Research Article

Certificateless Multisignature Scheme Suitable for Network Coding

Huifang Yu,1,2 Zhewei Qi,1 Danqing Liu,3 and Ke Yang1

1School of Cyberspace Security, Xi’an University of Posts & Telecommunications, Xi’an 710121, China
2School of Information Engineering, Qinghai Communications Technical College, Xining 810003, China
3School of Computer, Qinghai Normal University, Xining 810008, China

Correspondence should be addressed to Huifang Yu; yuhuifang@qhnu.edu.cn

Received 7 August 2021; Revised 30 September 2021; Accepted 20 October 2021; Published 16 November 2021

Academic Editor: Weizhi Meng

Copyright © 2021 Huifang Yu 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.

Network coding can save the wireless network resources and improve the network throughput by combining the routing with coding. Traditional multisignature from certificateless cryptosystem is not suitable for the network coding environment. In this paper, we propose a certificateless multisignature scheme suitable for network coding (NC-CLMSS) by using the sequential multisignature and homomorphic hash function. NC-CLMSS is based on the CDH and ECDL problems, and its security is detailedly proved in the random oracle (RO) model. In NC-CLMSS, the source node generates a multisignature for the message, and the intermediate node linearly combines the receiving message. NC-CLMSS can resist the pollution and forgery attacks, and it has the fixed signature length and relatively high computation efficiency.

1. Introduction

As the network information interaction technology, the network coding [1] has routing and coding functions and allows the router to encode the received data. Network coding has the merits of high transmission efficiency, fast speed, strong robustness, and good stability, but it is vulnerable to the pollution attacks in the data transmission process. In recent years, the researchers have proposed a series of network-coding signature schemes [2–6] to solve the network coding contamination, where the schemes in [4, 5] effectively solved the replay attacks by using the time stamps; the certificateless network-coding homomorphism signature [6] is designed by using the homomorphic hash function; it can resist the replay attacks with forgery attacks at the same time and has lower computational overhead with the communication cost.

In real scenario, there are many applications to use the signature technology. With the development of communication technology, the scholars proposed many signature varieties (including multisignature) suitable for various application scenarios, such as medical field [7–10], privacy security [11], vehicle-mounted network [12, 13], multicast network [14, 15], e-government [16], e-commerce [17], and campus management facilities [18]. Multisignature first generates the partial signature of the same message, and then, the signature collector integrates the partial signatures into a signature. In terms of the order of partial signatures, multisignature can be divided into sequential multisignature [19] and broadcast multisignature [20, 21]. Compared with ordinary multisignature, the sequential multisignature has the following characteristics: (1) the signature length has nothing to do with the number of signatures; (2) instead of using the public key of each signer, the group public key can be used to verify the signature; (3) signers sign the messages in a concrete order, otherwise a valid multisignature cannot be obtained; (4) it is not computationally feasible to obtain the valid signatures without the joint operation of all signers. From now on, there is no sequential multisignature suitable for network coding, as described in Figure 1, so we will...
devise such a scheme to resist the pollution and forgery attacks in wireless networks.

1.1. Contributions. For the above reasons, a new certificateless multisignature scheme for the network coding (NC-CLMSS) is devised by combining the certificateless public key with sequential multisignature. In NC-CLMSS, the users at the source node generate the sequential multisignatures for the messages in a fixed order and transfer the signed messages from the router to the intermediate node. Intermediate node performs the linear combination of received information. Meanwhile, the destination node can verify the correctness of the signature without knowing the signer private key. Destination node filters out the contaminated information and forwards the validated data to the next receiving node. NC-CLMSS overcomes the key escrow and certificate management issues; moreover, it can resist the forgery attacks with pollution attacks in the multisource network-coding environment and has relatively better transmission efficiency.

