Hindawi
Journal of Sensors
Volume 2020, Article ID 5132861, 16 pages
https://doi.org/10.1155/2020/5132861

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

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 attentions to personal privacy issues. Usually, people intend to conceal their identities when they use multireceiver 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 parings 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 (SPMOOECC), 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 SPMOOECC, 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 SPMOOECC 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 SPMOOECC during encryption and decryption processes and minimizes the number of SPMOOECC, 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.

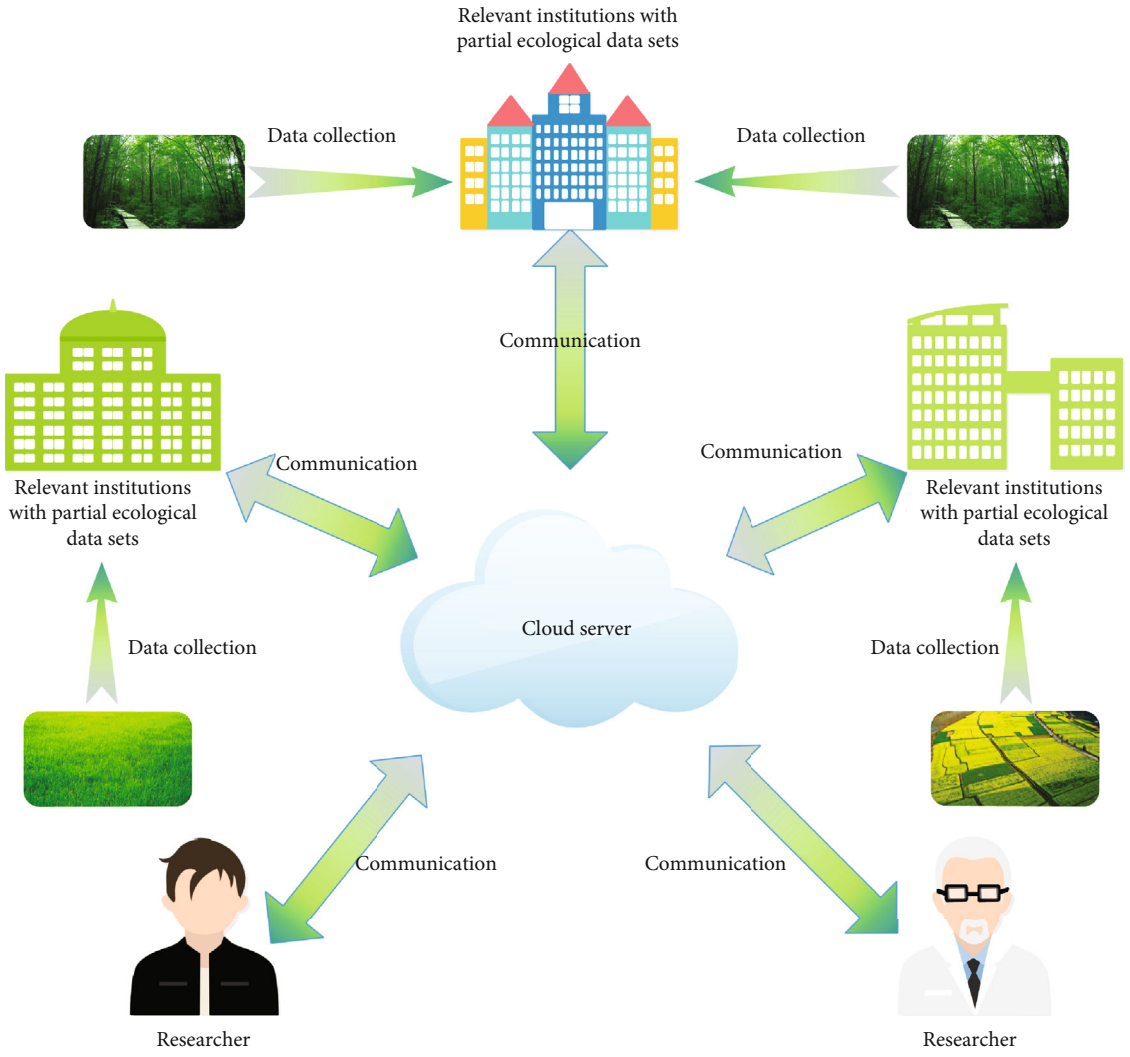

Figure 1: Ecological data sharing application framework.

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 $xyP \in G_p$ is CDHP

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

$$Adv_A^{CDHP} = \Pr\left[A(P, xP, yP) = xyP\right]. \quad (1)$$

CDHP hypothesis: we believe that Adv_A^{CDHP} 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

$$Adv_B^{ECDLP} = \Pr\left[B(P, xP) = x\right]. \quad (2)$$

ECDLP hypothesis: we believe that Adv_B^{ECDLP} can be ignored for any PPT algorithm B.

2.2. Algorithm Model

Definition 3. The algorithm model of this proposal consists of *SetUpAlgorithm*, *SetSecretValueAlgorithm*, *ExtractPartialPrivateKeyAlgorithm*, *SetPublicAndPrivateKeysAlgorithm*, *SigncryptionAlgorithm*, and *DesigncryptionAlgorithm*:

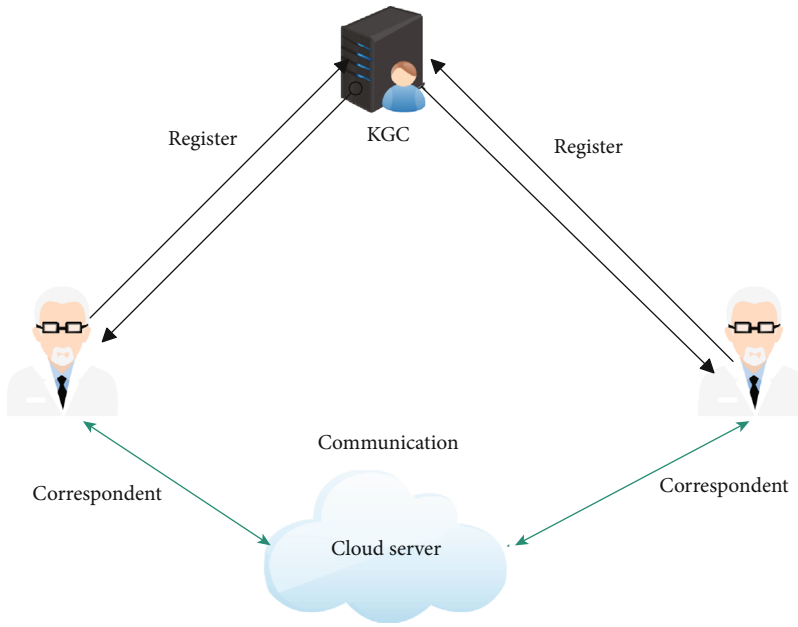

Figure 2: Ecological data sharing network framework.

Table 1: Notations.

Name	Meaning
BPO	Bilinear pairing operations
MTPHF	Map-to-point hash function
ECC	Elliptic curve cryptography
KGC	Key generation center
SPMOOECC	Scalar point multiplication operations on elliptic curve cryptography
CDHP	Computational Diffie-Hellman problem
ECDLP	Elliptic curve discrete logarithm problem
G_p	A cycle group of points on ECC
PK_i	Public key of user i, i represents the user's identity
P	Generator of G_p
p	Large prime integer
Pr	The probability of an event
SK_i	Private key of user i, i represents the user's identity
Z_p^*	Nonzero multiplicative group with large prime p
IME	Identity-based multireceiver encryption
IMS	Identity-based multireceiver signcryption
CME	Certificateless multireceiver encryption
CMS	Certificateless multireceiver signcryption

SetUpAlgorithm: input λ, λ 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 .

ExtractPartialPrivateKeyAlgorithm: enter the user's information ID_i, the master key s, the secret value parameter X_i, and the public key 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 $\mathbf{M}$, the recipient's public key PK_i $(1 \le i \le n)$, the private key SK_c of THE transmitter, and the information ID_c of the transmitter. The transmitter initiates the algorithm to obtain the ciphertext $\mathbf{C} = \text{Signcryption}(\text{pars}, \mathbf{M}, L, PK_i, SK_c, ID_c)$.

