PLOS ONE

RESEARCH ARTICLE

An improved pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks

Lifeng Zhou[1☯], **Xinchun Yin**[1,2☯*]

1 College of Information Engineering, Yangzhou University, Yangzhou, Jiangsu, China, **2** College of Guangling, Yangzhou University, Yangzhou, Jiangsu, China

☯ These authors contributed equally to this work.
* xcyin@yzu.edu.cn

Abstract

In healthcare wireless medical sensor networks (HWMSNs), the medical sensor nodes are employed to collect medical data which is transmitted to doctors for diagnosis and treatment. In HWMSNs, medical data is vulnerable to various attacks through public channels. In addition, leakage of patients' information happens frequently. Hence, secure communication and privacy preservation are major concerns in HWMSNs. To solve the above issues, Zhan et al. put forward a pairing-free certificateless aggregate signature (PF-CLAS) scheme. However, according to our cryptanalysis, the malicious medical sensor node (MSN_i) can generate the forged signature by replacing the public key in the PF-CLAS scheme. Hence, to address this security flaw, we design the improved PF-CLAS scheme that can achieve unforgeability, anonymity, and traceability. Since we have changed the construction of the partial private key, the improved PF-CLAS scheme can resist Type I and Type II attacks under the Elliptic Curve Discrete Logarithm assumption. In terms of the performance evaluation, the proposed scheme outperforms related CLAS schemes, which is more suitable for HWMSNs environments.

OPEN ACCESS

Citation: Zhou L, Yin X (2022) An improved pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks. PLoS ONE 17(7): e0268484. https://doi.org/10.1371/journal.pone.0268484

Editor: Pandi Vijayakumar, University College of Engineering Tindivanam, INDIA

Received: March 15, 2022

Accepted: May 1, 2022

Published: July 11, 2022

Data Availability Statement: All relevant data are within the paper and its Supporting information files.

Funding: This work is supported by the National Natural Science Foundation of China (No. 61472343), funded by Xinchun Yin. The funder has the role in data collection, preparation of the manuscript, validation of the manuscript, review of the manuscript, and editing of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

With the rapid development of wireless body area networks, healthcare wireless medical sensor networks (HWMSNs) are driving the progress of intelligent medical treatment. In the current HWMSNs environment, patients use wearable and implantable medical devices from which multifarious medical data is collected [1]. Then the data is transmitted to doctors for real-time processing and feedback. Since the outbreak of COVID-19, hospitals have been using HWMSNs to monitor and treat the symptoms of patients [2]. However, medical data is transmitted through insecure public channels, and adversaries are able to eavesdrop on, tamper with, and forge the data readily [3, 4]. Upon tampering and forgery, doctors may make accurate diagnoses that can harm patients [5]. Furthermore, if the identities of patients are exposed in the form of plaintext, the patients' real identities will be divulged [6–8].

Consequently, it is of great importance to guarantee secure communication and privacy preservation in HWMSNs.

In recent years, various technologies have been used for HWMSNs [9, 10]. To ensure the security of medical data, Mamta *et al.* [11] adopted the blockchain technology to design a decentralized and efficient attribute-based searchable encryption scheme. Nguyen *et al.* [12] put forward a blockchain-based intrusion detection and data transmission scheme that can realize the high-security level of the system. To guarantee secure communication and build trustworthiness among nodes in networks, Mirsadeghi *et al.* [13] presented a trust infrastructure-based authentication scheme by using digital signature and encryption technologies. Vijayakumar *et al.* [14] constructed a secure and lightweight communication scheme that can provide authentication and confidentiality to the multicast SMS communication. To achieve privacy preservation in HWMSNs, Xu *et al.* [15] proposed a sanitizable signature scheme that can hide the sensitive data of patients.

In 2003, a cryptographic technology called aggregate signature (AS) was proposed by Boneh *et al.* [16]. They showed that the AS can realize the authentication and integrity of the message with high efficiency, which makes it suitable for resource-constrained environments. Therefore, many authentication schemes using the AS have been proposed [17–22]. In 2004, Lysyanskaya *et al.* [17] constructed an ordered AS scheme based on a one-way function with trapdoors. Signers need to aggregate their signatures in the corresponding order. Whereas Lysyanskaya *et al.*'s scheme is based on the traditional public key infrastructure, which greatly increases the burden of key management and verification overhead. Soon after, Cheon *et al.* [18] proposed the first identity-based AS scheme that avoided complex certificate management issues. However, most identity-based AS schemes suffer from key escrow problems. Certificateless public key cryptography is considered one of the solutions to overcome these [23]. In [24], the full private key consists of the partial private key generated by the key generation center (KGC) and the secret value selected by the unmanned aerial vehicle. The aerial vehicle only knows its secret value and cannot achieve the partial private key of KGC. Hence, Gong *et al.* [20] extended the AS to certificateless public key cryptography and first proposed two certificateless AS (CLAS) schemes. Nevertheless, the complicated verification algorithm caused these two schemes to be inefficient.

Shortly after, the CLAS technology was widely applied to HWMSNs environments to address security and privacy problems. In 2018, Kumar *et al.* [25] designed a CLAS scheme to ensure the secure transmission of medical data in HWMSNs. Nevertheless, Wu *et al.* [26] proved that Kumar *et al.*'s scheme [25] is vulnerable to malicious medical server attacks. To ensure the high efficiency of verification and the identity privacy of patients, Liu *et al.* [27] devised a certificateless anonymous batch verification scheme and asserted that their scheme can authenticate all medical data in one time. Unfortunately, Zhang *et al.* [28] declared that Liu *et al.*'s scheme [27] is unable to withstand malicious participant attacks and malicious data center attacks. In 2019, Gayathri *et al.* [29] devised an anonymous CLAS scheme without bilinear pairings to further reduce the computational overhead. However, Liu *et al.* [30] substantiated that Gayathri *et al.*'s scheme [29] cannot withstand malicious MS attacks and public key replacement attacks. In addition, Liu *et al.* [30] proposed an improved scheme to resist the above attacks.

Recently, Zhan *et al.* [31] found that Liu *et al.*'s improved scheme [30] is insecure for the reason that it cannot withstand malicious MS attacks. To solve these security issues, Zhan *et al.* [31] put forward a pairing-free CLAS (PF-CLAS) scheme for HWMSNs. In addition, Zhan *et al.* [31] asserted that the PF-CLAS scheme has high computational efficiency and is secure against forgery attacks on any message. However, after our analysis, we found that the PF-CLAS scheme is unable to achieve the expected target.

Contribution

The contributions of the proposed work are shown below.

1. We analyze Zhan *et al*.'s PF-CLAS scheme that cannot withstand malicious MSN_i attacks. Simultaneously, the process of how malicious MSN_i attacks successfully forge the signature is shown.
2. The reasons why Zhan *et al*.'s PF-CLAS scheme is insecure against malicious MSN_i attacks are explained. In addition, we design an improved PF-CLAS to address this security vulnerability.
3. We substantiate that our improved PF-CLAS scheme is secure under the random oracle model. Furthermore, the performance evaluation reveals that the proposed scheme is more efficient than the existing related schemes.

Preliminaries

In this section, we introduce the complexity assumption, system model, security requirement, and security model in HWMSNs environments.

Complexity assumption

Elliptic Curve Discrete Logarithm Problem (ECDLP). The group G has the prime order q and generator P. Given two random points P, $Q \in G$, it is hard to work out $a \in Z_q^*$.

Computational Diffie-Hellman Problem (CDHP). The group G has the order q and genera-tor P. Given two random points aP, $bP \in G$, it is hard to work out $abP \in G$, where $a, b \in Z_q^*$.

System model of HWMSNs

As is described in Fig 1, the system model contains four entities: Medical Sensor Node (MSN_i), Medical Server (MS), Cluster Head (CH), and Authorized Healthcare Professionals (AHP). MS is able to generate the public parameters and send it to MSN_i. When MSN_i applies for the partial private key, MS will utilize the master secret key to generate the partial private key and send it to MSN_i. Simultaneously, MSN_i takes advantage of its secret key and partial private key to create the signature and transmits it to CH. Multiple signatures can be aggregated into one signature by CH. Afterward, the aggregate signature can be transmitted to MS by CH. MS sends the aggregate signature to AHP after confirming the validity of the aggregate signature.

Security requirements of HWMSNs

Message Authentication and Integrity. The messages received by the receiver are reliable and have not been tampered with during transmission.

Anonymity. No entity can know the real identity of MSN_i except MS and MSN_i itself.

Traceability. If abnormal MSN_i provides false medical data, MS will trace and extract the real identity of MSN_i.

Security model of HWMSNs

The CLAS scheme contains two types of adversaries: malicious MSN_i and malicious MS.

Malicious MSN_i. It is Type I adversary $\mathcal{A}_1$ in HWMSNs environments. Malicious MSN_i can replace the public key of MSN_i, but it is incapable of achieving the master secret key s.

