PAGS: PUF-based Anonymous Group Signature for Node Authentication in Edge Computing

Junqing Lu[1], Jian Shen[1,2,3], Chin-Feng Lai[4], Fei Gao[2]

[1] School of Computer and Software, Nanjing University of Information Science & Technology, China
[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
[3] Cyberspace Security Research Center, Peng Cheng Laboratory, China
[4] Engineering Science, National Cheng Kung University, Taiwan
ljq_nuist@126.com, s_shenjian@126.com, cinfon@ieee.org, gaof@bupt.edu.cn

Abstract

With the development of 5G communication technology and the Internet of Things (IoT) technology, edge computing faces big challenges and opportunities. In an edge computing system, edge servers deployed at the edge of the network, between the cloud server and terminal nodes, avoid long-distance data transmission. Nowadays, the quantity of terminal nodes increases rapidly due to the wide application of edge computing technology. Furthermore, the data transmitted by terminal nodes usually involves people's private information. Therefore, a privacy-preserving group authentication scheme to ensure data security and privacy is urgent to be designed. And terminal nodes are vulnerable to be compromised and then brings the leakage of secret keys. To enhance the security of nodes, we introduce physical unclonable functions (PUFs) into the group signature scheme to realize privacy-preserving group authentication for edge computing systems in this paper. Besides, the formal security models and security proofs are provided to prove the security of the proposed scheme. Finally, experiments are done not only on the raspberry pi 4 but also on the desktop. The results demonstrate that the proposed scheme is suitable for terminal nodes.

Keywords: Edge computing, Terminal nodes, Privacy-preserving, Authentication, Group signature

1 Introduction

Edge computing, the extension of cloud computing, makes up for some shortcomings of cloud computing. Cloud computing requires terminal nodes to upload data to cloud servers. The cloud server can provide high-performance computing for resource-constrained devices. However, with the rapid development of 5G communication technology, big data technology, and the Internet of Things technology, the era of the Internet of Everything is coming. The dramatic increase of network data brings a big challenge to cloud computing. Besides, most terminal nodes are far away from the cloud server. Long-distance data transmission between them brings a lot of usability and security issues. For example, the communication between terminal nodes and the cloud server will be delayed or even interrupted when the network is congested. This causes the cloud server to fail

to provide real-time services for terminal nodes. In addition, uploading data to the cloud brings the leakage of privacy. The concept of edge computing is proposed [1-2] to deal with the above issues. In edge computing, edge servers are deployed at the edge of the network. The distance between nodes and servers is greatly shortened. Terminal nodes nearby upload data to an edge server instead of the cloud server to obtain computing results. Part of work of the cloud server is shouldered by the edge server. This reduces the burden of the cloud server, avoids long-distance data transmission and ensures that terminal nodes can obtain real-time service. Furthermore, this mechanism prevents the leakage of user's privacy information.

Although the emergence of edge computing solves many problems and makes up several deficiencies, there are still many issues that remained to be deal with. The increasing application of edge computing technology leads to exponential growth in the number of terminal nodes. It puts forward the demand for group management of nodes. More than that, terminal nodes are generally deployed in industries, commerce, medical, automotive, and so on, involving billions of people. The security of the edge computing system is closely related to the security of human life and property. Therefore, the anonymous group authentication schemes for edge computing are urgent to be designed. In recent years, several authentication schemes for edge computing have been proposed. However, there are several shortcomings in these

*Corresponding Author: Jian Shen; E-mail: s_shenjian@126.com
DOI: 10.53106/160792642021112206003

existing schemes. Several schemes require terminal nodes to register with the remote trusted server. Several schemes are unable to against key leakage attacks and so on.

1.1 Motivation

As is stated above, the rapid growth of the number of terminal nodes puts forward demands for group authentication schemes. Anonymity needs to be realized to ensure the privacy-preserving of terminal nodes. Furthermore, we consider that those terminal nodes are vulnerable to be corrupted which usually brings the leakage of secret keys. Thus, we employ physical unclonable functions (PUFs) to construct a novel group signature scheme called PAGS to implement anonymous group authentication for edge computing.

1.2 Contributions

In this paper, we propose a PUF-based anonymous group signature (PAGS) scheme. Then, we provide formal security models and proofs. Besides, the experiments are performed to demonstrate the practicability of the PAGS scheme. And the details are as follows.

- **A PUF-based group signature scheme called PAGS is proposed.** Terminal nodes are vulnerable to be attacked because they are usually deployed at the edge of the network. Those attacks will bring the leakage of secret keys. Therefore, we introduce PUFs into the group signature scheme and embed the PUF instance into the terminal nodes to increase its ability to resist partial attacks.
- **The hiding of challenge-response pairs for PUF instances is implemented.** When employing PUF instances to realize authentication, the server needs to store many raw challenge-response pairs (CRPs) of PUF instances during the registration phase. Once several CRPs leak, the adversary can adopt machine learning to attack PUF instances. In this scheme, the zero-knowledge proofs are utilized to hide CRPs.
- **The formal security proofs and experimental analysis are provided.** We set up the formal security model for the PAGS scheme, and then we provide formal security proofs and analysis to prove that the PAGS scheme satisfies anonymity, traceability, and non-frameability. Finally, we test the time cost of the PAGS scheme on a raspberry pi 4 test board and a desktop.

1.3 Related Work

Group signature. The concept of group signature was originally proposed by Chaum et al. [3] in 1991. Since the suggestion of group signature, many protocols have been designed in the static model. The group membership in those protocols is fixed after the group registration phase. Then, the concept of partial dynamic group signature was proposed by Kiayias et al. [4-5]. It allows users to join the group at any time (i.e. partial dynamic group membership). Hwang et al. [6] and Emura et al. [7] proposed fully dynamic group signature schemes that support users to join or leave at any time. Besides, various group signature scheme variants were proposed. Such as ring signature [8], attribute-based group signature [9], certificateless group signature [10], linkable group signature [11] and so on. Dynamic group signature schemes are usually used to construct cryptography protocols for various scenarios due to their anonymity, non-frameability, traceability, etc. Such as VANETs [12], E-cash schemes [13], cloud auditing protocols [14], etc. At present, the research of group signature mainly focuses on implementing member revocation efficiently [15], adding practical functions [16], constructing lattice-based schemes [17] and so on.

Authentication in Edge Computing Environment. Cui et al. [18] designed a message authentication scheme based on edge computing for VANETs. Jia et al. [19] proposed an ID-based anonymous mutual authentication scheme for mobile edge computing. Wang et al. [20] utilized blockchain to construct a mutual authentication scheme for smart grid edge computing infrastructure. Jangirala et al. [21] put forward an RFID-based authentication scheme for mobile edge computing. Li et al. [22] constructed a new authentication architecture for mobile edge computing and proposed a lightweight authentication scheme based on this architecture. These schemes [19-22] are not suitable for the group management of terminal nodes. Gao et al. [23] suggested an ID-based short group signature scheme to implement access authentication. Zhang et al. [24] proposed a group signature scheme for blockchain-based mobile edge computing. Although [23-24] realize group authentication, they are infeasible to ensure security under key leakage attacks.