DesigncryptionAlgorithm: enter the public parameter pars, the recipient's private key SK_i, the recipient's information ID_i, and ciphertext $\mathbf{C}$. The receiver uses the algorithm to gain the plaintext $\mathbf{M} = \text{Designcryption}(\text{pars}, \mathbf{C}, SK_i, ID_i)$ and verifies $\mathbf{M}$ using the sender's public key PK_c.

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 $\mathscr{A}_1$ and $\mathscr{A}_2$ [20]. $\mathscr{A}_1$ does not know the system master key but $\mathscr{A}_2$ does, and $\mathscr{A}_1$ can replace the user's public key but $\mathscr{A}_2$ 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 $\mathscr{A}_1$'s IN-CMRS-CA and *Game2* for $\mathscr{A}_2$'s IN-CMRS-CA.

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

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

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

SetSecretValueQuery: $\mathscr{A}_1$ requires $\mathscr{C}$ to run the *SetSecretValueQuery* on the *ID*. After the request is received, $\mathscr{C}$ uses the *SetSecretValueAlgorithm* to get the user's secret value, and $\mathscr{C}$ transmits the result to $\mathscr{A}_1$.

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

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

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

SigncryptionQuery: $\mathscr{A}_1$ lets $\mathscr{C}$ perform the *SigncryptionQuery* with a series of information and plaintext **M**. After the request is received, $\mathscr{C}$ randomly selects the information ID_s to execute the *SigncryptionAlgorithm* to obtain the ciphertext **C** and transmits **C** to $\mathscr{A}_1$.

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

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

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

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

$$\mathrm{Adv}^{\mathrm{IN-CMRS-CA}}(A_1) = |2\ \Pr[\varphi^* = \varphi] - 1. \tag{3}$$

Definition 6. The probability that any $\mathscr{A}_1$ in the case of IN-CMRS-CA can obtain the *Game1* victory in time τ can satisfy $\mathrm{Adv}^{\mathrm{IN-CMRS-CA}}(A_1) \le \varepsilon$, indicating that the scheme conforms to (τ, ε) *-IN-CMRS-CA* security, ε represents a nonnegligible probability advantage and τ represents the time of a polynomial operation.

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

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

Phase 7. $\mathscr{A}_2$ makes a series of queries to $\mathscr{C}$ like Phase 4 of *Game1*, but $\mathscr{A}_2$ cannot carry out *PublicKeyReplacementQuery*. $\mathscr{C}$ responds accordingly.

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

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

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

$$\mathrm{Adv}^{\mathrm{IN-CMRS-CA}}(A_2) = |2\ \Pr[\varphi^* = \varphi] - 1. \tag{4}$$

Definition 9. The probability that any $\mathscr{A}_2$ in the case of IN-CMRS-CA can obtain the *Game2* victory in time τ can satisfy $\mathrm{Adv}^{\mathrm{IN-CMRS-CA}}(A_2) \le \varepsilon$, indicating that the scheme conforms to (τ, ε) -IN-CMRS-CA security, ε represents a nonnegligible probability advantage and τ represents a time of a polynomial operation.

2.3.2. Unforgeability. The unforgeability of the proposal in this paper is prescribed as the strong unforgeability of certificateless 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 $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively.

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

Setup. This procedure is similar to Setup in *Game1*.

Attack: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform flexible queries. The queries are similar to Phase 4 in *Game1*, after which $\mathscr{C}$ responds.

Forgery: $\mathscr{A}_1$ uses $L = \{\mathrm{ID}_1, \mathrm{ID}_2, \cdots, \mathrm{ID}_n\}$ and plaintext $\mathbf{M}$ to forge ciphertext $C*$. If any recipient in L can correctly decrypt $C*$, then $\mathscr{A}_1$ wins, otherwise $\mathscr{A}_1$ fails. In this process, *SigncryptionQuery* cannot get $C*$, other restrictions are similar to Phase 5 in *Game1*.

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

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

Setup. This procedure is similar to *Setup* in *Game2*.

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

Forgery: $\mathscr{A}_2$ uses $L = \{\mathrm{ID}_1, \mathrm{ID}_2, \cdots, \mathrm{ID}_n\}$and plaintext $\mathbf{M}$ to forge ciphertext $C*$. If any recipient in L can correctly decrypt $C*$, then $\mathscr{A}_2$ wins, otherwise $\mathscr{A}_2$ fails. In this process, *SigncryptionQuery* cannot get $C*$, and the other restrictions are similar to Phase 5 in *Game2*.

Definition 11. The probability that any $\mathscr{A}_2$ in the case of SU-CMRS-PA can obtain the *Game4* victory in time τ can satisfy $\mathrm{Adv}^{\mathrm{SU-CMRS-PA}}(A_2) \le \varepsilon$, indicating that the scheme conforms to (τ, ε) -SU-CMRS-PA security, ε represents a probabilistic advantage that is not negligible. τ 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 $\mathscr{A}_1$ and $\mathscr{A}_2$.

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

Setup. $\mathscr{C}$ performs the corresponding steps for obtaining the public parameter *pars* and the master key *s*, then transmits *pars* to $\mathscr{A}_1$ and secretly saves *s*. After receiving, $\mathscr{A}_1$ selects $L = \{ID_0, ID_1\}$ and sends L to $\mathscr{C}$.

Phase 12. This procedure is similar to Phase 4 in *Game1*.

Challenge: $\mathscr{A}_1$ picks identity list $L^* = \{\mathrm{ID}_2, \mathrm{ID}_3, \cdots, \mathrm{ID}_n\}$ and plaintext $\mathbf{M}$, $\mathscr{A}_1$ transmits them to $\mathscr{C}$. $\mathscr{C}$ selects $e \in \{0, 1\}$ and forms the ciphertext $C*$ with a new set of identity list $L^{**} = \{\mathrm{ID}_e, \mathrm{ID}_2, \mathrm{ID}_3, \cdots, \mathrm{ID}_n\}$, and $\mathscr{C}$ transmits $C*$ to $\mathscr{A}_1$.

Phase 13. This procedure is similar to Phase 5 in *Game1*.

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

$$\mathrm{Adv}^{\mathrm{ANO-CMRS-CA}}(A_1) = |2\ \Pr[e^* = e] - 1. \tag{5}$$

Definition 14. The probability that any $\mathscr{A}_1$ in the case of I ANO-CMRS-CA can obtain the *Game5* victory in time τ can satisfy $\mathrm{Adv}^{\mathrm{ANO-CMRS-CA}}(A_1) \le \varepsilon$, indicating that the scheme conforms to (τ, ε) -ANO-CMRS-CA security, ε represents a nonnegligible probability advantage and τ represents the time of a polynomial operation.

Game6: this process is the interaction between $\mathscr{C}$ and $\mathscr{A}_2$ under ANO-CMRS-CA. The details are the following:

Setup. $\mathscr{C}$ performs the corresponding steps for obtaining the public parameter *pars* and the master key *s*, then transmits the result to $\mathscr{A}_2$, $\mathscr{A}_2$ selects $L = \{\mathrm{ID}_0, \mathrm{ID}_1\}$ and output L.

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

Challenge: $\mathscr{A}_2$ picks identity list $L^* = \{\mathrm{ID}_2, \mathrm{ID}_3, \cdots, \mathrm{ID}_n\}$ and a plaintext $\mathbf{M}$, $\mathscr{A}_2$ transmits L^* and $\mathbf{M}$ to $\mathscr{C}$. $\mathscr{C}$ selects $e \in \{0, 1\}$ and forms a ciphertext with a new set of identity list $L^{**} = \{\mathrm{ID}_e, \mathrm{ID}_2, \mathrm{ID}_3, \cdots, \mathrm{ID}_n\}$, and $\mathscr{C}$ sends C^* to $\mathscr{A}_2$.

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

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

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

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

3. The Proposed Scheme

Our scheme includes *SetUpAlgorithm*, *KeyExtractAlgorithm*, *SigncryptionAlgorithm*, and *DesigncryptionAlgorithm*. The *KeyExtractAlgorithm* includes *SetSecretValueAlgorithm*, *ExtractPartialPrivateKeyAlgorithm*, and *SetPublicAndPrivateKeysAlgorithm*.