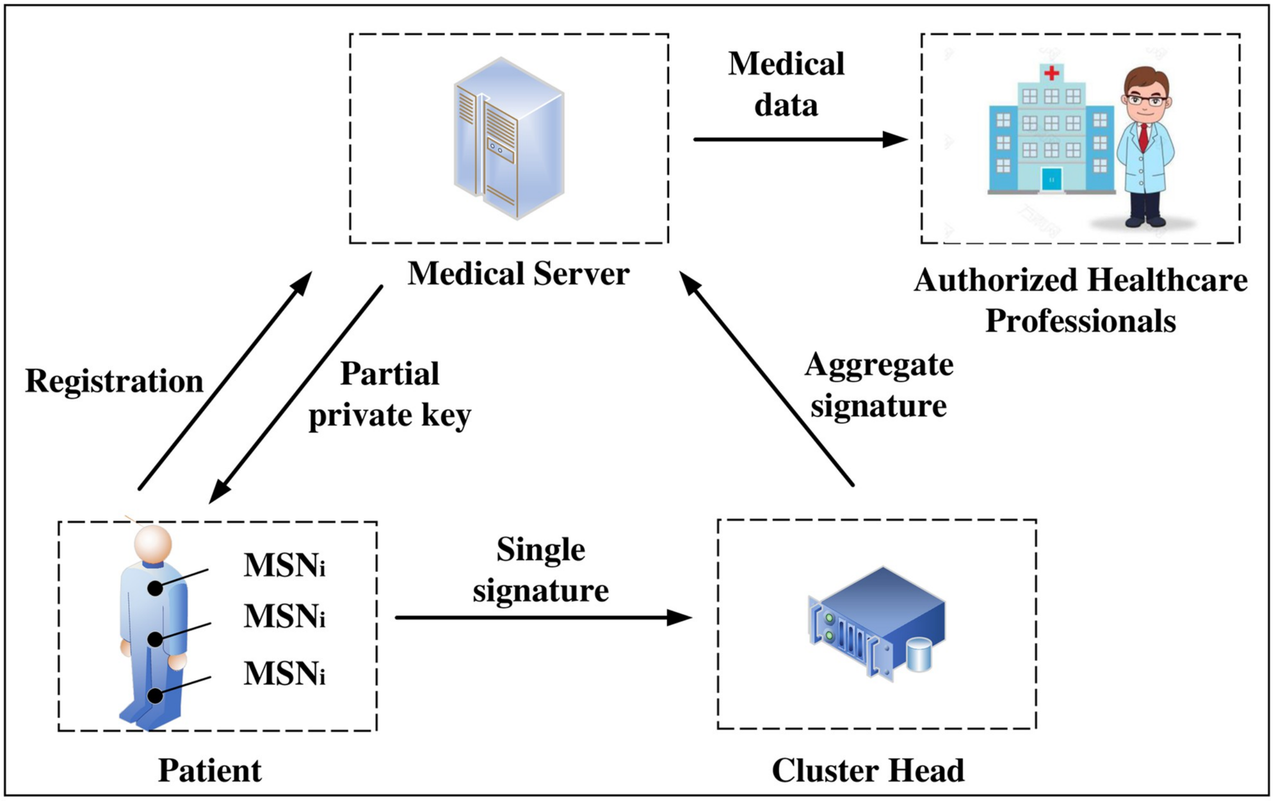

Fig 1. System model for HWMSNs.

https://doi.org/10.1371/journal.pone.0268484.g001

Malicious MS. It is Type II adversary $\mathcal{A}_2$ in HWMSNs environments. Malicious MS can achieve the master secret key s, but it is incapable of replacing public keys.

The existential unforgeability of the PF-CLAS scheme is guaranteed by the following two games.

Game 1:

Setup: The *System Initialization* algorithm is executed by the challenger ζ_1. Given the security parameter v, the algorithm returns system parameters *params* and master secret key s. ζ_1 transmits *params* to $\mathcal{A}_1$ while s is kept secretly.

Query Phase: $\mathcal{A}_1$ carries out a bounded number of queries in polynomial time. The specific process is shown below.

- *PPK Query:* When $\mathcal{A}_1$ makes queries on the partial private key with PID_i, ζ_1 returns d_i to $\mathcal{A}_1$.
- *PK Query:* When $\mathcal{A}_1$ makes queries on the public key of MSN_i, ζ_1 returns PK_i to $\mathcal{A}_1$.
- *SV Query:* When $\mathcal{A}_1$ makes queries on the secret value of MSN_i with PID_i, ζ_1 returns x_i to $\mathcal{A}_1$.
- *PK Replacement Query:* When $\mathcal{A}_1$ chooses a new public key PK_i^* of MSN_i with PID_i, ζ_1 records this replacement.
- *Signature Query:* When $\mathcal{A}_1$ makes queries on the signature with PID_i and PK_i, ζ_1 returns σ_i to $\mathcal{A}_1$ in the tuple (m_i, PID_i, PK_i).

Forgery: $\mathcal{A}_1$ returns identities $\{PID_1^*, \cdots, PID_n^*\}$, public keys $\{PK_1^*, \cdots, PK_n^*\}$, messages $\{m_1^*, \cdots, m_n^*\}$, timestamps $\{t_1^*, \cdots, t_n^*\}$, and an AS σ^*. $\mathcal{A}_1$ can win *Game 1* if the following three situations happen:

1. σ^* is a valid CLAS;
2. *PPK Query* has never been performed for at least one of $\{PID_1^*, \cdots, PID_n^*\}$;
3. *Signature Query* under the tuple (PID_i^*, m_i^*, t_i^*) has never been performed, where $1 \leq i \leq n$.

Game 2:

Setup: The *System Initialization* algorithm is executed by the challenger ζ_2. Given the security parameter v, the algorithm returns system parameters *params* and master key s. ζ_2 transmits *params* and s to $\mathcal{A}_2$.

Query Phase: $\mathcal{A}_2$ carries out a bounded number of queries in polynomial time. The specific process is shown below.

- *PK Query:* When $\mathcal{A}_2$ makes queries on the public key of MSN_i, ζ_2 returns PK_i to $\mathcal{A}_2$.
- *SV Query:* When $\mathcal{A}_2$ makes queries on the secret value of MSN_i with PID_i, ζ_2 returns x_i to $\mathcal{A}_2$.
- *Signature Query:* When $\mathcal{A}_2$ makes queries on the signature with PID_i and PK_i, ζ_2 returns σ_i to $\mathcal{A}_2$ in the tuple (m_i, PID_i, PK_i).

Forgery: $\mathcal{A}_2$ returns identities $\{PID_1^*, \cdots, PID_n^*\}$, public keys $\{PK_1^*, \cdots, PK_n^*\}$, messages $\{m_1^*, \cdots, m_n^*\}$, timestamps $\{t_1^*, \cdots, t_n^*\}$, and an AS σ^*. $\mathcal{A}_2$ can win *Game 2* if the following three situations happen:

1. σ^* is a valid CLAS;
2. *SV Query* has never been performed for at least one of $\{PID_1^*, \cdots, PID_n^*\}$;
3. *Signature Query* under the tuple (PID_i^*, m_i^*, t_i^*) has never been performed, where $1 \leq i \leq n$.

Review of PF-CLAS scheme in [31]

Here, we summarize the notations of the PF-CLAS scheme in Table 1 and review the PF-CLAS scheme in [31].

Table 1. Notations used in PF-CLAS scheme.

Notation	Description
q	A prime number
P	A generator of G
s	Master secret key
P_{pub}	Master public key
k	Security parameter
params	System parameter
RID_i	Real identity of MSN_i
PID_i	Pseudo identity of MSN_i
T_i	Valid time period of pseudo identity
d_i	Partial private key of MSN_i
x_i	Secret value of MSN_i
(pk_i, sk_i)	Public and private key pair of MSN_i
σ	An aggregate signature
t_i	Current timestamp

https://doi.org/10.1371/journal.pone.0268484.t001

- **System Initialization** $(1^k) \rightarrow (params)$: Given the security parameter $k \in Z_q^*$, MS performs the following procedures:
  a. Selects an additive group G of order q and its generator P.
  b. Selects $s \in Z_q^*$ as the master secret key at random and computes $P_{pub} = sP$ as the master public key.
  c. Selects hash functions: $H : G \times G \rightarrow Z_q^*$, $H_1 : \{0,1\}^* \times G \times G \rightarrow Z_q^*$ and $H_2 : \{0,1\}^* \times \{0,1\}^* \times G \times \{0,1\}^* \times G \rightarrow Z_q^*$.
  d. Publishes $params = \{P, G, q, P_{pub}, H, H_{i,\ i=1,2,3}\}$ as the system parameter and keeps s secretly.
- **Generate-PPK** $(params, s, RID_i) \rightarrow (PID_i, d_i)$: Given s, RID_i and $params$, MS performs the following procedures:
  a. Selects $r_i \in Z_q^*$ randomly and calculates $R_i = r_i P$.
  b. Computes $PID_i = RID_i \oplus H(r_i P_{pub}, T_i)$, where T_i is the valid time period of PID_i.
  c. Computes $l_i = H_1(R_i, PID_i, P_{pub})$, $d_i = (r_i + sl_i) \bmod q$.
  d. Sets $D_i = (d_i, R_i)$ as the private key and sends (D_i, PID_i) to MSN$_i$ through secure channels.
- **Generate-PK/SK** $(params, PID_i, d_i) \rightarrow (pk_i, sk_i)$: Given $params$, PID_i and d_i, MSN$_i$ performs the following procedures:
  a. Verifies whether the equation $d_i P = R_i + l_i P_{pub}$ holds, if it holds, MSN$_i$ accepts the private key d_i. Otherwise, it needs to reapply to MS for the partial private key.
  b. Selects $x_i \in Z_q^*$ randomly and calculates $X_i = x_i P$.
  c. Sets $pk_i = (R_i, X_i)$ as its own public key and $sk_i = (d_i, x_i)$ as its own private key.
- **Generate-Signature** $(params, PID_i, pk_i, sk_i, m_i, t_i) \rightarrow (\sigma_i)$: Given $params$, PID_i, pk_i, sk_i, a message m_i and timestamp t_i, MSN$_i$ performs the following procedures:
  a. Chooses $y_i \in Z_q^*$ randomly and calculates $Y_i = y_i P$.
  b. Calculates $a_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$ and $b_i = H_3(PID_i, m_i, t_i, pk_i)$.
  c. Calculates $w_i = [a_i y_i + b_i(d_i + x_i)] \bmod q$.
  d. Outputs $\sigma_i = (Y_i, w_i)$ and transmits $(\sigma_i, m_i, t_i, pk_i)$ to CH through public channels.
- **Verify-Signature** $(params, pk_i, \{m_i, t_i\}) \rightarrow$ **VALID** or **INVALID**: Given $params$, pk_i and a set of message signature pairs (m_i, σ_i), CH performs the following procedures:
  a. Computes $l_i = H_1(R_i, PID_i, P_{pub})$, $a_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$ and $b_i = H_3(PID_i, m_i, t_i, pk_i)$.
  b. Verifies whether the equation $W_i - a_i Y_i = b_i(X_i + R_i + l_i P_{pub})$ holds, if it holds, CH outputs **VALID** and accepts the signature. Otherwise, CH outputs **INVALID** and rejects the signature.
- **Generate-AS** $(params, pk_i, \{m_i, t_i, \sigma_i\}_{1 \le i \le n}) \rightarrow (\sigma)$: Given $params$ and a set of message signature pairs (m_i, σ_i), CH performs the following procedures:
  a. Computes $a_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$.
  b. Computes $A = \sum_{i=1}^{n} a_i Y_i$.

c. Computes $w = \sum_{i=1}^{n} w_i$.

d. Outputs an aggregate signature $\sigma = (A, w)$ and transmits (σ, m_i, t_i, pk_i) to MS through public channels.

