Hindawi
Security and Communication Networks
Volume 2018, Article ID 2595273, 13 pages
https://doi.org/10.1155/2018/2595273

Research Article

New Certificateless Aggregate Signature Scheme for Healthcare Multimedia Social Network on Cloud Environment

Libing Wu,[1,2] Zhiyan Xu,[1,3] Debiao He,[1,4] and Xianmin Wang[5]

[1]*The Computer School, Wuhan University, Wuhan, China*
[2]*The Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei, China*
[3]*The College of Computer, Hubei University of Education, Wuhan, China*
[4]*The State Key Laboratory of Cryptology, Beijing, China*
[5]*The School of Computer Science and Educational Software, Guangzhou University, Guangzhou, China*

Correspondence should be addressed to Zhiyan Xu; csxzy@whu.edu.cn

Received 2 March 2018; Accepted 29 April 2018; Published 13 June 2018

Academic Editor: Ilsun You

With the application of sensor technology in the field of healthcare, online data sharing in healthcare industry attracts more and more attention since it has many advantages, such as high efficiency, low latency, breaking the geographical location, and time constraints. However, due to the direct involvement of patient health information, the privacy and integrity of medical data have become a matter of much concern to the healthcare industry. To retain data privacy and integrity, a number of digital signature schemes have been introduced in recent years. Unfortunately, most of them suffer serious security attacks and do not perform well in terms of computation overhead and communication overhead. Very recently, Pankaj Kumar *et al.* proposed a certificateless aggregate signature scheme for healthcare wireless sensor network. They claimed that their signature scheme was able to withstand a variety of attacks. However, in this paper, we find that their scheme fails to achieve its purpose since it is vulnerable to signature forgery attack and give the detailed attack process. Then, we propose a new certificateless aggregate signature scheme to fix the security flaws and formally prove that our proposed scheme is secure under the computationally hard Diffie-Hellman assumption. Security analysis and performance evaluation demonstrate that the security of our proposal is improved while reducing the computation cost. Compared with Pankaj Kumar *et al.*'s scheme, our proposed scheme is more efficient and suitable for the healthcare wireless sensor networks (HWSNs) to maintain security at various levels.

1. Introduction

Wireless sensor network (WSN) has been widely used in many fields such as retail, entertainment, medicine, tourism, industry, and emergency management [1], and it provides many new opportunities for traditional applications, of which healthcare is one of them. Researchers have invented many sensor-based miniature medical devices to replace the traditional thousands of wires connected to hospital equipment and to increase the mobility of devices. The combination of computer network technology and medical field makes the healthcare industry have more broad prospects for development [2].

The application of wireless sensor network technology is mainly divided into two categories: medical applications and nonmedical applications [3]. There are two main types of devices used in medical applications: wearable devices and implanted devices. The first category refers to medical devices that are used on or near the surface of a human body, and the human body can move with the wearable devices. The second category refers to medical devices injected in/with the human body.

As shown in Figure 1, there is a general healthcare wireless sensor network (HWSN) architecture, which consists of the following five components [4]: sensor, central control unit, patient, cloud based network, and healthcare professional. The medical sensor node implanted on the patient's body, using air as a transmission medium, can transmit patient's health data to a remote central control unit (CCU) for further processing, then the health data is sent to the healthcare

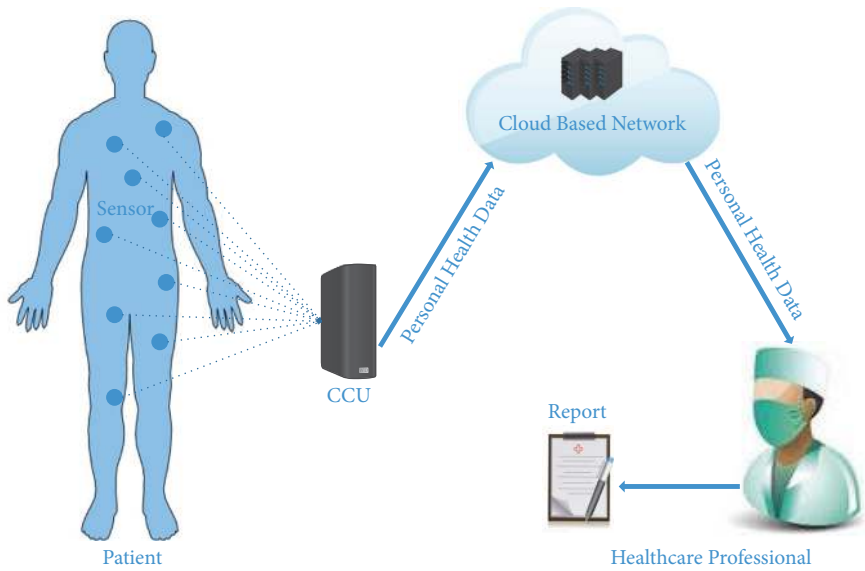

Figure 1: A general healthcare wireless sensor network architecture.

professional by CCU via Internet, and the patient's medical report is further generated.

In the HWSN, information is transmitted from medical sensor devices to the healthcare professional who analyzes the medical information and further provides a suitable solution. If the attacker modifies the medical message halfway, the healthcare professional could make a wrong diagnosis based on the modified message, which can be very dangerous to human life. Because of the direct involvement of patient health information, it is of crucial importance to address the issue of privacy and integrity of personal health data [5–7].

Motivated with the above scenario, many digital signature schemes are proposed for healthcare wireless sensor network (HWSN) to protect the privacy and integrity of patient medical information. In this paper, we first review Pankaj Kumar*et al.*'s certificateless aggregate signature (CL-AS) scheme [8] and point out a previously undiscovered security flaw in the scheme; that is, we reveal that their proposed scheme suffers the signature forgery attack. We then propose a new CL-AS scheme for the issues of security and privacy in HWSN.

1.1. Our Research Contributions. In this paper, we propose a new CL-AS scheme which could better protect the integrity and privacy of data in HMSN. The main contributions of this paper are summarized as below:

(i) *Firstly*, we identify a security weakness against Pankaj Kumar *et al.*'s CL-AS scheme for HWSN.

(ii) *Secondly*, we redefine the architecture of a HWSN, which is more close to the actual application environment.

(iii) *Thirdly*, we propose a CL-AS scheme for HWSN to mend this security flaw, and our new scheme can satisfy the security requirements.

(iv) *Finally*, we prove the security of our proposed CL-AS scheme and show that it can improve the security while reducing the computation cost compared with Pankaj Kumar *et al.*'s CL-AS scheme.

1.2. Organization of the Paper. The remainder of this paper is organized as below. Section 2 introduces the related work. Section 3 presents the problem statements associated with this paper and then briefly reviews the CL-AS scheme for HWSN in Section 4. In Section 5, we demonstrate an attack against Pankaj Kumar *et al.*'s CL-AS scheme for the HWSN. Furthermore, we present details of the proposed CL-AS scheme in Section 6. In Sections 7 and 8, the security proof and performance analysis of our scheme are described later. Finally, we give a summary of this paper in the last section.

2. Related Work

In the traditional PKI-based public key cryptography (PKC), as the number of users increases, PKC will face a variety of certificate management issues such as certificate distribution, storage, revocation, and high computational cost [11].

Although identity-based public key cryptography (IBC) [12, 13] can solve the problem of certificate management existing in PKC, it has inherent key escrow issue. This is because the user's private key is generated by the key generation center (KGC) based on the user's identity; that is, KGC can access any user's private key in IBC.

To solve the above problems, Al-Riyami *et al.* proposed certificateless public key cryptosystem (CL-PKC) [14]. Because it does not use certificates and the private key is generated both by KGC and by the user himself, it can solve certificate management issue in PKC and the key escrow issue in IBC. Since Al-Riyami *et al.* introduced the notion of CL-PKC [14], it has attracted more and more research attention, and many certificateless signature (CLS) schemes [15–21] have been introduced by researchers.

Huang *et al.* [15] proved that the CLS scheme proposed in [14] is vulnerable to the public key replacement attack and further proposed an improved certificateless signature scheme to solve this weakness. Similarly, Li *et al.* [16] also proposed a new CLS scheme to improve the security of the scheme proposed in [17], which is subject to the public key replacement attack as well. For a malicious KGC attack that exists in some certificateless signature schemes, Au *et al.* [18] proposed an enhanced security model that allows malicious KGC to generate key pairs in any way. Nevertheless, the certificateless encryption and signature schemes proposed in [19–21] have been found to be insecure against malicious KGC attack.

