Research Article

Anonymous and Efficient Certificateless Multirecipient Signcryption Scheme for Ecological Data Sharing

Pengfei Su, Yong Xie, and Ping Liu

Department of Computer Technology and Application, Qinghai University, Xining 810016, China

Correspondence should be addressed to Ping Liu; 247750940@qq.com

Received 22 November 2019; Revised 22 June 2020; Accepted 29 July 2020; Published 19 August 2020

Academic Editor: Yuan Li

Copyright © 2020 Pengfei Su et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Air pollution, water pollution, soil erosion, land desertification, and other environmental issues are becoming more and more serious. And ecological security has become a key issue for the sustainable development of the world, so research on ecology has received more and more attention. At present, ecological data is collected and stored separately by various departments or agencies. In order to conduct better research, various institutions or individuals begin to share their own data. However, data sharing between different organizations is affected by many factors, especially data security issues. At the moment, there is no scheme that has been commonly recognized to solve the problem of ecological data sharing. To provide a secure data sharing way for ecological research, a certificateless multireceiver signcryption scheme is proposed. In this paper, the proposed scheme can solve the key escrow problem, and it can improve efficiency on the basis of ensuring security by adopting elliptic curve cryptography (ECC). A rigorous security analysis proves that the scheme can resist various security attacks and ensure privacy protection based on a random oracle model. Performance analysis also shows that this scheme has the advantage of computational overhead compared to the same type of scheme. Therefore, the scheme is very suitable for the safe sharing of ecological data.

1. Introduction

Ecological data is receiving more and more attentions nowadays. It plays an important role in global climate change prediction, ecological network observation and simulation, regional air pollution control, and so on. However, the scope, degree, quality, and usability of data sets mastered by different research institutions vary greatly, which is not conductive to the research in related fields. In order to solve this problem, relevant organizations begin to share their own data, which enables researchers to find and reuse relevant data. Combining data from multiple sources can better raise new questions and accelerate the pace of science. Sharing of relevant data can also make the related scientific research more transparent, which helps boost public trust.

In the process of information sharing, on the one hand, the sender of the information only wants the authorized receiver to receive the correct information in order to prevent the information from being maliciously revealed. On the other hand, the recipient of the message wants to verify the sender of the message to prevent the message from being tampered with and the sender from being forged.

In view of all the requirements above, the information sent needs to be confidential and verifiable. The confidentiality needs to be implemented by encryption, and the verification needs to be implemented by signature. Traditionally, the method of first signing and then encrypting is adopted, but it is too computationally intensive resulted in the low efficiency. In order to improve the efficiency on the basis of ensuring confidentiality and verification of messages, Zheng [1] proposed the concept of signcryption, which enables encryption and signature to be performed simultaneously; therefore, encryption and signature functions can be realized by one signcryption.

With the development of communication, the identity-based multireceiver encryption (IMRE) proposal was presented by Baek et al. [2]; this proposal can encrypt multiple recipients’ messages by one calculation. And then, the IMRE schemes [3–5] were proposed successively. Subsequently, on the basis of the scheme in [2], the identity-based multireceiver
signcryption (IMRS) proposal was presented by Duan and Cao [6]. According to this scheme, the sender only needs to perform a signcryption operation to simultaneously send messages to multiple authorized recipients, and each authorized recipient can perform corresponding decryption and verification. The ciphertext received by the receiver does not contain a list of receiver identity information, and the receiver cannot find the relevant information in the ciphertext. It is more effective than one-to-one scheme and is an ideal choice for information sharing. Then, many IMRS schemes [7–14] were proposed. What is more, communicators are paying more and more attention to personal privacy issues. Usually, people intend to conceal their identities when they use multi-receiver signcryption-related devices. In order to realize anonymity, Fan et al. [15] proposed the first anonymous IMRE scheme. However, Wang et al. [16] proposed that the scheme in [15] does not truly achieve that goal. Also, Wang et al. [16] proposed a new anonymity scheme, but Li and Pang [17] proposed that the scheme in [16] still could not completely realize anonymity either. Later, Fan and Tseng [18] proposed an anonymous IMRE scheme and the receiver of the scheme has the capability of identity verification, but the efficiency is not high because of the excessive bilinear pairings operations in the scheme. Then, Pang et al. [19] put forward an absolutely cryptonymous IMRS proposal which can realize the anonymity of the receiver and sender. Gradually, people have found a serious problem that these multireceiver schemes have key escrow problems because they cannot block malicious key generation center (KGC) attacks. If nothing is done to solve this problem, the KGC can view all users’ communications and can pretend to be any user, which is insecure.

Given the above requirements, Al-Riyami and Paterson [20] proposed a certificateless public key cryptosystem. For the user in this system, he and KGC jointly generate his private key, and the system parameters and his private key jointly determine his public key. The scheme in [20] can solve the problem of certificate management in traditional public key cryptography and key escrow in identity-based cryptosystems, so the certificateless multirecipient scheme has become a research hotspot. Influenced by the scheme in [20], the proposals [21–24] were successively put forward. Selvi et al. [25] came up with the earliest certificateless multireceiver signcryption (CMRS) scheme, but it does not have the confidentiality of messages under external attacks and is inefficient due to the use of a large number of bilinear pairing operations (BPO).

Gradually, people began to pursue efficiency on the basis of satisfying the safety function. A cryptonymous certificateless multireceiver encryption (CMRE) proposal was presented by Islam et al. [26]. In order to improve efficiency, the scheme in [26] uses scalar point multiplication operations on elliptic curve cryptography (SPMEOECC), which does not use BPO. Since schemes in [26] can still further improve efficiency, schemes in [27–30] have been proposed successively. The scheme in [27] used the BPO and map-to-point hash function (MTPHF), scheme in [28] and scheme in [30] used lots of SPMEOECC, and scheme in [29] used BPO in the decryption step, all of which greatly limit the efficiency of the scheme. Among them, schemes in [26–30] have no signature function and cannot resist forgery attacks, and schemes in [25, 27–29] did not successfully achieve fair decryption, nor did they implement the verification of part of the private key.

Through the analysis of the schemes, we found that the multirecipient signcryption scheme is required to solve the main problems as below during the communication process:

1. Key escrow problem: malicious KGC can forge encrypted ciphertext and decrypt ciphertext, which dramatically reduces the confidentiality of the message. In order to prevent malicious KGC attacks, we need to solve the key escrow problem

2. The calculation efficiency is not ideal: in the process of encryption or decryption, BPO or a large number of SPMEOECC will result in inefficiency

In view of the requirements of ecological data sharing communication and taking into account the shortcomings of existing communication mechanisms, we propose a CMRS scheme for ecological data sharing. Relevant institutions or researchers can send data to other researchers or related institutions through our program. The communication process is shown in Figures 1 and 2. When multiple researchers or related organizations share relevant ecological data (such as forest ecological data, grassland ecological data, and desert ecological data), they first exchange key with KGC to obtain their own public key and private key; then, they signcrypt the data and send the signcrypted message to the cloud space; finally, the signcrypted data in the cloud space will be forwarded to the authorized recipient and the receiver performs verification and decryption after receiving the message.

The specific features of the scheme are as follows:

1. The scheme realizes the anonymity of the receiver. Not only the attacker cannot obtain the information of the receiver, but also the receiver does not know the information of other receivers

2. The scheme uses a certificateless method and can perform partial private key verification. It not only solves the key escrow problem but also ensures the correctness of some private keys received from KGC

3. The scheme is of high efficiency. Instead of using BPO and exponential operations in the signcryption and decryption phases, the scheme uses the SPMEOECC during encryption and decryption processes and minimizes the number of SPMEOECC, which greatly increases efficiency

Finally, in using the stochastic prediction model, we prove the confidentiality, unforgeability, and recipient anonymity of our CMRS scheme based on the difficult problems.

The rest of the paper is as follows: in the second part, the initial knowledge is described. In the third part, the proposed scheme is described. In addition, in the fourth part, the correctness and security of the scheme were analyzed. Next, in the fifth part, the proposed scheme and the existing scheme are compared in terms of efficiency and functionality. Finally, the sixth section summarizes this paper.
For ease of understanding, the symbols used are listed in Table 1.

2. Preliminaries

We will make an introduction of difficult problems, algorithm model, and security models.

2.1. Difficult Problems. It is specified that \( G_p \) is a cyclic group on ECC, the generator of \( G_p \) is the \( P \), and \( Z_p^* \) is the nonzero multiplication group, which depend on the big prime number \( p \). The elliptic curve discrete logarithm problem (ECDLP) and Diffie-Hellman problem (CDHP) are described below:

(1) CDHP. Known \( (P, xP, yP) \in G_p \), \( x, y \in Z_p^* \), calculating \( xy \) \( P \in G_p \) is CDHP

Definition 1. If the probabilistic polynomial time (PPT) algorithm \( A \) can solve CDHP, the probability advantage is specified as

\[
Adv^\text{CDHP}_A = \Pr[A(P, xP, yP) = xyP].
\] (1)

CDHP hypothesis: we believe that \( Adv^\text{CDHP}_A \) can be ignored for any PPT algorithm \( A \).

(2) ECDLP. Known \( (P, xP) \in G_p \), \( x \in Z_p^* \), calculating \( x \) is ECDLP

Definition 2. If the PPT algorithm \( B \) can solve CDHP, the probability advantage is specified as

\[
Adv^\text{ECDLP}_B = \Pr[B(P, xP) = x].
\] (2)

ECDLP hypothesis: we believe that \( Adv^\text{ECDLP}_B \) can be ignored for any PPT algorithm \( B \).

