F_p)$, where G is an additive cyclic group of order q. Then, the $HGWN$ generates its private key and public key $\{k_h, K_h\}$, where $k_h \in Z_q^*$ and $K_h = k_h P$. Consistent with the above procedure, the $FGWN$ chooses its private key and public key $\{k_f, K_f\}$, where $k_f \in Z_q^*$ and $K_f = k_f P$. Finally, the hash function $h(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^l$ is chosen to be used in the scheme, where l is the length of the output length of the hash function.

Table 1. Symbol description.

Symbol	Description
SA	System administrator
U_i	ith user node
SN_j	jth sensor node
SC_i	Smart card of U_i
$HGWN$	Home gateway node
$FGWN$	Foreign gateway node
ID_i	Identity of U_i
SID_j	Identity of SN_j
PW_i	Password of U_i
BIO_i	Biometric information of U_i
k_h, K_h	Private key and public key of $HGWN$
k_f, K_f	Private key and public key of $FGWN$
r_h, r_{hg}, r_f, r_{fg}	Random numbers
a, b, c, d	Random numbers $\in Z_q^*$
P	A point on the elliptic curve
$T_1, T_2, ..., T_8$	Timestamps
ΔT	Acceptable maximum transmission delay
SK	Session key
$h()$	One-way hash function
$\oplus$	Exclusive-or operation
$\|\|$	Concatenation operation
$Gen()$	Fuzzy extractor probabilistic generation procedure
$Rep()$	Fuzzy extractor deterministic reproduction procedure

3.2. Registration Phase

The registration phase is divided into a user registration phase and a sensor registration phase. All the messages in this phase are transmitted via a secure channel.

3.2.1. User Registration Phase

The procedure is also shown in Figure 5.

Step 1: U_i selects their identity ID_i and password PW_i, and inputs biometric information BIO_i. The fuzzy extractor is used to compute biometric key data σ_i and public parameter θ_i, namely $Gen(BIO_i) \rightarrow (\sigma_i, \theta_i)$. SC_i stores the public parameter θ_i in its memory. Then, U_i figures out $HID_i = h(ID_i||\sigma_i)$ and $HPW_i = h(PW_i||\sigma_i)$, and sends $\{HID_i, HPW_i\}$ to the nearest $HGWN$ via a secure channel.

Step 2: Upon receiving $\{HID_i, HPW_i\}$ from U_i, the $HGWN$ generates a random number r_h and calculates $A_i = h(HID_i||k_h||r_h) \oplus HID_i$, $B_i = h(HID_i||HPW_i||r_h)$, and $C_i = HID_i \oplus r_h$. The $HGWN$ stores $\{HID_i, r_h\}$ in its memory. Then, the $HGWN$ sends $\{A_i, B_i, C_i\}$ to U_i via a secure channel.

Step 3: Upon getting $\{A_i, B_i, C_i\}$ from $HGWN$, U_i stores $\{A_i, B_i, C_i, \theta_i\}$ into its own SC_i.

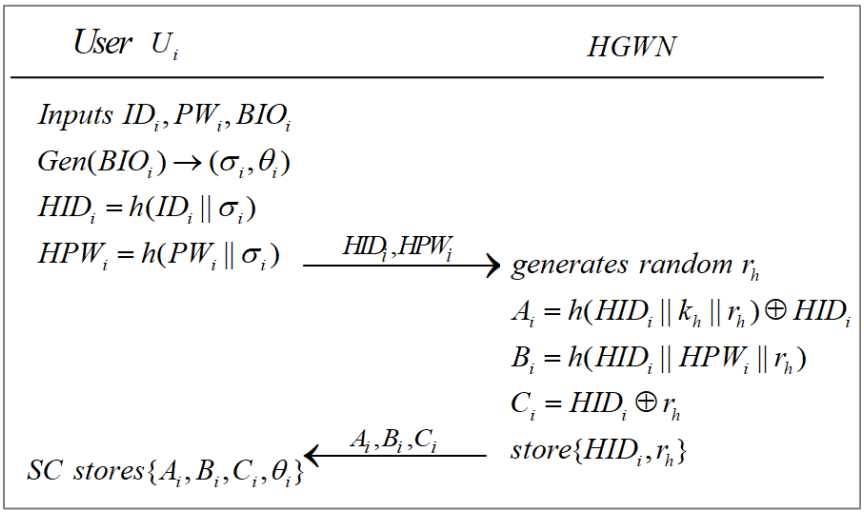

Figure 5. User registration phase.

3.2.2. Sensor Registration Phase

Sensor registration process is shown in Figure 6. SA assigns a unique identity to each sensor node. SN_j sends its own identity SID_j to the nearest $HGWN$ via a secure channel for registration. Then, the $HGWN$ calculates $A_{gs} = h(SID_j||k_h)$ and stores $\{SID_j, A_{gs}\}$ in its memory. After that, the $HGWN$ sends A_{gs} to SN_j via a secure channel. After receiving A_{gs} from the $HGWN$, SN_j stores $\{SID_j, A_{gs}\}$ in its own memory.

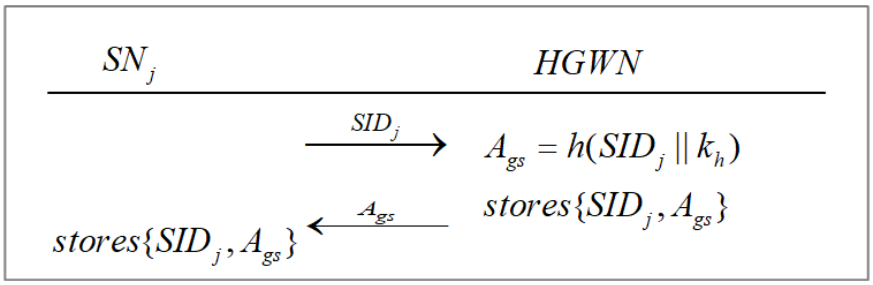

Figure 6. Sensor registration phase.

3.3. User Login Phase

U_i inserts their smart card SC_i to a terminal, and inputs identity ID_i, password PW_i and biometric information BIO_i. Then, the terminal reproduces the biometric key data σ_i through the fuzzy extractor, namely $Rep(BIO_i, \theta_i) \rightarrow \sigma_i$. The terminal computes $HID_i = h(ID_i||\sigma_i)$, $HPW_i = h(PW_i||\sigma_i)$, $r'_h = HID_i \oplus C_i$ and $B'_i = h(HID_i||HPW_i||r'_h)$. Subsequently, the terminal checks whether $B'_i \stackrel{?}{=} B_i$. If the equation is not held, at least one parameter is incorrect, which leads to the login request being refused by the terminal and no subsequent authentication process being performed. Otherwise, U_i's login is successful, and the terminal generates a random number $a \in Z_q^*$, and a timestamp T_1. At last, the terminal computes $A_h = A_i \oplus HID_i$, $D_1 = aP$, $D_2 = aK_h$, $M_1 = HID_i \oplus h(D_2)$, $M_2 = SID_j \oplus h(D_2) \oplus A_h$, and $M_3 = h(HID_i||A_h||D_2||M_1||M_2||T_1)$. This process is demonstrated in Figure 7.

U_i	*Terminal*
Inserts smart card SC_i *to a terminal*	
Inputs ID_i, PW_i, BIO_i	*computes* $Rep(BIO_i, \theta_i) \rightarrow \sigma_i$
	$HID_i = h(ID_i \| \sigma_i)$
	$HPW_i = h(PW_i \| \sigma_i)$
	$r_h^{'} = HID_i \oplus C_i$
	$B_i^{'} = h(HID_i \| HPW_i \| r_h^{'})$
	if $B_i^{'} \neq B_i$ *aborts*
	generates random $a \in Z_q^*$, *timestamp* T_1
	computes $A_h = A_i \oplus HID_i$
	$D_1 = aP, D_2 = aK_h$
	$M_1 = HID_i \oplus h(D_2)$
	$M_2 = SID_j \oplus h(D_2) \oplus A_h$
	$M_3 = h(HID_i \| A_h \| D_2 \| M_1 \| M_2 \| T_1)$

Figure 7. User login phase.

3.4. Authentication and Key Agreement Phase

In this section, two cases are considered: authentication and key agreement in a home region and a foreign region, respectively.