Boneh *et al.* proposed the concept of aggregate signature [22] in Eurocrypt 2003. The aggregator can aggregate n different signatures with respect to n messages from n users into a single short signature, which can reduce the bandwidth and computational effort of tiny devices used in HWSN. Therefore, the aggregate signature is a more suitable choice in resource-constrained HWSN.

Combining certificateless public key cryptography with aggregate signature, Gong *et al.* [9] proposed the first CL-AS scheme, but they did not give a formal security proof to the scheme. After pioneer work [9], many CL-AS schemes [10, 23–28] have been proposed for various practical applications. Zhang and Zhang [23] redefined the concept and security model for CL-AS. Furthermore, they put forward a new CL-AS scheme, but their scheme need clock synchronized while generating the aggregate signature, and, more seriously, the scheme has been proved that it cannot resist malicious KGC attack. Xiong *et al.* [24] presented a CL-AS scheme, but He *et al.* [25] showed that their scheme was forgeable and further proposed a new CL-AS scheme. The CL-AS scheme proposed in [10] has been found to be insecure against the malicious-but-passive KGC attack by the researchers in [26–28].

Recently, He and Zeadally [29] present an authentication scheme for the Ambient Assisted Living (AAL) system, which provides technical support for medical monitoring and telehealth services. He *et al.* [30] put forward an efficient certificateless public auditing scheme for cloud-assisted wireless body area networks. Very recently, Pankaj Kumar *et al.* proposed a CL-AS scheme for secure communication in HWSN [8], which is claimed to be able to achieve the message authentication and integrity audit functions while also achieving nonrepudiation and confidentiality. Unfortunately, we find that their CL-AS scheme is insecure and vulnerable to signature forgery attack from a malicious-but-passive KGC.

3. Problem Statement

Bilinear map and related hard problems are first described and then system model of our proposed CL-AS scheme is presented in this section. After that, system components of CL-AS scheme are also described.

3.1. Bilinear Map. Suppose that G_1 and G_2 are two cyclic groups with the same prime order q, where G_1 is an additive cyclic group with a generator P and G_2 is a multiplicative cyclic group. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. For all $P, Q, T \in G_1$, $a, b \in Z_q^*$, and e should satisfy the properties as follows:

(1) Bilinearity: $e(P, Q + T) = e(P, Q)e(P, T)$ and $e(aP, bQ) = e(abP, Q) = e(P, abQ)$.

(2) Nondegeneracy: there exists $P \in G_1$ such that $e(P, P) \neq 1$.

(3) Computability: there exists efficient algorithm to calculate $e(P, Q)$.

3.2. Complexity Assumption

(1) *Computational Diffie-Hellman (CDH) Problem:* Given a generator P of an additive cyclic group G_1 with the order q and a random instance (aP, bP), it is difficult to compute abP, where a and b are unknown.

(2) *Computational Diffie-Hellman (CDH) Assumption:* There does not exist adversary A, can solve the *CDH* problem in probabilistic polynomial time with a nonnegligible probability ϵ, where $\epsilon > 0$ is a very small number.

3.3. System Model. The architecture of our healthcare wireless sensor network is shown in Figure 2. There are five entities in the framework of a healthcare wireless sensor network: medical sensor node (MSN), medical server (MS), authorized healthcare professional (AHP), signature aggregator (SA), and aggregate signature verifier (ASV). Each entity is specifically defined as follows:

(1) *Medical sensor node.* Medical sensor node is a resource-limited medical small device on patient's body belonging to the Care-District. Let ID_i denote the identity and (sk_i, pk_i) denote the key pair of the sensor node. Each sensor node can use its private key to generate a signature for the relevant message and send the signature to the signature aggregator.

(2) *Medical server.* Medical server is a device with strong computing power and plenty of storage space, which can handle a large amount of data received from sensors. It transmits the processed patient's medical information to the AHP. In addition, it is responsible for generating system parameters *params*, its own key pair (s, MS_{pub}), and the partial private key ppk_i for each sensor node corresponding to its identity and then secretly sends ppk_i to the sensor node.

(3) *Healthcare professional.* Healthcare professional refers to an authorized medical personnel who provides patients with appropriate prescriptions by analyzing the data information sensed by the sensors.

(4) *Aggregator.* Aggregator refers to a certain computing power of device. It is responsible for collecting a single signature from Care-District and then generating an aggregate signature and sending it to the MS. Suppose that each Care-District contains one aggregator and many sensors.

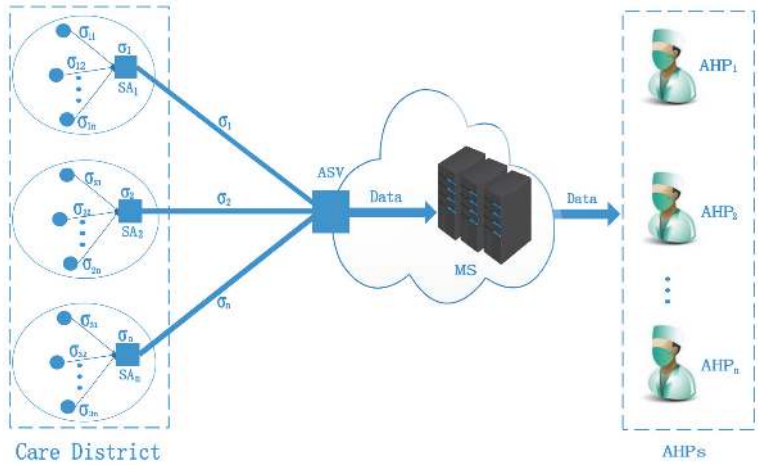

Figure 2: The architecture of our healthcare wireless sensor network.

(5) *Aggregate signature verifier.* Aggregate signature verifier refers to a certain computing power of equipment. It is responsible for verifying an aggregate signature from different Care-District and then outputting a verification result.

3.4. System Components. Our CL-AS scheme is a collection of the following seven polynomial time algorithms as below:

(1) $Setup(1^k) \rightarrow (params, s, MS_{pub})$ is a probabilistic algorithm executed by the MS, where k is a security parameter, $params$ is the system parameters, (s, MS_{pub}) is the key pair of MS, that is, s is the master secret key, and MS_{pub} is the master public key.

(2) $Partial\text{-}Private\text{-}Key\text{-}Gen\ (params, s, MS_{pub}ID_i) \rightarrow ppk_i$ is a probabilistic algorithm executed by the MS, where $params$ is the system parameters, (s, MS_{pub}) is the key pair of MS, $ID_i \in \{0,1\}^*$ is a MSN's identity, and ppk_i is the partial private key corresponding to the identity ID_i of the MSN.

(3) $User\text{-}Key\text{-}pair\text{-}Gen(params, ppk_i) \rightarrow (sk_i, pk_i)$ is a randomized algorithm executed by the MSN with identity ID_i, where $params$ is the system parameters, (s, MS_{pub}) is the key pair of MS, and (sk_i, pk_i) is the key pair of the MSN with the identity ID_i.

(4) $Sign(params, (sk_i, pk_i), \Delta, ID_i, m_i) \rightarrow \sigma_i$ is a randomized algorithm executed by the signer, where $params$ is the system parameters, (sk_i, pk_i) is the key pair of the signer, Δ is the state information, ID_i is the signer's identity, m_i is the message, and σ_i is the signature on the message m_i.

(5) $Verify(params, \Delta, ID_i, m_i, pk_i, \sigma_i) \rightarrow$ {"0", "1"} is a probabilistic algorithm executed by the verifier, where $params$ is the system parameters, ID_i is the signer's identity, pk_i is the public key of the signer, m_i is the message, and σ_i is the signature on the message m_i, 1 or 0 as outputs to indicate whether the signature σ_i is validated.

(6) $Aggregate(params, ID_i, m_i, pk_i, \sigma_i)_{1 \le i \le n} \rightarrow \sigma$ is a deterministic algorithm executed by the aggregator, where $params$ is the system parameters, ID_i is the signer's identity, pk_i is the public key of the signer, m_i is the message, σ_i is the signature on the message m_i, and σ_i is the signature on the message m_i.

(7) $Aggregate\text{-}Verify\ (params, \Delta, ID_i, m_i, pk_i, \sigma)_{1 \le i \le n} \rightarrow$ {"0", "1"} is a deterministic algorithm executed by the aggregate verifier, where $params$ is the system parameters and σ is the aggregate signature of the message m_i on the identity ID_i with public key pk_i. 1 or 0 act as outputs to indicate whether aggregate signature σ is validated.

3.5. Attack Model. In the attack model, we introduce an adversary $A \in \{A_1, A_2\}$ in our model. A's ultimate goal is to successfully forge the user's signature on the message. A possesses with the following capabilities:

(1) A can access any hash oracle and corresponding queries in the security model.