2.2. Algorithm Model

Definition 3. The algorithm model of this proposal consists of \( \text{SetUpAlgorithm}, \text{SetSecretValueAlgorithm}, \text{ExtractPartialPrivateKeyAlgorithm}, \text{SetPublicAndPrivateKeyAlgorithm}, \text{SignatureAlgorithm} \), and \( \text{DecryptionAlgorithm} \):
SetUpAlgorithm: input $\lambda$, $\lambda$ is the security parameter; KGC executes the algorithm; the system’s master key $s$ and the public parameter $pars$ are generated by KGC; and KGC exposes $pars$ and secretly saves $s$.

SetSecretValueAlgorithm: enter $ID_i$, $ID_i$ is the user’s information; the user executes the algorithm; and the user gets the secret value $x_i$ and the secret value parameter $X_i$.

ExtractPrivateKeyAlgorithm: enter the user’s information $ID_i$, the master key $s$, the secret value parameter $X_i$, and the public parameter $pars$. For the user, KGC uses this algorithm to obtain the partial private key $z_i$ and key generation parameter $W_i$.

SetPublicAndPrivateKeysAlgorithm: input the parameter $pars$ of the system, the information $ID_i$ of the user, the partial private key $z_i$ of the user, the secret value $x_i$, and the key generation parameter $W_i$ and the user uses the algorithm to obtain public key pair $PK_i$ and the private key pair $SK_i$.

SigncryptionAlgorithm: input the system’s public parameter $pars$, receiver information $L = \{ID_1, ID_2 \cdots ID_n\}$, plaintext $M$, the recipient’s public key $PK_i (1 \leq i \leq n)$, the private key $SK_i$ of THE transmitter, and the information IDc of the transmitter. The transmitter initiates the algorithm to obtain the ciphertext $C = Signcryption(pars, M, L, PK_i, SK_i, ID_c)$.

DesigncryptionAlgorithm: enter the public parameter $pars$, the recipient’s private key $SK_i$, the recipient’s information $ID_i$, and ciphertext $C$. The receiver uses the algorithm to gain the plaintext $M = Designcryption(pars, C, SK_i, ID_c)$ and verifies $M$ using the sender’s public key $PK_i$.

2.3. Security Models. The security model required for our scheme consists of information confidentiality, unforgeability, and recipient anonymity. In each security model, there are two types of opponents, called $A_1$ and $A_2$ [20]. $A_1$ does not know the system master key but $A_2$ does, and $A_1$ can replace the user’s public key but $A_2$ cannot. Specific descriptions are as follows:

2.3.1. Information Confidentiality. Information confidentiality means that an attacker cannot successfully decrypt a plaintext message in his or her own attack. In this scheme, information confidentiality is the indistinguishability of certificateless signcryption in the context of selecting ciphertext.
attack and selective multiple ID (IN-CMRS-CA) [25]. We
defined Game1 for $\mathcal{A}_1$’s IN-CMRS-CA and Game2 for $\mathcal{A}_2$’s
IN-CMRS-CA.

Game1: this game is the interaction of opponent $\mathcal{A}_1$ and
challenger $\mathcal{C}$ in the context of IN-CMRS-CA.

Setup. $\mathcal{C}$ performs the corresponding steps for obtaining the
public parameter $\text{pars}$ and the master key $s$, then sends the
$\text{pars}$ to the $\mathcal{A}_1$, and secretly saves $s$. After $\mathcal{A}_1$ receives the
$\text{pars}$, the $\mathcal{A}_1$ outputs a group of target identities $L = \{\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n\}$, where $n$ is a positive integer.

Phase 4. $\mathcal{A}_1$ requires $\mathcal{C}$ to perform flexible queries, and $\mathcal{C}$
performs feedback. The details are the following:

SetSecretValueQuery: $\mathcal{A}_1$ requires $\mathcal{C}$ to run the SetSecret-
ValueQuery on the ID. After the request is received, $\mathcal{C}$ uses the
SetSecretValueAlgorithm to get the user’s secret value, and $\mathcal{C}$
transmits the result to $\mathcal{A}_1$.

ExtractPartialPrivateKeyQuery: $\mathcal{A}_1$ lets $\mathcal{C}$ run the Ex-
tractPartialPrivateKeyQuery against the ID. After the request is
received, $\mathcal{C}$ executes the ExtractPartialPrivateKeyAlgorithm
to gain the partial private key of the user and sends result
to $\mathcal{A}_1$.

SetPublicAndPrivateKeysQuery: $\mathcal{A}_1$ requires $\mathcal{C}$ to run the
SetPublicAndPrivateKeysQuery on the ID. After receiving the
request, $\mathcal{C}$ executes the SetPublicAndPrivateKeysAlgorithm
to get the public and private keys of user and returns them
to $\mathcal{A}_1$.

PublicKeyReplacementQuery: $\mathcal{A}_1$ requires $\mathcal{C}$ to use the
PK$_{\text{ID}}$ to run the PublicKeyReplacementQuery on the ID.
Upon receipt, $\mathcal{C}$ will retain the PK$_{\text{ID}}$ as the new public key.

SigncryptionQuery: $\mathcal{A}_1$ lets $\mathcal{C}$ perform the Signcryption-
Query with a series of information and plaintext $\text{M}$. After
the request is received, $\mathcal{C}$ randomly selects the information
$\text{ID}_1$ to execute the SigncryptionAlgorithm to obtain the
ciphertext $\text{C}$ and transmits $\text{C}$ to $\mathcal{A}_1$.

DesigncryptionQuery: $\mathcal{A}_1$ requires $\mathcal{C}$ to perform the
DesigncryptionQuery for ciphertext $\text{C}$. After the request is
received, $\mathcal{C}$ executes the DesigncryptionAlgorithm to obtain the
plaintext $\text{M}$, after that, verify that $\text{M}$ is compliant, and
returns $\text{M}$ to $\mathcal{A}_1$.

Challenge: $\mathcal{A}_1$ chooses a pair of plaintext $(M_0, M_1)$, the
length of the plaintext is the same and transmits $(M_0, M_1)$
to $\mathcal{C}$. $\mathcal{C}$ selects $\varphi \in \{0, 1\}$; then, $\mathcal{C}$ uses the selected plaintext
$M_\varphi$ to obtain $C*$ and $\mathcal{C}$ transmits $C*$ to $\mathcal{A}_1$.

Phase 5. $\mathcal{A}_1$ wants $\mathcal{C}$ to provide the same query as Phase 4, but $\mathcal{A}_1$
cannot perform the private key portion of SetPubli-
cAndPrivateKeysQuery and ExtractPartialPrivateKeyQuery
on the user who has replaced the public key. $\mathcal{A}_1$ cannot
perform DesigncryptionQuery on ciphertext $\text{C}*$.

Guess: $\mathcal{A}_1$ gives $\varphi^*$, and if $\varphi^* = \varphi$ can be determined, $\mathcal{A}_1$
wins, otherwise, $\mathcal{A}_1$ fails. The probability advantage of $\mathcal{A}_1$
winning is defined as follows:

\[
\text{Adv}^{\text{IN-CMRS-CA}}(\mathcal{A}_1) = |2 \Pr[\varphi^* = \varphi] - 1|.
\]

Definition 6. The probability that any $\mathcal{A}_1$ in the case of IN-
CMRS-CA can obtain the Game1 victory in time $\tau$ can satisfy
$\text{Adv}^{\text{IN-CMRS-CA}}(\mathcal{A}_1) \leq \epsilon$, indicating that the scheme conforms
to $(\tau, \epsilon)$ - IN-CMRS-CA security, $\epsilon$ represents a nonnegligible
probability advantage and $\tau$ represents the time of a poly-
nomial operation.

Game2: this game is the interaction of opponent $\mathcal{A}_2$ and
challenger $\mathcal{C}$ in the context of IN-CMRS-CA. The details are the
following:

Setup. $\mathcal{C}$ performs the corresponding steps for obtaining the
public parameter $\text{pars}$ and the master key $s$, then transmits the
result to the $\mathcal{A}_2$. After successful reception, $\mathcal{A}_2$ gives a
set of target identities $L = \{\text{ID}_1, \text{ID}_2, \text{ID}_3, \ldots, \text{ID}_n\}$, $n$ is a
positive integer.

Phase 7. $\mathcal{A}_2$ makes a series of queries to $\mathcal{C}$ like Phase 4 of
Game1, but $\mathcal{A}_2$ cannot carry out PublicKeyReplacementQu-
ery. $\mathcal{C}$ responds accordingly.

Challenge: $\mathcal{A}_2$ selects a pair of plaintext $(M_0, M_1)$; the
length of the plaintext is the same and transmits $(M_0, M_1)$
to $\mathcal{C}$. $\mathcal{C}$ selects $\varphi \in \{0, 1\}$; then, $\mathcal{C}$ uses the selected plaintext
$M_\varphi$ to obtain $C*$, $\mathcal{C}$ transmits $C*$ to $\mathcal{A}_2$.

Phase 8. $\mathcal{A}_2$ needs $\mathcal{C}$ to perform the same query as Phase 4, but $\mathcal{A}_2$
cannot execute the SetSecretValueQuery for the target
identity $L$, and $\mathcal{A}_2$ cannot run the DesigncryptionQuery
ciphertext $C*$.