2. Preliminaries

2.1. Bilinear Pairing. Assume $G_1$ and $G_2$ are additive and multiplication cyclic groups with the prime order $q$, respectively. $P$ is a generator of the cyclic group $G_1$, $e: G_1 \times G_1 \rightarrow G_2$ is an admissible bilinear pairing if $e$ is a map with the following properties: $e(aP, bP) = e(P, P)^{ab}$, for any $a, b \in \mathbb{Z}_q^*$, and $P \in G_1$; $e(P, P) \neq 1$; there exists an efficient algorithm to compute $e(P, Q)$, for any $P, Q \in G_1$.

Definition 1. (ECDL problem). Given $(P, aP) \in G_1$, for any $a \in \mathbb{Z}_q^*$, the ECDL (elliptic curve discrete logarithm) problem is to calculate $a \in \mathbb{Z}_q^*$. 

Definition 2. (CDH problem). Given $(P, aP, bP) \in G_1$, for any $a, b \in \mathbb{Z}_q^*$, the CDH (computational Diffie–Hellman) problem is to calculate $abP \in G_1$.

2.2. Multisource Network Model. Multisource network coding [22] has a set of source nodes. In the multisource model, each encoding message has a uniformly assigned two-dimensional index. Model for multisource transmission network is shown in Figure 2.

Multisource network coding is regarded as a directed acyclic graph $R = (E', V)$, where $E'$ is the set of edges in the network and $V$ is the set of all nodes. $U = \{u_1, u_2, \ldots, u_m\} \subset V$ is the set of the source nodes and $D = \{d_1, d_2, \ldots, d_k\} \subset V$ is the set of the sink nodes; $m$ multicast messages are expressed by $v = (v_1, v_2, \ldots, v_m)$; the source nodes’ set $U$ sends $v = (v_1, v_2, \ldots, v_m)$ to the sink nodes $D$, where each message vector $v_i$ is composed of $n$ elements over finite field $F$, where $v_i$ is written as

$$v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n}) \in F, \quad 1 \leq i \leq m.$$  

Let $j$ be the unique index uniformly assigned to each message, and the same multicast message sent by different source nodes has the same index. Each packet $w = (w_1, w_2, \ldots, w_l)$ can be sent by arbitrary intermediate node in network, and $w$ is the linear combination of $l$ messages received by this node.

2.3. Symbol Descriptions. In Table 1, the readers can see the meaning of notations relevant to this article.

3. Formal Definition

3.1. Algorithm Definition. A NC-CLMSS is defined by six polynomial time algorithms as follows.

Setup: input a security parameter $\rho$ and finally output the master key $s$ with a system parameter set $\mu$.

Extract: input $\mu$ with the user identity $ID$, and finally output a pair $(R_{s}, D_{s})$ of partial public/private keys.

KeyGen: input $\mu$ with the user identity $ID$, and finally output a pair $(x_{s}, P_{s})$ of public/private keys.
Multisignature: input \( \mu \), the master key \( s \), the message \( \nu_1 \), and finally output a signature \( \sigma \).

Combination: input the message vector \( w_1, \ldots, w_m \) and finally output a combined signature \( \sigma \).

Verification: input \( \mu \), \( \sigma_i \), and the public key \( (R_i, P_i) \), and the message \( \nu_i \); the verifier outputs a result based on the verification case.

3.2 Security Model. A NC-CLMSS must meet the existential unforgeability against the adaptive chosen-message attacks (UF-CMA). For the UF-CMA security model of NC-CLMSS, we think about the game EXP1/EXP2 between a challenger \( C \) and a polynomial time adversary \( A_1 \) or \( A_2 \).

Firstly, \( \sigma^{\text{setup}}(\rho) \xrightarrow{\sigma} A_1 \) and \( \sigma^{\text{setup}}(\rho) \xrightarrow{\sigma}, s A_2 \), where \( A_1 \) is a malicious user who can change any user public key but cannot know the master private key; \( A_2 \) is a malicious KGC who knows the system master key but cannot change any user public key. After that, \( A_1 \) or \( A_2 \) carries out the adaptive queries as follows:

\[
\sigma^{\text{partialprivatekey}}(ID_i) \xrightarrow{D_i} \begin{cases} A_1 & \text{if } P_i \text{ was not replaced} \\ A_2 & \text{otherwise} \end{cases}
\]

\[
\sigma^{\text{publickey}}(ID_i) \xrightarrow{(R_i, P_i)} \begin{cases} A_1 & \text{if } P_i \text{ was not replaced} \\ A_2 & \text{otherwise} \end{cases}
\]

\[
\sigma^{\text{replacepublickey}}(ID_i) \xrightarrow{(R_i', P_i')} A_1,
\]