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

(1) Input λ, λ 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}, $P_{\text{pub}} = sP$

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

$$\begin{aligned} H_1 &= \{0,1\}* \times G_p \times G_p \longrightarrow Z_p^*, \\ H_2 &= Gp \times \{0,1\}* \longrightarrow Z_p^*, \\ H_3 &= Gp \times \{0,1\}* \longrightarrow Z_p^*, \\ H_4 &= Z_p^* \longrightarrow \{0,1\}*, \\ H_5 &= \{0,1\}* \times \{0,1\}* \times G_p \times G_p \times \{0,1\}* \\ &\quad \times G_p \times Z_p^* \times Z_p^* \times Z_p^* \times \cdots . \times Z_p^* \times Z_p^* \longrightarrow Z_p^* \end{aligned} \tag{7}$$

(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 $= \{Ep, G_p, E_k, D_k, p, P, P_{\text{pub}}, H_1, H_2, H_3, H_4, H_5\}$.

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 ID_i and 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 $h_i = H_1(\text{ID}_i, X_i, W_i)$, and then calculates $y_i = w_i + h_i s \bmod q$ and $z_i = y_i + H_2(sX_i, \text{ID}_i) \bmod q$, finally sending z_i and W_i to the user ID_i via a secure channel

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

The user ID_i public key pair is $\text{PK}_i = (X_i, Y_i)$, private key pair is $\text{SK}_i = (x_i, y_i)$. PK_i is published by KGC.

3.3. Signcryption Algorithm. The sender uses the public parameter *pars*, its own private key SK_c, plaintext $\mathbf{M}$, a set of recipient information $L = \{\text{ID}_1, \text{ID}_2, \cdots, \text{ID}_n\}$, the recipient's public key $\text{PK}_i\ (1 \le i \le n)$, and the information ID_c of a transmitter for signcryption:

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

(2) Calculate $\alpha_i = H_3(r X_i, \text{ID}_i),\ i = 1, 2, 3, \cdots n$

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

$$\begin{aligned} f(x) = .\prod_{j=1}^{n}(x - \alpha_i) + f(x) &= \prod_{j=1}^{n}(x - \alpha_i) + \xi (\text{mod } p) \\ &= a_0 + a_1 x \cdots + a_{n-1} x_{n-1} + x_n, a_i \in Z_p^* \end{aligned} \tag{8}$$

(4) Calculate $k = H_4(\xi)$, $J = E_k(\mathbf{M} \| ID_c)$.

(5) Calculate $h_c = H_5(\mathbf{M}, ID_c, X_c, PK_c, J, R, a0, a1, \cdots . a_n)$, where $\sigma = x_c + y_c + h_c r \bmod q$

(6) Generate ciphertext $\mathbf{C} = \{J, R, a0, a1, \cdots . a_n, \sigma\}$ and send the ciphertext to all recipients

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

(1) Calculate $\alpha i^* = H_3(x_i, R, \text{ID}_i)$

(2) Calculate $f(\alpha i^*) \longrightarrow \xi$

(3) Calculate $k = H_4(\xi)$ and $\mathbf{M} \| \text{ID}_c = D_k(J)$

(4) Search and obtain PK_c, calculate $h_c = H_5(\mathbf{M}, ID_c, X_c, PK_c, J, R, a_0, a_1, \cdots . a_n)$.

And judge whether $\sigma P = X_c + Y_c + h_c R$ is established, If successful, the recipient receives the plaintext $\mathbf{M}$ and then exits. Otherwise, the recipient rejects and quits.

4. Correctness Analysis and Security Proofs

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, $y_i P = W_i + h_i P_{\text{pub}}$ needs to be established. The specific inference of this equation is as follows:

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

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 $\sigma P = X_c + Y_c + h_c R$ guarantees the correctness of the *DesigncryptionAlgorithm*. The specific inference of the formula is as follows:

$$\sigma P = (x_i + y_i + h_c r)P = x_i P + y_i P + h_c r P = X_c + Y_c + h_c R. \tag{10}$$

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 $\mathscr{A}_1$ under the IN-CMRS-CA, in the polynomial time τ, if $\mathscr{A}_1$ wins the Game1 victory with a probability advantage ε that cannot be ignored, ($\mathscr{A}_1$ can request up to q_i hash queries $H_i (i = 1, 2, 3, 4, 5)$, q_c KeyQuery, SetSecretValueQuery, q_b ExtractPrivateKeyQuery, q_{pk} SetPublicAndPrivateKeysQuery, q_r PublicKeyReplacementQuery, q_a SigncryptionQuery, and q_d DesigncryptionQuery), then within time $\tau' \le \tau + (2q_c + 3q_d)O(\tau_s)$, challenger $\mathscr{C}$'s interaction with opponent $\mathscr{A}_1$ makes $\mathscr{C}$ overcome CDHP with a probability advantage of $\varepsilon' \ge 2(\varepsilon - q_d\, q_5/2^k)/nq_3$, τ_s is the time of SPMOOECC.*

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

Setup. $\mathscr{C}$ performs the corresponding steps for obtaining the public parameter $pars = (E_p, G_p, E_k, D_k, p, P, P_{\text{pub}} = aP, H_1, H_2, H_3, H_4, H_5)$ and master key $= a \in Z_p^*$, and then transmits *pars* to $\mathscr{A}_1$ and save *s*. After the recipient receives it, $\mathscr{A}_1$ selects the target identity $L = \{\text{ID}_1, \text{ID}_2, \cdots, \text{ID}_n\}$ and transmits L to $\mathscr{C}$, n denotes a positive whole number. H_1, H_2, H_3, H_4, and H_5 are all random oracles; they are controlled by $\mathscr{C}$. The details are the following:

(1) *H_1HashQuery*: the tuple (ID_j, X_j, W_j) is taken as input, and $\mathscr{A}_1$ requires $\mathscr{C}$ to execute *H_1HashQuery*. After receiving, $\mathscr{C}$ checks whether the tuple $(\text{ID}_j, X_j, W_j, d_j)$ exists in list L_1. If it exists, $\mathscr{C}$ will send d_j to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects $d_j \in Z_p^*$ at random and sends d_j to $\mathscr{A}_1$ and restores the tuple $(\text{ID}_j, X_j, W_j, d_j)$ into list L_1

(2) *H_2HashQuery*: the tuple (sX_j, ID_j) and $(X_j P_{\text{pub}}, \text{ID}_j)$ are taken as input, and $\mathscr{A}_1$ requires $\mathscr{C}$ to execute *H_2HashQuery*. After receiving, $\mathscr{C}$ checks whether the tuple $(sX_j, \text{ID}_j, \theta_j)$ and $(X_j P_{\text{pub}}, ID_j, \delta_j)$ exist in the list L_2. If yes, $\mathscr{C}$ will send θ_j and δ_j to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects two integer θ_j and $\delta_j \in Z_p^*$ at random and transmits them to $\mathscr{A}_1$ and restores the tuple $(sX_j, \text{ID}_j, \theta_j)$ and $(X_j P_{\text{pub}}, ID_j, \delta_j)$ into list L_2

(3) *H_3HashQuery*: the tuple (rX_j, ID_j) is taken as input, and $\mathscr{A}_1$ requires $\mathscr{C}$ to execute *H_3HashQuery*. After receiving, $\mathscr{C}$ judges whether or not the tuple $(rX_j, \text{ID}_j, \alpha_j)$ exists in the list L_3. If it exists, $\mathscr{C}$ will send α_j to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects $\alpha_j \in Z_p^*$ at random and sends α_j to $\mathscr{A}_1$ and restores the tuple $(rX_j, \text{ID}_j, \alpha_j)$ into list L_3

(4) *H_4HashQuery*: the tuple (ξ) is taken as input, and $\mathscr{A}_1$ requires $\mathscr{C}$ to execute *H_4HashQuery*. After receiving, $\mathscr{C}$ judges whether or not the tuple (ξ, k) exists in the list L_4. If it exists, $\mathscr{C}$ will send k to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects $k \in Z_p^*$ at random, and sends k to $\mathscr{A}_1$, and restores the tuple (ξ, k) into list L_4