- **Verify-AS** $(params, \{m_i, t_i\}_{1 \le i \le n}, \sigma) \rightarrow$ **VALID** or **INVALID**: Given $params$, pk_i, $\{m_i, t_i\}_{1 \le i \le n}$ and σ, MS performs the following procedures:

a. Computes $l_i = H_1(R_i, PID_i, P_{pub})$, $a_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$ and $b_i = H_3(PID_i, m_i, t_i, pk_i)$, where $1 \le i \le n$.

b. Checks whether the equation $wP - A = \sum_{i=1}^{n}[b_i(X_i + R_i + l_i P_{pub})]$ holds. If it holds, MS outputs **VALID** and accepts the aggregate signature σ. Otherwise, MS outputs **INVALID** and rejects the aggregate signature σ.

Cryptanalysis of PF-CLAS schemes

In this section, we first describe the detailed process of malicious MSN$_i$ attacks, and then show the reason why this scheme cannot resist this type of attack. Finally, we present methods to withstand malicious MSN$_i$ attacks.

Forgery attacks from malicious MSN$_i$

Although malicious MSN$_i$ hardly gets the master key s, it can replace the public key pk_i. In addition, if malicious MSN$_i$ eliminates $l_i P_{pub}$ by replacing the public key pk_i, then it will bypass the system master key s to forge a valid signature. Malicious MSN$_i$ can forge the valid signature on any stochastically chosen message m_i^* that satisfies the condition $m_i^* \neq m_i$. The concrete descriptions are shown below.

1. **Public Key Replacement:** Malicious MSN$_i$ executes the following procedures to replace the original public key pk_i.

a. Selects $x_i^{'} \in Z_q^*$ and $r_i^{'} \in Z_q^*$ randomly.

b. Calculates $R_i^{'} = r_i^{'} P$ and $l_i = H_1(PID_i, R_i^{'}, P_{pub})$, where PID_i and P_{pub} are public.

c. Computes $X_i^{'} = x_i^{'} P - l_i P_{pub}$ to replace X_i and sets $pk_i^{'} = (R_i^{'}, X_i^{'})$ as the new public key.

2. **Forgery:** Malicious MSN$_i$ executes the following procedures to forge the signature $\sigma_i^{'}$.

a. Chooses $y_i^{'} \in Z_q^*$ and computes $Y_i^{'} = y_i^{'} P$.

b. Computes $a_i = H_2(PID_i, m_i^*, t_i^*, Y_i^{'}, pk_i^{'})$, $b_i = H_3(PID_i, m_i^*, t_i^*, pk_i^{'})$ and $w_i^{'} = [a_i y_i^{'} + b_i(x_i^{'} + r_i^{'})] \bmod q$.

c. Sets $\sigma_i^{'} = (Y_i^{'}, w_i^{'})$ as the forged signature and sends $< PID_i, pk_i^{'}, m_i^* \| t_i^*, \sigma_i^{'} >$ to CH.

3. **Verification:** CH executes the following procedures to check the validity of the forged signature $\sigma_i^{'}$.

a. Calculates $a_i = H_2(PID_i, m_i^*, t_i^*, Y_i^{'}, pk_i^{'})$, $l_i = H_1(PID_i, P_{pub}, R_i^{'})$ and $b_i = H_3(PID_i, m_i^*, t_i^*, pk_i^{'})$.

b. Checks whether the equation $w_i^{'} P - a_i Y_i^{'} = b_i(X_i^{'} + R_i^{'} + l_i P_{pub})$ holds. If the equation holds, CH takes over the forged signature. Otherwise, malicious MSN$_i$ fails to forge the signature.

4. **Correctness of the Forged Signature:** The validity of forged signature σ_i' is supported by the verifiable equation.

$$
\begin{aligned}
w_i'P - a_iY_i' &= [a_iy_i' + b_i(x_i' + r_i')]P - a_iY_i' \\
&= a_iy_i'P + b_i(x_i'P + r_i'P) - a_iY_i' \\
&= a_iy_i'P + b_i(x_i'P + R_i') - a_iY_i' \\
&= a_iY_i' + b_i(x_i'P + R_i') - a_iY_i' \\
&= a_iY_i' + b_i(X_i' + R_i' + l_iP_{pub}) - a_iY_i' \\
&= b_i(X_i' + R_i' + l_iP_{pub}).
\end{aligned}
$$

Comments on the reason for malicious MSN_i attacks

Although Zhan *et al.*'s scheme [31] has strived to solve the vulnerabilities of Liu *et al.*'s scheme in [30], it still suffers from malicious MSN_i attacks. In Zhan *et al.*'s PF-CLAS scheme [31], there's no connection between d_i and X_i, which is the main reason why malicious MSN_i can succeed in launching public key replacement attacks. The partial private key is defined as $d_i = r_i + sl_i$ in the literature [31], where $l_i = H_1(R_i, PID_i, P_{pub})$. We can easily find that hash function l_i does not contain the public key X_i, implying that the change of X_i cannot influence the partial private key d_i. Hence, malicious MSN_i can bypass d_i by replacing X_i with $X_i' = x_i'P - l_iP_{pub}$. To avoid the public key replacement attacks launched by malicious MSN_i, we only need to add the element X_i to hash functions l_i in **Generate-PPK** algorithm. After modification, it is obvious that the equation $d_i P = R_i + l_i P_{pub}$ will not be valid if the public key X_i is replaced by adversaries, where $l_i = H_1(R_i, PID_i, P_{pub}, X_i)$.

Improved PF-CLAS scheme

In this section, we devise an improved PF-CLAS scheme to avoid malicious MSN_i attacks in HWMSNs. The detailed algorithms are shown as follows.

- **System Initialization** $(1^k)\rightarrow(params)$: Given the security parameter $k \in Z_q^*$, MS performs the following procedures:
  a. Selects an additive group G of order q and its generator P.
  b. Selects $s \in Z_q^*$ as the master secret key at random and computes $P_{pub} = sP$ as the master public key.
  c. Selects hash functions: $H : G \times \{0,1\}^* \rightarrow Z_q^*$, $H_1 : G \times \{0,1\}^* \times G \times G \rightarrow Z_q^*$, $H_2 : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times G \times G \rightarrow Z_q^*$ and $H_3 : \{0,1\}^* \rightarrow Z_q^*$.
  d. Publishes $params = \{P, G, q, P_{pub}, H, H_{i,\ i=1,2,3}\}$ as the system parameter and keeps s secretly.
- **Generate-SV** $(params)\rightarrow(x_i, X_i)$: Given $params$, MSN_i performs the following procedures:
  a. Selects $x_i \in Z_q^*$ randomly and calculates $X_i = x_i P$.
  b. Transmits X_i to MS through public channels.
- **Generate-PPK** $(params, s, RID_i, X_i)\rightarrow(PID_i, d_i)$: Given s, RID_i and $params$, MS performs the following procedures:
  a. Selects $r_i \in Z_q^*$ randomly and calculates $R_i = r_i P$.

b. Computes $PID_i = RID_i \oplus H(r_i P_{pub}, T_i)$, $l_i = H_1(R_i, PID_i, P_{pub}, X_i)$ and $d_i = (r_i + sl_i) \bmod q$.

c. Sets $D_i = (d_i, R_i)$ as the private key and sends (D_i, PID_i) to MSN$_i$ through secure channels.

- **Generate-PK/SK** $(params, PID_i, d_i) \rightarrow (pk_i, sk_i)$: Given $params$, PID_i and d_i, MSN$_i$ performs the following procedures:

a. Verifies whether the equation $d_i P = R_i + l_i P_{pub}$ holds, if it holds, MSN$_i$ accepts the private key d_i. Otherwise, it needs to reapply to MS for the partial private key.

b. Sets $pk_i = (R_i, X_i)$ as its own public key and $sk_i = (d_i, x_i)$ as its own private key.

- **Generate-Signature** $(params, PID_i, pk_i, sk_i, m_i, t_i) \rightarrow (\sigma_i)$: Given $params$, PID_i, pk_i, sk_i, a message m_i and timestamp t_i, MSN$_i$ performs the following procedures:

a. Chooses $y_i \in Z_q^*$ randomly and calculates $Y_i = y_i P$.

b. Calculates $b_i = H_2(PID_i, m_i, t_i, pk_i, Y_i)$ and $w_i = [y_i + b_i(d_i + x_i)] \bmod q$.

c. Outputs $\sigma_i = (Y_i, w_i)$ and transmits $(\sigma_i, m_i, t_i, pk_i)$ to CH through public channels.

- **Verify-Signature** $(params, pk_i, \{m_i, t_i\}) \rightarrow$ **VALID** or **INVALID**: Given $params$, pk_i and a set of message signature pairs (m_i, σ_i), CH performs the following procedures:

a. Computes $l_i = H_1(R_i, PID_i, P_{pub}, X_i)$, $b_i = H_2(PID_i, m_i, t_i, pk_i, Y_i)$.

b. Verifies whether the equation $W_i - Y_i = b_i(X_i + R_i + l_i P_{pub})$ holds, if it holds, CH outputs **VALID** and accepts the signature. Otherwise, CH outputs **INVALID** and rejects the signature.

- **Generate-AS** $(params, pk_i, \{m_i, t_i, \sigma_i\}_{1 \le i \le n}, pk_{ver}) \rightarrow (\sigma)$: Given $params$, pk_{ver} and the tuple (σ_i, m_i, t_i), CH performs the following procedures:

a. Computes $w = \sum_{i=1}^{n} w_i$.

b. Outputs an aggregate signature $\sigma = (Y_1, Y_2, \ldots, Y_n, w)$ and transmits (σ, m_i, t_i, pk_i) to MS through public channels.