Guess: $\mathcal{A}_2$ gives $\varphi^*$, and if $\varphi^* = \varphi$ can be determined, $\mathcal{A}_2$
wins, otherwise, $\mathcal{A}_2$ fails. The probability advantage of $\mathcal{A}_2$
winning is defined as follows:

\[
\text{Adv}^{\text{IN-CMRS-CA}}(\mathcal{A}_2) = |2 \Pr[\varphi^* = \varphi] - 1|.
\]

Definition 9. The probability that any $\mathcal{A}_2$ in the case of IN-
CMRS-CA can obtain the Game2 victory in time $\tau$ can satisfy
$\text{Adv}^{\text{IN-CMRS-CA}}(\mathcal{A}_2) \leq \epsilon$, indicating that the scheme conforms
to $(\tau, \epsilon)$ - IN-CMRS-CA security, $\epsilon$ represents a nonnegligible
probability advantage and $\tau$ represents a time of a poly-
nomial operation.

2.3.2. Unforgeability. The unforgeability of the proposal in this
paper is prescribed as the strong unforgeability of certifi-
cateless signcryption in the context of selecting ciphertext
attacks and optional multiple ID (SU-CMRS-PA) [25].

Game3 and Game4 are used to describe the SU-CMRS-PA
of the opponent $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

Game3: this is the mutual response between $\mathcal{C}$ and $\mathcal{A}_2$
under SU-CMRS-PA. The details are the following:

Setup. This procedure is similar to Setup in Game1.

\[
\text{Adv}^{\text{SU-CMRS-PA}}(\mathcal{A}_1) = |2 \Pr[\varphi^* = \varphi] - 1|.
\]
 Forgery: $\mathcal{A}_1$ uses $L = \{ID_1, ID_2, \ldots, ID_n\}$ and plaintext $M$ to forge ciphertext $C^*$. If any recipient in $L$ can correctly decrypt $C^*$, then $\mathcal{A}_1$ wins, otherwise $\mathcal{A}_1$ fails. In this process, $\text{SigncryptionQuery}$ cannot get $C^*$, other restrictions are similar to Phase 5 in Game1.

Definition 10. The probability that any $\mathcal{A}_1$ in the case of SU-CMRS-PA can obtain the Game3 victory in time $\tau$ can satisfy $\text{Adv}^{\text{SU-CMRS-PA}}(A_1) \leq \varepsilon$, indicating that the scheme conforms to $(\tau, \varepsilon)$-SU-CMRS-PA security, $\varepsilon$ represents a nonnegligible probability advantage and $\tau$ represents the time of a polynomial operation.

$\text{Game4}$: this process is the interaction between $C$ and $\mathcal{A}_2$ under SU-CMRS-PA. The details are the following:

Setup. This procedure is similar to Setup in Game2.

Attack: $\mathcal{A}_2$ performs flexible queries for $C$. The queries are similar to Phase 4 in Game2, after which $C$ responds.

 Forgery: $\mathcal{A}_2$ uses $L = \{ID_1, ID_2, \ldots, ID_n\}$ and plaintext $M$ to forge ciphertext $C^*$. If any recipient in $L$ can correctly decrypt $C^*$, then $\mathcal{A}_2$ wins, otherwise $\mathcal{A}_2$ fails. In this process, $\text{SigncryptionQuery}$ cannot get $C^*$, and the other restrictions are similar to Phase 5 in Game2.

Definition 11. The probability that any $\mathcal{A}_2$ in the case of SU-CMRS-PA can obtain the Game4 victory in time $\tau$ can satisfy $\text{Adv}^{\text{SU-CMRS-PA}}(A_2) \leq \varepsilon$, indicating that the scheme conforms to $(\tau, \varepsilon)$-SU-CMRS-PA security, $\varepsilon$ represents a probabilistic advantage that is not negligible. $\tau$ represents the time of a polynomial operation.

2.3.3. Receiver Anonymity. In this scheme, receiver anonymity is prescribed as the anonymous indistinguishability of certificateless signcryption in the context of selecting ciphertext attacks and selective multiple ID (ANO-CMRS-CA) [26]. Game5 and Game6 each implement ANO-CMRS-CA for $\mathcal{A}_1$ and $\mathcal{A}_2$.

$\text{Game5}$: this process is the interaction between $C$ and $\mathcal{A}_1$ under ANO-CMRS-CA. The details are the following:

Setup. $C$ performs the corresponding steps for obtaining the public parameter $\text{pars}$ and the master key $s$, then transmits the result to $\mathcal{A}_2$, $\mathcal{A}_2$ selects $L = \{ID_0, ID_1\}$ and outputs $L$.

Phase 15. This procedure is similar to Phase 4 in Game2.

 Challenge: $\mathcal{A}_2$ picks identity list $L^* = \{ID_2, ID_3, \ldots, ID_n\}$ and a plaintext $M$, $\mathcal{A}_2$ transmits $L^*$ and $M$ to $C$. $C$ selects $e \in \{0, 1\}$ and forms a ciphertext with a new set of identity list $L^{**} = \{ID_e, ID_2, ID_3, \ldots, ID_n\}$, and $C$ sends $C^*$ to $\mathcal{A}_2$.

Phase 16. This procedure is similar to Phase 5 in Game2.

 Guess: $\mathcal{A}_2$ gives $e^*$, and if $e^* = e$ can be determined, $\mathcal{A}_2$ wins, otherwise, $\mathcal{A}_2$ fails. The probability advantage of $\mathcal{A}_2$ winning is defined as follows:

$$\text{Adv}^{\text{ANO-CMRS-CA}}(A_2) = \left|2 \Pr[e^* = e] - 1\right|. \quad (6)$$

Definition 17. The probability that any $\mathcal{A}_2$ in the case of ANO-CMRS-CA can obtain the Game6 victory in time $\tau$ can satisfy $\text{Adv}^{\text{ANO-CMRS-CA}}(A_2) \leq \varepsilon$, indicating that the scheme conforms to $(\tau, \varepsilon)$-ANO-CMRS-CA security, $\varepsilon$ represents a nonnegligible probability advantage and $\tau$ represents the time of a polynomial operation.

3. The Proposed Scheme

Our scheme includes $\text{SetUpAlgorithm}$, $\text{KeyExtractAlgorithm}$, $\text{SigncryptionAlgorithm}$, and $\text{DesigncryptionAlgorithm}$. The $\text{KeyExtractAlgorithm}$ includes $\text{SetSecretValueAlgorithm}$, $\text{ExtractPartialPrivateKeyAlgorithm}$, and $\text{SetPublicAndPrivateKeyAlgorithm}$.

3.1. Setup Algorithm. KGC performs the corresponding steps for obtaining the public parameter and master keys. The details are the following:

(1) Input $\lambda, \lambda$ is the security parameter and KGC selects a large prime number $p$ on the finite field at random and generates a finite field $F_p$, with $p$ order. Next, a suitable elliptic curve $E_p$ is generated on the domain $F_p$, and an addition cycle group $G_p$ is determined on the elliptic curve $E_p$, and the generator of $G_p$ is $P$.
(2) Select a positive whole number \( s \in Z_p^* \) at random, \( s \) is the system master key and secretly save and calculate the system public key \( P_{pub}, P_{pub} = sP \).

(3) Select five safe and reliable hash function as follows:

\[
\begin{align*}
H_1 &= \{0,1\} \times G_P \times G_P \rightarrow Z_p^*, \\
H_2 &= G_P \times \{0,1\} \rightarrow Z_p^*, \\
H_3 &= G_P \times \{0,1\} \rightarrow Z_p^*, \\
H_4 &= Z_p^* \rightarrow \{0,1\} *, \\
H_5 &= \{0,1\} \times \{0,1\} \times G_P \times G_P \times \{0,1\} * \\
&\quad \times G_P \times Z_p^* \times Z_p^* \times Z_p^* \times \cdots \times Z_p^* \times Z_p^* \rightarrow Z_p^*
\end{align*}
\]

(4) The appropriate symmetric encryption algorithm \( E_k \) and decryption algorithm \( D_k \) are selected at random. \( k \) as a symmetric key.

(5) Output the public parameters \( pars = \{Ep, G_p, E_k, D_k, p, P_{pub}, H_1, H_2, H_3, H_4, H_5\} \).

3.2. Key Extract Algorithm. The user and KGC perform the corresponding steps for obtaining the user’s private and public keys.

(1) SetSecretValueAlgorithm: user ID_i randomly selects \( x_i \in Z_p^* \) as a secret value and keeps it in secret, calculates \( X_i = x_i P \), and sends \( X_i \) to KGC via a secure channel.

(2) ExtractPartialPrivateKeyAlgorithm: after receiving, KGC randomly selects the integer \( w_i \in Z_p^* \), then calculates \( W_i = w_i P \) and \( h_i = H_1 (ID_i, X_i, W_i) \), and then calculates \( y_i = w_i + h_i s \mod q \) and \( z_i = y_i + H_2 (sX_i, ID_i) \mod q \), finally sending \( z_i \) and \( W_i \) to the user ID_i via a secure channel.

(3) SetPublicAndPrivateKeyAlgorithm: after the user ID_i receives \( z_i \) and \( W_i \), it calculates \( y_i = z_i - H_2 (x_i P_{pub}, ID_i) \) and \( Y_i = y_i P \). Then, check if the equation \( y_i P = W_i + h_i P_{pub} \) is established, and if so, the user receives successfully. Otherwise, the reception fails.

