Efficient Anonymous Authentication Based on Physically Unclonable Function in Industrial Internet of Things

Qingyang Zhang, Jing Wu, Hong Zhong, Debiao He, *Member, IEEE*, and Jie Cui, *Senior Member, IEEE*

***Abstract*— Owing to the open Industrial Internet of Things (IIoT) environment, information interacting between devices and servers is transmitted over the public channel, which may lead to privacy breach of the device identity. Furthermore, communication entities are not fully trusted, and they may maliciously disclose the device identity information. Therefore, the anonymity of devices must be guaranteed. In addition, IIoT is resource-constrained, and complex algorithms are unsuitable for the IIoT system. Several researchers have attempted to design anonymous authentication schemes. The one-authentication-multiple-access approach allows devices to access server resources multiple times after a single authentication, and its authentication overhead is independent of the number of accesses. This can reduce the computational burden for devices that need to access the server frequently. However, existing anonymous authentication schemes do not support multiple accesses after one authentication, and still suffer from privacy issues and low efficiency for devices that need frequent access to the server. To address these issues, we propose a new anonymous authentication scheme that uses group signature technology to ensure device anonymity and uses Merkle hash tree technology to achieve multiple accesses after one authentication, thereby greatly reducing the authentication overhead of IIoT devices. Then, we validate the security of the scheme using the random oracle model and the BAN logic. Finally, compared with other related schemes, the experimental results show that our proposed scheme is more efficient and practical for resource-constrained IIoTs than other schemes.**

***Index Terms*— Anonymity, authentication, Industrial Internet of Things (IIoT), physically unclonable function (PUF), Merkle hash tree (MHT).**

I. Introduction

THE Industrial Internet of Things (IIoT) [1], [2] refers to applying and expanding Internet of Things (IoT) technology in industry. Ubiquitous mobile network communication technology, intelligent analysis technology, intelligent terminals with interacting and sensing capabilities, and mobile computing patterns are applied to each segment of the factory, which improves product quality, increases production efficiency, reduces production costs, and decreases resource consumption and pollution [3]. With the increasing maturity of IoT technology, applying it in industrial scenarios will become widespread, eventually realizing the transformation from traditional to intelligent industries.

In the IIoT system, complex calculations or massive amounts of data are delivered to the servers for operation or storage. When the nodes need to access or request these resources, they first request the gateway (equivalent to the trusted authority in the factory) to issue membership certificates for them, and then use the certificates to request the servers to authorize and authenticate. After successful authentication, the nodes access the services provided by the servers. In this way, resource-constrained nodes avoid complex operations such as high computation and storage costs. However, due to the openness of the network in the IIoT system, there is a risk of privacy breach when the identity information of terminal devices is transmitted over the open channel in the communication process; further, communication entities are not fully trusted, and dishonest entities may maliciously leak the identity information of the devices. As device privacy may be associated with factory privacy, device privacy leakage can lead to factory privacy leakage, resulting in immeasurable economic losses to factory and threaten the safety of factory personnel [4], [5]. Therefore, it is necessary to design an anonymous device authentication protocol to ensure the security of IIoT device identity authentication and protect device privacy.

Anonymous authentication [6], [7], [8] is an effective means of solving the security and privacy threats mentioned above, which authenticates terminal devices and ensures the anonymity of the real identity of the devices. Therefore, it has been the focus of research in IIoT. The group signature is a common cryptographic primitive for anonymous authentication [9], [10], [11], and its anonymity, untraceability, and unlinkability can meet the needs of privacy protection in the IIoT system. Therefore, it is widely used in anonymous authentication mechanisms [12], [13]. For example, Wang et al. [14] designed a novel group-signature-based IoT terminal device authentication scheme that protected the real identity of the devices and achieved a balance between privacy and security.

Manuscript received 7 May 2022; revised 5 September 2022; accepted 7 October 2022. Date of publication 31 October 2022; date of current version 7 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62272002, Grant 62202005, and Grant U1936220; in part by the Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31; in part by the Anhui Provincial Natural Science Foundation under Grant 2208085QF198; and in part by the Special Fund for Key Program of Science and Technology of Anhui Province, China, under Grant 202003A05020043. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Husrev Taha Sencar. *(Corresponding author: Hong Zhong.)*

Qingyang Zhang, Jing Wu, Hong Zhong, and Jie Cui are with the School of Computer Science and Technology, and the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: zhongh@ahu.edu.cn).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shanghai Key Laboratory of Privacy Preserving Computation, Matrix Elements Technologies, Shanghai 201204, China (e-mail: hedebiao@163.com).

Digital Object Identifier 10.1109/TIFS.2022.3218432

Although these schemes guarantee the anonymity of users or devices, they also suffer from efficiency issues. When devices wish to continually access server resources, they must be authenticated continually by the servers. In each authentication, the devices must prove the validity of their member certificates to the servers, which requires a certain amount of computation and communication overhead that increases linearly with the number of device accesses. This is inefficient for resource-constrained devices. A user, for example, places an urgent order for a batch of products from a factory. He constantly inquires of the factory in order to master the production schedule. Traditional anonymous authentication schemes based on group signature need the manufacturer to validate the user identity each time before disclosing important production information to him. The efficiency of this authentication method still needs to be improved. To address the aforementioned issues, Huang et al. [15] introduced the concept of multiple accesses after one authentication and proposed an anonymous authentication scheme for pay-as-you-go cloud computing. Inspired by this scheme, this paper designs a new anonymous authentication scheme for the IIoT environment that supports devices to access the servers multiple times after one authentication and improves authentication efficiency while satisfying the required security features.

In addition, we also employ the physically unclonable function to generate the private keys of the devices. In this way, the devices do not need to store the private keys and only need to use the physically unclonable function to calculate, which effectively reduces the risk of device private key leakage.

A. Related Work

Inspired by the pay-as-you-go anonymous authentication protocol proposed by Huang et al. [15], which uses a zero-knowledge-sequence-proof mechanism to achieve multiple accesses after one authentication, but it still suffers from low efficiency. We aim to propose an effective anonymous authentication scheme for IIoT, using group signature technology and Merkle hash tree technology. In this case, we mainly introduce group signature and Merkle hash tree techniques in this section.

1) Group Signature: In the IIoT system, it is not enough to realize authentication, but also to prevent the leakage of private information such as the identity of legitimate devices. Therefore, how to choose a privacy-aware cryptographic primitive for authentication is crucial. The concept of group signature was first introduced by Chaum et al. [16] in 1991. It allows any group member to sign a message on behalf of the group, and the signature can be verified by the group public key without revealing the identity of the signer. And when there is a dispute about the signature result, the manager can trace the identity of the signing group member. Thus, group signature can be a cryptographic primitive for privacy-preserving applications, which guarantees the anonymity of honest signers and enables identity tracing of dishonest signers. Subsequently, Ateniese et al. [17] proposed a provably secure group signature protocol in 2000, which is resistant to collusion attacks. In 2016, Bootle et al. [18] applied one-time encryption technique and randomized certificate to the scheme and designed a group signature scheme. The signature length of this scheme is small, but the key and certificate used in each signature are different, which greatly increases the overhead of the signer. Hwang et al. [9] proposed a new group signature scheme supporting controlled linkability, which has a very short signature length and can be well-used in resource-constrained privacy-enhancing scenarios. Due to the characteristics of group signature technology, it has been applied in varieties of authentication schemes [19], [20], [21], which effectively protect the real identity of devices. Sudarsono et al. [22] proposed an anonymous authentication system based on group signature, which guarantees anonymity when users access the service. Several researchers applied group signature to wireless communication network to realize user authentication and avoid the leakage of users identity information [23], [24], [25]. And some researchers applied group signature to the IIoT scenario to protects the anonymity of nodes [26], [27], [28]. These schemes all meet the needs of user anonymity, but when devices wish to access the server resources frequently, they must be authenticated constantly by the server, so that the authentication overhead increases linearly with the number of accesses. This method still has efficiency issues for resource-constrained IIoT devices.

2) Merkle Hash Tree: The structural characteristics of the Merkle hash tree give it a significant advantage in authentication, and it can realize multiple accesses after one authentication by additionally numbering each leaf node. Suppose a device requests access to server resource twice in a single authentication. The device sends the hash value L_j of the leaf node, the corresponding number N_j and the path information API to the server and proves the relationship between the leaf node order and the number of accesses. In 2014, Li et al. [29] proposed an authentication scheme based on the Merkle hash tree technique, which requires less computation overhead and communication overhead than the traditional RSA-based authentication scheme. Koo et al. [30] developed an online authentication scheme based on Merkle hash tree, which has good validity and high reliability. Xu et al. [31] proposed a dynamic Merkle hash tree authentication scheme combined with fully homomorphic encryption, which has the advantage of being lightweight in user-side operation. After that, Sun et al. [32] proposed a Merkle hash tree-based entity authentication scheme, the scheme verifies the identity of nodes and determines whether the nodes are legitimate by computing the Merkle tree root, which reduces the computation costs and memory requirements of nodes and greatly reduces the latency time in the authentication process. Later, Nesa et al. [33] in 2020 combined Merkle hash tree and chaotic mapping to design a lightweight IoT authentication scheme, which has low computation and storage costs.

Although the aforementioned protocols have done some work to address privacy, security, or efficiency issues encountered during device authentication or user authentication, they cannot be implemented efficiently at the same time. Therefore, we propose a new IIoT-oriented anonymous authentication protocol based on previous research, which satisfies the

security requirements needed for anonymous authentication and improves the authentication efficiency.

B. Contributions

The major contributions of this paper can be summarized as:

- We design an efficient anonymous authentication scheme using group signature and Merkle hash tree technologies, which realizes anonymity in the IIoT device authentication process and allows devices to access servers anonymously. The performance analysis shows that our scheme is more efficient than the other related schemes.
- Our proposed scheme achieves a one-authentication-multiple-access approach, allowing each legitimate device to access the server resources multiple times after a single authentication.
- We conduct a formal security analysis to prove that our scheme is secure and satisfies anonymity, detectability, and exculpability.

C. Organization of the Rest Paper

The remainder of this paper is organized as follows. Section II introduces the relevant preparatory knowledge. Section III describes the system and security model, the system threats and objectives, and the syntax of the authentication scheme. Section IV describes the overview of the proposed scheme and the concrete structure of the proposed scheme. Section V presents the security analysis. Section VI introduces the experimental configuration environment and experimental results. Finally, Section VII presents a summary of our work.

II. Preliminaries

In this section, we briefly discuss the cryptography basics that will be used in this paper, including the definitions of bilinear maps, physically unclonable function, Merkle hash tree, and complexity assumptions. All the notations used in this paper are summarized in Table I.

TABLE I
DESCRIPTION OF THE ACRONYMS

Notations	Definitions
D_i	The real identity of the device i
GW	Gateway
SE	Server
L_j	The hash value of the j_{th} leaf node
$L_{i,j}$	The hash value of the internal node
N_j	The number of the j_{th} leaf node
API	The path information of the Merkle hash tree
PUF	Physically unclonable function
$H_1(\cdot)$	Cryptographic hash function: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$
$H_2(\cdot)$	Cryptographic hash function: $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1 \times \mathbb{G}_1$
G_{pk}/G_{sk}	Group public key/secret key
S_{pk}/S_{sk}	Server public key/secret key
D_{pk}/D_{sk}	Device public key/secret key
M_{pk}/M_{sk}	Member public key/secret key
IDL	The identity list of devices
LOG	Access log
Δ	Tracking list
$\|\|$	Concatenation operation
ΔT	Predefined system time delay

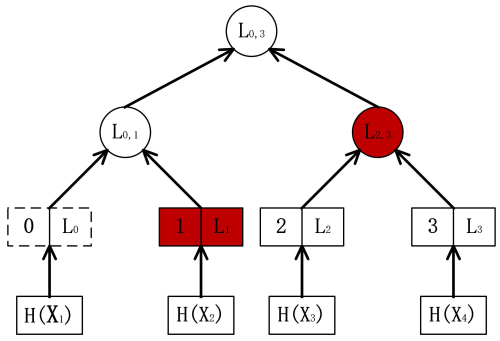


Fig. 1. Merkle hash tree construction with 4 leaf nodes.