- **Verify-AS** $(params, \{m_i, t_i\}_{1 \le i \le n}, \sigma, sk_{ver}) \rightarrow$ **VALID** or **INVALID**: Given $params$, pk_i, $\{m_i, t_i\}_{1 \le i \le n}$ and σ, MS performs the following procedures:

a. Computes $l_i = H_1(R_i, PID_i, P_{pub}, X_i)$ and $b_i = H_2(PID_i, m_i, t_i, pk_i, Y_i)$.

b. Checks whether the equation $wP - \sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} [b_i(X_i + R_i + l_i P_{pub})]$ holds. If it holds, MS outputs **VALID** and accepts σ. Otherwise, MS outputs **INVALID** and rejects σ.

Correctness

Given $params$, pk_i, $\{m_i, t_i\}_{1 \le i \le n}$ and σ_i, the validity of the following equation is checked by CH.

$$\begin{aligned} W_i - Y_i &= [y_i + b_i(d_i + x_i)]P - Y_i \\ &= y_i P + b_i(d_i + x_i)P - Y_i \\ &= Y_i + b_i d_i P + b_i x_i P - Y_i \\ &= b_i(x_i + d_i)P \\ &= b_i(X_i + R_i + l_i P_{pub}). \end{aligned}$$

Given *params*, pk_i, $\{m_i, t_i\}_{1 \leq i \leq n}$ and σ, the validity of the following equation is checked by MS.

$$\begin{aligned} wP - \sum_{i=1}^{n} Y_i &= \sum_{i=1}^{n}[y_i + b_i(d_i + x_i)]P - \sum_{i=1}^{n} Y_i \\ &= \sum_{i=1}^{n} y_i P + \sum_{i=1}^{n} b_i(d_i P + x_i P) - \sum_{i=1}^{n} Y_i \\ &= \sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} b_i(d_i P + x_i P) - \sum_{i=1}^{n} Y_i \\ &= \sum_{i=1}^{n} b_i(d_i P + x_i P) \\ &= \sum_{i=1}^{n}[b_i(X_i + R_i + l_i P_{pub})]. \end{aligned}$$

Security analysis

In this section, we give **Theorem 1** and **Theorem 2** to prove that our improved PF-CLAS scheme can resist malicious MSN_i attacks and malicious MS attacks.

Theorem 1: If $\mathcal{A}_1$ (malicious MSN_i) can successfully forge the signature in polynomial time with the non-negligible probability ε_1, then there will be a challenger ζ_1 that can work out the ECDLP with the probability $\left(1 - \frac{1}{e}\right)\left(\frac{\varepsilon_1}{eq_{h_i}}\right)\left(1 - \frac{1}{q_{ppk}+q_v+q_s+1}\right)$, where e, q_{h_i}, q_s, q_{ppk}, q_v are the natural logarithm base and the most times of *Hash Query*, *Signature Query*, *PPK Query*, *SV Query*.

Proof: The challenger ζ_1 is a solver of the ECDLP. Given the tuple $(P, P_{pub} = sP) \in G \times G$, the goal of ζ_1 is to calculate $s \in Z_q^*$.

Setup: ζ_1 performs **System Initialization** algorithm to generate *params* and s. ζ_1 sends *params* to $\mathcal{A}_1$ and keeps s secretly.

Query Phase: The challenger ζ_1 cannot get the identity PID_i which is selected by $\mathcal{A}_1$. Therefore, ζ_1 guesses a random identity PID_i^* as the identity, where ζ_1 can correctly guess with probability $c = 1 - \frac{1}{q_{ppk}+q_v+q_s+1}$.

- H_1 *Query:* ζ_1 creates an empty $list_1$. When receiving a query $H_1(R_i, PID_i, P_{pub}, X_i)$ from $\mathcal{A}_1$, if there is a tuple $(R_i, PID_i, P_{pub}, X_i, l_i)$ in the $list_1$, ζ_1 will return l_i to $\mathcal{A}_1$; Otherwise, ζ_1 selects $l_i \in Z_q^*$ at random and adds the tuple $(R_i, PID_i, P_{pub}, X_i, l_i)$ into $list_1$. Finally, ζ_1 returns l_i to $\mathcal{A}_1$.
- H_2 *Query:* ζ_1 creates an empty $list_2$. When receiving a query $H_2(m_i, PID_i, t_i, pk_i, Y_i)$ from $\mathcal{A}_1$, if there is a tuple $(m_i, PID_i, t_i, pk_i, Y_i, b_i)$ in the $list_2$, ζ_1 will return b_i to $\mathcal{A}_1$; Otherwise, ζ_1 selects $b_i \in Z_q^*$ at random and adds the tuple $(m_i, PID_i, t_i, pk_i, Y_i, b_i)$ into $list_2$. Finally, ζ_1 returns b_i to $\mathcal{A}_1$.
- *SV Query:* ζ_1 creates an empty $list_3$. When receiving a query about the secret value of MSN_i from $\mathcal{A}_1$, if there is x_i in the $list_3$, ζ_1 will return x_i to $\mathcal{A}_1$; Otherwise, ζ_1 selects $x_i \in Z_q^*$ at random and adds x_i into $list_3$. Finally, ζ_1 returns x_i to $\mathcal{A}_1$.
- *PPK Query:* ζ_1 creates an empty $list_4$. When receiving a query about the partial private key of MSN_i with PID_i from $\mathcal{A}_1$, if there is a tuple (R_i, PID_i, d_i) in the $list_4$, ζ_1 will return (R_i, d_i) to $\mathcal{A}_1$; Otherwise, ζ_1 queries the corresponding tuple $(R_i, PID_i, P_{pub}, X_i, l_i)$ of MSN_i with $PID_i \in list_1$, selects $d_i \in Z_q^*$ at random, computes $R_i = d_i P - l_i P_{pub}$ and adds the tuple (R_i, PID_i, d_i) into $list_4$. Finally, ζ_1 returns (R_i, d_i) to $\mathcal{A}_1$.

- *PK Query:* ζ_1 creates an empty $list_5$. When receiving a query about the public key of MSN$_i$ with PID_i from $\mathcal{A}_1$, if there is a tuple (R_i, PID_i, X_i) in the $list_5$, ζ_1 will return (R_i, X_i) to $\mathcal{A}_1$; Otherwise, ζ_1 performs following steps.

    1. If $PID_i \neq PID_i^*$, ζ_1 selects $x_i, d_i, l_i \in Z_q^*$ at random, computes $X_i = x_i P$ and $R_i = d_i P - l_i P_{pub}$. Then, ζ_1 adds the tuple (R_i, PID_i, X_i) into $list_5$ and returns (R_i, X_i) to $\mathcal{A}_1$.
    2. If $PID_i = PID_i^*$, ζ_1 selects $x_i, r_i \in Z_q^*$ at random, computes $X_i = x_i P$ and $R_i = r_i P$. Then, ζ_1 sets d_i as $\perp$ and adds the tuple (R_i, PID_i, X_i) into $list_5$. Finally, it returns (R_i, X_i) to $\mathcal{A}_1$.

- *PK Replacement Query:* When $\mathcal{A}_1$ selects a new public key $pk_i^* = (X_i^*, R_i^*)$ and sends (PID_i, pk_i^*) to ζ_1. When receiving a query about the public key replacement of MSN$_i$ with PID_i from $\mathcal{A}_1$, ζ_1 updates $list_5$ and records this replacement.

- *Signature Query:* ζ_1 creates an empty $list_6$. When receiving a query about the signature of MSN$_i$ with PID_i from $\mathcal{A}_1$, if there is a tuple $(m_i, PID_i, x_i, \omega_i)$ in the $list_6$, ζ_1 selects $y_i \in Z_q^*$ at random, computes $Y_i = y_i P$, $b_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$ and $w_i = y_i + b_i(x_i + d_i) \bmod q$. Then ζ_1 returns (Y_i, w_i) to $\mathcal{A}_1$; Otherwise, ζ_1 selects $w_i \in Z_q^*$ at random, computes $Y_i = w_i P - b_i(X_i + R_i + l_i P_{pub})$ and adds the tuple (Y_i, w_i) into $list_6$. Finally, ζ_1 returns (Y_i, w_i) to $\mathcal{A}_1$.

Forgery: After polynomial bounded times of queries, $\mathcal{A}_1$ outputs forged signature $\sigma_i^* = (Y_i^*, w_i^*)$ under the tuple $(PID_i^*, m_i^*, t_i^*, X_i^*)$. According to the forking lemma [32], $\mathcal{A}_1$ generates another forged signature $\sigma_i^{*(2)} = (Y_i^{*(2)}, w_i^{*(2)})$. Therefore, according to the equation $w_i^* P = Y_i^* + b_i^*(X_i^* + R_i^* + l_i^* P_{pub})$ and the equation $w_i^{*(2)} P = Y_i^{*(2)} + b_i^{*(2)}(X_i^{*(2)} + R_i^{*(2)} + l_i^* P_{pub})$, s can be obtained as a valid solution. Otherwise, ζ_1 cannot handle the ECDLP.

In order to succeed in forging a signature, the outputs of ζ_1 need to satisfy the following conditions:

1. T_1: ζ_1 has never aborted the process of quering;
2. T_2: ζ_1 has never aborted the process of forging the signature;
3. T_3: σ_i^* is a valid signature.

According to the above conditions, we can get that $P_r[T_1] \geq 1 - c$, $P_r[T_1 \mid T_2] \geq (1 - c)c^{q_{ppk}+q_v+q_s}$ and $P_r[T_1 \mid T_2 \wedge T_3] \geq (1-c)c^{q_{ppk}+q_v+q_s}\left(1-\frac{1}{e}\right)\frac{\varepsilon_1}{q_{h_i}} \geq \left(1-\frac{1}{e}\right)\left(\frac{\varepsilon_1}{eq_{h_i}}\right)\left(1-\frac{1}{q_{ppk}+q_v+q_s+1}\right)$. Consequently, the probability that ζ_1 can work out the ECDLP is $\left(1-\frac{1}{e}\right)\left(\frac{\varepsilon_1}{eq_{h_i}}\right)\left(1-\frac{1}{q_{ppk}+q_v+q_s+1}\right)$.