3.4.1. Authentication and Key Agreement in the HGWN

When a user and the sensor that they want to access are in the same region controlled by the same $HGWN$, as illustrated in Figure 8, each entity will execute the following steps.

Step 1: U_i sends the login request message $\{M_1, M_2, M_3, D_1, T_1\}$ to the $HGWN$.

Step 2: After receiving $\{M_1, M_2, M_3, D_1, T_1\}$ from U_i, the $HGWN$ checks whether $|T_1' - T_1| < \Delta T$ is satisfied, where T_1' is the current timestamp the $HGWN$ acquired and ΔT is the acceptable maximum transmission delay. If the inequality is not true, namely T_1 is not fresh, the $HGWN$ aborts the current session. Otherwise, the $HGWN$ computes $D_2' = k_h D_1$ and $HID_i' = M_1 \oplus h(D_2')$ to find r_h stored in its own memory. Subsequently, the $HGWN$ calculates $A_h' = h(HID_i' || k_h || r_h)$, $SID_j' = M_2 \oplus h(D_2') \oplus A_h'$, and $M_3' = h(HID_i' || A_h' || D_2' || M_1 || M_2 || T_1)$, and checks whether $M_3' \stackrel{?}{=} M_3$. The current session is aborted if $M_3' \neq M_3$. Otherwise, the $HGWN$ seeks A_{gs} from its own memory through SID_j, generates a random number r_{hg}, a timestamp T_2, and calculates $M_4 = r_{hg} \oplus h(A_{gs} || T_2)$, $M_5 = h(SID_j || r_{hg} || A_{gs} || D_1 || T_2)$. Finally, the $HGWN$ sends $\{M_4, M_5, D_1, T_2\}$ to SN_j.

Step 3: When SN_j receives $\{M_4, M_5, D_1, T_2\}$ from the $HGWN$, SN_j obtains the current timestamp T_2' and verifies whether $|T_2' - T_2| < \Delta T$. If the inequality is not held, then SN_j terminates the current session. Otherwise, SN_j figures out $r_{hg}' = h(A_{gs} || T_2) \oplus M_4$, $M_5' = h(SID_j || r_{hg}' || A_{gs} || D_1 || T_2)$, and examines whether $M_5' \stackrel{?}{=} M_5$. The current session is terminated if $M_5' \neq M_5$. Otherwise, SN_j generates a random number $b \in Z_q^*$, a timestamp T_3, and figures out $D_3 = bP$, $D_4 = bK_h$, $SK = h(D_1 || D_3 || bD_1)$, $M_6 = h(SID_j || r_{hg} || A_{gs} || D_4 || T_3)$, and $M_7 = h(SK || D_1 || D_3)$. Lastly, SN_j transmits $\{M_6, M_7, D_3, T_3\}$ to the $HGWN$.

Step 4: After getting $\{M_6, M_7, D_3, T_3\}$ from SN_j, the $HGWN$ acquires the current timestamp T_3' and verifies whether $|T_3' - T_3| < \Delta T$. If the verification fails, the $HGWN$ aborts the current session. Otherwise, the $HGWN$ calculates $D_4' = k_h D_3$, $M_6' = h(SID_j || r_{hg} || A_{gs} || D_4' || T_3)$, and checks whether $M_6' \stackrel{?}{=} M_6$. If $M_6' \neq M_6$, the $HGWN$ aborts the current session. Otherwise, the $HGWN$ generates a timestamp T_4, calculates $M_8 = h(HID_i || A_h || D_1 || D_3 || M_7 || T_4)$, and dispatches $\{M_7, M_8, D_3, T_4\}$ to U_i.

Step 5: Upon receiving $\{M_7, M_8, D_3, T_4\}$ from the $HGWN$, U_i obtains the current timestamp T_4' and checks whether $|T_4' - T_4| < \Delta T$. If the verification fails, the current session is rejected by U_i. Otherwise, U_i computes $M_8' = h(HID_i||A_h||D_1||D_3||M_7||T_4)$ and checks whether $M_8' \stackrel{?}{=} M_8$. If $M_8' \neq M_8$, U_i aborts the current session. Otherwise, U_i computes $SK' = h(D_1||D_3||aD_3)$, $M_7' = h(SK'||D_1||D_3)$, and verifies whether $M_7' \stackrel{?}{=} M_7$. If not, U_i declines to establish a session key with SN_j. Otherwise, U_i and SN_j share an identical session key, and the authentication process is successfully completed.

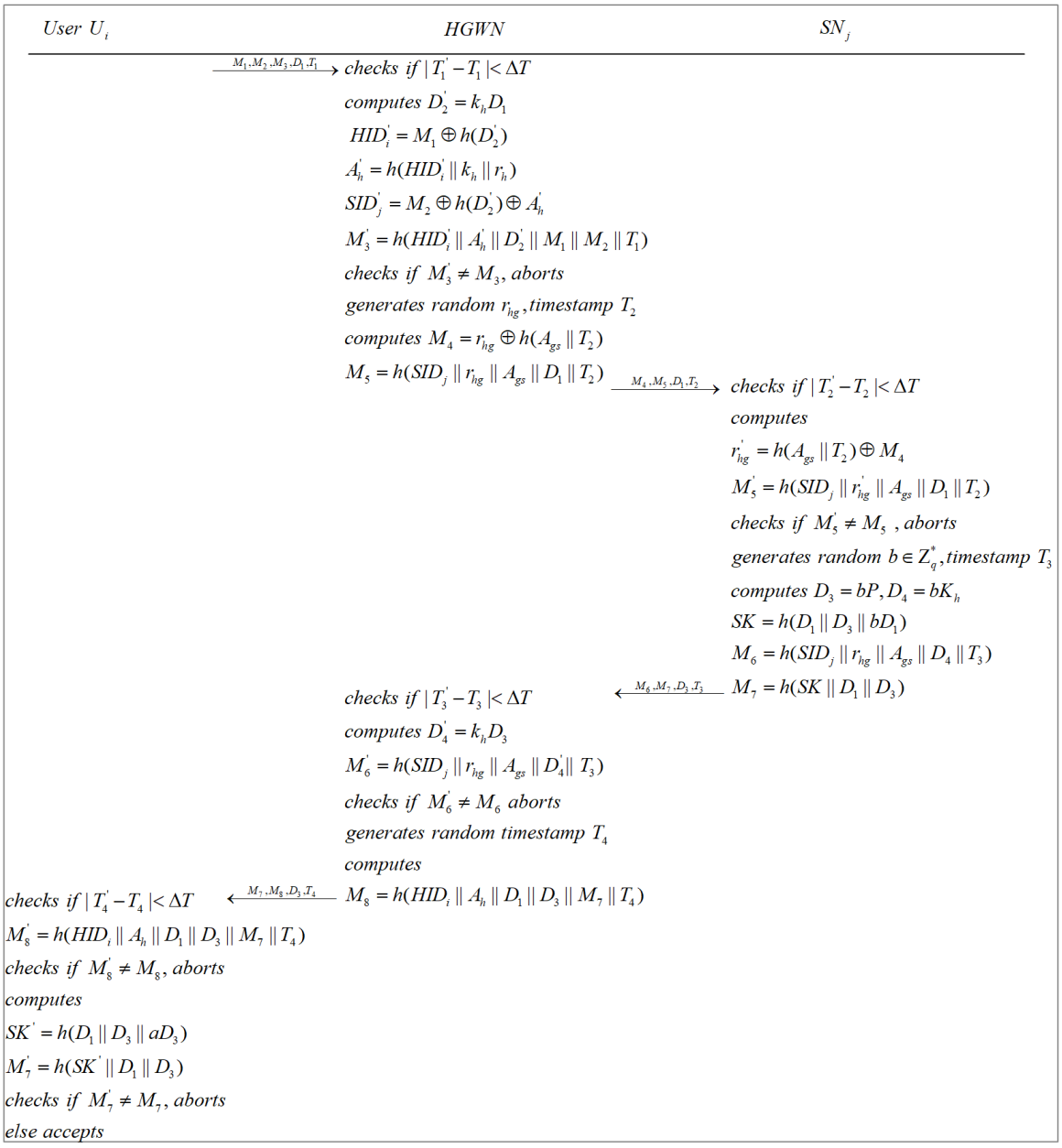

Figure 8. Authentication and key agreement in the HGWN.