A. Bilinear Maps

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)$ are three multiplicative groups of prime order q. The elements g, h are generators of the group $\mathbb{G}_1$, $\mathbb{G}_2$, respectively. These three groups are equipped with a computable bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ satisfies the following properties:

(1) Bilinearity: $\forall\ g_1, g_2 \in \mathbb{G}_1$, $h_1, h_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q$, $e(g_1^a, h_1^b) = e(g_1, h_1)^{ab}$, $e(g_1 g_2, h_1) = e(g_1, h_1)e(g_2, h_1)$ and $e(g_1, h_1 h_2) = e(g_1, h_1)e(g_1, h_2)$.
(2) Non-degeneracy: $\exists\ g_1 \in \mathbb{G}_1, h_1 \in \mathbb{G}_2$ such that $e(g_1, h_1) \neq 1$.
(3) Computability: $\forall\ g_1, h_1 \in \mathbb{G}_1, \mathbb{G}_2$, there exists an efficient algorithm to compute $e(g_1, h_1)$.

B. Physically Unclonable Function

The physically unclonable function (PUF) is a hardware function-implementing circuits that relies on chip characteristics and can be understood as a one-way function. It inputs a challenge to a physical entity and outputs an unpredictable response using random differences in its intrinsic physical construction. Since the physical microstructure of each device is unique, the generated challenge-response pair is also unique. In general, a PUF has the following characteristics:

(1) The output of the PUF depends on the physical structure of the system.
(2) The output of the PUF is unpredictable.
(3) The PUF is simple to evaluate as well as to construct.
(4) The PUF is unclonable.

C. Merkle Hash Tree

The construction of the Merkle hash tree is mainly based on a one-way collision-resistant hash function. The value of each leaf node in the tree is computed by a hash function, and then the hash values of internal nodes are calculated

recursively until the root node, where the hash values of the inner nodes are derived from their child nodes. Each leaf node is verified by the authentication path information (API), which is computationally inexpensive because only hash operations are used. In this paper, we make a slight modification to the leaf node construction of the Merkle hash tree. We describe the establishment of the Merkle hash tree through a demonstration. As illustrated in Fig. 1, for nodes with a given number of N (N = 4), the hash values of the given data are calculated, and the calculation result is kept as the hash value of the leaf, i.e., $L_i = H(X_i)(i = 0, 1, 2, 3)$, respectively. In addition, we numbered each leaf node in order to achieve multiple accesses after one authentication. Then two leaf nodes are aggregated into one parent node by a hash operation. In this way, the value of the root node can be computed recursively. For example, the hash value of the node $L_{0,1}$ is equal to $H(0||L_0||1||L_1)$, i.e., $L_{0,1} = H(0||L_0||1||L_1)$, and the value of the root node $L_{0,3}$ is $L_{0,3} = H(L_{0,1}||L_{2,3})$. Therefore, each leaf node can be validated by the root node $L_{0,3}$ and the corresponding authentication path information API. For example, the leaf node L_0 can be authenticated by the server who stores the value $L_{0,3}$ as follows: the device D_i sends the hash value of the leaf node L_0, the corresponding number N_j and $API = (L_1, L_{2,3})$ to the server SE. Then SE calculates $L_{0,1} = H(0||L_0||1||L_1)$, $L_{0,3} = H(L_{0,1}||L_{2,3})$, and compares whether the $L_{0,3}$ is equal to the $L_{0,3}$ stored before. If the two values are the same, the device D_i is valid.

D. Complexity Assumptions

***Definition** 1 (Decision Linear Assumption):* For a bilinear group description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g, h, q, e)$, given a tuple $(\widetilde{g_1}, \widetilde{g_2}, \widetilde{g_3}, \widetilde{g_1}^a, \widetilde{g_2}^b, \widetilde{g_3}^c) \in \mathbb{G}_1$ as input, where $a, b, c \in \mathbb{Z}_q$, if $c = a + b$, output 1, otherwise output 0. We say that DL assumption holds, if for all probabilistic polynomial time adversary $\mathcal{A}$,

$$Adv^{DL} = | Pr[\mathcal{A}(\widetilde{g_1}, \widetilde{g_2}, \widetilde{g_3}, \widetilde{g_1}^a, \widetilde{g_2}^b, \widetilde{g_3}^{a+b}) = 1] - Pr[\mathcal{A}(\widetilde{g_1}, \widetilde{g_2}, \widetilde{g_3}, \widetilde{g_1}^a, \widetilde{g_2}^b, \widetilde{g_3}^c) = 1] | \leq \epsilon.$$

***Definition** 2 (k-Strong Diffie-Hellman Assumption):* For a bilinear group description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g, h, q, e)$, given a tuple of k elements $(h^a, h^{a^2}, \ldots, h^{a^k})$, we say that k-SDH assumption holds, if for all probabilistic polynomial time adversary $\mathcal{A}$,

$$Pr[\mathcal{A}(\mathcal{G}, (h^a, h^{a^2}, \ldots, h^{a^k})) = (b, g^{\frac{1}{a+b}})] \leq \epsilon.$$

III. System Model and Goals

In this section, we first present the system model for the proposed authentication scheme, and then briefly introduce the security threats and goals. Besides, we describe the syntax and security definitions of the scheme.

A. System Model

There are three entities in our authentication system as shown in Fig. 2: the IIoT devices (D_i), the gateway (GW) and the servers (SEs).

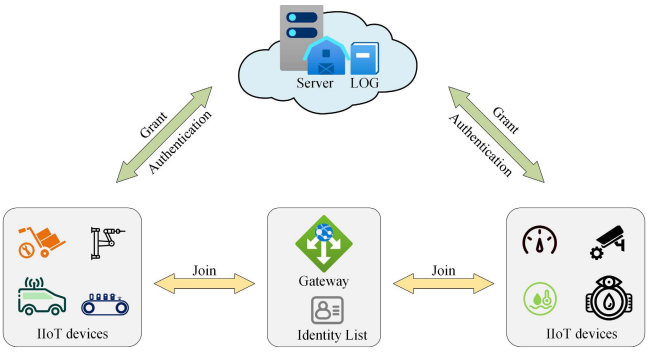


Fig. 2. System model.

- **Gateway** (**GW**): In our model, the gateway is a trusted entity that generates global parameters for IIoT devices and servers. It is equivalent to a trusted authority in a factory and is responsible for issuing certificates for IIoT devices. When an IIoT device requests to join the group managed by the gateway, GW needs to issue a group certificate for the device and add the identity D_i of the device to the identity list IDL. When a dispute arises, GW is the only entity that can reveal the real identity of the device.
- **IIoT devices** (**D_i**): In our model, each IIoT device is equipped with a PUF to generate its own private key. Each device has poor computing power and limited storage capacity. When a device wants to access the server outside the factory, it needs to prove to the server that it is a legitimate group member, and then obtain authorization and an access log from the server, which is then used to authenticate with the server, and after successful authentication, the IIoT device can access the server resources.
- **Servers** (**SEs**): Servers are semi-trusted entities that provide resources for IIoT devices and handle some complex operations. SEs need to verify the legitimacy of the device and provide services for authorized devices.

B. Security Threats and Goals

There are the following security threats in our system:

- **Internal Threats**
  - **Semi-trusted SEs**: In our scheme, we consider the server as a semi-trusted entity, i.e., it is honest but curious. Specifically, it honestly performs granting and authentication protocols, but it is curious about all the access logs it maintains and wants to know additional information about the devices, especially the identity of the authenticated devices.
- **External Threats**
  - **Eavesdropping Attack**: Since the communication channel between servers and IIoT devices is insecure, an attacker may try to eavesdrop on the information they interact with, leading to the leakage of some private information and compromising system security.
  - **Impersonation Attack**: An attacker may attempt to impersonate as a legitimate IIoT device and interact

with the server, causing the server to accept the attacker's authentication. When an attacker executes an impersonating attack and succeeds, the attacker can gain access to the server on behalf of the legitimate device. In this regard, an attacker typically impersonates as a legitimate device by repeatedly intercepting the messages between the device and the server and reusing that messages.

– **Collision Attack**: Multiple dishonest IIoT devices may collude together to execute all of the above attacks.

To get rid of the above security threats, our scheme must satisfy the following goals:

- **Correctness**: An honest terminal device should always be anonymously authenticated by a server SE.
- **Relaxed Anonymity**: If the authenticated device honestly performs the authentication process with the different servers, it is impossible to identify whether two authentication processes are performed by the same device. But the authentication procedures with the same server are linkable.
- **Detectability**: The $Trace$ algorithm does not output an empty set, if a set of collusive group devices use the same authentication path information in different authentication processes, i.e., a leaf node is reused.
- **Exculpability**: The $Trace$ algorithm does not output the legitimate devices that honestly perform each process, even if all devices collude together.
- **Structure-security**: The adversary cannot change the structure of the Merkle hash tree, and the forged authentication path information of the adversary cannot be authenticated by the server.

C. Syntax of Anonymous Authentication Scheme

In this paper, the proposed scheme consists of three algorithms: $GWSetup$, $SESetup$, $Trace$ and three protocols: $Join$, $Grant$, $Auth$. The general implementation process of the protocols and algorithms are as follows:

(1) $\mathbf{GWSetup}(\mathbf{1}^{\theta}) \rightarrow (\mathbf{G_{pk}}, \mathbf{G_{sk}})$: On input a unary string 1^{θ}, where θ is a security parameter, GW employs this algorithm to obtain a group public/secret key pair (G_{pk}, G_{sk}). Meanwhile, the algorithm also generates some public parameters.

(2) $\mathbf{SESetup}(\mathbf{1}^{\theta}) \rightarrow (\mathbf{S_{pk}}, \mathbf{S_{sk}})$: On input a unary string 1^{θ}, the server SE invokes this algorithm to obtain a public/secret key pair (S_{pk}, S_{sk})

(3) $\mathbf{Join_{GW-D}}(\mathbf{G_{sk}}, \mathbf{G_{pk}}, \mathbf{D_{pk}}) \rightarrow (\mathbf{M_{pk}}, \mathbf{M_{sk}}, \mathbf{IDL})$: The interactive protocol is performed jointly by a device which is embedded a PUF and the GW in a secure environment. The device obtains a group device public/secret pair (M_{pk}, M_{sk}), then the GW adds the identity information of each D_i to the identity list IDL, which is maintained by GW.

(4) $\mathbf{Grant_{SE-D}}(\mathbf{G_{pk}}, \mathbf{S_{pk}}, \mathbf{S_{sk}}, \mathbf{M_{pk}}, \mathbf{LOG}) \rightarrow (\mathbf{MHT}, \mathbf{LOG})$: The server SE verifies the validity of group devices, grants access times to devices, and records relevant information in the access log LOG. In the meantime, D_i constructs a Merkle hash tree MHT, which is secretly stored in a local database for subsequent authentication. When the leaf nodes of the Merkle hash tree run out and the device wants to continue accessing the server, it can re-execute the process and request authorization from the server. Note that this process can be done when the device is idle and can be executed concurrently with the authentication process.