Theorem 2: If $\mathcal{A}_2$ (malicious MS) can successfully forge the signature in polynomial time with the non-negligible probability ε_2, then there will be a challenger ζ_2 that can work out the ECDLP with the probability $\left(1-\frac{1}{e}\right)\left(\frac{\varepsilon_2}{eq_{h_i}}\right)\left(1-\frac{1}{q_v+q_s+1}\right)$, where e, q_{h_i}, q_s, q_v are the natural logarithm base and the most times of *Hash Query*, *Signature Query*, *SV Query*.

Proof: The challenger ζ_2 is a solver of the ECDLP. Given the tuple $(P, X_i = x_i P) \in G \times G$, the goal of ζ_2 is to calculate $x_i \in Z_q^*$.

Setup: ζ_2 performs **System Initialization** algorithm to generate *params* and s. ζ_2 sends *params* and s to $\mathcal{A}_2$.

Query Phase: The challenger ζ_2 cannot get the identity PID_i which is selected by $\mathcal{A}_2$. Therefore, ζ_2 guesses a random identity PID_i^* as the identity, where ζ_2 can correctly guess with probability $c = 1 - \frac{1}{q_v+q_s+1}$.

- *H_1 Query:* ζ_2 creates an empty $list_1$. When receiving a query $H_1(R_i, PID_i, P_{pub}, X_i)$ from $\mathcal{A}_2$, if there is a tuple $(R_i, PID_i, P_{pub}, X_i, l_i)$ in the $list_1$, ζ_2 will return l_i to $\mathcal{A}_2$; Otherwise, ζ_2 selects $l_i \in Z_q^*$ at random and adds the tuple $(R_i, PID_i, P_{pub}, X_i, l_i)$ into $list_1$. Finally, ζ_2 returns l_i to $\mathcal{A}_2$.
- *H_2 Query:* ζ_2 creates an empty $list_2$. When receiving a query $H_2(m_i, PID_i, t_i, pk_i, Y_i)$ from $\mathcal{A}_2$, if there is a tuple $(m_i, PID_i, t_i, pk_i, Y_i, b_i)$ in the $list_2$, ζ_2 will return b_i to $\mathcal{A}_2$; Otherwise, ζ_2 selects $b_i \in Z_q^*$ at random and adds the tuple $(m_i, PID_i, t_i, pk_i, Y_i, b_i)$ into $list_2$. Finally, ζ_2 returns b_i to $\mathcal{A}_2$.
- *SV Query:* ζ_2 creates an empty $list_3$. When receiving a query about the secret value of MSN_i from $\mathcal{A}_2$, if there is x_i in the $list_3$, ζ_2 will return x_i to $\mathcal{A}_2$; Otherwise, ζ_2 selects $x_i \in Z_q^*$ at random and adds the tuple x_i into $list_3$. Finally, ζ_2 returns x_i to $\mathcal{A}_2$.
- *PK Query:* ζ_2 creates an empty $list_4$. When receiving a query about the public key of MSN_i with PID_i from $\mathcal{A}_2$, if there is a tuple (R_i, PID_i, X_i) in the $list_4$, ζ_2 will return (R_i, X_i) to $\mathcal{A}_2$; Otherwise, ζ_2 performs following steps.
    1. If $PID_i \neq PID_i^*$, ζ_2 selects $x_i, d_i, l_i \in Z_q^*$ at random, computes $X_i = x_i P$ and $R_i = d_i P - l_i P_{pub}$. Then, ζ_2 adds the tuple (R_i, PID_i, X_i) into $list_4$ and returns (R_i, X_i) to $\mathcal{A}_2$.
    2. If $PID_i = PID_i^*$, ζ_2 selects $x_i, r_i \in Z_q^*$ at random, computes $X_i = x_i P$ and $R_i = r_i P$. Then, ζ_2 sets d_i as $\perp$ and adds the tuple (R_i, PID_i, X_i) into $list_4$. Finally, it returns (R_i, X_i) to $\mathcal{A}_2$.
- *Signature Query:* ζ_2 creates an empty $list_5$. When receiving a query about the signature of MSN_i with PID_i from $\mathcal{A}_2$, if there is a tuple $(m_i, PID_i, x_i, \omega_i)$ in the $list_5$, ζ_2 selects $y_i \in Z_q^*$ at random, computes $Y_i = y_i P$, $b_i = H_2(PID_i, m_i, t_i, Y_i, pk_i)$ and $w_i = y_i + b_i(x_i + d_i) \bmod q$. Then ζ_2 returns (Y_i, w_i) to $\mathcal{A}_2$; Otherwise, ζ_2 selects $w_i \in Z_q^*$ at random, computes $Y_i = w_i P - b_i(X_i + R_i + l_i P_{pub})$ and adds the tuple (Y_i, w_i) into $list_5$. Finally, ζ_2 returns (Y_i, w_i) to $\mathcal{A}_2$.

Forgery: After polynomial bounded times of queries, $\mathcal{A}_2$ outputs forged signature $\sigma_i^* = (Y_i^*, w_i^*)$ under the tuple $(PID_i^*, m_i^*, t_i^*, R_i^*)$. According to the forking lemma [32], $\mathcal{A}_2$ generates another forged signature $\sigma_i^{*(2)} = (Y_i^{*(2)}, w_i^{*(2)})$. Therefore, according to the equation $w_i^* P = Y_i^* + b_i^*(X_i + R_i^* + h_i^* P_{pub})$ and the equation $w_i^{*(2)} P = Y_i^{*(2)} + b_i^{*(2)}(X_i + R_i^{*(2)} + h_i^{*(2)} P_{pub})$, x_i can be obtained as a valid solution. Otherwise, ζ_2 cannot handle the ECDLP.

In order to succeed in forging a signature, the outputs of ζ_2 need to satisfy the following conditions:

1. T_1: ζ_2 has never aborted the process of quering;
2. T_2: ζ_2 has never aborted the process of forging the signature;
3. T_3: σ_i^* is a valid signature.

According to the above conditions, we can get that $P_r[T_1] \geq 1 - c$, $P_r[T_1 \mid T_2] \geq (1-c)c^{q_v+q_s}$ and $P_r[T_1 \mid T_2 \wedge T_3] \geq (1-c)c^{q_v+q_s}\left(1-\frac{1}{e}\right)\frac{\varepsilon_2}{q_{h_i}} \geq \left(1-\frac{1}{e}\right)\left(\frac{\varepsilon_2}{eq_{h_i}}\right)\left(1-\frac{1}{q_v+q_s+1}\right)$. Consequently, the probability that ζ_2 can work out the ECDLP is $\left(1-\frac{1}{e}\right)\left(\frac{\varepsilon_2}{eq_{h_i}}\right)\left(1-\frac{1}{q_v+q_s+1}\right)$.

Other security analysis

1. *Message authentication and integrity:* According to **Theorem 1** and **Theorem 2**, neither Type I nor Type II attackers can pass the verification by forging a signature.

Table 2. Runtime of cryptographic operations.

Operations	Abbreviations	Runtime (ms)
Pairing-based scalar multiplication	T_{sm}	2.2560
Pairing-based point addition	T_{pa}	0.1732
Bilinear pairing computation	T_p	4.6028
Map-to-point hash	T_h	5.1240
ECC-based scalar multiplication	T_{esm}	0.7648
ECC-based point addition	T_{epa}	0.0435

https://doi.org/10.1371/journal.pone.0268484.t002

2. *Anonymity*: In the improved PF-CLAS scheme, PID_i is the pseudo identity of MSN_i, where $PID_i = RID_i \oplus H(r_i P_{pub}, T_i)$. Any adversary cannot extract the real identity of MSN_i. Hence, our scheme provides strong anonymity.
3. *Traceability*: If MSN_i transmits illegal information, MS can track abnormal MSN_i and extract its real identity by computing $RID_i = PID_i \oplus H(sR_i)$, where $sR_i = r_i P_{pub}$.

Performance evaluation

In this section, we will provide the performance analysis in terms of computational overhead, communication overhead, and security features. In the meantime, the efficiency of the improved scheme will be compared with the related schemes [15, 24, 25, 27, 29, 33, 34]. We utilize MIRACL library to simulate cryptographic operations on a Windows 10 laptop with an Intel i7–1195G7 @2.9 GHz processor and 8 GB of memory. The measured runtime of different operations is shown in Table 2.

Computational overhead

As is described in Table 3, we mainly count the computational overhead of **Generate-Signature** algorithm, **Verify-Signature** algorithm, **Generate-AS** algorithm, and **Verify-AS** algorithm. In Xu *et al.*'s scheme [15], the computational overhead of the single signing and verification is ≈ 143.7864 ms. Similarly, Kumar *et al.*'s scheme [25], Liu *et al.*'s scheme [27], and Shen *et al.*'s scheme [34] need 38.724 ms, 21.444ms, 36.1212 ms, respectively. As is shown in Fig 2 the computational overhead of the above schemes is extremely high. The root cause is that these schemes all use bilinear pairing and map-to-point hash operations to construct the signature. Hence, we use pairing-free operations to improve the efficiency of the improved

Table 3. Comparison of computational overhead.