1.4 Organization

The organization of this paper is as follows. Section 2 provides preliminaries used throughout the whole paper. Section 3 provides the system model, security requirements and the formal security model for anonymity. Section 4 provides the concrete construction of the PAGS scheme. Section 5 provides the formal security proofs and analysis to prove the security of the scheme. Section 6 provides the experimental results of the PAGS scheme. Finally, the conclusion of this paper is shown in Section 7.

2 Preliminaries

In this section, the problems and assumptions used in this paper are introduced firstly. Then, the concept of the physically unclonable function and its

applications are described. Finally, we give the syntax of the dynamic group signature scheme to show the basic idea of the PAGS scheme.

2.1 Problems and Assumptions

2.1.1 The Decision Diffie-Hellman Problem

The decision Diffie-Hellman problem (DDHP) in $\mathbb{G}_1$ can be described as follows. Let g_1 stand for its generator. Pick random values a, b from $\mathbb{Z}_p^*$. Given $g_1^a, g_1^b \in \mathbb{G}_1$ and a random value $Z \in \mathbb{G}_1$, the algorithm $\mathcal{A}$ outputs whether $Z \stackrel{?}{=} g_1^{ab}$. When the algorithm $\mathcal{A}$ satisfies the following formula, we can say that $\mathcal{A}$ has advantage ϵ in solving DDHP.

$$\begin{aligned} &Pr\left[A\left(Z \middle| \left(\mathbb{G}_1, g_1, g_1^a, g_1^b \right) \right) \to 1 \right] \\ &-Pr\left[A\left(g_1^{ab} \middle| \left(\mathbb{G}_1, g_1, g_1^a, g_1^b \right) \right) \to 1 \right] \end{aligned} \geq \epsilon$$

***Definition* 1:** If there is no probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ solving the DDHP in t time with non-negligible advantage ϵ, we can say that the (t,ϵ)-DDH assumption holds.

2.1.2 The Computational Diffie-Hellman Problem

The computational Diffie-Hellman problem (CDHP) in $\mathbb{G}_2$ can be described as follows. Let g_2 stand for the generator of $\mathbb{G}_2$. Pick random value a, b from $\mathbb{Z}_p^*$. Given $g_2^a, g_2^b \in \mathbb{G}_2$, the algorithm $\mathcal{A}$ outputs g_2^{ab}. When the algorithm $\mathcal{A}$ satisfies the following formula, we can say that $\mathcal{A}$ has advantage ϵ in solving CDHP.

$$Pr\left[A\left(\mathbb{G}_2, g_2, g_2^a, g_2^b \right) \to g_2^{ab} \right] \geq \epsilon$$

***Definition* 2:** If there is no PPT algorithm $\mathcal{A}$ solving the CDHP in t time with non-negligible advantage ϵ, we can say that the (t,ϵ)-CDH assumption holds.

2.1.3 The Decision Bilinear Diffie-Hellman Problem

The decision bilinear Diffie-Hellman problem (DBDHP) in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ can be described as follows. Let $g_1, g_2, g_T = e(g_1, g_2)$ stand for the generators of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively. Pick random values a,b,c from $\mathbb{Z}_p^*$. Given $g_1^a, g_1^b \in \mathbb{G}_1, g_2^c \in \mathbb{G}_2$ and a random value $Z \in \mathbb{G}_T$, the algorithm $\mathcal{A}$ outputs whether $Z \stackrel{?}{=} e(g_1, g_2)^{abc}$. When the algorithm $\mathcal{A}$ satisfies the following formula, we can say that $\mathcal{A}$ has advantage ϵ in solving DBDHP.

$$\begin{aligned} &Pr\left[A\left(Z \middle| (g_1^a, g_1^b, g_2^c) \right) \to 1 \right] \\ &Pr\left[A\left(e(g_1, g_2)^{abc} \middle| (g_1^a, g_1^b, g_2^c) \right) \to 1 \right] \end{aligned} \geq \epsilon$$

***Definition* 3:** If there is no PPT algorithm $\mathcal{A}$ solving the DBDHP in t time with non-negligible advantage ϵ, we can say that the (t,ϵ)-DBDH assumption holds.

2.2 Dynamic Group Signature (Basic Idea)

Setup: System parameters are generated by this algorithm. It inputs the security parameter λ and outputs system parameters *param*.

KGen: The group manager (GM) and terminal nodes can obtain their public-private key pairs via this algorithm. It inputs system parameters *param* and outputs public-private key pairs for different entities.

Join/Issue: A terminal node can become a valid member of a group via this algorithm. It inputs the node's public keys, zero-knowledge proofs, group manager's secret keys, several system parameters and outputs the node's group certificate.

Revoke: The GM can revoke the terminal node's right to generate a valid group signature via this algorithm. It inputs revocation pattern, the node's identity that to be revoked, group manager's secret keys and outputs the updated revocation lists: C-RL and G-RL.

Sign: A terminal node can generate a valid group signature via this algorithm. It inputs the node's secret key, the node's group certificate, several system parameters and outputs a group signature.

Verify: The public can verify whether a group signature is valid. In the revocation check phase, the algorithm inputs group signature, the C-RL list, the G-RL list and outputs "abort" or "continue". In the signature check phase, the algorithm inputs group signature, group manager's public keys, several system parameters and outputs "valid" or "invalid".

Trace: The GM can obtain the node's identity via this algorithm. It inputs a group signature, group manager's secret key and outputs the identity of the terminal node, along with a zero-knowledge proof that proves the correctness of the Trace algorithm.

Judge: The public can verify whether the result of the Trace algorithm is right via this algorithm. It inputs group signature, a zero-knowledge proof, the node's public key and outputs "right" or "wrong".

3 System and Security

In this section, system model and security requirements including anonymity, traceability and non-frameability for the proposed PAGS scheme are provided, along with the corresponding security models

3.1 System Model

There are three entities in our system, namely terminal node, group manager (GM) and edge server. The system model is shown in Figure 1.

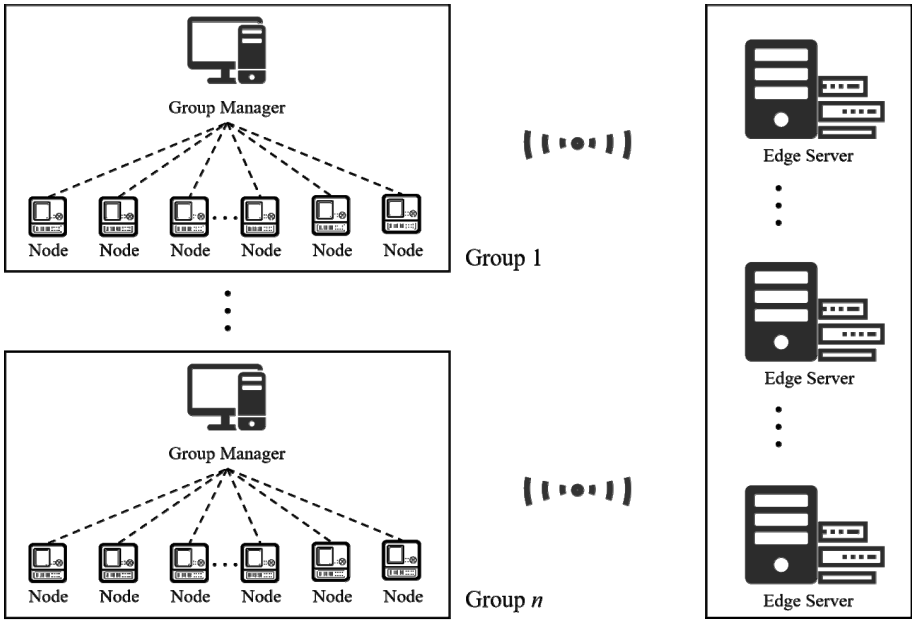

Figure 1. System Model

(1) **Terminal node:** The terminal node, a resource-constrained device, is deployed outside. And it plays the role of the signer in our scheme.