3.4.2. Authentication and Key Agreement in the FGWN

When a user requires access to a sensor that is in a foreign region and registered in a $FGWN$, this phase can be completed with the assistance of the $HGWN$ and the $FGWN$, as illustrated in Figures 9 and 10.

Step 1: U_i computes the login request message $\{M_1, M_2, M_3, D_1, T_1\}$ as in the User Login Phase Section and sends them to the $HGWN$.

Step 2: After receiving $\{M_1, M_2, M_3, D_1, T_1\}$ from U_i, the $HGWN$ obtains the current timestamp T_1' and verifies T_1's validity, namely $|T_1' - T_1| < \Delta T$. If the verification fails, the $HGWN$ aborts. Otherwise, the $HGWN$ calculates $D_2' = k_h D_1$, $HID_i' = M_1 \oplus h(D_2')$, $A_h' = h(HID_i'||k_h||r_h)$, $SID_j' = M_2 \oplus A_h' \oplus h(D_2')$, and $M_3' = h(HID_i'||A_h'||D_2'||M_1||M_2||T_1)$. Subsequently, the $HGWN$ checks whether $M_3' \stackrel{?}{=} M_3$. The current session is aborted if $M_3' \neq M_3$. Next, if SID_j is not in the $HGWN$'s database, the $HGWN$ broadcasts the target sensor's identity SID_j to the rest of the gateway nodes. If any $FGWN$ finds SID_j in its database, it will react to the $HGWN$ and broadcasts its own public key K_f in WSNs. Subsequently, the $HGWN$ generates a random number $b \in Z_q^*$, timestamp T_2, and computes $D_3 = bP$, $D_4 = bK_f$, $(b + k_h)K_f$, and $M_4 = h(SID_j||D_3||(b + k_h)K_f||T_2)$. Finally, the $HGWN$ dispatches $\{M_4, D_3, T_2\}$ to the corresponding $FGWN$.

Step 3: Upon receiving $\{M_4, D_3, T_2\}$ from the $HGWN$, the corresponding $FGWN$ obtains the current timestamp T_2' and verifies whether $|T_2' - T_2| < \Delta T$. If not, the $FGWN$ terminates the current session. Otherwise, the $FGWN$ computes $D_4' = k_f D_3$, $D_4' + k_f K_h$, and $M_4' = h(SID_j||D_3||D_4' + k_f K_h||T_2)$, and examines whether $M_4' \stackrel{?}{=} M_4$. the $FGWN$ terminates the current session if $M_4' \neq M_4$. Otherwise, the $FGWN$ generates random numbers $c \in Z_q^*$, r_f, a timestamp T_3, and calculates $D_5 = cP$, $D_6 = cK_h$, $(c + k_f)K_h$, $A_f = h(HID_i||k_f||r_f)$, $M_5 = A_f \oplus h(D_6)$, and $M_6 = h(SID_j||A_f||(c + k_f)K_h||M_5||T_3)$. Then, the $FWGN$ transmits $\{M_5, M_6, D_5, T_3\}$ to the $HGWN$.

Step 4: Upon getting $\{M_5, M_6, D_5, T_3\}$ from the $FGWN$, the $HGWN$ acquires the current timestamp T_3' and verifies whether $|T_3' - T_3| < \Delta T$. If the verification fails, the $HGWN$ rejects the current session. Otherwise, the $HGWN$ figures out $D_6' = k_h D_5$, $D_6' + k_h K_f$, $A_f' = M_5 \oplus h(D_6')$, and $M_6' = h(SID_j||A_f'||D_6' + k_h K_f||M_5||T_3)$, and checks whether $M_6' \stackrel{?}{=} M_6$. If $M_6' \neq M_6$, the $HGWN$ rejects the current session. Otherwise, the $HGWN$ generates a timestamp T_4, calculates $M_7 = A_f \oplus A_h$, $M_8 = h(HID_i||SID_j||A_h||A_f||M_7||T_4)$, and dispatches $\{M_7, M_8, T_4\}$ to U_i.

Step 5: After receiving $\{M_7, M_8, T_4\}$ from the $HWGN$, U_i gets the current timestamp T_4' and checks whether $|T_4' - T_4| < \Delta T$. If not, the current session is rejected by U_i. Otherwise, U_i computes $A_f' = M_7 \oplus A_h$, $M_8' = h(HID_i||SID_j||A_h||A_f'||M_7||T_4)$ and checks whether $M_8' \stackrel{?}{=} M_8$. If $M_8' \neq M_8$, U_i rejects the current session. Otherwise, U_i generates a timestamp T_5 and computes $D_{2f} = aK_f$, $M_9 = HID_i \oplus h(D_{2f})$, $M_{10} = h(HID_i||A_f||D_{2f}||M_9||T_5)$, and delivers $\{M_9, M_{10}, T_5\}$ to the $FGWN$.

Step 6: After receiving $\{M_9, M_{10}, T_5\}$ from U_i, the $FGWN$ obtains the current timestamp T_5' and checks whether $|T_5' - T_5| < \Delta T$ is satisfied. If failed, the $FGWN$ aborts the current session. Otherwise, the $FGWN$ computes $D_{2f}' = k_f D_1$, $HID_i' = M_9 \oplus h(D_{2f}')$, $M_{10}' = h(HID_i'||A_f||D_{2f}'||M_9||T_5)$, and checks whether $M_{10}' \stackrel{?}{=} M_{10}$. The current session is aborted if $M_{10}' \neq M_{10}$. Otherwise, $FGWN$ generates a random number r_{fg}, a timestamp T_6, and calculates $M_{11} = r_{fg} \oplus h(A_{fs}||T_6)$, $M_{12} = h(SID_j||r_{fg}||A_{fs}||D_1||T_6)$. Finally, $FGWN$ sends $\{M_{11}, M_{12}, T_6\}$ to SN_j.

Step 7: When SN_j receives $\{M_{11}, M_{12}, T_6\}$ from the $FGWN$, SN_j obtains the current timestamp T_6' and verifies whether $|T_6' - T_6| < \Delta T$. If not, SN_j aborts the current session. Otherwise, SN_j computes $r_{fg}' = h(A_{fs}||T_6) \oplus M_{11}$, $M_{12}' = h(SID_j||r_{fg}'||A_{fs}||D_1||T_6)$, and examines whether $M_{12}' \stackrel{?}{=} M_{12}$. The current session is aborted if $M_{12}' \neq M_{12}$. Otherwise, SN_j generates a random number $d \in Z_q^*$, a timestamp T_7, and figures out $D_7 = dP$, $D_8 = dK_f$, $SK = h(D_7||dD_1)$, $M_{13} = h(SID_j||r_{fg}||A_{fs}||D_8||T_7)$, and $M_{14} = h(SK||D_7)$. After that, SN_j transmits $\{M_{13}, M_{14}, D_7, T_7\}$ to the $FGWN$.

Step 8: After getting $\{M_{13}, M_{14}, D_7, T_7\}$ from SN_j, the $FGWN$ acquires the current timestamp T_7' and verifies whether $|T_7' - T_7| < \Delta T$. If the verification fails, the $FGWN$ aborts the current session. Otherwise, the $FGWN$ computes $D_8' = k_f D_7$, $M_{13}' = h(SID_j||r_{fg}||A_{fs}||D_8'||T_7)$, and checks whether $M_{13}' \stackrel{?}{=} M_{13}$. If $M_{13}' \neq M_{13}$, the $FGWN$

aborts the current session. Otherwise, the $FGWN$ generates a timestamp T_8, calculates $M_{15} = h(HID_i||A_f||D_1||D_7||M_{14}||T_8)$, and dispatches $\{M_{14}, M_{15}, D_7, T_8\}$ to U_i.

Step 9: After receiving $\{M_{14}, M_{15}, D_7, T_8\}$ from the $FGWN$, U_i obtains the current timestamp T_8' and checks whether $|T_8' - T_8| < \Delta T$. If not, U_i rejects the current session. Otherwise, U_i computes $M_{15}' = h(HID_i||A_f||D_1||D_7||M_{14}||T_8)$ and checks whether $M_{15}' \stackrel{?}{=} M_{15}$. If $M_{15}' \neq M_{15}$, U_i aborts the current session. Otherwise, U_i figures out $SK' = h(D_7||aD_7)$, $M_{14}' = h(SK'||D_7)$, and verifies whether $M_{14}' \stackrel{?}{=} M_{14}$. If the verification fails, U_i declines to establish a session key with SN_j. Otherwise, U_i and SN_j share an identical session key, and the authentication process is successfully completed.