(2) A_1 simulates an outsider attacker, who cannot obtain the master key but can replace any user's public key with a value of his choice.

(3) A_2 simulates an honest-but-curious MS, who is an insider attacker and has no power to replace any user's public key but can access the system master key.

4. Review of Pankaj Kumar *et al.*'s Scheme

Pankaj Kumar *et al.*'s CL-AS scheme is composed of seven algorithms, i.e., $Setup$, $Partial-Private-Key-Gen$, $Private-Key-Gen$, $Sign$, $Verify$, $Aggregate$, and $Aggregate-Verify$. The scheme details are described as below.

4.1. Setup. By executing the following operations, after entering the security parameter k, the MS generates the system parameter $params$.

(1) Generates two cyclic groups G_1 and G_2 with the same order q, where q is a prime. P being a generator of G_1. $e : G_1 \times G_1 \to G_2$ being a bilinear pairing.

(2) Randomly selects a number $\alpha \in Z_q^*$, computes $MS_{pub} = \alpha P$, and sets α as the master key and MS_{pub} as the public key of MS

(3) Defines three hash functions: $H_1 : \{0,1\} \to G_1$, $H_2 : \{0,1\} \to G_1$, $H_3 : \{0,1\} \to Z_q^*$

(4) Keeps α secret and $params = (G_1, G_2, P, q, e, MS_{pub}, H_1, H_2, H_3)$ public.

4.2. Partial-Private-Key-Gen. By executing the following operations, MS generates the user's partial private key:

(1) Given ID_i as the identity of a MS, it computes $Q_{ID_i} = H(ID_i)$ and $ppk_{ID_i} = \alpha.Q_{ID_i}$ and sets ppk_{ID_i} as the user's partial private key.

(2) It secretly sends ppk_{ID_i} to the corresponding MSN.

4.3. Private-Key-Gen. By executing the following operations, a sensor with the identity ID_i generates its private key and public key:

(1) Selects a random number x_i as the secret value.

(2) Sets $\{ppk_{ID_i}, x_i\}$ as its private key.

(3) Computes $PK_{ID_i} = x_i P$ as its public key.

4.4. Sign. By executing the following operations, a signer with the identity ID_i generates a signature σ_i on the message m_i:

(1) Inputs system parameters $params$, private key ppk_{ID_i}, secret key x_i, state information Δ, and private-public key pair (x_i, PK_{ID_i})

(2) Selects $r_i \in Z_q^*$ randomly and then computes $R_i = r_i P$

(3) Computes $W = H_2(\Delta)$ and $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

(4) Computes $V_i = ppk_{ID_i} + r_i W + h_i x_i MS_{pub}$

(5) Outputs (R_i, V_i) as the signature of message m_i.

4.5. Verify. By executing the following operations, the verifier verifies the signature $\sigma_i = (R_i, V_i)$ of message m_i on identity ID_i:

(1) Inputs the state information Δ

(2) Computes $Q_{ID_i} = H_1(ID_i)$, $W = H_2(\Delta)$ and $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

(3) Verifies

$$e(V_i, P) = e\left(Q_{ID_i} + h_i PK_{ID_i}, MS_{pub}\right) e(R_i, W). \quad (1)$$

(4) If (1) holds, emits 1 and the verifier accepts the signature σ_i; otherwise emits 0 and rejects.

4.6. Aggregate. By executing the following operations, an aggregator generates the aggregate signature σ from user-message-public key-signature pairs $(ID_i, m_i, PK_i, \sigma_i)_{1 \le i \le n}$:

(1) Inputs n tuples $(ID_i, m_i, PK_i, \sigma_i)$, where $1 \le i \le n$

(2) Computes $V = \sum_{i=1}^{n} V_i$

(3) Outputs $\sigma = (R, V)$ as the aggregate signature, where $R = \{R_1, R_1, \ldots, R_n\}$.

4.7. Aggregate-Verify. By executing the following operations, the aggregate verifier verifies the validity of the aggregate signature $\sigma = (R, V)$:

(1) Inputs the state information Δ, the tuples $(ID_i, m_i, PK_i, \sigma_i)_{1 \le i \le n}$, and the aggregate signature $\sigma = (R, V)$

(2) For $1 \le i \le n$, computes $Q_{ID_i} = H_1(ID_i)$, $W = H_2(\Delta)$, and $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

(3) Verifies

$$e(V, P) = e\left(\sum_{i=1}^{n}\left(Q_{ID_i} + h_i PK_{ID_i}\right), MS_{pub}\right) e\left(\sum_{i=1}^{n} R_i, W\right). \quad (2)$$

(4) If (2) holds, emits 1 and the verifier accepts the aggregate signature σ; otherwise emits 0 and rejects.

5. Attack on Pankaj Kumar *et al.*'s CL-AS Scheme

As we know that the signature of $\sigma_i = (R_i, V_i)$ of message m_i on identity ID_i should be unforgeable. However, a malicious MS or an outside attacker may try to forge the signature. Once the MS or the outside attacker successfully forges the signature directly or indirectly, he/she finishes the signature forgery attack.

In this section, we mainly consider the type 2 adversary A_2 and first make a security analysis for Pankaj Kumar *et al.*'s CL-AS scheme, and then we demonstrate that it is vulnerable to the signature forgery attack, the attack details are described as follows.

Setup. The challenger executes the *Setup* algorithm to generate system parameters $params$ and master key α. Then it returns $params$ and α to the adversary A_2.

Queries. The adversary A_2 could get the signature σ_j on the message m_j signed by S_i with the identity ID_i via signature queries, where

$$\sigma_j = \begin{cases} R_j = r_j P \\ V_j = ppk_{ID_i} + r_j W + h_j x_i MS_{pub} \end{cases} \quad (3)$$

Forgery. In order to forge the signature σ_k^* on m_k signed by S_i with the identity ID_i, the adversary A_2 implements its attack as follows:

(1) Lets $R_k^* = r_k^* P = R_j = r_j P$

(2) Computes $h_k^* = H_3(m_k, ID_i, PK_{ID_i}, R_k^*)$

Verify. It is easy to verify the validity of the forged signature σ_k^*. The verifier calculates $Q_{ID_i} = H_1(ID_i)$ and $W = H_2(\Delta)$. Furthermore, the verifier calculates $h_k^* = H_3(m_k, ID_i, PK_{ID_i}, R_k^*)$. Then we use the forged signature σ_k^* to verify (1) and the concrete process is as follows:

$$\begin{aligned} e(V_k^*, P) &= e\left(ppk_{ID_i} + r_j W + h_k^* \alpha PK_{ID_i}, P\right) \\ &= e\left(\alpha Q_{ID_i}, P\right) e\left(r_j W, P\right) e\left(h_k^* \alpha \mathrm{PK}_{ID_i}, P\right) \\ &= e\left(Q_{ID_i}, MS_{pub}\right) e\left(r_j P, W\right) e\left(h_k^* PK_{ID_i}, MS_{pub}\right) \\ &= e\left(Q_{ID_i} + h_k^* PK_{ID_i}, MS_{pub}\right) e\left(R_k^*, W\right) \end{aligned} \tag{4}$$

Aggregate-Verify. It is easy to verify the validity of the forged signature σ^*. For $1 \le i \le n$, the verifier calculates $Q_{ID_i} = H_1(ID_i)$ and $h_k^* = H_3(m_k, ID_i, PK_{ID_i}, R_k^*)$. Furthermore, the verifier calculates $W = H_2(\Delta)$. Then we use the forged signature to verify (2); the concrete process is as follows:

$$\begin{aligned} e(V^*, P) &= e\left(\sum_{k=1}^{n}\left(ppk_{ID_i} + r_k^* W + h_k^* \alpha PK_{ID_i}\right), P\right) \\ &= e\left(\sum_{k=1}^{n}\left(\alpha Q_{ID_i} + h_k^* \alpha PK_{ID_i}\right), P\right) e\left(\sum_{k=1}^{n} r_j W, P\right) \\ &= e\left(\sum_{k=1}^{n}(_{ID_i} + h_k^* PK_{ID_i}), MS_{pub}\right) e\left(\sum_{k=1}^{n} r_j W, P\right) \\ &= e\left(\sum_{k=1}^{n}\left(Q_{ID_i} + h_k^* PK_{ID_i}\right), MS_{pub}\right) e\left(\sum_{k=1}^{n} R_k^*, W\right) \end{aligned} \tag{5}$$

We can find that the signature verifications (1) and (2) hold. That is, the forged signature pass verification and the malicious KGC can forge the signature successfully; Pankaj Kumar *et al.*'s CL-AS scheme is insecure.

