PFCBAS: Pairing Free and Provable Certificate-Based Aggregate Signature Scheme for e-Healthcare Monitoring System

Girraj Kumar Verma, B. B. Singh, Neeraj Kumar, *Senior Member, IEEE*, Omprakash Kaiwartya, Mohammad S. Obaidat, *Fellow, IEEE and Fellow SCS*

***Abstract*—Recently, one of the most popular technologies of the modern era, the Internet of Things (IoT) allows the deployment and usage of various real-time test beds for various smart applications. One such application is the e-healthcare in which patients healthcare related data is transmitted to the nearest base station and then to local or remote server as per the requirements. The data related to patients health is sensitive and needs special protection. Therefore, the integrity and authentication of sources of data generation are vital issues. However, several authentication or signature schemes which have been introduced in the past for this purpose are ID-based or having certificate-less settings. Thus, these proposed schemes suffer from key escrow and key distribution problems. To mitigate these issues, this article presents a certificate-based pairing free aggregate signature scheme. The proposed scheme uses the merits of public key cryptography (PKC) and identity based public key cryptography (IDBPKC). The scheme is proven to be unforgeable, assuming the hardness of ECDLP. The performance analysis shows that proposed CBPFAS scheme executes in $0.78(n+1)$ $\mathbf{m}_{\blacktriangle}$ in comparison to $9.63+1.17n$ $\mathbf{m}_{\blacktriangle}$ in [1], $9.63+0.78n$ $\mathbf{m}_{\blacktriangle}$ in [2], $9.63+3.39n$ $\mathbf{m}_{\blacktriangle}$ in [3] and $9.63+1.17n$ $\mathbf{m}_{\blacktriangle}$ in [4].**

***Index Terms*—Authentication, Aggregate Signature, Certificate-based Signature, Key Escrow, e-Healthcare.**

I. Introduction

Recently, the deployment of IoT devices has escalated exponentially in various applications (Figure-1). Such as- mobile computing, e-healthcare, Industrial IoT, among others. However, devices used in IoT adopt different technologies, processing architectures, design methods and communication architecture. Among all these applications, e-Healthcare monitoring systems (e-HMS) of a remote location patient is one of the important applications. In e-HMS, sensors collect data from patient and send to the main location server (hospital) which may be at local or global sites. As the data of patient is highly sensitive, breach of the sensitive information may cause serious problems (such as casualty of patient)[5]. Being IoT based, the architecture of e-HMS is heterogeneous with respect to type of devices, technology and applications. This heterogeneity increases the threats to security and privacy leakage. The recent WHO reports [6] on the security and privacy of e-healthcare data shows that 55% of members countries (77% Europian and 67% Americas) have passed security and privacy legislation to protect the the patient's data.

In e-HMS, the monitoring of the patients was done by consulting doctors with desktop conference systems. However, it was applicable for first-aid only. The development of sensors revolutionized the e-HMS with inclusion of Body Sensor Networks (BSNs). In BSNs, sensors were attached to patient's body (either implanted or wearable) to collect data such as ECG, EOG, EEG, pH-value, pulse rate, blood pressure, etc. Then, it is communicated to the hospital server [7]. However, a slight changes in the values may cause severe effects on the health of the patients. Due to the patient's life involvement, the security challenges are very important. To solve these issues, several projects such as Ubimon [8], MobiHealth [9], HealthGear [10] and CodeBlue [11] were funded by different agencies across the globe.

The important security threats to BSNs are data modification, impersonation, Eavesdropping and Replaying [12]. The public key cryptography (PKC) [13], [12] can be a solution to mitigate such threats. These solutions consist of encryption, signing, key exchange, authentication, etc. Traditionally, in PKC, a user selects its public/private keys pair. Then, certification on public key is obtained from a certification authority (CA). However, the verification of certificate creates the third party queries problem. To complete it, a huge infrastructure is required. To simplify management of certificates in traditional PKC, Shamir [14] pioneered the idea of identity-based PKC (IDBPKC). In IDBPKC, a trusted authority (TA) generates every user's secrete key. The corresponding public key can be computed from a unique ID information such as IP-address of the system, e-mail, social security number, etc. After private key generation, TA sends private key to a legitimate user in a secure way. Here, TA knows the user's private key. Thus, TA is capable of signing or decrypting the illegal documents. This is called key escrow problem in IDBPKC. The private key distribution to a legitimate user is also a serious problem.

To eliminate inborn key escrow, Al-Riyami and Paterson [15] devised the concept of certificate-less PKC (CLPKC). In CLPKC, TA generates a partial secret key of user. TA

G. K. Verma is with Department of Mathematics, Hindustan College of Science and Technology,Farah,Mathura, India (Email:girrajv@gmail.com)

B. B. Singh is with Department of Computer Science, Govt. K. R. G. (PG) College, Gwalior, India (Email: bbsinghkrg@gmail.com)

N. Kumar is with Department of Computer Science and Engineering, Thapar University, Patiala, India (Email: neeraj.kumar@thapar.edu)

O.Kaiwartya is with Nottingham Trent Unviveristy, Nottingham (U.K.) email: omprakash.kaiwartya@ntu.ac.uk

M.S. Obaidat is with the ECE Departmnet, Nazarbayev Univeristy, Astana, Kazakhstan, King Abdullah School of Information Technology, University of Jordan, Jordan, and University of Science and Technology Beijing, China. (e-mail: msobaidat@gmail.com)

communicates this key to user via a secure channel. After receiving this, user randomly picks secret information and computes its full private key. Then, user displays the public key correspondingly. So, TA is incapable to know the full secret key. Thus, CLPKC is free from key escrow. In CLPKC, private key distribution problem is still inbuilt. Besides this, in CLPKC public key is not certified, so it suffers from "public key replacement" attack as well [16].

Parallel to CLPKC, Gentry proposed the concept of certificate-based PKC (CBPKC) to solve inborn key escrow. However, the merits of traditional PKC and IDBPKC are preserved. In this notion, user creates its key pair (private/public) as it does in PKC and then obtains a certification on identity (ID) and public key from CA. In CBPKC, certificates are implicitly used as a decryption (or signing) key and thus, each time an updated certificate is obtained by its owner. Therefore, CBPKC eliminates the secret key distribution problem and key escrow of IDBPKC. Based on the updating of certificates, CBPKC also solves the problems associated with certificate revocation [17].

A. Roadmap of the article

Following is the roadmap of the article. Section II describes the related literature review. In Section III, motivation to propose the scheme is described. In section IV, basics on elliptic curves over finite fields are given. This section also addresses the formal syntax of the proposed PFCBAS scheme, system architecture of e-HMS and security considerations. Section V consists of the detailed steps of the proposed PFCBAS scheme and Section VI presents the security proof discussion. In section VII, a detailed performance analysis discussion is described. Finally, in Section VIII, the article is concluded.

II. Related work

In 2004, Kang *et al.* devised the first certificate-based signature (CBS) to import the merits of CBPKC and IDBPKC in digital signature technology [18]. Their construction includes the short signing for certification, and ID-based signing to sign the document. In EuroPKI 2007, Li *et al.* coined the "Key Replacement Attack" in the CBPKC, and presented the refinement of existing security models [19]. They presented the cryptanalysis of [18] with respect to this attack. Then, an improved and EUF-ACMA CBS scheme under CDH-assumption was proposed by them. The signature length of their scheme was shorter than proposed in [18]. To optimize the computational cost, in 2008, Liu *et al.* [20] introduced the first two pairing free CBS schemes. Their first scheme was the most efficient among existing CBS schemes and the second was based on standard model and so was strongly secure. However, Zhang successfully performed the cryptanalysis of the scheme [20] and two attacks were executed on this scheme [21]. After this, Zhang presented an improved CBS scheme as well. To overcome the "Key Replacement Attack", Li *et al.* introduced two CBS schemes. Their first scheme was secure in ROM and second scheme was in standard model [22]. To reduce the signature length, Li *et al.* firstly introduced the construction of a short CBS scheme in ROM [23]. Their signature consists of only one element from elliptic curve group and consumes two pairing operations (one during signing and one for verification). Thus, it was the shortest and efficient CBS scheme from pairings. In 2012, Zhou *et al.* proposed a short and efficient CBS [24] scheme to deploy in wireless cooperative networks and their scheme also needs one group element as a signature. However unfortunately Cheng *et al.* introduced a universal forgery attack on the scheme by Zhou *et al.* [25]. In 2013, Li, Wang and Zhang [26] successfully executed the cryptanalysis on Min and Wang's scheme [27] and proposed a new provably secure pairing free CBS scheme from discrete logarithm (DL) problem. In 2016, Zhou and Cui introduced a CBS [28] scheme secure against the malicious-but-passive certifier attack. In 2017, Verma *et al.* proposed a short CBS scheme to deploy on WSNs [29]. Their scheme is the shortest pairing based construction. Recently, Verma *et al.* devised the first CB-proxy blind signature from pairings [30]. The scheme is secure in ROM.

