A Time-sensitive Token-Based Anonymous Authentication and Dynamic Group Key Agreement Scheme for Industry 5.0

Zisang Xu, Wei Liang, Kuan-Ching Li, Jianbo Xu, Albert Y. Zomaya, Jixin Zhang

***Abstract*—In Industry 5.0, the massive number of Internet of Things (IoT) devices have increasing demands for group communication with high communication efficiency and low energy consumption. However, group communication meets continuously increasing security risk challenges. Existing authentication and group key agreement schemes have encountered many problems, such as lack of anonymity and untraceability. In this investigation, we propose an anonymous authentication and dynamic group key agreement scheme based on the Blockchain and token mechanism, where each group member can apply for a time-sensitive token during the first authentication and only needs to check the validity of the token in the subsequent authentication, reducing the computational and transmission costs considerably. The verification on the security of the proposed scheme is tackled through mathematical analysis and validated using ProVerif, and comparisons with existing schemes demonstrate that the proposed scheme reduces the security risks and each group member's energy consumption.**

***Index Terms*—anonymous authentication, Blockchain, cryptography, Industry 5.0, group key agreement.**

I. Introduction

THE Internet of Things (IoT) or Cyber-Physical Systems (CPS) [1] is one of the key components of Industry 5.0, where nearly all information collected from a physical space during production is sent to cyberspace through it [2]. To reduce the communication energy consumption of IoT devices, an increasing number of IoT applications have been supplied to higher demands for group communication. In group communication, devices typically broadcast information rather than point-to-point communication, improving communication efficiency and reducing energy consumption. That is, group communication permits the exchange of messages with minimal resources [3]. However, it can also provide increased targets to adversaries, thereby substantially increasing risks on the security [4].

To achieve the aims of hosting secure and reliable group communication, transmitted messages must be encrypted with a group key, and only group members within the same group key can decrypt such a message. Therefore, a group key is essential to ensuring group communication security, and thus, reliable and secure authentication and group key agreement scheme are crucial.

Recently, there have proposed the design of various authentication and group key agreement schemes for group communication. Unfortunately, most of these schemes still show issues, such as:

- *Anonymity and untraceability*: A group key agreement scheme generally requires vast resources to ensure anonymity and untraceability, so most of the schemes select to disclose the identity of all group members, such as [5], [6].
- *Forward and backward secrecy*: In dynamic groups, when a member joins the group, it receives specific secret parameters about the group. When a group member leaves the group, the group member cannot be forced to delete all the previously obtained hidden parameters. Therefore, some schemes cannot ensure the forward or backward secrecy of the group key in a dynamic group, such as [7].
- *Lightweight*: Some schemes remain inadequately lightweight, such as requiring large computation or communication costs for pairwise mutual authentication among group members [7], [8].

To solve the abovementioned problems in the era of Industry 5.0, we design a Token-based Anonymous Authentication and dynamic Group Key Agreement (TAAGKA) scheme based on the Blockchain with the following characteristics:

- In the proposed scheme, each group member can apply for a time-sensitive token. When authenticating group members with tokens, only a small computation cost is required to complete the authentication process.
- With the help of the token mechanism, our scheme need not consume vast resources to ensure the anonymity and untraceability of the group members.

This work was supported in part by National Natural Science Foundation of China under Grant No. 61872138, 62002106, and 62072170, in part by the Fundamental Research Funds for the Central Universities (No. 531118010527), in part by the Key Research and Development Program in Hunan Province (Grant 2022GK2015), in part by the Hunan Provincial Natural Science Foundation of China (2021JJ30341), and in part by the Research Foundation of Education Bureau of Hunan Province, China (Grant 2020C0080). *(Corresponding author: Wei Liang.)*

Zisang Xu is with the Computer and Communication Engineer Institute, Changsha University of Science and Technology, Changsha, 410114, China (e-mail: xzsszx111@csust.edu.cn).

Wei Liang and Jianbo Xu are with the School of Computer science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan 411201, China (e-mail: weiliang99@hnu.edu.cn; jbxu@hnust.edu.cn).

Kuan-Ching Li is with the Department of Computer Science and Information Engineering, Providence University, Taichung 43301, Taiwan (e-mail: kuancli@pu.edu.tw).

Albert Y. Zomaya is with the School of Information Technologies, The University of Sydney, Sydney, New South Wales Australia 2006 (e-mail: albert.zomaya@sydney.edu.au).

Jixin Zhang is with the School of Computer Science, Hubei University of Technology, Wuhan, 430068, China (e-mail: zhangjx@hbut.edu.cn).

- The proposed scheme can guarantee forward and backward secrecy when a group member joins or leaves the group.

The remaining of this article is organized as follows. First, section II presents a survey of related works, Section III introduces the network model, threat model, token mechanism, and TAAGKA scheme, while Section IV depicts the security and correctness of the proposed scheme. Then, section V evaluated the scheme's performance and discussions of industrial applications, and finally, concluding remarks and aims for future work are depicted in Section VI.

II. Related Work

Industry 5.0 is a new concept proposed in recent years. At present, the literature for Industry 5.0 mainly focuses on discussing its development trend and future, and there are few works in the literature on the security of Industry 5.0. However, as an indispensable part of Industry 5.0, the security of the IoT is bound to be closely related to the security of Industry 5.0 [2], [9]. Therefore, in this section, we mainly discuss authentication and group key agreement schemes for the IoT. The abovementioned existing schemes are classified into the following three types.

The Centralized Group Key Agreement (CGKA) scheme typically consists of only one central node responsible for managing all the keys of an entire group and group communication tasks. As a traditional group key agreement scheme, the CGKA scheme usually requires vast computing and communication resources and faces single node failure. Although, for example, Islam et al. [10] proposed a CGKA scheme for the Internet of Vehicles (IoV) in 2018 to improve the efficiency of authentication, in general, the CGKA scheme has faded out of the research field of most researchers.

In a Multi-Center Group Key Agreement (MCGKA) scheme, group members are divided into several subgroups, and each subgroup has a group controller whose function is similar to that of the central node in a CGKA scheme. In the MCGKA scheme proposed by Gupta et al. [5], all group members need to complete authentication with the group controller individually before they can perform group key negotiation, which leads to a lot of computing time. In the certificateless authenticated group key agreement protocol proposed by Mandal et al. [6], powerful nodes are usually regarded as group controllers and manage a group composed of many low-power mobile nodes. In the scheme proposed by Xu et al. [11] and Liu et al. [12], the group controller will perform batch authentication on the group members and get rid of the dependence on the group key manager through the designed key distribution mechanism. In the scheme proposed by Naresh et al. [13], group members are divided into a fixed number of subgroups, and the last group member is designated as the group controller. In general, each subgroup in the MCGKA scheme is usually independent of the other. However, such schemes typically encounter key distribution efficiency and management issues as well as cross-subgroup authentication.

In a Distributed Group Key Agreement (DGKA) scheme, all group members are equal, and no group controller exists. Based on the signature scheme of Gap DiffieHellman groups, the scheme proposed by Wang et al. [14] can guarantee the anonymity of each group member. The scheme proposed by Zhang et al. [8] can achieve key self-certification and cross-domain authentication. Meanwhile, the scheme proposed by Kavitha et al. [15] was designed for the IoT and uses the ElGamal algorithm and hyperelliptic curve digital signature. In the scheme developed by Alphonse and Reddy [16], all group members form a binary tree structure. Before negotiating the group key, all the group members must wait until the last root node authentication is completed. In the scheme proposed by Zheng et al. [7], before the group key is negotiated, each group member must mutually authenticate only its left and right neighbors once, which significantly reduces the computation and communication costs. Based on the chaotic mapping, Lee and Chen [17] proposed two schemes. The first one lacks forward secrecy, though the computational cost is meager; the second has perfect forward secrecy, but unfortunately, the computational costs are high. Since there is no group controller, the DGKA scheme requires to consume a large amount of computation or communication cycles and resources to complete the pairwise mutual authentication between group members.

III. Design of TAAGKA Scheme

In this section, we first introduce the network model and threat model of our scheme. Next, we present the token mechanism we use. Finally, all phases in the TAAGKA scheme and an algorithm for selecting the authentication phase are detailed.