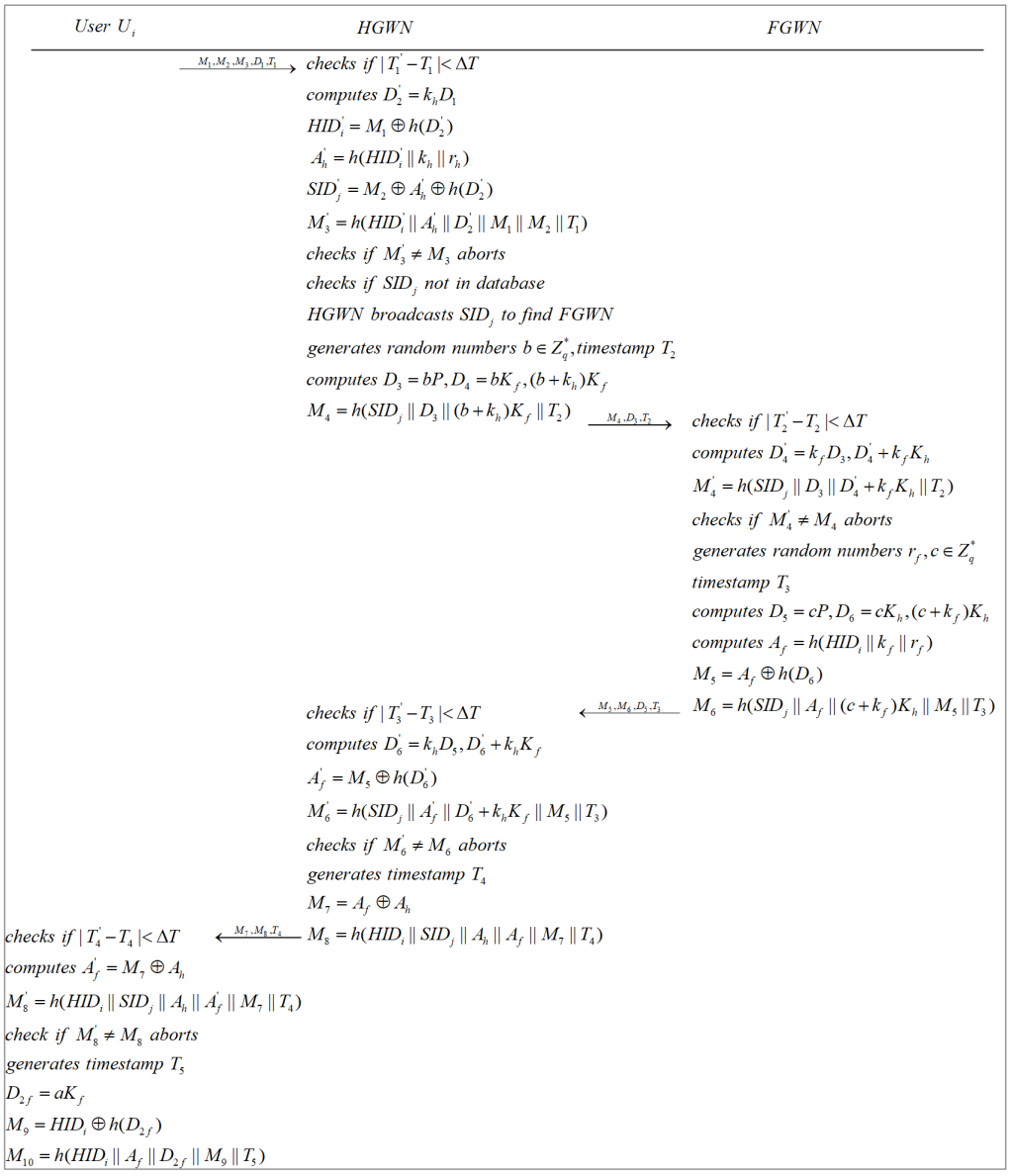

Figure 9. Authentication and key agreement phase 1 in the FGWN.

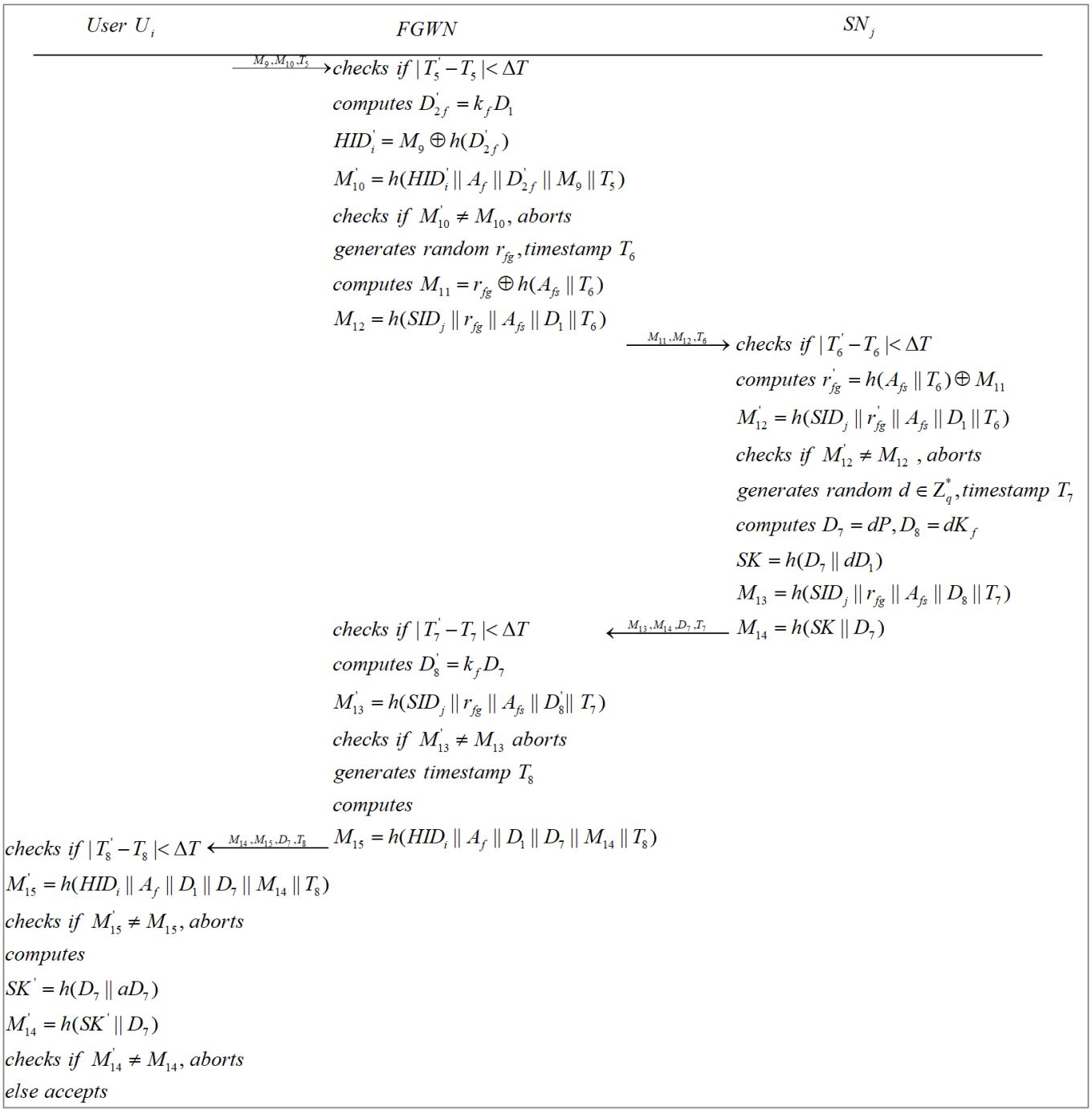

Figure 10. Authentication and key agreement phase 2 in the FGWN.