Parallel to Gentry [17], Boneh *et al.* coined the notion of aggregation of digital signatures [31]. In this aggregation method, m signatures on m documents from m signers can be compressed to make a single short signature and the verifier is convinced to the fact that indeed m signers made a signature on corresponding documents. Due to compression method, this aggregate signature (AS) reduces the total band-width required to transmit the m signatures. This technique also reduces the total computation cost of verification process. Since Boneh *et al.* [31], several AS schemes in IDBPKC or CLPKC [3], [2], [4], [1] are devised by the research community. In certificate-based setting, Baek *et al.* [32] introduced the first sequential aggregate signature scheme to short the total bandwidth of m signatures on m messages by m signers. Their scheme is provably secure in ROM over the M-LRSW and DH-inversion assumptions. However, the construction is based on pairings. In 2016, Chen *et al.* [33] introduced the first pairing free CB-AS (PFCBAS) scheme to improve the efficiency. Recently, Ma *et al.* [34] presented a new and efficient CB-AS scheme over pairings. However, their scheme uses aggregation of different signatures on the same document by different signers. Therefore, the applications are restricted and thus, not worthy enough.

III. Motivation and Contribution

The emerging deployment of WSNs to e-HMS is a common practice. Being IoT based, e-HMS is a heterogeneous architecture. Due to the heterogenity, the risk of authentication and integrity of data has magnified. The foundation of e-HMS on WSNs also increases the need of efficient technologies. To provide integrity and authentication, several digital signature schemes have been devised in literature [1], [35], [2], [3], [4], [36], [37], [38]. However, there are some problems with these schemes. First, being ID-based or CLPKC-based construction, the schemes suffer from inborn key escrow and secrete key distribution obstacles. Second, most of these schemes are pairing based construction. Since, pairing is the

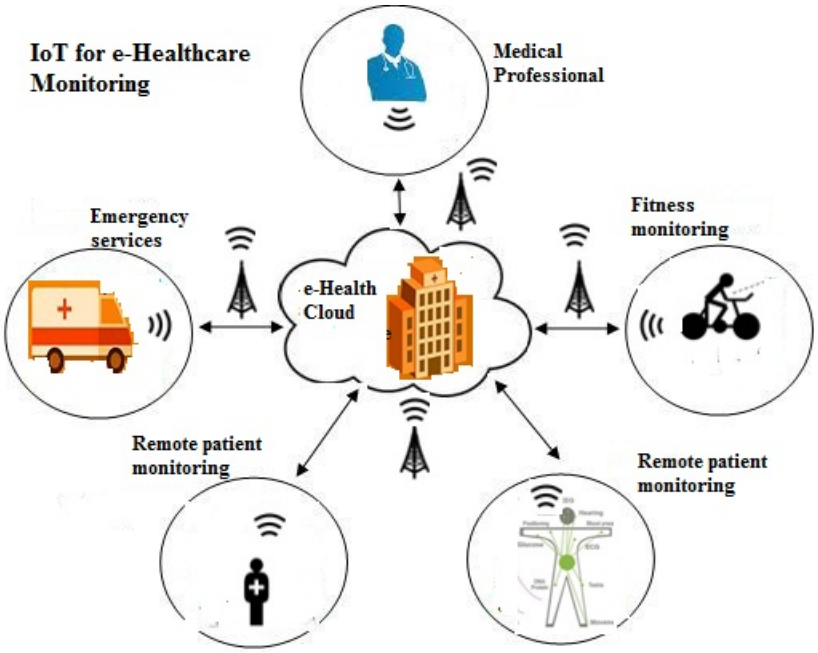

Fig. 1. IoT e-Healthcare Applications Scenario

most expensive operation and therefore, in order to reduce the bandwidth and computation cost, Zhou *et al.* proposed the first PFCBAS scheme [33]. They proved the security in ROM under elliptic curve DL assumption. By analyzing this scheme, we found that it does not satisfy the most important attribute "*Correctness*" of verification phase. After analysis, we found that the signature is not correctly generated. Thus, it is not the first PFCBAS scheme based on ECDLP. By proposing a new PFCBAS scheme, we try to solve the above problems. Due to pairing free construction, the proposed PFCBAS scheme is more efficient. Being CBS-based scheme, it solves key escrow and key distribution problems as well.

The detailed contributions of the article are as follows:

- A certificate-based aggregate signature without pairing is devised and is called PFCBAS.
- The proposed PFCBAS scheme is secure in ROM under ECDLP assumption.
- By performance analysis, we show that the proposed PFCBAS scheme is efficient in comparison to the existing competing related schemes.

IV. Preliminaries

This section describes basics on elliptic curves and ECDLP in finite fields. The system architecture of e-HMS and syntax of proposed PFCBAS scheme are also described. The definitions of security architecture, adversarial oracles and games towards EUF-ACMA in ROM are also presented.

A. Basics on Elliptic Curves:

Let, F_q be a finite field with prime q and E/F_q be an elliptic curve over this finite field. Then, E/F_q is defined as the set of all points (x, y) such that $y^2 = x^3 + ax + b$, where a and b are constants from F_q and satisfy $4a^3 + 27b^2 \neq 0$. We consider an additive cyclic subgroup G_T of E/F_q with P as generator, containing all pairs (x, y) of affine co-ordinates on E/F_q and a special point $\mathcal{O}$ at infinity. Since, under addition, G_T is a group and so for $s \in \mathbb{Z}_q^*$, sP can be computed as $P+P+P+.....+P$ (s times) and is called scalar multiplication.

B. e-HMS Architecture

In e-HMS, sensors are deployed in the body of a patient. Then, these sensors collect data like body temperature, pulse rate, pH-value, blood pressure, etc. (Figure-2). To communicate the collected data to a medical professional is a sensitive issue. To solve this issue, the following architecture is considered in which there are sensors, local center (district center), main hospital (health center) server and trained medical professionals [35].

- The sensors are assumed to be deployed to patient's body as implanted or wearable devices. The signing algorithm and required system parameters are programmed on the sensor. Thus, sensor first collects the required data and then signs on it with its secret key. This data is sent to local center (district center).
- District center works as aggregator. It verifies the received message signature pairs. Then, it classifies the messages based on the professionals needed and it runs aggregation phase. After execution, it sends the batch of messages along-with aggregated signature to the hospital server.
- Hospital server passes the received batch to a medical professional. However, a copy of this is preserved in this server. This preserved copy may be used to analyze the data.
- Lastly, a medical professional receives the data and aggregated signature. Then, it verifies the aggregated signature

to check the authenticity of data as well as sender (i.e. sensors or patient). Professionals have smart devices with limited resources. Therefore, the aggregation and verification process must be efficient. After satisfaction, professional will be able to define directives.

In this architecture, sensors play the role of signer. Therefore, signing process must be efficient. Medical professional runs the aggregate verify phase. So, this phase should also be efficient enough. Since, the aggregate verify is more efficient than verifying separately and thus, efficiency of this step is achieved by aggregation.

C. Syntax of proposed PFCBAS Scheme

The proposed PFCBAS scheme is a 7-tuple $(CBSetup, CBUKeyGen, CBUCertGen, CBSignGen, CBSignVer, CBSignAgg, CBSignAggVer)$ of Polynomial bounded (PPT) algorithms perform the functions shown below:

1) *CBSetup:* After input security parameter (1^l), this algorithm returns system parameter $CBParams$ and master private key of CA.
2) *CBUKeyGen:* During execution, user randomly selects a number and outputs its (public, private) key pair by itself. Then, he/she keeps private key with itself and displays public key.
3) *CBUCertGen:* User sends her/his ID and public key to CA and then CA certifies these by using master private key. CA then sends the certificate through a public channel.
4) *CBSignGen:* During this stage, signer generates a signature on a document by using her/his secrete key along with certificate.
5) *CBSignVer:* The verifier (Cindy) verifies the validity of signature on a document by taking public key and ID of signer and master public key of CA. If it is valid, then output is accepted, otherwise, it is rejected.
6) *CBSignAgg:* During this stage, the aggregator obtains the m signatures on m documents from m signers and then combines them to create a compressed signature. The output is aggregated signature on m documents.
7) *CBSignAggVer:* During this stage, the medical professional receives the aggregated short signature on m documents and takes the public keys, IDs and system parameters $CBParams$ as input. The output is Accept or Reject, based on the validity.

D. Threat Model

The security framework of PFCBAS scheme combines the provable security of underlying PFCBS scheme along with the security of aggregation. Therefore, the proposed PFCBAS scheme is secure (EUF-ACMA) in ROM if both of the schemes are EUF-ACMA. Based on capabilities, two different forgers $\mathcal{F}_1$ (Type-1) and $\mathcal{F}_2$ (Type-2) are considered and following are the detailed definitions of forgers $\mathcal{F}_1, \mathcal{F}_2$ and the oracles executed by them [26], [30], [29].