Table 1: Meaning of the various notations.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>KGC</td>
<td>The key generation center</td>
</tr>
<tr>
<td>( \rho )</td>
<td>A security parameter</td>
</tr>
<tr>
<td>( \eta )</td>
<td>The public parameter set</td>
</tr>
<tr>
<td>IDi</td>
<td>Some user identity</td>
</tr>
<tr>
<td>Ni</td>
<td>User ( N_i \ (i \in {1, 2, \ldots, n}) )</td>
</tr>
<tr>
<td>S</td>
<td>The master secret key</td>
</tr>
<tr>
<td>si</td>
<td>The full private key of a user</td>
</tr>
<tr>
<td>PKi</td>
<td>The full public key of a user</td>
</tr>
<tr>
<td>Di</td>
<td>The partial private key of a user</td>
</tr>
<tr>
<td>Ri</td>
<td>The partial public key of a user</td>
</tr>
<tr>
<td>Pi</td>
<td>The user public key</td>
</tr>
<tr>
<td>P_{pub}</td>
<td>The system public key</td>
</tr>
<tr>
<td>Hi ( (i = 0, 1, 2) )</td>
<td>Hash function ( (i = 0, 1, 2) )</td>
</tr>
<tr>
<td>( \sigma_i )</td>
<td>A signature of the message ( \nu_i )</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Multisignature</td>
</tr>
</tbody>
</table>

Figure 2: Model for multisource transmission network.
4.2 Extract. \( \mathcal{C}_{\text{multisignature}}(v_i) \xrightarrow{\sigma_i} A_1 \xrightarrow{A_2} \)

\( \mathcal{C}_{\text{combination}}(\sigma_i) \xrightarrow{\sigma_i} A_1 \xrightarrow{A_2} \)

\( \mathcal{C}_{\text{verification}}(\sigma, \sigma_i) \) message or \( A_1 \xrightarrow{A_2} \)

Finally, \( A_1/A_2 \) outputs a forged signature \( \sigma^* \). In the adaptive queries, \( A_1 \) should not request the full private key of \( ID_i \); \( A_2 \) cannot request the private key of \( ID_n \). In addition, \( \sigma^* \) should not be returned by any multisignature oracle. \( A_1/A_2 \) wins in EXP1/EXP2 if \( \mathcal{C}_{\text{verification}}(\sigma, \sigma_i) \xrightarrow{\text{not}} A_1/A_2 \).

Assume Adv (\( \rho \)) denotes the adversary advantage in EXP1/EXP2; then, Adv (\( \rho \)) is defined as the probability which \( A_1/A_2 \) succeeds in EXP1/EXP2.

**Definition 3.** A NC-CLMSS is said to be UF-CMA secure if no polynomial time adversary \( A_1/A_2 \) succeeds in EXP1/EXP2 with a non-negligible advantage.

### 4. NC-CLMSS Instance

4.1 Setup. Given a security parameter \( \rho \), KGC (key generation center) chooses cyclic groups \( G_1 \) and \( G_2 \) with the prime order \( q \), as described in Section 2.1. \( P \) is a generator of \( G_1 \) and \( e \): \( G_1 \times G_1 \rightarrow G_2 \). KGC selects secure hash functions: \( H_0: \{0, 1\}^* \times G_1 \rightarrow Z_q^* \), \( H_1: \{0, 1\}^* \rightarrow Z_q^* \), \( H_2: \{0, 1\}^* \rightarrow G_1 \). KGC chooses a master key \( s \in \mathbb{G}_q^* \) and maintains its secret and then calculates the system public key \( P_{\text{pub}} = sP \). Finally, KGC publishes the system parameter set: \( \mu = \{G_1, G_2, q, P, e, P_{\text{pub}}, H_0, H_1, H_2\} \).