3.5. User Password Update Phase

U_i inserts their smart card SC_i into the terminal, and enters identity ID_i, password PW_i, and biometric information BIO_i. Then, the terminal reproduces the biometric key data $Rep(BIO_i, \theta_i) \rightarrow \sigma_i$ and reads secret parameter $C_i = HID_i \oplus r_h$ in SC_i to calculate $HID_i = h(ID_i||\sigma_i)$, $HPW_i = h(PW_i||\sigma_i)$, and $r_h' = HID_i \oplus C_i$. Next, the terminal checks $B_i \stackrel{?}{=} h(HID_i||HPW_i||r_h')$. If the equation is not held, this update request is rejected. Otherwise, this request is acknowledged, and the subsequent phase is performed. In the update phase, U_i enters a new password PW_i^{new}. Subsequently, the terminal computes $HPW_i^{new} = h(PW_i^{new}||\sigma_i)$ and updates $B_i^{new} = h(HID_i||HPW_i^{new}||r_h')$ in SC_i.

4. Security Analysis

4.1. Formal Security Proof

The security of our protocol is proved under the ROM.

4.1.1. Formal Security Model

The security of the presented protocol dependent on the CK model [28].

Participants: In this model, the adversary $\mathscr{A}$ controls the communication between all participants. For the single-gateway scenario, there are three types of participants in this protocol P: the user U, the gateway $HGWN$, and the sensor SN. Each principal has a large number of instances, which are usually treated as the actions of specific protocols run by each principal. U^i, $HGWN^k$, and SN^j represent the ith instance of U, kth instance of $HGWN$, and jth instance of SN in P separately. Moreover, I denotes any other instance.

Queries: The interaction between $\mathscr{A}$ and the protocol principals occurs merely through oracle queries, which simulate $\mathscr{A}$'s capabilities to break P in a real attack. $\mathscr{A}$ is allowed to execute the following queries.

$Execute(U^i, HGWN^k, SN^j)$: $\mathscr{A}$ uses this query to simulate a passive attack, and they can obtain the entire transcript as a result of the conversation among U, $HGWN$, and SN.

$Send(I^i, m)$: It models an active attack of $\mathscr{A}$, who forges a message m and sends it to instance I^i. Subsequently, I^i returns the processing outcomes of the message m to $\mathscr{A}$ according to P. If the message m is invalid, the query is ignored.

$SKReveal(I^i)$: This query simulates that $\mathscr{A}$ can obtain session key SK of any completed session.

$SSReveal(I^i)$: This query can be asked of an incomplete session and receives the internal state in return.

$Corrupt(I^i)$: This query can help $\mathscr{A}$ obtain the private key of I^i, which is usually used to simulate the forward secrecy of protocols. $\mathscr{A}$ can obtain the private key of U, $HGWN$, and SN.

$Test(I^i)$: $\mathscr{A}$ asks this query to a fresh instance. Then, $\mathscr{A}$ can continue to ask other queries, as long as the tested session remains fresh. In other words, if I^i has been asked $SSReveal(I^i)$, $SKReveal(I^i)$, or $Corrupt(I^i)$, both I^i and its partner cannot be asked by a *Test* query.

$Test(I^i)$ query is used to evaluate the semantic security of a session key. Only one test query is allowed to be executed during the whole game. To answer the test query, we imagine a challenger who flips a coin to define a bit b. If there is no session key established for instance I^i, then $\perp$ is returned. If the query has already been asked, then it outputs the same answer as above. Otherwise, if $b = 1$, I^i returns the real session key. If $b = 0$, I^i returns an entirely random string of the same length as the session key. The final output of $Test(I^i)$ is a bit b', which is the guessing value of b. The adversary wins this game if and only if $b' = b$.

4.1.2. Security Proof

Suppose $\mathscr{A}$ is the adversary who can break protocol P in polynomial time. q_{hash} and q_{send} refer to the number of hash query oracles and send query oracles, respectively. $Adv_P^{ECDHP}(t)$ represents the advantage of an adversary who can resolve the intractable $ECDHP$ in polynomial time. Now, the advantage of $\mathscr{A}$ that breaks the semantic security of our authentication and key agreement (AKA) protocol is defined:

$$Adv_P^{AKA}(\mathscr{A}) \leq \frac{q_{hash}^2}{2^l} + \frac{q_{send}}{2^{l-1}} + 2Adv_P^{ECDHP}(t) \quad (1)$$

Proof. Game i (i = 0, 1, 2, 3, 4) is used to perform the whole procedure of P. The event WG_i signifies that $\mathscr{A}$ guesses the bit b correctly to win the game. □

Game 0: In the random oracle model, the real attack on P is modeled, and the following formula can be obtained:

$$Adv_P^{AKA}(\mathscr{A}) = |2\Pr[WG_0] - 1| \tag{2}$$

Game 1: $\mathscr{A}$ carries out *Execute* queries to model an eavesdropping attack. Even if we take *Execute* queries into consideration, the probability of an adversary who can win the game has not increased.

$$Pr[WG_1] = Pr[WG_0] \tag{3}$$

Game 2: Hash oracles are added to the foundation of *Game* 1 by *Game* 2. This game models the active attack, and $\mathscr{A}$ attempts to trick a legitimate principal into accepting the modified message. When the collision happens between the constructed information and the real authentication information, $\mathscr{A}$ gets the secret information and wins the game. According to the birthday paradox, the maximum probability of the hash oracle collision is $\frac{q_{hash}^2}{2^{l+1}}$, and we have:

$$|Pr[WG_2] - Pr[WG_1]| \leq \frac{q_{hash}^2}{2^{l+1}} \tag{4}$$

Game 3: *Send* queries are added. This game models the active attack, and $\mathscr{A}$ attempts to trick a legitimate principal into accepting the modified message. Therefore, we have:

$$|Pr[WG_3] - Pr[WG_2]| \leq \frac{q_{send}}{2^l} \tag{5}$$

Game 4: In this game, $\mathscr{A}$ asks *Execute* queries eavesdropping on all exchanged messages $\{M_1, M_2, M_3, D_1, T_1\}$, $\{M_4, M_5, D_1, T_2\}$, $\{M_6, M_7, D_3, T_3\}$, and $\{M_7, M_8, D_3, T_4\}$. $\mathscr{A}$ executes $Corrupt(I^i)$ to obtain the private key of this entity, where I is equal to U, $HGWN$, and SN successively, and thus $\mathscr{A}$ can obtain all the private keys. $SKReveal(I^i)$ can be executed in this game. It will answer an SK if the target instance has formed an SK. $\mathscr{A}$ executes $SSReveal(I^i)$ to get the internal state of an incomplete session. In order to compute the session key, $\mathscr{A}$ has to resolve the intractable $ECDHP$ to get a or b from $D_1 = aP$ or $D_3 = bP$. Let $Adv_P^{ECDHP}(t)$ be the advantage of $\mathscr{A}$, who can resolve the $ECDHP$ in polynomial time. As a result, we get:

$$|Pr[WG_4] - Pr[WG_3]| \leq Adv_P^{ECDHP}(t) \tag{6}$$

At the end of *Game* 4, all the queries are simulated, so what $\mathscr{A}$ can do is to guess the bit b to win the game after performing *Test* query. Now, we have the following:

$$Pr[WG_4] = \frac{1}{2} \tag{7}$$

According to Equations (2)–(7), we can obtain Equation (1). It indicates that the adversary has negligible advantage in winning the game. Therefore, our protocol is secure under the random oracle model.

4.2. Formal Verification Using Scyther

Scyther is a tool for the formal analysis of security protocols under the perfect cryptography assumption, in which it is assumed that all cryptographic functions are perfect. In this section, we formally analyze the security of the proposed protocol based on Scyther in the $HGWN$ and $FGWN$. The results in Figures 11 and 12 illustrate that the scheme is correct and secure against many adversary models under the Scyther security checks.