(2) **GM:** The group manager is the authority of a group that contained a large number of terminal nodes. A terminal node can register with the GM to become a valid group member.

(3) **Edge server:** The edge server, a server that is deployed at the edge of the network, plays the role of the verifier in our scheme.

3.2 Security Requirements

The proposed PAGS scheme should satisfy the following security requirements.

- **Anonymity:** Anonymity represents that it is infeasible for the PPT adversary A to obtain the identity of the terminal node just by a group signature and system parameters, even he conspires with several group members.
- **Traceability:** Traceability requires that it is infeasible for the PPT adversary $\mathcal{A}$ to generate a valid group signature σ^* that can be traced to an invalid node, even he conspires with several group members. The invalid node means that it has never joined the group or has been revoked by the group manager.
- **Non-frameability:** Non-frameability indicates that it is infeasible for the PPT adversary $\mathcal{A}$, who conspires with several group members, to generate a valid group signature σ^* that can be traced to a valid node, even the group manager has been corrupted.

3.3 Security Models

In this section, the security model for anonymity is provided. We mainly follow the security models, which are designed for a single authority scheme, that was proposed by Bootle et al. [25]. We have made adaptive modifications to the original models. The details are as follows.

- **Anonymity:** Bootle et al. [25] set six oracles for the anonymity game, namely AddHU, ReadReg, Update, SendToM, Chal_b and Open. We add four oracles, namely Hash, CorruptNode, LeakInfo. And we replace the SendToM oracle with a Sign oracle. Details of all oracles are shown in Figure 2. Modifications are marked with dashed. The security experiment for anonymity are defined as follows:

$$\text{Exp}_{PAGS,\mathcal{A}}^{Anonymity}(\lambda):$$
$$(param) \leftarrow PAGS.Setup(\lambda)$$
$$GR, HN, CN, RN, HList, CS = \varnothing,$$
$$(gpk, gsk) \leftarrow PAGS.KGen(param)$$
$$\hat{b} \leftarrow \{0,1\}$$
$$\hat{b}^* \leftarrow \mathcal{A}^{\left[\begin{array}{c}\text{Hash, CorruptNode, LeakInfo,}\\ \text{ReadReg, Revoke, AddHN ,}\\ \text{Sign, Chal}_{\hat{b}}\text{, Open}\end{array}\right]}(gpk, param)$$
$$\text{return}: \hat{b}=\hat{b}^*$$

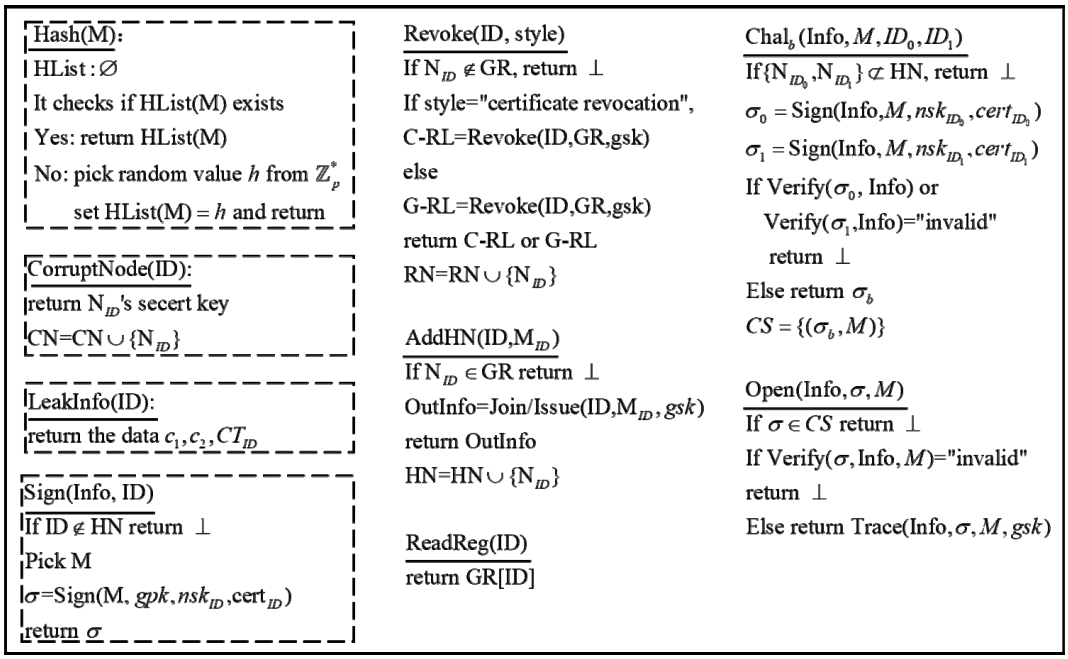

Figure 2. Oracles for Anonymity Games

4 Construction

In this section, we provide the concrete construction of the proposed PUF-based group signature scheme. And the PAGS scheme is made up of eight PPT algorithms.

PAGS.Setup (λ): It chooses two multiplicative cyclic group with order p and their generators, namely $\mathbb{G}_1, \mathbb{G}_2$ and their generator φ, ψ firstly. Then, it sets a bilinear pairing $e:(\mathbb{G}_1,\mathbb{G}_2)\to\mathbb{G}_T$ and gets $\mathbb{G}_T$'s generator $g_T = e(\varphi,\psi)$. Third, it picks $g, \hat{g}, w$ from $\mathbb{G}_1$. Finally, it sets hash function $H:\{0,1\}^*\to\mathbb{Z}_p^*$ and outputs $param=(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\varphi,\psi,g,\hat{g},w,e,H)$.