4.2 Extract. Given the identity \( ID_i \) of the user \( N_i \) and \( \mu \), KGC randomly chooses \( r_i \in Z_q^* \) and calculates \( R_i = r_iP \), \( h_i = H_2(ID_i, R_i) \), and \( D_i = r_i + h_is \), where \( ID_i \in \{ID_1, ID_2, \ldots, ID_n\} \). \( D_i \) is the partial private key of \( N_i \) and \( R_i \) is the partial public key of \( N_i \).

4.3 KeyGen. Given the identity \( ID_i \) of the user \( N_i \) and \( \mu \), this user \( N_i \) (\( i \in \{1, 2, \ldots, n\} \)) randomly chooses a secret value \( x_i \in Z_q^* \) and calculates the public key \( P_i = x_iP \). Note that \( ID_i \in \{ID_1, ID_2, \ldots, ID_n\} \). \( PK_i = (P_i, R_i) \) is the full public key of \( N_i \), and \( s_i = (x_i, D_i) \) are the full private key of \( N_i \).

4.4 Multisignature. Assume \( L = \{ID_1, ID_2, \ldots, ID_n\} \) is an identity set of \( n \) users and \( N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_n \) denotes the signature sequence of \( n \) users. In other words, the user \( N_i \) (\( i \in \{1, 2, \ldots, n\} \)) signs the message \( v_i \) with the sequence \( N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_n \). Initially, \( N_1 \) calculates

\[
I_1 = H_1(ID_1, L, P_1, R_1, v_1),
\]

\[
T = H_2(v_1, L, P_{\text{pub}}),
\]

\[
\sigma_1 = \text{SIGN}_1 = (I_1x_1 + D_1)T,
\]

where \( \sigma_i \) is the partial signature of the message \( v_i \) from the user \( N_i \). User \( N_1 \) delivers \( (v_1, \sigma_1) \) to the user \( N_2 \). After receiving \( (v_i, \sigma_{i-1}) \), the user \( N_i \) (\( i = 2, 3, \ldots, n \)) calculates

\[
I_j = H_1(ID_j, L, P_j, R_j, v_j),
\]

\[
h_j = H_2(ID_j, R_j), \quad 1 \leq j \leq i - 1,
\]

\[
T = H_2(v_i, L, P_{\text{pub}}).
\]

If the equality \( e(\sigma_{i-1}, P) = e(T, \sum_{j=1}^{i-1} (I_jP_j + R_j + h_jP_{\text{pub}})) \) holds, the user \( N_i \) calculates

\[
I_j = H_1(ID_j, L, P_j, R_j, v_j), \quad (1 \leq i \leq n),
\]

\[
\sigma_i = \sigma_{i-1} + \text{SIGN}_i = \sigma_{i-1} + (I_iX_i + D_i)T
\]

Then, the signature of the user \( N_i \) is \( \sigma_n = \sum_{i=1}^{n} \text{SIGN}_i \).

Finally, \( \alpha = (\xi_0, \xi_1, \ldots, \xi_m) \) and \( \beta = (\beta_1, \beta_2, \ldots, \beta_m) \), the intermediate node combines the message vector as follows:

\[
w = \sum_{i=1}^{m} \xi_i \beta_i.
\]

Then, the message vector \( v_i \) is also denoted as \( w = \sum_{i=1}^{m} \xi_i \beta_i \), and the signature corresponding to the message vector \( w \) is \( \sigma = (\sigma_1, \sigma_2, \ldots, \sigma_d) \). Signature process corresponding to \( w \) is \( \sigma_j = \sum_{i=1}^{m} \xi_i \beta_i \), where \( \sigma_{i,j} \) (\( 1 \leq i \leq m, 1 \leq j \leq l \)) represents the \( j \)-th element of \( \sigma \). Finally, the intermediate node outputs the combined signature \( \sigma = \prod_{i=1}^{m} \sigma_0 \).