- Type-1($\mathcal{F}_1$ Forger): This forger plays the role of dishonest signer or has control over the signer. Thus, it is capable to perform public key replacement of a user. However, it knows nothing about corresponding certificate or private key.
- Type-2($\mathcal{F}_2$ Forger): This forger plays the role of malicious CA or controls the master private key of CA. However, it is incapable to replace the public key.

Following are the oracles requested by $\mathcal{F}_1$ and $\mathcal{F}_2$:

- **CBSetup(.)-Oracle:** The challenger ($\mathcal{CH}$) is responsible for running this oracle. The output is master private key/master public key of CA and system parameters $CBParams$.
- **CBUKeyGen(.)-Oracle:** Forger put forwards the ID to this oracle, to get the corresponding keys. By running this, $\mathcal{CH}$ obtains the keys and forwards the pk_{ID} (public key) to forger.
- **PublicKReplace(.)-Oracle:** A forger can replace pk_{ID} (public key) with user ID. However, no need to get the related private key. Forger can do this repeatedly.
- **Corruption(.)-Oracle:** Forger put forward the user ID to obtain the private key. Challenger checks the recorded output of CBUKeyGen(.)-Oracle and takes the tuple (ID, pk_{ID}, sk_{ID}) and then sends sk_{ID} to forger as a response.
- **CBUCertGen(.)-Oracle:** Forger requests certification on (ID, pk_{ID}) and $\mathcal{CH}$ runs CBUCertGen(.)-Oracle and sends the certificate obtained to forger.
- **CBSignGen(.)-Oracle:** Forger requests (ID, m) to get the signature and $\mathcal{CH}$ executes the oracle and obtains the signature. Then, $\mathcal{CH}$ sends the output to the forger and recorded the response.

E. EUF-ACMIA against $\mathcal{F}_1$ (Game-I)

Based on the power of $\mathcal{F}_1$, $\mathcal{F}_1$ is able to request **CBSetup(.)**, **CBUKeyGen(.)**, **CBUCertGen(.)**, **PublicKReplace(.)**, **Corruption(.)** and **CBSignGen(.)** oracles, respectively. Then, forger obtains the corresponding outputs.

Output: At last, $\mathcal{F}_1$ outputs a forged signature $(ID^*, \sigma^*, m^*, pk^*_{ID})$, such that:

- $\mathcal{F}_1$ has never requested ID^* to **Corruption(.)-Oracle**.
- $\mathcal{F}_1$ has never requested (ID^*, pk^*_{ID}) to **CBUCertGen(.)-Oracle**.
- (m^*, σ^*) is a valid forged message and signature pair, while $\mathcal{F}_1$ has never requested (m^*, ID^*) to **CBSignGen(.)-Oracle**.

The probability of winning the Game-I is the success probability of $\mathcal{F}_1$ and is denoted by $Succ^{EUF-ACMIA}_{\mathcal{F}_1, PFCBAS}$.

Definition.1: Our proposed PFCBAS scheme is EUF-ACMIA secure in the ROM against adversary $\mathcal{F}_1$, if the probability $Succ^{EUF-ACMIA}_{\mathcal{F}_1, PFCBAS}$ is negligible in the above Game-I.

F. EUF-ACMIA against $\mathcal{F}_2$ (Game-II)

Since, forger $\mathcal{F}_2$ (Type-II) controls CA, so $\mathcal{F}_2$ can request to **CBSetup(.)**, **CBUKeyGen(.)**, **Corruption(.)** and **CBSignGen(.)** oracles, respectively. Then, forger can obtain

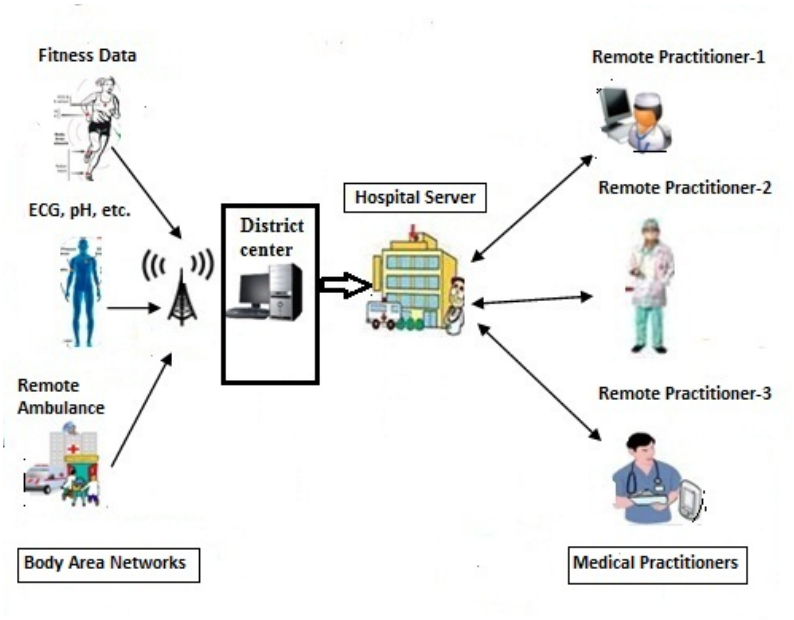

Fig. 2. Proposed Architecture of e-HMS

the corresponding outputs.

Forgery: At last, $\mathcal{F}_2$ outputs a forged signature $(ID^*, \sigma^*, m^*, pk^*_{ID})$, such that:

- $\mathcal{F}_2$ has never requested ID^* towards **Corruption(.)-Oracle**.
- The message signature pair (m^*, σ^*) passes verification correctly, while $\mathcal{F}_2$ has never requested (m^*, ID^*) towards **CBSignGen(.)-Oracle**.

The probability to win the above Game-II is the success probability of $\mathcal{F}_2$ and is denoted by $Succ^{EUF-ACMIA}_{\mathcal{F}_2,PFCBAS}$.

Definition.2: Our introduced PFCBAS scheme is EUF-ACMIA secure in the ROM against adversary $\mathcal{F}_2$, if the probability $Succ^{EUF-ACMIA}_{\mathcal{F}_2,PFCBAS}$ is negligible in the above Game-II.

V. Proposed PFCBAS Scheme

The detailed steps of PFCBAS scheme are as follows [26], [30] (Figure-3).

1) CBSetup: By inputting 1^μ (security parameter), this PPT algorithm outputs system parameter $CBParams = (E/F_q, G_T, P, q, pk_{CA}, H_0, H_1, H_2)$ and master private key sk_{CA} of CA as follows:
   - Here G_T be the cyclic group of points of E/F_q as defined in Section-2.
   - It randomly picks $s \in \mathbb{Z}^*_q$ and $P \in G_T$, then $pk_{CA} = sP$ is certifier's public key and $sk_{CA} = s$ is certifier's private key.
   - $H_0 : \{0,1\}^* \times G_T \times G_T \rightarrow \mathbb{Z}^*_q, H_1 : \{0,1\}^* \times G_T \times G_T \times G_T \rightarrow \mathbb{Z}^*_q$ and $H_2 : \{0,1\}^* \times G_T \times G_T \times G_T \times G_T \rightarrow \mathbb{Z}^*_q$ are three cryptographic hash functions.
2) CBUKeyGen: Every user U_i (with ID_i) randomly selects $x_{ID_i} \in \mathbb{Z}^*_q$ as his/her private key then computes $pk_{ID_i} = x_{ID_i}P$ as public key.
3) CBUCertGen: On input $CBParams$, ID_i and public key pk_{ID_i} from user U_i, CA randomly chooses $r_i \in \mathbb{Z}^*_q$ and computes $R_i = r_iP$ and $t_i = r_i + sH_0(ID_i, pk_{ID_i}, R_i)$ and then outputs $Cert_{ID_i} = (R_i, t_i)$. The user can verify the certificate by $t_iP = R_i + H_0(ID_i, pk_{ID_i}, R_i)pk_{CA}$.
4) CBSignGen: Signer (with ID_i) generates a signature $\sigma_i = (R_i, U_i, Z_i)$on message $m_i \in \{0,1\}^*$ by computing the following:
   - It randomly selects $k_i \in \mathbb{Z}^*_q$ and computes $U_i = k_iP$.
   - It will compute $h_{1i} = H_1(m_i, pk_{ID_i}, U_i, R_i)$ and $h_{2i} = H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)$.
   - Then it will find out: $Z_i = t_i + x_{ID_i}h_{1i} + k_ih_{2i}$
5) CBSignVer: The aggregator outputs accept or reject message by checking the equality: $Z_iP = R_i + H_0(ID_i, pk_{ID_i}, R_i)pk_{CA} + H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i$.
6) CBSignAggregate: The aggregator (node) receives $\{(m_i, \sigma_i = (R_i, U_i, Z_i)); 1 \leq i \leq n\}$ and computes $Z = \sum_{i=1}^{n} Z_i$. Then, it will display Z as aggregated signature on $(m_1, m_2, m_3.....m_n)$.
7) CBSignAggVer: The medical professional accepts the aggregate signature $(Z, R_1, R_2, R_3.....R_n, U_1, U_2, U_3, ...U_n)$ on messages $(m_1, m_2, m_3.....m_n)$ iff $ZP = \sum_{i=1}^{n}(R_i) + (\sum_{i=1}^{n} H_0(ID_i, pk_{ID_i}, R_i))pk_{CA} +$