A. Network Model and Threat Models

1) Network Model: The network model of our scheme comprises two main components, namely, Device (DE) and Private Key Generator (PKG), which are illustrated in Fig. 1. The DEs are general nodes, equal, configured with specific computing resources, and have mobile capabilities. Each PKG is similar to a group controller responsible for key generation, distribution, management, and group communication tasks. Our network model consists of multiple PKGs, generally rich in various resources, and wired communication is used between them. Each PKG may manage one or more groups, and several DEs exist in each group. Due to the mobility of DE, each group is dynamic, which means that DE may join or leave a group at any time.

To ensure the consistency of the authentication parameters in different PKGs, we consider that all the PKGs form a blockchain network and jointly maintain a ledger that stores the DE authentication parameters, which can also effectively solve the problem of cross-group authentication. Furthermore, since all nodes in the blockchain node are known, it is more appropriate to use a private chain or a consortium chain. Therefore, we use the Proof-of-Stake (PoS) consensus mechanism to create new blocks, such as Ouroboros, a provably secure PoS protocol [18], instead of the Proof-of-Work mechanism [19], which makes our scheme consume fewer resources and require less computing time when creating new blocks.

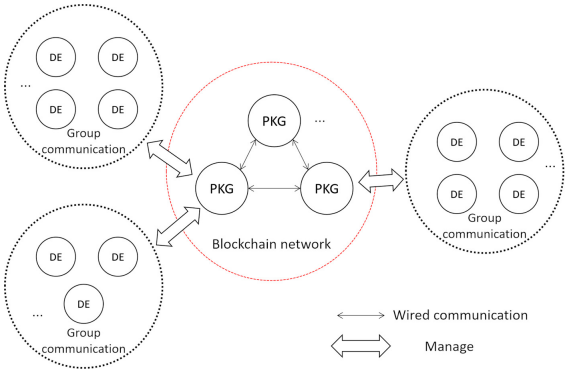


Fig. 1. The network model used in our scheme.

2) Threat Models: The threat models used in this scheme are listed below.

- All PKGs are considered as semi-trusted nodes, which means that PKG will faithfully execute the scheme but will interact with other nodes as much as possible to obtain private information and will not conspire with other PKGs.
- An adversary can intercept exchanged messages over unsecured channels. In addition, he/she can inject new messages into the network and replace, replay, or modify previously intercepted messages.
- An adversary can capture any number of DEs, and once a DE is captured, the adversary can extract relevant secret parameters from the DE's memory.

B. Token Mechanism

Before the TAAGKA scheme is described in detail, the token mechanism used in the proposed scheme must first be introduced. Like its design, each DE can apply for a time-sensitive token from the corresponding PKG at any time, and the token is valid only within the authorized time. Thus, the computation cost can be substantially reduced when re-authenticating with the same PKG once a token is obtained.

We divide the time zones into several successive time blocks, and each time block contains several consecutive time slots. The time zone is the maximum time period during which the token can be used usually. When the system running time exceeds the time zone, all the tokens will become invalid. The parameter z represents the maximum number of time slots in one time block, and the time block is to set an appropriate z to strike a balance between security and the computational cost of the verification token. Therefore, the time zone, time block, and time slot need to be carefully defined according to the security and performance requirements of the actual application scenario. To facilitate the illustration and description of our ideas, we assume that in our scheme, the time zone is one month, the time block is one day, and the time slot is one hour. Thus, the value of z is 24.

C. The TAAGKA Scheme

Table I lists the definitions of the symbols used in our scheme. In our scheme, a list L containing the parameters required to verify each DE's token in each PKG. These parameters in L are classified according to GID and sorted according to IDD. All GIDs are public. As group members join or leave, the parameters in L change frequently. Therefore, we consider not storing L in the Blockchain to reduce the computation cost. We assume the existence of $\mathrm{DE_i}$ ($1 \le i \le n$), $\mathrm{PKG_j}$ ($1 \le j \le m$), and GID_k ($1 \le k \le u$) in the system, where n, m, and u are the number of DEs, PKGs, and groups, respectively. The proposed scheme consists of 7 phases, and the details are as follows.

TABLE I
SYMBOLS USED IN OUR SCHEME.

Symbol	Description
TID_i	Temporary identity of the $\mathrm{DE_i}$
IDD_i	The identity of $\mathrm{DE_i}$
IDP_j	The identity of $\mathrm{PKG_j}$
GID_k	The identity of different groups
s, P_{pub}	Private key and public key of all PKGs
W_i, A_i, B_i	The DE's public key
S_i, a_i, b_i	The DE's private key
ST_i, ET_i	The authorized time slot range $[ST_i, ET_i]$
CT	The current time slot
$date$	Date of applying for the token
$Seeda_i$, $Seedb_i$	Two seeds used to generate TS_{a_i} and TS_{b_i}
TS_{a_i}, TS_{b_i}	Two secret tokens applied by DE_i
AT	Authentication token
T_1 to T_5	Timestamp
t_{new}	The timestamp when the latest information was received
Δt	Maximum communication transmission delay
E_k, D_k	Symmetric encryption and decryption algorithm with key k
TK	Symmetric key
GK	Group key
$h^a(b)$	Perform $a+1$ hash operation on b
$\oplus$	Bitwise XOR operation
(a, b)	Concatenation of data a and data b

1) Initialization Phase: In this phase, the system administrator needs to perform the following steps.

Step I1: Generates a cyclic additive group G_1 of order p, a cyclic multiplicative group G_2 of order p, a generator Q of G_1, and $e : G_1 \times G_1 \to G_2$ is a bilinear map. For points (x, y) belonging to G_1 or G_2, we only focus on x. For example, for $Q(x_Q, y_Q)$ and a private key s', we can obtain $(x_{s'Q}, y_{s'Q})$ by point multiplication operation $s'Q$, and the corresponding public key P'_{pub} is $x_{s'Q}$.

Step I2: Picks a private key s, and computes the corresponding public key through $P_{pub} = sQ$.

Step I3: Generates two random constants n_{1_j}, n_{2_j}, and a unique identity IDP_j for each $\mathrm{PKG_j}$, and stores s, n_{1_j}, n_{2_j}, and IDP_j in the memory of each $\mathrm{PKG_j}$ in a secure environment.

Step I4: Publishes parameters $\{p, G_1, G_2, Q, e, P_{pub}, h(.), E_k, D_k\}$, where $h(.) : \{0,1\}^* \to \{0,1\}^l$ is a hash function.

2) Registration Phase: All DEs must choose a PKG to perform the registration phase and obtain corresponding authentication parameters before entering the system. Here, we assume that a $\mathrm{DE_i}$ elects to register with the $\mathrm{PKG_j}$, then the details of this phase are described as follows.

Step R1: The $\mathrm{PKG_j}$ generates a unique identity IDD_i, computes $W_i = h(IDD_i)$, $S_i = sW_i$, and sends parameters IDD_i, S_i, W_i to $\mathrm{DE_i}$ through a secure channel.

Step R2: The $\mathrm{DE_i}$ stores IDD_i, S_i, and W_i in its memory.

Step R3: The $\mathrm{PKG_j}$ creates a new block containing IDD_i and W_i. All PKGs verify the block through a consensus mechanism and link the block to the Blockchain.

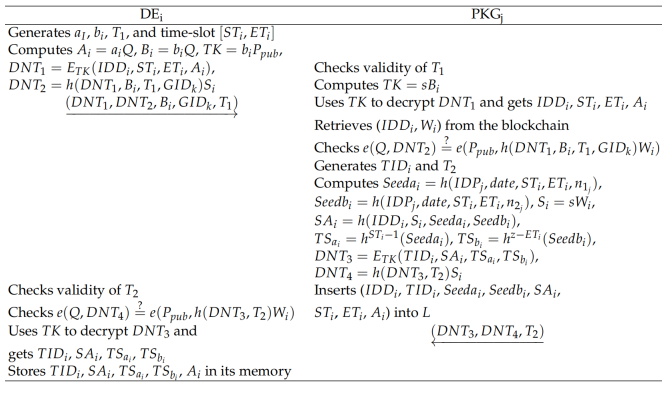


Fig. 2. The authentication without token phase in our scheme.