(5) *H_5HashQuery*: The tuple $(\mathbf{M}, ID_j, X_j, PK_j, J, R, a_0, a_1, \cdots . a_n)$ is taken as input, and $\mathscr{A}_1$ requires $\mathscr{C}$ to execute *H_5HashQuery*. After receiving, $\mathscr{C}$ judges whether or not the tuple $(\mathbf{M}, ID_j, X_j, PK_j, J, R, a_0, a_1, \cdots . a_n, h_j)$ exists in list L_5. If it exists, $\mathscr{C}$ will send h_j to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects $hj \in Z_p^*$ at random, and sends h_j to $\mathscr{A}_1$ and restores the tuple $(\mathbf{M}, ID_j, X_j, PK_j, J, R, a_0, a_1, \cdots . a_n, h_j)$ into list L_5

Phase 21. $\mathscr{A}_1$ requires $\mathscr{C}$ to perform flexible queries and $\mathscr{C}$ performs feedback. The details are the following:

(1) *KeyQuery*: $\mathscr{C}$ checks if the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ exists in the list L_C. If it exists, $\mathscr{C}$ reserves the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$. Otherwise, $\mathscr{C}$ does the following:

(a) If $\text{ID}_j = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ selects two integers $x_j, w_j \in Z_p^*$ at random, sets $X_j = x_j P$ and $\text{SK}_j \longleftarrow \perp$, computes $W_j = w_j P, y_j = w_j + H_1 (ID_i, X_i, W_i)\, s \bmod q$, $Y_j = y_j P, PK_j = (X_j, Y_j)$ and then updates tuples $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C and $(\text{ID}_j, X_j, W_j, d_j)$ in list L_1, respectively

(b) If $\text{ID}_j ! = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ selects two integers $x_j, z_j \in Z_p^*$ at random, sets $X_j = x_j P$, computes $y_i = z_i - H_2 (x_i P_{\text{pub}}, \text{ID}_i)$, $Y_j = y_j P$, $\text{PK}_j = (X_j, Y_j)$, $\text{SK}_j = (x_j, y_j)$ and then updates tuples $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C and $(\text{ID}_j, X_j, W_j, d_j)$ in list L_1, respectively

(2) *SetSecretValueQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to execute the *SetSecretValueQuery* on ID_j. After receiving the request, $\mathscr{C}$ checks whether there are tuples $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C. if it exists, $\mathscr{C}$ sends x_j to $\mathscr{A}_1$, otherwise, C performs the *KeyQuery* to generate a tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ and then sends x_j to $\mathscr{A}_1$

(3) *ExtractPartialPrivateKeyQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform the *ExtractPartialPrivateKeyQuery* on ID_j.

After receiving the request, the details are the followings:

(a) If $\mathrm{ID}_j = \mathrm{ID}_i$, $i = 1, 2, 3, \cdots, n$,$\mathscr{C}$ sends "failure" to $\mathscr{A}_1$

(b) If $\mathrm{ID}_j! = \mathrm{ID}_i$, $i = 1, 2, 3, \cdots, n$,$\mathscr{C}$ judges whether or not the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, w_j, x_j, z_j)$ exists in the list L_C, and if so, $\mathscr{C}$ sends z_j to $\mathscr{A}_1$, otherwise, $\mathscr{C}$ performs the *KeyQuery* to generate tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ and sends z_j to $\mathscr{A}_1$

(4) *SetPublicAndPrivateKeysQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to execute the public key portion of *SetPublicAndPrivateKeysQuery* on the ID_j. After the recipient receives it, $\mathscr{C}$ judges whether or not the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ exists in list L_C. If it exists, $\mathscr{C}$ sends PK_j to $\mathscr{A}_1$, otherwise $\mathscr{C}$ performs *KeyQuery* to generate a tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ and then send PK_j to $\mathscr{A}_1$

$\mathscr{A}_1$ requires $\mathscr{C}$ to execute the private key portion of *SetPublicAndPrivateKeysQuery* on the ID_j. After the recipient receives it, $\mathscr{C}$ responds:

(a) If $\mathrm{ID}_j = \mathrm{ID}_i$, for $i = 1, 2, 3, \cdots, n$,$\mathscr{C}$ sends "failure" to $\mathscr{A}_1$

(b) If $\mathrm{ID}_j! = \mathrm{ID}_i$, for $i = 1, 2, 3, \cdots, n$,$\mathscr{C}$ judges whether or not the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, w_j, x_j, z_j)$ exists in list L_C, and if so, $\mathscr{C}$ sends SK_j to $\mathscr{A}_1$, otherwise, $\mathscr{C}$ uses *KeyQuery* to generate tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ and sends SK_j to $\mathscr{A}_1$

(5) *PublicKeyReplacementQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform a *PublicKeyReplacementQuery* on ID_j with PK_j'. After the recipient receives it, $\mathscr{C}$queries the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, w_j, x_j, z_j)$ in list L_C, and uses PK_j' instead of PK_j. Next, $\mathscr{C}$ restores the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ in list L_C

(6) *SigncryptionQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform the *SigncryptionQuery* for plaintext **M** and information ID_s. After receiving the request, $\mathscr{C}$ determines if $\mathrm{ID}_s ! = \mathrm{ID}_i$ is correct, $i = 1, 2, 3, \cdots, n$. If the expression is correct, $\mathscr{C}$ executes the *SetPublicAndPrivateKeysQuery* to generate the private key SK_s and ciphertext **C**, and then sends **C** to $\mathscr{A}_1$ if not, $\mathscr{C}$ implement s the corresponding operation

(a) Select $r \in Z_p^*$ at random and calculate Select

$$\begin{aligned} R &= rP, \\ rX_j &= rx_jP, \\ \alpha_j &= H_3\left(rX_j, ID_j\right), j = 1, 2, 3, \cdots, n; \\ \xi &\in Z_p^* \end{aligned} \tag{11}$$

at random and define the formula:

$$f(x) = \prod_{j=1}^{n}(x - \alpha_i) + \xi(\mathrm{mod}\ p) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + x^n,\ a_j \epsilon Z_{p.}^* \tag{12}$$

(b) Calculate $k = H_4(\xi), J = E_k(M \| ID_s), h = H_5(M, ID_s, X_s, PKs, J, R, a_0, a_1, \cdots, a_{n-1}), \sigma = x_s + y_s + h \mod q$;

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

(d) Output the ciphertext $C = (J, R, a_0, a_1, \cdots, a_{n-1}, \sigma)$ to $\mathscr{A}_1$

(7) *DesigncryptionQuery*: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform a *DesigncryptionQuery* for ciphertext **C**. After receiving the request, $\mathscr{C}$ selects ID_j at random and verifies whether $\mathrm{ID}_j = \mathrm{ID}_i$ is true, $i = 1, 2, 3, \cdots, n$. If the formula is true, $\mathscr{C}$ sends "failure" to $\mathscr{A}_1$ Otherwise, $\mathscr{C}$ will perform the corresponding operation:

(a) Search for the tuple $(\mathrm{ID}_j, \mathrm{SK}_j, \mathrm{PK}_j, x_j, z_j)$ in list L_C to obtain SK_j and compute $\alpha_j = H_3(x_jR, \mathrm{ID}_j)$

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

(c) Calculate $k = H_4(\xi), \mathbf{M} \| \mathrm{ID}_s = D_k(J)$

(d) Consider whether the equation $\sigma P = X_s + Y_s + h_sR$ can be established. If it is established, $\mathscr{C}$ sends **M** to $\mathscr{A}_1$ If not, $\mathscr{C}$ sends "failure" to $\mathscr{A}_1$

Challenge: $\mathscr{A}_1$ selects the plaintext (M_0, M_1) at random, they are equal in length, and they are transmitted to $\mathscr{C}$. After receiving, $\mathscr{C}$ selects a bit $\varphi \in \{0, 1\}$ *at* random, $\mathscr{C}$ uses M_φ to get the ciphertext $C*$, and M_φ is the selected plaintext. The details are the following:

(a) Stipulate $R = y\mathrm{PK}_i, rX_i = y(\mathrm{PK}_i + P_{\mathrm{pub}})$ and $\alpha_i = H_3(rX_i,), i = 1, 2, 3, \cdots, n$

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

$$f(x) = \prod_{j=1}^{n}(x - \alpha_i) + \xi(\mathrm{mod}\ p) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + x^n,\ a_j \epsilon Z_{p*}^p \tag{13}$$

(c) Calculate $k = H_4(\xi), J^* = E_k(\|\mathrm{ID}_s)$ and $h = (\mathbf{M}, \mathrm{ID}_s, X_s, \mathrm{PK}_s, J, R, a_0, a_1, \cdots a_n)$

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

(e) Output the $C* = (J^*, W_i, z, h, a_0, a_1, \cdots, a_{n-1})$ to $\mathscr{A}_1$

Phase 22. $\mathscr{A}_1$ requires $\mathscr{C}$ to perform the same queries as Phase 4, but $\mathscr{A}_1$ cannot run the *DesigncryptionQuery* against $C*$.