(5) $\mathbf{Auth_{SE-D}}(\mathbf{G_{pk}}, \mathbf{G_{sk}}, \mathbf{S_{pk}}, \mathbf{S_{sk}}, \mathbf{M_{pk}}, \mathbf{LOG}) \rightarrow (\mathbf{LOG})$: D_i and SE execute the interactive protocol together. SE verifies the device D_i based on the authentication path information API. If the leaf node value sent by the device is valid and has not been used before, the server accepts the device and updates the authentication transcript.

(6) $\mathbf{Trace}(\mathbf{LOG}, \mathbf{IDL}) \rightarrow \Delta$: If the device uses an invalid leaf node, the server requests the gateway to initiate the tracking algorithm. GW executes the $Trace$ algorithm using identity list IDL and access log LOG, then the algorithm outputs a set of identifies of all dishonest devices.

D. Security Model

An anonymous authentication scheme is secure if it satisfies the following security features: correctness, anonymity and traceability. Before describing the security definitions, we first introduce the oracles that we need as follows. The list oracle $OList$ manages the identity list IDL, which is securely maintained by the gateway GW. The corruption oracle $OCorrupt$ takes the identity of a participating entity as input, generates the private key SK_{ID} of the corresponding entity, where the identity D_i ($i \in [1, N_D]$) and adds D_i to the corruption record $CList$, i.e., $CList = CList \cup \{D_i\}$. The joining oracle $OJoin$ on behalf of GW or D_i honestly invoke the $Join$ protocol. The granting oracle $OGrant$ on behalf of SE or D_i honestly invoke the $Grant$ protocol. The authentication oracle $OAuth$ on behalf of SE or D_i honestly invoke the $Auth$ protocol. Next, we describe security definitions as follows.

(1) *Correctness*: An honest server should accept the signature generated by an honest group device and allow the authenticated device to access its resources as many times as permitted with overwhelming probability as defined below.
   – $GWSetup(1^{\theta}) \rightarrow (G_{pk}, G_{sk})$
   – $SESetup(1^{\theta}) \rightarrow (S_{pk}, S_{sk})$
   – $Join_{GW-D}(G_{sk}, G_{pk}, D_{pk}) \rightarrow (IDL)$
   – $Grant_{SE-D}(G_{pk}, S_{pk}, S_{sk}, M_{pk}, LOG) \rightarrow (MHT, LOG, Accept)$
   – $Auth_{SE-D}(G_{pk}, G_{sk}, S_{pk}, S_{sk}, M_{pk}, LOG) \rightarrow (LOG, Accept)$
   – $Trace(LOG, IDL) \rightarrow \Delta$

   An anonymous authentication scheme should ensure that all six of the above steps are performed correctly.

(2) *Anonymity*: An anonymous authentication scheme is anonymous, if there is no probability polynomial time adversary can win the game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$ as follows:

- *Setup*: Challenger $\mathcal{C}$ runs initialization algorithms $GWSetup$ and $SESetup$, then it gives the obtained public keys and public parameters to adversary $\mathcal{A}$.
- *Phase 1*: $\mathcal{A}$ queries the oracles $OList$, $OJoin$, $OGrant$ and $OCorrupt$.
- *Challenge*: $\mathcal{A}$ sends two randomly selected D_0, D_1 to $\mathcal{C}$. $\mathcal{C}$ randomly picks a bit $b \leftarrow \{0, 1\}$, then generates the corresponding access log (T_0, T_1) and sends to $\mathcal{A}$.
- *Phase 2*: Adversary $\mathcal{A}$ can execute the same type of queries as in phase 1. $\mathcal{A}$ can make $OList$ and $OGrant_D$ as in phase 1. Meanwhile, $\mathcal{A}$ cannot perform $OCorrupt$ query on D_0, D_1.
- *Response*: Finally, adversary $\mathcal{A}$ returns a bit b', if $b = b'$, $\mathcal{A}$ wins the game.

***Definition** 3 (Anonymity):* We use $Adv_{\mathcal{A}}^{R-Anon} =\mid Pr(b = b') - \frac{1}{2} \mid$ to indicate the advantage of $\mathcal{A}$ in winning anonymous game. An anonymous authentication scheme satisfies anonymity if the advantage $Adv_{\mathcal{A}}^{R-Anon}$ of winning the above game is negligible for any PPT adversary $\mathcal{A}$.

(3) *Traceability*: In the game, we assume that $N_c =\mid CList \mid$ is the number of devices that collude with $\mathcal{A}$. An anonymous authentication protocol is traceable, if there is no probability polynomial time adversary can win the game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$ as follows:
- *Setup*: Challenger $\mathcal{C}$ runs initialization algorithms $GWSetup$ and $SESetup$, then it gives the obtained public keys and public parameters to adversary $\mathcal{A}$.
- *Phase 1*: $\mathcal{A}$ queries the oracles $OList$, $OJoin$, $OGrant$ and $OCorrupt$.
- *Challenge*: $\mathcal{A}$ executes granting phase with any sever on behalf of a corrupted device $D_i \in CList$. If the authentication succeeds, SE update the access log LOG.
- *Response*: Finally, adversary $\mathcal{A}$ wins the game if: (1) Π' is considered valid by SE and (2) the trace algorithm did not track illegal device, i.e., $Trace(LOG, LIST) \cap CList = \emptyset$.

***Definition** 4 (Traceability):* We use $Adv_{\mathcal{A}}^{Trace}$ to indicate the advantage of $\mathcal{A}$ in winning traceable game. An anonymous authentication scheme satisfies traceability if the advantage $Adv_{\mathcal{A}}^{Trace}$ of winning the above game is negligible for any PPT adversary $\mathcal{A}$.

(4) *Structure-security*: $\mathcal{A}$ queries the challenger $\mathcal{C}$ about leaf nodes $(L_1, \ldots, L_m)$, and then $\mathcal{C}$ sends the corresponding authentication path information to the adversary. $\mathcal{A}$ tries to destroy the structure of the tree and forge an authentication path information. A more formal definition is shown below:
- *Setup*: Challenger $\mathcal{C}$ runs initialization algorithms $GWSetup$ and $SESetup$, then it gives the obtained public keys and public parameters to adversary $\mathcal{A}$.
- *Phase 1*: $\mathcal{A}$ queries the oracles $OList$, $OJoin$, $OGrant$, $OAuth$ and $OCorrupt$.
- *Challenge*: $\mathcal{A}$ executes authentication phase with any sever on behalf of a corrupted device $D_i \in CList$. If the authentication succeeds, SE update the access log LOG and $\mathcal{A}$ has access to the server resources.
- *Response*: Finally, adversary $\mathcal{A}$ wins the game if: (1) API' is considered valid by SE and (2) the trace algorithm did not track illegal device, i.e., $Trace(LOG, LIST) \cap CList = \emptyset$.

***Definition** 5 (Structure-security):* We use $Adv_{\mathcal{A}}^{mht}$ to indicate the advantage of $\mathcal{A}$ in winning the above game. A Merkle hash tree is secure if the advantage $Adv_{\mathcal{A}}^{mht}$ of winning the above game is negligible for any PPT adversary $\mathcal{A}$.

IV. Proposed Scheme

In this section, we first give an overview of the proposed scheme. Then, we provide the details of the proposed scheme that consists of three algorithms and three protocols: $GWSetup$, $SESetup$, $Join$, $Grant$, $Auth$ and $Trace$.

A. Overview of the Proposed Scheme

First, in the system initialization phase, the gateway (GW) and server (SE) initialize some public parameters and generate their public and private keys. Second, in the join phase, GW issues group membership certificates for IIoT devices (D_i) and adds the device information to the identity list (IDL). Then, D_i uses the certificates to generate their own member public keys and private keys. In the grant phase, D_i generates a Merkle hash tree (MHT), stores it in the local database and proves to the server the validity of the membership certificate it has. After successful verification, SE grants the device permission to access its resources and records a transcript. In the authentication phase, the device sends an access request to the server using the previously generated parameters. The server verifies the legitimacy of the device and updates the authentication log. After authentication succeeds, the device can access the resources of the server. Finally, if the system detects illegal behavior of a device, the gateway performs a trace step and adds the dishonest device to the trace list.

B. Details of the Proposed Scheme

GWSetup$(\mathbf{1}^\theta) \rightarrow (\mathbf{G_{pk}}, \mathbf{G_{sk}})$. Let us presume that the security parameter is θ. On input a unary string 1^θ, the setup algorithm for GW uses the Bilinear Pairing Instance Generator to output a group description $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g, h, q, e)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ are three cyclic multiplicative groups of prime order q. g, h are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively, and an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. GW randomly selects a group element $g_1 \in_R \mathbb{G}_1$, and two one-way hash functions: $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1 \times \mathbb{G}_1$. Then it generates two random numbers u, $v \in_R \mathbb{Z}_q$ and computes $U = g^u$, $V = h^v$. GW establishes an identity list IDL and initializes it as empty. Finally, it secretly stores group secret key $G_{sk} = (u, v)$ and publishes group public key $G_{pk} = (g, g_1, h, U, V)$ and public parameters $para = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, q, e, H_1, H_2)$

SESetup$(\mathbf{1}^\theta) \rightarrow (\mathbf{S_{pk}}, \mathbf{S_{sk}})$. On input a security parameter θ in unary presentation, the server SE randomly chooses $\mu \in_R \mathbb{Z}_q$. Then it calculates $P = h^\mu$. SE sets the public/secret

key pair as $(S_{pk}, S_{sk}) = (P, \mu)$. Meanwhile, SE initializes an empty access log LOG.

$\mathbf{Join_{GW-D}(G_{sk}, G_{pk}, U_{pk})}$. This protocol is performed jointly by device who is embedded a PUF instance and the GW in a secret and secure communication channel as follows:

(1) Device D_i randomly chooses one challenge $c_i \in \{0,1\}^\theta$ and applies it to PUF_i to calculate the response r_i: $r_i = PUF_i(c_i)$. Then it computes $F_i = g_1^{r_i}$ and constructs a corresponding proof $\Pi_1 = NIZK\{r_i : F_i = g_1^{r_i}\}$. It then sends (D_i, F_i, Π_1) to GW.

(2) GW randomly selects $t_i \in_R \mathbb{Z}_q$, it then computes $A_i = (gF_i)^{\frac{1}{t_i+\nu}}$ and sends (t_i, A_i) to D_i. In the meantime, it adds the entry (D_i, g^{t_i}) to the identity list. i.e., $IDL = IDL \cup \{(D_i, g^{t_i})\}$.

(3) D_i verifies the relation $A^{t_i+\nu} = gg_1^{r_i}$ by checking whether equation $e(A_i, V)e(A_i, h)^{t_i}e(g,h)^{-r_i} = e(g,h)$ holds or not. If not, device D_i requests to join again. Otherwise, the new member D_i sets member public/secret key pair as $(M_{pk_i}, M_{sk_i}) = ((t_i, A_i, c_i), (r_i))$. Please note that D_i do not need to store the secret key, it can directly restore it with PUF_i when using it.