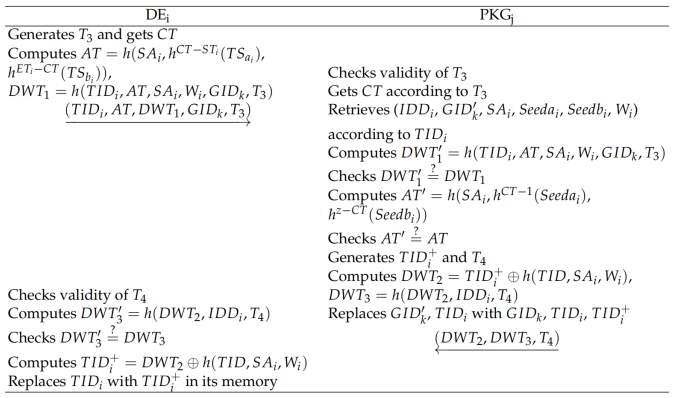


Fig. 3. The authentication with token phase in our scheme.

3) Authentication without Token Phase: When a $\mathrm{DE_i}$ wants to join a new group GID_k across PKG, or the token owned by the $\mathrm{DE_i}$ has expired, it needs to execute this phase to apply for a new token. Fig. 2 shows the authentication without token phase in our scheme. The details of this phase are as follows:

Step NT1: $\mathrm{DE_i} \rightarrow \mathrm{PKG_j}$: ($DNT_1$, DNT_2, B_i, GID_k, T_1), the $\mathrm{DE_i}$ performs the following operations:

- Generates a random a_i, b_i, a timestamp T_1, and determines the time-slot range $[ST_i, ET_i]$ as needed, where $1 \leq ST_i \leq ET_i \leq z$.
- Computes A_i=a_iQ, B_i=b_iQ, TK=b_iP_{pub}, DNT_1=$E_{TK}(IDD_i, ST_i, ET_i, A_i)$, DNT_2=$h(DNT_1, B_i, T_1, GID_k)S_i$, and sends message ($DNT_1$, DNT_2, B_i, GID_k, T_1) to $\mathrm{PKG_j}$.

Step NT2: $\mathrm{PKG_j} \rightarrow \mathrm{DE_i}$: ($DNT_3$, DNT_4, T_2), the $\mathrm{PKG_j}$ performs the following operations:

- Checks that $t_{new} - T_1 < \Delta t$ holds or not. Aborts if the check fails.
- Computes $TK = sB_i$, and uses TK to decrypt DNT_1 to get the parameters IDD_i, ST_i, ET_i, and A_i.
- Retrieves the corresponding tuple (IDD_i, W_i) from the blockchain according to IDD_i.
- Checks whether the condition $e(Q, DNT_2) \stackrel{?}{=} e(P_{pub}, h(DNT_1, B_i, T_1, GID_k)W_i)$ is satisfied. Aborts if the check fails.
- Generates a unique TID_i, a timestamp T_2, and computes $Seeda_i = h(IDP_j, date, ST_i, ET_i, n_{1_j})$, $Seedb_i$=$h(IDP_j, date, ST_i, ET_i, n_{2_j})$, S_i=sW_i, SA_i=$h(IDD_i, S_i, Seeda_i, Seedb_i)$, TS_{a_i}=$h^{ST_i-1}(Seeda_i)$, TS_{b_i}=$h^{z-ET_i}(Seedb_i)$, DNT_3=$E_{TK}(TID_i, SA_i, TS_{a_i}, TS_{b_i})$, DNT_4=$h(DNT_3, T_2)S_i$.
- According to the IDD_i, inserts the tuple (IDD_i, GID_k, TID_i, $Seeda_i$, $Seedb_i$, SA_i, ST_i, ET_i, A_i) into the appropriate position in the list L.
- Sends message (DNT_3, DNT_4, T_2) to $\mathrm{DE_i}$.

Step NT3: The $\mathrm{DE_i}$ performs the following operations:

- Checks that $t_{new} - T_2 < \Delta t$ holds or not. Aborts if the check fails.
- Checks whether the condition $e(Q, DNT_4) \stackrel{?}{=} e(P_{pub}, h(DNT_3, T_2)W_i)$ is satisfied. Aborts if the check fails.
- Uses TK to decrypt DNT_3 to get the parameters TID_i, SA_i, TS_{a_i}, TS_{b_i}.
- Stores the parameters TID_i, SA_i, TS_{a_i}, TS_{b_i}, A_i in its memory.

4) Authentication with Token Phase: If the token owned by the $\mathrm{DE_i}$ is valid, it only needs to perform this phase when authenticating with the $\mathrm{PKG_j}$, as shown in Fig. 3 the authentication with token phase. The detailed description of this phase is as follows.

Step WT1: $\mathrm{DE_i} \rightarrow \mathrm{PKG_j}$: ($TID_i$, AT, DWT_1, GID_k, T_3), the $\mathrm{DE_i}$ performs the following operations:

- Generates a timestamp T_3, and gets the current time-slot CT according to T_3.
- Computes AT=$h(SA_i, h^{CT-ST_i}(TS_{a_i}), h^{ET_i-CT}(TS_{b_i}))$, DWT_1=$h(TID_i, AT, SA_i, W_i, GID_k, T_3)$, and sends $(TID_i, AT, DWT_1, GID_k, T_3)$ to the $\mathrm{PKG_j}$.

Step WT2: $\mathrm{PKG_j} \rightarrow \mathrm{DE_i}$: ($DWT_2$, DWT_3, T_4), the $\mathrm{PKG_j}$ performs the following operations:

- Checks that $t_{new} - T_3 < \Delta t$ holds or not. Aborts if the check fails.
- Gets the current time-slot CT according to T_3, and gets the corresponding tuple (IDD_i, SA_i, $Seeda_i$, $Seedb_i$, W_i) according to TID_i.
- Computes $DWT'_1 = h(TID_i, AT, SA_i, W_i, GID_k, T_3)$, and checks whether the condition $DWT'_1 \stackrel{?}{=} DWT_1$ is satisfied. Aborts if the check fails.
- Computes AT'=$h(SA_i, h^{CT-1}(Seeda_i), h^{z-CT}(Seedb_i))$, and checks whether the condition $AT' \stackrel{?}{=} AT$ is satisfied. Aborts if the check fails.
- Generates a new unique TID_i^+, a timestamp T_4, computes $DWT_2 = TID_i^+ \oplus h(TID, SA_i, W_i)$, $DWT_3 = h(DWT_2, IDD_i, T_4)$, sends message (DWT_2, DWT_3, T_4) to the $\mathrm{DE_i}$.
- Replaces tuples (IDD_i, GID'_k, TID_i, $Seeda_i$, $Seedb_i$, SA_i, ST_i, ET_i, A_i) with tuples (IDD_i, GID_k, TID_i, TID_i^+, $Seeda_i$, $Seedb_i$, SA_i, ST_i, ET_i, A_i) in its memory.

Step WT3: The $\mathrm{DE_i}$ performs the following operations:

- Checks that $t_{new} - T_4 < \Delta t$ holds or not. Aborts if the check fails.

- Computes $DWT_3' = h(DWT_2, IDD_i, T_4)$, and checks whether the condition $DWT_3' \stackrel{?}{=} DWT_3$ is satisfied. Aborts if the check fails.
- Computes $TID_i^+ = DWT_2 \oplus h(TID, SA_i, W_i)$.
- Replaces TID_i with TID_i^+ in its memory.

The PKG will periodically check the validity of each token in L, and delete the parameters related to the invalid token from L. In addition, since there are two different authentication phases in our scheme, we designed Algorithm 1 so that after receiving a message, the PKG can automatically determine which authentication phase should be executed. Since whether it is step WT1 or step NT1, the message received by PKG includes 5 parameters, so we named these parameters PA_1, PA_2, PA_3, PA_4, and PA_5. In Algorithm 1, first of all, no matter which authentication phase the message received by PKG belongs to, PA_5 is a timestamp. Therefore, the validity of PA_5 needs to be checked first. If PA_5 is invalid, the session will be closed. If valid, the PKG searches its own memory for the same TID_i as PA_1. If it does not exist, the PKG executes the authentication without token phase. If the same TID_i exists, DWT_1' will be calculated according to step WT2. If DWT_1' is equal to PA_3, the PKG continues to execute the authentication with token phase, otherwise executes the authentication without token phase.