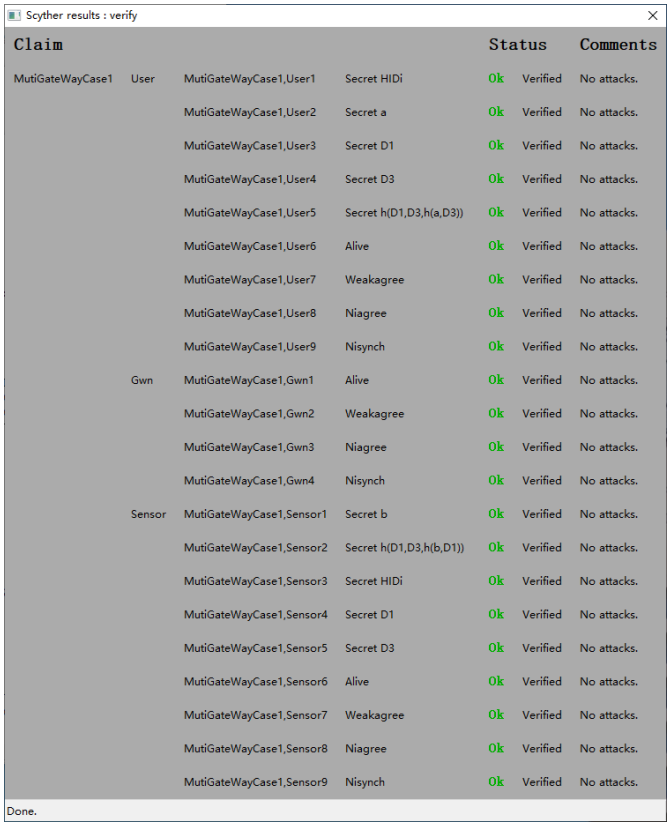

Figure 11. Simulation result in HGWN.

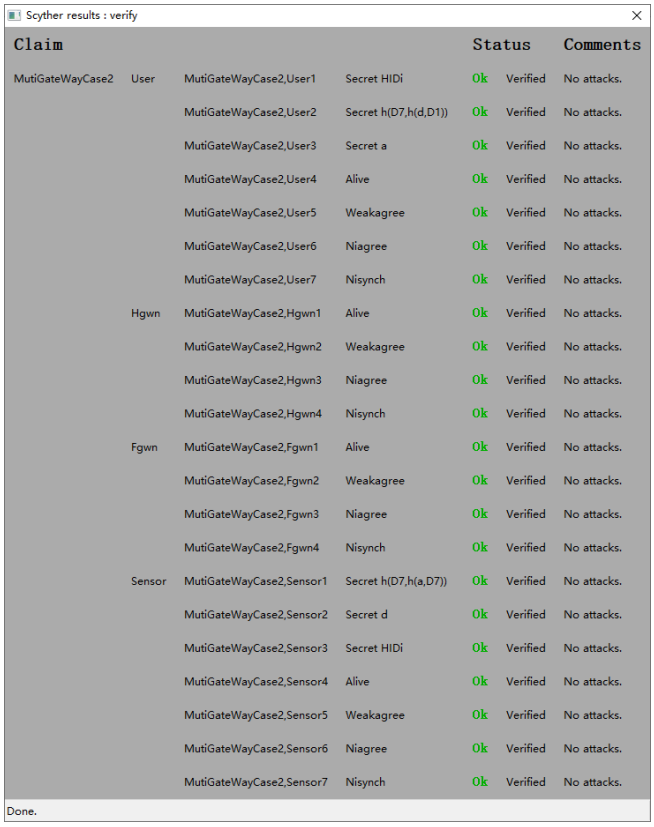

Figure 12. Simulation result in FGWN.

4.3. Informal Security Analysis

4.3.1. Mutual Authentication

In the home region, the $HGWN$ authenticates U_i by relying on $M_3 = h(HID_i||A_h||D_2||M_1||M_2||T_1)$, where D_2 is possessed by U_i and can be recovered by the $HGWN$ from D_1 and its private key k_h. U_i authenticates the $HGWN$ using A_h contained in $M_8 = h(HID_i||A_h||D_1||D_3||M_7||T_4)$, which can only be calculated by U_i and $HGWN$. Any other principals cannot obtain A_h. The $HGWN$ verifies SN_j dependent on $M_6 = h(SID_j||r_{hg}||A_{gs}||D_4||T_3)$, where D_4 is possessed by SN_j and can be recovered by the $HGWN$ from D_3 and k_h. SN_j verifies the $HGWN$ using A_{gs} contained in $M_5 = h(SID_j||r_{hg}||A_{gs}||D_1||T_2)$, which can be calculated by the $HGWN$ and stored in SN_j's memory. U_i can verify the legitimacy of SK using M_7.

In the foreign region, there is a similar process as above. The $HGWN$ authenticates U_i by relying on the secret parameter D_2 only shared by both parties. U_i authenticates the $HGWN$ using A_h contained in $M_8 = h(HID_i||SID_j||A_h||A_f||M_7||T_4)$. The $FGWN$ and $HGWN$ implement mutual authentication using $(b + k_h)K_f$ and $(c + k_f)K_h$, respectively, which are both the secret parameters and can only be computed by themselves and verified by the other party. The $FGWN$ authenticates U_i dependent on $M_{10} = h(HID_i||A_f||D_{2f}||M_9||T_5)$, where D_{2f} is possessed by U_i and can be retrieved by the $FGWN$ from D_1 and its private key k_f. U_i authenticates the $FGWN$ by relying on A_f contained in $M_{15} = h(HID_i||A_f||D_1||D_7||M_{14}||T_8)$, which can be calculated by the $FGWN$ and retrieved by U_i. SN_j verifies the $FGWN$ using A_{fs} contained in $M_{12} = h(SID_j||r_{fg}||A_{fs}||D_1||T_6)$, which can be only calculated by the $FGWN$ using k_f and stored in SN_j's memory. The $FGWN$ verifies SN_j dependent on $M_{13} = h(SID_j||r_{fg}||A_{fs}||D_8||T_7)$, where D_8 is possessed by SN_j and can be retrieved by the $FGWN$ from D_7 and k_f. U_i can verify the legitimacy of SK using M_{14}.

4.3.2. Session Key Agreement

$SK = h(D_1||D_3||bD_1) = h(D_1||D_3||aD_3) = h(aP||bP||abP)$ is established between U_i and SN_j in the home region. Similarly, in the foreign region, U_i and SN_j share a common session key $SK = h(D_7||dD_1) = h(D_7||aD_7) = h(dP||adP)$. The established SK can be used for subsequent communication between U_i and SN_j.

4.3.3. Forward and Backward Secrecy

Forward secrecy is used to guarantee that previously established session keys remain secure in the event that the long-term private keys are compromised. Identically, backward secrecy affords the guarantee that a session key that will be established in the future remains secure even if the long-term private keys are compromised.

The proposed protocol uses the $ECDHP$ to achieve forward and backward secrecy. In the home region, U_i and SN_j share a common session key $SK = h(aP||bP||abP)$, which is related to the random numbers a and b generated by U_i and SN_j, respectively. In the foreign region, U_i and SN_j share a common session key $SK = h(dP||adP)$, which is related to the random numbers a and d generated by U_i and SN_j, respectively. If all the long-term private keys of U_i, $HGWN$, $FGWN$, and SN_j are compromised by an adversary, since the adversary has to resolve the intractable $ECDHP$ to get abP or adP from aP, bP, or aP, dP, respectively, the previous or future session key is still secure. Consequently, forward and backward secrecy can be guaranteed.

4.3.4. User Anonymity and Untraceability

In the proposed protocol, the real identity ID_i cannot be acquired by the adversary from the interaction messages. In the home region, there is only the legitimate gateway node who, in possession of private key k_h, can calculate D_2 to recover U_i's pseudonym HID_i and sensor's identity SID_j. Simultaneously, considering the one-way nature of the hash function, it is difficult for the adversary to acquire HID_i from M_3, M_8 and SID_j from M_5, M_6, respectively. In the foreign region, the adversary without gateway node's private

key cannot compute D_2 to recover HID_i. Likewise, considering the one-way nature of hash function, the adversary is unable to get HID_i from M_3, M_8, M_{10}, M_{15}. As a result, user anonymity can be achieved. In addition, because of the login request message being updated at each session round, the adversary is unable to trace a specific user. Therefore, the user's untraceability is guaranteed.