4.6 Verify. After receiving the multisignature and combination signature, the verifier calculates \( I_1 = H_1(ID_1, L, P_1, R_1, v_1) \) (\( 1 \leq i \leq n \)) and \( T = H_2(v_1, L, P_{\text{pub}}) \).

If the equality \( e(\sigma_n, P) = e(T, \sum_{i=1}^{m} (I_iP_i + R_i + h_iP_{\text{pub}})) \) holds, the multisignature is valid and invalid otherwise.
5. Correctness Analysis

5.1. Single Signature Verification. Given the signature $\sigma_i$ of the message $v_i$, then the signature verification process of the user $N_i$ ($i \in \{1, 2, \ldots, n\}$) is as follows:

$$e(\sigma_n, P) = e \left( \prod_{j=1}^{n} (l_i x_i + D_i), t, P \right)$$

$$= e \left( T, \sum_{i=1}^{n} (l_i x_i + D_i), P \right)$$

$$= e \left( T, \sum_{i=1}^{n} (l_i P_i + R_i + h_i P_{pub}) \right).$$

5.2. Combination Verification. Given the message ($v_1, \ldots, v_m$) and global coding vector $\beta = (\beta_1, \ldots, \beta_m)$, $w = \sum_{j=1}^{m} \beta_j v_j$ is the message vector received by the intermediate node, and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i)$ is the multisignature corresponding to $w$. In the verification phase, it is necessary to check the correctness of the following equality:

$$e(\sigma_n, P) = e \left( \prod_{j=1}^{n} (l_i P_i + R_i + h_i P_{pub}) \right).$$

where $h_i = H_0(1D_i, R_i)$. In the multisource network coding, the intermediate nodes combine the messages from different source nodes and form a combination signature. Different source nodes may send the same message. In order to distinguish the possible combination of the same message vector, the global coding vector is expressed as $\beta_j = \sum_{k=1}^{d} \beta_{j,k} (u_k)$, where the global coding vector $\beta_{j,k} (u_k) \in \beta_j (u_k), \ldots, \beta_j (u_d)$ and source node user $u_k \in \{u_1, u_2, \ldots, u_d\}$. Then, the message vector is expressed as $w = \sum_{j=1}^{m} \sum_{k=1}^{d} \beta_{j,k} (u_k) v_j$. Hence, the multisignature of message vector $w$ can be expressed as $\sigma = \prod_{j=1}^{m} \prod_{k=1}^{d} (\sigma_j (u_k) \beta_{j,k} (u_k)$, and then, the i-th component in the multisignature can be expressed as $\sigma_i = \prod_{j=1}^{m} \prod_{k=1}^{d} (\sigma_j (u_k) \beta_{j,k} (u_k)$ where $\sigma_j (u_k)$ is the i-th component of multisignature $\sigma_j (u_k)$. Then, the relevant equality is verified as follows:

$$e(\sigma_n, P) = e \left( \prod_{j=1}^{n} (l_i x_i + D_i), t, P \right)$$

$$= e \left( \prod_{j=1}^{n} \prod_{k=1}^{d} (l_i, x_i + D_i), t, P \right)$$

$$= e \left( \prod_{j=1}^{n} \prod_{k=1}^{d} \beta_{j,k} \cdot x_i + D_i, t, P \right)$$

$$= e \left( \prod_{j=1}^{n} \prod_{k=1}^{d} \beta_{j,k} \cdot x_i + D_i, t, P \right)$$

From the verification process of single message, we know $e(\sigma_n, P) = e(T, \sum_{i=1}^{n} (l_i P_i + R_i + h_i P_{pub}))$. Then, the verification process is denoted as

$$e \left( \prod_{j=1}^{n} \prod_{k=1}^{d} \beta_{j,k} \cdot P_i + R_i + h_i P_{pub} \right)$$

$$= e \left( T, \sum_{i=1}^{n} (l_i P_i + R_i + h_i P_{pub}) \right).$$

6. Security Analysis

Theorem 1. In the RO model, if the polynomial time adversary $A_1$ can break the UF-CMA-1 security of NC-CLMSS, a challenge algorithm $C$ can solve the CDH (computational Diffie–Hellman) problem.