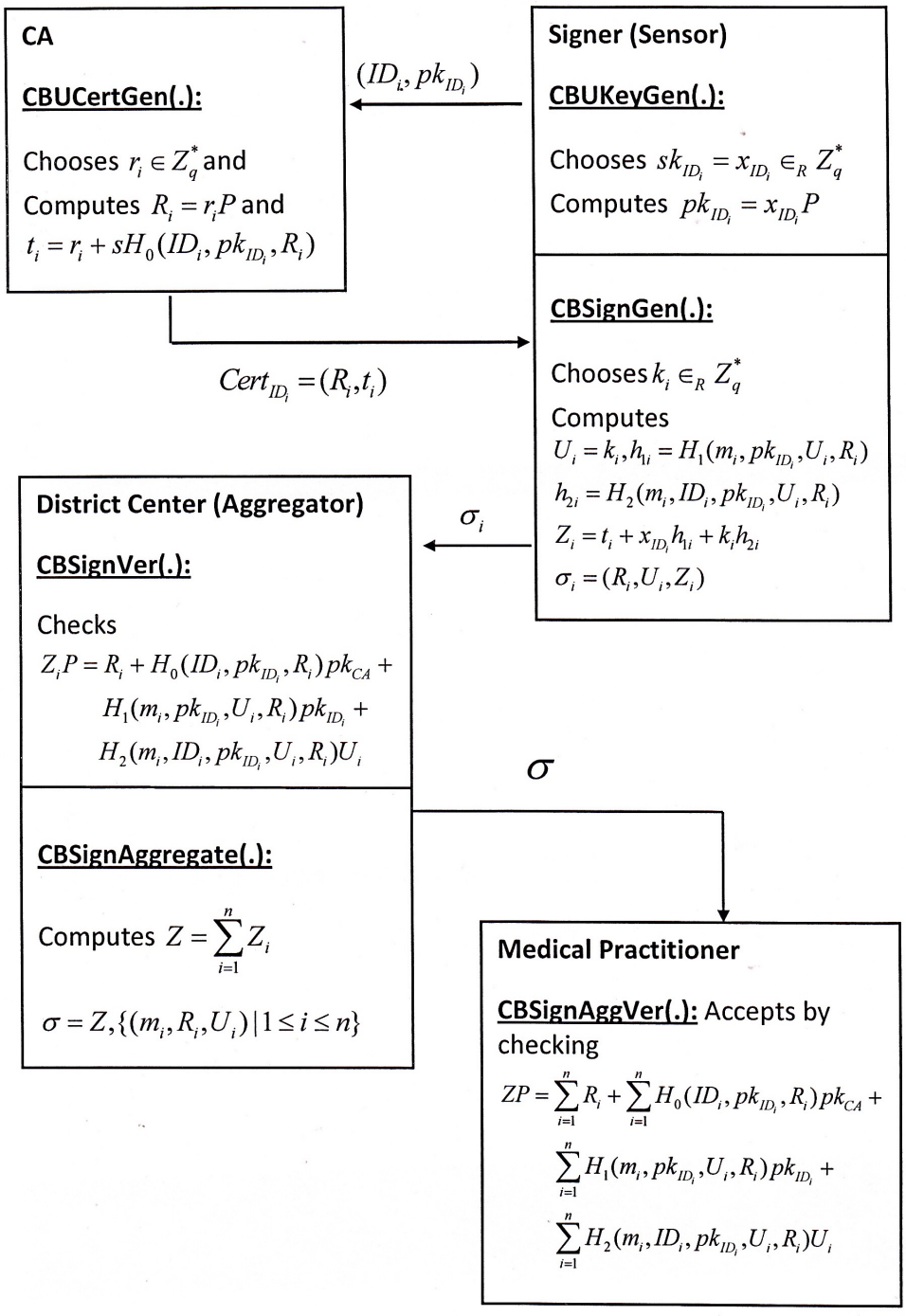

Fig. 3. Proposed PFCBAS Scheme

$$\sum_{i=1}^{n} H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + \sum_{i=1}^{n} H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i.$$

VI. Analysis of the proposed PFCBAS

This section presents the provable security discussions of PFCBAS. The theorems to prove EUF-ACMA in ROM and correctness are described here. The performance discussion is also presented in this section. To explain performance analysis, the computational cost and energy consumption are analyzed.

A. Correctness

Theorem 1: The introduced PFCBS and PFCBAS satisfy correctness.

Proof: Since, $Z_iP = (t_i + x_{ID_i}h_{1i} + k_ih_{2i})P = r_i + sH_0(ID_i, pk_{ID_i}, R_i) + x_{ID_i}h_{1i} + k_ih_{2i})P = R_i + H_0(ID_i, pk_{ID_i}, R_i)pk_{CA} + H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i$ and similarly $ZP = \sum_{i=1}^{n}(R_i) + (\sum_{i=1}^{n} H_0(ID_i, pk_{ID_i}, R_i))pk_{CA} + \sum_{i=1}^{n} H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + \sum_{i=1}^{n} H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i$. Thus, verification can be done correctly.

B. Proof of Security

The security of a digital signature scheme is presented by existential unforgeability under adaptively chosen message and ID attack in random oracle model. This section contains the discussion on provable security of introduced PFCBAS scheme. The discussion is defined with respect to forgers, defined in the Section 2.

Game-I(EUF-ACMIA against $\mathcal{F}_1$)

To prove the EUF-ACMA in ROM of PFCBAS, we prove the following lemma.

Lemma 1: In the ROM, forger $\mathcal{F}_1$ (type-1) can (t, q, ϵ) break the introduced PFCBS. The success probability of forging the signature is $Succ^{EUF-CMA}_{\mathcal{F}_1,PFCBS} \geq \epsilon$. Then, a polynomial algorithm $\mathcal{B}$ that solves a random instance of ECDLP with success probability $Succ^{ECDLP}_{\mathcal{B},G_1} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$, where q be the maximum of all requests made by $\mathcal{F}_1$ can be created.

Proof: To construct the proof, the challenge is $(P, Q = \alpha P) \in G_T \times G_T$, a random instance of ECDLP to be taken by $\mathcal{B}$ as input. $\mathcal{B}$ maintains the records of all responses in the lists denoted as $List_{uk}, List_{UH_0}, List_{UH_1}, List_{UH_2}, List_{UC}$ and $List_{PFS}$. All the lists are taken initially empty. In the $List_{uk}$, the responses of **CBUKeyGen(.)-Oracle** are stored. $List_{UH_0}, List_{UH_1}$, and $List_{UH_2}$ record the responses from $H_0(.), H_1(.)$ and $H_2(.)$ oracles, respectively. The responses from **CBUCertGen(.)-Oracle** are contained in $List_{UC}$. $List_{PFS}$ contains the responses from **CBSignGen(.)-Oracle**.