4.3.5. Illegal Login Detection

A user needs to input their identity, password, and biometric information to complete login, and if the terminal declines this session, at least one of these three items is incorrect. In our protocol, when the incoming information is invalid, the identification parameter B_i cannot be recovered correctly, which leads to the login request being aborted by the terminal. This mechanism guarantees the system can check illegal login requests quickly.

4.3.6. Stolen Smart Card Attack

The secret parameters $\{A_i, B_i, C_i, \theta_i\}$ are stored in U_i's smart card, where $A_i = h(HID_i||k_h||r_h) \oplus HID_i$, $B_i = h(HID_i||HPW_i||r_h)$, $C_i = HID_i \oplus r_h$, and θ_i is generated by $Gen(BIO_i)$. If U_i's smart card is lost and obtained by the adversary, then the adversary can get $\{A_i, B_i, C_i, \theta_i\}$, but they are still unable to acquire the correct identity, password, and biometric key data. The adversary cannot compute a correct HID_i through C_i without r_h. The biometric key data σ_i also cannot be recovered correctly without a real BIO_i. Furthermore, even in this case, there is no chance for an adversary to get the password. As a result, the login request message M_1, M_2, M_3 cannot be figured out without the correct HID_i. Our protocol can be resistant to stolen smart card attack.

4.3.7. Replay Attack

The timestamp mechanism is used to guarantee the freshness of transmitted messages in our scheme. When the message is exchanged, the node first checks whether the time difference between the received timestamp and its own timestamp is within the acceptable maximum delay allowed by the system. Expired messages will be rejected. As a result, the protocol is capable of defending against replay attack.

4.3.8. Privileged Insider Attack

During the registration phase, user transmits $\{HID_i, HPW_i\}$ to the $HGWN$ via a secure channel. It is assumed that an internal malicious privileged node who executes privileged insider attack in order to get user's password PW_i after getting $\{HID_i, HPW_i\}$. However, the obtained values are hash values consisting of password and biometric key data. Considering the one-way nature of the hash function, it is intractable for the privileged node to extract PW_i from HPW_i. Therefore, our protocol can be resistant to privileged insider attack.

4.3.9. Desynchronization Attack

In the proposed protocol, the user does not store the same secret values with the gateway node. All participants in the protocol are not required to update any information when a session is accomplished. Accordingly, the protocol can resist a desynchronization attack.

4.3.10. Impersonation Attack

In our protocol, in order to forge a user, a valid login request $\{M_1, M_2, M_3, D_1, T_1\}$ is necessary. Nevertheless, the adversary has no capacity to figure out the true M_1, M_2, M_3, D_1 without the correct HID_i, SID_j, A_h, D_2. As a result, the adversary fails to impersonate a legitimate user.

In addition, when the fake gateway node receives the correct login request, it cannot retrieve the true D_2 without the real private key. Therefore, the adversary is also unable to impersonate a legitimate gateway node.

Moreover, if the adversary wants to forge a sensor node, they need to recover r_{hg} and generate M_5, M_6, which all depend on A_{gs} that is only computed by the $HGWN$ and stored in the sensor's memory. Consequently, this scheme is protected against a sensor impersonation attack.

5. Performance and Security Comparison

In order to illustrate the balance between the security and usability of our protocol, the comparative consequences of the security and overhead of our scheme with other associated schemes are as follows, where Case-1 and Case-2 represent the protocol designed in the home region and the foreign region, respectively. According to [17,29–31], all operations were implemented in MATLAB on a four-core, 3.2 GHz computer with 8 GB of memory.

5.1. Security Features Comparison

The statistics of the security attributes that each scheme can satisfy are summarized in Table 2, where ✓ represents that this literature can satisfy this corresponding security attribute in Table 2, whereas × represents that it cannot achieve. All the indicators listed in Table 2 were achieved by our scheme. Moreover, none of the studies in the literature [13,17–20] has the capability to achieve forward and backward secrecy. However, the implementation of ECC in our scheme enables ours to accomplish forward and backward secrecy.

Table 2. Security comparison.

Security Properties	**[13]**	**[17]**	**[18]**	**[19]**	**[20]**	**Ours**
Mutual authentication	×	✓	✓	✓	✓	✓
Session key agreement	✓	✓	✓	×	×	✓
Forward and backward secrecy	×	×	×	×	×	✓
User anonymity	✓	×	×	×	×	✓
Untraceability property	×	×	✓	×	×	✓
Illegal login detection	×	✓	✓	×	✓	✓
Stolen smart card attack	×	×	✓	✓	✓	✓
Replay attack	✓	✓	✓	✓	✓	✓
Insider attack	✓	✓	✓	×	✓	✓
Desynchronization attack	×	×	✓	✓	×	✓
Impersonation attack	×	×	✓	×	✓	✓

5.2. Communication Cost Comparison

In order to calculate the communication cost, we assumed that the identity, random number, hash digest, ECC point, and timestamp were 160 bits, 160 bits, 160 bits, 320 bits, and 32 bits, respectively. Additionally, the symmetric encryption/decryption using AES-128 required 128 bits for a 128-bit plaintext block. We evaluated the communication overhead between our protocol and other relevant protocols [13,17–20] during the login and authentication phases according to the overall quantity of transmitted messages. Table 3 shows the comparison results. Compared with [19], the transmitted number of messages was identical to our scheme, and there were similar communications costs as ours, but our scheme met more security attributes. As we can see, in order to compare with previous protocols [13,17–20], we chose SHA-1 [32] as the hash function. However, to achieve more security, we recommend using SHA-256 [32] as the hash function.

Table 3. Communication cost comparison.

Scheme		Number of Messages	Communication Cost (bits)
[13]	Case-1	2	1504
[17]	Case-1	4	2528
	Case-2	5	3008
[18]	Case-1	3	2784
	Case-2	6	4704
[19]	Case-1	4	2688
	Case-2	8	4480
[20]	Case-1	4	2368
	Case-2	7	3904
Ours	Case-1	4	2848
	Case-2	8	4416

5.3. Computation Cost Comparison

Table 4 lists the approximate required computational time of various cryptographic operations, which were used as a comparative standard. Table 5 compares the computational overhead of our scheme and other relevant schemes during the login, authentication, and key agreement phases. The total cost of the proposed scheme increased slightly. Nevertheless, most of the cost was calculated on the gateway side with strong computational power rather than the resource-limited sensor side. Accordingly, integrated with both security and communication cost, our protocol was relatively secure with an acceptable overhead.

Table 4. Execution time of various cryptographic operations.

Symbol	Description	Approximate Computation Time (s)
T_h	Hash function	0.00032
T_{ecm}	ECC point multiplication	0.0171
T_{eca}	ECC point addition	0.0044
T_{sym}	Symmetric encryption/decryption	0.0056
T_{fe}	Fuzzy extractor function	0.0171

Table 5. Computational cost comparison.

Protocols		User	HGWN	FGWN	Sensor	Total (s)
[13]	Case-1	$4T_h + 2T_{ecm} + 1T_{eca}$	$4T_h + 6T_{ecm} + 3T_{eca}$	-	$3T_h + 2T_{ecm} + 2T_{eca}$	0.20092
[17]	Case-1	$7T_h$	$8T_h$	-	$5T_h$	0.00640
	Case-2	$8T_h$	$1T_h$	$7T_h$	$5T_h$	0.00672
[18]	Case-1	$9T_h + 1T_{fe} + 1T_{sym}$	$5T_h + 2T_{sym}$	-	$3T_h + 1T_{sym}$	0.04494
	Case-2	$10T_h + 1T_{fe} + 2T_{sym}$	0	$5T_h + 2T_{sym}$	$4T_h + 1T_{sym}$	0.05118
[19]	Case-1	$9T_h$	$11T_h$	-	$4T_h$	0.00768
	Case-2	$11T_h$	$7T_h$	$7T_h$	$4T_h$	0.00928
[20]	Case-1	$10T_h$	$14T_h$	-	$7T_h$	0.00992
	Case-2	$14T_h$	$6T_h$	$17T_h$	$6T_h$	0.01376
Ours	Case-1	$9T_h + 1T_{fe} + 3T_{ecm}$	$8T_h + 2T_{ecm}$	-	$5T_h + 3T_{ecm}$	0.16094
	Case-2	$12T_h + 1T_{fe} + 4T_{ecm}$	$8T_h + 6T_{ecm} + 2T_{eca}$	$10T_h + 7T_{ecm} + 2T_{eca}$	$5T_h + 3T_{ecm}$	0.38780

6. Conclusions

In this paper, we designed an authentication protocol based on ECC using three factors, applied to the IIoT environment. The proposed scheme was appropriate for single-gateway scenarios, and we also extended it to multigateway scenarios. Furthermore, forward and backward secrecy was realized in our scheme utilizing the intractable ECDHP. The

formal security analysis under the ROM indicated that the proposed protocol was able to satisfy semantic security. We simulated our scheme using the formal verification tool Scyther, and the result showed that our scheme was secure. The informal security analysis proved our protocol was capable of satisfying most common security properties. Finally, compared with other representative protocols, the comparative results of security attributes, communication, and computation cost in Tables 2, 3 and 5 clearly showed that our protocols could achieve many security attributes at a reasonable computation cost.