Schemes	Sign (ms)	Verify (ms)	AggregateSign (ms)	AggregateVerify (ms)
[15]	$14T_{sm}+4T_{pa}+T_h \approx 37.4008$	$22T_p+T_h \approx 106.3856$	—	—
[24]	$T_{esm} \approx 0.7648$	$4T_{esm}+3T_{epa} \approx 3.1897$	—	—
[25]	$3T_{sm}+2T_{pa}+T_h \approx 12.2384$	$3T_p+T_{sm}+T_{pa}+2T_h \approx 26.4856$	$(n-1)T_{pa} \approx 8.4846$	$3T_p+nT_{sm}+(3n-2)T_{pa}+(n+1)T_h \approx 413.566$
[27]	$2T_{sm}+T_{pa} \approx 4.6852$	$2T_p+T_{sm}+T_{pa}+T_h \approx 16.7588$	$nT_{sm}+(3n-1)T_{pa} \approx 138.607$	$2T_p+T_{pa} \approx 9.3788$
[29]	$2T_{esm} \approx 1.5296$	$5T_{esm}+3T_{epa} \approx 3.9545$	$2nT_{esm}+(2n-2)T_{epa} \approx 11.911$	$(2n+1)T_{esm}+(2n+1)T_{epa} \approx 81.6383$
[33]	$2T_{esm} \approx 1.5296$	$4T_{esm}+3T_{epa} \approx 3.1897$	$(n-1)T_{epa} \approx 2.1315$	$(2n+1)T_{esm}+3nT_{epa} \approx 83.7698$
[34]	$3T_{sm}+T_{pa}+T_h \approx 12.0652$	$3T_p+2T_h \approx 24.0560$	$nT_p+T_{sm}+(n-1)T_{pa} \approx 236.28$	$nT_p+T_{sm}+(n-1)T_{pa} \approx 236.28$
Our scheme	$T_{esm} \approx 0.7648$	$4T_{esm}+3T_{epa} \approx 3.1897$	$(n-1)T_{epa} \approx 2.1315$	$(2n+1)T_{esm}+(4n-1)T_{epa} \approx 85.9013$

We set the number of signatures participating in the aggregation as $n = 50$.

https://doi.org/10.1371/journal.pone.0268484.t003

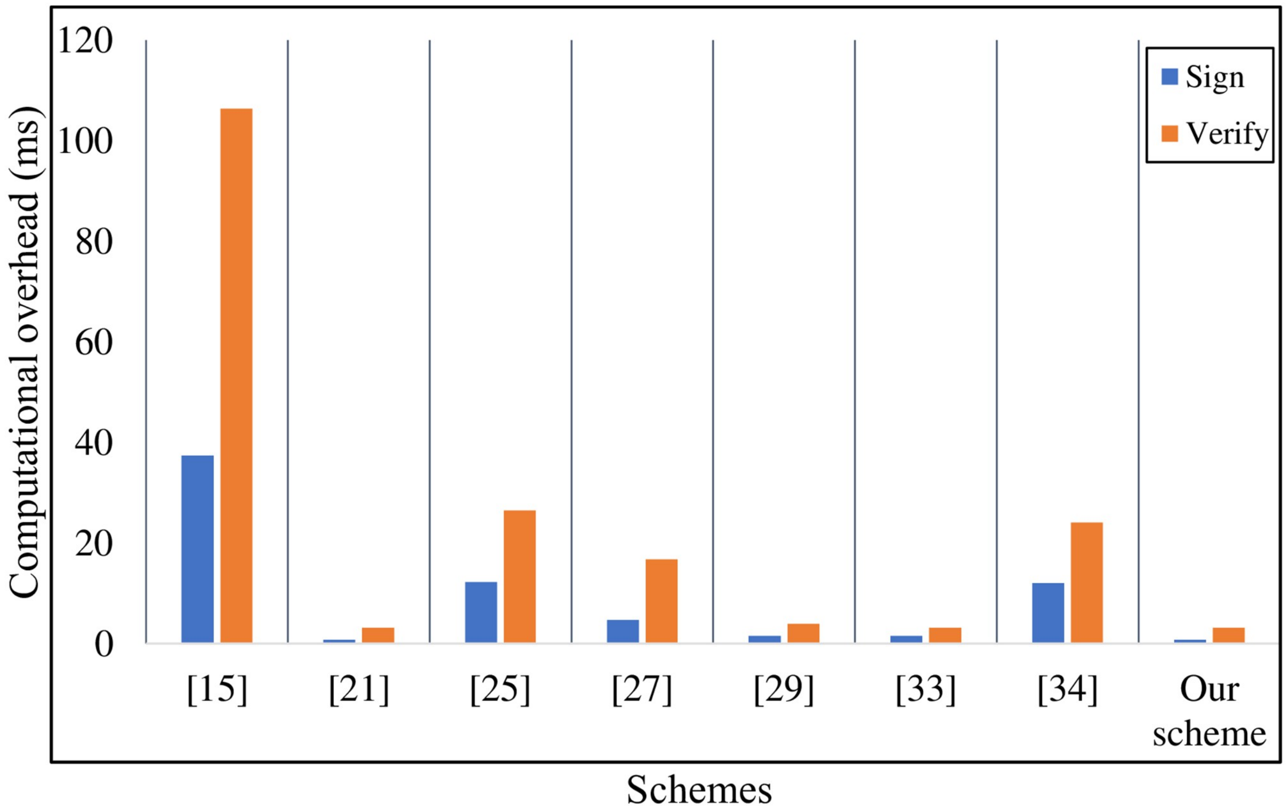

Fig 2. Computational overhead of the single signing and verification.

https://doi.org/10.1371/journal.pone.0268484.g002

PF-CLAS scheme. In literatures [24, 29, 33], their schemes also don't use bilinear pairings. Hence, they only need 3.9545 ms, 5.4841 ms, 4.1793 ms, respectively. The computational overhead of the single signing and verification only needs 3.9545ms, which saves 97.2%, 89.8%, 81.6%, 27.9%, 5.4%, 89.1% of the computational overhead than Xu *et al.*'s scheme [15], Kumar *et al.*'s scheme [25], Liu *et al.*'s scheme [27], Gayathri *et al.*'s scheme [29], Verma *et al.*'s scheme [33], Shen *et al.*'s scheme [34]. In the aggregate signing and aggregate verification phases, we set the number of signatures participating in the aggregation as $n = 50$. Since references [15, 24] have no connection with the aggregate signature, we don't describe them too much. As is shown in Fig 3, the computational overhead of the aggregate signing and verification of Kumar *et al.*'s scheme [25], Liu *et al.*'s scheme [27], Gayathri *et al.*'s scheme [29], Verma *et al.*'s scheme [33], Shen *et al.*'s scheme [34] is 422.0506 ms, 147.9858 ms, 95.5493 ms, 85.9013 ms, 472.56 ms, respectively. Our improved PF-CLAS scheme needs 88.0328 ms, which saves 79.2%, 41%, 7.9%, 27.9%, 5.4%, 81.4% than Kumar *et al.*'s scheme [25], Liu *et al.*'s scheme [27], Gayathri *et al.*'s scheme [29], Shen *et al.*'s scheme [34]. Although the total computational overhead of Verma *et al.*'s scheme [33] is basically the same as our scheme, Verma *et al.*'s scheme [33] cannot achieve secure communication. Hence, the computational overhead of our improved PF-CLAS scheme reaches the upstream level of the relevant schemes.

Communication overhead

As shown in Table 4, we list parameters and length specifications for pairing-based and ECC-based schemes [29]. In addition, the size of the group $|\ Z_q^*\ |$ is 160 bits in our scheme. In [15,

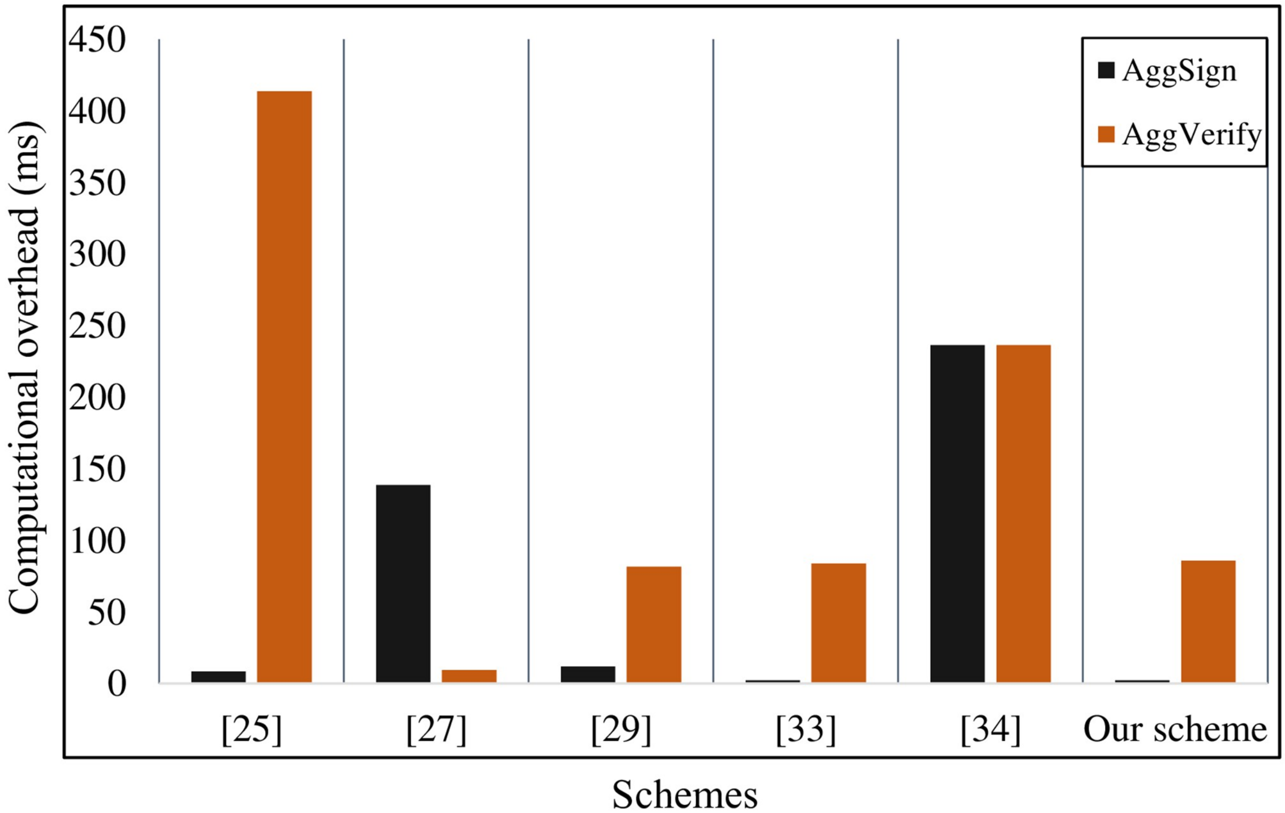

Fig 3. Computational overhead of the aggregate signing and aggregate verification.

https://doi.org/10.1371/journal.pone.0268484.g003

25, 27, 34], the communication overhead of the single signature is 1024 bits, 2048 bits, 2048 bits, and 2048 bits, respectively, because all the elements of σ_i belong to G_1. In our improved PF-CLAS, we set σ_i as (Y_i, w_i), where $Y_i \in G$, $w_i \in Z_q^*$. Compared with the schemes [15, 24, 25, 27, 29, 34], the communication overhead of the single signature in our scheme is reduced by 53.1%, 40%, 76.57%, 76.57%, 25%, 76.57%. As is described in Fig 4, it is obvious that our scheme has higher efficiency than the above schemes in the single signature phase. Since references [15, 24] have no connection with the aggregate signature, we don't describe them too much in the aggregate signature phase. In the meantime, we can know from Fig 5 that the communication overhead of the aggregate signatures in our scheme is lower than Kumar *et al.*'s scheme [25] and Shen *et al.*'s scheme [34] with the increase of the number of medical sensor nodes. Although Liu *et al.*'s scheme [27] and Gayathri *et al.*'s scheme [29] have lower communication overhead than our scheme, their schemes have serious security flaws. As shown in Table 5, even though our scheme has the same communication overhead as Verma *et al.*'s scheme [33], their scheme cannot meet the security requirements of HWMSNs. Therefore, our scheme has certain advantages in terms of communication overhead.