- CBSetup(.)-Oracle: To solve ECDLP, $\mathcal{B}$ firstly sets $Q = pk_{CA}$ (master public key of CA) and gives the $CBParams = (E/F_q, G_T, P, pk_{CA}, H_0, H_1, q, H_2)$ to $\mathcal{F}_1$. Then, $\mathcal{B}$ randomly selects an index k satisfying $1 \leq k \leq q_{H_0}$, where q_{H_0} is the number of requests made by $\mathcal{F}_1$ to random hash oracle $H_0(.)$. Now, $\mathcal{B}$ executes the following experiment to output the solution of ECDLP:
- CBUKeyGen(.)-Oracle: $\mathcal{F}_1$ requests the key generation queries on ID_i, with $1 \leq i \leq q_{key}$. $\mathcal{B}$ stores the responses in $List_{UK}$. To respond, $\mathcal{B}$ scans the $List_{UK}$ to check the entry $(ID_i, pk_{ID_i}, x_{ID_i})$. If the entry is found, reply pk_{ID_i} to $\mathcal{F}_1$. Otherwise, randomly selects $x_{ID_i} \in \mathbb{Z}^*_q$, computes $pk_{ID_i} = x_{ID_i}P$, updates $List_{UK}$ by adding $(ID_i, pk_{ID_i}, x_{ID_i})$ and sends the response to $\mathcal{F}_1$.
- $H_0(.) - Queries$: On queried pair (ID, PK_{ID}) to $H_0(.) - Oracle$, $\mathcal{B}$ scans the $List_{UH_0}$ to check the existence of $(ID_i, pk_{ID_i}, R_i, d_{0i})$. If the tuple is found, the entry is picked. Otherwise, $\mathcal{B}$ randomly selects $d_{0i} \in \mathbb{Z}^*_q$ and outputs $d_{0i} = H_0(ID_i, pk_{ID_i}, R_i)$ as hashed value and updates $List_{UH_0}$ by adding $(ID_i, pk_{ID_i}, R_i, d_{0i})$.
- $H_1(.) - Queries$: On requested tuple $(m_i, pk_{ID_i}, R_i, U_i)$ to $H_1(.) - Oracle$, $\mathcal{B}$ scans the $List_{UH_1}$ to check the existence of $(m_i, pk_{ID_i}, R_i, U_i, d_{1i})$. If the tuple is there, the latter will pick it. Otherwise, $\mathcal{B}$ randomly picks $d_{1i} \in \mathbb{Z}^*_q$ and sets $d_{1i} = H_1(m_i, pk_{ID_i}, R_i, U_i)$. Then, $\mathcal{F}_1$ receives d_{1i} as response. The $List_{UH_1}$ is updated by adding $(m_i, pk_{ID_i}, R_i, U_i, d_{1i})$.
- $H_2(.)$-Queries: On requested tuple $(m_i, ID_i, pk_{ID_i}, R_i, U_i)$, $\mathcal{B}$ scans the $List_{UH_2}$ to check the existence of $(m_i, ID_i, pk_{ID_i}, R_i, U_i, d_{2i})$. If the tuple is found, the latter picks d_{2i} from it. Otherwise, it picks a $d_{2i} \in_R \mathbb{Z}^*_q$ and sets $d_{2i} = H_2((m_i, ID_i, pk_{ID_i}, R_i, U_i))$. Lastly, $\mathcal{B}$ sends d_{2i} to forger and updates the $List_{UH_2}$ by adding $(m_i, ID_i, pk_{ID_i}, R_i, U_i, d_{2i})$.
- Public-Key-Replace(.)-Oracle: On requested pair (ID_i, pk'_{ID_i}), $\mathcal{B}$ searches the $List_{UK}$ to check the existence of $(ID_i, pk_{ID_i}, x_{ID_i})$. Then it will replace the tuple with $(ID_i, pk'_{ID_i}, \perp)$.
- Corruption(.)-Oracle: On a requested ID, $\mathcal{B}$ scans the $List_{UK}$ to check the existence of corresponding tuple (ID, pk_{ID}, x_{ID}). If tuple is found, then the response x_{ID} is sent to $\mathcal{F}_1$. Otherwise, $\mathcal{B}$ picks $x_{ID} \in \mathbb{Z}^*_q$ in a random manner and sets $pk_{ID} = x_{ID}P$. Then it will update the $List_{UK}$ and sends x_{ID} to $\mathcal{F}_1$.
- CBUCertGen(.)-Oracle: On requested pair (pk_{ID_i}, ID_i), $\mathcal{B}$ responds as follows:
  1) If $i \neq k$, $\mathcal{B}$ scans the $List_{UC}$ to check the existence of corresponding $Cert_{ID_i} = (R_i, t_i)$. If the entry is found, $\mathcal{B}$ responds it to $\mathcal{F}_1$. Otherwise, it will randomly pick $t_i, d_i \in \mathbb{Z}^*_q$ and computes $R_i = t_iP - d_iQ$. Then, $\mathcal{B}$ will scan the $List_{UH_0}$ to check that (ID_i, pk_{ID_i}, R_i) has already been defined. If yes, $\mathcal{B}$ re-chooses $t_i, d_i \in \mathbb{Z}^*_q$ until there is no collision. Next, $\mathcal{B}$ updates the $List_{UH_0}$ and $List_{UC}$ and outputs $Cert_{ID_i} = (R_i, t_i)$ to $\mathcal{F}_1$.
  2) If $i = k$, $\mathcal{B}$ stops and reports failure.
- CBSignGen(.)-Oracle On a requested (m_i, ID_i), $\mathcal{B}$ requests $CBUKeyGen(.) - Oracle$ and gets pk_{ID_i} and x_{ID_i}. If $x_{ID_i} = \perp$, $\mathcal{F}_1$ is required to reply the cor-

responding private key x_{ID_i}. Otherwise, $\mathcal{B}$ does the followings:

1) If $i \neq k$, $\mathcal{B}$ runs $CBUCertGen(.) - Oracle$ and creates a signature on (m_i, ID_i) with $(Cert_{ID_i}, x_{ID_i})$.
2) If $i = k$, $\mathcal{B}$ randomly picks $z_k, d_{0k}, d_{2k}, d_{1k} \in \mathbb{Z}_q^*$ and computes $R_k = t_k P - d_{0k} Q$ and $U_k = ((d_{2k} - t_k)P - d_{1k} pk_{ID_k})\lambda_k^{-1}$. Then, $\mathcal{B}$ sets $H_0(ID_k, pk_{ID_k}, w_k) = d_{0k}$, $H_1(m_k, pk_{ID_k}, U_k, w_k) = d_{1k}$, $H_2(m_k, ID_k, pk_{ID_k}, U_k, w_k) = d_{2k}$. If the hash values collide, it will re-choose the values and compute again. Then it will update the $List_{UH_0}, List_{UH_1}$, and $List_{UH_2}$ and returns (U_k, R_k, z_k) as signature to $\mathcal{F}_1$.

Eventually, $\mathcal{F}_1$ outputs a forged signature $\sigma^* = (U^*, R^*, Z^*)$ on message m^*.
If $ID^* \neq ID_k$, $\mathcal{B}$ aborts and reports failure. Otherwise, by Forking lemma [39], $\mathcal{B}$ outputs two signatures (R_1^*, U_1^*, Z_1^*) and (R_1^*, U_1^*, Z_2^*) on a same message m^* with the same random tuple but different hashed values H_0. Thus, $Z_1^* P = R_1^* + h_{01}^* pk_{CA} + h_1^* pk_{ID^*} + h_2^* U_1^*$ and $Z_2^* P = R_1^* + h_{02}^* pk_{CA} + h_1^* pk_{ID^*} + h_2^* U_1^*$. Thus, $\mathcal{B}$ computes $\alpha = \frac{Z_1^* - Z_2^*}{h_{01}^* - h_{02}^*}$ as solution to ECDLP.
Based on the simulation analysis, the success probability of solving ECDLP depends on the following events:
E_1: $\mathcal{B}$ does not abort during simulation.
E_2: $\mathcal{F}_1$ is able to create a valid forgery.
E_3: The forgery is done with respect to targeted identity.
Thus, $P[E_1] \geq \left(1 - \frac{1}{q}\right)^q, P[E_2|E_1] = \epsilon$ and $P[E_3|E_1 \wedge E_2] = \frac{1}{q}$. Therefore, $Succ_{\mathcal{B},G_1}^{(ECDLP)} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$. Where, q is the maximum number of all queries made by forger $\mathcal{F}_1$.

Theorem 2: In the ROM, forger $\mathcal{F}_1$ (type-1) can (t, q, ϵ) break the introduced PFCBAS. The success probability of forging the signature is $Succ_{\mathcal{F}_1,PFCBAS}^{EUF-CMA} \geq \epsilon$. Then, a polynomial algorithm $\mathcal{B}$ can be created, which solves a random instance of ECDLP with success probability $Succ_{\mathcal{B},G_1}^{ECDLP} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$, where q is the maximum of all requests made by $\mathcal{F}_1$.

Proof: To construct the proof, the challenge is $(P, Q = \alpha P) \in G_T \times G_T$; a random instance of ECDLP to be taken by $\mathcal{B}$ as input. The aim of $\mathcal{B}$ is to outputs α as solution.

$\mathcal{B}$ sets all the parameters as in Lemma 1 and sets $Q = pk_{CA}$. Then forger $\mathcal{F}_1$ requests all the oracles and obtains the responses as in Lemma 1.

Eventually, by Forking lemma [39] $\mathcal{B}$ outputs two aggregate signatures (R^*, U^*, Z^*) and (R^*, U^*, Z^{**}) on same message set $\{m_1^*, m_2^*, m_3^*, ...m_n^*\}$ under users $\{ID_1^*, ID_2^*, ID_3^*, ...ID_n^*\}$ with the same random tuple but different hashed values H_0. It is needed that for some $w \in \{1, 2, 3.....n\}$, ID_w^* is not requested to Corruption(.)-Oracle and (ID_w^*, m_w^*) is not queried to CBSignGen(.)-Oracle. Now, without loss of generality, we assume that $w = 1$. Thus, $Z^* = \sum_{i=2}^{n} Z_i^* + Z_1^*$ and $Z^{**} = \sum_{i=2}^{n} Z_i^{**} + Z_1^{**}$ where, $Z_1^* P = R_1^* + h_0^* pk_{CA} + h_1^* pk_{ID_w^*} + h_2^* U_1^*$ and $Z_1^{**} P = R_1^* + h_0^{**} pk_{CA} + h_1^* pk_{ID_w^*} + h_2^* U_1^*$. Thus, $\mathcal{B}$ computes $\alpha = \frac{Z_1^* - Z_1^{**}}{h_0^* - h_0^{**}}$ as solution to ECDLP.
The probability is same as in Lemma 1.