Author Contributions: Conceptualization, X.Z. and D.L.; methodology, X.Z. and D.L.; software, D.L.; validation, X.Z. and D.L.; formal analysis, X.Z. and D.L.; investigation, X.Z. and D.L.; resources, X.Z., D.L., and H.L.; writing—original draft preparation, X.Z. and D.L.; writing—review and editing, X.Z., D.L., and H.L.; supervision, X.Z. and H.L.; project administration, X.Z. and H.L.; funding acquisition, X.Z. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant 61732022, the Shaanxi Innovation Team Project under grant 2018TD-007, and the Natural Science Foundation of Shaanxi Province under grant 2019ZDLGY12-09.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

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

Abbreviations

The following abbreviations are used in this manuscript:

IIoT	Industrial Internet of things
WSNs	Wireless sensor networks
XOR	Exclusive OR
ECC	Elliptic curve cryptography
ECDHP	Elliptic curve Diffie–Hellman problem
ECDLP	Elliptic curve discrete logarithm problem
AES	Advanced Encryption Standard
HGWN	Home gateway node
FGWN	Foreign gateway node
ROM	Random oracle model
AKA	Authentication and key agreement
SHA-1	Secure Hash Standard 1
SHA-256	Secure Hash Standard 256

References

1. Farag, H.M.; Österberg, P.; Gidlund, M. Congestion Detection and Control for 6TiSCH Networks in IIoT Applications. In Proceedings of the 2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, 7–11 June 2020; pp. 1–6.
2. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
3. Far, H.A.N.; Bayat, M.; Das, A.K.; Fotouhi, M.; Pournaghi, S.M.; Doostari, M. LAPTAS: lightweight anonymous privacy-preserving three-factor authentication scheme for WSN-based IIoT. *Wirel. Netw.* **2021**, *27*, 1389–1412.
4. Choudhary, K.; Gaba, G.S.; Butun, I.; Kumar, P. MAKE-IT—A Lightweight Mutual Authentication and Key Exchange Protocol for Industrial Internet of Things. *Sensors* **2020**, *20*, 5166. [CrossRef] [PubMed]
5. Ma, C.; Wang, D.; Zhao, S. Security flaws in two improved remote user authentication schemes using smart cards. *Int. J. Commun. Syst.* **2014**, *27*, 2215–2227. [CrossRef]
6. Sun, D. Security and Privacy Analysis of Vinoth et al.'s Authenticated Key Agreement Scheme for Industrial IoT. *Symmetry* **2021**, *13*, 1952. [CrossRef]
7. Kumari, S.; Khan, M.K.; Atiquzzaman, M. User authentication schemes for wireless sensor networks: A review. *Ad Hoc Netw.* **2015**, *27*, 159–194. [CrossRef]

8. Das, M.L. Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1086–1090. [CrossRef]
9. Nyang, D.; Lee, M. Improvement of Das's Two-Factor Authentication Protocol in Wireless Sensor Networks. Cryptology ePrint Archive. 2009. Available online: https://eprint.iacr.org/2009/631 (accessed on 1 October 2022.)
10. Vaidya, B.; Makrakis, D.; Mouftah, H.T. Improved two-factor user authentication in wireless sensor networks. In Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, Niagara Falls, ON, Canada, 11–13 October 2010; pp. 600–606.
11. He, D.; Gao, Y.; Chan, S.; Chen, C.; Bu, J. An Enhanced Two-factor User Authentication Scheme in Wireless Sensor Networks. *Ad Hoc Sens. Wirel. Netw.* **2010**, *10*, 361–371.
12. Turkanovic, M.; Brumen, B.; Hölbl, M. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Netw.* **2014**, *20*, 96–112. [CrossRef]
13. Yeh, H.; Chen, T.; Liu, P.; Kim, T.; Wei, H. A Secured Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography. *Sensors* **2011**, *11*, 4767–4779. [CrossRef]
14. Shi, W.; Gong, P. A New User Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 730831 . [CrossRef]
15. Chang, C.; Le, H. A Provably Secure, Efficient, and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 357–366. [CrossRef]
16. Li, X.; Peng, J.; Niu, J.; Wu, F.; Liao, J.; Choo, K.R. A Robust and Energy Efficient Authentication Protocol for Industrial Internet of Things. *IEEE Internet Things J.* **2018**, *5*, 1606–1615. [CrossRef]
17. Amin, R.; Biswas, G.P. A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. *Ad Hoc Netw.* **2016**, *36*, 58–80. [CrossRef]
18. Das, A.K.; Sutrala, A.K.; Kumari, S.; Odelu, V.; Wazid, M.; Li, X. An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks. *Secur. Commun. Netw.* **2016**, *9*, 2070–2092. [CrossRef]
19. Wu, F.; Xu, L.; Kumari, S.; Li, X.; Shen, J.; Choo, K.R.; Wazid, M.; Das, A.K. An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in IoT deployment. *J. Netw. Comput. Appl.* **2017**, *89*, 72–85. [CrossRef]
20. Srinivas, J.; Mukhopadhyay, S.; Mishra, D. Secure and efficient user authentication scheme for multi-gateway wireless sensor networks. *Ad Hoc Netw.* **2017**, *54*, 147–169. [CrossRef]
21. Wang, D.; Li, W.; Wang, P. Measuring Two-Factor Authentication Schemes for Real-Time Data Access in Industrial Wireless Sensor Networks. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4081–4092. [CrossRef]
22. Bellare, M.; Rogaway, P. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS'93, Fairfax, VA, USA, 3–5 November 1993; pp. 62–73.
23. Cremers, C.J.F. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In Proceedings of the 20th International Conference, CAV 2008, Princeton, NJ, USA, 7–14 July 2008; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5123, pp. 414–418.
24. Koblitz, N. Elliptic Curve Cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
25. Miller, V.S. Use of Elliptic Curves in Cryptography. In Proceedings of the Advances in Cryptology—CRYPTO '85, Santa Barbara, CA, USA, 18–22 August 1985; Lecture Notes in Computer Science; Williams, H.C., Ed.; Springer: Berlin/Heidelberg, Germany, 1985; Volume 218, pp. 417–426.
26. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–207. [CrossRef]
27. Dodis, Y.; Reyzin, L.; Smith, A.D. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In Proceedings of the Advances in Cryptology—EUROCRYPT, Interlaken, Switzerland, 2–6 May 2004; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 523–540.
28. Canetti, R.; Krawczyk, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Proceedings of the EuroCrypt, Innsbruck, Austria, 6–10 May 2001; Pfitzmann, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045, pp. 453–474.
29. Srinivas, J.; Das, A.K.; Kumar, N.; Rodrigues, J.J.P.C. Cloud Centric Authentication for Wearable Healthcare Monitoring System. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 942–956. [CrossRef]
30. Challa, S.; Das, A.K.; Odelu, V.; Kumar, N.; Kumari, S.; Khan, M.K.; Vasilakos, A.V. An efficient ECC-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks. *Comput. Electr. Eng.* **2018**, *69*, 534–554. [CrossRef]
31. Lee, C.; Chen, C.; Wu, P.; Chen, T. Three-factor control protocol based on elliptic curve cryptosystem for universal serial bus mass storage devices. *IET Comput. Digit. Tech.* **2013**, *7*, 48–56. [CrossRef]
32. Dang, Q.H. Secure hash standard. In *US Doc/NIST FIPS Publication 180-4*; NIST: Gaithersburg, MD, USA, 2015.