Table 4. Length of parameters in bilinear pairing and ECC.

Type of the scheme	Type of the curve	Pairing	Cyclic group	Size of the prime	Size of the group
Bilinear Pairing	$E: y^2 = x^3 + x \bmod p$	$e: G_1 \times G_1 \rightarrow G_T$	$G_1(P)$	p=512 bits	$\|G_1\|$=1024 bits
ECC	$E: y^2 = x^3 + ax + b \bmod p$	—	$G(P)$	p=160 bits	$\|G\|$=320 bits

https://doi.org/10.1371/journal.pone.0268484.t004

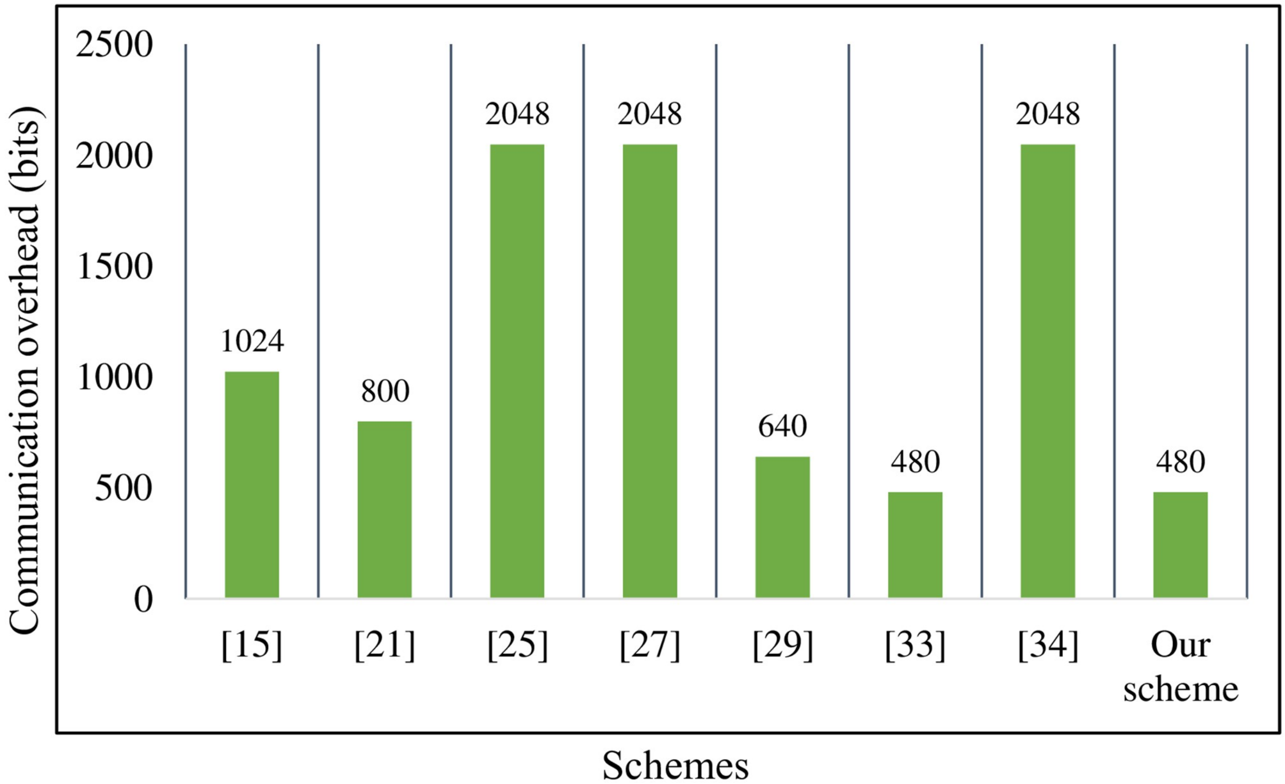

Fig 4. Communication overhead of single signatures.

https://doi.org/10.1371/journal.pone.0268484.g004

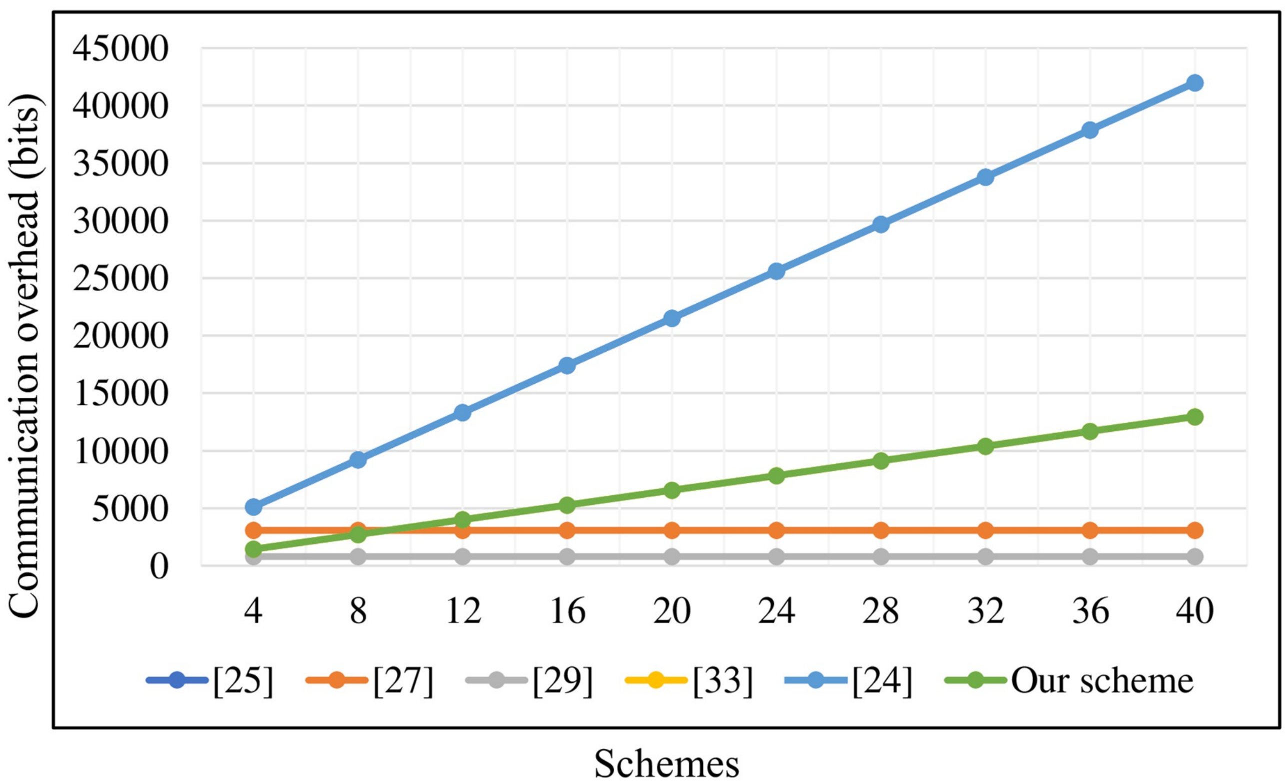

Fig 5. Communication overhead of aggregate signatures.

https://doi.org/10.1371/journal.pone.0268484.g005

Table 5. Comparison of communication overhead and security features.

Schemes	Single signatures	Aggregate signatures	Type I attacks	Type II attacks	Anonymity	Traceability
[15]	$\|G_1\| = 1024$ bits	—	—	—	×	×
[24]	$\|G\| + 3\|Z_q^*\| = 800$ bits	—	✓	✓	×	×
[25]	$2\|G_1\| = 2048$ bits	$(n+1)\|G_1\| = 1024(n+1)$ bits	✓	×	×	×
[27]	$2\|G_1\| = 2048$ bits	$3\|G_1\| = 3072$ bits	×	×	✓	×
[29]	$\|G\| + 2\|Z_q^*\| = 640$ bits	$2\|G\| + \|Z_q^*\| = 800$ bits	×	×	✓	✓
[33]	$\|G\| + \|Z_q^*\| = 480$ bits	$n\|G\| + \|Z_q^*\| = 160(2n+1)$ bits	×	✓	×	×
[34]	$2\|G_1\| = 2048$ bits	$(n+1)\|G_1\| = 1024(n+1)$ bits	✓	✓	×	×
Our scheme	$\|G\| + \|Z_q^*\| = 480$ bits	$n\|G\| + \|Z_q^*\| = 160(2n+1)$ bits	✓	✓	✓	✓

https://doi.org/10.1371/journal.pone.0268484.t005

Security features

As shown in Table 5, Xu *et al.*'s scheme [15], Xu *et al.*'s scheme [24], Kumar *et al.*'s scheme [25], Verma *et al.*'s scheme [33], and Shen *et al.*'s scheme [34] don't consider the anonymity of patients' identities and tracing of malicious medical sensor nodes, which are unsuitable for HWMSNs scenarios. Although Liu *et al.*'s scheme [27] and Gayathri *et al.*'s scheme [29] can meet the security requirements of HWMSNs, these schemes have security drawbacks that cannot withstand Type I and Type II attacks. The proposed scheme has been proved that resist Type I and Type II attacks under the random oracle model. Besides, our scheme is able to realize anonymity and traceability, which is more practical in HWMSNs.