6. Our Proposed CL-AS Scheme

To overcome the security flaw of the original scheme, we propose a new CL-AS scheme. Our CL-AS scheme includes seven phases: *Setup*, *Partial−Private−Key−Gen*, *Private−Key−Gen*, *Sign*, *Verify*, *Aggregate*, and *Aggregate−Verify*. The scheme details are described as below.

6.1. Setup. By executing the following operations, MS generates the system parameters after taking a security parameter k:

(1) Generates two cyclic groups G_1 and G_2 with the same order q, where q is a prime. P being a generator of G_1. $e: G_1 \times G_1 \to G_2$ being a bilinear pairing.

(2) Randomly selects a number $s \in Z_q^*$ as the master key of MS and calculates $MS_{pub} = sP$ as the public key of MS

(3) Chooses four hash functions: $H_1: \{0,1\} \to G_1$, $H_2: \{0,1\} \to G_1$, $h_1: \{0,1\} \to Z_q^*$, and $h_2: \{0,1\} \to Z_q^*$

(4) Keeps the master key s secret and the system parameters $params = (G_1, G_2, P, e, q, MS_{pub}, H_1, H_2, h_1, h_2)$ public.

6.2. Partial-Private-Key-Gen. By executing the following operations, MS generates the MSN's partial private key:

(1) Given ID_i as a MSN's identity, MS first computes $Q_i = H_1(ID_i)$ and then computes the MSN's partial private key $ppk_i = s.Q_i$.

(2) It secretly sends ppk_i to the corresponding MSN.

6.3. Private-Key-Gen. By executing the following operations, a MSN with the identity ID_i generates its private key and public key:

(1) Selects a random number x_i as the secret value.

(2) Sets $sk_i = \{ppk_i, x_i\}$ as its private key.

(3) Computes $pk_i = x_i P$ as its public key.

6.4. Sign. By executing the following operations, a signer with the identity ID_i generates a signature σ_i on the message m_i:

(1) Inputs system parameters $params$, state information Δ, and private-public key pair (sk_i, pk_i)

(2) Selects $r_i \in Z_q^*$ randomly and then calculates $R_i = r_i P$

(3) Computes $\alpha_i = h_1(ID_i, pk_i, R_i)$, $\beta_i = h_2(m_i, ID_i, pk_i, R_i)$, and $U = H_2(\Delta)$

(4) Computes $V_i = \alpha_i ppk_i + r_i MS_{pub} + \beta_i x_i U$

(5) Outputs $\sigma_i = (R_i, V_i)$ as the signature of message m_i.

6.5. Verify. By executing the following operations, the verifier verifies the signature $\sigma_i = (R_i, V_i)$ of message m_i on identity ID_i:

(1) Inputs the state information Δ.

(2) Computes $\alpha_i = h_1(ID_i, pk_i, R_i)$, $\beta_i = h_2(m_i, ID_i, pk_i, R_i)$, $Q_i = H_1(ID_i)$, and $U = H_2(\Delta)$

(3) Verifies

$$e(V_i, P) = e\left(\alpha_i Q_i + R_i, MS_{pub}\right) e\left(\beta_i pk_i, U\right) \tag{6}$$

(4) If (6) holds, emits 1 and the verifier accepts the signature σ_i; otherwise emits 0 and rejects.

6.6. Aggregate. By executing the following operations, an aggregator generates the aggregate signature σ from user-message-public key-signature pairs $(ID_i, m_i, pk_i, \sigma_i)_{1 \le i \le n}$:

(1) Inputs n tuples $(ID_i, m_i, pk_i, \sigma_i)$, where $1 \le i \le n$

(2) Computes $V = \sum_{i=1}^{n} V_i$

(3) Outputs $\sigma = (R, V)$ as the aggregate signature, where $R = \{R_1, R_1, \ldots, R_n\}$.

6.7. Aggregate-Verify. By executing the following operations, the aggregate verifier verifies the validity of the aggregate signature $\sigma = (R, V)$:

(1) Inputs the state information Δ, the tuples $(ID_i, m_i, pk_i, \sigma_i)_{1\le i\le n}$, and the aggregate signature $\sigma = (R, V)$

(2) Computes $U = H_2(\Delta)$, furthermore, for $1 \le i \le n$, computes $Q_i = H_1(ID_i)$, $\alpha_i = h_1(ID_i, pk_i, R_i)$ and $\beta_i = h_2(m_i, ID_i, pk_i, R_i)$

(3) Verifies

$$e(V, P) = e\left(\sum_{i=1}^{n}(\alpha_i Q_i + R_i), MS_{pub}\right) e\left(\sum_{i=1}^{n}\beta_i pk_i, U\right). \quad (7)$$

(4) If (7) holds, emits 1 and the verifier accepts the aggregate signature σ; otherwise emits 0 and rejects.

7. Security Analysis

A certificateless aggregate signature scheme should satisfy the following requirements: correctness and unforgeability.

7.1. Correctness

Theorem 1. *The proposed certificateless aggregate scheme is correct, if and only if the single signature and aggregate signature generated by our scheme make (1) and (2) hold. The correctness of the protocol is elaborated as follows:*

$$\begin{aligned} e(V_i, P) &= e\left(\alpha_i ppk_i + r_i MS_{pub} + \beta_i x_i U, P\right) \\ &= e(\alpha_i ppk_i, P)\, e\left(r_i MS_{pub}, P\right) e(\beta_i x_i U, P) \\ &= e(\alpha_i Q_i, sP)\, e\left(r_i P, MS_{pub}\right) e(\beta_i x_i P, U) \\ &= e\left(\alpha_i Q_i, MS_{pub}\right) e\left(r_i P, MS_{pub}\right) e(\beta_i x_i P, U) \\ &= e\left(\alpha_i Q_i + R_i, MS_{pub}\right) e(\beta_i pk_i, U) \end{aligned} \quad (8)$$

and

$$\begin{aligned} e(V, P) &= e\left(\sum_{i=1}^{n}\alpha_i ppk_i + r_i MS_{pub} + \beta_i x_i U, P\right) \\ &= e\left(\sum_{i=1}^{n}\alpha_i ppk_i, P\right) e\left(\sum_{i=1}^{n} r_i MS_{pub}, P\right) \\ &\cdot e\left(\sum_{i=1}^{n}\beta_i x_i U, P\right) = e\left(\sum_{i=1}^{n}\alpha_i Q_i, sP\right) \\ &\cdot e\left(\sum_{i=1}^{n} r_i P, MS_{pub}\right) e\left(\sum_{i=1}^{n}\beta_i x_i P, U\right) \\ &= e\left(\sum_{i=1}^{n}\alpha_i Q_i, MS_{pub}\right) e\left(\sum_{i=1}^{n} r_i P, MS_{pub}\right) \\ &\cdot e\left(\sum_{i=1}^{n}\beta_i x_i P, U\right) = e\left(\sum_{i=1}^{n}(\alpha_i Q_i + R_i), MS_{pub}\right) \\ &\cdot e\left(\sum_{i=1}^{n}\beta_i pk_i, U\right) \end{aligned} \quad (9)$$

7.2. Unforgeability. In this subsection, we first give the security model of CL-AS scheme and then give the security proof to show that the proposal is secure under the security model.

Security Model. There are two types of adversaries in the CL-AS security model: A_1 and A_2. A_1 simulates an outsider attacker, who cannot obtain the master key but can replace any user's public key with a value of his choice, while A_2 simulates an honest-but-curious KGC, who is an insider attacker and has no power to replace any user's public key but can access the system master key.

Definition 2. The security model of a CL-AS scheme is defined by two games (denoted by **Game1** and **Game2**) played between an adversary $A \in \{A_1, A_2\}$ and a challenger C; more details are defined below.

The adversary A can access the following random oracle machines in the scheme:

Hash queries: A can access any hash oracle in the scheme, including H_1, H_2, h_1, and h_2.

Setup: C performs the *Setup* algorithm to generate the master key s and the system parameter list *params*. Then C gives the corresponding response for different types of adversary.

Reveal-Partial-private-key: While A submits a partial private key query on the identity ID_i to challenger C, it checks if there is a record that corresponds to the identity ID_i in the L^{ppk} list and, if found, sends ppk_i to A; otherwise, if $ID_i = ID_{ts}$ it aborts; otherwise, it generates the partial private key ppk_i, sends it to A, and stores it in the list L^{ppk}.

Reveal-Secret-key: While A submits a secret value query on the identity ID_i to challenger C, it checks if there is a record that corresponds to the identity ID_i in the list L^x and, if found, sends x_i to A; otherwise, if $ID_i = ID_{ts}$ it aborts; otherwise, it generates the secret value x_i and sends it to A and stores it in the list L^x.