The user ID_i, public key pair is \( PK_i = (X_i, Y_i) \), private key pair is \( SK_i = (x_i, y_i) \). PK_i is published by KGC.

3.3. Signcryption Algorithm. The sender uses the public parameter \( pars \), its own private key \( SK_i \), plaintext \( M \), a set of recipient information \( L = \{ID_1, ID_2, \ldots, ID_n\} \), the recipient’s public key \( PK_i, (1 \leq i \leq n) \), and the information \( ID_i \) of a transmitter for signcryption:

(1) Select the integer \( r \in Z_p^* \) at random and then calculate \( R = rP \).

(2) Calculate \( \alpha_i = H_3 \langle r X_i, ID_i \rangle, i = 1, 2, 3, \ldots n \).

(3) Select the whole number \( \xi \) at random, \( \xi \in Z_p^* \), then calculate the formula:

\[
f(x) = \prod_{j=1}^{n}(x - \alpha_j) + f(x) = \prod_{j=1}^{n}(x - \alpha_j) + \xi \mod p = a_0 + a_1 x + \cdots + a_{n-1} x_{n-1} + x_n, a_i \in Z_p^* \tag{8}
\]

(4) Calculate \( k = H_4 (\xi), J = E_k (M || ID) \).

(5) Calculate \( h_i = H_5 (M, ID, X_i, PK_i, J, R, a_0, a_1, \ldots a_n) \), where \( \sigma = x_c + y_c + h_i r \mod q \).

(6) Generate ciphertext \( C = \{J, R, a_0, a_1, \ldots a_n, \sigma\} \) and send the ciphertext to all recipients.

3.4. Designcryption Algorithm. The receiver ID_i decrypts after receiving the ciphertext \( C = \{J, R, a_0, a_1, \cdots a_n, \sigma\} \). Proceed as follows:

(1) Calculate \( ai^* = H_3 \langle x_i, R, ID_i \rangle \).

(2) Calculate \( f (ai^*) \rightarrow \xi \).

(3) Calculate \( k = H_4 (\xi) \) and \( M || ID = D_k (J) \).

(4) Search and obtain \( PK_i \), calculate \( h_i = H_5 (M, ID_i, X_i, PK_i, J, R, a_0, a_1, \cdots a_n) \).

And judge whether \( \sigma P = X_c + Y_c + h_i R \) is established, If successful, the recipient receives the plaintext \( M \) and then exits. Otherwise, the recipient rejects and quits.


4.1. Correctness Analysis

Theorem 18. Confirm that the part private key of the user in KeyExtractAlgorithm is right.

Proof. In order for the user’s partial private key to be verified correctly, \( y_i P = W_i + h_i P_{pub} \) needs to be established. The specific inference of this equation is as follows:

\[
y_i P = (w_i + h_i s) P = w_i P + h_i s P = W_i + h_i P_{pub}. \tag{9}
\]

The establishment of the formula \( y_i P = W_i + h_i P_{pub} \) determines that the verification of the part private key is right in the KeyExtractAlgorithm.

Theorem 19. The DesigncryptionAlgorithm is right.

Proof. The formula \( \sigma P = X_c + Y_c + h_i R \) guarantees the correctness of the DesigncryptionAlgorithm. The specific inference of the formula is as follows:

\[
\sigma P = (x_i + y_i + h_i r) P = x_i P + y_i P + h_i r P = X_c + Y_c + h_i R. \tag{10}
\]
The formula $\sigma P = X_c + Y_c + h_c R$ holds, so it is determined that the DesigncryptionAlgorithm is correct.

4.2. Security Proofs. We have security proofs of the proposal based on the safety model outlined in section 2. Theorem 20 and Theorem 23 guarantee confidentiality of the message, Theorem 26 and Theorem 27 ensure unforgeability, and recipient anonymity is ensured by Theorem 28 and Theorem 31.

**Theorem 20.** There is an opponent $A_1$ under the IN-CMRS-CA, in the polynomial time $\tau$, if $A_1$ wins the Game1 victory with a probability advantage $\epsilon$ that cannot be ignored, ($A_1$ can request up to $q_1$ hash queries $H_i (i = 1, 2, 3, 4, 5)$, $q_2$ Key-Query, SetSecretValueQuery, $q_6$ ExtractPrivateKeyKeyQuery, $q_7$ SetPublicAndPrivateKeysQuery, $q_9$ PublicKeyReplacementQuery, $q_10$ SigncryptionQuery, and $q_12$ DesigncryptionQuery), then within time $\tau$, $A_1$ can request up to $yP + 1$ queries with a probability advantage of $\epsilon$ and opponent $A_1$ makes $\mathcal{C}$ overcome CDHP with a probability advantage of $\epsilon \geq 2(1 - q_d q_3 / 2^k) / q_d q_3$, $\tau_s$ is the time of SPMOOECC.

**Proof.** Under a nonnegligible probability, it is assumed that opponent $A_1$ can attack the IN-CMRS-CA security, the probability advantage is $\epsilon$, and under the stochastic prediction model, $A_1$ requires challenger $C$ to perform queries. Given $(P, xP, yP)$, the time-limited polynomial, $C$ obtains $xyP$ through opponent $A_1$'s interaction to overcome CDHP. $C$ and opponent $A_1$ will interact with the following:

**Setup.** $C$ performs the corresponding steps for obtaining the public parameter $\mathbb{P} = (p, G, H, \mathbb{D}, R, P P, P P = a P, H_1, H_2, H_3, H_4, H_5)$ and master key $\mathcal{M} = \{a \in Z_p^*, \mathcal{A}_1 \}$ and then transmits $\mathbb{P}$ to $A_1$ and save $s$. After the recipient receives it, $A_1$ selects the target identity $L = \{ID_1, ID_2, \ldots, ID_n\}$ and transmits $L$ to $C$, $n$ denotes a positive whole number. $H_1, H_2, H_3, H_4, H_5$ are all random oracles; they are controlled by $C$. The details are the following:

1. $H_1$HashQuery: The tuple $(ID_j, X_j, W_j)$ is taken as input, and $A_1$ requires $C$ to execute $H_1$HashQuery. After receiving, $C$ checks whether the tuple $(ID_j, X_j, W_j)$ exists in list $L_1$. If it exists, $C$ will send $d_j$ to $A_1$. Otherwise, $C$ selects $d_j \in Z_p^*$ at random and sends $d_j$ to $A_j$ and restores the tuple $(ID_j, X_j, W_j, d_j)$ into list $L_1$

2. $H_2$HashQuery: the tuple $(sX_j, ID_j)$ and $(X_j, P P) = (ID_j, d_j)$ are taken as input, and $A_1$ requires $C$ to execute $H_2$HashQuery. After receiving, $C$ checks whether the tuple $(sX_j, ID_j, \theta)$ and $(X_j, P P, ID_j, d_j)$ exist in the list $L_2$. If yes, $C$ will send $\theta$ and $d_j$ to $A_1$. Otherwise, $C$ selects two integers $\theta$ and $d_j \in Z_p^*$ at random and transmits them to $A_1$ and restores the tuple $(sX_j, ID_j, \theta)$ and $(X_j, P P, ID_j, d_j)$ into list $L_2$

3. $H_3$HashQuery: the tuple $(rX_j, ID_j)$ is taken as input, and $A_1$ requires $C$ to execute $H_3$HashQuery. After receiving, $C$ judges whether or not the tuple $(rX_j, ID_j)$ exists in the list $L_1$. If it exists, $C$ will send $a_j$ to $A_1$. Otherwise, $C$ selects $a_j \in Z_p^*$ at random and sends $a_j$ to $A_1$ and restores the tuple $(rX_j, ID_j, a_j)$ into list $L_3$

4. $H_4$HashQuery: the tuple $(\xi)$ is taken as input, and $A_1$ requires $C$ to execute $H_4$HashQuery. After receiving, $C$ judges whether or not the tuple $(\xi, k)$ exists in the list $L_4$. If it exists, $C$ will send $k$ to $A_1$. Otherwise, $C$ selects $k \in Z_p^*$ at random, and sends $k$ to $A_1$, and restores the tuple $(\xi, k)$ into list $L_4$

5. $H_5$HashQuery: The tuple $(M, ID_j, X_j, PK_j, I, R, a_0, a_1, \ldots, a_n)$ is taken as input, and $A_1$ requires $C$ to execute $H_5$HashQuery. After receiving, $C$ judges whether or not the tuple $(M, ID_j, X_j, PK_j, I, R, a_0, a_1, \ldots, a_n, h_j)$ exists in list $L_5$. If it exists, $C$ will send $h_j$ to $A_1$. Otherwise, $C$ selects $h_j \in Z_p^*$ at random, and sends $h_j$ to $A_1$ and restores the tuple $(M, ID_j, X_j, PK_j, I, R, a_0, a_1, \ldots, a_n, h_j)$ into list $L_5$

Phase 21. $A_1$ requires $C$ to perform flexible queries and $C$ performs feedback. The details are the following:

