J Med Syst (2013) 37:9965
DOI 10.1007/s10916-013-9965-0

ORIGINAL PAPER

An Efficient and Provably-Secure Certificateless Public Key Encryption Scheme for Telecare Medicine Information Systems

Rui Guo · Qiaoyan Wen · Huixian Shi · Zhengping Jin · Hua Zhang

Received: 16 May 2013 / Accepted: 1 August 2013 / Published online: 31 August 2013

Abstract Telecare Medicine Information Systems (TMIS) promote the traditional medical and healthcare services by information and communication technology. Since the physician and caregiver can monitor the patient's physiological condition remotely in TMIS, the confidentiality of this sensitive data should be protected, which is the key issue in the Health Insurance Portability and Accountability Act. In this paper, we propose an efficient certificateless public key encryption scheme without bilinear pairing for TMIS. Our proposal is proved to be secure in the random oracle model under the hardness assumption of computational Diffie-Hellman problem. Moreover, after modifying the original model of the certificateless encryption, this scheme achieves Girault's trust level 3. Compared with the related protocols, the perform evaluations show that our scheme is more efficient and appropriate to collocate with low power mobile devices for TMIS.

Keywords Certificateless Public Key Encryption · IND-CCA Secure · Without Bilinear Pairing · Telecare Medicine Information Systems

Introduction

Telecare Medicine Information Systems (TMIS), a typical telemedicine technology based on the wireless mobile telecommunication, consist of the lightweight devices with limited memory, small bandwidth and low power [1]. In TMIS, as shown in Fig. 1, the patient's physiological condition (e.g., blood pressure, pulse oximeter and temperature) can be monitored in time by a physician and caregiver remotely, which is possible to bring the advantages of telemedicine directly into the patient's home. These electronic medical records (EMR) transmitted in public channel should be protected for ensuring patient's privacy.

The Health Insurance Portability and Accountability Act [2], enacted by the United States Congress in 1996, provided the guidelines for privacy and security regulations that the confidentiality of the information between patient and physician should be assured. The privacy and security issues vest the patient's rights to understand and control the use of his/her sensitive information, such as name, telephone number, medical record number [3, 4]. Thus, a secure encryption scheme is essential to safeguard the confidentiality of the data related to personal health in TMIS.

According to the above descriptions, the requirements of the encryption scheme for TMIS should own the following properties.

1. *Efficiency*: In TMIS, low power consumption, limited memory space and small bandwidth are the most important issues for medical mobile device designing. The patient wishes he/she can be served anywhere for a long period. Thus, an efficient protocol is significant for extending the executing time of this mobile device.
2. *Confidentiality*: The data transmitted in the public channel between patient and doctor is all sensitive to the patient, which refers to his/her privacy. The patient does not want anyone (include the medical server) to access his/her privacy except physician. A secure protocol designed for TMIS should be provided to protect the confidentiality of the patient's sensitive information.

In the traditional public key encryption primitive, to encrypt a message, a public key infrastructure (PKI) is used to

R. Guo (✉) · Q. Wen · Z. Jin · H. Zhang
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
e-mail: grbupt@gmail.com

H. Shi
Department of Mathematics and Information Science, Shaanxi Normal University, Xi'an 710062, China

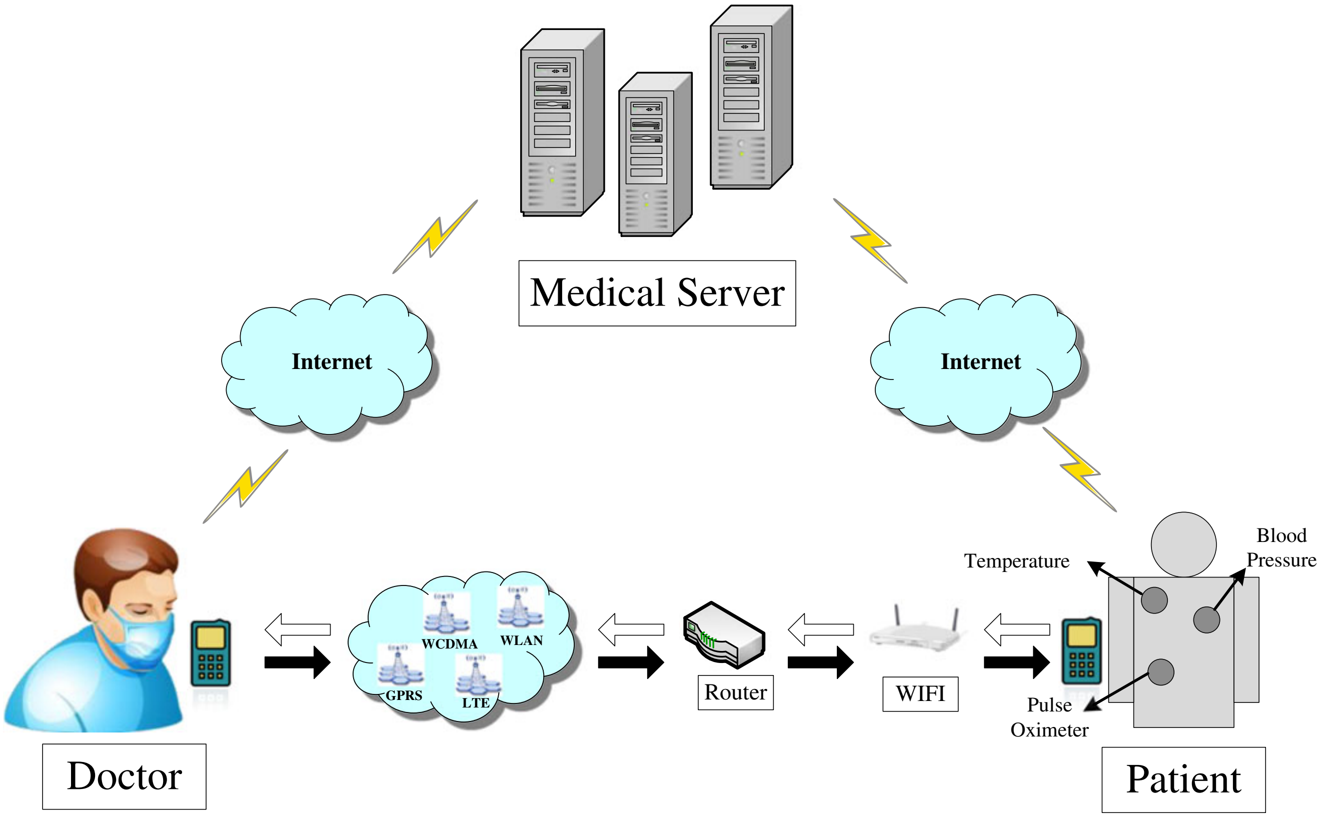

Fig. 1 Telecare medicine information system architecture

provide an assurance through the certificates issued by a certification authority (CA). However, a PKI is responsible for managing the certificate, including distribution, storage, revocation and verification of certificates, which places a computational burden on the entity. In addition, the computational ability and memory space of mobile devices in TMIS are limited. Therefore, the PKI-based encryption scheme is not suitable for TMIS. To avoid the management of digital certificate, Shamir [5] proposed the notion of identity based public key cryptography (ID-PKC) by deriving the user's public key directly from its identity information, such as e-mail address and IP address. Moreover, Boneh and Franklin presented a practical identity based encryption (IBE) scheme firstly in [6]. Nevertheless, the inherent key escrow problem in ID-PKC is a great drawback [7] since that the malicious medical server (MS) enables to eavesdrop all the sensitive messages about the patient. Hence, these two cryptographic primitive are not suitable for protecting the entity's privacy in lightweight mobile devices, such as in TMIS.