Reveal-Public-Key: When adversary A submits a public key query on the identity ID_i to challenger C, it checks if there is a record that corresponds to the identity ID_i in the list L^{pk}, if found, sends pk_i to A; otherwise it generates the public key pk_i, sends it to A and stores it in the list L^{pk}.

Replace-Public-key: While A submits a query that replaces the public key on the identity ID_i with $A's$ choice of public key pk_i^* to challenger C, C checks if there is a record that corresponds to the identity ID_i in the list L^{pk} and, if found, then it updates the corresponding item

(ID_i, x_i, pk_i, ppk_i) to $(ID_i, x_i, pk_i^*, ppk_i)$ in the list L^{pk}; otherwise it aborts.

Sign: While A submits a signature query on the message m_i with the signer's identity ID_i to challenger C, C executes one of the following operations:

(1) If the target user ID_i has not been created, it aborts.

(2) If the target user ID_i has been created and the related user public key pk_i has not been replaced, then it returns a valid signature σ_i.

(3) If the target user ID_i has been created and the corresponding user public key pk_i has been replaced with pk_i^*, then it returns a signature σ_i^*.™

We, respectively, define two games to describe two different types of attackers in the CLS, as shown below.

Game1: The challenger C interacts with adversary A_1 as follows:

(1) Inputting k as a security parameter, C performs the *Setup* algorithm to generate the master key s and the system parameter list *params*. Then C sends *params* to A_1 and keeps s secret.

(2) A_1 is capable of accessing any hash oracle in the scheme and *Reveal* − *Partial* − *Private* − *Key*, *Reveal* − *Secret* − *Key*, *Reveal* − *Public* − *Key*, *Replace* − *Public* − *Key*, and *Sign* queries at any stage during the simulation in polynomial bound.

Forgery: A_1 outputs an aggregate signature σ^* with respect to n user-message-public key-signature pairs $(ID_i^*, m_i^*, pk_i^*, \sigma_i^*)$, where $1 \le i \le n$. We say that A_1 wins *Game*1 if and only if the following conditions are met:

(1) σ^* is a valid aggregate signature with respect to user-message-public key-signature pairs $(ID_i^*, m_i^*, pk_i^*, \sigma_i^*)$, where $1 \le i \le n$.

(2) The targeted identity ID_i^* has not been submitted during the *Reveal* − *Partial* − *Private* − *Key* query.

(3) (ID_i^*, m_i^*) has not been submitted during the *Sign* query.

Game2: The challenger C interacts with adversary A_2 as follows:

(1) Inputting k as a security parameter, C performs the *Setup* algorithm to generate the master key s and the system parameter list *params*. Then C sends *params* and s to A_2.

(2) A_2 is capable of accessing any hash oracle in the scheme and *Reveal* − *Partial* − *Private* − *Key*, *Reveal* − *Public* − *Key*, and *Sign* queries at any stage during the simulation in polynomial bound.

Forgery: A_2 outputs an aggregate signature σ^* with respect to n user-message-public key-signature pairs $(ID_i^*, m_i^*, pk_i^*, \sigma_i^*)$, where $1 \le i \le n$. We say that A_2 wins *Game*2 if and only if the following conditions are met:

(1) σ^* is a valid aggregate signature with respect to user-message-public key-signature pairs $(ID_i^*, m_i^*, pk_i^*, \sigma_i^*)$, where $1 \le i \le n$.

(2) The targeted identity ID_i^* has not been submitted during the *Reveal* − *Secret* − *Key* query.

(3) (ID_i^*, m_i^*) has not been submitted during the *Sign* query.

Provable Security. In this section, we demonstrate that the new CL-AS scheme is secure under the security model described in the previous subsection. Our security proof consists of two parts.

In this section, we prove that our proposed CL-AS scheme is secure under the security model present in the previous section, and the specific process is described in the following two parts: (1) the CL-AS is unforgeable to type 1 adversary A_1 and (2) the CL-AS scheme is unforgeable to type 2 adversary A_2.

Theorem 3. *The proposed CL-AS scheme is existentially unforgeable against type 1 adversary A_1, if the CDH problem is difficult to solve in G_1.*

Proof. We can prove the unforgeability of our CL-AS scheme against type 1 adversary with **Game1** that involves A_1 and an algorithm called simulator C. □

Given a random instance of the CDH problem $(P, Q_1 = aP, Q_2 = bP)$, where P is a generator of G_1, our ultimate goal is to find the result of abP by solving the CDH problem.

(i) *Setup:* C randomly chooses ID_{ts} as the target identity of sensor challenged, sets $MS_{pub} = Q_1 = aP$, and generates and returns system parameter $params = \{G_1, G_2, P, e, q, MS_{pub}, H_1, H_2, h_1, h_2\}$ to A_1. A_1 performs the inquiries as follows:

(ii) H_1 *query:* C maintains a list denoted L^{H_1}, and the structure of L^{H_1} is $(ID_i, \delta_i, \varepsilon_i, Q_i)$; all the elements in L^{H_1} are initialized to null. When A_1 performs the query with the identity ID_i, C checks whether a tuple $(ID_i, \delta_i, \varepsilon_i, Q_i)$ exists in L^{H_1}; if it exists, it returns Q_i to A_1; otherwise, C randomly selects $\varepsilon_i \in \{0, 1\}$ and $\delta_i \in Z_q^*$. If $\varepsilon_i = 0$, set $Q_i = \delta_i P$; otherwise, if $\varepsilon_i = 1$, set $Q_i = \delta_i Q_2 = \delta_i bP$. It returns Q_i to A_1 and stores $(ID_i, \delta_i, \varepsilon_i, Q_i)$ to L^{H_1}.

(iii) H_2 *query:* C maintains a list denoted L^{H_2}, and the structure of L^{H_2} is (MS_{pub}, ϑ, U), all the elements in L^{H_2} are initialized to null. When A_1 executes the query with MS_{pub}, C checks if a tuple (MS_{pub}, ϑ, U) exists in L^{H_2}; if it exists, it returns U to A_1; otherwise, C randomly selects $\vartheta \in Z_q^*$ and computes $U = \vartheta P$. It returns U to A_1 and stores (MS_{pub}, ϑ, U) to L^{H_2}.

(iv) h_1 *query:* C maintains a list denoted L^{h_1}, and the structure of L^{h_1} is $(ID_i, pk_i, R_i, \alpha_i)$, all the elements in L^{h_1} are initialized to null. When A_1 executes the query with the tuple (ID_i, pk_i, R_i), C check whether a tuple

$(ID_i, pk_i, R_i, \alpha_i)$ exists in L^{h_1}; if it exists, it returns α_i to A_1; otherwise, C randomly selects α_i. It returns α_i to A_1 and stores $(ID_i, pk_i, R_i, \alpha_i)$ to L^{h_1}.

(v) *h_2 query:* C maintains a list denoted L^{h_2}, and the structure of L^{h_2} is $(m_i, ID_i, pk_i, R_i, \beta_i)$, all the elements in L^{h_2} are initialized to null. When A_1 executes the query with the tuple (m_i, ID_i, pk_i, R_i), C checks if a tuple $(m_i, ID_i, pk_i, R_i, \beta_i)$ exists in L^{h_2}; if it exists, it returns β_i to A_1; otherwise, C randomly selects β_i. It returns β_i to A_1 and stores $(m_i, ID_i, pk_i, R_i, \beta_i)$ to L^{h_2}.

(vi) *Reveal-Partial-Private-Key queries:* C maintains a list denoted L^{ppk}, and the structure of L^{ppk} is (ID_i, ppk_i), all the elements in L^{ppk} are initialized to null. When A_1 executes the query with ID_i, C first checks whether $ID_i = ID_{ts}$; if it holds, output $\perp$; otherwise, C checks whether a tuple (ID_i, ppk_i) exists in L^{ppk}; if it exists, it returns ppk_i to A_1; otherwise, C recalls the corresponding tuple $(ID_i, \delta_i, \varepsilon_i, Q_i)$ from the list L^{H_1} and computes $ppk_i = \delta_i MS_{pub} = a\delta_i P$. It returns ppk_i to A_1 and stores (ID_i, ppk_i) to L^{ppk}.

(vii) *Reveal-Secret-Key-queries:* C maintains a list denoted L^x, and the structure of L^x is (ID_i, x_i), all the elements in L^x are initialized to null. When A_1 performs the query with the identity ID_i, C first checks if $ID_i = ID_{ts}$; if it holds, output $\perp$; otherwise, C checks whether a tuple exists in (ID_i, x_i); if it exists, it returns x_i to A_1; otherwise, C randomly selects x_i. It returns x_i to A_1 and stores x_i to (ID_i, x_i).