1. KeyQuery: $C$ checks if the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ exists in the list $L_C$. If it exists, $C$ reserves the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$. Otherwise, $C$ does the following:

   a) If $ID_j = ID_0, i = 1, 2, 3, \ldots, n$, $C$ selects two integers $x_j, w_j \in Z_p^*$ at random, sets $X_j = x_j P$ and $SK_j = \left\{x_j, P P, H_1, H_2, H_3, H_4, H_5, X_j, W_j\right\}$, computes $W_j = w_j P, Y_j = x_j P, PK_j = \left\{x_j, Y_j\right\}$ and then updates tuples $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_C$ and $(ID_j, X_j, W_j, d_j)$ in list $L_1$, respectively

   b) If $ID_j \neq ID_0, i = 1, 2, 3, \ldots, n$, $C$ selects two integers $x_j, z_j \in Z_p^*$ at random, sets $X_j = x_j P$, computes $y_j = z_j - H_2 (x_j, P P)$, $Y_j = y_j P, PK_j = \left\{x_j, Y_j\right\}$, $SK_j = \left\{x_j, y_j\right\}$ and then updates tuples $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_C$ and $(ID_j, X_j, W_j, d_j)$ in list $L_1$, respectively

2. SetSecretValueQuery: $A_1$ requires $C$ to execute the SetSecretValueQuery on $ID_j$. After receiving the request, $C$ checks whether there are tuples $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_C$. If it exists, $C$ sends $x_j$ to $A_1$, otherwise, $C$ performs the KeyQuery to generate a tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ and then sends $x_j$ to $A_1$

3. ExtractPartialPrivateKeyQuery: $A_1$ requires $C$ to perform the ExtractPartialPrivateKeyQuery on $ID_j$.
After receiving the request, the details are the followings:

(a) If ID$_j$ = ID$_i$, for i = 1, 2, 3, ..., n, $C$ sends “failure” to $A_1$

(b) If ID$_j$ = ID$_i$, for i = 1, 2, 3, ..., n, $C$ judges whether or not the tuple (ID$_j$, SK$_j$, PK$_j$, w$_j$, x$_j$, z$_j$) exists in the list $L_C$, and if so, $C$ sends z$_j$ to $A_1$, otherwise, $C$ performs KeyQuery to generate tuple (ID$_j$, SK$_j$, PK$_j$, x$_j$, z$_j$) and sends z$_j$ to $A_1$

(4) SetPublicAndPrivateKeysQuery: $A_1$ requires $C$ to execute the public key portion of SetPublicAndPrivateKeysQuery on the ID$_j$. After the recipient receives it, $C$ judges whether or not the tuple (ID$_j$, SK$_j$, PK$_j$, x$_j$, z$_j$) exists in list $L_C$. If it exists, $C$ sends PK$_j$ to $A_1$, otherwise $C$ performs KeyQuery to generate a tuple (ID$_j$, SK$_j$, PK$_j$, x$_j$, z$_j$) and then send PK$_j$ to $A_1$

$A_1$ requires $C$ to execute the private key portion of SetPublicAndPrivateKeysQuery on the ID$_j$. After the recipient receives it, $C$ responds:

(a) If ID$_j$ = ID$_i$, for i = 1, 2, 3, ..., n, $C$ sends “failure” to $A_1$

(b) If ID$_j$ = ID$_i$, for i = 1, 2, 3, ..., n, $C$ judges whether or not the tuple (ID$_j$, SK$_j$, PK$_j$, x$_j$, z$_j$) exists in list $L_C$, and if so, $C$ sends SK$_j$ to $A_1$, otherwise, $C$ uses KeyQuery to generate tuple (ID$_j$, SK$_j$, PK$_j$, x$_j$, z$_j$) and sends SK$_j$ to $A_1$

(5) PublicKeyReplacementQuery: $A_1$ requires $C$ to perform a PublicKeyReplacementQuery on ID$_j$ with PK$_j'. After the recipient receives it, $C$ queries the tuple (ID$_j$, SK$_j$, PK$_j', w_j, x_j, z_j$) in list $L_C$, and uses PK$_j'$ instead of PK$_j$. Next, $C$ restores the tuple (ID$_j$, SK$_j$, PK$_j', x_j, z_j$) in list $L_C$

(6) SigncryptionQuery: $A_1$ requires $C$ to perform the SigncryptionQuery for plaintext M and information ID$_p$. After receiving the request, $C$ determines if ID$_j$ = ID$_i$ is correct, i = 1, 2, 3, ..., n. If the expression is correct, $C$ executes the SetPublicAndPrivateKeysQuery to generate the private key SK$_j$ and ciphertext C, and then sends C to $A_1$ if not, $C$ implement s the corresponding operation

(a) Select r $\epsilon$ $Z^*_p$ at random and calculate Select

\[
R = rP,
\]
\[
rX_j = rx_jP,
\]
\[
a_j = H_3(rX_j, ID_j), j = 1, 2, 3, ..., n;
\]
\[
\xi \in Z^*_p
\]
\[
f(x) = \prod_{j=1}^{n}(x-a_j) + \xi (mod p) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} + x^n, a_e \epsilon Z^*_p
\]

(b) Calculate $k = H_4(\xi), j = E_k(\|ID_d), h = H_5(M, ID_d, X_d, PK_d, j, R, a_0, a_1, \ldots, a_{n-1}, \sigma = x_i + y_i + h \mod q$;

(c) Select $z \epsilon Z^*_p$ at random

(d) Output the ciphertext $C = (j, R, a_0, a_1, \ldots, a_{n-1}, \sigma)$ to $A_1$

(7) DesigncryptionQuery: $A_1$ requires $C$ to perform a DesigncryptionQuery for ciphertext C. After receiving the request, $C$ selects ID$_d$ at random and verifies whether ID$_j$ = ID$_d$ is true, i = 1, 2, 3, ..., n. If the formula is true, $C$ sends “failure” to $A_1$. Otherwise, $C$ will perform the corresponding operation:

(a) Search for the tuple (ID$_d$, SK$_d$, PK$_j$, x$_j$, z$_j$) in list $L_C$ to obtain SK$_d$ and compute $a_j = H_3(x_jR, ID_d)$

(b) Calculate $f(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} + x^n$, generate $\xi$ with $a_j$ and $f(x)$.

(c) Calculate $k = H_4(\xi), M||ID_d = D_k(j)$

(d) Consider whether the equation $\sigma P = X_j + Y_j + hR$ can be established. If it is established, $C$ sends $M$ to $A_1$. If not, $C$ sends “failure” to $A_1$

Challenge: $A_1$ selects the plaintext $(M_0, M_1)$ at random, they are equal in length, and they are transmitted to $C$. After receiving, $C$ selects a bit q $\epsilon \{0, 1\}$ at random, $C$ uses $M_q$ to get the ciphertext $C*$, and $M_0$ is the selected plaintext. The details are the following:

(a) Stipulate $R = yPK_i, Rx_i = y(PK_i + P_{pub})$ and $a_i = H_3(rX_i), i = 1, 2, 3, ..., n$.

(b) Select $\xi \epsilon Z^*_p$ at random and define the formula:

\[
f(x) = \prod_{j=1}^{n}(x-a_j) + \xi (mod p) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} + x^n, a_e \epsilon Z^*_p
\]

(c) Calculate $k = H_4(\xi), J* = E_k(\|ID_d)$ and $h = (M, ID_d, X_d, PK_d, j, R, a_0, a_1, \ldots, a_{n-1})$

(d) Select $z \epsilon Z^*_p$ at random
Phase 22. $A_1$ requires $C$ to perform the same queries as Phase 4, but $A_1$ cannot run the DesigncryptQuery against $C$. 

Guess: $A_1$ gives a $q^*$ at random. If $q^* = q$, then $A_1$ wins, $C$ outputs CDHP's solution $xyP = rX_i - R$. Otherwise, $C$ outputs "failure." 

From the above discussion, it can be concluded that the $H_i$Hash provides a valid ciphertext during the DesigncryptQuery; therefore, the probability of ciphertext being denied is no more than $q_d/2^k$. During the attack process, since $A_1$ requires $C$ to execute the $q_d$ DesigncryptQuery, the probability of decryption success is $\epsilon_d \geq \epsilon - q_d/2^k$. The $H_i$Hash satisfies CDHP during the guessing process, so the probability that $xyP$ is correctly calculated by $C$ is at least $\epsilon_d = 2/nq_d$. In the running time $\tau \leq \tau + (2q_d + 3q_d) O(\tau)$, $C$'s interaction with opponent $A_1$ makes $C$ overcome CDHP, and the probability advantage is $\epsilon \geq \epsilon_{\epsilon_d \geq 2(\epsilon - q_d/2^k)}/nq_d \tau$, $\tau$ is the time of the SPMOEOCC.

**Theorem 23.** There is an opponent $A_2$ under the IN-CLMS-CA. In the polynomial time $\tau$, if the $A_2$ wins the Game2 victory with a probability advantage $\epsilon$ that cannot be ignored, ($A_2$ can request up to $q_2$ hash queries $H_i (i = 1, 2, 3, 4, 5)$, $q_2$, KeyQuery, $q_2$, SetSecretValueQuery, $q_2$, ExtractPrivateKeyQuery, $q_2$, SetPublicAndPrivateKeysQuery, $q_2$, SigncryptQuery, and $q_2$, DesigncryptQuery), then within time $\tau \leq \tau + (3q_2 + 3q_2) O(\tau)$, challenger $C$'s interaction with opponent $A_2$ makes $C$ overcome CDHP with a probability advantage of $\epsilon \geq 2(\epsilon - q_d/2^k)/nq_d \tau$, $\tau$ is the time of the SPMOEOCC.