To solve these problems above, certificateless public key cryptography (CL-PKC) was introduced by Al-Riyami and Paterson [8]. In the certificateless encryption (CLE) protocol, the user combines a secret value picked by itself with the partial private key obtained from the key generation center (KGC) to generate the private key, rather than generating it completely by private key generator (PKG) in IBE. Consequently, KGC cannot access the user's private key to decrypt his/her ciphertext any more.

Several CLE schemes have been proposed in the last few years [9–14]. Libert and Quisquater [9] gave a method to achieve generic CLE constructions which were provably chosen ciphertext attacks (CCA) secure in the random oracle model. In 2007, Huang and Wong [10] proposed a generic construction of certificateless encryption which could be proven secure against the malicious-but-passive KGC attack in the standard model, and their scheme was the first one to be proven secure in the standard model. In order to resist the strong adversaries in the standard model, Dent et al. [11] presented the first strongly secure CLE scheme. In 2010, Sun and Li [12] constructed a short-ciphertext CLE scheme in the standard model with achieving adaptive chosen ciphertext security (CCA2-secure). Due to the property of CL-PKC, the above CLE constructions made use of IBE as a building block, which resulted in pairing based schemes. Being aware of the computation cost in the pairing based CLE schemes, Baek et al. [13] proposed a CLE scheme without pairing firstly in the random oracle model. In 2011, Lai et al. [14] modified Baek et al's scheme to enjoy the Girault's trust level 3 [15], the same trust level reached by a traditional PKI.

In this paper, we modify the original CLE model in [8] to achieve the Girault's trust level 3. This revised model limits the power of MS to generate false public key for the patient.

Moreover, based on this model, we propose an efficient CLE scheme without pairing for TMIS, and prove that it is secure in the random oracle model against the chosen ciphertext attacks. The new proposal needs only one scalar multiplication in decryption phase, which reduces the computational cost of patient considerably. Finally, after comparing the computation and communication cost between this scheme and others, we find that our protocol offers a better performance in efficiency.

In the next section, we briefly review the notions of computational assumptions, the Girault's trust level, and the model of CLE and its security. In Section A new CLE scheme, we propose a new CLE protocol for TMIS and analyze the security of it. In Section Comparisons, we compare the efficiency with related schemes and conclude the paper in Section Conclusions.

Preliminaries

Computational assumptions

The following computational hardness assumptions will be used in the rest of the paper.

Definition 1 Discrete Logarithm (DL) problem: Let G be an additive cyclic group with prime order p, and P be a generator of G. Given ($P, Q \in G$), find an integer $x \in Z_p^*$ satisfying $Q=xP$.

The DL assumption is that there is no polynomial time algorithm that can solve the DL problem with non-negligible probability.

Definition 2 Computational Diffie-Hellman (CDH) problem: Let G be an additive cyclic group with prime order p, and P be a generator of G. Given ($Q=xP, R=yP) \in G^2$ for any $x,y \in Z_p^*$, compute xyP.

The CDH assumption is that there is no polynomial time algorithm that can solve the CDH problem with non-negligible probability.

Let algorithm $\mathcal{A}$ be a CDH adversary who has the advantage $Adv(\mathcal{A}) = |\Pr[\mathcal{A}(P,xP,yP)=xyP]|$ in solving the CDH problem. This probability is measured over random choices of $x,y \in Z_p^*$ and the point P. Adversary $\mathcal{A}$ solves the CDH problem with (t, ε) if and only if the advantage of $\mathcal{A}$ is greater than ε in running time t. The CDH problem is said to be (t, ε)-intractable if there is no algorithm $\mathcal{A}$ that solves this problem with (t, ε).

Girault's trust level

The Girault's trust level [15] provides the trust hierarchy for public key cryptography, which can be used to evaluate the credibility of the authority.

Level 1. The authority (e.g., the CA in a PKI, the KGC in an identity based or certificateless cryptography) knows (or can easily compute) users' secret keys. Therefore, the authority can impersonate any user at any time without being detected.

Level 2. The authority does not know (or cannot easily compute) users' secret keys. Nevertheless, it can still impersonate users by generating false guarantees (e.g., false certificates in a PKI, false public keys in a certificateless cryptography).

Level 3. The authority cannot compute users' secret keys, and it can be proven that it generates false guarantees of users if it does so.

According to these definitions, we can easily find that the original certificateless cryptography falls into Level 2, and the traditional PKI achieves Level 3.

Certificateless public key encryption

In this subsection, we revise the original model of CLE in [8], and the improved one promotes the trust level of MS to Level 3.

A CLE scheme for TMIS consists of seven probabilistic polynomial time (PPT) algorithms: Setup, Patient-Key-Generation, Partial-Key-Extract, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt. These algorithms are defined as follows:

Setup	On input a security parameter 1^k, MS returns the system parameters *params*, master public key *mpk* and the master secret key *msk*. After this algorithm is over, the MS publishes *params* and *mpk*, and keeps the *msk* secretly.
Patient-Key-Generation	On input the system parameters *params*, the patient returns a secret key *sk* and a public key *pk*.
Partial-Key-Extract	On input *params*, *msk*, the patient's identity ID_P and his/her public key *pk*, MS executes this algorithm and returns a partial private key D_P to the patient via a confidential and authentic channel, and a partial public key P_P.
Set-Private-Key	On input *params*, the patient's partial private key D_P and secret key *sk*, this algorithm returns the patient's private key SK_P.
Set-Public-Key	On input *params*, the patient's partial public key P_P and public key *pk*, this algorithm returns the public key PK_P to the patient.
Encrypt	Running by the doctor. On input *params*, message M, the patient's identity ID_P, and

his/her public key PK_P, this algorithm returns a ciphertext C.

Decrypt Running this deterministic algorithm by the patient. On input *params*, the ciphertext C, and his/her private key SK_P, this algorithm returns a plaintext message M or a "*Reject*" message.

The Patient-Key-Generation algorithm in this model must be operated prior to the Partial-Key-Extract algorithm. In this way, the patient chooses his/her secret key *sk* and public key *pk* firstly. Then the MS binds the patient's public key with his/her identity ID_P to generate the patient's partial key D_P. Specifically, in this model of CLE, although MS can replace the patient's public key *pk*, there will exist two working public keys for one patient, such as *pk* and *pk'*. Moreover, two working different public keys PK_P and $PK_P^{'}$ binding one patient can result from two partial private keys, and only the MS has ability to generate these two working partial private keys. Therefore, the MS's forgery is easily tracked, which indicates that our proposal achieves the Girault's trust level 3.

Security model

In CLE scheme, as defined in [8], there are two types of adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$. A Type-I adversary $\mathcal{A}_I$ acts as a dishonest user who does not have the MS's master secret key but it is able to replace the public keys of arbitrary patient with its own choices value. By contrast, a Type-II adversary $\mathcal{A}_{II}$ acts as an honest-but-curious MS who controls the master secret key *msk* (hence it can compute the patient's partial secret key). Besides, Type-II adversary $\mathcal{A}_{II}$ is allowed to receive private keys for arbitrary identities but cannot replace any patient's public key.