$\mathbf{Grant_{SE-D}(G_{pk}, S_{pk}, S_{sk}, M_{pk}, LOG)}$. This protocol is performed jointly by device who is embedded a PUF instance and the GW in a public communication channel as follows:

(1) Firstly, device D_i creates a Merkle hash tree as follows: D_i randomly chooses 128 numbers $r_j \in_R \mathbb{Z}_q$, where $j = 0, 1, 2, \ldots, 127$, calculates the hash value, i.e., $L_j = H_1(r_i||r_j)$, then uses the number and hash value as the value of the leaf node. Then D_i computes the value of internal nodes, which are derived from their child nodes. For instance, the hash value of internal nodes $L_{2,3} = H_1(2||L_2||3||L_3)$ and $L_{8,9} = H_1(8||L_8||9||L_9)$. Therefore, D_i can recursively compute the hash value of the parent node of all nodes. Finally, the hash value of the root node is calculated, i.e., $L_{0,127} = H_1(L_{0,63}||L_{64,127})$. At this point, device D_i has generated a Merkle hash tree. Then D_i securely stores the Merkle hash tree in his local database.

(2) D_i gets the current timestamp T_{D_i} and generates the ciphertext of the root node hash value with the public key of SE, where $m_i = Enc_{S_{pk}}(L_{0,127}, T_{D_i})$,

(3) Device D_i picks two random numbers $\alpha, \beta \in_R \mathbb{Z}_q$, then calculates:

$$\begin{aligned}(x, y) &= H_2(S_{pk}||m_i)\\ a_1 &= x^\alpha, \quad a_2 = A_i y^\alpha, \quad a_3 = \alpha \cdot t_i\\ B_i &= g^{t_i+\beta}, \quad E_i = V^\beta\end{aligned}$$

(4) Then device D_i generates the signature Π as follows:

- Firstly, D_i chooses $f_\alpha, f_\beta, f_{t_i}, f_{r_i}, f_{a_3} \in_R \mathbb{Z}_q$.
- It then calculates

$$\begin{aligned}F_1 &= a_1^{f_{t_i}} x^{-f_{a_3}}\\ F_2 &= \frac{e(y,h)^{f_{a_3}} e(y,V)^{f_\alpha} e(g_1,h)^{f_{r_i}}}{e(a_2,h)^{f_{t_i}}}\\ F_{B_i} &= g^{f_{t_i}+f_\beta}, \quad F_{E_i} = V^{f_\beta}\end{aligned}$$

- Then D_i generates a challenge $C = H_1(G_{pk}, S_{pk}, m_i, a_1, a_2, B_i, E_i, F_1, F_2, F_{B_i}, F_{E_i})$.
- D_i generates $l_\alpha = f_\alpha + C\alpha$, $l_\beta = f_\beta + C\beta$, $l_{t_i} = f_{t_i} + Ct_i$, $l_{r_i} = f_{r_i} + Cr_i$, $l_{a_3} = f_{a_3} + Ca_3$, and the signature $\Pi_2 = (C, l_\alpha, l_\beta, l_{t_i}, l_{r_i}, l_{a_3})$.

(5) D_i transports $(m_i, T_{D_i}, a_1, a_2, B_i, E_i, \Pi_2)$ to SE.

(6) After receiving the message, SE executes the following steps to verify the signature:

- SE first checks inequality $|T'_{D_i} - T_{D_i}| \le \Delta T$ to determine whether the timestamp is valid. If not, aborts. Otherwise, SE decrypts the ciphertext m_i with the private key S_{sk} to get the root value $L_{0,127}$, where $(L_{0,127}, T_{D_i}) = Dec_{S_{sk}}(m_i)$.
- Then SE calculates:

$$\begin{aligned}\widetilde{F_1} &= a_1^{l_{t_i}} x^{-l_{a_3}}\\ \widetilde{F_2} &= \frac{e(y,h)^{l_{a_3}} e(y,V)^{l_\alpha} e(g_1,h)^{l_{r_i}}}{e(a_2,h)^{l_{t_i}}} \left(\frac{e(g,h)}{e(a_2,V)}\right)^C\\ \widetilde{F_{B_i}} &= g^{l_{t_i}+l_\beta} B_i^{-C}, \ \widetilde{F_{E_i}} = V^{l_\beta} E_i^{-C}\end{aligned}$$

- SE generates challenge $\widetilde{C} = H_1(G_{pk}, S_{pk}, m_i, a_1, a_2, B_i, E_i, \widetilde{F_1}, \widetilde{F_2}, \widetilde{F_{B_i}}, \widetilde{F_{E_i}})$, and verifies the validity of the equation $\widetilde{C} = C$. If it is invalid, aborts.
- Otherwise, SE adds the transcript $T_i = \{i, (n, p = 0), a_1, a_2, B_i, E_i, L_{0,127}, \Pi\}$ to the access log LOG. SE shares the index i to D_i.

$\mathbf{Auth_{SE-D}(G_{pk}, G_{sk}, S_{pk}, S_{sk}, M_{pk}, LOG)}$. On requiring l times accesses in one authentication, sets $n = p + l$, where n denotes the total number of access times of the device D_i, p indicates the number of previous access times, and l represents the number of current access counts. D_i and SE execute $Auth$ protocol as follows:

(1) The device D_i chooses a leaf node L_p and its corresponding number N_j, then sends (l, N_j, L_p, API) to SE, where API represents the authentication path information. For example, D_i selects node L_0, whose authentication path $API = (L_1, L_{2,3}, L_{4,7}, \ldots, L_{64,127})$

(2) On receiving the message, SE first retrieves the transcript $T_i \in LOG$, checks whether the hash value L_p of the leaf node is recorded in T_i and compares whether p is equal to the number N_j of the leaf node. If not, outputs reject and aborts. Otherwise, it calculates $n = p + l$, and recursively computes the hash value R_i of the root node according to the leaf node L_p and the authentication path information API, and compares it with the value $L_{0,127}$ of the previously stores root node. If not, terminates. Otherwise, SE updates the protocol transcript as $T_i = \{i, (n, p = n), a_1, B_i, E_i, L_{0,127}, (N_j, L_p, API), \Pi\}$.

$\mathbf{Trace(LOG, IDL)} \rightarrow \Delta$. For all $i \in [1, N_D]$, Let $T_i = \{i, (n, p = n), a_1, a_2, B_i, E_i, L_{0,127}, (N_j, L_p, API), \Pi\}$ be one of the transcripts in LOG. If the device uses invalid leaf node value, gateway GW computes $\widetilde{g} = \frac{B_i}{E_i^{1/v}}$, then it retrieves the identity list IDL to obtain the real identity D_i of the dishonest device and adds it to the tracking list, i.e., $\Delta = \Delta \cup \{D_i\}$.

V. Security Analysis

In this section, we analyze the security of the proposed scheme by using the random oracle model and the BAN logic analysis.

A. Analysis Under the Random Oracle Model

In this section, we demonstrate the security of the protocol we proposed in Section IV. We first prove that the protocol satisfies correctness, anonymity and traceability under k-SDH assumption and DL assumption. Then we prove the security of the Merkle hash tree. In addition, we suppose that the hash function is chosen from a universal one-way family and a random oracle and is collision-resistant.

Theorem 1: The anonymous authentication scheme satisfies correctness.

Proof: To show the correctness, we demonstrate that a signature constructed by an honest device can be authenticated by the server. Firstly, we treat some parameters: group public key $G_{pk} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g, g_1, h, q, e, H_1, H_2, U, V)$, member public/secret key pair as $(M_{pk_i}, M_{sk_i}) = ((t_i, A_i, c_i), (r_i))$ and a k-SDH tuple $(h^a, h^{a^2}, \ldots, h^{a^k})$ as the output of the $Setup$ algorithm and the $Join$ algorithm.

An honest device with membership certificate (A_i, t_i) and private key (r_i) generates a signature $\Pi = (C, l_\alpha, l_\beta, l_{t_i}, l_{r_i}, l_{a_3})$. The hash value calculated by the server SE must be equal to the challenge C sent by the device D_i. A signature is valid only if all the input of the server is exactly the same as that of the device. Firstly,

$$\begin{aligned}\widetilde{F_1} &= a_1^{l_{t_i}} x^{-l_{a_3}} = (x^\alpha)^{f_{t_i}+Ct_i} \cdot x^{-f_{a_3}-Ct_i\alpha} \\ &= (x^\alpha)^{f_{t_i}} \cdot x^{-f_{a_3}} = F_1\end{aligned}$$

Secondly,

$$\begin{aligned}\widetilde{F_{B_i}} &= g^{l_{t_i}+l_\beta} B_i^{-C} \\ &= g^{f_{t_i}+Ct_i+f_\beta+C\beta} g^{-Ct_i-C\beta} = g^{f_{t_i}+f_\beta} = F_{B_i} \\ \widetilde{F_{E_i}} &= V^{l_\beta} E_i^{-C} = V^{f_\beta+C\beta} V^{-C\beta} = V^{f_\beta} = F_{E_i}\end{aligned}$$

Finally,

$$\begin{aligned}\widetilde{F_2} &= \frac{e(y,h)^{l_{a_3}} e(y,V)^{l_\alpha} e(g_1,h)^{l_{r_i}}}{e(a_2,h)^{l_{t_i}}} \cdot \left(\frac{e(g,h)}{e(a_2,V)}\right)^C \\ &= \frac{e(y,h)^{f_{a_3}} e(y,V)^{f_\alpha} e(g_1,h)^{f_{r_i}}}{e(a_2,h)^{f_{t_i}}} \\ &\quad \cdot \left(\frac{e(y,h)^{a_3} e(y,V)^{\alpha} e(g_1,h)^{r_i}}{e(a_2,h)^{t_i}} \cdot \frac{e(g,h)}{e(a_2,V)}\right)^C \\ &= F_2 \cdot \left(\frac{e(y,h)^{a_3} e(y,V)^{\alpha} e(g_1,h)^{r_i}}{e(a_2,h)^{t_i}} \cdot \frac{e(g,h)}{e(a_2,V)}\right)^C \\ &= F_2\end{aligned}$$

Therefore, SE can successfully validate the signature generated by the honest device. □

Theorem 2: The anonymous authentication scheme satisfies anonymity under DL assumption and considering the hash function as a random oracle.

Proof: Assuming $\mathcal{A}$ break the anonymous game, we construct an algorithm $\mathcal{B}$ that can break the DL assumption in $\mathbb{G}_1$. $\mathcal{B}$ is given a tuple $(x_0, x_1, y, Z_0 = x_0^a, Z_1 = x_1^b, W)$, where $x_0, x_1, y \in_R \mathbb{G}_1$, $a, b \in_R \mathbb{Z}_q$ and $W = y^{a+b}$ or $Z \in_R \mathbb{G}_1$. $\mathcal{B}$ determines which W is given during the interaction with $\mathcal{A}$ as follows:

(1) *Setup*: Let g, h are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively. $\mathcal{B}$ does the following:
- $\mathcal{B}$ randomly chooses two number $u, v \in_R \mathbb{Z}_q$, and computes $U = g^u$, $V = h^v$. Then it sets group public/secret key pair as $(G_{pk}, G_{sk}) = ((g, h, U, V), (u, v))$ and sends G_{pk} to $\mathcal{A}$.
- $\mathcal{B}$ randomly chooses two device identifiers $(0, 1) \in [1, N_D]$ and holds $(0, 1)$ in secret. Then it generates key pair $(M_{pk_j}, M_{sk_j}) = ((A_j, t_j, c_j), (r_j))$
- *Corruption queries*: for all devices $D_j \neq D_0, D_1$.
- $\mathcal{B}$ randomly chooses a number $E \in_R \mathbb{G}_1$.