Guess: $\mathscr{A}_1$ gives a φ^* at random. If $\varphi^* = \varphi$, then $\mathscr{A}_1$ wins, $\mathscr{C}$ outputs CDHP's solution $xyP = rX_i - R$. Otherwise, $\mathscr{C}$ outputs "failure."

From the above discussion, it can be concluded that the H_5*hash* provides a valid ciphertext during the *DesigncryptionQuery*; therefore, the probability of ciphertext being denied is no more than $q_5/2^k$. During the attack process, since $\mathscr{A}_1$ requires $\mathscr{C}$ to execute the q_d *DesigncryptionQuery*, the probability of $\mathscr{C}$ decryption success is $\varepsilon_d \geq \varepsilon - q_5 q_d/2^k$. The H_3 *hash* satisfies CDHP during the guessing process, so the probability that xyP is correctly calculated by $\mathscr{C}$ is at least $\varepsilon_g = 2/nq_3$. in the running time $\tau^{'} \leq \tau + (2q_c + 3q_d)O(\tau_s)$, $\mathscr{C}$'s interaction with opponent $\mathscr{A}_1$ makes $\mathscr{C}$ overcome CDHP, and the probability advantage is $\varepsilon^{'} \geq \varepsilon_d \varepsilon_g \geq 2(\varepsilon - q_d q_5/2^k)/nq_3$,$\tau_s$ is the time of the SPMOOECC.

Theorem 23. *There is an opponent $\mathscr{A}_2$ under the IN-CLMS-CA. In the polynomial time τ, if the $\mathscr{A}_2$ wins the Game2 victory with a probability advantage ε that cannot be ignored, ($\mathscr{A}_2$ can request up to q_i hash queries $H_i (i = 1, 2, 3, 4, 5)$, q_c KeyQuery, q_e SetSecretValueQuery, q_b ExtractPrivateKeyQuery, q_{pk} SetPublicAndPrivateKeysQuery, q_a SigncryptionQuery, and q_d DesigncryptionQuery.), then within time $\tau^{'} \leq \tau + (3q_c + 3q_d)$ $O(\tau_s)$, challenger $\mathscr{C}$'s interaction with opponent $\mathscr{A}_2$ makes $\mathscr{C}$ overcome CDHP with a probability advantage of $\varepsilon^{'} \geq 2(\varepsilon - q_d q_5/2^k)/nq_3$, τ_s is the time of SPMOOECC.*

Proof. Under a nonnegligible probability, it is assumed that the opponent $\mathscr{A}_2$ can attack the IN-CMRS-CA security, the probability advantage is ε, and under the stochastic prediction model, $\mathscr{A}_2$ requires challenger $\mathscr{C}$ to perform a series of queries. Given (P, xP, yP), in the time-limited polynomial, $\mathscr{C}$ obtains xyP through the opponent $\mathscr{A}_2$ interaction to overcome CDHP. $\mathscr{C}$ and opponent $\mathscr{A}_2$ will interact with the following:

Setup. $\mathscr{C}$ performs the corresponding steps for obtaining the public parameter pars $= (E_p, G_p, E_k, D_k, p, P, K = xP, P_{\text{pub}}, H_1, H_2, H_3, H_4, H_5)$ and master key $s \in Z_p^*$, and then returns the result to $\mathscr{A}_2$, where $x \in Z_p^*$. After receiving the result, $\mathscr{A}_2$ outputs $L = \{\text{ID}_1, \text{ID}_2, \text{ID}_3, \cdots, \text{ID}_n\}$, where n is a positive integer. H_1, H_2, H_3, H_4, and H_5 are random predictions controlled by $\mathscr{C}$, the interactions between $\mathscr{A}_2$ and $\mathscr{C}$ is similar to *Setup* in Theorem 20.

Phase 24. $\mathscr{A}_2$ requires $\mathscr{C}$ to perform flexible queries and $\mathscr{C}$ performs feedback. The details are the following:

(1) *KeyQuery:* $\mathscr{C}$ checks if there is a tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in the list L_C. If it exists, $\mathscr{C}$ reserves the tuple. Otherwise, $\mathscr{C}$ will perform related operations as follows:

(a) If $\text{ID}_j = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ selects two integers $xj, wj \in Z_p^*$ at random and computes $W_j = w_j P$, $y_j = w_j + H_1(\text{ID}_j, X_j, W_j)s \bmod q$, $Y_j = y_j P$, $\text{PK}_j = (X_j, Y_j)$, and then updates tuples $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C and $(\text{ID}_j, X_j, W_j, h_j)$ in list L_1, respectively, where $X_j = x_j P$ and $\text{SK}_j \longleftarrow \perp$

(b) If $\text{ID}_j ! = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$,$\mathscr{C}$ selects two integers $xj, zj \in Z_p^*$ at random, and sets $X_j = x_j P$, computes $y_i = z_i - H_2(x_i P_{\text{pub}}, ID_i)$, $Y_j = y_j P$, $\text{PK}_j = (X_j, Y_j)$, $\text{SK}_j = (x_j, y_j)$, and then updates tuples $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C and $(\text{ID}_j, X_j, W_j, h_j)$ in list L_1, respectively

(2) *SetSecretValueQuery*: $\mathscr{A}_2$ requires $\mathscr{C}$ to execute *SetSecretValueQuery* for *c*. Upon receipt of the request, $\mathscr{C}$ executes as follows:

(a) If $\text{ID}_j = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ sends "failure" to $\mathscr{A}_2$

(b) If $\text{ID}_j ! = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ judges whether or not the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ exists in list L_C. If yes, $\mathscr{C}$ returns x_j to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ uses *KeyQuery* to generate the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ and sends x_j to $\mathscr{A}_2$.

(3) *ExtractPartialPrivateKeyQuery*: $\mathscr{A}_2$ requires $\mathscr{C}$ to perform *ExtractPartialPrivateKeyQuery* against ID_j. After the recipient receives it, $\mathscr{C}$ checks whether there is a tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ in list L_C. If it exists, $\mathscr{C}$ sends z_j to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ runs *KeyQuery* to generate tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ and sends z_j to $\mathscr{A}_2$

(4) *SetPublicAndPrivateKeysQuery*: $\mathscr{A}_2$ requires $\mathscr{C}$ to perform the public key portion of *SetPublicAndPrivateKeysQuery* against ID_j. After receiving the request, $\mathscr{C}$ checks whether there is a tuple $(ID_j, SK_j, PK_j, x_j, z_j)$ in list L_C. If it exists, $\mathscr{C}$ sends PK_j to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ runs *KeyQuery* to generate tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ and sends PK_j to $\mathscr{A}_2$

$\mathscr{A}_2$ requires $\mathscr{C}$ to perform the private key portion of *SetPublicAndPrivateKeysQuery* against ID_j. After the recipient receives it, $\mathscr{C}$ responded with the following response:

(a) If $\text{ID}_j = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ sends "failure" to $\mathscr{A}_2$

(b) If $\text{ID}_j ! = \text{ID}_i$, $i = 1, 2, 3, \cdots, n$, $\mathscr{C}$ judges whether or not the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ exists in list L_C. If yes, $\mathscr{C}$ returns SK_j to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ run the *KeyQuery* to generate the tuple $(\text{ID}_j, \text{SK}_j, \text{PK}_j, x_j, z_j)$ and sends SK_j to $\mathscr{A}_2$