Definition 3 A CLE scheme Π is said to be secure against adaptive chosen ciphertext attack (IND-CCA secure) if neither polynomial bounded adversary $\mathcal{A}$ of Type-I nor Type-II has a non-negligible advantage against the challenger in the following game:

Setup The challenger C takes a security parameter 1^k as input, and operates the Setup algorithm in Section Certificateless public key encryption. Then it gives the resulting public parameters *params* and *mpk* to $\mathcal{A}$. If $\mathcal{A}$ is of Type-I, it keeps the master secret key *msk* to itself. Otherwise (i.e., if $\mathcal{A}$ is of Type-II), returns *msk* to $\mathcal{A}$.

Phase 1 A can query the following oracles:

(1) *Public-Key-Request-Oracle*: Upon receiving a public key query for a user's identity ID, C computes (*sk*, *pk*) and the related (P_{ID}, D_{ID}), then it generates PK_{ID} and sends it to $\mathcal{A}$.

(2) *Partial-Key-Extract-Oracle*: (For Type-I adversary only.) Upon receiving a partial key query for a user's identity ID and *pk*, C computes (P_{ID}, D_{ID}) and sends them to $\mathcal{A}$.

(3) *Private-Key-Request-Oracle*: Upon receiving a private key query for a user's identity ID, C computes (*sk*, *pk*) and (P_{ID}, D_{ID}), then generates SK_{ID} and sends it to $\mathcal{A}$. It outputs ⊥ (denotes failure) if the user's public key has been replaced in the case of Type-I adversary.

(4) *Public-Key-Replace-Oracle*: (For Type-I adversary only.) For identity ID and its valid public key, $\mathcal{A}$ replaces this public key with a new one of its choice. This new value will be recorded and used by C in the coming computations or responses to the adversary's queries.

(5) *Decryption-Oracle*: On input a ciphertext and an identity ID, C returns the corresponding plaintext by using of the private key of the user, even if the public key for ID has been replaced.

Challenge Phase Once $\mathcal{A}$ decides that Phase 1 is over, it outputs and submits two messages (M_0, M_1), together with a challenger's identity ID^*. Note that $\mathcal{A}$ is not allowed to know the private key of ID^* in any way. The challenger C picks a random bit $\beta \in \{0, 1\}$ and computes C^*, which is the encryption of M_β under the current public key PK_{ID^*}. If the output of Encrypt is ⊥, adversary $\mathcal{A}$ loses the game. Otherwise, C^* is delivered to $\mathcal{A}$.

Phase 2 $\mathcal{A}$ issues a new sequence of queries as in Phase 1. However, a decryption query on the challenge ciphertext C^* for the combination of (ID^*, PK_{ID^*}) is not allowed.

Guess $\mathcal{A}$ outputs its guess β' for β. It wins the game if $\beta' = \beta$.

The guessing advantage of $\mathcal{A}$ in this game is defined to be $Adv(\mathcal{A}) = \left|Pr(\beta' = \beta) - \frac{1}{2}\right|$. $\mathcal{A}$ breaks an IND-CCA secure CLE scheme Π with (t, q_H, q_{par}, q_{pub}, q_{prv}, q_D, ε) if and only if the advantage of $\mathcal{A}$ that makes q_H times to a random oracle $H(\cdot)$, q_{par} times *Partial-Key-Extract-Oracle*, q_{pub} times *Public-Key-Request-Oracle*, q_{prv} times *Private-Key-Request-Oracle* and q_D times *Decryption-Oracle* queries is greater than ε within running time t. The scheme Π is said to be (t, q_H, q_{par}, q_{pub}, q_{prv},

q_D, ε)-IND-CCA secure if there is no adversary $\mathcal{A}$ that breaks IND-CCA secure scheme Π with (t, q_H, q_{par}, q_{pub}, q_{prv}, q_D, ε).

A new CLE scheme

In this section, we propose a new CLE scheme without bilinear pairing to protect the confidentiality of data between the patient and the doctor. The notations used throughout this paper are listed in Table 1.

Construction

The proposed CLE scheme as shown in Fig. 2 consists of the following seven PPT algorithms.

Setup: Let G be a cyclic group of prime order p with an arbitrary generator $P \in G$. The MS selects $x \in Z_p^*$ randomly and computes $X=xP$ as the master public key. Then, it chooses two collision resistant hash functions $H_1 : \{0,1\}^{l_0} \times G^* \times G^* \to Z_p^*$ and $H_2 : G^* \to \{0,1\}^l$. The system parameters are $params = \{p, G, P, X, H_1, H_2\}$, and the master secret key is $msk=x$.

Patient-Key-Generation: The patient picks $y \in Z_p^*$ uniformly at random and computes $Y=yP$, and he/she returns $(sk, pk)=(y, Y)$.

Partial-Key-Extract: The MS picks $\alpha \in Z_p^*$ at random and computes $r_P=\alpha P$ and $z_P=\alpha+xH_1 \quad (ID_P \| r_P \| pk)$, where ID_P is the patient's identity. After that, MS returns $(P_P, D_P)=(r_P, z_P)$ as the patient's partial key.

Set-Private-Key: Set $SK_P=(sk,D_p)=(y,z_P)$, it returns SK_P as the patient's private key.

Set-Public-Key: Let $PK_P=(pk,P_p)=(Y,r_P)$, it returns PK_P as the patient's public key.

Table 1 Notation defined in this scheme

ID_p	the identity of Patient
$H_i(\cdot)$	the collision-resistant hash function (i=1, 2)
p	the large prime number
G	the cyclic additive group
P	the generator of G
x	the master secret key
X	the master public key
P_p	the Patient's partial public key
D_p	the Patient's partial private key
PK_p	the Patient's public key
SK_p	the Patient's private key
$\|\|$	the concatenation operation
$\oplus$	the bitwise XOR operation
N	the set of positive integer

Encrypt: Let the bit-length of M be l_1, where $l=l_0+l_1 \in N$ (N denotes the set of positive integer). The doctor picks $u \in Z_p^*$ randomly and computes the ciphertext:

$c_1 = uP$,

$c_2=H_2(u(Y+r_P+H_1(\text{ID}_P \| r_P \| pk)X)) \oplus (M \| \text{ID}_P)$.

Note that the bit-length of $M \| \text{ID}_P$ is equal to l. Then, the doctor delivers the ciphertext $C=(c_1, c_2)$ to the patient.

Decrypt: To decrypt C, the patient computes

$$M' \Big\| ID_P' = H_2((z_P + y) \cdot c_1) \oplus c_2.$$

Check whether $\text{ID}_P{}'=\text{ID}_P$. If not, output "*Reject*". Else, the patient returns M' as the plaintext of C.

The above decryption algorithm is consistent if C is a valid ciphertext, then it can derive that:

$$\begin{aligned}
&H_2((z_P + y) \cdot c_1) \oplus c_2 \\
&= H_2((\alpha + xH_1(\text{ID}_P \| r_P \| pk) + y) \cdot uP) \oplus c_2 \\
&= H_2((r_P + XH_1(\text{ID}_P \| r_P \| pk) + Y) \cdot u) \oplus c_2 \\
&= H_2((r_P + XH_1(\text{ID}_P \| r_P \| pk) + Y) \cdot u) \oplus \\
&\quad H_2(u(Y + r_P + H_1(ID_P \| r_P \| pk)X)) \oplus ((M \| \text{ID}_P) \\
&= M \| \text{ID}_P.
\end{aligned}$$

Security analysis

In this subsection, we prove that the proposed CLE scheme constructed in the previous section is secure in the random oracle model.