Algorithm 1 Authentication phase selection of the PKG

```
Receives parameters PA1, PA2, PA3, PA4, PA5;
Checks validity of timestamp PA5;
if PA5 is valid then
    Searches its own memory for the same TIDi as PA1;
    if TIDi exists then
        Computes DWT1' according to step WT2;
        if DWT1' = PA3 then
            Continues to perform the authentication with token phase;
        else
            Executes the authentication without token phase;
        end if
    else
        Executes the authentication without token phase;
    end if
else
    Aborts this session;
end if
```

5) Group Key Generation Phase: After all DEs in the group have completed authentication with the PKG, the group key generation phase can be executed.

Step K1: The $\mathrm{PKG_j}$ generates a timestamp T_5, computes auxiliary parameters $X_1 = e(A_1 - A_n, Q)$, $X_2 = e(A_2 - A_1, Q)$, $\cdots$, $X_n = e(A_n - A_{n-1}, Q)$, HDE_1=$h(TID_1, sW_1, IDD_1, T_5, X_1, X_2, \cdots, X_n)$, HDE_2=$h(TID_1, sW_2, IDD_2, T_5, X_1, X_2, \cdots, X_n)$, $\cdots$, $HDE_n = h(TID_n, sW_n, IDD_n, T_5, X_1, X_2, \cdots, X_n)$, and broadcasts $(< TID_1, X_1, HDE_1 >, < TID_2, X_2, HDE_2 >, \cdots, < TID_n, X_n, HDE_n >, T_5)$.

Step K2: Each DE Checks that $t_{new} - T_5 < \Delta t$ holds or not. Broadcasts an authentication failure message if the check fails. Then the DE computes HDE_i'=$h(TID_i, S_i, IDD_i, T_5, X_1, X_2, \cdots, X_n)$, and checks whether the condition $HDE_i' \stackrel{?}{=} HDE_i$ is satisfied. Broadcasts an error message if the check fails. Finally, the DE computes k=$e(nA_i, Q)X_{i+1}^{n-1}X_{i+2}^{n-2}\cdots X_{i-1}$, and group key $GK = h(k, X_1, X_2, \cdots, X_n)$.

6) DE Join Phase: When a registered $\mathrm{DE_e}$ intends to join a group managed by the $\mathrm{PKG_j}$, it needs to perform the DE join phase. The details of this phase are described as follows.

Step J1: The $\mathrm{DE_e}$ needs to perform the authentication without token phase or the authentication with token phase to pass the authentication.

Step J2: The $\mathrm{PKG_j}$ generates a random number r_{a_i} for each $\mathrm{DE_i}$, computes $PA_{a_i} = r_{a_i}A_i$, $DJ_{1_i} = h(A_i, PA_{a_i})$ and sends (PA_{a_i}, DJ_{1_i}) to each $\mathrm{DE_i}$ including $\mathrm{DE_e}$.

Step J3: Each $\mathrm{DE_i}$ computes $DJ_{1_i}' = h(A_i, PA_{a_i})$, and checks whether the condition $DJ_{1_i}' \stackrel{?}{=} DJ_{1_i}$ is satisfied including $\mathrm{DE_e}$. Aborts if the check fails.

Step J4: Each $\mathrm{DE_i}$ generates a random number r_{b_i}, computes $PA_{b_i} = r_{b_i}A_i$, $A_i^+ = r_{b_i}PA_{a_i}$, $DJ_{2_i} = h(A_i, PA_{b_i})$, and sends (PA_{b_i}, DJ_{2_i}) to the $\mathrm{PKG_j}$. After sending the tuple, each $\mathrm{DE_i}$ will set a maximum waiting time. If the $\mathrm{PKG_j}$ starts to perform the group key generation phase within the maximum waiting time, each $\mathrm{DE_i}$ will replace A_i with A_i^+ in its memory, otherwise it will broadcast an error message.

Step J5: The $\mathrm{PKG_j}$ computes $DJ_{2_i}' = h(A_i, PA_{b_i})$, and checks whether the condition $DJ_{2_i}' \stackrel{?}{=} DJ_{2_i}$ is satisfied. Aborts if the check fails.

Step J6: The $\mathrm{PKG_j}$ computes $A_i^+ = r_{a_i}PA_{b_i}$, replaces A_i with A_i^+ in L. After all A_i in the group have been updated, the group key generation phase is executed.

7) DE Leave Phase: When a $\mathrm{DE_e}$ wants to leave a group managed by the $\mathrm{PKG_j}$, the DE leave phase needs to be performed. The details of this phase are described as follows.

Step L1: The $\mathrm{DE_e}$ sends TID_e and leave request to $\mathrm{PKG_j}$.

Step L2: The $\mathrm{PKG_j}$ deletes all parameters related to $\mathrm{DE_e}$ in L according to TID_e.

Step L3: The $\mathrm{PKG_j}$ generates a random number r_{a_i} for each $\mathrm{DE_i}$ except $\mathrm{DE_e}$, computes $PA_{a_i} = r_{a_i}A_i$, $DJ_{1_i} = h(A_i, PA_{a_i})$ and sends (PA_{a_i}, DJ_{1_i}) to each $\mathrm{DE_i}$.

Step L4: Perform the same process from step J3 to step J6 in DE join phase.

IV. Security Analysis

In this section, we first use mathematics to analyze the correctness of the TAAGKA scheme. Then use the ProVerif tool to verify the security of the TAAGKA scheme. Finally, we describe the informal security analysis of the TAAGKA scheme.

A. Correctness Analysis

Lemma 1. *Given the value* $(TS_{a_i}, TS_{b_i}, SA_i, ST_i, ET_i)$, *for any* CT *that does not satisfy* $1 \leq ST_i \leq CT \leq ET_i \leq z$, *the* $\mathrm{DE_i}$ *is restricted from deriving the token* $AT = h(SA_i, h^{CT-ST_i}(TS_{a_i}), h^{ET_i-CT}(TS_{b_i}))$.

Proof. According to Table I, $h^a(b)$ is perform $a+1$ hash operation on b, which means that a cannot be a negative number. Therefore, the $\mathrm{DE_i}$ can only compute the value $h^{CT-ST_i}(TS_{a_i})$ for $CT \geq ST_i$, and the value $h^{ET_i-CT}(TS_{b_i})$ for $CT \leq ET_i$, which makes it can be easily concluded that only when $1 \leq ST_i \leq CT \leq ET_i \leq z$ is satisfied, the $\mathrm{DE_i}$ can compute a valid token AT. □

Theorem 1. *In the authentication without token phase, the mutual authentication between the* $\mathrm{DE_i}$ *and* $\mathrm{PKG_j}$ *is valid.*

Proof. In the authentication without token phase, the mutual authentication between $\mathrm{DE_i}$ and $\mathrm{PKG_j}$ is realized by verifying $e(Q, DNT_2) \stackrel{?}{=} e(P_{pub}, h(DNT_1, B_i, T_1)W_i)$ and $e(Q, DNT_4) \stackrel{?}{=} e(P_{pub}, h(DNT_3, T_2)W_i)$. Therefore, the correctness is proved as follows:

$$\begin{aligned} e(Q, DNT_2) &= e(Q, h(DNT_1, B_i, T_1, GID_k)S_i) \\ &= e(Q, h(DNT_1, B_i, T_1, GID_k)sW_i) \\ &= e(sQ, h(DNT_1, B_i, T_1, GID_k)W_i) \\ &= e(P_{pub}, h(DNT_1, B_i, T_1, GID_k)W_i). \end{aligned} \tag{1}$$

$$\begin{aligned} e(Q, DNT_4) &= e(Q, h(DNT_3, T_2)S_i) \\ &= e(Q, h(DNT_3, T_2)sW_i) \\ &= e(sQ, h(DNT_3, T_2)W_i) \\ &= e(P_{pub}, h(DNT_3, T_2)W_i). \end{aligned} \tag{2}$$

Due to the Elliptic Curve Discrete Logarithm problem (ECDL), the adversary cannot obtain s from P_{pub} or Q in polynomial time. In addition, since the adversary cannot obtain S_i, he cannot forge a DE by creating valid DNT_2 or DNT_4. Moreover, due to the decisional Diffie-Hellman problem, it is impossible for an adversary to find a parameter $d \in G_1$ that can be verified by $e(Q, DNT_2) = e(P_{pub}, d)$ or $e(Q, DNT_4) = e(P_{pub}, d)$ in polynomial time. □

Theorem 2. *In the authentication with token phase, the* $\mathrm{PKG_j}$ *can verify the token* AT *computed by the* $\mathrm{DE_i}$.