Proof. $C$ receives a random instance $(P, aP, bP) \in G_1$ of CDH problem, and its aim is to use $A_1$ (the subroutine of $C$) to calculate $abP \in G_1$. $C$ maintains the initially empty lists $L_0, L_1, L_2$, and $L_3$ to store the query-answer values of several oracles. Firstly, $\mathcal{C}_2^{\text{ret}}(p) \rightarrow \mathcal{O}_1$, where $p = \text{aP}$. Then, $A_1$ adaptively issues the polynomial time queries as follows:

$H_0$ queries: $A_1$ issues an $H_0$ query. $C$ outputs $h_1$ to $A_1$ if the relevant tuple is in the list $L_0$; otherwise, $C$ returns a random $h_i \in \mathbb{Z}_n^*$ and stores $(ID_i, R_i, h_i)$ in $L_0$.

$H_1$ queries: $A_1$ issues an $H_1$ query. $C$ returns $l_1$ if a matching tuple is in the list $L_1$; otherwise, $C$ returns $l_1 \in \mathbb{Z}_n^*$ and stores $(ID_i, L, P_i, R_i, v_i, l_1)$ in $L_1$.

$H_2$ queries: $A_1$ issues an $H_2$ query. If it is not the $\theta$-th query $(\theta \in \{1, 2, \ldots, q_0\})$ ($q_0$ is the query times relevant to the $H_0$ oracle) and a matching tuple is in the list $L_2$, $C$ outputs $T = LP (I \in \mathbb{Z}_n^*)$ and stores $(v_i, l_i, P_{pub}, T)$ in $L_2$; otherwise, $C$ returns $T = bP$ and stores $(v_i, l_i, P_{pub}, \cdot, T)$ in $L_2$.

Partial private key queries: $A_1$ requests a partial private key of $ID_i$. If it is not the $\theta$-th query, $C$ chooses $r_1 \in \mathbb{Z}_n^*$ to calculate $R_i = r_1P$ such that $D_i$ satisfies $D_iP = R_i + h_i P_{pub}$ and finally returns $D_i$ as the answer and stores $(ID_i, r_i, R_i, D_i^{\theta \rightarrow \bot})$ in the list $L_3$; otherwise, $C$ fails and aborts the game.

Public key queries: $A_1$ requests a public key of $ID_i$. $C$ calculates $P_i = x_i P (x_i \in \mathbb{Z}_n^*)$ and finally returns $PK_i = (R_i, P_i)$ and updates the list $L_3$ with $(ID_i, r_i, R_i, D_i^{\theta \rightarrow \bot})$.

Secret value queries: $A_1$ requests a secret value of $ID_i$. $C$ returns $x_i$ from $L_3$ if the corresponding public key has not been replaced.

Public key replacement: if it is not the $\theta$-th query, the public key of $ID_i$ is replaced by $A_1$; otherwise, $C$ fails and aborts the game.

Multisignature queries: for a multisignature query of message $v_i$, $C$ runs the relevant algorithm and returns a result if it is not the $\theta$-th query; otherwise, $C$ signs $v_i$. 