Theorem 1. Given H_1 and H_2 are two collision resistant hash functions. This CLE scheme is IND-CCA secure in the random oracle model assuming that there is no polynomial time algorithm that can solve the CDH problem with non-negligible probability.

This theorem following from two lemmas will show that our CLE scheme is secure against the Type-I and Type-II adversaries whose behaviors are described in **Definition 3**.

Lemma 1. This CLE scheme is $\left(t, q_{H_1}, q_{H_2}, q_{par}, q_{pub}, q_{prv}, q_D, \varepsilon\right)$ -IND-CCA secure against the Type-I adversary $\mathcal{A}$ in the random oracle assuming the CDH problem is (t', ε')-intractable, where

$$\varepsilon' > \frac{1}{q_{H_2}} \left(\frac{2\varepsilon}{e\left(q_{prv}+1\right)} - \frac{q_D q_{H_1}}{2^{l_0} p^2} - \frac{q_D}{p} \right),$$

$$t' > t + 2\left(q_{par} + q_{pub} + q_{prv}\right) t_{sm} + q_D q_{H_2} t_{sm} + 2t_{sm},$$

and t_{sm} is the time for computing scalar multiplication of the additive cyclic group G.

Proof Assuming there exists a Type-I adversary $\mathcal{A}_I$ simulating an "outside" adversary, who replaces the public key of arbitrary identities but cannot corrupt the master secret key.

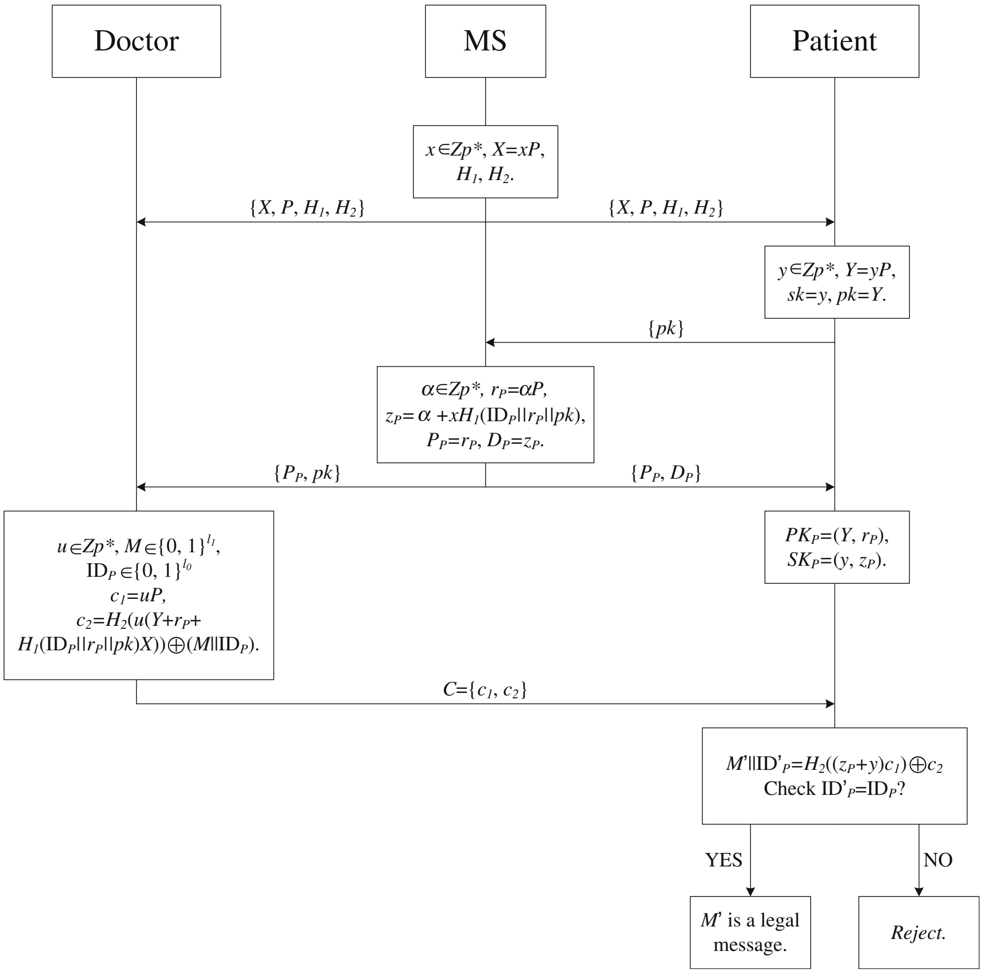
Doctor
MS
Patient
x∈Zp*, X=xP, H1, H2.
{X, P, H1, H2}
{X, P, H1, H2}
y∈Zp*, Y=yP, sk=y, pk=Y.
{pk}
α∈Zp*, rP=αP, zP=α+xH1(IDP||rP||pk), PP=rP, DP=zP.
{PP, pk}
{PP, DP}
u∈Zp*, M∈{0, 1}^l1, IDP∈{0, 1}^l0, c1=uP, c2=H2(u(Y+rP+H1(IDP||rP||pk)X))⊕(M||IDP).
PKP=(Y, rP), SKP=(y, zP).
C={c1, c2}
M'||ID'P=H2((zP+y)c1)⊕c2 Check ID'P=IDP?
YES
NO
M' is a legal message.
Reject.

Fig. 2 Our CLE scheme for TMIS

Suppose that there is another PPT algorithm $\mathcal{B}$ can solve the CDH problem in the instance of (p, P, aP, xP) with probability at least ε' and the time at most t' by interacting with $\mathcal{A}_I$. To solve this problem, $\mathcal{B}$ needs to simulate a challenger to perform each algorithm of IND-CCA game for $\mathcal{A}_I$ as follows:

Setup Algorithm $\mathcal{B}$ sets $X=xP$, where $x \in Z_p^*$ is the master secret key that is unknown to $\mathcal{B}$. Then, $\mathcal{B}$ gives $\mathcal{A}_I$ the *params*$=\{p, G. P, X, H_1, H_2\}$ as CLE system parameters, where H_1 and H_2 are random oracles. Adversary $\mathcal{A}_I$ may make queries of these two random oracles at any time during its attack. $\mathcal{B}$ responds as follows:

H_1 **queries** $\mathcal{B}$ maintains a list of tuples $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ in H_1-**List** L_1. On receiving a query (ID, r_{ID}, Y) to H_1:

(1) If $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ already appears on the list L_1, $\mathcal{B}$ responds v as an answer.
(2) Otherwise, pick $v \in Z_p^*$ randomly, add $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ to L_1 and return v as an answer.

H_2 **queries** $\mathcal{B}$ maintains a list of tuples $\langle(\mathrm{ID},T),R\rangle$ in H_2-**List** L_2. On receiving a query (ID, T) to H_2:

(1) If $\langle(\mathrm{ID},T),R\rangle$ exists in the list L_2, $\mathcal{B}$ responds R as an answer.