Proof. Under a nonnegligible probability, it is assumed that the opponent $A_2$ can attack the IN-CMRS-CA security, the probability advantage is $\epsilon$, and under the stochastic prediction model, $A_2$ requires challenger $C$ to perform a series of queries. Given $(P, xP, yP)$, in the time-limited polynomial, $C$ obtains $xyP$ through the opponent's $A_2$ interaction to overcome CDHP. $C$ and opponent $A_2$ will interact with the following:

Set up. $C$ performs the corresponding steps for obtaining the public parameter $\text{par} = (E_p, G_p, E_k, D_k, P, P, K = xP, P_p, P_p, H_1, H_2, H_3, H_4, H_5)$ and master key $s \in Z_p^\ast$, and then returns the result to $A_2$, where $x \in Z_p^\ast$. After receiving the result, $A_2$ outputs $L = \{ID_1, ID_2, ID_3, \ldots, ID_n\}$, where $n$ is a positive integer. $H_1, H_2, H_3, H_4,$ and $H_5$ are random predictions controlled by $C$, the interactions between $A_2$ and $C$ is similar to Setup in Theorem 20.

Phase 24. $A_2$ requires $C$ to perform flexible queries and $C$ performs feedback. The details are the following:

1. **KeyQuery**: $C$ checks if there is a tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ in the list $L_2$. If it exists, $C$ reserves the tuple. Otherwise, $C$ will perform related operations as follows:

(a) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ selects two integers $x_j, w_j \in \mathbb{Z}_p^\ast$ at random and computes $W_j = w_j, y_j = y_j + H_1(ID_j, X_j, W_j) s \bmod q_j$, $Y_j = y_j P, PK_j = (X_j, Y_j)$, and then updates tuples $(ID_j, SK_j, P K_j, x_j, z_j)$ in list $L_2$ and $(ID_j, X_j, W_j, h_j)$ in list $L_1$, respectively, where $X_j = x_j P$ and $SK_j \leftarrow \perp$.

(b) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ selects two integers $x_j, z_j \in \mathbb{Z}_p^\ast$ at random, and sets $X_j = x_j P$, computes $y_j = z_j - H_2(x_j, P_p, ID_j)$, $Y_j = y_j P, PK_j = (X_j, Y_j)$, and then updates tuples $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_2$ and $(ID_j, X_j, W_j, h_j)$ in list $L_1$, respectively.

(2) **SetSecretValueQuery**: $A_2$ requires $C$ to execute SetSecretValueQuery for $c$. Upon receipt of the request, $C$ executes as follows:

(a) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ sends "failure" to $A_2$.

(b) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ judges whether or not the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ exists in list $L_2$. If yes, $C$ returns $x_j$ to $A_2$. Otherwise, $C$ uses KeyQuery to generate the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ and sends $x_j$ to $A_2$.

(3) **ExtractPartialPrivateKeyQuery**: $A_2$ requires $C$ to perform ExtractPartialPrivateKeyQuery against $ID_j$. After receiving the request, $C$ checks whether there is a tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_2$. If it exists, $C$ sends $z_j$ to $A_2$. Otherwise, $C$ runs KeyQuery to generate tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ and sends $z_j$ to $A_2$.

(4) **SetPublicAndPrivateKeysQuery**: $A_2$ requires $C$ to perform the public key portion of SetPublicAndPrivateKeysQuery against $ID_j$. After receiving the request, $C$ checks whether there is a tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ in list $L_2$. If it exists, $C$ sends PK to $A_2$. Otherwise, $C$ runs KeyQuery to generate tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ and sends PK to $A_2$.

$A_2$ requires $C$ to perform the private key portion of SetPublicAndPrivateKeysQuery against $ID_j$. After the recipient receives it, $C$ responded with the following response:

(a) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ sends "failure" to $A_2$.

(b) If $ID_j = ID_i, i = 1, 2, 3, \ldots, n$, $C$ judges whether or not the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ exists in list $L_2$. If yes, $C$ returns SK to $A_2$. Otherwise, $C$ runs the KeyQuery to generate the tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ and sends SK to $A_2$.

(5) **SigncryptQuery**: this section is similar to the SigncryptQuery in Theorem 20.
(6) DesigncryptionQuery: this section is similar to the DesigncryptionQuery in Theorem 20

**Challenge:** $\mathcal{A}_2$ selects plaintext $(M_0, M_1)$ at random, they are equal in length, and transmits the result to $\mathcal{C}$. After receiving the result, $\mathcal{C}$ selects $\varphi \in \{0,1\}$ at random and obtains the ciphertext $C\ast$ generated by $M_\varphi$:

(a) Stipulate $R = y(PK_i + Y), rX_i = y(PK_i + P_{pub})$ and $a_i = H_3(rX_i, ID_i), i = 1, 2, 3, \ldots, n, Y = K + P_{pub}$

(b) Select $\xi \in Z_p^\ast$ at random and define the formula:

$$f(x) = \sum_{j=1}^{n}(x - a_j) + \xi(\mod \ p) = a_0 + a_1x + \ldots + a_{n-1}x_{n-1} + x_n, a_0 \in Z_p^\ast$$

(c) Calculate $k = H_3(\xi), J^\ast = E_2(M_\varphi || ID_i)$ and

$$h = H_5(m, ID_i, X_i, Y_i, J, R, a_0, a_1, \ldots, a_n);$$

(d) Select $z \in Z_p^\ast$ at random

(e) Output the ciphertext $C\ast = (J^\ast, R, z, h, a_0, a_1, \ldots, a_{n-1})$ to $\mathcal{A}_2$

**Phase 25.** $\mathcal{A}_2$ requires $\mathcal{C}$ to perform the same queries as Phase 4, but $\mathcal{A}_2$ cannot run the DesigncryptionQuery against $C\ast$.

**Guess:** $\mathcal{A}_2$ gives a $\varphi^\ast$ at random. If $\varphi^\ast = \varphi$, then $\mathcal{A}_2$ wins, $\mathcal{C}$ outputs CDHP's solution $xyP = R - rX_i$, if not, $\mathcal{C}$ outputs “failure.”

From the above discussion, it can be concluded that the $H_i, hash$ provides a valid ciphertext during the DesigncryptionQuery, therefore, the probability of ciphertext being denied is no more than $q_2/2^k$. During the attack process, since $\mathcal{A}_2$ requires $\mathcal{C}$ to execute the $q_2$ DesigncryptionQuery, the probability of $\mathcal{C}$ decryption success is $\varepsilon_a \geq \varepsilon - q_2/2^k$. The $H_i, hash$ satisfies CDHP during the guessing process, so the probability that $xyP$ is correctly calculated by $\mathcal{C}$ is at least $\varepsilon_a = 2/nq_1$. In the running time $\tau \leq \tau + (3q_2 + 3q_3)O(\tau), \mathcal{C}$'s interaction with opponent $\mathcal{A}_2$ makes $\mathcal{C}$ overcome CDHP, and the probability advantage is $\tau_i(\tau) \geq 2(\varepsilon - q_2/2^k)/nq_1, \tau_i$ is the time of the SPMOOECC.

**Theorem 26.** There is an opponent $\mathcal{A}_1$ under the SU-CMRS-PA. In the polynomial time $\tau$, if the $\mathcal{A}_1$ wins the Game3 victory with a probability advantage that cannot be ignored (i.e., can obtain queries similar to that obtained by $\mathcal{A}_2$ in Theorem 20), then within time $\tau \leq \tau + (2q_2 + 2q_3)O(\tau)$, challenger $\mathcal{C}$ can overcome CDHP by interacting with opponent $\mathcal{A}_1$ with a probability advantage of $\varepsilon \geq (\varepsilon - q_2/2^k)/2, \tau_i$ is the time of the SPMOOECC.

**Proof.** Under a nonnegligible probability, it is assumed that the opponent $\mathcal{A}_1$ can attack the SU-CMRS-PA security, the probability advantage is $\varepsilon$, under the stochastic prediction model, $\mathcal{A}_1$ requires challenger $\mathcal{C}$ to perform a series of queries. Given $(P, xP, yP)$, in the time-limited polynomial, $\mathcal{C}$ obtains $xyP$ through the opponent $\mathcal{A}_1$ interaction to overcome CDHP. $\mathcal{C}$ and opponent $\mathcal{A}_1$ will interact with the following:

**Setup.** The process is similar to Setup in Theorem 20.

**Attack:** $\mathcal{A}_1$ requires $\mathcal{C}$ to perform flexible queries similar to Phase 4 in Theorem 20.

**Forgery:** $\mathcal{A}_1$ falsifies ciphertext $C^\ast = (J, R, z, h, a_0, a_1, \ldots, a_n)$ using $L \equiv \{ID_1, ID_2, \ldots, ID_n\}$ and a plaintext $M$. If the formula $aP = X_i + Y_i + h_iR$ can be established, $C \ast$ faked successfully, defining $PK_i = y^{-1}PK_i$ and $rX_i = y(PK_i + P_{pub}), \mathcal{C}$ calculates $rX_i = PK_i + xyP$, and outputs $xyP = rX_i - PK_i, xyP$ is the CDHP’s solution. Otherwise, $\mathcal{C}$ returns “failure.”