(5) *SigncryptionQuery*: this section is similar to the *SigncryptionQuery* in Theorem 20

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

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

(a) Stipulate $R = y(PK_i + Y)$, $rX_i = y(PK_i + P_{\text{pub}})$ and $\alpha_i = H_3(rX_i, \text{ID}_i)$, $i = 1, 2, 3, \cdots, n$, $Y = K + P_{\text{pub}}$

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

$$f(x) = \prod_{j=1}^{n}(x - \alpha_i) + \xi(\text{mod } p) = a_0 + a_1 x + \cdots + a_{n-1}x_{n-1} + x_n, a_i \epsilon Z_p^* \quad (14)$$

(c) $_{\text{Calculate}}\, k = H_3(\xi)$, $J^* = E_k(\mathbf{M}_\varphi \| ID_s)$ and

$$h = H_5\,(m, ID_s, X_s, Y_s, J, R, a_0, a_1, \cdots . a_n)\,; \quad (15)$$

(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 $\mathscr{A}_2$

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

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

From the above discussion, it can be concluded that the H_5*hash* provides a valid ciphertext during the *DesigncryptionQuery*, therefore, the probability of ciphertext being denied is no more than $q_5/2^k$. During the attack process, since $\mathscr{A}_2$ requires $\mathscr{C}$ to execute the q_d *DesigncryptionQuery*, the probability of $\mathscr{C}$ decryption success is $\varepsilon_d \geq \varepsilon - q_5 q_d/2^k$. The H_3 *hash* satisfies CDHP during the guessing process, so the probability that *xyP* is correctly calculated by $\mathscr{C}$ is at least $\varepsilon_g = 2/nq_3$. In the running time $\tau' \leq \tau + (3q_c + 3q_d)O(\tau_s)$, $\mathscr{C}$'s interaction with opponent $\mathscr{A}_2$ makes $\mathscr{C}$ overcome CDHP, and the probability advantage is $\tau_s(\tau_s)\varepsilon' \geq 2(\varepsilon - q_d q_5/2^k)/nq_3$, τ_s is the time of the SPMOOECC.

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

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

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

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

Forgery: $\mathscr{A}_2$ falsifies ciphertext $C^* = (J, R, z, h, a_0, a_1, \cdots, a_{n-1})$ using $L = \{\text{ID}_1, \text{ID}_2, \cdots, \text{ID}_n\}$ and a plaintext $\mathbf{M}$. If the formula $\sigma P = X_s + Y_s + h_s R$ can be established, $C*$ faked successfully, defining $PK'_i = y^{-1}PK_i$ and $rX_i = y(\text{PK}_i + P_{\text{pub}})$, $\mathscr{C}$ calculates $rX_i = PK_i' + xyP$, and outputs $xyP = rX_i - PK_i$, *xyP* is the CDHP's solution. Otherwise $\mathscr{C}$ returns *"failure"*.

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

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

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

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

Attack: $\mathscr{A}_1$ requires $\mathscr{C}$ to perform flexible queries similar to Phase 4 in Theorem 23.

Forgery: $\mathscr{A}_2$ falsifies ciphertext $C^* = (J, R, z, h, a_0, a_1, \cdots, a_{n-1})$ using $L = \{\text{ID}_1, \text{ID}_2, \cdots, \text{ID}_n\}$ and a plaintext $\mathbf{M}$. If the formula $\sigma P = X_s + Y_s + h_s R$ can be established, $C*$ faked successfully, defining $\text{PK}'_i = y^{-1}\text{PK}_i$ and $rX_i = y(\text{PK}_i + P_{\text{pub}})$, $\mathscr{C}$ calculates $rX_i = \text{PK}_i + xyP$ and outputs $xyP = rX_i - \text{PK}_i$, *xyP* is the CDHP's solution. Otherwise $\mathscr{C}$ returns "failure."

From the above discussion, it is concluded that the probability of success of the q_a *SigncryptionQuery* is at least $\varepsilon_a = \varepsilon - q_a/2^k$ calculates *xyP* in the forgery process, and the

probability of xyP correct is at least $\varepsilon_g = 1/2$ Therefore, in the running time $\tau' \le \tau + (3q_c + 2q_a)O(\tau_s)$, $\mathscr{C}$'s interaction with opponent $\mathscr{A}_2$ makes $\mathscr{C}$ overcome CDHP, and the probability advantage is $\varepsilon' \ge \varepsilon_a \varepsilon_{g=}(\varepsilon - q_a/2^k)/2$. τ_s is the time of the SPMOOECC.

Theorem 28. *There is an opponent $\mathscr{A}_1$ under the ANO-CMRS-CA. In the polynomial time τ, if the $\mathscr{A}_1$ wins the Game5 victory with a probability advantage ε that cannot be ignored ($\mathscr{A}_1$ can obtain queries similar to that obtained by $\mathscr{A}_1$ in Theorem 20), within time $\tau' \le \tau + (2q_c + 3q_d)O(\tau_s)$, challenger $\mathscr{C}$ can overcome CDHP by interacting with opponent $\mathscr{A}_1$ with a probability advantage of $\varepsilon' \ge (\varepsilon - q_d q_5/2^k)/nq_3$, τ_s is the time of SPMOOECC.*

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

Setup. $\mathscr{C}$ execution of this algorithm is used to generate the public parameter $pars = (E_p, G_p, E_k, D_k, p, P, P_{\text{pub}} = xP, H_1, H_2, H_3, H_4, H_5)$ and the master key $s = a \in Z_p^*$, then sends the *pars* to the $\mathscr{A}_1$ and secretly saves s. After the $\mathscr{A}_1$ receives the *pars*, the $\mathscr{A}_1$ outputs $L = \{\text{ID}_0, \text{ID}_1\}$, where n is a positive integer. H_1, H_2, H_3, H_4, and H_5 are all random oracles, they are controlled by $\mathscr{C}$. The interaction is similar to *Setup* in Theorem 20, and the two sides of the interaction are $\mathscr{A}_1$ and $\mathscr{C}$.

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

Challenge: $\mathscr{A}_1$ selects the target identifier $L^* = \{\text{ID}_2, \text{ID}_3, \cdots, \text{ID}_n\}$ and plaintext $\mathbf{M}$, and sends L^* and $\mathbf{M}$ to $\mathscr{C}$. After receiving, $\mathscr{C}$ selects a bit $e \in \{0, 1\}$ and uses the new target identifier $L^{**} = \{\text{ID}_e, \text{ID}_2, \text{ID}_3, \cdots, \text{ID}_n\}$ to obtain the ciphertext $C*$:

(a) Calculate $R = y(\text{PK}_i), rX_i = y(\text{PK}_i + P_{\text{pub}})$ and $\alpha_j = H_3(rX_i, \text{ID}_i)$

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

$$f(x) = \prod_{j=1}^{n}(x - \alpha_i) + \xi(\text{mod } p) = a_0 + a_1 x \cdots + a_{n-1}x_{n-1} + x_n, a_i \in Z_p^* \quad (16)$$

(c) Calculate $k = H_3(\xi)$, $J^* = E_k(\mathbf{M}_\varphi \| \text{ID}_s)$ and $h^* = H_4(\mathbf{M}, \text{ID}_s, X_s, Y_s, J, R, a_0, a_1, \cdots a_n)$

(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 $\mathscr{A}_1$

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

Guess: $\mathscr{A}_1$ gives a e^* at random. If $e^* = e$, then $\mathscr{A}_1$ wins, $\mathscr{C}$ outputs CDHP's solution $xyP = rX_i - R$. If not, $\mathscr{C}$ outputs "failure."

From the above discussion, it can be concluded that the $H_5 hash$ provides a valid ciphertext during the *DesigncryptionQuery*, therefore, the probability of ciphertext being denied is no more than $q_5/2^k$. During the attack process, since $\mathscr{A}_1$ requires $\mathscr{C}$ to execute the q_d *DesigncryptionQuery*, the probability of $\mathscr{C}$ decryption success is $\varepsilon_d \ge \varepsilon - q_5 q_d/2^k$. The H_3 *hash* satisfies CDHP during the guessing process, so the probability that xyP is correctly calculated by $\mathscr{C}$ is at least $\varepsilon_g = 1/nq_3$ $\mathscr{C}$ can solve CDHP by interacting with opponent $\mathscr{A}_1$ in the running time $\tau' \le \tau + (2q_c + 3q_d)O(\tau_s)$, and the probability advantage is $\varepsilon' \ge \varepsilon_d \varepsilon_g \ge (\varepsilon - q_d q_5/2^k)/nq_3$, τ_s is the time of the SPMOOECC.