Proof. In the authentication with token phase, the $\mathrm{DE_i}$ gets the token AT through computing AT=$h(SA_i, h^{CT-ST_i}(TS_{a_i}), h^{ET_i-CT}(TS_{b_i}))$, and the $\mathrm{PKG_j}$ gets the AT through computing AT=$h(SA_i, h^{CT-1}(Seeda_i), h^{z-CT}(Seedb_i))$, then,

$$\begin{aligned} AT &= h(SA_i, h^{CT-ST_i}(TS_{a_i}), h^{ET_i-CT}(TS_{b_i})) \\ &= h(SA_i, h^{CT-ST_i+ST_i-1}(Seeda_i), \\ &\quad h^{ET_i-CT+z-ET_i}(Seedb_i)) \\ &= h(SA_i, h^{CT-1}(Seeda_i), h^{z-CT}(Seedb_i)). \end{aligned} \tag{3}$$

According to Lemma 1, only when $1 \leq ST_i \leq CT \leq ET_i \leq z$ is satisfied, the token AT computed by the $\mathrm{DE_i}$ can pass the $\mathrm{PKG_j}$ verification. □

Theorem 3. *In the group key generation phase, all DEs can compute the same group key.*

Proof. All DEs can receive the message ($< TID_1, X_1, HDE_1 >, < TID_2, X_2, HDE_2 >, \cdots, < TID_n, X_n, HDE_n >, T_5$), then,

```
free c:channel.
type QP.

const q: QP.
free Q,W: QP.
free date,GID:bitstring.
free IDD,IDP,n1,n2,s:bitstring [private].

fun pm(QP,bitstring):QP.
equation forall x:bitstring,y:bitstring;
pm(pm(q, x), y) = pm(pm(q, y), x).
fun exp(bitstring, bitstring): bitstring.
fun minus(bitstring, bitstring): bitstring.
fun senc(bitstring, QP): bitstring.
reduc forall m: bitstring, k: QP;
sdec(senc(m,k),k) = m.
fun con(bitstring,bitstring):bitstring.
reduc forall m: bitstring, n: bitstring;
lde(con(m,n)) = m.
reduc forall m: bitstring, n: bitstring;
rde(con(m,n)) = n.
fun h(bitstring):bitstring.
fun QP_to_BS(QP):bitstring [typeConverter].
fun BS_to_QP(bitstring):QP [typeConverter].

query secret s.
query secret IDD.
query secret TK.

let DE(IDD:bitstring,Q:QP,S:QP,
Ppub:QP) =
new a:bitstring;
new b:bitstring;
new T1:bitstring;
new ST:bitstring;
new ET:bitstring;
let A=pm(Q,a) in
let B=pm(Q,b) in
let TK=pm(Ppub,b) in
let DNT1=senc(con(IDD,con(ST,
con(ET,QP_to_BS(A)))),TK) in
let DNT2=pm(S,h(con(DNT1,
con(QP_to_BS(B),con(T1,GID))))) in
out (c,(DNT1,DNT2,B,GID,T1));
in (c,(DNT3:bitstring,
DNT4:bitstring,T2:bitstring));
let DATA1=sdec(DNT3,TK) in 0.

let PKG(IDP:bitstring,n1:bitstring,
n2:bitstring,s:bitstring,date:bitstring,
x:bitstring,z:bitstring,Q:QP,W:QP) =
in (c,(DNT1:bitstring,DNT2:bitstring,
B:QP,GID:bitstring,T1:bitstring));
let TK=pm(B,s) in
let DATA2=sdec(DNT1,TK) in
let IDD=lde(DATA2) in
let ST=lde(rde(DATA2)) in
let ET=lde(rde(rde(DATA2))) in
let A=rde(rde(rde(DATA2))) in
new TID:bitstring;
new T2:bitstring;
let Seeda=h(con(IDP,con(date,
con(ST,con(ET,n1))))) in
let Seedb=h(con(IDP,con(date,
con(ST,con(ET,n2))))) in
let S=pm(W,s) in
let SA=h(con(IDD,con(QP_to_BS(S),
con(Seeda,Seedb)))) in
let TSa=exp(h(Seeda),
minus(ST,x)) in
let TSb=exp(h(Seedb),
minus(z,ET)) in
let DNT3=senc(con(TID,con(SA,
con(TSa,TSb))),TK) in
let DNT4=pm(S,h(con(DNT3,T2))) in
out (c,(DNT3,DNT4,T2)).

process
new x:bitstring;
new z:bitstring;
let Ppub=pm(Q,s) in
let S=pm(W,s) in
(!IDE(IDD,Q,S,Ppub)) | (!PKG(IDP,
n1,n2,s,date,x,z,Q,W))
```

Fig. 4. The simulation code of the authentication without token phase.

$$\begin{aligned} k &= e(nA_i, Q)X_{i+1}^{n-1}X_{i+2}^{n-2}\cdots X_{i-1} \\ &= e(a_iQ, Q)^n X_{i+1}^{n-1}X_{i+2}^{n-2}\cdots X_{i-1} \\ &= e(Q, Q)^{na_i+(n-1)(a_{i+1}-a_i)+(n-2)(a_{i+2}-a_{i+1})+\cdots} \\ &\quad {}^{+(a_{i-1}-a_{i-2})} \\ &= e(Q, Q)^{a_1+a_2+\cdots+a_i}. \end{aligned} \tag{4}$$

Therefore, all DEs can get the same group key GK through $GK = h(k, X_1, X_2, \cdots, X_n)$. □

B. Simulation Based on ProVerif

ProVerif is an automatic cryptographic scheme verification tool [20] and is widely used to prove the security of various schemes. In this section, we use ProVerif to prove the security of the authentication without token phase, authentication with token phase, and the group key generation phase. Fig. 4 demonstrates the simulation code of the authentication without token phase in our scheme. Fig. 5 shows the simulation code of the authentication with token phase and group key generation phase. Since all DEs are equal, adding more DEs in the ProVerif simulation will increase the repeated code without affecting the results. Therefore, we assume that only two DEs exist in the group. The simulation results of the different phases are reported in Fig. 6a and Fig. 6b. The results indicate that an adversary cannot obtain s, IDD, TK, A_i, TID^+, and GK. In addition, all events are executed in order.