(2) Otherwise, choose $R \in \{0, 1\}^l$ uniformly at random, add $\langle(\mathrm{ID},T),R\rangle$ to L_2 and return R as an answer.

Phase 1 $\mathcal{A}_I$ can issue a number of the following oracle queries.

Partial-Key-Extract-Oracle $\mathcal{B}$ maintains a **PartialKeyList** of tuples $\langle \mathrm{ID},(r_{ID},z_{ID})\rangle$. On receiving a query ID, $\mathcal{B}$ responds as follows:

(1) If $\langle \mathrm{ID},(r_{ID},z_{ID})\rangle$ exists in **PartialKeyList**, return (r_{ID}, z_{ID}) as an answer.
(2) Otherwise, pick $z_{ID}, v \in Z_p^*$ at random, and compute $r_{ID}=z_{ID}P-vX$. Add (ID, r_{ID}, v) to L_1 and $\langle \mathrm{ID},(r_{ID},z_{ID})\rangle$ to **PartialKeyList**, return (r_{ID}, z_{ID}) as an answer.

Public-Key-Request-Oracle $\mathcal{B}$ maintains a **PublicKeyList** of tuples $\langle$ID,(r_{ID},Y),$coin\rangle$. On receiving a query ID, $\mathcal{B}$ responds as follows:

(1) If $\langle$ID,(r_{ID},Y),$coin\rangle$ exists in **PublicKeyList**, return PK_{ID}=(r_{ID},Y) as an answer.
(2) Otherwise, choose $coin \in \{0, 1\}$ at random so that Pr[$coin$=0]=δ. (δ will be defined later.)
(3) If $coin$=0, do the following:

(a) If $\langle$ID,(r_{ID},z_{ID})$\rangle$ exists in **PartialKeyList**, pick $y \in Z_p{}^*$ at random and compute Y=yP. Then, add $\langle$ID,(y,z_{ID})$\rangle$ to **PrivateKeyList** (which will be defined later) and $\langle$ID,(r_{ID},Y),$coin\rangle$ to **PublicKeyList** respectively, return PK_{ID}=(r_{ID},Y) as an answer.
(b) Otherwise, run the *Partial-Key-Extract-Oracle* to get partial keys (r_{ID}, z_{ID}) about ID. Pick $y \in Z_p{}^*$ at random and compute Y=yP. Then, add $\langle$ID,(r_{ID},z_{ID})$\rangle$ to **PrivateKeyList** and $\langle$ID,(r_{ID},Y),$coin\rangle$ to **PublicKeyList** respectively, return PK_{ID}=(r_{ID}, Y) as an answer.

(4) Otherwise (if $coin$=1), pick α,$y \in Z_p{}^*$ at random and compute r_{ID}=αP, Y=yP, add $\langle$ID,(y,$*$),$\alpha\rangle$ to **PrivateKeyList** (where "$*$" denotes the arbitrary value), and $\langle$ID,(r_{ID},Y),$coin\rangle$ to **PublicKeyList**, return PK_{ID}=(r_{ID}, Y) as an answer.

Private-Key-Request-Oracle $\mathcal{B}$ maintains a **PrivateKeyList** of tuples $\langle$ID,(y,z_{ID}),$\alpha\rangle$. On receiving a query ID, β responds as follows:

(1) Run *Public-Key-Request-Oracle* on ID to get a tuple $\langle$ID,(r_{ID},Y),$coin\rangle$ from **PublicKeyList**.
(2) If $coin$=0, search a tuple $\langle$ID,(y,z_{ID}),$\alpha\rangle$ in **PrivateKeyList** and return SK_{ID}=(y, z_{ID}) as answer.
(3) Otherwise, return "*Abort*" and terminate.

Public-Key-Replace-Oracle $\mathcal{A}_I$ may replace any public key with a new value of its choice and $\mathcal{B}$ records all the changes.

Decryption-Oracle On receiving a query $\langle$ID,PK_{ID},$C\rangle$, where C=(c_1, c_2) and PK_{ID}=(r_{ID}, Y). $\mathcal{B}$ responds as follows:

(1) Search a tuple $\langle$ID,(r_{ID},Y),$coin\rangle$ in **PublicKeyList**.
(2) If such a tuple exists and $coin$=0.

(a) Search **PrivateKeyList** for a tuple $\langle$ID,(y,z_{ID})$\rangle$.
(b) Compute $M'\|$ID$'$=H_2((z_{ID}+y)$\cdot c_1$)$\oplus c_2$.
(c) If ID$'$=ID, return M'as plaintext and "*Reject*" otherwise.

(3) Else, if such a tuple exists and $coin$=1.

(a) Perform H_1 **queries** to get a tuple $\langle$(ID,r_{ID},Y),$v\rangle$.
(b) If there exists $\langle$(ID,T),$R\rangle \in L_2$ such that c_2=$R \oplus$($M\|$ID), return M as plaintext and "*Reject*" otherwise.

(4) Else, if such a tuple does not exist (which means that the public key of a target user is replaced by A_I), perform the same algorithm in (3)

Challenge Phase Once $\mathcal{A}_I$ decides that Phase 1 is over, it outputs two messages (M_0, M_1) and a challenge identity ID*. On receiving a challenge query $\langle$ID*, (M_0,M_1)$\rangle$, $\mathcal{B}$ responds as follows:

(1) Run *Public-Key-Request-Oracle* on ID* to get a tuple $\langle ID^*, (r_{ID^*}, Y^*), coin\rangle \in$ **PublicKeyList**.
(2) If $coin$=0, return "*Abort*" and terminate.
(3) Otherwise, do the following:

(a) Search a tuple $\langle$ID*,(y^*,$*$),$\alpha^*\rangle$ in **PrivateKeyList**. (In this case, we know that $r_{ID^*} = \alpha^* P, Y^* = y^* P$.)
(b) Pick $c_2{}^* \in \{0,1\}^l$ and $\beta \in \{0, 1\}$ at random.
(c) Set $c_1^* = aP, \Gamma = ar_{ID^*}$ and $v^* = H_1(ID^*\|r_{ID^*}\|Y^*)$.
(d) Define
$$H_2(a(Y^* + r_{ID^*} + H_1\ (ID^*\|r_I\ D^*\|Y^*)X)) = c_2^* \oplus (M_\beta\|ID^*).$$
Note that $\mathcal{B}$ does not know "a".

(4) Return C^*=($c_1{}^*$,$c_2{}^*$) as the target ciphertext.

Phase 2 $\mathcal{A}_I$ makes the same queries as it did in Phase 1. However, there is no *Partial-Key-Extract-Oracle* or *Private-Key-Request-Oracle* query on ID* is allowed. Also, no *Decryption-Oracle* query should be made on the ciphertext C^*=($c_1{}^*$,$c_2{}^*$) for the combination of ID* and PK_{ID^*} that encrypted plaintext M_β.

Guess $\mathcal{A}_I$ outputs a guess β' for β, and wins the game if β'= β. Then, $\mathcal{B}$ will be able to solve the CDH problem by computing $\left(c_1^* \cdot z_{ID^*} - \Gamma\right)/v^*$.