(viii) *Reveal-Public-Key queries:* C maintains a list denoted L^{pk}, and the structure of L^{pk} is (ID_i, pk_i), all the elements in L^{pk} are initialized to null. When A_1 performs the query with the identity ID_i, C checks whether a tuple (ID_i, pk_i) exists in L^{pk}; if it exists pk_i, it returns pk_i to A_1; otherwise, it accesses L^x to get x_i and computes $pk_i = x_i P$. It returns pk_i to A_1 and stores (ID_i, pk_i) to L^{pk}.

(ix) *Replace-Public-Key queries:* When A_1 executes the query with the identity (ID_i, pk_i^*), in response, C replaces the real public key pk_i of ID_i with pk_i^* chosen by A_1 in the list L^{pk}.

(x) *Sign queries:* When A_1 performs the query with the user identity ID_i and public key pk_i, message m_i, C accesses L^{H_1}, L^{H_2}, L^{h_1}, and L^{h_2} to get ε_i, Q_i, U, α_i, and β_i, respectively. Furthermore, C randomly selects r_i and computes $R_i = r_i P$; if $\varepsilon_i = 0$, C computes $V_i = \delta_i \alpha_i MS_{pub} + r_i MS_{pub} + \beta_i \vartheta pk_i$; otherwise, if $\varepsilon_i = 1$, C computes $V_i = \delta_i \alpha_i bMS_{pub} + r_i MS_{pub} + \beta_i \vartheta pk_i$. It returns $\sigma_i = (R_i, V_i)$ to A_1 as the signature of the message m_i on the user identity ID_i with the public key pk_i.

(xi) *Forgery:* Finally, A_1 outputs a forged aggregate signature $\sigma^* = (R^*, V^*)$ from message-identity-public key pairs (m_i^*, ID_i^*, pk_i^*), where $1 \le i \le n$. If all $\varepsilon_i = 0$ hold, A_1 aborts; otherwise, without loss of generality, let $ID_{ts} = ID_1$; that is, $\varepsilon_1 = 1$, $\varepsilon_i = 0$ $(2 \le i \le n)$, and then the forged signature should make the following hold:

$$e(V^*, P) = e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, U\right) \tag{10}$$

where $Q_i^* = \delta_i^* P$ $(2 \le i \le n)$, $Q_1^* = \delta_1^* bP$, $U = \vartheta P$, $V^* = \sum_{i=1}^{n} V_i^*$, and $R^* = \{R_1^*, R_2^*, \ldots, R_n^*\}$.

Furthermore, the derivation process is shown as follows:

$$\begin{aligned}
&e(V^*, P) = e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, U\right) \\
&\Longrightarrow e(V^*, P) = e\left(R_1^* + \sum_{i=2}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\alpha_1^* Q_1^*, MS_{pub}\right) e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, \vartheta P\right) \\
&\Longrightarrow e\left(\alpha_1^* Q_1^*, MS_{pub}\right) = e(V^*, P) \cdot \left[e\left(R_1^* + \sum_{i=2}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, \vartheta P\right)\right]^{-1} \\
&\Longrightarrow e(\delta_1^* \alpha_1^* abP, P) = e(V^*, P) \cdot \left[e\left(R_1^* + \sum_{i=2}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, \vartheta P\right)\right]^{-1} \\
&\Longrightarrow \delta_1^* \alpha_1^* abP = V^* - \left[\left(r_1^* + \sum_{i=2}^{n} (\delta_i^* + r_i^*)\right) MS_{pub} + \sum_{i=1}^{n} \beta_i^* x_i^* \vartheta\right] \\
&\Longrightarrow abP = \left(V^* - \left(r_1^* + \sum_{i=2}^{n} (\delta_i^* + r_i^*)\right) MS_{pub} - \sum_{i=1}^{n} \beta_i^* x_i^* \vartheta\right) (\delta_1^* \alpha_1^*)^{-1}
\end{aligned} \tag{11}$$

However, this contradicts the CDH assumption; thus the single signature and aggregate signature generated by the new scheme are unforgeable.

Theorem 4. *The proposed certificateless aggregate scheme is existentially unforgeable against type 2 adversary A_2, if the CDH problem is difficult to solve in G_1.*

Proof. We can prove the unforgeability of our CL-AS scheme against type 2 adversary with **Game2** that involves A_2 and an algorithm called simulator C. □

Given a random instance of the CDH problem $(P, Q_1 = aP, Q_2 = bP)$, where P is a generator of G_1, our ultimate goal is to find the result of abP by solving the CDH problem.

(i) *Setup:* C randomly chooses ID_{ts} as the target identity of sensor challenged, sets $MS_{pub} = \lambda P$, and returns master key λ and system parameter $params = \{G_1, G_2, P, e, q, MS_{pub}, H_1, H_2, h_1, h_2\}$ to A_2. A_2 performs the inquiries as follows.

(ii) h_1, h_2, and *Reveal-Secret-Value queries* are the same as the corresponding queries in Theorem 3. Since A_2 can access the master key, there is no need to the *Reveal-Partial-Private-Key queries* and *Replace-Public-Key queries*.

(iii) H_1 *query:* C maintains a list denoted L^{H_1}, and the structure of L^{H_1} is (ID_i, δ_i, Q_i), all the elements in L^{H_1} are initialized to null. When A_2 performs the query with the identity ID_i, C checks whether a tuple L^{H_1} is (ID_i, δ_i, Q_i) exists in L^{H_1}; if it exists, it returns Q_i to A_2; otherwise, C randomly selects δ_i and computes $Q_i = \delta_i P$. It returns Q_i to A_2 and stores (ID_i, δ_i, Q_i) to L^{H_1}.

(iv) H_2 *query:* C maintains a list denoted L^{H_2}, and the structure of L^{H_2} is (MS_{pub}, ϑ, Z), all the elements in L^{H_2} are initialized to null. When A_2 executes the query with MS_{pub}, C checks whether a tuple (MS_{pub}, ϑ, Z) exists in L^{H_2}; if it exists, it returns U to A_2; otherwise, C randomly selects $\vartheta \in Z_q^*$ and computes $U = \vartheta Q_1 = \vartheta aP$. It returns U to A_2 and stores (MS_{pub}, ϑ, U) to L^{H_2}.

(v) *Reveal-Public-Key queries:* C maintains a list denoted L^{pk}, and the structure of L^{pk} is (ID_i, ω_i, pk_i), all the elements in L^{pk} are initialized to null. When A_2 performs the query with the identity ID_i, C checks whether a tuple (ID_i, ω_i, pk_i) exists in L^{pk}; if it exists pk_i, it returns pk_i to A_2; otherwise, C randomly selects $\omega_i \in \{0, 1\}$; if $\omega_i = 0$, C accesses L^x to get x_i and computes $pk_i = x_i P$; otherwise, if $\omega_i = 1$, C randomly selects $x_i \in Z_q^*$ and computes $pk_i = x_i Q_2 = x_i bP$. It returns pk_i to A_2 and stores (ID_i, ω_i, pk_i) to L^{pk}.

(vi) *Sign queries:* When A_2 performs the query with user's identity ID_i and public key pk_i, message m_i, C accesses L^{H_1}, L^{H_2}, L^{h_1}, L^{h_2}, and L^{pk} to get $Q_i, U, \alpha_i, \beta_i$, and ω_i, respectively. Furthermore, C randomly selects r_i and computes $R_i = r_i P$; if $\omega_i = 0$, C computes $V_i = \delta_i \alpha_i MS_{pub} + r_i MS_{pub} + \beta_i \vartheta pk_i$; otherwise, if $\omega_i = 1$, C computes $V_i = \delta_i \alpha_i MS_{pub} + r_i MS_{pub} + \beta_i \vartheta apk_i$. It returns $\sigma_i = (R_i, V_i)$ to A_2 as the signature of the message m_i on user's identity ID_i with the public key pk_i.

(vii) *Forgery:* Finally, A_2 outputs a forged aggregate signature $\sigma^* = (R^*, V^*)$ from message-identity-public key pairs (m_i^*, ID_i^*, pk_i^*), where $1 \le i \le n$. If all $\omega_i = 0$ hold, A_2 aborts; otherwise, without loss of generality, let $ID_{ts} = ID_1$; that is, $\omega_1 = 1$, $\omega_i = 0$ $(2 \le i \le n)$, and then the forged aggregate signature should satisfy:

$$e(V^*, P) = e\left(\sum_{i=1}^{n} \alpha_i^* Q_i^* + R_i^*\right), MS_{pub}) e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, U\right) \tag{12}$$

where $Q_i^* = \delta_i^* P$, $pk_1^* = x_1^* bP$, $pk_i^* = x_i^* P$ $(2 \le i \le n)$, $U = \vartheta aP$, $V^* = \sum_{i=1}^{n} V_i^*$, and $R^* = \{R_1^*, R_2^*, \ldots, R_n^*\}$.