C. Informal Security Analysis

1) Anonymity and Untraceability: Among the messages sent during the authentication without token phase and authentication with token phase, only DNT_1, DWT_3, and HDE_i contain the IDD_i information. However, IDD_i in DWT_3 and HDE_i is protected by $h()$. In addition, if an adversary wants to get IDD_i from DNT_1, he/she must get the TK key. However, according to the computational Diffie-Hellman problem, the adversary cannot obtain TK from P_{pub}, B_i, or Q in polynomial time. As for untraceability, all sent messages are not fixed. An adversary can only track a $\mathrm{DE_i}$ through TID_i. However, TID_i is updated after each authentication phase is completed, and an adversary will not be able to predict the new TID_i^+; thus, the untraceability of our scheme is guaranteed.

```
free c:channel.

free CT,Q,GID:bitstring.
free SA,TSa,TSb,ST,ET,S,Seeda,
Seedb:bitstring [private].

fun exp(bitstring, bitstring): bitstring.
fun e(bitstring, bitstring): bitstring.
fun ml(bitstring, bitstring): bitstring.
fun minus(bitstring, bitstring): bitstring.
fun con(bitstring,bitstring):bitstring.
fun h(bitstring):bitstring.
fun xor(bitstring, bitstring): bitstring.
reduc forall m: bitstring, n: bitstring;
rxor(xor(m,n)) = m.

event acc1().
event acc2().
event acc3().
query secret IDD.
query secret A1.
query secret A2.
query secret sTID.
query secret GK.
query event (acc3())==>event (acc2()).
query event (acc1())==>event (acc3()).

let DE(IDD:bitstring,SA:bitstring,
TSa:bitstring,TSb:bitstring,ST:bitstring,
ET:bitstring,W:bitstring,A1:bitstring,
TID:bitstring)=
new T3:bitstring;
let AT=h(con(SA,con(h(exp(TSa,
minus(CT,ST))),h(exp(TSb,
minus(ET,CT)))))) in
let DWT1=h(con(TID,con(AT,
con(SA,con(W,con(GID,T3)))))) in
out (c,(TID,AT,DWT1,GID,T3));
in (c,(DWT2:bitstring,
DWT3:bitstring,T4:bitstring));
let tDWT3=h(con(DWT2,con(IDD,T4))) in
if (tDWT3=DWT3) then event acc1();
let sTID=xor(DWT2,h(con(TID,
con(SA,W)))) in
in (c,(TID:bitstring,X1:bitstring,
HDE1:bitstring,T5:bitstring));
let tHDE1=h(con(TID,con(S,
con(IDD,con(T5,X1))))) in
let k=ml(ml(e(A1,Q),
e(A1,Q)),X1) in
let GK=h(con(k,X1)) in 0.

let PKG(SA:bitstring,Seeda:bitstring,
Seedb:bitstring,W:bitstring,x:bitstring,
z:bitstring,IDD:bitstring,A1:bitstring,
A2:bitstring,S:bitstring) =
in (c,(TID:bitstring,AT:bitstring,
DWT1:bitstring,GID:bitstring,T3:bitstring));
let tDWT1=h(con(TID,con(AT,con(SA,
con(W,T3))))) in
if (tDWT1=DWT1) then event acc2();
let tAT=h(con(SA,con(h(exp(Seeda,
minus(CT,x))),h(exp(Seedb,
minus(z,CT)))))) in
if (tAT=AT) then event acc3();
new sTID:bitstring;
new T4:bitstring;
let DWT2=xor(sTID,h(con(TID,
con(SA,W)))) in
let DWT3=h(con(DWT2,
con(IDD,T4))) in
out(c,(DWT2,DWT3,T4));
new T5:bitstring;
let X1=e(minus(A1,A2),Q) in
let HDE1=h(con(TID,con(S,
con(IDD,con(T5,X1))))) in
out (c,(TID,X1,HDE1,T5)).

process
new x:bitstring;
new z:bitstring;
new W:bitstring;
new TID:bitstring;
new IDD:bitstring;
new A1:bitstring;
new A2:bitstring;
(!DE(IDD,SA,TSa,TSb,ST,ET,
W,A1,TID)) | (!PKG(SA,Seeda,
Seedb,W,x,z,IDD,A1,A2,S))
```

Fig. 5. The simulation code of the authentication with token phase and the group key generation phase.

```
--------------------------------
Verification summary:

Query secret s_1 is true.

Query secret IDD_2, IDD_1 is true.

Query secret TK_1, TK is true.

--------------------------------
```

(a) The authentication without token phase.

```
----------------------------------------
Verification summary:

Query secret IDD_2, IDD_1, IDD is true.

Query secret A1_2, A1_1, A1 is true.

Query secret A2_1, A2 is true.

Query secret sTID_1, sTID is true.

Query secret GK is true.

Query event(acc3) ==> event(acc2) is true.

Query event(acc1) ==> event(acc3) is true.

----------------------------------------
```

(b) The authentication with token phase and the group key generation phase.

Fig. 6. The simulation results of different phase in our scheme.

2) Impersonation Attack: In our scheme, each message contains an authentication parameter, such as DNT_2, DNT_4, DWT_1, DWT_3, and all HDE_i. An adversary must be able to create a valid authentication parameter before performing an impersonation attack. For DNT_2 and DNT_4, an adversary cannot obtain any S_i, whereas for DWT_1, DWT_3, and all HDE_i, an adversary cannot obtain any IDD_i or SA_i. Therefore, it is impossible to perform impersonation attacks on any DE or PKG.

3) Capture Attack: According to the threat model, an adversary can obtain multiple S_i from all the DEs it captures. However, according to the ECDL, an adversary cannot calculate s in polynomial time, which means no new legal DE can be created no matter how many DEs are captured by the adversary. Therefore, the capture attack on DE will not affect the normal operation of the entire system.

4) Replay Attack: Timestamps T_1 to T_5 are used in our scheme to prevent replay attacks. When a receiver receives a message, it will first verify the validity of the timestamp. In addition, each timestamp is included in the authentication parameters, which means that the timestamps cannot be modified.

5) Forward and Backward Secrecy: The symmetric key TK is generated based on the Diffie-Hellman key exchange scheme. As for group key GK, according to Theorem 3, GK can be calculated by getting all A_i in the group. In our scheme, when a DE joins or leaves the group, all the other group members will update A_i. Therefore, the forward and backward secrecy of TK and GK is guaranteed.

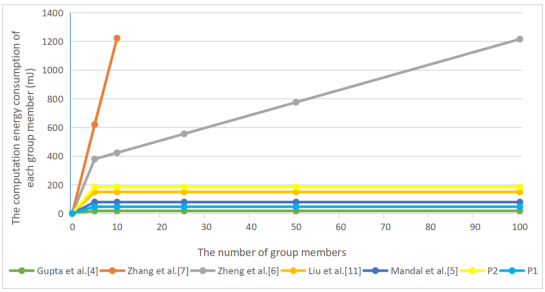


Fig. 7. The computation energy consumption of each group member.

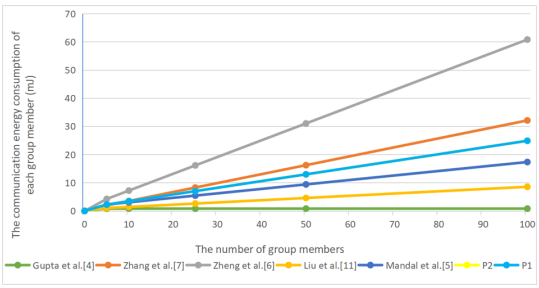


Fig. 8. The communication energy consumption of each group member.

6) Single Node Failure: Since all PKGs in our scheme jointly maintain a ledger that stores all DE-related information based on the Blockchain, once a PKG_j fails, it can be replaced by other working PKGs. Thus, all group members under the jurisdiction of PKG_j only need to perform the authentication without token phase and group key generation phase with the new PKG online to generate a new group key.

V. Performance Analysis and Applications

This section analyzes the performance of the authentication without token phase, authentication with token phase, and group key generation phase of our scheme. Since there are two different authentication phases, we divide the analysis into two parts, namely, P1 and P2. P1 analyzes only the performance of the authentication with token phase and group key generation phase, whereas P2 analyzes only the performance of the authentication without token phase and group key generation phase.

A. Computation and Communication Cost

We use t_h, t_{pa}, t_{pm}, t_{sym}, and t_{bp} to represent the required computing time to execute a hash function operation, point addition operation on the Elliptic Curve Cryptography (ECC), point multiplication operation on the ECC, symmetric encryption or decryption, and a bilinear pairing operation, respectively. As the computation cost of the XOR operation is meager, we ignore this operation. In our scheme, setting different time slots and applying for tokens with different valid durations will affect the computation cost of the DE. Therefore, we use M to represent the valid duration of the

TABLE II
THE COMPARISON OF COMPUTATION COST AND COMMUNICATION COST BETWEEN OUR SCHEME AND OTHER SCHEMES.

	Total computation cost of each group member	Total length of message sent of each group member	Total length of message received of each group member
Gupta et al. [5]	$2t_{pm} + 4t_h + t_{sym} + (n+1)t_{pa}$	$3C$	$4C$
Zhang et al. [8]	$(3n+2)t_{pm} + 2nt_{bp}$	$4C$	$4(n-1)C$
Zheng et al. [7]	$(n+6)t_{pm} + 6t_{bp} + (n+9)t_h + 4t_{sym} + (3n+1)t_{pa}$	$7C + 2T$	$(7C + 2T)(n-1)$
Liu et al. [12]	$t_{pm} + 3t_{bp}$	$3C + T$	$(n+1)C$
Mandal et al. [6]	$9t_{pm} + 10t_h$	$6C + T$	$(2n+5)C + T$
P1	$(7+M)t_h + t_{bp}$	$4C + T$	$(3n+4)C + 2T$
P2	$5t_{pm} + 3t_{bp} + 4t_h + 2t_{sym}$	$4C + T$	$(3n+4)C + 2T$

TABLE III
ENERGY CONSUMPTION FOR COMPUTING AND COMMUNICATION.