Analysis We denote the event that ID* has been queried to H_1 as **Ask** $H_1{}^*$. Also, by **Ask** $H_2{}^*$, we denote the event that $\langle(\mathrm{ID}^*,T^*),R^*\rangle$ has been queried to H_2. Provided that the event **Ask** $H_2{}^*$ happens, $\mathcal{B}$ will enable to solve the CDH problem by picking a tuple $\langle(\mathrm{ID}^*,T^*),R^*\rangle$ in L_2 and compute $\left(c_1^*\cdot z_{ID^*}-\Gamma\right)/v^*$ with probability at least $1/q_{H_2}$. Hence, we have $\varepsilon'\geq\left(1/q_{H_2}\right)Pr\left[AskH_2^*\right]$.

If $\mathcal{B}$ does not abort during the game, the simulations of *Partial-Key-Extract-Oracle*, *Public-Key-Request-Oracle*, *Private-Key-Request-Oracle* and the target ciphertext is identically distributed as the real attack in our construction. Because $\mathcal{B}$'s responses to all hash queries are uniformly and independently distributed as in the real attack, and all responses to $\mathcal{A}_I$ can pass the validity test unless $\mathcal{B}$ aborts in the game.

Thus, we find that when a public key PK_{ID} has not been replaced or produced under $coin=1$, the simulation is perfect as $\mathcal{B}$ knowing the corresponding private key SK_{ID}. Otherwise, a simulation error may occur in *Decryption-Oracle*, and let **DecErr** denote this event. Suppose that ID, $PK_{ID}=(r_{ID},Y)$ and $C=(c_1,\ c_2)$ have been issued as a valid decryption query. Even if C is a valid ciphertext, there is a possibility that C can be produced without querying $\langle(\mathrm{ID},T),R\rangle$ to H_2. Let **Valid** be an event that C is a valid ciphertext, **Ask** H_1 and **Ask** H_2 be events that (ID, r_{ID}, Y) has been queried to H_1 and (ID,T) to H_2 respectively. Since **DecErr** is an event that **Valid**|¬**Ask** H_2 happens during the entire simulation and q_D *Decryption-Oracle* queries are operated, we have Pr[**DecErr**]=q_D Pr[**Valid**|¬**Ask** H_2]. However,

$$\begin{aligned}Pr[(\mathbf{Valid}|\neg\mathbf{Ask}H_2]&\leq Pr[(\mathbf{Valid}\wedge\mathbf{Ask}H_1|\neg\mathbf{Ask}H_2]\\&\quad+Pr[(\mathbf{Valid}\wedge\neg\mathbf{Ask}H_1|\neg\mathbf{Ask}H_2]\\&\leq Pr[(\mathbf{Ask}H_1|\neg\mathbf{Ask}H_2]\\&\quad+Pr[(\mathbf{Valid}|\neg\mathbf{Ask}H_1\wedge\neg\mathbf{Ask}H_2]\\&\leq\left(q_{H_1}/\left(2^{l_0}p^2\right)\right)+(1/p).\end{aligned}$$

Let the event ($\mathbf{Ask}H_2{}^*\vee$**DecErr**)|¬**Abort** be denoted by **E**, where **Abort** is an event that $\mathcal{B}$ aborts during the simulation. The probability that ¬**Abort** happens is given by $\delta^{q_{prv}}(1-\delta)$ which is maximized at $\delta=1-1/(q_{prv}+1)$. Hence, we have Pr[¬**Abort**]≤$1/(e(q_{prv}+1))$, where e denotes the base of the natural logarithm.

If **E** does not happen, it is clear that $\mathcal{A}_I$ does not gain any advantage greater than 1/2 to guess β due to the randomness of the output of the random oracle H_2. Namely, we have Pr[$\beta'=\beta$|¬**E**]≤1/2.

By definition of ε, we have

$$\begin{aligned}&\varepsilon<|Pr[\beta'=\beta]-(1/2)|\\&=|Pr[\beta'=\beta|\neg\mathbf{E}]Pr[\neg\mathbf{E}]+Pr[\beta'=\beta|\mathbf{E}]Pr[\mathbf{E}]-(1/2)|\\&\leq|(1/2)Pr[\neg\mathbf{E}]+Pr[\mathbf{E}]-(1/2)|\\&=|(1/2)(1-Pr[\mathbf{E}])+Pr[\mathbf{E}]-(1/2)|\\&=(1/2)Pr[\mathbf{E}]\\&\leq\left(Pr\left[\mathbf{Ask}H_2^*\right]+Pr[\mathbf{DecErr}]\right)/(2Pr[\neg\mathbf{Abort}])\\&\leq\left(e\left(q_{prv}+1\right)/2\right)\left(q_{H_2}\varepsilon'+\left(q_Dq_{H_1}/\left(2^{l_0}p^2\right)\right)+(q_D/p)\right).\end{aligned}$$

Consequently, we obtain

$$\varepsilon'>\frac{1}{q_{H_2}}\left(\frac{2\varepsilon}{e\left(q_{prv}+1\right)}-\frac{q_Dq_{H_1}}{2^{l_0}p^2}-\frac{q_D}{p}\right).$$

The running time of adversary $\mathcal{B}$ is

$$t'>t+2\left(q_{par}+q_{pub}+q_{prv}\right)t_{sm}+q_Dq_{H_2}t_{sm}+2t_{sm},$$

Where t_{sm} denotes the time for computing scalar multiplication of the additive cyclic group G.

The following lemma shows that our CLE scheme is secure against the Type-II adversary.

Lemma 2. This CLE scheme is $(t,q_{H_1},q_{H_2},q_{par},q_{pub},q_{prv},q_D,\varepsilon)$ -IND-CCA secure against the Type-II adversary $\mathcal{A}$ in the random oracle assuming the CDH problem is (t', ε')-intractable, where

$$\varepsilon'>\frac{1}{q_{H_2}}\left(\frac{2\varepsilon}{e\left(q_{prv}+1\right)}-\frac{q_Dq_{H_1}}{2^{l_0}p^2}-\frac{q_D}{p}\right),$$
$$t'>t+2(q_{pub}+q_{prv})t_{sm}+q_Dq_{H_2}t_{sm}+2t_{sm},$$

and t_{sm} is the time for computing scalar multiplication of the additive cyclic group G.

Proof Assuming there exists an algorithm $\mathcal{A}_{II}$ who models an "insider" adversary. Suppose that another PPT algorithm $\mathcal{B}$ enables to solve the CDH problem though $\mathcal{A}_{II}$ with probability at least ε' and the time at most t'.

$\mathcal{B}$ is given (p, P, aP, bP) as an instance of the CDH problem. In order to solve this problem, $\mathcal{B}$ needs to simulate a challenger to execute each phase of IND-CCA game for $\mathcal{A}_{II}$ as follows:

Setup Algorithm B picks the master secret key $x\in Z_p{}^*$
randomly and computes $X=xP$. Then, $\mathcal{B}$ gives the system parameters

$params=\{p,G,P,X,H_1,H_2\}$ to $\mathcal{A}_{II}$, where H_1 and H_2 are random oracles.
Adversary $\mathcal{A}_{II}$ queries these two random oracles at any time during its attack. $\mathcal{B}$ responds as follows:

H_1 **queries**

$\mathcal{B}$ maintains a list of tuples $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ in H_1 - **List** L_1. On receiving a query (ID,r_{ID},Y) to H_1:

(1) If $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ already appears on the list L_1, return v as an answer.
(2) Otherwise, pick $v\in Z_p^*$ at random, add $\langle(\mathrm{ID},r_{ID},Y),v\rangle$ to L_1 and return v as an answer.