Conclusion

In this paper, we found that Zhan *et al.*'s PF-CLAS scheme [31] cannot withstand malicious MSN_i attacks. In the meantime, we showed the reason why this scheme was vulnerable to malicious MSN_i attacks. It is obvious that Zhan *et al.*'s scheme cannot guarantee the identity privacy of patients and secure transmission of medical data. Hence, we gave methods to fix the vulnerability and constructed an improved PF-CLAS scheme that could ensure provable security. In addition, the performance evaluation indicated that our improved scheme can realize privacy preservation and secure communication at low overhead. In the future, how to combine blockchain and edge computing technologies to design a more lightweight and secure CLAS scheme for HWMSNs is still an interesting problem.

Supporting information

S1 Data. Runtime of cryptographic operations.
(XLS)

S2 Data. Comparison of computational overhead.
(XLS)

Acknowledgments

We thank the anonymous reviewer for your review and approval.

Author Contributions

Conceptualization: Lifeng Zhou.

Data curation: Lifeng Zhou.

Formal analysis: Lifeng Zhou.

Funding acquisition: Xinchun Yin.

Investigation: Lifeng Zhou.

Methodology: Lifeng Zhou.

Project administration: Xinchun Yin.

Resources: Xinchun Yin.

Software: Lifeng Zhou.

Supervision: Xinchun Yin.

Validation: Xinchun Yin.

Visualization: Lifeng Zhou.

Writing – original draft: Lifeng Zhou.

Writing – review & editing: Xinchun Yin.

References

1. Hossain ME, Khan A, Moni MA, Uddin S. Use of electronic health data for disease prediction: A comprehensive literature review. IEEE/ACM transactions on computational biology and bioinformatics. 2019; 18(2):745–758. https://doi.org/10.1109/TCBB.2019.2937862
2. Masud M, Gaba GS, Alqahtani S, Muhammad G, Gupta BB, Kumar P, et al. A Lightweight and Robust Secure Key Establishment Protocol for Internet of Medical Things in COVID-19 Patients Care. IEEE Internet of Things Journal. 2021; 8(21):15694–15703. https://doi.org/10.1109/JIOT.2020.3047662
3. Vijayakumar P, Obaidat MS, Azees M, Islam SH, Kumar N. Efficient and secure anonymous authentication with location privacy for IoT-based WBANs. IEEE Transactions on Industrial Informatics. 2019; 16(4):2603–2611. https://doi.org/10.1109/TII.2019.2925071
4. Xu Z, He D, Kumar N, Choo K-KR. Efficient certificateless aggregate signature scheme for performing secure routing in VANETs. Security and Communication Networks. 2020; 2020(2):1–12. https://doi.org/10.1155/2020/8872586
5. Kumar M, Chand S. A lightweight cloud-assisted identity-based anonymous authentication and key agreement protocol for secure wireless body area network. IEEE Systems Journal. 2021; 15(2):2779–2786. https://doi.org/10.1109/JSYST.2020.2990749
6. Jegadeesan S, Azees M, Sekar A, Al-Turjman F. Lightweight Privacy and Confidentiality Preserving Anonymous Authentication Scheme for WBANs. IEEE Transactions on Industrial Informatics. 2022; 18(5): 3484–3491. https://doi.org/10.1109/TII.2021.3097759
7. Ye X, Xu G, Cheng X, Li Y, Qin Z. Certificateless-based anonymous authentication and aggregate signature scheme for vehicular ad hoc networks. Wireless Communications and Mobile Computing. 2021; 2021(5):1–16. https://doi.org/10.1155/2021/5574255
8. Odelu V, Saha S, Prasath R, Sadineni L, Conti M, Jo M. Efficient privacy preserving device authentication in WBANs for industrial e-health applications. Computers & Security. 2019; 83:300–312. https://doi.org/10.1016/j.cose.2019.03.002
9. Al-Ayyoub M, AlZu'bi S, Jararweh Y, Shehab MA, Gupta BB. Accelerating 3D medical volume segmentation using GPUs. Multimedia Tools and Applications. 2018; 77(4):4939–4958. https://doi.org/10.1007/s11042-016-4218-0
10. Al-Qerem A, Alauthman M, Almomani A, Gupta B. IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. Soft Computing. 2020; 24(8):5695–5711. https://doi.org/10.1007/s00500-019-04220-y
11. Gupta BB, Li K-C, Leung VC, Psannis KE, Yamaguchi S. Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system. IEEE/CAA Journal of Automatica Sinica. 2021; 8(12):1877–1890. https://doi.org/10.1109/JAS.2021.1004003
12. Nguyen GN, Le Viet NH, Elhoseny M, Shankar K, Gupta B, Abd El-Latif AA. Secure blockchain enabled Cyber–physical systems in healthcare using deep belief network with ResNet model. Journal of Parallel and Distributed Computing. 2021; 153:150–160. https://doi.org/10.1016/j.jpdc.2021.03.011

13. Mirsadeghi F, Rafsanjani MK, Gupta BB. A trust infrastructure based authentication method for clustered vehicular ad hoc networks. Peer-to-Peer Networking and Applications. 2021; 14(4):2537–2553. https://doi.org/10.1007/s12083-020-01010-4

14. Vijayakumar P, Pandiaraja P, Karuppiah M, Deborah LJ. An efficient secure communication for healthcare system using wearable devices. Computers & Electrical Engineering. 2017; 63:232–245. https://doi.org/10.1016/j.compeleceng.2017.04.014

15. Xu Z, Luo M, Kumar N, Vijayakumar P, Li L. Privacy-protection scheme based on sanitizable signature for smart mobile medical scenarios. Wireless Communications and Mobile Computing. 2020; 2020 (1):1–10. https://doi.org/10.1155/2020/8874862

16. Boneh D, Gentry C, Lynn B, Shacham H. Aggregate and verifiably encrypted signatures from bilinear maps. International conference on the theory and applications of cryptographic techniques. 2003:416-432.

17. Lysyanskaya A, Micali S, Reyzin L, Shacham H, editors. Sequential aggregate signatures from trapdoor permutations. International Conference on the Theory and Applications of Cryptographic Techniques. 2004:74-90.

18. Cheon JH, Kim Y, Yoon HJ. A new ID-based signature with batch verification. Cryptology EPrint Archive. 2004: 131.

19. Lin X, Sun X, Ho PH, Shen X. GSIS: A Secure and Privacy-Preserving Protocol for Vehicular Communications. IEEE Transactions on Vehicular Technology. 2007; 56(6):3442–3456. https://doi.org/10.1109/TVT.2007.906878

20. Gong Z, Long Y, Hong X, Chen K. Two certificateless aggregate signatures from bilinear maps. Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. 2007:188-193.

21. Ma D. Practical forward secure sequential aggregate signatures. Proceedings of the 2008 ACM symposium on Information, computer and communications security. 2008:341–352. https://doi.org/10.1145/1368310.1368361

22. Chen J, Yue H, Huang Z. Secure certificate-based aggregate signature scheme. Computer Engineering and Applications. 2013; 49(21):60–64.

23. Xu Z, He D, Vijayakumar P, Choo K-KR, Li L. Efficient NTRU lattice-based certificateless signature scheme for medical cyber-physical systems. Journal of medical systems. 2020; 44(5):1–8. https://doi.org/10.1007/s10916-020-1527-7 PMID: 32189085

24. Xu Z, Luo M, Vijayakumar P, Peng C, Wang L. Efficient certificateless designated verifier proxy signature scheme using UAV network for sustainable smart city. Sustainable Cities and Society. 2022; 80:103771. https://doi.org/10.1016/j.scs.2022.103771

25. Kumar P, Kumari S, Sharma V, Sangaiah AK, Wei J, Li X. A certificateless aggregate signature scheme for healthcare wireless sensor network. Sustainable Computing: Informatics and Systems. 2018; 18:80–89.

26. Wu L, Xu Z, He D, Wang X. New certificateless aggregate signature scheme for healthcare multimedia social network on cloud environment. Security and Communication Networks. 2018; 2018:1–13. https://doi.org/10.1155/2018/5783976

27. Liu J, Cao H, Li Q, Cai F, Du X, Guizani M. A large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing. IEEE Internet of things Journal. 2018; 6(2):1321–1330. https://doi.org/10.1109/JIOT.2018.2828463

28. Zhang Y, Shu J, Liu X, Jin L, Dong Z. Comments on "A Large-Scale Concurrent Data Anonymous Batch Verification Scheme for Mobile Healthcare Crowd Sensing". IEEE Internet of Things Journal. 2019; 6(1):1287–1290. https://doi.org/10.1109/JIOT.2018.2862381

29. Gayathri N, Thumbur G, Kumar PR, Rahman MZU, Reddy PV. Efficient and secure pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks. IEEE Internet of Things Journal. 2019; 6(5):9064–9075. https://doi.org/10.1109/JIOT.2019.2927089

30. Liu J, Wang L, Yu Y. Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks. IEEE Internet of Things Journal. 2020; 7(6):5256–5266. https://doi.org/10.1109/JIOT.2020.2979613

31. Zhan Y, Wang B, Lu R. Cryptanalysis and improvement of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks. IEEE Internet of Things Journal. 2021; 8(7):5973–5984. https://doi.org/10.1109/JIOT.2020.3033337

32. Pointcheval D, Stern J. Security proofs for signature schemes. International conference on the theory and applications of cryptographic techniques. 1996: 387-398.

33. Verma GK, Kumar N, Gope P, Singh B, Singh H. SCBS: A Short Certificate-Based Signature Scheme With Efficient Aggregation for Industrial-Internet-of-Things Environment. IEEE Internet of Things Journal. 2021; 8(11):9305–9316. https://doi.org/10.1109/JIOT.2021.3055843

34. Shen L, Ma J, Liu X, Wei F, Miao M. A secure and efficient ID-based aggregate signature scheme for wireless sensor networks. IEEE Internet of Things Journal. 2016; 4(2):546–54. https://doi.org/10.1109/JIOT.2016.2557487