From the above discussion, it is concluded that the probability of success of the $q_2$ SigncryptionQuery is at least $\varepsilon_a = \varepsilon - q_2/2^k$, calculates $xyP$ in the forgery process, and the probability of $xP$ correct is at least $\varepsilon_a = 1/2$. Therefore, $\mathcal{C}$ can overcome CDHP by interacting with opponent $\mathcal{A}_1$ within the running time $\tau \leq \tau + (2q_2 + 2q_3)O(\tau), \mathcal{C}$'s probability advantage is $\varepsilon \geq (\varepsilon - q_2/2^k)/2, \tau_i$ is the time of the SPMOOECC.

**Theorem 27.** There is an opponent $\mathcal{A}_2$ under the SU-CMRS-PA. In the polynomial time $\tau$, if the $\mathcal{A}_2$ wins the Game4 victory with a probability advantage $\varepsilon$ that cannot be ignored (i.e., can obtain queries similar to that obtained by $\mathcal{A}_2$ in Theorem 23), then within time $\tau \leq \tau + (3q_2 + 2q_3)O(\tau)$, challenger $\mathcal{C}$ can overcome CDHP by interacting with opponent $\mathcal{A}_1$ with a probability advantage of $\varepsilon \geq (\varepsilon - q_2/2^k)/2, \tau_i$ is the time of the SPMOOECC.

**Proof.** Under a non-negligible probability, it is assumed that the opponent $\mathcal{A}_2$ can attack the SU-CMRS-PA security, the probability advantage is $\varepsilon$, under the stochastic prediction model, $\mathcal{A}_2$ requires challenger $\mathcal{C}$ to perform a series of queries. Given $(P, xP, yP)$, in the time-limited polynomial, $\mathcal{C}$ obtains $xyP$ through the opponent $\mathcal{A}_2$ interaction to overcome CDHP. The details are the following:

**Setup.** The process is similar to Setup in Theorem 23.

**Attack:** $\mathcal{A}_2$ requires $\mathcal{C}$ to perform flexible queries similar to Phase 4 in Theorem 23.

**Forgery:** $\mathcal{A}_2$ falsifies ciphertext $C^\ast = (J, R, z, h, a_0, a_1, \ldots, a_n)$ using $L \equiv \{ID_1, ID_2, \ldots, ID_n\}$ and a plaintext $M$. If the formula $aP = X_i + Y_i + h_iR$ can be established, $C^\ast$ faked successfully, defining $PK_i = y^{-1}PK_i$ and $rX_i = y(PK_i + P_{pub}), \mathcal{C}$ calculates $rX_i = PK_i + xyP$, and outputs $xyP = rX_i - PK_i, xyP$ is the CDHP’s solution. Otherwise, $\mathcal{C}$ returns “failure.”

From the above discussion, it is concluded that the probability of success of the $q_2$ SigncryptionQuery is at least $\varepsilon_a = \varepsilon - q_2/2^k$, calculates $xyP$ in the forgery process, and the
probability of $xyP$ correct is at least $\varepsilon_g = 1/2$. Therefore, in the running time $\tau \leq \tau + (3q_{A} + 3q_{i})O(\tau_{i})$, $C^*$'s interaction with opponent $\mathcal{A}_1$ makes $C^*$ overcome CDHP, and the probability advantage is $\varepsilon \geq \varepsilon_g\varepsilon_{s}\varepsilon_{z}\varepsilon_{\text{adv}}$. The time of the SPMOOECC.

**Theorem 28.** There is an opponent $\mathcal{A}'_1$ under the ANO-CMRS-CA. In the polynomial time $\tau$, if the $\mathcal{A}'_1$ wins the Game6 victory with a probability advantage $\varepsilon$ that cannot be ignored ($\mathcal{A}_1$ can obtain queries similar to that obtained by $\mathcal{A}'_1$ in Theorem 20), within time $\tau \leq \tau + (2q_{A} + 3q_{i})O(\tau_{i})$, challenger $C^*$ can overcome CDHP by interacting with opponent $\mathcal{A}'_1$ with a probability advantage of $\varepsilon \geq (\varepsilon - q_{ds}/2^k)/nq_{p}$, $\tau_{s}$ is the time of the SPMOOECC.

**Proof.** Under a non-negligible probability, it is assumed that the opponent $\mathcal{A}'_1$ can attack the ANO-CMRS-CA security, the probability advantage is $\varepsilon$, under the stochastic prediction model, $\mathcal{A}'_1$ requires challenger $C^*$ to perform a series of queries. Given $(P, xP, yP)$, in the time-limited polynomial, $C^*$ obtains $xyP$ through the opponent $\mathcal{A}'_1$ interaction to overcome CDHP. The details are the followings:

**Setup.** $C^*$ execution of this algorithm is used to generate the public parameter $\{E_{p}, E_{d}, D_{a}, P, P_{pub} = xP, H_{1}, H_{2}, H_{3}, H_{4}, H_{5}\}$ and master key $s = a \in Z_{p'}^*$, then sends the $\{a\}$ to the $\mathcal{A}'_1$ and secretly saves $s$. After the $\mathcal{A}'_1$ receives the $\{a\}$, the $\mathcal{A}'_1$ outputs $L = \{ID_{0}, ID_{1}\}$, where $n$ is a positive integer. $H_{1}, H_{2}, H_{3}, H_{4}, H_{5}$ are all random oracles; they are controlled by $C^*$. The interaction is similar to Setup in Theorem 20, and the two sides of the interaction are $\mathcal{A}'_1$ and $C^*$.

**Phase 29.** $\mathcal{A}'_1$ makes a series of queries to $C^*$ like Phase 4 in Theorem 20.

**Challenge:** $\mathcal{A}'_1$ selects the target identifier $L^* = \{ID_{2}, ID_{3}, \cdots, ID_{n}\}$ and plaintext $M$, and sends $L^* \leftrightarrow M$ to $C^*$. After receiving, $C^*$ selects a bit $e \in \{0, 1\}$ and uses the new target identifier $L^* = \{ID_{e}, ID_{2}, ID_{3}, \cdots, ID_{n}\}$ to obtain the ciphertext $C^*$:

(a) Calculate $R = y(PK_{e}), rX_{i} = y(PK_{e} + P_{pub})$ and $a_{j} = H_{1}(rX_{i}, ID_{j})$

(b) Select $\xi \in Z_{p}^*$ at random and define the formula:

$$f(x) = \prod_{j=1}^{n}(x - a_{j} + \xi(mod p)) = a_{0} + a_{1}x \cdots + a_{n-1}x_{n-1} + x_{n}, a_{i} \in Z_{p}^*$$

(16)

(c) Calculate $k = H_{1}(\xi), J^* = E_{k}(M_{e} || ID_{1})$ and $h^* = H_{4}(M_{e}, ID_{e}, Y, I, R, a_{0}, a_{1}, \cdots, a_{n})$

(d) Select $z \in Z_{p}^*$ at random

(e) Output the ciphertext $C^* = (J^* \leftrightarrow R, z, h, a_{0}, a_{1}, \cdots, a_{n-1})$ to $\mathcal{A}'_1$

**Phase 30.** $\mathcal{A}'_1$ requires $C^*$ to execute the queries, and the contents are similar to Phase 5, but $\mathcal{A}'_1$ cannot run the DesigncationQuery against $C^*$.

**Guess:** $\mathcal{A}'_1$ gives a $\varepsilon$ at random. If $\varepsilon = \varepsilon_{s}$, then $\mathcal{A}'_1$ wins, $C^*$ outputs CDHP’s solution $xyP = rX_{i} - R$. If not, $\varepsilon$ outputs “failure.”

From the above discussion, it can be concluded that the $H_{2}$ hash provides a valid ciphertext during the DesigncationQuery, therefore, the probability of ciphertext being denied is no more than $q_{ds}/2^k$. During the attack process, since $\mathcal{A}'_1$ requires $C^*$ to execute the $q_{ds}$ DesigncationQuery, the probability of $C^*$ decryption success is $\varepsilon_{s} \geq \varepsilon - q_{ds}/2^k$. The $H_{2}$ hash satisfies CDHP during the guessing process, so the probability that $xyP$ is correctly calculated by $C^*$ is at least $\varepsilon_{s} = 1/nq_{p}$, $C^*$ can solve CDHP by interacting with opponent $\mathcal{A}'_1$ in the running time $\tau \leq \tau + (2q_{A} + 3q_{i})O(\tau_{i})$, and the probability advantage is $\varepsilon \geq \varepsilon_{s} \geq \varepsilon - q_{ds}/2^k/nq_{p}$, $\tau_{s}$ is the time of the SPMOOECC.

**Theorem 31.** There is an opponent $\mathcal{A}'_2$ under the ANO-CMRS-CA. In the polynomial time $\tau$, if the $\mathcal{A}'_2$ wins the Game6 victory with a probability advantage $\varepsilon$ that cannot be ignored ($\mathcal{A}_1$ can obtain queries similar to that obtained by $\mathcal{A}'_2$ in Theorem 23.), then within time $\tau \leq \tau + (3q_{A} + 3q_{i})O(\tau_{i})$, challenger $C^*$ can overcome CDHP by interacting with opponent $\mathcal{A}'_2$ with a probability advantage of $\varepsilon \geq \varepsilon - q_{ds}/2^k/nq_{p}$, $\tau_{s}$ is the time of the SPMOOECC.