Operations	**Energy Consumption**
One symmetric encryption or decryption	0.00217 mJ
One point multiplication operation on ECC	8.8 mJ
One point addition operation on ECC	0.001085 mJ
One hash function operation	0.000108 mJ
One bilinear pairing operation	47 mJ
Transmitting a bit	0.00066 mJ
Receiving a bit	0.00031 mJ

DE application token. M is usually very small. This is due to, if most DEs tend to apply for longer-term tokens, the length of the time slot should be increased to reduce the computation cost. For P1, the total computation cost of each DE is $(7+M)t_h + t_{bp}$, and the total computation cost is $2nt_{bp} + (14+M)nt_h$. For P2, the total computation cost of each DE is $5t_{pm}+3t_{bp}+4t_h+2t_{sym}$, and the total computation cost is $9nt_{pm} + 6nt_{bp} + 12nt_h + 4nt_{sym}$, where n is the number of DEs. As for communication costs. Symbols C and T represent the length of the general parameter and the length of the timestamp, respectively. We assume that C is 256 bits and T is 64 bits. For P1 and P2, the total message length sent by each DE is $4C + T$, the total message length received by each DE is $(3n+4)C + 2T$, the total length of the message to be sent is $9nC + (2n+1)T$, and the total length of the received message is $(3n^2 + 6n)C + 3nT$.

B. Comparisons with Existing Schemes

In this section, we compare our scheme with the scheme proposed by Zhang et al. [8], Gupta et al. [5], Zheng et al. [7], Liu et al. [12], and Mandal et al. [6] in terms of computation cost, communication cost, energy consumption, functionality, and security. Table II summarizes the comparison between our scheme and the schemes as mentioned earlier in terms of computation cost and communication cost. Note that in the scheme of Mandal et al. [6], only the computation cost of the third round of their scheme is counted. Therefore, we re-stated the computation cost of their scheme and shown in Table II. As a result, it is easy to find that P2 has a higher computational cost than P1 from Table II. To facilitate the comparison of computational and communication energy consumption between our scheme and related schemes, we summarize the required energy consumption to perform different operations (such as point multiplication operation on ECC, symmetric encryption or decryption, and others) based on a "StrongARM" microprocessor running at 133 MHz, as shown in Table III through literature [21].

Fig. 7 and Fig. 8 show the computation energy consumption and communication energy consumption, respectively, required by each group member in the different schemes. To not affect the display of the other schemes, we do not fully report the energy consumption of each group member in the scheme proposed by Zhang et al. [8]. Note that in Fig. 8, the lines of P1 and P2 overlap. In Fig. 7, we assume that M is 3. Even if M is 24, that is, the maximum value in our scheme, the computation energy consumption of P1 is still lower than the scheme of Mandal et al. [6] For security and functionality, we show the comparison results of our scheme and existing representative schemes in Table IV. According to Fig. 7, Fig. 8, and Table IV, we can summarize as follows: 1) Our scheme requires very low computational energy consumption. Although the scheme of Gupta et al. [5] requires lower computational energy consumption, the identity of DE in their scheme is public. Therefore, their scheme cannot guarantee the anonymity and untraceability of DE. 2) The communication energy consumption of our scheme is moderate. Although the schemes of Mandal et al. [6] and Liu et al. [12] are lower than ours, the identity of DE in the scheme of Mandal et al. [6] is public. Therefore, the scheme of Mandal et al. [6] lacks anonymity and untraceability. In addition, because the scheme of Liu et al. [12] does not use timestamps or other means to resist replay attacks, their scheme is vulnerable to replay attacks. 3) On the premise of ensuring security and anonymity, our scheme requires the lowest energy consumption for computing and communication. In addition, the scheme of Gupta et al. [5] uses point-to-point communication between the group controllers and group members. Although it can reduce the communication cost, it will also greatly increase the total computing time. Moreover, once the group controller in their scheme fails, the entire subgroup cannot continue to work normally.

C. Industrial Applications

It can be found from Table IV that our scheme can guarantee the anonymity and untraceability of group members, which some existing schemes cannot guarantee. Therefore, the proposed scheme can be used in some Industry 5.0 applications

TABLE IV
THE COMPARISON OF SECURITY AND FUNCTIONALITY BETWEEN OUR SCHEME AND EXISTING REPRESENTATIVE SCHEMES.

	Gupta et al. [5]	Zhang et al. [8]	Zheng et al. [7]	Liu et al. [12]	Mandal et al. [6]	Our Scheme
A1	√	√	√	√	√	√
A2	√	√	√	√	√	√
A3	√	√	√	√	√	√
A4	√	√	√	×	√	√
A5	×	√	√	√	√	√
A6	×	×	×	√	×	√
A7	√	√	×	√	√	√
A8	×	×	×	×	×	√

A1: Providing mutual authentication, A2: Providing group key security, A3: Capture attack, A4: Replay attack, A5: Single node failure, A6: Anonymity and untraceability, A7: Forward and backward secrecy, A8: Token mechanism.

that have high requirements for privacy, such as the industrial network [22] or the IoV. The IoV is mainly composed of vehicles, roadside units, and trusted authorities [23]. A trusted authority can be regarded as a PKG, which manages several roadside units. The vehicles within the communication range of each roadside unit belong to a group. Owing to the high-speed mobility of vehicles, the time within the communication range of one roadside unit may be concise, and the vehicle's location information is susceptible to privacy data. Therefore, the authentication efficiency can be improved by applying the proposed scheme, and concurrently, the risk of privacy leakage is reduced.

VI. Conclusion and Future Work

To ensure the security of group communication, we design a TAAGKA scheme for Industry 5.0. With the help of Blockchain technology, all the PKGs can safely share a ledger storing information about the DEs, thereby effectively solving the problem of cross-group authentication. Furthermore, with the support of the token mechanism, when a DE has a valid token, the computation cost of authentication can be significantly reduced. The proposed scheme can also guarantee the anonymity and untraceability of a DE and the forward and backward secrecy of a group key. Mathematical analysis and ProVerif are used to prove the security and correctness of the proposed scheme. Finally, comparisons with related schemes show that the proposed scheme reduces security risks, and on the premise of ensuring security and anonymity, the proposed scheme requires the lowest energy consumption of each group member for computing and communication.

As future work, we intend to consider combining more emerging technologies, such as artificial intelligence, in security schemes further to improve the security at low energy consumption and affordable complexity on all aspects of Industry 5.0.

Appendix A
Bilinear Pairing

Let G_1 be cyclic additive groups of prime order q and G_2 be multiplicative groups of prime order q. The g_1 is the generator of G_1. The bilinear pairing is defined as $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:

- Bilinear: $\forall P, Q \in G_1$ and $\forall a, b \in Z_q^*$, such that $e(aP, bQ) = e(P, Q)^{ab}$
- Non-degenerate: $\forall P, Q \in G_1$ satisfying $e(P, Q) \neq 1$.
- Computable: There is an effective algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

In addition, there exists the following hard problems:

- Computational Diffie-Hellman problem (CDH): For $a, b \in Z_q^*$, given g_1, ag_1, and bg_1, compute abg_1.
- Decisional Diffie-Hellman problem (DDH): For $a, b, c \in Z_q^*$, given g_1, ag_1, bg_1, and cg_1, decide if $e(ag_1, bg_1) = e(g_1, cg_1)$.
- Elliptic Curve Discrete Logarithm problem (ECDL): For $a \in Z_q^*$, given $P, aP \in G_1$, compute a.

References

[1] W. Liang, S. Xie, J. Cai, J. Xu, Y. Hu, Y. Xu, and M. Qiu, "Deep neural network security collaborative filtering scheme for service recommendation in intelligent cyber-physical systems," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[2] V. V. Martynov, D. N. Shavaleeva, and A. A. Zaytseva, "Information technology as the basis for transformation into a digital society and industry 5.0," in *2019 International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT QM IS)*, 2019, pp. 539–543.