Furthermore, the derivation process is shown as below:

$$\begin{aligned}
&e(V^*, P) = e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=1}^{n} \beta_i^* pk_i^*, U\right) \\
&\Longrightarrow e(V^*, P) = e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e(\alpha_1^* pk_1^*, U)\, e\left(\sum_{i=2}^{n} \beta_i^* pk_i^*, U\right) \\
&\Longrightarrow e(\beta_1^* pk_1^*, U) = e(V^*, P) \cdot \left[e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=2}^{n} \beta_i^* pk_i^*, U\right)\right]^{-1} \\
&\Longrightarrow e(\beta_1^* \vartheta abP, P) = e(V^*, P) \cdot \left[e\left(\sum_{i=1}^{n} (\alpha_i^* Q_i^* + R_i^*), MS_{pub}\right) \cdot e\left(\sum_{i=2}^{n} \beta_i^* pk_i^*, U\right)\right]^{-1} \\
&\Longrightarrow \beta_1^* \vartheta abP = V^* - \left[\left(\sum_{i=1}^{n} (\delta_i^* \alpha_i^* + r_i^*)\right) MS_{pub} + \sum_{i=1}^{n} \beta_i^* x_i^* \vartheta\right]
\end{aligned} \tag{13}$$

Table 1: Security comparisons.

	B_{11}	B_{12}	B_{13}	SP	B_{21}	B_{22}	B_{23}	SP
Gong's Scheme [9]	*No*	*Yes*	*No*	*L*	*Yes*	*No*	*No*	*L*
liu's Scheme [10]	*Yes*	*Yes*	*Yes*	*H*	*No*	*No*	*No*	*L*
kumar's Scheme [8]	*Yes*	*Yes*	*Yes*	*H*	*No*	*No*	*No*	*L*
Our proposed Scheme	*Yes*	*Yes*	*Yes*	*H*	*Yes*	*Yes*	*Yes*	*H*

Table 2: symbol-operation-execution time.

Symbol	Operation	Time (ms)
T_{mts}	The time of performing a general hash operation in Z_q^*	0.0002
T_{mtp}	The time of performing a map-to-point operation in G_1	9.773
T_{ecc-pa}	The time of performing a point addition operation in G_1	0.022
T_{ecc-pm}	The time of performing a point multiplication operation in G_1	3.740
T_{bp}	The time of performing a bilinear pairing operation	11.515

$$\Longrightarrow abP = \left(V^* - \left(\sum_{i=1}^{n}\left(\delta_i^*\alpha_i^* + r_i^*\right)\right)MS_{pub} - \sum_{i=1}^{n}\beta_i^* x_i^* \vartheta\right)\left(\beta_1^*\vartheta\right)^{-1} \tag{14}$$

However, this contradicts the CDH assumption; thus the single signature and aggregate signature generated by the new scheme are unforgeable.

8. Security Comparisons and Performance Analysis

In this section, we first compare the security of the newly proposed CL-AS scheme with the other three CL-AS schemes and further analyze the performance of the new CL-AS scheme by evaluating the computation overhead.

8.1. Security Comparisons. In this subsection, we compare the security of the newly proposed CL-AS scheme with the other three CL-AS schemes [8–10]. For the convenience of description, let A_1 and A_2 denote the type1 and the type2 adversaries, respectively. Furthermore, the two types of adversaries are divided into three levels [31], where B_{i1} denotes general adversary, B_{i2} denotes strong adversary, B_{i3} denotes super adversary, respectively, and $i \in \{1, 2\}$; the value of i corresponds to the type i adversary. *Yes* denotes that it can satisfy the corresponding security requirement and *No* denotes that it cannot satisfy the corresponding security requirement. L denotes the weaker security and H denotes the stronger security under the corresponding attack types. SP denotes the security performance. The security comparisons of the various schemes are listed in Table 1.

As shown in Table 1, we can find that the first three schemes (i.e., Gong's scheme [9], liu's scheme [10], and kumar's scheme [8]) cannot satisfy all the security requirements. Especially for Gong's two CL-AS schemes [9], under the attacks of the type1 and the type2 adversaries, none of them can meet the security levels of B_3. liu and kumar's schemes cannot resist the malicious KGC attack (B_3 level). In contrast, our CL-AS scheme can meet all the security requirements. Hence, our proposed CL-AS scheme has better security than that of the other three schemes.

8.2. Performance Analysis. In this section, we analyze the performance of our CL-AS scheme by evaluating the computation overhead. Compared with that of kumar *et al.*'s scheme, our implementation shows that the new proposal can satisfy the security requirement and provide an improved security while reducing the computation cost.

In order to achieve a credible security level, we choose q and p as 160-bits prime number and 512-bits prime number, respectively. A ate pairing $e : G_1 \times G_1 \rightarrow G_2$ is used in our experiments, where G_1 and G_2 are cyclic groups with the same order q, defined on the super singular elliptic curve $E(F_p) : y^2 = x^3 + 1$.

We have implemented kumar *et al.*'s scheme and the newly proposed scheme with the MIRACL library [32] on a personal computer (Lenovo with Intel I5-3470 3.20G Hz processor, 4G bytes memory and Window 7 operating system). For the sake of simplicity, we firstly define the corresponding relations related symbol-operation-execution time as shown in Table 2.

Because *Setup*, *Partial-Private-Key-Gen*, and *Private-Key-Gen* phases are executed by MS or user and all of them are one-time operation, we laid stress on the comparisons of the computation cost in *Sign*, *Verify*, *Aggregate*, and *Aggregate-Verify* phases.

In *Sign* phase, the user in kumar *et al.*'s scheme needs to perform one general hash operation in Z_q^*, one map-to-point hash operation in G_1, two-point addition operations in G_1 and three-point multiplication operations in G_1. Therefore, the running time of the *Sign* phase is $T_{mts} + T_{mtp} + 2T_{ecc-pa} + 3T_{ecc-pm}$, whereas the user in the new proposal needs to perform two general hash operations in Z_q^*, one map-to-point hash operation in G_1, two-point addition operations in G_1, and four-point multiplication operations in G_1. Therefore,

Table 3: Computation cost comparisons (millisecond).

	kumar's Scheme [8]	**Our Proposed Scheme**
Sign	$T_{mts} + T_{mtp} + 2T_{ecc-pa} + 3T_{ecc-pm} \approx 21.0372$	$2T_{mts} + T_{mtp} + 2T_{ecc-pa} + 4T_{ecc-pm} \approx 24.7774$
Verify	$T_{mts} + T_{mtp} + T_{ecc-pa} + T_{ecc-pm} + 3T_{bp} \approx 48.0802$	$2T_{mts} + T_{mtp} + T_{ecc-pa} + 2T_{ecc-pm} + 3T_{bp} \approx 51.8204$
Agrregate	$99T_{ecc-pa} \approx 2.178$	$99T_{ecc-pa} \approx 2.178$
Aggregate–Verify	$100T_{mts} + 200T_{mtp} + 298T_{ecc-pa} + 100T_{ecc-pm} + 3T_{bp} \approx 2369.721$	$200T_{mts} + 101T_{mtp} + 298T_{ecc-pa} + 200T_{ecc-pm} + 3T_{bp} \approx 1776.214$

the running time of the *Sign* phase in our proposed scheme is $2T_{mts} + T_{mtp} + 2T_{ecc-pa} + 4T_{ecc-pm}$ milliseconds.

In *Verify* phase, the verifier in kumar *et al.*'s scheme needs to perform one general hash operation in Z_q^*, one map-to-point hash operation in G_1, one-point addition operation in G_1, one-point multiplication operation in G_1, and three-bilinear pairing operations. Therefore, the running time of the *Verify* phase is $T_{mts} + T_{mtp} + T_{ecc-pa} + T_{ecc-pm} + 3T_{bp}$, whereas the verifier in the new proposal needs to perform two general hash operations in Z_q^*, one map-to-point hash operation in G_1, one-point addition operation in G_1, two-point multiplication operation in G_1, and three-bilinear pairing operations. Therefore, the running time of the *Verify* phase in our proposed scheme is $2T_{mts} + T_{mtp} + T_{ecc-pa} + 2T_{ecc-pm} + 3T_{bp}$ milliseconds.

In *Aggregate* phase, the aggregator in kumar *et al.*'s scheme needs to perform $n - 1$ point addition operations in G_1, whereas the aggregator in the new proposal needs to perform $n - 1$ point addition operations in G_1. We can find that the running time of the *Aggregate* phase in the two schemes is equal to $(n-1)T_{ecc-pa}$ milliseconds.