Security and Communication Networks
with the sequence $N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_n$. Firstly, C calculates for $N_1$ as follows:

$$
\begin{align*}
I_1 &= H_1(ID_1, L, P_1, R_1, v_1), \\
T &= H_2(v_1, L, P_{\text{pub}}), \\
\sigma_1 &= \text{SIGN}_1 \\
&= (l_1x_1 + D_1)T,
\end{align*}
$$

where $\sigma_1$ is the partial signature of $v_1$ for $N_1$. Then, C calculates for $N_i$ ($i \in \{1, 2, \ldots, n\}$) relevant to $(v_i, \sigma_{i-1})$ as follows:

$$
\begin{align*}
I_j &= H_1(ID_j, L, P_j, R_j, v_j), \\
h_j &= H_2(ID_j, R_j), \\
1 \leq j \leq i - 1, \\
T &= H_2(v_i, L, P_{\text{pub}}).
\end{align*}
$$

If $e(\sigma_{i-1}, P) = e(T, \sum_{j=1}^{i-1} (l_jP_j + R_j + h_jP_{\text{pub}}))$ holds, C calculates for $N_i$ as follows:

$$
\begin{align*}
I_i &= H_1(ID_i, L, P_i, R_i, v_i), \\
\sigma_i &= \sigma_{i-1} + \text{SIGN}_i \\
&= \sigma_{i-1} + (l_ix_i + D_i)T.
\end{align*}
$$

Finally, C calculates $\sigma_n = \sum_{i=1}^{n} \text{SIGN}_i$ and delivers $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ which is sent to $A_1$.

Combination queries: $A_1$ requests a combination query. For the local coding vector $\alpha = (a_1, \ldots, a_m)$, global vector $\beta = (b_1, \ldots, b_m)$, and message vector $(w_1, w_2, \ldots, w_m)$, C combines the message vector $w = \sum_{i=1}^{m} a_iw_i$. Then, the message vector is also denoted as $w = \sum_{i=1}^{m} b_i^\prime v_i$, and the signature process relevant to $w$ is $\sigma_j = \sum_{i=1}^{m} h_{i,j}^\alpha$, where $\sigma_j, j (1 \leq i \leq m$ and $1 \leq j \leq l)$ denotes the j-th element of $\sigma_n$. Finally, C outputs a combined signature $\sigma = \prod_{i=1}^{m} \sigma_i^\beta$.

Verification queries: $A_1$ requests a verification query. C runs the verification algorithm and returns a result if it is not the $\theta$-th query; otherwise, C calculates $I_l = H_1(ID, L, P, R, v_l)$ ($1 \leq i \leq n$) and $T = H_2(v_l, L, P_{\text{pub}})$. If the equality $e(\sigma_n, P) = e(T, \sum_{i=1}^{n} (l_iP_i + R_i + h_iP_{\text{pub}}))$ holds, C returns $\sigma$ and $\theta$.

Finally, $A_1$ outputs a forgery signature $\sigma^*$. In the adaptive queries, $A_1$ cannot request a full private key of $ID_o$ and $\sigma^*$ is not returned by any multisignature oracle. If it is not the $\theta$-th query, C fails and aborts the game; otherwise, C calls the $H_0$, $H_1$, and $H_2$ oracles and then searches the list $L_s$. Finally, C verifies the following equality:

$$
e(\sigma_n^*, P) = e(T^*, \sum_{i=1}^{n} (l_i^*P_i + R_i^* + h_iP_{\text{pub}}))
$$

$$
= e(bP, \sum_{i=1}^{n} (l_iP_i^* + R_i^*) + \sum_{i=1}^{n} h_i^*aP)
$$

$$
= e(\sum_{i=1}^{n} (l_i^*x_i^*bP + r_i^*bP) + \sum_{i=1}^{n} h_i^*abP, P).
$$

From the above equality, we can obtain the solution of CDH problem:

$$
\sigma_n^* = \sum_{i=1}^{n} (l_i^*x_i^*bP + r_i^*bP) + \sum_{i=1}^{n} h_i^*abP,
$$

$$\Rightarrow abP = \frac{\sigma_n^* - \sum_{i=1}^{n}(l_i^*x_i^*bP + r_i^*bP)}{\sum_{i=1}^{n} h_i^*}.
$$

**6.1. Probability Estimation.** Probability that $C$ succeeds in the above-mentioned game is estimated as follows. Here, it is necessary to think about three events:

- $E_1$ is the event that $C$ does not abort the game.
- $E_2$ is the event that $A_1$ successfully forge a signature.
- $E_3$ is the event that there exists at least one record of nontarget identity in successful forgery case.