Now, In order to understand the rest of our simulation, we define $A_0 = \frac{W\alpha}{y^a}$, $A_1 = y^b\alpha$. In the following, $\mathcal{B}$ regards $((A_0, t_0, c_0), (r_0))$ and $((A_1, t_1, c_1), (r_1))$ as the key pairs of D_0, D_1. In fact, since $\mathcal{B}$ does not know the random numbers a and b, $\mathcal{B}$ does not know the key pairs $((A_0, t_0, c_0), (r_0))$ and $((A_1, t_1, c_1), (r_1))$ of the device D_0 and D_1.

(2) *Hash queries*: Adversary $\mathcal{A}$ can query the hash oracles H_1, H_2 at any time, and $\mathcal{B}$ provides $\mathcal{A}$ with random values.

(3) *Phase 1*: $\mathcal{A}$ can query the oracles $OList$, $OJoin$, $OGrant$ and $OCorrupt$. $\mathcal{B}$ uses the certification Cer_i to respond to the query if $i \neq (0, 1)$. $\mathcal{B}$ makes the following responses for the queries for devices D_0 and D_1:
- *Grant queries*: given a device $i \in \{0, 1\}$, $\mathcal{B}$ generates a signature using the key pair $(M_{pk}, M_{sk}) = ((A_i, t_i, c_i), (r_i))$ of the device D_i as follows: To construct a signature for device $i = 0$, $\mathcal{B}$ first chooses three random $\beta, \gamma, \delta \in_R \mathbb{Z}_q$ and computes:

$$\begin{aligned}a_1 &= Z_0 x_0^\beta, a_2 = WEy^\beta Z_0^\gamma x_0^{\beta\gamma} \\ \widetilde{x} &= x_0^\delta, \widetilde{y} = (yx_0)^\delta\end{aligned}$$

Then sets $\alpha = (a + \beta)/\delta \in \mathbb{Z}_q$, $a_1 = \widetilde{x}^\alpha$ and $a_2 = A_0\widetilde{y}^\alpha$.

Secondly, to construct a signature for device $i = 1$, $\mathcal{B}$ first chooses three random $\beta, \gamma, \delta \in_R \mathbb{Z}_q$ and computes:

$$\begin{aligned}a_1 &= Z_1 x_1^\beta, a_2 = EZ_1^\gamma x_1^{\beta\gamma} / y^\beta \\ \widetilde{x} &= x_1^\delta, \widetilde{y} = (x_1^\gamma / y)^\delta\end{aligned}$$

Then sets $\alpha = (b + \beta)/\delta \in \mathbb{Z}_q$, $a_1 = \widetilde{x}^\alpha$ and $a_2 = A_1\widetilde{y}^\alpha$. Next, $\mathcal{B}$ selects random numbers $C, l_\alpha, l_\beta, l_{t_i}, l_{r_i}, l_{a_3} \in_R \mathbb{Z}_q$ and calculates F_1, F_2, F_{B_i}, F_{E_i} as in Section IV. In the meanwhile, $\mathcal{A}$ has conducted a hash query on $H_1(G_{pk}, S_{pk}, L_{0,127}, a_1, a_2, B_i, E_i, F_1, F_2, F_{B_i}, F_{E_i})$, $\mathcal{B}$ outputs failure and aborts, because these parameters $(a_1, a_2, F_1, F_2, F_{B_i}, F_{E_i})$ are calculated from random numbers. Otherwise, $\mathcal{B}$ defines $C = H_1(G_{pk}, S_{pk}, L_{0,127}, a_1, a_2, B_i, E_i, F_1, F_2, F_{B_i}, F_{E_i})$
- *Corruption queries*: If $\mathcal{A}$ invokes a corruption query for device D_0 or D_1, $\mathcal{B}$ outputs failure and aborts.

(4) *Challenge*: $\mathcal{A}$ outputs two devices D_0^*, D_1^* for the challenge. $\mathcal{B}$ outputs failure and aborts if $(D_0^*, D_1^*) \neq (D_0, D_1)$. If not, let $D_0^* = D_0$ and $D_1^* = D_1$. $\mathcal{B}$ randomly chooses a number $b \in \{0, 1\}$ and computes a signature Π^* and obtain an access log T_i^* using the key pair of the

device D_b as in phase 1. Then $\mathcal{B}$ sends the Π^* and T_i^* to $\mathcal{A}$.

(5) *Phase 2*: $\mathcal{A}$ can make the same query as in the phase 1 expect for $OCorrupt$ query on D_0 and D_1. The responses of $\mathcal{B}$ are same as phase 1.

(6) *Output*: $\mathcal{A}$ output a bit $b' \in 0, 1$. If $b' = b$, $\mathcal{B}$ outputs 0, i.e., W is chosen randomly in $\mathbb{G}_1$; otherwise, $\mathcal{B}$ outputs 1, i.e., $W = y^{a+b}$.

We assume that $\mathcal{B}$ does not suspend and simulate the anonymity game when W is chosen randomly in $\mathbb{G}_1$. Let ϵ be the advantage that $\mathcal{A}$ successfully guesses b. Thus, $Pr[b = b'] > \frac{1}{2} + \epsilon$. When $W = y^{a+b}$, $Pr[b = b'] = \frac{1}{2}$. Hence, if $\mathcal{B}$ does not terminate, it has probability at least $\frac{\epsilon}{2}$ to break DL assumption.

If $\mathcal{B}$ successfully guesses D_0^* and D_1^*, it also does not terminate. The probability that $OGrant$ query causes $\mathcal{B}$ to terminate is at most q_H/q. Hence, the probability that $\mathcal{B}$ terminates because of the signature of $\mathcal{A}$ is at most $q_G q_H/q$. $\mathcal{A}$ cannot get any information about the choice of D_0 and D_1. Therefore, the probability that the choice of challenge and the query do not cause $\mathcal{B}$ to terminate is at least $1/n^2$. To summarize, $\mathcal{B}$ breaks the DL assumption with advantage at least $\frac{\epsilon}{2}(\frac{1}{n^2} - \frac{q_G q_H}{q})$. □

Theorem 3: The anonymous authentication scheme satisfies traceability under $k - SDH$ assumption and considering the hash function as a random oracle.

Proof: Our proof divides into three parts. Firstly, we introduce a framework that interacts with an adversary breaking the traceability game. Secondly, we illustrate how to instantiate the framework for different types of adversaries. Thirdly, we demonstrate that how to use the forking lemma [34] to the framework instance to obtain a solution of the SDH assumption.

(1) *Setup*: Let g, h are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively. Given $U = g^u$, $V = h^v$ and (A_i, t_i, r_i), where $i \in [1, N_D]$. We use $t_i = *$ to indicate that t_i corresponding to A_i is unknown for each i, otherwise (A_i, t_i, r_i) is an SDH pair with $e(A_i, h)^{t_i} e(A_i, V) e(g_1, h)^{-r_i} = e(g, h)$. Then we invoke setup algorithm and send the group public key (g, h, U, V) to $\mathcal{A}$. We perform oracle queries and responses as follows.

(2) *Hash queries*: Whenever, $\mathcal{A}$ can make the hash queries to get challenge C or give it random values.

(3) *Grant queries*: $\mathcal{A}$ requests the signature corresponding to index i. We run the granting procedure with (A_i, t_i, r_i) to generate a signature Π and access log T_i if $t_i \neq *$, then return Π, T_i to $\mathcal{A}$. Otherwise, we compute a challenge, a signature and obtain an access log as we did in the granting phase of Section 4. If the calculated challenge values conflict, we output failure and abort. Or else, we send Π and T_i to $\mathcal{A}$.

(4) *Secret key queries*: $\mathcal{A}$ requests the secret key of the device D_i. We offer (A_i, t_i, r_i) to $\mathcal{A}$ if $t_i \neq *$. If not, we output failure and abort.

(5) *Output*: $\mathcal{A}$ outputs a forged signature $\Pi = (C, l_\alpha, l_\beta, l_{t_i}, l_{r_i}, l_{a_3})$, obtains a corresponding access log $T_i = \{i, (k, p = 0), a_1, a_2, B_i, E_i, L_{0,127}, \Pi\}$ and a trace list $LIST^*$, if $\mathcal{A}$ succeeds. $\mathcal{A}$ fails, if $\mathcal{A}$ is in the trace list. In addition, for forgery, A^* should not be included in the collusive set of the adversary $\mathcal{A}$. We output signature Π and access log LOG, if $A^* \neq A_{i^*}$ and $t_{i^*} = *$. Otherwise, we output failure and abort.

Next, we consider two types of adversary: $Type - I$ $\mathcal{A}_I$: $\mathcal{A}_I$ forges a signature with $A^* \notin A_1, A_2, \ldots, A_{N_D}$, $Type - II$ $\mathcal{A}_{II}$: $\mathcal{A}_{II}$ forges a signature with $A^* = A_{i^*}$ for each D_{i^*}, and $\mathcal{A}_{II}$ does not initiate a secret key query on D_{i^*}.

Given a group description $\mathcal{G}_n = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g, h, e)$ and a $q - SDH$ tuple $(g', h', h'^{r_i}, h'^{r_i^2}, \ldots, h'^{r_i^q})$. We use the concept of Boneh and Boyen's Lemma 3.2 [35], getting $q - 1$ SDH pair (A_i, t_i, r_i) with $e(A_i, V)e(A_i, h)^{t_i} e(g, h) = e(g, h)$ for each i. According to Boneh and Boyen's Lemma 3.2 [35], any SDH tuple (A_i, t_i) pair can be converted to a solution for the original $q - SDH$ instance.

$Type - I$ $\mathcal{A}_I$: For a $(t, q_H, q_G, N_D, \epsilon)$ $Type - I$ $\mathcal{A}_I$, we convert an instance of $(N_D + 1)$ SDH into values (g, h, U, V) and N_D tuples (A_i, t_i, r_i). Then We apply the framework to adversary $\mathcal{A}$ with these values. The environment of $\mathcal{A}$ is perfectly emulated. As long as A succeeds, the framework will succeed. Therefore, we get the probability ϵ of Type I adversary $\mathcal{A}_I$.

$Type - II$ $\mathcal{A}_{II}$: For a $(t, q_H, q_G, N_D, \epsilon)$ $Type - II$ $\mathcal{A}_{II}$, we convert an instance of $N_D - SDH$ into values (g, h, U, V) and $N_D - 1$ tuples (A_i, t_i, r_i) which are distributed in N_D pairs (A_i, t_i, r_i). We make the following padding for the unpopulated items in the random index i^*. We first choose $A_{i^*} \in_R \mathbb{G}_1$ and install $t_{i^*} = *$. $\mathcal{A}_{II}$ plays the game under the framework, and the framework will output success only if $\mathcal{A}_{II}$ never queries the private key oracle on the index i^* and successfully forges a signature. Because the group signature generated by the protocol simulator called by the granting oracle is indistinguishable from the group signature of the device containing the A_{i^*} in the SDH tuple, the index i^* is independent of $\mathcal{A}_{II}$'s view unless it makes the private key query oracle on i^*. Eventually, When $\mathcal{A}_{II}$ generates a fake signature Π, gets an access log LOG and $\mathcal{A}_{II} \notin LIST$, it means that $\mathcal{A}_{II}$ has not requested the private key of some device D_i, and the value of i^* remains independent of $\mathcal{A}_{II}$'s view. It is easy to see that the probability of the adversary $\mathcal{A}_{II}$ generating a forged signature that can be traced to device D_{i^*} is at least ϵ/N_D,

Now, we demonstrate how to apply the framework to a type I or type II adversary. The methodology and notation of the following proof are the same as the forking lemma [34]. We denote by $\mathcal{A}$ an adversary and assume that the probability of success of the framework is ϵ'. We denote signatures by (Π_0, C, Π_1), where $\Pi_0 = (x, y, a_1, a_2, F_1, F_2, F_{B_i}, F_{E_i}, B_i, E_i)$, C is calculated from a random oracle H_1, and $\Pi_1 = (l_\alpha, l_\beta, l_{t_i}, l_{r_i}, l_{a_3})$.