H_2 **queries**

$\mathcal{B}$ maintains a list of tuples $\langle(\mathrm{ID},T),R\rangle$ in H_2-**List** L_2. On receiving a query (ID,T) to H_2:

(1) If $\langle(\mathrm{ID},T),R\rangle$ exists in the list L_2, return R as an answer.
(2) Otherwise, choose $R\in\{0,1\}^l$ uniformly at random, add $\langle(\mathrm{ID},T),R\rangle$ to L_2 and return R as an answer.

Phase 1 *Public-Key-Request-Oracle*

$\mathcal{A}_{II}$ issues the following oracle queries.
$\mathcal{B}$ maintains a **PublicKeyList** of tuples $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$. On receiving a query ID, $\mathcal{B}$ responds as follows:

(1) If $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ exists in **PublicKey List**, return $PK_{ID}=(r_{ID},Y)$ as an answer.
(2) Otherwise, pick $coin\in\{0,1\}$ at random so that $\Pr[coin=0]=\delta$. (δ is the same as it in the proof of **Lemma 1**.)
(3) If $coin=0$, choose $y,\alpha\in Z_p^*$ at random and compute $Y=yP$, $r_{ID}=\alpha P$ and $z_{ID}=\alpha+xH_1(ID\|r_{ID}\|Y)$. Then, add $\langle\mathrm{ID},(y,z_{ID}),\alpha\rangle$ to **PrivateKeyList** and add $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ to **PublicKeyList** respectively, return $PK_{ID}=(r_{ID},Y)$ as an answer.
(4) Otherwise (if $coin=1$), pick $\alpha,y\in Z_p^*$ at random and compute $r_{ID}=\alpha aP$, $Y=yP$ and $z_{ID}=\alpha+bxH_1(ID\|r_{ID}\|Y)$. Then, add $\langle\mathrm{ID},(y,*),\alpha\rangle$ to **PrivateKeyList** (where "*" denotes the arbitrary value), and $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ to **PublicKeyList**,return $PK_{ID}=(r_{ID},Y)$ as an answer.

Private-Key-Request-Oracle

$\mathcal{B}$ maintains a **PrivateKeyList** of tuples $\langle\mathrm{ID},(y,z_{ID}),\alpha\rangle$. On receiving a query ID, $\mathcal{B}$ responds as follows:

(1) Perform *Public-Key-Request-Oracle* on ID to get a tuple $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ from **PublicKeyList**.
(2) If $coin=0$, search **PrivateKeyList** for a tuple $\langle\mathrm{ID},(y,z_{ID}),\alpha\rangle$ and return $SK_{ID}=(y,z_{ID})$ as an answer.
(3) Otherwise, return "*Abort*" and terminate.

Decryption-Oracle

On receiving a query $<\mathrm{ID},PK_{ID},C>$, where $C=(c_1,c_2)$ and $PK_{ID}=(r_{ID},Y)$. $\mathcal{B}$ responds as follows:

(1) Search a tuple $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ in **PublicKeyList**.
If such a tuple exists and $coin=0$, search **PrivateKeyList** for a tuple $\langle\mathrm{ID},(y,z_{ID})\rangle$
(Note that $\langle\mathrm{ID},(r_{ID},Y),coin\rangle$ must exist in **PublicKeyList**. While $coin=0$, the tuple $\langle\mathrm{ID},(y,z_{ID}),\alpha\rangle$ exists in **PrivateKeyList**). Then, set $SK_{ID}=(y,z_{ID})$ and operate Decrypt.
Finally, return the results of Decrypt algorithm.
(2) Otherwise (if $coin=1$), run H_1 **queries** to access a tuple $\langle(\mathrm{ID},r_{ID},Y),v\rangle$. If there exists $\langle(\mathrm{ID},T),R\rangle$ in L_2 such that $c_2=R\oplus(M\|\mathrm{ID})$, return M as plaintext and "*Reject*" otherwise.

Challenge Phase

When Phase 1 is over, $\mathcal{A}_{II}$ output two messages (M_0,M_1) and a challenge identity ID*. On receiving a challenge query $<\mathrm{ID}^*,(M_0,M_1)>$:

(1) Taking ID* as input, $\mathcal{B}$ runs *PublicKey-Request-Oracle* and gets a tuple $\langle ID^*,(r_{ID^*},Y^*),coin\rangle$ from **PublicKey List**.
(2) If $coin=0$, return "*Abort*" and terminate.
(3) Otherwise, do the following:

(a) Search for a tuple $\langle ID^*,(y^*,z_{ID^*}),\alpha^*\rangle$ from **PrivateKeyList**. (In this case, we know that $Y^*=y^*P$, $r_{ID^*}=\alpha^*aP$.)
(b) Choose $c_2^*\in\{0,1\}^l$ and $\beta\in\{0,1\}$ randomly.
(c) Set $c_1^*=aP$ and $v^*=H_1(ID^*\|r_{ID^*}\|Y^*)$.
(d) Define $H_2(a(Y^*+r_{ID^*}+H_1(ID^*\|r_{ID^*}\|Y^*)X))=c_2^*\oplus(M_\beta\|ID^*)$. Note that $\mathcal{B}$ does not know "a".

(4) Return $C^*=(c_1^*,c_2^*)$ as the target ciphertext.

Phase 2

$\mathcal{A}_{II}$ repeats the same methods as in Phase 1. Moreover, no private key extraction on ID* is allowed and no decryption query

Table 2 Cryptographic operation time

Pairing	Exponentiation	Scalar multiplication
2.5 ms	3.75 ms	0.62 ms

can be made on the ciphertext C^* that encrypted plaintext M_β.

Guess $\mathcal{A}_{II}$ outputs a guess β' for β, and wins the game if $\beta'=\beta$. Then, $\mathcal{B}$ will be able to solve the CDH problem by computing $\left(c_1^* \cdot z_{ID^*} - r_{ID^*}\right)/(x \cdot v^*)$.

Analysis Similar to Analysis in the proof of **Lemma 1**.

Consequently, we obtain

$$\varepsilon' > \frac{1}{q_{H_2}} \left(\frac{2\varepsilon}{e\left(q_{prv}+1\right)} - \frac{q_D q_{H_1}}{2^{l_0} p^2} - \frac{q_D}{p} \right).$$

The running time of adversary $\mathcal{B}$ is

$$t' > t + 2\left(q_{pub} + q_{prv}\right) t_{sm} + q_D q_{H_2} t_{sm} + 2t_{sm},$$

Where t_{sm} denotes the time for computing scalar multiplication of the additive cyclic group G.

To sum up, we complete the proof of **Theorem 1**.

Comparisons

In this section, we compare our CLE scheme with previous protocols on the computation complexity of encryption (**Enc**) and decryption (**Dec**), the bandwidth of the ciphertext (**Bandwidth**) and the running time (**Time**) of each scheme. Without considering the addition of two points, the hash function and exclusive-OR operations, we denote the cost of a bilinear pairing by P, the cost of an exponentiation by E, and the cost of a scalar multiplication in the additive cyclic group by S.