PAGS.KGen (*param*): The GM chooses random values $s, s_t, \hat{r}$ from $\mathbb{Z}_p^*$ as its *gsk*. Then, it computes $mpk=\psi^s, h=w^{s_t}$ and $\hat{R}=\psi^{\hat{r}}$ as its *gpk*. The key generation phase for nodes is completed before they are deployed. The GM sets a fixed value k ($k\geq 2$). For $1\leq i\leq k$, the node picks PUF challenges c_{i1}, c_{i2} from $\mathbb{Z}_p^*$, computes $r_{i1}=PUF(c_{i1})$ and $r_{i2}=PUF(c_{i2})$ as PUF responses, and sets secret key $x_i=H(r_{i1}\|r_{i2})$, public key $pk_{ID_i}:(npk_{i1}=\varphi^{x_i}, npk_{i2}=\psi^{x_i})$. Then, it picks random value v_i from $\mathbb{Z}_p^*$, computes comment $C_i=(\varphi^{v_i},\psi^{v_i})$, challenge $F_i=H(ID\|pk_{ID_i}\|C_i\|c_{i1}\|c_{i2})$, response $R_i=v_i-F_i\cdot x_i$, sets $P_i=(C_i,F_i,R_i)$ and sends ID, pk_{ID_i}, P_i, c_{i1}, c_{i2} to the GM. The GM checks if $F_i\overset{?}{=}H(ID\|pk_{ID_i}\|C_i\|c_{i1}\|c_{i2})$, computes $C_i^*=(\varphi^{R_i}npk_{i1}^{F_i},\psi^{R_i}npk_{i2}^{F_i})$, $F_i^*=H(ID|pk_{ID_i}\|C_i^*\|c_{i1}\|c_{i2})$ and checks whether $F_i\overset{?}{=}F_i^*$. If yes, it stores NPK[ID, i]=(c_{i1}, c_{i2}, pk_{ID_i}). Otherwise, it aborts.

PAGS.Join/Issue (*gpk*, *gsk*, ID, pk_{ID}, P, c_1, c_2, *param*): The node sends its ID to the GM. The GM randomly chooses c_1 and c_2 from NPK[ID,*] and sends them to the node. The node computes $r_1=PUF(c_1)$, $r_2=PUF(c_2)$ and $x=H(r_1\|r_2)$. Then, it picks random value v from $\mathbb{Z}_p^*$, computes comment $C=(\varphi^v,\psi^v)$, challenge $F=H(ID\|pk_{ID}\|C\|c_1\|c_2)$, response $R=v-F\cdot x$, sets $P=(C,F,R)$ and sends ID, pk_{ID}, P, c_1, c_2 to the GM. The GM checks whether $F\overset{?}{=}H(ID\|pk_{ID}\|C\|c_1\|c_2)$ and whether (c_1, c_2, pk_{ID}) is in NPK[ID,*]. Then, it computes $C^*=(\varphi^R npk_1^F,\psi^R npk_2^F)$, $F^*=H(ID\|pk_{ID}\|C^*\|c_1\|c_2)$ and checks whether $F\overset{?}{=}F^*$. Until now, the identification of the node has been completed. The GM picks random values z_{ID}, y_{ID} from $\mathbb{Z}_p^*$ and computes $cert_{ID}=(g\hat{g}^{z_{ID}}\varphi^x)^{1/(y_{ID}+s)}$. It computes the session key $K=npk_2^x$, encrypts $z_{ID}, y_{ID}, cert_{ID}$ by K to obtain ciphertext CT_{ID} and sends it to the node. Finally, it stores $GR[ID]=(npk_1,npk_2,CT_{ID})$. The node computes $r_1=PUF(c_1)$, $r_2=PUF(c_2)$ and $x=H(r_1\|r_2)$, computes session key $K=mpk^x$ and use it to decrypt ciphertext CT_{ID} to obtain z_{ID}, $y_{ID}, cert_{ID}$. Then, it checks whether $e(cert_{ID}, mpk\psi^{y_{ID}})\overset{?}{=}e(g\hat{g}^{z_{ID}}\varphi^x,\psi)$. If the equation is true, the node stores CT_{ID} locally. Note that, only PUF

challenge c_1, c_2 and ciphertext CT_{ID} are stored on the node after the algorithm.

PAGS.Revoke (ID, style, *gsk*, GR, *param*): The PAGS scheme adopts flexible revocation patterns according to different scenarios.

- ***Group Certificate Revocation*:** When the node's group certificate leaks, this method is employed to temporarily revoke the node. Until the GM issues a new group certificate for the node, it can sign again. For each node N_{ID} that to be revoked, the GM adds its group certificate $cert_{ID}$ into the list C-RL.
- ***Public Key Revocation*:** For each node N_{ID} that to be revoked, the GM finds its public key npk_2, computes $\hat{D}_{ID} = npk_2^{\hat{r}}$ and stores in the list G-RL.

PAGS.Sign $(c_1, c_2, gpk, param)$: First, the node computes $r_1 = PUF(c_1), \quad r_2 = PUF(c_2)$ and $x = H(r_1 \| r_2)$. It computes $K = mpk^x$ and decrypts CT_{ID} to obtain $z_{ID}, y_{ID}, cert_{ID}$. Second, the node picks random values d, u from $\mathbb{Z}_p^*$, computes $\Theta_1 = w^d$, $\Theta_2 = \psi^d$, $\Theta_3 = h^d cert_{ID}$, $\Theta_4 = e(npk_1, \hat{R})^u$, $\Theta_5 = \varphi^u$ and sets $\beta = dy$, $\gamma = xu$. Third, it picks random values η_x, η_z, η_y, η_d, η_u, η_β, η_γ from $\mathbb{Z}_p^*$. Then, it computes $A_1 = w^{\eta_d}$, $A_2 = \psi^{\eta_d}$, $A_3 = e(h, mpk)^{\eta_d} e(\hat{g}, \psi)^{\eta_z} e(\varphi, \psi)^{\eta_x} e(h, \psi)^{\eta_\beta} e(\Theta_3, \psi)^{-\eta_y}$, $A_4 = e(\varphi, \hat{R})^{\eta_\gamma}$, $A_5 = \varphi^{\eta_u}$. Finally, it computes $c = H(\Theta_1 \cdots \Theta_5 \| A_1 \cdots A_5 \| M)$, computes $\theta_x = \eta_x + cx$, $\theta_z = \eta_z + cz$, $\theta_y = \eta_y + cy$, $\theta_d = \eta_d + cd$, $\theta_u = \eta_u + cu$, $\theta_\beta = \eta_\beta + c\beta$, $\theta_\gamma = \eta_\gamma + c\gamma$ and outputs $\sigma = (c, \Theta_1 \cdots \Theta_5, \theta_x \cdots \theta_\gamma, M)$ as a group signature.

PAGS.Verify (*gpk*, *param*, σ):

- ***Revocation Check***: For each element in C-RL list, the verifier checks whether $e(\Theta_3 / cert, \psi) \stackrel{?}{=} e(h, \Theta_2)$. If the above equation stands, the verifier outputs "revoked" and aborts. For each element in G-RL list, the verifier checks whether $\Theta_4 \stackrel{?}{=} e(\Theta_5, \hat{D}_{ID})$. If it stands, the verifier outputs "revoked" and aborts.
- ***Signature Check***: The verifier computes $\hat{A}_1 = w^{\theta_d} \Theta_1^{-c}$, $\hat{A}_2 = \psi^{\theta_d} \Theta_2^{-c}$, $\hat{A}_3 = e(h, mpk)^{\theta_d} \ e(\hat{g}, \psi)^{\theta_z} e(\varphi, \psi)^{\theta_x} e(h, \psi)^{\theta_\beta} e(\Theta_3, \psi)^{-\theta_y} e(\Theta_3, mpk)^{-c} \ e(g, \psi)^c$, $\hat{A}_4 = e(\varphi, \hat{R})^{\theta_\gamma} \Theta_4^{-c}$, $\hat{A}_5 = \varphi^{\theta_u} \Theta_5^{-c}$ and $\hat{c} = H(\Theta_1 \cdots \Theta_5 \| \hat{A}_1 \cdots \hat{A}_5 \| M)$. Then it checks whether $\hat{c} = c$. If the above equation stands, the verifier outputs "valid". Otherwise, it outputs "invalid".