Theorem 31. *There is an opponent $\mathscr{A}_2$ under the ANO-CMRS-CA In the polynomial time τ, if the $\mathscr{A}_2$ wins the Game6 victory with a probability advantage ε that cannot be ignored ($\mathscr{A}_2$ can obtain queries similar to that obtained by $\mathscr{A}_2$ in Theorem 23.), then within time $\tau' \le \tau + (3q_c + 3q_d)O(\tau_s)$, challenger $\mathscr{C}$ can overcome CDHP by interacting with opponent $\mathscr{A}_2$ with a probability advantage of $\varepsilon' \ge \varepsilon - q_d q_5/2^k/nq_3$, τ_s is the time of SPMOOECC.*

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

Setup. $\mathscr{C}$ performs the corresponding steps for obtaining the public parameter $pars = (E_p, G_p, E_k, D_k, p, P, K = xP, P_{\text{pub}}, H_1, H_2, H_3, H_4, H_5)$ and master key $s \in Z_p^*$ and then returns the result to $\mathscr{A}_2$, where $x \in Z_p^*$. After receiving the result, $\mathscr{A}_2$ outputs $L = \{\text{ID}_0, \text{ID}_1\}$. H_1, H_2, H_3, H_4, and H_5 are random predictions controlled by $\mathscr{C}$. The interaction is similar to *Setup* in Theorem 20, and the two sides of the interaction are $\mathscr{A}_2$ and $\mathscr{C}$.

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

Challenge: $\mathscr{A}_2$ selects the target identifier $L^* = \{\text{ID}_2, \text{ID}_3, \cdots, \text{ID}_n\}$and plaintext $\mathbf{M}$ and sends L^* and $\mathbf{M}$ to $\mathscr{C}$. After receiving, $\mathscr{C}$ selects a bit $e \in \{0, 1\}$ and uses the new target identifier $L^{**} = \{\text{ID}_e, \text{ID}_2, \text{ID}_3, \cdots, \text{ID}_n\}$ to obtain the ciphertext $C*$:

(a) Calculate $R = y(PK_i + Y), rX_i = y(PK_i + P_{pub})$ and $\alpha_j = H_3(rX_i, ID_i), Y = K + P_{pub}, i = e, 2, 3, \cdots, n$

Table 2: Comparison of functions.

Schemes	Decryption fairness	Receiver anonymity	Partial private key verifiability	Signature
Selvi et al. [25]	N	N	N	Y
Hung et al. [27]	N	Y	N	N
Zhu et al. [29]	N	N	N	N
Win et al. [30]	N	N	Y	N
Our program	Y	Y	Y	Y

Table 3: Symbols' definition.

Symbols	Symbols' definition
T_{m}	Time of a modular multiplication
T_{pm}	Time of a scalar point multiplication on ECC operation, $T_{\mathrm{pm}} \approx 29T_{\mathrm{m}}$.
$\mathrm{T_{pa}}$	Time of a point addition on ECC operation, $T_{\mathrm{pa}} \approx 0.12T_{\mathrm{m}}$.
$\mathrm{T_b}$	Time of a bilinear pairing operation, $T_{\mathrm{b}} \approx 87T_{\mathrm{m}}$.
T_t	Time of modular inversion operation, $T_t \approx 11.6T_{\mathrm{m}}$.
$\mathrm{T_{be}}$	Time of a bilinear pairing exponentiation operation, $T_{\mathrm{be}} \approx 43.5T_{\mathrm{m}}$.
$\mathrm{T_h}$	Time of a map-to-point hash function operation, $T_{\mathrm{h}} \approx 29T_{\mathrm{m}}$

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

$$f(x) = \prod_{j=1}^{n}(x - \alpha_i) + \xi(\mathrm{mod}\ p) = a_0 + a_1 x \cdots + a_{n-1}x_{n-1} + x_n, a_i \in Z_p^* \quad (17)$$

(c) Calculate $k = H_3(\xi), J^* = E_k(\mathbf{M}_\varphi \| ID_s)$ and $h* = H_4(\mathbf{M}, ID_s, X_s, Y_s, J, R, a_0, a_1, \cdots a_n)$

(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 $\mathscr{A}_2$

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

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

From the above discussion, it can be concluded that the H_5*hash* provides a valid ciphertext during the *DesigncryptionQuery*; therefore, the probability of ciphertext being denied is no more than $q_5/2^k$. During the attack process, since $\mathscr{A}_2$ requires $\mathscr{C}$ to execute the q_d *DesigncryptionQuery process*, the probability of $\mathscr{C}$ decryption success is $\varepsilon_d \geq \varepsilon - q_5 q_d/2^k$. The H_3 *hash* satisfies CDHP during the guessing process, so the probability that xyP is correctly calculated by $\mathscr{C}$ is at least $\varepsilon_g = 1/nq_3$. In the running time $\tau' \leq \tau + (3q_c + 3q_d)O(\tau_s)$, $\mathscr{C}$'s interaction with opponent $\mathscr{A}_2$ makes $\mathscr{C}$ overcome CDHP, and the probability advantage is $\varepsilon' \geq \varepsilon_d \varepsilon_g \geq (\varepsilon - q_d q_5/2^k)/nq_3$, τ_s 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 signcryptions. 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

Table 4: Comparison of efficiency.

Schemes	Signcryption/encryption	Decryption/designcryption
Selvi et al. [25]	$(n+1)T_{pm} + T_i + (n+1)T_b + (n+1)T_{be} \approx (159.5n + 171.1)T_m$	$T_{pm} + T_{pa} + 2T_b + T_{be} \approx 246.62T_m$
Hung et al. [27]	$(n+1)T_{pm} + nT_{be} + nT_b + nT_h \approx (188.5n + 29)T_m$	$T_{pm} + T_b \approx 116T_m$
Zhu et al. [29]	$(2n+3)T_{pm} + nT_{pa} + T_b \approx (58.12n + 174)T_m$	$2T_b + T_{pm} + T_i \approx 214.6T_m$
Win et al. [30]	$(n+2)T_{pm} + nT_{pa} \approx (29.12n + 58)T_m$	$4T_{pm} + 4T_{pa} + 2T_i \approx 139.68T_m$
Our scheme	$(n+1)T_{pm} \approx (29n + 29)T_m$	$3T_{pm} + 2T_{pa} \approx 87.24T_m$

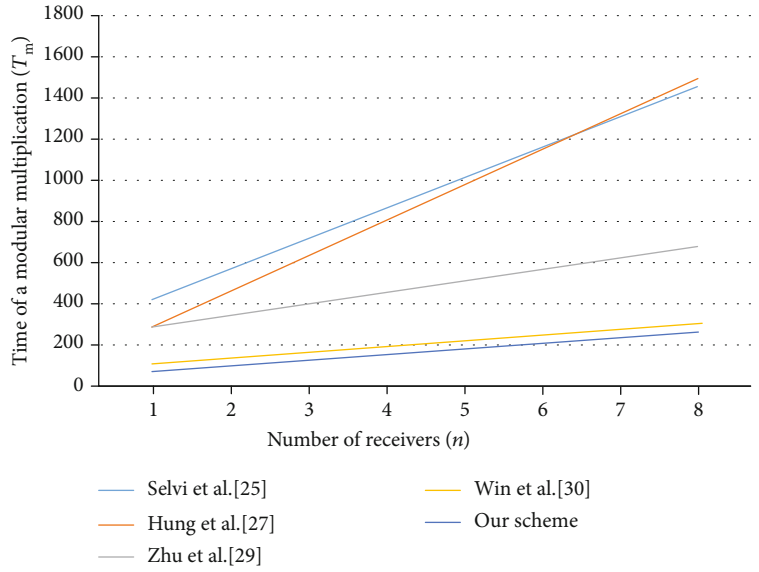

Figure 3: Efficiency in signcryption/encryption. The unit is T_m.