[3] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv. (CSUR)*, vol. 35, no. 3, pp. 309–329, 2003.

[4] K. Gu, L. Yang, Y. Wang, and S. Wen, "Traceable identity-based group signature," *RAIRO-Theor. Inf. Appl.*, vol. 50, no. 3, pp. 193–226, 2016.

[5] S. Gupta, A. Kumar, and N. Kumar, "Design of ecc based authenticated group key agreement protocol using self-certified public keys," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–5.

[6] S. Mandal, S. Mohanty, and B. Majhi, "Cl-agka: certificateless authenticated group key agreement protocol for mobile networks," *Wirel. Netw.*, vol. 26, pp. 3011–3031, 2020.

[7] J. Zheng, C. Yang, J. Xue, and C. Zhang, "A dynamic id-based authenticated group key agreement protocol," in *2015 4th National Conference on Electrical, Electronics and Computer Engineering*. Atlantis Press, 2015.

[8] Q. Zhang, Y. Gan, Q. Zhang, R. Wang, and Y.-A. Tan, "A dynamic and cross-domain authentication asymmetric group key agreement in telemedicine application," *IEEE Access*, vol. 6, pp. 24 064–24 074, 2018.

[9] W. Liang, Y. Li, J. Xu, Z. Qin, and L. Kuan-Ching, "Qos prediction and adversarial attack protection for distributed services under dlaas," *IEEE Trans. Comput.*, pp. 1–14, 2021.

[10] S. H. Islam, M. S. Obaidat, P. Vijayakumar, E. Abdulhay, F. Li, and M. K. C. Reddy, "A robust and efficient password-based conditional privacy preserving authentication and group-key agreement protocol for vanets," *Future Gener. Comp. Sy.*, vol. 84, pp. 216–227, 2018.

[11] G. Xu, X. Li, L. Jiao, W. Wang, A. Liu, C. Su, X. Zheng, S. Liu, and X. Cheng, "Bagkd: A batch authentication and group key distribution protocol for vanets," *IEEE Commun. Mag.*, vol. 58, no. 7, pp. 35–41, 2020.

[12] L. Liu, Y. Wang, J. Zhang, and Q. Yang, "A secure and efficient group key agreement scheme for vanet," *Sensors*, vol. 19, no. 3, 2019.

[13] V. S. Naresh, S. Reddi, and N. V. Murthy, "A provably secure cluster-based hybrid hierarchical group key agreement for large wireless ad hoc networks," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, p. 26, 2019.

[14] L. Wang, Y. Tian, D. Zhang, and Y. Lu, "Constant-round authenticated and dynamic group key agreement protocol for d2d group communications," *Inform. Sciences*, vol. 503, pp. 61–71, 2019.

[15] S. Kavitha, P. Alphonse, and Y. V. Reddy, "An improved authentication and security on efficient generalized group key agreement using hyper elliptic curve based public key cryptography for iot health care system," *J. Med. syst.*, vol. 43, no. 8, p. 260, 2019.

[16] P. Alphonse and Y. V. Reddy, "A method for obtaining authenticated scalable and efficient group key agreement for wireless ad-hoc networks," *Cluster Comput.*, vol. 22, no. 2, pp. 3145–3151, 2019.

[17] T.-F. Lee and M. Chen, "Lightweight identity-based group key agreements using extended chaotic maps for wireless sensor networks," *IEEE Sensors J.*, vol. 19, no. 22, pp. 10 910–10 916, 2019.
[18] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.
[19] J. Wang, L. Wu, K.-K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1984–1992, 2020.
[20] B. Blanchet and B. Smyth, "Automated reasoning for equivalences in the applied pi calculus with barriers," *Journal of Computer Security*, vol. 26, no. 3, pp. 367–422, 2018.
[21] Z. Xu, F. Li, H. Deng, M. Tan, J. Zhang, and J. Xu, "A blockchain-based authentication and dynamic group key agreement protocol," *Sensors*, vol. 20, no. 17, p. 4835, 2020.
[22] W. Liang, K.-C. Li, J. Long, X. Kui, and A. Y. Zomaya, "An industrial network intrusion detection algorithm based on multifeature data clustering optimization model," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2063–2071, 2020.
[23] Q. Feng, D. He, S. Zeadally, and K. Liang, "Bpas: Blockchain-assisted privacy-preserving authentication system for vehicular ad hoc networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4146–4155, 2020.

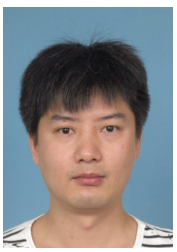
Zisang Xu received the bachelor's degree in electronic information engineering from Wuhan Institute of Technology in 2013. In 2016, he received a master's degree in computer science and technology from Hunan University of Science and Technology. In 2020, he received a Ph.D. in computer science and technology from Hunan University. He had worked in the Department of Information Engineering at Chinese University of Hong Kong as a research assistance. He is currently a lecturer in the School of Computer and Communication Engineering, Changsha University of Science and Technology. His research interests include embedded systems, information security, and cryptography.

Wei Liang is currently a Professor at the School of Computer science and Engineering, Hunan University of Science and Technology, China. He received his Ph.D. degree at Hunan University in 2013 and was a postdoctoral scholar at Lehigh University, USA, during 2014-2016. He served as Application Track Chair of IEEE Trustcom 2015, a Workshop Chair of IEEE Trustcom WSN 2015 and IEEE Trustcom WSN 2016. He has published more than 110 journal/conference papers such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Emerging Topics in Computing, and IEEE Internet of Things Journal. His research interests include Blockchain security technology, Networks Security Protection, embedded system and Hardware IP protection, Fog computing, and Security management in WSN. He is a Member of the IEEE.

Kuan-Ching Li is currently appointed as Distinguished Professor at Providence University, Taiwan. He is a recipient of awards and funding support from several agencies and high-tech companies, as also received distinguished chair professorships from universities in several countries. He has been actively involved in many major conferences and workshops in program/general/steering conference chairman positions and as a program committee member, and has organized numerous conferences related to high-performance computing and computational science and engineering. Besides publication of journal and conference papers, he is the co-author/co-editor of several technical professional books published by CRC Press, Springer, McGraw-Hill, and IGI Global. His topics of interest include parallel and distributed computing, Big Data, and emerging technologies. He is a Member of the AAAS, a Senior Member of the IEEE, and a Fellow of the IET.

Jianbo Xu received the M.S. degree in Department of Computer Science and Technology from National University of Defense Technology, China, in 1994 and the Ph.D. degree in College of Computer Science and Electronic Engineering from Hunan University, China, in 2003. Since 2003, he has been a Professor with the School of Computer science and Engineering, Hunan University of Science and Technology. He has published many journal/conference papers. His research interests include Network Security and Distributed Computing.

Albert Y. Zomaya (SM'97,F'04) is the Chair Professor of High Performance Computing & Networking in the School of Information Technologies, University of Sydney, and he also serves as the Director of the Centre for Distributed and High Performance Computing. Professor Zomaya published more than 550 scientific papers and articles and is author, co-author or editor of more than 20 books. He is the Founding Editor in Chief of the IEEE Transactions on Sustainable Computing and serves as an associate editor for more than 20 leading journals. Professor Zomaya served as an Editor in Chief for the IEEE Transactions on Computers (2011-2014). Professor Zomaya is the IEEE Computer Society Technical Achievement Award (2014), and the ACM MSWIM Reginald A. Fessenden Award (2017). He is a Chartered Engineer, a Fellow of AAAS, IEEE, and IET. Professor Zomayas research interests are in the areas of parallel and distributed computing and complex systems.



Jixin Zhang received the B.S. degree in Mathematics, and the M.S. degree in computer science and technology from Wuhan University of Technology, in 2011 and 2014, respectively. He received Ph.D degree in computer science and technology from Hunan University. He had worked in the Department of Information Engineering at Chinese University of Hong Kong as a research assistance. Currently, he is an assistant professor of Hubei University of Technology. His primary researches focus on security and machine learning.