PAGS.Trace (*gpk*, *gsk*, *param*, σ): First, the GM checks whether the group signature σ is valid. If the above equation does not hold, then it aborts. Otherwise, it computes $cert_{ID^*} = \Theta_3 / \Theta_1^{s_t}$ and obtains the signer's public key by searching the GR list (criteria: $cert_{ID^*} = GR[ID].cert_{ID}$). Second, the GM generates a zero-knowledge proof π as follows to prove that the result of the PAGS.Trace algorithm is right: It picks a random value $\hat{d}$ from $\mathbb{Z}_p^*$ and sets $T_\pi = cert_{ID} \cdot g^{\hat{d}/(s+y_{ID})}$. It needs to generate a zero-knowledge proof π to prove that $e(T_\pi, mpk\psi^{y_{ID^*}}) = e(g^{(1+\hat{d})} \hat{g}^{z_{ID^*}} \varphi^x, \psi)$ holds. It picks random values $j_{\hat{d}}$, j_z, j_y from $\mathbb{Z}_p^*$ and computes $A_\pi = e(g, \psi)^{j_{\hat{d}}} e(\hat{g}, \psi)^{j_z} e(T_\pi, \psi)^{-j_y}$, $c_\pi = H(T_\pi \| A_\pi)$, $J_{\hat{d}} = j_{\hat{d}} + c_\pi \times \hat{d}$, $J_z = j_z + c_\pi \times z$, $J_y = j_y + c_\pi \times y$. It outputs $\pi = (c_\pi, T_\pi, J_{\hat{d}}, J_z, J_y)$. Finally, it outputs N_{ID}'s public key npk_1 and proof π.

PAGS.Judge (*gpk*, npk_1, *param*, σ, π): First, the verifier checks whether the group signature σ is valid. Then, it judges whether the GM has tracked to the correct signer by verifying whether the zero-knowledge proof π is valid. It computes $\hat{A}_\pi = e(g, \psi)^{J_{\hat{d}}} e(\hat{g}, \psi)^{J_z} e(T_\pi, \psi)^{-J_y} \cdot e(T_\pi, mpk)^{-c_\pi} e(npk_1, \psi)^{c_\pi} e(g, \psi)^{c_\pi}$ and $\hat{c}_\pi = H(T_\pi \| \hat{A}_\pi)$. If $\hat{c}_\pi \stackrel{?}{=} c_\pi$, it outputs "right". Otherwise, it outputs "wrong".

5 Security Proof and Analysis

In this section, the formal security proofs for anonymity and the security analysis for traceability and non-frameability are provided to prove the anonymity, traceability and non-frameability of the PAGS scheme.

5.1 Formal Security Proof

***Theorem* 1:** When the DDH, CDH and DBDH assumption hold, the PAGS scheme satisfies anonymity.

***Lemma* 1:** If there is a PPT adversary $\mathcal{A}$ that can break the anonymity of the PAGS scheme with the advantage ϵ, then we can construct an algorithm $\mathcal{B}_{DDH}$ that can solve the DDH problem with advantage $\epsilon/2(2/q_N - q_s q_h / p - n/q_s)$, or an algorithm $\mathcal{B}_{CDH}$ that can solve the CDH problem with advantage $\epsilon/2$, or an algorithm $\mathcal{B}_{DBDH}$ that can solve the DBDH problem with advantage $\epsilon/2(2/(q_N + 1) - q_s q_h / p)$.

***Proof*:** The group signature in our PAGS scheme is made up of $c, \Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \theta_x, \theta_z, \theta_y, \theta_d, \theta_u, \theta_\beta, \theta_\gamma$. First, the hash value c is random and $\mathcal{A}$ can't obtain any information about the signer's identity. Second, $\theta_x, \theta_z, \theta_y, \theta_d, \theta_u, \theta_\beta, \theta_\gamma$ are calculated from random values and $\mathcal{A}$ can't obtain the signer's identity too. Therefore, we only need to consider whether $\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5$ will reveal the identity of the signer.

We consider three kinds of adversaries. Details are as follows.

Type I adversary $\mathcal{A}_1$: The target of $\mathcal{A}_1$ is (Θ_1, Θ_3). The proofs will be reduced to the DDH assumption. We construct an algorithm $\mathcal{B}_{DDH}$ to solve the DDH problem. The input of $\mathcal{B}$ is (w, w^a, w^b, Z) where $a, b \in \mathbb{Z}_p^*$ and Z is $w^{ab}(p_1 = 1)$ or a random value from $\mathbb{G}_1(p_1 = 0)$.

KGen: $\mathcal{B}$ picks $\tilde{s}$, $\tilde{r}$ from $\mathbb{Z}_p^*$, sets $s_t = a$, $gsk = (\tilde{s}, \tilde{r}, a)$ and $gpk = (\psi^{\tilde{s}}, \psi^{\tilde{r}}, w^a)$. $\mathcal{B}$ guesses $\mathcal{A}$'s target ID^*.

Query Phase: $\mathcal{B}_{DDH}$ is able to respond $\mathcal{A}_1$'s queries except Open oracle because he has group secret keys $(\tilde{s}, \tilde{r})$, but not s_t.

- *Sign query phase*: if $ID \neq ID^*$, $\mathcal{B}_{DDH}$ responds as usual. Otherwise, $\mathcal{B}_{DDH}$ sets $d = b$, $\Theta_1 = w^b$, $\Theta_3 = Z \cdot cert_{ID^*}$ and picks $\Theta_2 \xleftarrow{\$} \mathbb{G}_2$, $\Theta_4 \xleftarrow{\$} \mathbb{G}_T$, $\Theta_5 \xleftarrow{\$} \mathbb{G}_1$. Then, $\mathcal{B}_{DDH}$ employs the simulation of signature proof of knowledge to generate σ^*. When the simulation fails (hash function collision happens), $\mathcal{B}_{DDH}$ aborts. Besides, $\mathcal{B}_{DDH}$ records the number of the Sign queries for ID^* and denotes it by n.
- *Open query phase*: $\mathcal{B}_{DDH}$ can't respond Open queries as usual because he doesn't have s_t. $\mathcal{B}_{DDH}$ can respond $\mathcal{A}_1$'s Open queries as follows: When $\mathcal{A}_1$ makes an Open query, $\mathcal{B}_{DDH}$ obtains all nodes' $cert_{ID}$ from GR list and checks whether $e(\Theta_3 / cert_{ID}, \psi) \stackrel{?}{=} e(h, \Theta_2)$. Once the above equation holds, $\mathcal{B}$ outputs ID as the response. It is worth noting that, if the adversary $\mathcal{A}_1$ make Open queries on the signature that is signed by N_{ID^*}, $\mathcal{B}_{DDH}$ aborts because he can't response.