We simulate the cryptographic operations by using of MIRACL (version 5.6.1, [16]). The experiments are performed on a laptop using the Intel Core i5-2400 at a frequency of 3.10 GHz with 3GB memory and Windows XP operation system. Then the average running time of each operation in 100 times is obtained and demonstrated in Table 2. For pairing-based schemes, in order to implement in practice efficiently, we use the Fast-Tate-Pairing in MIRACL, which is defined over the MNT curve E/F_q [17] with characteristic a 160-bit prime and embedding degree 4. For ECC-based protocols, we employ the parameters secp160r1 [18], where $p=2^{160}-2^{31}-1$. Furthermore, we denote the length of an element in a multiplicative group to be 1024-bit. Based on the above parameter settings, the total running time to finish one round of Encrypt-Decrypt in different schemes are illustrated in Table 3. In addition, we simulate the whole procedure of our CLE scheme and the operation time is only 3.76 ms.

Table 3 Comparison of the related schemes

Scheme	Enc	Dec	Bandwidth	Time
[9]	2E+3S	1P+1E	40 bytes	15.61 ms
[11]	1P+4E	2P	512 bytes	22.5 ms
[12]	2P+2E+2S	2P+1S	36 bytes	19.36 ms
[13]	4E	3E	148 bytes	26.25 ms
[14]	3E+2S	2E	148 bytes	19.99 ms
Ours	3S	1S	40 bytes	2.48 ms

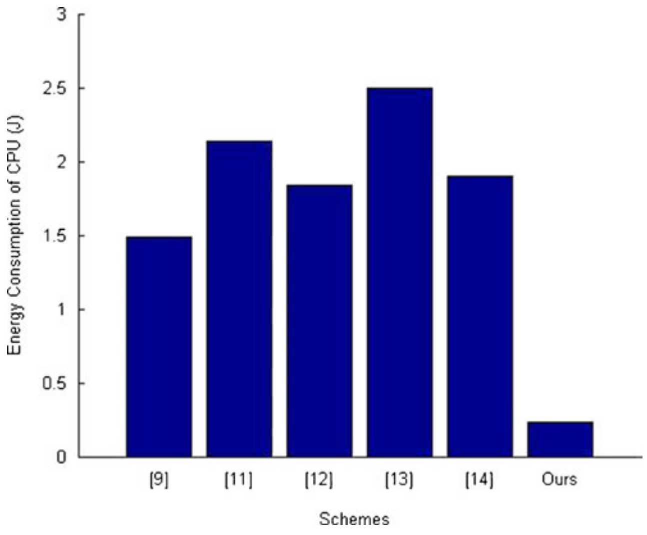

Fig. 3 Energy consumption of CPU

To the energy consumption, it is calculated as $W=P\times t$ based on the power (P) and execution time (t). Suppose that the max power of central processing unit (CPU) is 95 W. Then the energy consumption of CPU in different schemes is demonstrated in Fig. 3, which indicates that when the CPU is at full capacity, our scheme consumes less energy than others in the process of encryption and decryption.

For the communication cost, we analyze it in terms of the bandwidth of the transmitted ciphertext. Suppose that the output of one way hash function is 160-bit, and the symmetric cipher is 128-bit (e.g., AES). In our protocol and [9], each ciphertext contains one point and one hash value, thus the bandwidths of our protocol and [9] are (160+160)/8=40 bytes respectively. In [13] and [14], each ciphertext contains one exponentiation and one hash value, thus the bandwidths of [13] and [14] are (1024+160)/8=148 bytes respectively. In Dent et al.'s scheme [11], the ciphertext contains four exponentiations, the bandwidth of it is (1024×4)/8=512 bytes. At last, in the scheme of [12], the ciphertext contains one point and one symmetric cipher, and therefore the bandwidth of it is (160+128)/8=36 bytes. The detailed comparison results are also listed in Table 3, and the bandwidth of our scheme is a smaller one.

Notably, the cost of computation and communication and the power consumption at the patient side of this scheme is far less than others. These analyses show that our scheme enables to provide an efficient method to protect the confidential information between patient and doctor in TMIS.

Conclusions

We have proposed an efficient certificateless encryption paradigm for TMIS to protect the privacy of patients. In point of security, it shows that our scheme is IND-CCA secure in the random oracle model under the hardness of CDH problem. Moreover, our protocol limits the power of the medical server to replace the patient's public key. A thorough performance evaluation and experiments indicate that our proposal is advantageous over the related schemes in efficiency. These attributes render our scheme a promising approach in the privacy protection of TMIS with lightweight devices.

Acknowledgments This work is supported by NSFC (Grant Nos. 61272057, 61202434, 61170270, 61100203, 61003286, 61121061), the Fundamental Research Funds for the Central Universities (Grant No. 2012RC0612, 2011YB01).

Conflict of interest The authors declare that we have no conflict of interest.

References

1. Wu, Z.-Y., Lee, Y.-C., Lai, F. P., Lee, H.-C., and Chung, Y. F., A secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* 36:1529–1535, 2012.
2. Health Insurance Portability and Accountability Act of 1996, 104th Congress. Public Law 104–191, 1996
3. Lee, W. B., and Lee, C. D., A cryptographic key management solution for HIPAA privacy/security regulations. *IEEE Trans. Inf. Technol. Biomed.* 12(1):34–41, 2008.
4. Al-Ameen, M., Liu, J. W., and Kwak, K., Security and privacy issues in wireless sensor networks for healthcare applications. *J. Med. Syst* 36:93–101, 2012.
5. Shamir, A., Identity-based cryptosystems and signature schemes. *Proc. Adv. Cryptology* 84:47–53, 1985.
6. Boneh, D., and Franklin, M., Identity based encryption from the Weil pairing. *Proc. Adv. Cryptology* 01:213–229, 2001.
7. Oh, J. H., Lee, K. K., and Moon, S. J., How to solve key escrow and identity revocation in identity based encryption schemes. *Proc. First International Conference on Information System Security*, 290–303, 2005.
8. Al-Riyami, S. S., and Paterson, K. G., Certificateless public key cryptography. *Proc. Adv. Cryptology* 03:452–473, 2003.
9. Libert, B., and Quisquater, J. J., On constructing certificateless cryptosystems from identity based encryption. *Proc. Public Key Cryptography*, 474–490, 2006.
10. Huang, Q., and Wong, D. S., Generic certificateless encryption in the standard model. *IWSEC* 278–291, 2007.
11. Dent, A. W., Libert, B., and Paterson, K. G., Certificateless encryptions strongly secure in the standard model. *Proc. PKC'08*, 344–359, 2008.
12. Sun, Y. X., and Li, H., Short-ciphertext and BDH-based CCA2 secure certificateless encryption. *Science China: Information Science* 53: 2005–2015, 2010.
13. Baek, J., Safavi-Naini, R., and Susilo, W., Certificateless public key encryption without pairing. *Proc. ISC'05*, 134–148, 2005.
14. Lai, J. Z., Kou, W. D., and Chen, K. F., Self-generated-certificate public key encryption without pairing and its application. *Information Sciences* 181:2422–2435, 2011.
15. Girault, M., Self-certificated public keys. *Proc. EUROCRYPTO* 91:490–497, 1992.
16. Scott, M., Miracl library, Available from: http://certivox.com/.
17. Miyaji, A, Nakabayashi M, Takano S. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundementals*, E84-A, 2001.
18. The Certicom Corporation, SEC2: Recommended domain parameters, Version 1.0, 2000.