**Proof.** Under a non-negligible probability, it is assumed that the opponent $\mathcal{A}'_2$ can attack the ANO-CMRS-CA security, the probability advantage is $\varepsilon$, under the stochastic prediction model, $\mathcal{A}'_2$ requires challenger $C^*$ to perform a series of queries. Given $(P, xP, yP)$, in the time-limited polynomial, $C^*$ obtains $xyP$ through the opponent $\mathcal{A}'_2$ interaction to overcome CDHP. The details are the followings:

**Setup.** $C^*$ performs the corresponding steps for obtaining the public parameter $\{E_{p}, E_{d}, E_{k}, D_{a}, P, K = xP, P_{pub}, H_{1}, H_{2}, H_{3}, H_{4}, H_{5}\}$ and master key $s \in Z_{p}^*$ and then returns the result to $\mathcal{A}'_2$, where $x \in Z_{p}^*$. After receiving the result, $\mathcal{A}'_2$ outputs $L = \{ID_{0}, ID_{e}\}, H_{1}, H_{2}, H_{3}, H_{4}, H_{5}$ are random predictions controlled by $C^*$. The interaction is similar to Setup in Theorem 20, and the two sides of the interaction are $\mathcal{A}'_2$ and $C^*$.

**Phase 32.** $\mathcal{A}'_2$ makes a series of queries to $C^*$ like Phase 4 in Theorem 23.

**Challenge:** $\mathcal{A}'_2$ selects the target identifier $L^* = \{ID_{2}, ID_{3}, \cdots, ID_{n}\}$ and plaintext $M$ and sends $L^* \leftrightarrow M$ to $C^*$. After receiving, $C^*$ selects a bit $e \in \{0, 1\}$ and uses the new target identifier $L^{**} = \{ID_{e}, ID_{2}, ID_{3}, \cdots, ID_{n}\}$ to obtain the ciphertext $C^*$:

(a) Calculate $R = y(PK_{e} + Y), rX_{i} = y(PK_{e} + P_{pub})$ and $a_{j} = H_{3}(rX_{i}, ID_{j})$, $Y = K + P_{pub}, i = e, 2, 3, \cdots, n$
Table 2: Comparison of functions.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Decryption fairness</th>
<th>Receiver anonymity</th>
<th>Partial private key verifiability</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selvi et al. [25]</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Hung et al. [27]</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Zhu et al. [29]</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Win et al. [30]</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Our program</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 3: Symbols’ definition.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Symbols’ definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_m$</td>
<td>Time of a modular multiplication</td>
</tr>
<tr>
<td>$T_{pm}$</td>
<td>Time of a scalar point multiplication on ECC operation, $T_{pm} = 29T_m$.</td>
</tr>
<tr>
<td>$T_{pa}$</td>
<td>Time of a point addition on ECC operation, $T_{pa} \approx 0.127T_m$.</td>
</tr>
<tr>
<td>$T_b$</td>
<td>Time of a bilinear pairing operation, $T_b \approx 87T_m$.</td>
</tr>
<tr>
<td>$T_{t}$</td>
<td>Time of modular inversion operation, $T_{t} \approx 11.6T_m$.</td>
</tr>
<tr>
<td>$T_{be}$</td>
<td>Time of a bilinear pairing exponentiation operation, $T_{be} \approx 43.5T_m$.</td>
</tr>
<tr>
<td>$T_h$</td>
<td>Time of a map-to-point hash function operation, $T_h \approx 29T_m$.</td>
</tr>
</tbody>
</table>

(b) Select $\xi \in Z_p^*$ at random and define the formula:

$$f(x) = \prod_{j=1}^{n} (x - a_j) + \xi (\text{mod} \ p) = a_0 + a_1 x \cdots + a_{n-1} x_{n-1} + x_n, a_i \in Z_p^*$$

(c) Calculate $k = H_3(\xi), J^* = E_k(M, \|ID_s\) and h * = H_4(M, ID_s, X_p, Y_p, J, R, a_0, a_1, \cdots, a_n)$

(d) Select $z \in Z_p^*$ at random

(e) Output the ciphertext $C^* = (J^*, R, z, h, a_0, a_1, \cdots, a_{n-1})$ to $\mathcal{A}_2$

Phase 33. $\mathcal{A}_2$ requires $\mathcal{C}$ to perform the same queries as Phase 5, but $\mathcal{A}_2$ cannot run the DesigncryptionQuery against $C^*$.

Guess: $\mathcal{A}_2$ gives a $e^*$ at random. If $e^* = e$, then $\mathcal{A}_2$ wins, $\mathcal{C}$ outputs CDHP’s solution $xyP = R - rX_i$. If not $\mathcal{C}$ outputs “failure.”

From the above discussion, it can be concluded that the $H_3$ hash provides a valid ciphertext during the DesigncryptionQuery; therefore, the probability of ciphertext being denied is no more than $q_d/2^b$. During the attack process, since $\mathcal{A}_2$ requires $\mathcal{C}$ to execute the $q_d$ DesigncryptionQuery process, the probability of $\mathcal{C}$’s decryption success is $\epsilon_d \geq \epsilon - q_d/2^b$. The $H_3$ hash satisfies CDHP during the guessing process, so the probability that $xyP$ is correctly calculated by $\mathcal{C}$ is at least $\epsilon_g = 1/nq_3$. In the running time $t^* \leq t + (3q_d + 3q_d)O(\tau), \mathcal{C}$’s interaction with opponent $\mathcal{A}_2$ makes $\mathcal{C}$ overcome CDHP, and the probability advantage is $\epsilon^* \geq \epsilon_d\epsilon_g \geq (\epsilon - q_dq_3/2^b)/nq_3$, $\tau_i$ is the time of the SPMOOECC.

5. Efficiency Analysis and Functional Comparison

Schemes in [25, 27, 29, 30] and our scheme are based on certificateless signcryption. To reflect the superiority of our scheme, we will compare our scheme with them in terms of functions and efficiency.

5.1. Functional Comparison. We have compared the functions of our scheme and the functions of schemes in [25, 27, 29, 30], as shown in Table 2.

From the graph, we can get the following results, schemes in [25, 27, 29, 30] do not satisfy decryption fairness and do not guarantee that each recipient has the same chance of decryption. Schemes in [25, 29, 30] are not anonymous to the recipient, which may result in the disclosure of the recipient information, greatly reducing the encryption effect. Schemes in [25, 27, 29] cannot implement the verifiability of partial private key, which means they can be attacked by malicious KGC. Schemes in [27, 29, 30] have no signature function and cannot prevent an attacker from sending an unreliable messages.

5.2. Efficiency Analysis. For the sake of comparison, we have defined some symbols to represent the calculation of different operations in the encryption process and the decryption process, and the corresponding data is taken from [26]. We only consider the calculations defined in Table 3, because what is shown in Table 3 is the key to computational performance.

With the integration of encryption and decryption processes, as shown in Table 4, Figures 3 and 4, our scheme is
better than the schemes in [25, 27, 29, 30], because the scheme in [25] and scheme in [29] use the BPO, scheme in [27] uses the BPO and MTPHF, and scheme in [30] uses lots of SPMOOECC. In this way, these operations all cause a lot of overheads, making the scheme less efficient. Our scheme also uses SPMOOECC, but we attempt to reduce the number of SPMOOECC. Therefore, the performance of our scheme is superior to the scheme in [25, 27, 29, 30].

Table 4: Comparison of efficiency.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Signryption/encryption</th>
<th>Decryption/designryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selvi et al. [25]</td>
<td>$(n + 1)T_{pm} + T_i + (n + 1)T_{be} = (159.5n + 171.1)T_m$</td>
<td>$T_{pm} + T_{pa} + 2T_b + 2T_{be} = 246.62T_m$</td>
</tr>
<tr>
<td>Hung et al. [27]</td>
<td>$(n + 1)T_{pm} + nT_{be} + nT_{b} + nT_{h} = (188.5n + 29)T_m$</td>
<td>$T_{pm} + T_b = 116T_m$</td>
</tr>
<tr>
<td>Zhu et al. [29]</td>
<td>$(2n + 3)T_{pm} + nT_{pa} + T_b = (58.12n + 174)T_m$</td>
<td>$2T_b + T_{pm} + T_i = 214.6T_m$</td>
</tr>
<tr>
<td>Win et al. [30]</td>
<td>$(n + 2)T_{pm} + nT_{pa} = (29.12n + 58)T_m$</td>
<td>$4T_{pm} + 4T_{pa} + 2T_i = 139.68T_m$</td>
</tr>
<tr>
<td>Our scheme</td>
<td>$(n + 1)T_{pm} = (29n + 29)T_m$</td>
<td>$3T_{pm} + 2T_{pa} = 87.24T_m$</td>
</tr>
</tbody>
</table>

Figure 3: Efficiency in signryption/encryption. The unit is $T_m$.

Figure 4: Efficiency in decryption/designryption. The unit is $T_m$. 
6. Conclusion

In this paper, we propose a CMRS scheme of high efficiency for ecological data sharing. After functional analysis, our scheme satisfies the security requirements, and it solves the problems of decryption fairness, recipient anonymity, and so on. After efficiency analysis, our scheme has high efficiency. It does not use BPO, but uses SPMOOEC for encryption and decryption, and ensures that SPMOOEC is used as little as possible, which greatly improves the efficiency of our scheme. After analysis, we can conclude that our scheme can be well applied to ecological data sharing.

Data Availability

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

Disclosure

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article.

Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China (61862052) and the Natural Science Function of Qinghai Province (2019-ZJ-7065).

References