In *Aggregate – Verify* phase, the aggregate verifier in kumar *et al.*'s scheme needs to perform n general hash operations in Z_q^*, $2n$ map-to-point hash operations in G_1, $3n-2$ point addition operations in G_1, n point multiplication operations in G_1, and three-bilinear pairing operations. Therefore, the running time of the *Aggregate – Verify* phase is $nT_{mts} + 2nT_{mtp} + (3n-2)T_{ecc-pa} + nT_{ecc-pm} + 3T_{bp}$ milliseconds, whereas the verifier in the new proposal needs to perform $2n$ general hash operations in Z_q^*, $n+1$ map-to-point hash operations in G_1, $3n-2$ point addition operations in G_1, $2n$ point multiplication operations in G_1, and three-bilinear pairing operations. Therefore, the running time of the *Aggregate – Verify* phase in our proposed scheme is $2nT_{mts} + (n+1)T_{mtp} + (3n-2)T_{ecc-pa} + 2nT_{ecc-pm} + 3T_{bp}$ milliseconds.

Assuming that $n = 100$ in the *Aggregate* and *Aggregate–Verify* phases, the computation overhead comparisons are shown in Table 3 and Figure 3. As can be seen from the results in Table 3 and Figure 3, the computation overhead of our proposed CL-AS scheme is slightly higher than that of kumar *et al.*'s scheme for *Sign* and *Verify* phases. In *Aggregate* phase, the computation overheads of the two schemes are equal, whereas in the *Agrregate – Verify* phase, the computation overhead of our scheme is much lower than that of kumar *et al.*'s scheme. However, compared with the total computation overheads of these four phases, our scheme's computation overhead is reduced by 24 percentage points compared with the that of kumar *et al.*'s scheme [8]. That is, we enforce the security in a large extent with the efficiency increased by 24 in computation overhead.

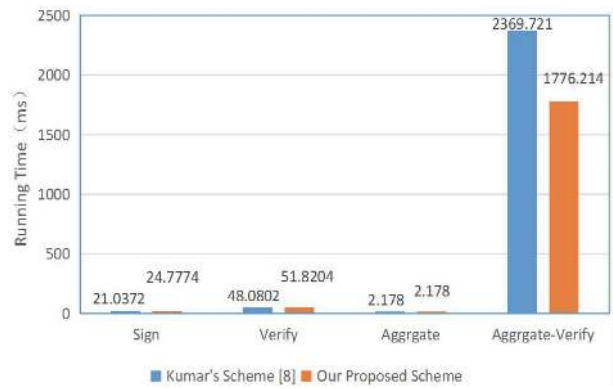

Figure 3: Computation cost comparisons.

9. Conclusion

To ensure the privacy and integrity of patients medical information, several CL-AS schemes have been put forward recently. In this paper, we first investigate the techniques of the data signature. Then we show that Pankaj Kumar *et al.*'s scheme is vulnerable against the malicious attack. This attack is a serious threat from the inside attacker acting as a MS, because it allows the adversary to forge a signature of message m_j using the signature of the message m_i on signer ID_i.

To overcome this security flaw, we put forward a new CL-AS scheme for the issues of integrity and privacy in HMSN. The security analysis shows that our proposed CL-AS scheme is provably secure and can meet the security requirements in HMSN. In addition, the detailed performance analysis and evaluation demonstrate that our CL-AS scheme can achieve a novel security level while reducing the computation cost. Our CL-AS scheme is robust against all types of attacks, making it more useful for protecting the integrity and privacy of patients medical information.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, 2009.

[2] Z. Zhang and K. Wang, "A trust model for multimedia social networks," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 969–979, 2013.

[3] M. Al Ameen, J. Liu, and K. Kwak, "Security and privacy issues in wireless sensor networks for healthcare applications," *Journal of Medical Systems*, vol. 36, no. 1, pp. 93–101, 2012.

[4] M. R. Yuce, S. W. P. Ng, N. L. Myo, J. Y. Khan, and W. Liu, "Wireless body sensor network using medical implant band," *Journal of Medical Systems*, vol. 31, no. 6, pp. 467–474, 2007.

[5] C. Gao, Q. Cheng, X. Li, and S. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Computing*, pp. 1–9, 2018.

[6] Z. Huang, S. Liu, X. Mao, K. Chen, and J. Li, "Insight of the protection for data security under selective opening attacks," *Information Sciences*, vol. 412-413, pp. 223–241, 2017.

[7] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Computing*, pp. 1–10, 2017.

[8] P. Kumar, S. Kumari, V. Sharma, A. K. Sangaiah, J. Wei, and X. Li, "A certificateless aggregate signature scheme for healthcare wireless sensor network," *Sustainable Computing*, 2017.

[9] G. Zheng, L. Yu, H. Xuan, and C. Kefei, "Two certificateless aggregate signatures from bilinear maps," in *Proceedings of the SNPD 2007: 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 188–193, chn, August 2007.

[10] H. Liu, S. Wang, M. Liang, and Y. Chen, "New construction of efficient certificateless aggregate signatures," *International Journal of Security and Its Applications*, vol. 8, no. 1, pp. 411–422, 2014.

[11] W. Diffie, W. Diffie, and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[12] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology (Santa Barbara, Calif., 1984)*, vol. 196 of *Lecture Notes in Comput. Sci.*, pp. 47–53, Springer, Berlin, 1985.

[13] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 64, no. 2, pp. 425–437, 2015.

[14] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," *Asiacrypt*, vol. 2894, pp. 452–473, 2003.

[15] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "On the security of certificateless signature schemes from Asiacrypt 2003," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 3810, pp. 13–25, 2005.

[16] J. Li, X. Huang, Y. Mu, and W. Wu, "Cryptanalysis and improvement of an efficient certificateless signature scheme," *Journal of Communications and Networks*, vol. 10, no. 1, pp. 10–17, 2008.

[17] W. Yap, S. Heng, and B. Goi, "An Efficient Certificateless Signature Scheme," in *Emerging Directions in Embedded and Ubiquitous Computing*, vol. 4097 of *Lecture Notes in Computer Science*, pp. 322–331, 2006.

[18] M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong, and G. Yang, "Malicious KGC attacks in certificateless cryptography," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, ASIACCS '07*, pp. 302–311, March 2007.

[19] A. W. Dent, B. t. Libert, and K. . Paterson, "Certificateless encryption schemes strongly secure in the standard model," in *Public key cryptography*, vol. 4939 of *Lecture Notes in Comput. Sci.*, pp. 344–359, Springer, Berlin, 2008.

[20] X. Li, K. Chen, and L. Sun, "Certificateless signature and proxy signature schemes from bilinear pairings," *Lithuanian Mathematical Journal*, vol. 45, no. 1, pp. 95–103, 2005.

[21] J. K. Liu, M. H. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS '07)*, pp. 273–283, March 2007.

[22] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Lecture Notes in Computer Science*, vol. 2656 of *Lecture Notes in Comput. Sci.*, pp. 416–432, Springer, Berlin, 2003.

[23] L. Zhang and F. Zhang, "A new certificateless aggregate signature scheme," *Computer Communications*, vol. 32, no. 6, pp. 1079–1085, 2009.

[24] H. Xiong, Z. Guan, Z. Chen, and F. Li, "An efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 219, pp. 225–235, 2013.

[25] D. He, M. Tian, and J. Chen, "Insecurity of an efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 268, pp. 458–462, 2014.

[26] L. Cheng, Q. Wen, Z. Jin, H. Zhang, and L. Zhou, "Cryptanalysis and improvement of a certificateless aggregate signature scheme," *Information Sciences*, vol. 295, pp. 337–346, 2015.

[27] F. Zhang, L. Shen, and G. Wu, "Notes on the security of certificateless aggregate signature schemes," *Information Sciences*, vol. 287, pp. 32–37, 2014.

[28] Y. Zhang and C. Wang, "Comment on new construction of efficient certificateless aggregate signatures," *International Journal of Security and Its Applications*, vol. 9, no. 1, pp. 147–154, 2015.

[29] D. He and S. Zeadally, "Authentication protocol for an ambient assisted living system," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 71–77, 2015.

[30] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal*, no. 99, pp. 1–10, 2015.

[31] D. He, S. Zeadally, B. Xu, and X. Huang, "An Efficient Identity-Based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[32] M. Scott, *Miracla multiprecision integer and rational arithmetic c/c++ library. shamus software ltd*, Dublin, Ireland, 2003, http://indigo.ie/mscott.