Game-II(EUF-ACMIA against $\mathcal{F}_2$)

To prove the EUF-ACMA in ROM of PFCBAS, we prove the following lemma.

Lemma 2: In the ROM, forger $\mathcal{F}_2$ (type-2) can (t, q, ϵ) forge the introduced PFCBS. The success probability of forging the signature is $Succ_{\mathcal{F}_2,PFCBS}^{EUF-CMA} \geq \epsilon$. Then, a polynomial algorithm $\mathcal{B}$ can be designed, which solves a random instance of ECDLP with success probability $Succ_{\mathcal{B},G_1}^{ECDLP} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$, where q is the maximum of all requests made by $\mathcal{F}_2$.

Proof: To construct the proof, the challenge is $(P, Q = \alpha P) \in G_T \times G_T$; a random instance of ECDLP to be taken by $\mathcal{B}$ as input. $\mathcal{B}$ maintains the records of all responses in the lists denoted as $List_{UK}, List_{UH_0}, List_{UH_1}, List_{UH_2}, List_{UC}$ and $List_{PFS}$. All the lists are taken initially empty. In the $List_{UK}$, the responses of CBUKeyGen(.)-Oracle are stored. $List_{UH_0}, List_{UH_1}$, and $List_{UH_2}$ record the responses from $H_0(.), H_1(.)$ and $H_2(.)$ oracles respectively. The responses from CBUCertGen(.)-Oracle, are contained in $List_{UC}$. $List_{PFS}$ contains the responses from CBSignGen(.)-Oracle.

- CBSetup(.)-Oracle: To solve ECDLP, we pick $s \in \mathbb{Z}_q^*$ randomly. Then, we set $pk_{CA} = sP$ (master public key of CA) and gives the $CBParams = (E/F_q, G_T, P, pk_{CA}, H_0, H_1, q, H_2)$ and s to $\mathcal{F}_2$. Then, $\mathcal{B}$ randomly selects an index k satisfying $1 \leq k \leq q_{H_0}$, where q_{H_0} is the number of requests made by $\mathcal{F}_2$ to random hash oracle $H_0(.)$. Now, $\mathcal{B}$ executes the following experiment to output the solution of ECDLP:
- CBUKeyGen(.)-Oracle: $\mathcal{F}_2$ requests the key generation queries on ID_i, with $1 \leq i \leq q_{key}$. $\mathcal{B}$ stores the responses in $List_{UK}$. To respond, $\mathcal{B}$ scans the $List_{UK}$ to check the entry $(ID_i, pk_{ID_i}, x_{ID_i})$. If the entry is found, a reply pk_{ID_i} is sent to $\mathcal{F}_2$. Otherwise, the following steps are made:
  - If $i \neq j$, then $\mathcal{B}$ randomly picks $x_{ID_i} \in \mathbb{Z}_q^*$, computes $pk_{ID_i} = x_{ID_i} P$, updates $List_{UK}$ by adding $(ID_i, pk_{ID_i}, x_{ID_i})$ and sends pk_{ID_i} to $\mathcal{F}_2$.
  - If $i = j$, then $\mathcal{B}$ sets $pk_{ID_i} = Q$. Next, $\mathcal{B}$ updates $List_{UK}$ by adding $(ID_i, pk_{ID_i}, \perp)$ and sends the pk_{ID_i} to $\mathcal{F}_2$
- $H_0(.) - Queries$: On queried pair (ID, PK_{ID}) to $H_0(.) - Oracle$, $\mathcal{B}$ scans the $List_{UH_0}$ to check the existence of $(ID_i, pk_{ID_i}, R_i, d_i)$. If the tuple is found, the entry is picked. Otherwise, $\mathcal{B}$ randomly selects $d_i \in \mathbb{Z}_q^*$

and outputs $d_i = H_0(ID_i, pk_{ID_i}, R_i)$ as hashed value and updates $List_{UH_0}$ by adding $(ID_i, pk_{ID_i}, R_i, d_i)$.

- $H_1(.) - Queries$: On requested tuple $(m_i, pk_{ID_i}, R_i, U_i)$ to $H_1(.) - Oracle$, $\mathcal{B}$ scans the $List_{UH_1}$ to check the existence of $(m_i, pk_{ID_i}, R_i, U_i, d_{1i})$. If the tuple is there, then it will pick it. Otherwise, $\mathcal{B}$ randomly picks $d_{1i} \in \mathbb{Z}_q^*$ and sets $d_{1i} = H_1(m_i, pk_{ID_i}, R_i, U_i)$. Then, $\mathcal{F}_2$ receives d_{1i} as response. The $List_{UH_1}$ is updated by adding $(m_i, pk_{ID_i}, R_i, U_i, d_{1i})$.
- $H_2(.)$-Queries: On requested tuple $(m_i, ID_i, pk_{ID_i}, R_i, U_i)$, $\mathcal{B}$ scans the $List_{UH_2}$ to check the existence of $(m_i, ID_i, pk_{ID_i}, R_i, U_i, d_{2i})$. If the tuple is found, then it will pick d_{2i} from it. Otherwise, it will pick a $d_{2i} \in_R \mathbb{Z}_q^*$ and sets $d_{2i} = H_2((m_i, ID_i, pk_{ID_i}, R_i, U_i))$. Lastly, $\mathcal{B}$ sends d_{2i} to forger and update the $List_{UH_2}$ by adding $(m_i, ID_i, pk_{ID_i}, R_i, U_i, d_{2i})$.
- Corruption(.)-Oracle: On a requested ID, $\mathcal{B}$ acts as follows:
  - If $i \neq j$, $\mathcal{B}$ scans the $List_{UK}$ to check the existence of corresponding tuple (ID, pk_{ID}, x_{ID}). If tuple is found, response will be sending x_{ID} to $\mathcal{F}_2$. Otherwise, $\mathcal{B}$ picks $x_{ID} \in \mathbb{Z}_q^*$ in a random manner and sets $pk_{ID} = x_{ID}P$. Then it will update the $List_{UK}$ and sends x_{ID} to $\mathcal{F}_2$.
  - If $i = j$, $\mathcal{B}$ respond $\perp$.
- CBSignGen(.)-Oracle On a requested (m_i, ID_i), $\mathcal{B}$ requests $CBUKeyGen(.) - Oracle$ and $Corruption(.) - Oracle$ and gets pk_{ID_i} and x_{ID_i}. Then, $\mathcal{B}$ performs the following steps:
  1) If $i \neq k$, $\mathcal{B}$ runs $CBUCertGen(.) - Oracle$ and creates a signature on (m_i, ID_i) with $(Cert_{ID_i}, x_{ID_i})$.
  2) If $i = k$, $\mathcal{B}$ randomly picks $t_k, d_k, z_k, e_k, \lambda_k \in \mathbb{Z}_q^*$ and computes $R_k = t_kP - d_kQ$ and $U_k = ((z_k - t_k)P - e_k pk_{ID_k})\lambda_k^{-1}$. Then, $\mathcal{B}$ sets $H_0(ID_k, pk_{ID_k}, w_k) = d_k$, $H_1(m_k, pk_{ID_k}, U_k, w_k) = e_k$, $H_2(m_k, ID_k, pk_{ID_k}, U_k, w_k) = \lambda_k$. If the hash values collide, it will re-choose the values and compute again. Next, it will update the $List_{UH_0}, List_{UH_1}$, and $List_{UH_2}$. Then, it will return (U_k, R_k, z_k) as signature to $\mathcal{F}_2$.

Eventually, $\mathcal{F}_2$ outputs a forged signature $\sigma^* = (U^*, R^*, Z^*)$ on message m^*.

If $ID^* \neq ID_k$, $\mathcal{B}$ aborts and reports failure. Otherwise, by Forking lemma [39], $\mathcal{B}$ outputs two signatures on a same message with the same random tuple but different hashed values $H_1(.)$. Thus, $Z_1^*P = R_1^* + h_0^* pk_{CA} + h_{11}^* pk_{ID^*} + h_2^* U_1^*$ and $Z_2^*P = R_1^* + h_0^* pk_{CA} + h_{12}^* pk_{ID^*} + h_2^* U_1^*$. Thus, $\mathcal{B}$ computes $\alpha = \frac{Z_1^* - Z_2^*}{h_{11}^* - h_{12}^*}$ as solution to ECDLP.

Based on the simulation, the success probability of solving ECDLP depends on the following events:

E_1: $\mathcal{B}$ does not abort during simulation.