The operation of the framework on $\mathcal{A}$ is completely described by the random string δ used by the framework, and the response vector h generated by the H_1 hash oracle. We set Q as the set of tuples (δ, h) so that the framework called on $\mathcal{A}$ successfully forges (Π_0, C, Π_1) and $\mathcal{A}$ queried the hash oracle H_1 on Π_0. Under the circumstances, we denote the index of h by $Ind(\delta, h)$ and define $\rho = Pr[Q] = \epsilon' - 1/q$, where $1/p$ is the probability that $\mathcal{A}$ guesses the hash of Π_0 without the

help of the hash oracle. For each $j \in [1, q_H]$, $j = Ind(\delta, f)$, we denote the set of auspicious indices j by J such that $Pr[Q_j|Q] \geq 1/(2q_H)$. Then

$$Pr[Ind(\delta, h) \in J|Q] \geq \frac{1}{2}.$$

We assume that $h|_a^b$ is the limit of h on its elements at the indices $a, a+1, \ldots, b$. We consider the heavy-rows lemma [36] with rows $R = (\delta, h|_1^{j-1})$ and columns $S = (h|_j^{q_H})$ for each $j \in J$. Obviously, $Pr_{(r,s)}[(r,s) \in Q_j] \geq \rho/(2q_H)$. We suppose that the heavy rows Φ_j are those rows such that, $Pr_{s'}[(r,s') \in Q_j] \geq \rho/(4q_H)$ for all $(r,s) \in \Phi_j$. According to the heavy-rows lemma, $Pr[\Phi_j|Q_j] \geq \frac{1}{2}$. Then a simple demonstration shows that $Pr[\exists j \in J : \Phi_j \cup Q_j|Q] \geq \frac{1}{4}$.

Hence, the probability that the framework runs on $\mathcal{A}$ and successfully forges (Π_0, C, Π_1) derived from a heavy row $(s,r) \in \Phi_j$ for $j \in J$, such that

$$Pr_{h'}[(\delta, h) \in Q_j|h'|_1^{j-1}] \geq \rho/(4q_H).$$

We obtain an SDH pair (A_i, t_i, r_i) from forged (Π_0, C, Π_1) and (Π_0, C', Π_1'). The extracted A is the same as the corresponding A in (a_1, a_2) of Π_0. The framework outputs success only if the (a_1, a_2) encoded by A is not in the t it knows. Thus, the extracted SDH pair (A, t, r) is not in the tuple we created ourselves and can be converted according to Boneh and Boyen's Lemma 3.2 [35] to answer the proposed $q-SDH$ assumption. □

In conclusion, we have testified the statements as follows.

Claim 1: For a $(t, q_H, q_G, N_D, \epsilon)$ $Type-I$ adversary $\mathcal{A}_I$, the advantage that we solve an instance of (N_D+1) SDH assumption in time $\Theta(1) \cdot t$ is $(\epsilon - 1/q)^2/(16q_H)$.

Claim 2: For a $(t, q_H, q_G, N_D, \epsilon)$ $Type-I$ adversary $\mathcal{A}_I$, the advantage that we solve an instance of N_D SDH assumption in time $\Theta(1) \cdot t$ is $(\epsilon/N_D - 1/q)^2/(16q_H)$.

The probability that we can guess which of the two adversaries a particular adversary is $\frac{1}{2}$, The theorem is then proved by assuming the more pessimistic scenario of Claim 2.

Theorem 4: The Merkle hash tree MHT is secure if the hash function H_1 is collision-resistant.

Proof: We suppose that the adversary $\mathcal{A}$ has sufficient computing power. The adversary $\mathcal{A}$ queries m leaf nodes in the tree. In order to win the game, adversary $\mathcal{A}$ must output a tuple (N_{j^*}, L^*, API^*) that is successfully authenticated by the server, i.e., authentication protocol $Auth$ outputs 1.

If the Merkle hash tree is insecure, then an adversary will win the above game in polynomial time. Below shows how to build an algorithm that can find a pair of hash collisions (denoted by CH) in the tree. In the following, we describe the specific CH algorithm:

(1) *Setup*: Adversary $\mathcal{A}$ queries the hash oracles H_1 to obtain the values of certain leaf nodes $(L_1, \ldots, L_m)$. Then CH runs $Grant$ to obtain the authentication path information API_j $(j \in [1, m])$ for each leaf node and sends it to $\mathcal{A}$. The API is denoted by:

$$P := \{(N_1, L_2, API_1), \ldots, (N_m, L_m, API_m)\}$$

(2) *Computing the collision*: $\mathcal{A}$ outputs a tuple $(N_{j^*}, L^*, API^*) \notin P$ where

$$API^* = ((L_1, \ldots, L_m), R_p)$$

where R_p represents the internal node. Let $path^* = (L_1, \ldots, L_m)$ denotes the path from leaf node L^*. If

$$Auth(l^*, N_{j^*}, L^*, API^*, LOG) \rightarrow (1, LOG')$$

it implies that the authentication path information API^* of the leaf node L^* successfully passes the authentication of the $Auth$ protocol while adversary $\mathcal{A}$ does not receive the tuple (N_{j^*}, L^*, API^*).

We represent the path from the leaf node as $path^*$, the path from the j_{th}^* leaf node as $path_{j^*}$ and the authentication path information from the j_{th}^* leaf node to the root node as API_{j^*}. Then, we discuss two cases: $Case-I$: $path^* \neq path_{j^*}$, because the root node of the two paths are the same, there must exist i satisfying $i \in [1, d]$ (d denotes the depth of the tree formed by the leaf nodes queried by the adversary), such that $path^*\_sub(i+1) = path_{j^*}\_sub(i+1)$ and $path^*\_sub(i) \neq path_{j^*}\_sub(i)$, where $path_{sub}^*(i)$ denotes the sub path from the leaf node i along the path. Therefore, we can find a collision as follows:

$$path^*[i+1] = H_1(path^*[i]||API^*[i])$$
$$path_{j^*}[i+1] = H_1(path_{j^*}[i]||API_{j^*}[i])$$
$$path^*[i+1] = path_{j^*}[i+1]$$

However, since $path^*[i] \neq path_{j^*}[i]$, it is in contrast to the assumption that the hash function H_1 is collision-resistant.

$Case-II$: $path^* = path_{j^*}$. On the one hand, we can find a pair of collisions if $L^* \neq L_{j^*}$. On the other hand, if $L^* = L_{j^*}$, it means that API^* is completely different from API_{j^*}. Because if $API^* = API_{j^*}$, then it means that the tuple $(N_{j^*}, L^*, API^*) \in P$, which contradicts the previous assumption that $(N_{j^*}, L^*, API^*) \notin P$. Assume there is an i such that $API^*[i] \neq API_{j^*}[i]$ and satisfies the following equations:

$$path^*[i+1] = H_1(path^*[i]||API^*[i])$$
$$path_{j^*}[i+1] = H_1(path_{j^*}[i]||API_{j^*}[i]).$$

We can infer that $path^*[i+1] = path_{j^*}[i+1]$, because $path^* = path_{j^*}$, so we can find a pair of hash collisions. This violates the collision-resistant property of the hash function.

Through the above analysis, we can conclude that the probability of the adversary $\mathcal{A}$ breaking the Merkle hash tree is negligible, which means that $\mathcal{A}$ cannot change the structure of the Merkle hash tree and forge the correct authentication path information to pass the server's authentication. Therefore, the Merkle hash tree MHT is secure.

□

B. Analysis Using the BAN Logic

In this section, we use the BAN logic [37] to analyze the security of our scheme. The BAN logic is a widely accepted security analysis tool that uses specific logical notation and inference formulas to analyze the security of a protocol and to infer whether the protocol achieves certain security goals [38], [39], [40]. First, we introduce the notations and reasoning rules used in the BAN logic. Then, the formal idealization of our scheme, the initial assumptions, the goals and the logical derivation to achieve the goals are described in detail.

1) Ban Logic Expression: The notations in the BAN logic are expressed as follows:

- $A| \equiv S$: A believes S.
- $A| \sim S$: A once said S.
- $A \lhd S$: A sees S.
- $A| \Rightarrow S$: A controls S, i.e., A has jurisdiction over S.
- $\#(S)$: S is fresh.
- $\overset{P}{\mapsto} A$: P is the public key of A.
- $A \overset{P}{\leftrightarrow} B$: P is the key shared between A and B.
- $A \overset{S}{\Leftrightarrow} B$: S is the secret shared between A and B.
- $\{S\}_P$: S is encrypted with the key P.
- $\{S\}_{P^{-1}}$: S is signed with the private key P^{-1}.

2) The Reasoning Rules of Ban Logic: The following describes the reasoning rules used in the BAN logic:

(1) Message-meaning Rule: If A believes that P is the key shared between A and B and A receives a message S encrypted under P, then A believes that B once sent S.

$$\frac{A| \equiv B \overset{P}{\longleftrightarrow} A, A \lhd \{S\}_P}{A| \equiv B| \sim S}$$

(2) Nonce-verification Rule: If A believes that S is fresh and B once sent S, then A believes that B believes S.

$$\frac{A| \equiv \#(S), A| \equiv B| \sim S}{A| \equiv B| \equiv S}$$

(3) Jurisdiction Rule: If A believes that B controls S and B believes S, then A believes S.

$$\frac{A| \equiv B| \Rightarrow S, A| \equiv B| \equiv S}{A| \equiv S}$$

(4) Fresh Rule: If A believes that S is fresh, then A believes that (S, T) is fresh.

$$\frac{A| \equiv \#(S)}{A| \equiv \#(S, T)}$$

(5) Belief Rule: If A believes that B believes the message (S, T), then A believes that B believes the message S.

$$\frac{A| \equiv B| \equiv (S, T)}{A| \equiv B| \equiv S}$$

(6) Receiving Rule: If A receives $\{S, T\}$, then A receives S.

$$\frac{A \lhd \{S, T\}}{A \lhd S}$$

3) Modeling Process: The authentication protocol is modeled as an idealized protocol of the BAN logic, and the details of the modeling process are described below:

(1) *Message Formalization*
Message formalization is to specify the exchanged messages. In the proposed scheme, the formalized message is as follows:

$$M : SE \lhd \{\{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i}\}_{P_{SE}}, \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i}\}_{P_{D_i}^{-1}}\}$$

Since the hash value $L_{0,127}$ of the root node in the Merkle hash tree is computed by the device D_i and the server needs to verify the device with $L_{0,127}$ in the subsequent authentication process, $L_{0,127}$ is a secret value shared between the device and the server.