Challenge Phase: The adversary $\mathcal{A}_1$ inputs M, ID_0, ID_1. $\mathcal{B}$ picks $\hat{b} \in \{0,1\}$. If $ID_b \neq ID^*$, $\mathcal{B}_{DDH}$ aborts. Otherwise, it generates σ_b by following the steps in the Sign query phase.

Test: $\mathcal{A}_1$ outputs its guess $\hat{b}^*$. If $\hat{b} = \hat{b}^*$, $\mathcal{B}_{DDH}$ outputs $Z = w^{ab}(p_1^* = 1)$. Otherwise, it outputs $Z \xleftarrow{\$} \mathbb{G}_1(p_1^* = 0)$.

Now, we consider the advantage of $\mathcal{B}_{DDH}$ (i.e. $Pr[p_1 = p_1^*] - 1/2$). We assume that $Pr[Z = w^{ab}] = Pr[Z \xleftarrow{\$} \mathbb{G}_1] = 1/2$. E stands for the event that $\mathcal{B}_{DDH}$ aborts. If the game aborts, $\mathcal{B}_{DDH}$ will not be able to answer based on the adversary A's answer. Thus, we can obtain that $Pr[p_1 = p_1^* \mid E] = 1/2$. When the game doesn't abort and Z is an random value (i.e. $p_1 = 0$), the challenged signature has nothing to do with node ID_b, so the probability of the adversary $\mathcal{A}_1$ correctly guessing $\hat{b}$ is $1/2$ (i.e. $Pr[p_1^* = 0 \mid \neg E \wedge p_1 = 0] = 1/2$). When the game doesn't abort and $Z = w^{ab}$ (i.e. $p_1 = 1$), the algorithm $\mathcal{B}_{DDH}$ executes the experiment perfectly and the probability of the adversary $\mathcal{A}_1$ correctly guessing $\hat{b}$ is $1/2 + \epsilon$ (i.e. $Pr[p_1^* = 1 \mid \neg E \wedge p_1 = 1] = 1/2 + \epsilon$).

$$
\begin{aligned}
Adv &= Pr[p_1 = p_1^*] - \frac{1}{2} \\
&= Pr[p_1 = p_1^* \mid E] Pr[E] + Pr[p_1 = p_1^* \mid \neg E] Pr[\neg E] - \frac{1}{2} \\
&= \frac{1}{2} - \frac{1}{2} Pr[\neg E] + Pr[\neg E]\left(\frac{1}{2} \times \frac{1}{2} + \left(\frac{1}{2} + \epsilon\right) \times \frac{1}{2}\right) - \frac{1}{2} \\
&= \frac{\epsilon}{2} Pr[\neg E]
\end{aligned}
$$

Then, we talk about the probability of the event $\neg E$ happening. The event $\neg E$ will occur only if all of the following conditions hold.

(1) $\mathcal{A}_1$ chooses the node N_{ID^*} as its target. The probability is $2/q_N$ where q_N is the number of AddHN queries (i.e. the number of honest nodes).

(2) No termination occurs during the simulation of signature proof of knowledge. The simulation fails when a hash collision occurs. The probability of termination is $q_s q_h / p$ where q_s is the number of the Sign queries and q_h is the number of the Hash queries.

(3) No termination occurs in Open queries. The probability of termination is n/q_s.

Therefore, we can obtain that $Pr[\neg E] = 2/q_N - q_s q_h / p - n/q_s$ and $Adv_{\mathcal{B}_{DDH}} = \epsilon/2(2/q_N - q_s q_h / p - n/q_s)$.

Type II adversary $\mathcal{A}_2$: The target of $\mathcal{A}_2$ is (Θ_2, Θ_3). The proofs will be reduced to the CDH assumption. $\mathcal{A}_2$ can break the anonymity of the PAGS scheme by executing following steps:

(1) $\mathcal{A}_2$ chooses N_{ID_0} and N_{ID_1} as his targets.

(2) $\mathcal{A}_2$ tries to obtain N_{ID_0}'s $cert_{ID_0}$ or N_{ID_1}'s $cert_{ID_1}$.

(3) For the σ_b that was output by $\mathcal{B}_{CDH}$, $\mathcal{A}_2$ checks whether $e(\Theta_3 / cert, \psi) \stackrel{?}{=} e(h, \Theta_2)$ to output $\hat{b}$.

$\mathcal{A}_2$ can obtain $cert_{ID_i}$'s ciphertext $(i \in \{0,1\})$ via LeakInfo queries and ReadReg queries. $\mathcal{A}_2$ can obtain $cert_{ID_i}$ from ciphertexts only if he breaks PUFs to obtain N_{ID_i}'s secret keys or solves CDH problem (i.e. computes ψ^{xs} according to ψ^s and ψ^x). Therefore, we can get that $\epsilon = Adv_{\mathcal{A}_2}^{PUFs} + 2 \times Adv_{\mathcal{B}_{CDH}}$ and $Adv_{\mathcal{B}_{CDH}} = \epsilon/2$ because $Adv_{\mathcal{A}_2}^{PUFs}$ is negligible.

Type III adversary $\mathcal{A}_3$: The target of $\mathcal{A}_3$ is (Θ_4, Θ_5). The proofs will be reduced to the DBDH assumption. First, we construct an algorithm $\mathcal{B}_{DBDH}$ to solve the DBDH problem. The input of $\mathcal{B}_{DBDH}$ is $(\varphi, \psi, \varphi^a, \varphi^b, \psi^c, Z)$ where $\{a,b,c\} \in \mathbb{Z}_p^*$, $\{\varphi^a, \varphi^b\} \in \mathbb{G}_1$, $\psi^c \in \mathbb{G}_2$ and Z is $e(\varphi,\psi)^{abc}$ $(p_1 = 1)$ or a random value from $\mathbb{G}_T$ $(p_1 = 0)$.

KGen: $\mathcal{B}$ picks $\tilde{s}$, $\tilde{s}_t$ from $\mathbb{Z}_p^*$, sets $\hat{r} = c$, $gsk = (\tilde{s}, \tilde{s}_t, c)$ and $gpk = (\psi^{\tilde{s}}, \psi^c, w^{\tilde{s}_t})$. $\mathcal{B}$ guesses $\mathcal{A}$'*s* target ID^*.

Query Phase: $\mathcal{B}_{DDH}$ is able to respond $\mathcal{A}_1$'*s* queries except Revoke queries because he has group secret keys $(\tilde{s}, \tilde{s}_t)$, but not $\hat{r}$.