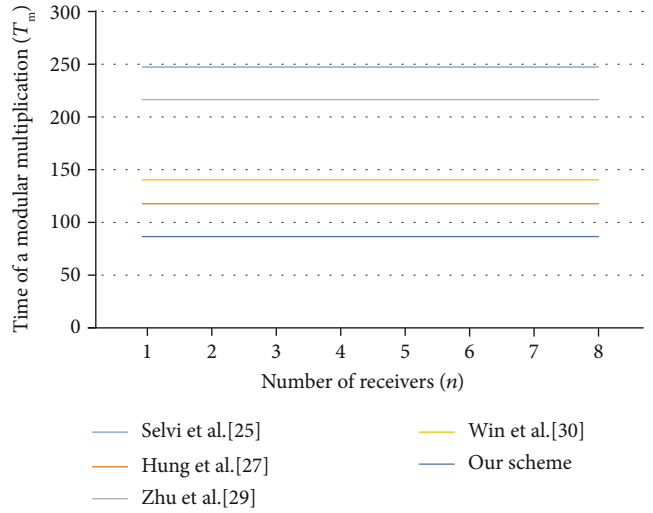

Figure 4: Efficiency in decryption/designcryption. The unit is T_m.

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].

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 SPMOOECC for encryption and decryption, and ensures that SPMOOECC 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

[1] Y. L. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) cost (signature) + cost (encryption)," in *Advances in Cryptology—CRYPTO*, vol. 1294, pp. 165–179, Springer, Berlin, Germany, 1997.

[2] J. Baek, R. Safavi-Naini, and W. Susilo, "Efficient multi-receiver identitybased encryption and its application to broadcast encryption," in *Public Key Cryptography—PKC (Lecture Notes in Computer Science)*, vol. 3386, pp. 380–397, Springer, Berlin, Germany, 2005.

[3] X. Du, Y. Wang, J. Ge, and Y. Wang, "An ID-based broadcast encryption scheme for key distribution," *IEEE Transactions on Broadcasting*, vol. 51, no. 2, pp. 264–266, 2005.

[4] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Advances in Cryptology—ASIACRYPT (Lecture Notes in Computer Science)*, vol. 4833, pp. 200–215, Springer, Berlin, Germany.

[5] Y. M. Tseng, T. T. Tsai, and Y. T. Wu, "Efficient revocable multi-receiver ID-based encryption," *Information Technology And Control*, vol. 42, no. 2, pp. 159–169, 2013.

[6] S. Duan and Z. Cao, "Efficient and provably secure multi-receiver identitybased signcryption," in *Information Security and Privacy*, vol. 4058, pp. 195–206, Springer, Berlin, Germany, 2006.

[7] S. Lal and P. Kushwah, *Anonymous ID based signcryption scheme for multiple receivers*, IACR Cryptology ePrint Archive, Las Vegas, NV, USA, 2009.

[8] L. Pang, L. Gao, H. Li, and Y. Wang, "Anonymous multi-receiver ID-based signcryption scheme," *IET Information Security*, vol. 9, no. 3, pp. 194–201, 2015.

[9] X. Zhang, C. Xu, and J. Xue, "Efficient multi-receiver identity-based signcryption from lattice assumption," *International Journal of Electronic Security and Digital Forensics*, vol. 10, no. 1, pp. 20–28, 2018.

[10] Z. Yu, Z. Jing, H. Yang, and C. Gu, "ID-based multi-receiver signcryption scheme in the standard model," *International Journal of Internet Protocol Technology*, vol. 10, no. 1, pp. 4–12, 2017.

[11] S. S. D. Selvi, S. S. Vivek, R. Srinivasan, and C. P. Rangan, "An efficient identity-based signcryption scheme for multiple receivers," in *Advances in Information and Computer Security (Lecture Notes in Computer Science)*, vol. 5824, pp. 71–88, Springer, Berlin, Germany, 2009.

[12] S. Khullar, V. Richhariya, and V. Richhariya, "An efficient identity based multi-receiver signcryption scheme using ECC," *nternational Journal of Advancements in Technolog*, vol. 2, no. 4, pp. 189–193, 2013.

[13] L. Pang, H. Li, L. Gao, and Y. Wang, "Completely anonymous multirecipient signcryption scheme with public verification," *PLoS One*, vol. 8, no. 5, article e63562, 2013.

[14] Y.-M. Tseng, T.-T. Tsai, S.-S. Huang, and H.-Y. Chien, "Efficient anonymous multi-receiver ID-based encryption with constant decryption cost," in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, pp. 131–137, Sapporo, Japan, April 2014.

[15] C.-I. Fan, L.-Y. Huang, and P.-H. Ho, "Anonymous multireceiver identity-based encryption," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1239–1249, 2010.

[16] H. Wang, Y. Zhang, H. Xiong, and B. Qin, "Cryptanalysis and improvements of an anonymous multi-receiver identity-based encryption scheme," *IET Information Security*, vol. 6, no. 1, pp. 20–27, 2012.

[17] H. Li and L. Pang, "Cryptanalysis of Wang et al.'s improved anonymous multi-receiver identity-based encryption scheme," *IET Information Security*, vol. 8, no. 1, pp. 8–11, 2014.

[18] C.-I. Fan and Y.-F. Tseng, "Anonymous multi-receiver identity-based authenticated encryption with CCA security," *Symmetry*, vol. 7, no. 4, pp. 1856–1881, 2015.

[19] L. Pang, X. Yan, H. Zhao, Y. Hu, and H. Li, "A novel multi-receiver signcryption scheme with complete anonymity," *PLoS One*, vol. 11, no. 11, article e0166173, 2016.

[20] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology-ASIACRYPT (LNCS)*, vol. 2894, pp. 452–473, Springer, Berlin, Germany, 2003.

[21] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "On the security of certificateless signature schemes from asiacrypt 2003," in *Proc. Int. Conf. Crypto. Netw. Secur., in Lecture Notes in Computer Science*, vol. 3810, pp. 13–25, Berlin, Germany: Springer, 2005.

[22] Z. F. Zhang, D. S. Wong, J. Xu, and D. G. Feng, "Certificateless public-key signature: Security model and efficient construction," in *Proc. Int. Conf. Appl. Crypto. Netw. Secur., in Lecture Notes in Computer Science*, vol. 3989, pp. 293–308, Berlin, Germany: Springer, 2006.

[23] X. Y. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signature revisited," in *Information Security and Privacy, in Lecture Notes in Computer Science*, vol. 4586, pp. 308–322, Springer, Berlin, Germany, 2007.

[24] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security - ASIACCS '08*, pp. 369–372, Tokyo, Japan, 2008.

[25] S. S. D. Selvi, S. S. Vivek, D. Shukla, and P. R. Chandrasekaran, "Efficient and provably secure CMS," in *Provable Security (Lecture Notes in Computer Science)*, vol. 5324, pp. 52–67, Springer, Berlin, Germany, 2008.

[26] S. H. Islam, M. K. Khan, and A. M. Al-Khouri, "Anonymous and provably secure certificateless multireceiver encryption without bilinear pairing," *Security and Communication Networks*, vol. 8, no. 13, 2231 pages, 2015.

[27] Y.-H. Hung, S.-S. Huang, Y.-M. Tseng, and T.-T. Tsai, "Efficient anonymous multireceiver certificateless encryption," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2602–2613, 2017.

[28] D. He, H. Wang, L. Wang, J. Shen, and X. Yang, "Efficient certificateless anonymous multi-receiver encryption scheme for mobile devices," *Soft Computing*, vol. 21, no. 22, pp. 6801–6810, 2017.

[29] J. Zhu, L.-L. Chen, X. Zhu, and L. Xie, "A new efficient certificateless multi-receiver public key encryption scheme," *International Journal of Computer Science Issues*, vol. 13, no. 6, pp. 1–7, 2016.

[30] E. K. Win, T. Yoshihisa, Y. Ishi, T. Kawakami, Y. Teranishi, and S. Shimojo, "A lightweight multi-receiver encryption scheme with mutual authentication," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 491–497, Turin, Italy, July 2017.