(2) *Initial Assumptions Declaration*
As described in our protocol, we have the following assumptions:

$$A1 : D_i| \equiv \overset{P_{SE}}{\longmapsto} SE$$
$$A2 : SE| \equiv \overset{P_{D_i}}{\longmapsto} D_i$$
$$A3 : D_i| \equiv D_i \overset{P_{SE}}{\longleftrightarrow} SE$$
$$A4 : SE| \equiv D| \Rightarrow \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\}$$
$$A5 : D_i| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\}$$
$$A6 : SE| \equiv \#(T_{D_i})$$

The assumptions $A1$, $A3$ mean that D_i believes that P_{SE} is the public key of the SE and P_{SE} is shared between D_i and SE, because P_{SE} is public for D_i. $A2$ means that SE believes that P_{D_i} is the public key of the D_i, because the SE needs to verify the membership credentials of the group device. $A4$ means that SE believes that D_i can generate the secret $L_{0,127}$ between them. $A5$ means that D_i believes that the shared secret value $L_{0,127}$, because $L_{0,127}$ is computed by D_i. $A6$ means that SE believes that the timestamp T_{D_i} is fresh. Therefore, these hypotheses are reasonable.

(3) *Expected Goals Declaration*
Finally, to prove that our scheme is secure, we need to prove that the following beliefs hold:

$$B1 : SE| \equiv D_i| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\}$$
$$B2 : SE| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\}$$

(4) *Logic Verification*
Now, we use the BAN logic to prove that the proposed scheme achieves the beliefs, the details process is described below:
From M and rule (6), we infer:

$$S2 : SE \lhd \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i}\}_{P_{SE}}$$

From $S2$, assumption $A3$ and rule (1), we infer:

$$S3 : SE| \equiv D_i| \sim \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i}\}$$

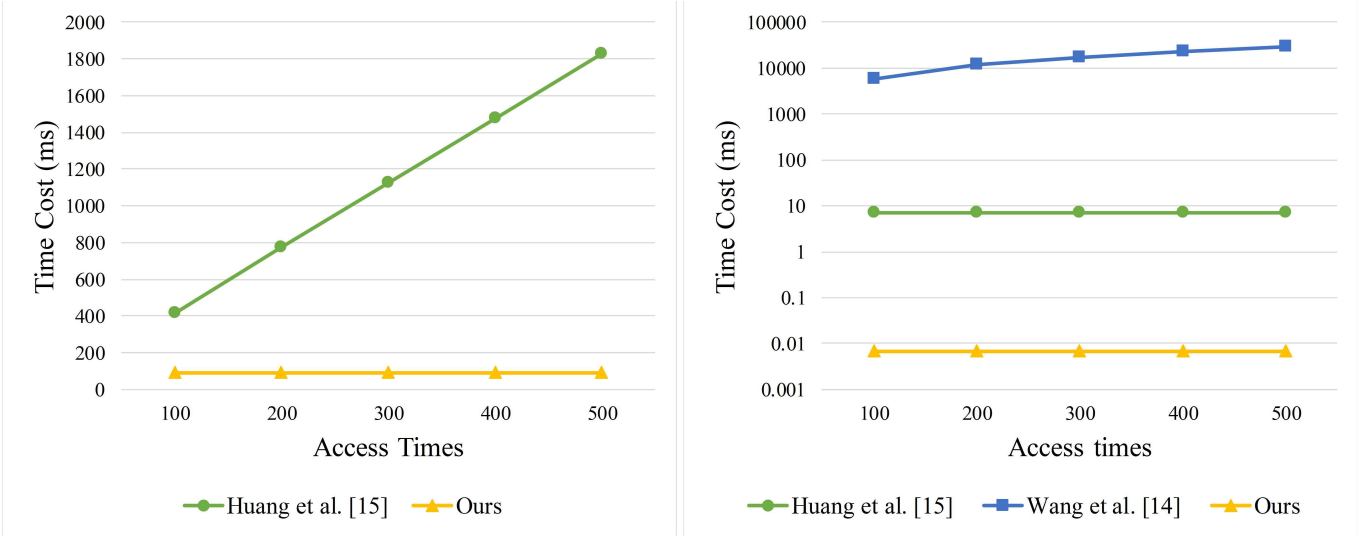


Fig. 3. Time cost under the same settings.

From $A6$ and rule (4), we infer:

$$S4: A| \equiv \#(D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i})$$

From $S3$, $S4$ and rule (2), we infer:

$$S5: A| \equiv B| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE, T_{D_i}\}$$

From $S5$ and rule (5), we infer:

$$S6: A| \equiv B| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\} \quad (B1)$$

From $S6$ and rule (3), we infer:

$$S7: A| \equiv \{D_i \overset{L_{0,127}}{\Longleftrightarrow} SE\} \quad (B2)$$

From the above analysis, we can conclude that our scheme achieves all the beliefs, implements the authentication of the device, and D_i shares the root node value $L_{0,127}$ of the Merkle hash tree with SE.

VI. Implementation and Comparisons

In this section, we first introduce the configuration to implement the experiment, and then we evaluate and compare the computation cost and communication cost of schemes [14], [15] with our scheme. These schemes are implemented on a host machine with a 3 GHz Intel Core i5-7400 CPU and 16 GB memory, and running Ubuntu-20.04 system. For software implementation, we use the charm 0.50 library in python 3.7 programming language. Additionally, our pairings use asymmetric MNT224 curve with embedding degree of 6 and the security level of 96 bits.

Then, we evaluate the performance of our scheme and those of Huang et al. [15] and Wang et al. [14]. TABLE II summarizes the execution time for major operations, including pairing (denoted by pa), exponentiation (exponentiation on the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ are denoted by e_1, e_2, e_3 respectively), hash (h_r represents hash H_1 mapped to $\mathbb{Z}_q$ and h_1 represents hash H_2 mapped to $\mathbb{G}_1$ group), dot multiplication (dot multiplication on the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ are denoted by m_1, m_2, m_3 respectively) and Enc, Dec denote the elliptic curve encryption and decryption time, respectively. We omit some overhead of lightweight operations with small proportions, which have little influence on system performance evaluation.

TABLE II
Runtime of Different Operations

Operation	Description	Time (ms)
pa	Bilinear pairing	4.28
e_1	Exponentiation on $\mathbb{G}_1$	0.88
e_2	Exponentiation on $\mathbb{G}_2$	5.76
e_3	Exponentiation on $\mathbb{G}_3$	1.18
m_1	Dot multiplication on $\mathbb{G}_1$	0.0021
m_2	Dot multiplication on $\mathbb{G}_2$	0.017
m_3	Dot multiplication on $\mathbb{G}_3$	0.0044
h_r	Hash mapped to $\mathbb{Z}_q$	0.00097
h_1	Hash mapped to $\mathbb{G}_1$	0.052
Enc	Public key encryption	1.39
Dec	Public key decryption	0.83

A. Computation Cost

We show the specific number of the operations in schemes [14], [15] and our scheme in TABLE III, and a comparative summary of the computation cost is described in Fig. 3. The execution time of the setup algorithm and joining protocol are omitted in scheme [14], [15] and our scheme, since they have less impact on the performance of the overall scheme. In order to improve efficiency, scheme [15] and our proposed scheme transfer the proof of the group membership certificate to the granting process. That is, devices do not need to generate such a proof in every authentication request. Therefore, scheme [15] and our scheme have a granting process and an authentication process, whereas scheme [14]

TABLE III
COMPARISONS OF COMPUTATION OVERHEAD

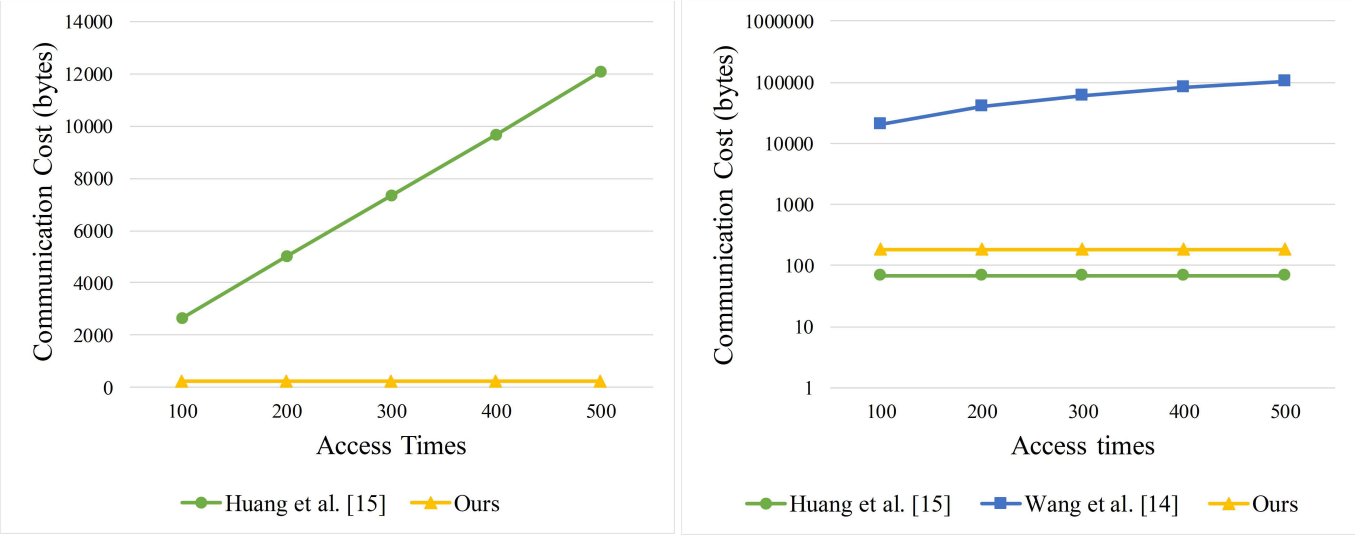
Schemes	Grant	Authentication
Huang *et al.* [15]	$2h_r + 2h_1 + 8pa$ $+ (27 + 4k)e_1 + 7e_3$ $+ (10 + k)m_1 + 6m_3$	$2h_r + 8e_1 + 2m_1$
Wang *et al.* [14]	-	$2h_r + 8pa$ $+ 17e_1 + 7e_3$ $+ 7m_1 + 6m_3$
Ours	$257h_r + 2h_1 + 10pa$ $+ 10e_1 + 4e_2 + 9e_3$ $+ 4m_1 + 1m_2 + 8m_3$ $+ 1Enc + 1Dec$	$7h_r$

(k indicates the number of accesses.)

TABLE IV
COMPARISONS OF COMMUNICATION OVERHEAD

Schemes	Grant	Authentication
Huang *et al.* [15]	$(6 + k)\|\mathbb{G}_1\| + 8\|\mathbb{Z}_q\|$	$1\|\mathbb{G}_1\| + 2\|\mathbb{Z}_q\|$
Wang *et al.* [14]	-	$3\|\mathbb{G}_1\| + 6\|Z_q\|$
Ours	$3\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\| + 7\|\mathbb{Z}_q\|$	$8\|\mathbb{Z}_q\|$

(k indicates the number of accesses.)



(a) Grant (b) Authentication

Fig. 4. Communication cost under the same settings.

has only one authentication process. From TABLE III, we can clearly see that the overhead of the authentication process in our scheme is minimal because our authentication process only involves hash operations. In the granting protocol, there is a linear relationship between the number of exponentiation and dot multiplication on the $\mathbb{G}_1$ group and the number of accesses k of scheme [15]. Specifically, from the experimental results depicted in Fig. 3, we can intuitively see that the computation overhead of our scheme is minimal and independent of the number of accesses. For every authentication process, our scheme costs about 0.0068 ms, less than 0.01 ms, scheme [15] costs about 7.05 ms. While in scheme [14], it takes nearly 57.5 ms and the computational overhead increases linearly with the number of accesses k.