E_2: $\mathcal{F}_2$ is able to create a valid forgery.

E_3: The forgery is done with respect to targeted identity.

Thus, $P[E_1] \geq \left(1 - \frac{1}{q}\right)^q$, $P[E_2|E_1] = \epsilon$ and $P[E_3|E_1 \wedge E_2] = \frac{1}{q}$. Therefore, $Succ_{\mathcal{B},G_1}^{(ECDLP)} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$, where, q is the maximum of all queries made by forger $\mathcal{F}_2$.

Theorem 3: In the ROM, forger $\mathcal{F}_2$ (type-2) can (t, q, ϵ) forge the introduced PFCBAS. The success probability of forging the signature is $Succ_{\mathcal{F}_2, PFCBAS}^{EUF-CMA} \geq \epsilon$. Then, a polynomial algorithm $\mathcal{B}$ can be designed. It will solve a random instance of ECDLP with success probability $Succ_{\mathcal{B},G_1}^{ECDLP} \geq \frac{1}{q}\left(1 - \frac{1}{q}\right)^q \epsilon$, where q is the maximum of all requests made by $\mathcal{F}_2$.

Proof: To construct the proof, the challenge is $(P, Q = \alpha P) \in G_T \times G_T$, a random instance of ECDLP to be taken by $\mathcal{B}$ as input. The aim of $\mathcal{B}$ is to outputs α as solution.

$\mathcal{B}$ sets all the parameters as in Lemma 2 and sets $Q = pk_{ID_w}$ for some $1 \leq w \leq n$. Then, forger $\mathcal{F}_1$ requests all the oracles and obtain the responses as in Lemma 2.

Eventually, by Forking lemma [39], $\mathcal{B}$ outputs two aggregate signatures (R^*, U^*, Z^*) and (R^*, U^*, Z^{**}) on same message set $\{m_1^*, m_2^*, m_3^*, ...m_n^*\}$ under users $\{ID_1^*, ID_2^*, ID_3^*, ...ID_n^*\}$ with the same random tuple but different hashed values H_1. It is needed that for above w, ID_w^* is not requested to Corruption(.)-Oracle and (ID_w^*, m_w^*) is not queried to CBSignGen(.)-Oracle. Now, without loss of generality, we suppose $w = 1$. Thus, $Z^* = \sum_{i=2}^{n} Z_i^* + Z_1^*$ and $Z^{**} = \sum_{i=2}^{n} Z_i^{**} + Z_1^{**}$. Where, $Z_1^*P = R_1^* + h_0^* pk_{CA} + h_1^* pk_{ID_w^*} + h_2^* U_1^*$ and $Z_1^{**}P = R_1^* + h_0^* pk_{CA} + h_1^{**} pk_{ID_w^*} + h_2^* U_1^*$. Thus, $\mathcal{B}$ computes $\alpha = \frac{Z_1^* - Z_1^{**}}{h_1^* - h_1^{**}}$ as solution to ECDLP.

The probability is same as in Lemma 2.

C. Security Analysis

Based on the above discussion, proposed PFCBAS scheme presents the following security features.

1) Data Modification: During $CBSignVer(.)$ and $CBSignAggVer(.)$ the equations $Z_iP = R_i + H_0(ID_i, pk_{ID_i}, R_i)pk_{CA} + H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i$ and $ZP = \sum_{i=1}^{n}(R_i) + (\sum_{i=1}^{n} H_0(ID_i, pk_{ID_i}, R_i))pk_{CA} + \sum_{i=1}^{n} H_1(m_i, pk_{ID_i}, U_i, R_i)pk_{ID_i} + \sum_{i=1}^{n} H_2(m_i, ID_i, pk_{ID_i}, U_i, R_i)U_i$ must be satisfied. Thus, modified message will output a reject. Therefore, PFCBAS resists the data modification attack.
2) Impersonation: From the unforgeability analysis and the theorems proved, it is evident that proposed PFCBAS opposes impersonation attack.

TABLE II
ENERGY CONSUMPTION (ML JOULE) COMPARISON OF OUR PFCBAS SCHEME WITH EXISTING AS SCHEMES

Scheme	CBSign	CBSignVer	CBSigAggVer
PFCBAS	4.243	16.972	(n+1)8.486
Kumar *et al.* [1]	12.72	117.504	104.77+12.72n
Cheng *et al.* [2]	16.972	113.260	104.77+8.486n
Zhang *et al.* [3]	13.708	117.504	104.77+12.72n
Kumar *et al.*[4]	12.72	117.504	104.77+12.72n

3) Replay: Since, during signing phase the certificates are implicitly used as signing key and thus, the violation of freshness will cause reject of signature verification. Hence, PFCBAS protects freshness.

VII. PERFORMANCE COMPARISON

This section presents the performance comparison of our PFCBAS scheme with existing efficient aggregate signature schemes. In literature, the work reported in [26] is the first pairing free CBS scheme from DLP in finite fields. The proposed PFCBAS scheme is also inspired from this technique. However, for e-HMS purpose, we focus on aggregation of signatures and therefore, we do not compare PFCBAS with this scheme. Recently, several efficient CLAS schemes have been proposed by the research community [1], [2], [3], [4]. Therefore, we compare our scheme with several CLAS schemes. We compute the total computational cost in miliseconds (msec.) and also compute the associated energy consumption. To compute, we follow [40], [1] where a Tate pairing on 159-bit subgroup of an MNT curve with an embedding degree 6 and with 80-bit security level. The CPU is Intel i7 (3.07 GHz) and the benchmarks for pairing computation is 3.21 msec and during the same experiment it consumes scalar multiplication, modular exponentiation and map to point hash are 0.39 msec, 0.39 msec and 0.09 msec, respectively. In Table-I, we denote pairing by P, scalar multiplication by S and map to point hash by H. Thus, our scheme computational costs are Sign (0.39 msec.), Verify (1.56 msec.) and Aggregate Verify (0.78(n+1) msec.).

The computational costs of [1] for Sign, Verify and Aggregate Verify are 1.17 msec (3 times of PFCBAS), 10.8 msec (approx 7 times of PFCBAS) and 9.63+1.17n msec (8.85+0.39n msec, more than PFCBAS), respectively. Scheme [2] takes 1.56 msec (4 times of PFCBAS), 10.41 msec (6.67 times of PFCBAS) and 9.63+0.78n msec (8.85 msec, more than PFCBAS), for Sign, Verify and Aggregate verify, respectively. The times needed of [3] for Sign, Verify and Aggregate Verify are 1.26 msec (3.23 times of PFCBAS), 12.84 msec (approx 8.23 times of PFCBAS) and 9.63+3.39n msec (8.85+2.61n msec, more than PFCBAS), respectively. Scheme [4] takes 1.17 msec (3 times of PFCBAS), 10.8 msec (approx 7 times of PFCBAS) and 9.63+1.17n msec (8.85+0.39n msec, more than PFCBAS), for Sign, Verify and Aggregate verify respectively. Therefore, our devised PFCBAS scheme is the most efficient aggregate signature (Figure-4).
Recently, the aggregate signature schemes are widely deployed to the e-HMS using sensors. Since, sensors are energy

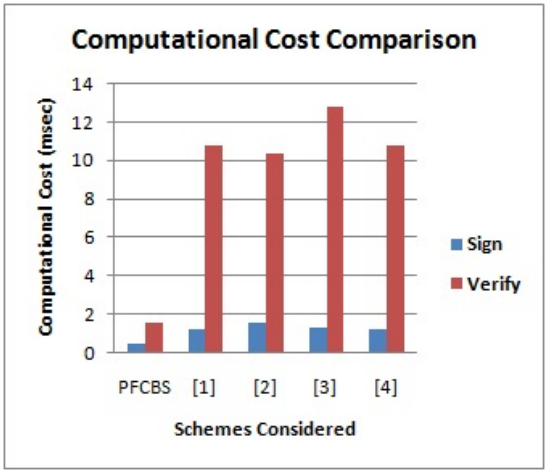

Fig. 4. Computational Cost Comparison

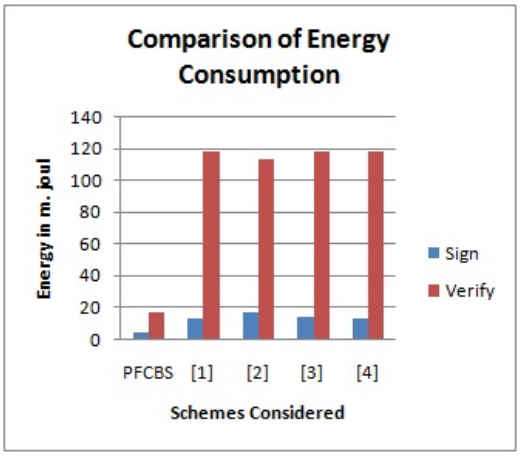

Fig. 5. Comparison of Energy Consumption

constrained devices, hence energy efficient schemes are the most appealing. We compute energy consumption by different schemes. To this end, we consider [1] where it is defined that $E_c = WT_c$. Here, W is 10.88W (maximum power of CPU), E_c energy consumed and T_c is computational cost. From Table-II, our PFCBAS scheme consumes 4.243+16.972+(n+1)8.486 mj (Milijoule) and other schemes consume more power. Therefore, our scheme supports green technology (Figure-5).