- *AddHN query phase*: $\mathcal{B}_{DBDH}$ can't response Revoke queries, because he doesn't possess secret key $\hat{r}$ (i.e. c). Thus, an additional value $(\psi^{\hat{r}})^x$ for each node needs to be generated in AddHN queries. Besides, $\mathcal{B}_{DBDH}$ inserts a node ID^* into the HN list and sets its secret key $x = a$, public key φ^a.
- *Sign query phase*: If $ID \neq ID^*$, $\mathcal{B}_{DBDH}$ responds as usual. Otherwise, $\mathcal{B}_{DBDH}$ sets $u = b$, $\Theta_4 = Z$, $\Theta_5 = \varphi^b$ and picks $\Theta_1 \overset{\$}{\leftarrow} \mathbb{G}_1$, $\Theta_2 \overset{\$}{\leftarrow} \mathbb{G}_2$, $\Theta_3 \overset{\$}{\leftarrow} \mathbb{G}_1$. Then, $\mathcal{B}_{DBDH}$ employs the simulation to generate σ^*. When the simulation fails (hash function collision happens), $\mathcal{B}_{DBDH}$ aborts.
- *Revoke query phase*: $\mathcal{B}_{DBDH}$ can response $\mathcal{A}_3$'*s* queries, because all node's $(\psi^{\hat{r}})^x$ are generated in AddHN queries and he can obtain all node's *cert*.

Challenge Phase: The adversary $\mathcal{A}_3$ inputs M, ID_0, ID_1. $\mathcal{B}$ picks $\hat{b} \in \{0,1\}$. If $ID_b \neq ID^*$, $\mathcal{B}_{DBDH}$ aborts. Otherwise, it generates σ_b by following the steps in the Sign query phase.

Test: $\mathcal{A}_3$ outputs its guess $\hat{b}^*$. If $\hat{b} = \hat{b}^*$, $\mathcal{B}_{DBDH}$ outputs $Z = e(\varphi,\psi)^{abc}$ $(p_1^* = 1)$. Otherwise, it outputs $Z \overset{\$}{\leftarrow} \mathbb{G}_T$ $(p_1^* = 0)$.

Now, we consider the advantage of $\mathcal{B}_{DBDH}$ (i.e. $Pr[p_1 = p_1^*] - 1/2$). We assume that $Pr[Z = e(\varphi,\psi)^{abc}] = Pr[Z \overset{\$}{\leftarrow} \mathbb{G}_T] = 1/2$. E stands for the event that $\mathcal{B}_{DBDH}$ aborts. It is easy to conclude that $Adv_{\mathcal{B}_{DBDH}} = Pr[\neg E] \times \epsilon/2$. The derivation is the same as above. Then, we talk about the probability of the event $\neg E$ happening. The event $\neg E$ will occur only if all of the following conditions hold.

(1) $\mathcal{A}_3$ chooses the node N_{ID^*} as its target. The probability is $2/(q_N + 1)$ where q_N is the number of AddHN queries.

(2) No termination occurs during the simulation of signature proof of knowledge. The simulation fails when a hash collision occurs. The probability of termination is $q_s q_h / p$ where q_s is the number of the Sign queries and q_h is the number of the Hash queries.

Therefore, $Pr[\neg E] = 2/(q_N + 1) - q_s q_h / p$ and $Adv_{\mathcal{B}_{DBDH}} = \epsilon/2(2/(q_N + 1) - q_s q_h / p)$.

5.2 Security Analysis

***Traceability*:** The adversary $\mathcal{A}$'*s* attack on the traceability of the PAGS scheme is essentially the forgery of the group certificate. If $\mathcal{A}$ can forge a group membership certificate for an invalid node, then $\mathcal{A}$ can control the invalid node to generate a valid group signature which can be traced to an invalid node. In the PAGS scheme, the GM generates a BBS+ signature as the node's certificate. And BBS+ signature is unforgeable if the *q*-SDH assumption holds [26].

***Non-frameability*:** Only the adversary $\mathcal{A}$ generates a valid signature proof of knowledge without a valid node's private keys, can he break the non-frameability of the PAGS scheme. And the signature proof of knowledge is unforgeable in the random oracle [27]. Therefore, if $\mathcal{A}$ can break the non-frameability of the PAGS scheme, then he can break the signature proof of knowledge.

6 Performance

In this section, we perform experiments of the PAGS scheme with the PBC library on the type d159 curve. The experiments are done on two platforms, namely a raspberry pi 4 and a desktop. According to the practical scenarios and system model, we test the running time of terminal nodes on the raspberry pi 4 and test the running time of verifiers on the desktop.

Hardware environment:

- Raspberry Pi 4 Model B with 2G RAM and 32G storage.
- A desktop with Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz, 6G RAM and 20G storage. It's worth noting that we perform experiments on a virtual machine. So only four of the CPU's six cores are used.

***Software environment*:** Ubuntu 18.04.1 with gcc 7.5.0, gmp 6.2.0 and pbc 0.5.14.

We test the benchmark of each operation on two platforms. And the symbols used in this section are described as follows: E_i stands for the exponential operation on group $\mathbb{G}_i$, *Pair* stands for the bilinear pairing operation, *Hash* stands for the hash function operation, *Enc* stands for the symmetric encryption operation and *Dec* stands for the symmetric decryption operation. Note that, we ignore the time cost of PUFs,

hash function, symmetric encryption and decryption because they are very small. In our experiment, the time cost of GM and terminal nodes are tested on the raspberry pi 4, the time cost of the verifier is tested on the desktop. The amount of each operation in each phase is given in Table 1. The experiment results are shown in Table 2. These results demonstrate that the proposed PAGS scheme is suitable for terminal nodes.

Table 1. The amount of each operation in each phase

Algorithm	Operation	
	Group Manager	Terminal Node
Join	$2E_1+2E_2+2Hash$	$E_1+E_2+2Hash$
Issue	$3E_1+E_2+Enc$	$2E_2+2E_T+Hash+Pair+Dec$
Revoke	$R\times E_2$	N/A
Sign	N/A	$4E_1+3E_2+7E_T+Pair+Hash$
Trace	$2E_1+3E_T+Pair+Hash$	N/A
	Verifier	
Verify	$4E_1+2E_2+7E_T+(2+2n_C+n_G)Pair+Hash$	
Judge	$6E_T+3Pair+Hash$	

Note. R is the number of terminal nodes that is about to be revoked via public key revocation. n_C (n_G) is the length of list C-RL(G-RL).

Table 2. Experiment Results on Raspberry Pi 4 and Desktop

Algorithm	Time Cost (ms)	
	Group Manager (Raspberry Pi 4)	Terminal Node (Raspberry Pi 4)
Join	22.41	11.20
Issue	13.54	33.31
Revoke	10.03 R	N/A
Sign	N/A	60.57
Trace	18.10	N/A
	Verifier	
Verify	15.35+3.20 n_{revoke}	
Judge	10.18	

Note. n_{revoke} is the number of the revoked terminal nodes. In this experiment, we set $n_C = n_G = \frac{n_{revoke}}{2}$.

7 Conclusion

In this paper, we design a PUF-based anonymous group signature (PAGS) for edge computing. Then, we provide the formal security model and proofs to prove that the PAGS scheme satisfies anonymity. Furthermore, the security analysis is given to discuss the traceability and non-frameability of the PAGS scheme. Finally, the experiments are performed on a Raspberry Pi 4 and a desktop. The results demonstrate that the PAGS scheme can be used in resource-constrained devices.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants No. U1836115, No. 61672295, No. 61922045, No. 61672290, No. 61877034, the Natural Science Foundation of Jiangsu Province under Grant No. BK20181408, the State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications under Grant SKLNST-2019-2-02, the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004, the CICAEET fund, and the PAPD fund.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13-16.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp. 637-646, October, 2016.