B. Communication Cost

From TABLE IV, we can see that the communication overhead of our protocol is constant in the granting phase, while the communication overhead of scheme [15] is related to the number of accesses and increases with the number of accesses. In the authentication phase, our scheme requires the transmission of 8 elements in $\mathbb{Z}_q$, and its communication overhead is smaller than that of the scheme proposed by

Wang et al. [14] and slightly higher than that of the scheme proposed by Huang et al. [15], but within a reasonable range. Using the asymmetric MNT224 curve, the elements in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{Z}_q$ require 188 bits, 340 bits and 176 bits (compressed) respectively. Hence, from Fig. 4 we can clearly see that the communication overhead of our scheme is always constant and independent of the number of accesses in granting and authentication processes, which require 2136 and 1408 bits, respectively. The communication overhead in authentication process of the scheme [15] is a little less than our scheme, while the communication overhead of the grant phase is much more than ours and is linearly related to the number of accesses. The authentication communication overhead of the scheme [14] is the largest, with a communication overhead of up to 81000 bits when the number of accesses $k = 500$. The above comparison and analysis of the experimental results demonstrate that our scheme is more efficient than other schemes.

VII. Conclusion

In this paper, we discussed security and privacy problems in the IIoT environment and proposed a new efficient anonymous authentication scheme for the IIoT environment. The scheme adopted a group signature technology to achieve device anonymity. In addition, to reduce authentication overhead, our scheme used Merkle hash tree technology to achieve multiple accesses after one authentication. Our scheme provided a more flexible and efficient authentication method for devices that requires frequent access to the server resources. The security analysis proved that our scheme is secure and meets the necessary security requirements. Finally, the experimental results illustrated that the performance of the proposed scheme is better than those of other relevant schemes. In the authentication phase, our scheme only costs 0.0068 ms, which reduces the computational overhead by approximately 99% compared to the other two schemes.

Acknowledgment

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

References

[1] Y. Liao, E. de Freitas Rocha Loures, and F. Deschamps, "Industrial Internet of Things: A systematic literature review and insights," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4515–4525, Dec. 2018.

[2] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[3] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1606–1616, Apr. 2019.

[4] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[5] Q. Zhang, J. Cui, H. Zhong, and L. Liu, "Toward data transmission security based on proxy broadcast re-encryption in edge collaboration," *ACM Trans. Sensor Netw.*, vol. 18, no. 3, pp. 1–27, Aug. 2022, doi: 10.1145/3529510.

[6] P. Gope, A. K. Das, N. Kumar, and Y. Cheng, "Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 4957–4968, Sep. 2019.

[7] L. Wei, J. Cui, H. Zhong, I. Bolodurina, and L. Liu, "A lightweight and conditional privacy-preserving authenticated key agreement scheme with multi-TA model for fog-based VANETs," *IEEE Trans. Dependable Secure Comput.*, Dec. 14, 2021, doi: 10.1109/TDSC.2021.3135016.

[8] J. Subramani, A. Maria, A. S. Rajasekaran, and F. Al-Turjman, "Lightweight privacy and confidentiality preserving anonymous authentication scheme for WBANs," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3484–3491, May 2022.

[9] J. Y. Hwang, L. Chen, H. S. Cho, and D. Nyang, "Short dynamic group signature scheme supporting controllable linkability," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1109–1124, Jun. 2015.

[10] J. Cui, J. Lu, H. Zhong, Q. Zhang, C. Gu, and L. Liu, "Parallel key-insulated multiuser searchable encryption for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4875–4883, Jul. 2022.

[11] Y. Jiang, S. Ge, and X. Shen, "AAAS: An anonymous authentication scheme based on group signature in VANETs," *IEEE Access*, vol. 8, pp. 98986–98998, 2020.

[12] J. Cui, F. Wang, Q. Zhang, Y. Xu, and H. Zhong, "Anonymous message authentication scheme for semitrusted edge-enabled IIoT," *IEEE Trans. Ind. Electron.*, vol. 68, no. 12, pp. 12921–12929, Dec. 2021.

[13] P. Wang, C.-M. Chen, S. Kumari, M. Shojafar, R. Tafazolli, and Y.-N. Liu, "HDMA: Hybrid D2D message authentication scheme for 5G-enabled VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5071–5080, Aug. 2021.

[14] Z. Wang, "A privacy-preserving and accountable authentication protocol for IoT end-devices with weaker identity," *Future Gener. Comput. Syst.*, vol. 82, pp. 342–348, May 2018.

[15] J. Huang, W. Susilo, F. Guo, G. Wu, Z. Zhao, and Q. Huang, "An anonymous authentication system for pay-as-you-go cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1280–1291, Apr. 2020.

[16] D. Chaum and E. V. Heyst, "Group signatures," in *Workshop on the Theory and Application of Cryptographic Techniques*. Cham, Switzerland: Springer, 1991, pp. 257–265.

[17] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2000, pp. 255–270.

[18] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth, "Foundations of fully dynamic group signatures," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.* Cham, Switzerland: Springer, 2016, pp. 117–136.

[19] Q. Yang, K. Xue, J. Xu, J. Wang, F. Li, and N. Yu, "AnFRA: Anonymous and fast roaming authentication for space information network," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 486–497, Jul. 2018.

[20] S. Zhang and J.-H. Lee, "A group signature and authentication scheme for blockchain-based mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4557–4565, May 2020.

[21] C. Lin, D. He, N. Kumar, X. Huang, P. Vijayakumar, and K.-K.-R. Choo, "HomeChain: A blockchain-based secure mutual authentication system for smart homes," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 818–829, Feb. 2020.

[22] A. Sudarsono, T. Nakanishi, Y. Nogami, and N. Funabiki, "Anonymous IEEE802.1X authentication system using group signatures," *Inf. Media Technol.*, vol. 5, no. 2, pp. 751–764, 2010.

[23] D. He, J. Bu, S. Chan, C. Chen, and M. Yin, "Privacy-preserving universal authentication protocol for wireless communications," *IEEE Trans. Wireless Commun.*, vol. 10, no. 2, pp. 431–436, Dec. 2010.

[24] D. He, J. Bu, S. Chan, and C. Chen, "Handauth: Efficient handover authentication with conditional privacy for wireless networks," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 616–622, Mar. 2013.

[25] J. K. Liu, C.-K. Chu, S. S. M. Chow, X. Huang, M. H. Au, and J. Zhou, "Time-bound anonymous authentication for roaming networks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 178–189, Jan. 2015.

[26] A. Sudarsono, M. U. H. A. Rasyid, and M. Yuliana, "An implementation of anonymous authentication system in wireless sensor network using VLR group signature," in *Proc. Int. Electron. Symp. (IES)*, Sep. 2016, pp. 277–282.

[27] J. Lu, J. Shen, P. Vijayakumar, and B. B. Gupta, "Blockchain-based secure data storage protocol for sensors in the industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5422–5431, Aug. 2022.

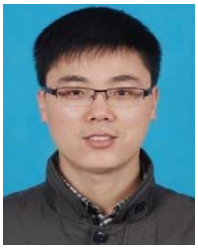
[28] C. Esposito, A. Castiglione, F. Palmieri, and A. D. Santis, "Integrity for an event notification within the industrial Internet of Things by using group signatures," *IEEE Trans. Ind. Inform.*, vol. 14, no. 8, pp. 3669–3678, Jan. 2018.
[29] H. Li, R. Lu, L. Zhou, B. Yang, and X. Shen, "An efficient Merkle-tree-based authentication scheme for smart grid," *IEEE Syst. J.*, vol. 8, no. 2, pp. 655–663, Jun. 2014.
[30] D. Koo, Y. Shin, J. Yun, and J. Hur, "An online data-oriented authentication based on Merkle tree with improved reliability," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 840–843.
[31] J. Xu, L. W. Wei, Y. Zhang, A. D. Wang, F. C. Zhou, and C.-Z. Gao, "Dynamic fully homomorphic encryption-based Merkle tree for lightweight streaming authenticated data structures," *J. Netw. Comput. Appl.*, vol. 107, pp. 113–124, Apr. 2018.
[32] J. Sun, D. Liu, L. Yin, G. Zhang, and S. Li, "Internet of Thing wireless sensor entity authentication scheme based on Merkle," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 1–6.
[33] N. Nesa and I. Banerjee, "A lightweight security protocol for IoT using Merkle hash tree and chaotic cryptography," in *Proc. Adv. Comput. Syst. Secur.* Cham, Switzerland: Springer, 2020, pp. 3–16.
[34] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2004, pp. 41–55.
[35] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Int. Conf. Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2004, pp. 56–73.
[36] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.
[37] M. Burrows, M. Abadi, and R. M. Needham, "A logic of authentication," *Proc. Roy. Soc. London A, Math., Phys. Eng. Sci.*, vol. 426, no. 1871, pp. 233–271, Dec. 1989.
[38] X. Li, T. Liu, M. S. Obaidat, F. Wu, and P. Vijayakumar, "A lightweight privacy-preserving authentication protocol for VANETs," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3547–3557, May 2020.
[39] K. Fan, Q. Luo, K. Zhang, and Y. Yang, "Cloud-based lightweight secure RFID mutual authentication protocol in IoT," *Inf. Sci.*, vol. 527, pp. 329–340, Jul. 2020.
[40] C. Lai, Y. Ma, R. Lu, Y. Zhang, and D. Zheng, "A novel authentication scheme supporting multiple user access for 5G and beyond," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 16, 2022, doi: 10.1109/TDSC.2022.3198723.

Qingyang Zhang was born in Anhui, China, in 1992. He received the B.Eng. and Ph.D. degrees in computer science from Anhui University in 2021. He is currently a Lecturer with the School of Computer Science and Technology, Anhui University. His research interests include edge computing, computer systems, and security.

Jing Wu is currently a Research Student with the School of Computer Science and Technology, Anhui University. Her research interests include security of the Internet of Things.

Hong Zhong was born in Anhui, China, in 1965. She received the Ph.D. degree in computer science from the University of Science and Technology of China in 2005. She is currently a Professor and a Ph.D. Supervisor of the School of Computer Science and Technology, Anhui University. She has over 200 scientific publications in reputable journals, such as the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and IEEE TRANSACTIONS ON BIG DATA, academic books, and international conferences. Her research interests include applied cryptography, the IoT security, vehicular ad hoc networks, cloud computing security, and software-defined networking (SDN).

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, and the Shanghai Key Laboratory of Privacy Preserving Computation, Matrix Elements Technologies, Shanghai, China. His work has been cited more than 10000 times at Google Scholar. He has published over 100 research papers in refereed international journals and conferences, such as the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSIC, and Usenix Security Symposium. His main research interests include cryptography and information security, in particular, cryptographic protocols. He was a recipient of the 2018 IEEE SYSTEMS JOURNAL Best Paper Award and the 2019 *IET Information Security* Best Paper Award. He is in the Editorial Board of several international journals, such as *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-Centric Computing and Information Sciences*.



Jie Cui (Senior Member, IEEE) was born in Henan, China, in 1980. He received the Ph.D. degree from the University of Science and Technology of China in 2012. He is currently a Professor and a Ph.D. Supervisor of the School of Computer Science and Technology, Anhui University. He has over 150 scientific publications in reputable journals, such as the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON MULTIMEDIA, academic books, and international conferences. His current research interests include applied cryptography, the IoT security, vehicular ad hoc networks, cloud computing security, and software-defined networking (SDN).