To sum up, the proposed PFCBAS scheme is the most efficient in terms of computational cost and energy consumption. Since, most of the aggregate signatures are designed in IDPKC or CLPKC and thus, they suffer with key escrow or secret key distribution problems, while our scheme resists these problems. Thus, it is the most appealing and efficient aggregation method to deploy on e-HMS.

VIII. CONCLUSION

Recently, e-HMS is widely used in smart cities. These monitoring systems are developed by using BSNs. The wireless communication in BSNs enhances the possibility of security

TABLE I
PERFORMANCE COMPARISON OF PROPOSED PFCBAS SCHEME WITH EXISTING AS SCHEMES

Scheme	CBSign	msec	CBSignVer	msec	CBSignAggVer	msec
PFCBAS	$1S$	0.39	$4S$	1.56	$(2n+2)S$	(n+1)0.78
Kumar *et al.* [1]	$3S$	1.17	$3P+3S$	10.8	$3P+3nS$	9.63+1.17n
Cheng *et al.* [2]	$4S$	1.56	$3P+2S$	10.41	$3P+2nS$	9.63+0.78n
Zhang *et al.* [3]	$3S+1H$	1.26	$4P$	12.84	$(3+n)P+2nH$	9.63+3.39n
Kumar *et al.* [4]	$3S$	1.17	$3P+3S$	10.8	$3P+3nS$	9.63+1.17n

threats. To provide authentication and integrity to e-HMS, this article introduced a pairing free certificate-based aggregate signature (PFCBAS) scheme. In e-HMS, the final signatures are verified by medical professional. Since, professionals use hand held devices such as mobile phone and therefore, the verification must be efficient. From Table-I, n signatures take $1.56n$ msec to be verified. While, n aggregated signatures take $0.78(n+1)$ msec to be verified. Thus, as n increases, aggregate verification takes $\approx 50\%$ computational cost. Hence, comparison of computational cost and energy consumption shows that PFCBAS is the most appealing to e-HMS. The proposed PFCBAS is proven to be secure under infeasibility of ECDLP. In future, we will apply the scheme for smart grid cyber-physical systems.

REFERENCES

[1] 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: Informatics and Systems*, 2017.

[2] 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.

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

[4] P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam, "Secure cls and cl-as schemes designed for vanets," *The Journal of Supercomputing*, pp. 1–23, 2018.

[5] H. Ng, M. Sim, and C. Tan, "Security issues of wireless sensor networks in healthcare applications," *BT Technology Journal*, vol. 24, no. 2, pp. 138–144, 2006.

[6] WHO, "Global diffusion of ehealth: Making universal health coverage achievable, report of the third global survey on ehealth." WHO, 2016.

[7] M. R. Yuce, S. W. 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.

[8] J. W. Ng, B. P. Lo, O. Wells, M. Sloman, N. Peters, A. Darzi, C. Toumazou, and G. Z. Yang, "Ubiquitous monitoring environment for wearable and implantable sensors (ubimon)," 2004.

[9] "The mobihealth project (ist-2001-36006) is funded by the european commission under the "information society technologies" programme."

[10] N. Oliver and F. Flores-Mangas, "Healthgear: a real-time wearable system for monitoring and analyzing physiological signals," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*. IEEE, 2006, pp. 4–pp.

[11] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *International workshop on wearable and implantable body sensor networks*, vol. 5. Boston, MA;, 2004.

[12] M. Obaidat and N. Boudriga, *Security of E-systems and Computer Networks*. Cambridge University Press, 2007.

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

[14] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the Theory and Application of Cryptographic Techniques*, vol. 196. Berlin, Heidelberg: Springer Verlag, 1984, pp. 47–53.

[15] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, vol. 2894. Berlin Heidelberg: Springer Verlag, 2003, pp. 452–473.

[16] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "On the security of certificateless signature schemes from asiacrypt 2003," in *International Conference on Cryptology and Network Security*, vol. 3810. Berlin, Heidelberg: Springer Verlag, 2005, pp. 13–25.

[17] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 2656. Berlin, Heidelberg: Springer Verlag, 2003, pp. 272–293.

[18] B. G. Kang, J. H. Park, and S. G. Hahn, "A certificate-based signature scheme," in *Cryptographers Track at the RSA Conference*, vol. 2984. Berlin, Heidelberg: Springer Verlag, 2004, pp. 99–111.

[19] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Certificate-based signature: security model and efficient construction," in *European Public Key Infrastructure Workshop*, vol. 4582. Berlin, Heidelberg: Springer Verlag, 2007, pp. 110–125.

[20] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate-based signature schemes without pairings or random oracles," in *International Conference on Information Security*, vol. 5222. Berlin, Heidelberg: Springer Verlag, 2008, pp. 285–297.

[21] J. Zhang, "On the security of a certificate-based signature scheme and its improvement with pairings," in *International Conference on Information Security Practice and Experience*. Springer, 2009, pp. 47–58.

[22] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Constructions of certificate-based signature secure against key replacement attacks," *Journal of Computer Security*, vol. 18, no. 3, pp. 421–449, 2010.

[23] J. Li, X. Huang, Y. Zhang, and L. Xu, "An efficient short certificate-based signature scheme," *Journal of Systems and Software*, vol. 85, no. 2, pp. 314–322, 2012.

[24] J. K. Liu, F. Bao, and J. Zhou, "Short and efficient certificate-based signature," in *International Conference on Research in Networking*. Valencia, Spain: Springer, 2011, pp. 167–178.

[25] L. Cheng, Y. Xiao, and G. Wang, "Cryptanalysis of a certificate-based on signature scheme," *Procedia Engineering*, vol. 29, pp. 2821–2825, 2012.

[26] J. Li, Z. Wang, and Y. Zhang, "Provably secure certificate-based signature scheme without pairings," *Information Sciences*, vol. 233, pp. 313–320, 2013.

[27] Y. Ming and Y. Wang, "Efficient certificate-based signature scheme," in *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, vol. 2. IEEE, 2009, pp. 87–90.

[28] C. Zhou and Z. Cui, "Certificate-based signature scheme in the standard model," *IET Information Security*, vol. 11, no. 5, pp. 256–260, 2016.

[29] G. K. Verma and B. Singh, "Short certificate-based proxy signature scheme from pairings," *Transaction on Emerging Telecommunication Technologies*, 2017.

[30] G. K. Verma, B. Singh, and H. Singh, "Provably secure certificate-based proxy blind signature scheme from pairings," *Information Sciences*, vol. 468, pp. 1–13, 2018.

[31] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Eurocrypt*, vol. 2656. Springer, 2003, pp. 416–432.

[32] J. K. Liu, J. Baek, and J. Zhou, "Certificate-based sequential aggregate signature," in *Proceedings of the second ACM conference on Wireless network security*. ACM, 2009, pp. 21–28.

[33] J.-N. Chen, Q.-S. Chen, and F.-M. Zou, "Certificate-based aggregate signature scheme without bilinear pairings," *Information Hiding and Multimedia Signal Processing*, vol. 7, no. 6, pp. 1330–1336, 2016.

[34] X. Ma, J. Shao, C. Zuo, and R. Meng, "Efficient certificate-based sig-

nature and its aggregation," in *International Conference on Information Security Practice and Experience*. Springer, 2017, pp. 391–408.

[35] G. K. Verma, B. Singh, and H. Singh, "Bandwidth efficient designated verifier proxy signature scheme for healthcare wireless sensor networks," *Ad Hoc Networks*, vol. 81, pp. 100–108, 2018.

[36] S. Challa, A. K. Das, V. Odelu, N. Kumar, S. Kumari, M. K. Khan, and A. V. Vasilakos, "An efficient ecc-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks," *Computers & Electrical Engineering*, vol. 69, pp. 534–554, 2018.

[37] A. K. Das, M. Wazid, N. Kumar, M. K. Khan, K.-K. R. Choo, and Y. Park, "Design of secure and lightweight authentication protocol for wearable devices environment," *IEEE journal of biomedical and health informatics*, vol. 22, no. 4, pp. 1310–1322, 2018.

[38] M. Wazid, A. K. Das, N. Kumar, M. Conti, and A. V. Vasilakos, "A novel authentication and key agreement scheme for implantable medical devices deployment," *IEEE journal of biomedical and health informatics*, vol. 22, no. 4, pp. 1299–1309, 2018.

[39] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[40] K.-A. Shim, "Cpas: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 4, pp. 1874–1883, 2012.