[3] D. Chaum, E. Van Heyst, Group Signatures, *Workshop on the Theory and Application of Cryptographic Techniques*, Brighton, United Kingdom, 1991, pp. 257-265.

[4] A. Kiayias, M. Yung, Group Signatures with Efficient Concurrent Join, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Aarhus, Denmark, 2005, pp. 198-214.

[5] A. Kiayias, M. Yung, Secure Scalable Group Signature with Dynamic Joins and Separable Authorities, *International Journal of Security and Networks*, Vol. 1, No. 1-2, pp. 24-45, September, 2016.

[6] J. Y. Hwang, L. Chen, H. S. Cho, D. Nyang, Short Dynamic Group Signature Scheme Supporting Controllable Linkability, *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 6, pp. 1109-1124, June, 2015.

[7] K. Emura, T. Hayashi, A Light-weight Group Signature Scheme with Time-token Dependent Linking, *Lightweight Cryptography for Security and Privacy*, Bochum, Germany, 2015, pp. 37-57.

[8] F. Zhang, K. Kim, ID-based Blind Signature and Ring Signature from Pairings, *International Conference on the Theory and Application of Cryptology and Information Security*, Queenstown, New Zealand, 2002, pp. 533-547.

[9] D. Khader, Attribute based Group Signatures, *IACR Cryptology ePrint Archive*, Vol. 2007, pp. 159, 2007.

[10] C. Ma, J. Ao, Certificateless Group Oriented Signature Secure Against Key Replacement Attack, *International Journal of Network Security*, Vol. 12, No. 1, pp. 1-6, January, 2011.

[11] J. Y. Hwang, S. Lee, B. H. Chung, H. S. Cho, D. Nyang,

Short Group Signatures with Controllable Linkability, *2011 Workshop on Lightweight Security & Privacy: Devices, Protocols, and Applications*, Istanbul, Turkey, 2011, pp. 44-52.

[12] Y. Hao, Y. Cheng, K. Ren, Distributed Key Management with Protection against RSU Compromise in Group Signature based VANETs, *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, New Orleans, LA, USA, 2008, pp. 1-5.

[13] H. Feng, J. Liu, Q. Wu, T. Xu, Two-Layer Group Signature and Application to E-Cash, 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSci Tech), Fukuoka, Japan, 2019, pp. 193-200.

[14] H. Tian, F. Nan, H. Jiang, C. C. Chang, J. Ning, Y. Huang, Public Auditing for Shared Cloud Data with Efficient and Secure Group Management, *Information Sciences*, Vol. 472, pp. 107-125, January, 2019.

[15] V. Kumar, H. Li, J. M. Park, K. Bian, Y. Yang, Group Signatures with Probabilistic Revocation: A Computationally-scalable Approach for Providing Privacy-preserving Authentication, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, Colorado, USA, 2015, pp. 1334-1345.

[16] M. Backes, L. Hanzlik, J. Schneider-Bensch, Membership Privacy for Fully Dynamic Group Signatures, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, United Kingdom, 2019, pp. 2181-2198.

[17] S. Canard, A. Georgescu, G. Kaim, A. Roux-Langlois, J. Traoré, Constant-Size Lattice-Based Group Signature with Forward Security in the Standard Model, *International Conference on Provable Security*, Singapore, 2020, pp. 24-44.

[18] J. Cui, L. Wei, J. Zhang, Y. Xu, H. Zhong, An Efficient Message-authentication Scheme based on Edge Computing for Vehicular Ad Hoc Networks, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 5, pp. 1621-1632, May, 2019.

[19] X. Jia, D. He, N. Kumar, K. K. R. Choo, A Provably Secure and Efficient Identity-based Anonymous Authentication Scheme for Mobile Edge Computing, *IEEE Systems Journal*, Vol. 14, No. 1, pp. 560-571, March, 2020.

[20] J. Wang, L. Wu, K. K. R. Choo, D. He, Blockchain-based Anonymous Authentication with Key Management for Smart Grid Edge Computing Infrastructure, *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 3, pp. 1984-1992, March, 2020.

[21] S. Jangirala, A. K. Das, A. V. Vasilakos, Designing Secure Lightweight Blockchain-enabled RFID-based Authentication Protocol for Supply Chains in 5G Mobile Edge Computing Environment, *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 11, pp. 7081-7093, November, 2020.

[22] Y. Li, Q. Cheng, X. Liu, X. Li, A Secure Anonymous Identity-Based Scheme in New Authentication Architecture for Mobile Edge Computing, *IEEE Systems Journal*, Vol. 15, No. 1, pp. 935-946, March, 2021.

[23] T. Gao, Y. Li, N. Guo, I. You, An Anonymous Access Authentication Scheme for Vehicular Ad Hoc Networks under Edge Computing, *International Journal of Distributed Sensor Networks*, Vol. 14, No. 2, pp. 1-15, February, 2018.

[24] S. Zhang, J. H. Lee, A Group Signature and Authentication Scheme for Blockchain-Based Mobile-Edge Computing, *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp. 4557-4565, May, 2020.

[25] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, Foundations of Fully Dynamic Group Signatures, *International Conference on Applied Cryptography and Network Security*, London, United Kingdom, 2016, pp. 117-136.

[26] M. H. Au, W. Susilo, Y. Mu, Constant-size Dynamic k-TAA, *International Conference on Security and Cryptography for Networks*, Maiori, Italy, 2006, pp. 111-125.

[27] M. H. Au, W. Susilo, Y. Mu, and S. S. Chow, Constant-size Dynamic K-times Anonymous Authentication, *IEEE Systems Journal*, Vol. 7, No. 2, pp. 249-261, June, 2013.

Biographies

Junqing Lu received the B.E. degree in software engineer from Nanjing University of Information Technology, China, in 2019. Since 2019, he has been studying for the M.E. degree at Nanjing University of Information Technology. His research interests include group signature, authentication, and authenticated key exchange.

Jian Shen received the M.E. and Ph.D. degrees in computer science from Chosun University, South Korea, in 2009 and 2012, respectively. Since 2012, he has been a Professor in the School of Computer and Software at Nanjing University of Information Science and Technology, Nanjing, China. His research interests include public cryptography, cloud computing and security, data auditing and sharing, and information security systems.

Chin-Feng Lai received the Ph.D. degree in engineering science from National Cheng Kung University, Tainan,Taiwan, in 2008. Since 2016, he has been an Associate Professor of Engineering Science, National Cheng Kung University, Tainan. His research focuses on Internet of Things, body sensor networks, e-healthcare, mobile cloud computing, etc.

Fei Gao received Ph.D. degree in cryptography from Beijing University of Posts and Telecommunications (BUPT) in 2007. Now, he is a professor in Institute of Network Technology, BUPT. His research interests include cryptography, information security, and quantum algorithm. Prof. Gao is a member of the Chinese Association for Cryptologic Research.