In $E_1$, there exists one time not querying the target identity, and then, $Pr[E_1] \geq \frac{1}{l_1 + l_s}$, where $l_1$ is the times of secret value query and $l_s$ is the query times of public key replacement, $E_2$ denotes that $A_1$ wins in the game, then $Pr[E_2|E_1] \geq \epsilon$, and $E_3$ at least occurs once time in $n$ queries, then $Pr[E_3|E_1 \land E_2] \geq 1/n$. Hence, the success probability that $C$ solves the CDH problem is:

$$
\epsilon' = Pr[E_1 \land E_2 \land E_3]
$$

$$
= Pr[E_1] \cdot Pr[E_2|E_1] \cdot Pr[E_3|E_1 \land E_2]
$$

$$
\geq \frac{\epsilon}{n \cdot (l_1 + l_s)}
$$

**Theorem 2.** In the RO model, if the polynomial time adversary $A_2$ can break the UF-CMA-II security of NC-CLMSS, a challenge algorithm $C$ must be able to solve the CDH problem.

Proof. $C$ receives a random instance $(P, aP, bP) \in G_1$ of CDH problem, and its aim is to utilize $A_2$ (the subroutine of $C$) to determine the value of $abP \in G_1$. $C$ maintains the initially empty lists $L_0$, $L_1$, $L_2$, and $L_3$ to save the query-answer values of several oracles. Firstly, $\sigma_C^{\text{setup}}(\rho) \rightarrow s, A_2.$
Then, $A_2$ adaptively performs the polynomial time queries as below:

$H_0$ queries: for an $H_0$ query, if $(ID_0, R_0, h_0)$ is in the list $L_0$, $C$ returns $h_0$; otherwise, $C$ returns $h_\ell \in \mathbb{Z}_q^*$ and stores $(ID_0, R_0, h_\ell)$ in $L_0$.

$H_1$ queries: for an $H_1$ query, if the matching tuple is in the list $L_1$, $C$ returns $l_i$; otherwise, $C$ returns $l_i \in \mathbb{Z}_q^*$ and stores $(ID_0, l_i, P_k, R_i, v_\ell, l_i)$ in $L_1$.

$H_2$ queries: for an $H_2$ query, if it is not the $\theta$-th query ($\theta \in \{1, 2, \ldots, q_0\}$ ($q_0$ is the query times relevant to $H_0$ oracle) and the relevant tuple is in the list $L_2$, $C$ randomly outputs $T = lP \in G_1$ ($l \in \mathbb{Z}_q^*$) as the answer; after that, $C$ stores $(v_\ell, l, P_{pub}, l, T)$ in $H_2$, otherwise, $C$ returns $T = bP \in G_1$ and stores $(v_\ell, l, P_{pub'}, l)$ in $L_2$.

Partial private key queries: for a partial private key query for identity $ID_0$, $C$ calculates $R_i = rP, D_i = r_i + h_\ell s$ ($r_i \in \mathbb{Z}_q^*$) and returns $D_i$ and stores $(ID_0, r_i, R_i, D_i, \cdot, \cdot)$ in the list $L_3$.

Public key queries: for a public key query for identity $ID_0$, if it is not the $\theta$-th query, $C$ calculates $P_i = xP$ ($x \in \mathbb{Z}_q^*$) and finally returns $PK_i = (R_i, P_i)$ and updates $L_3$ with $(ID_0, r_i, R_i, D_i, xP)$; otherwise, $C$ returns $PK_i = (R_i, P_i)$ and updates $L_3$ with $(ID_0, r_i, R_i, D_i, \cdot, \cdot)$.

Signature queries: $A_2$ issues a multisignature query for message $v_\ell$. If it is not the $\theta$-th query, $C$ runs the multisignature algorithm to output a result; otherwise, $C$ signs $v_\ell$ with the sequence $N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_n$. Firstly, $C$ calculates for $N_1$ as follows:

\[
\begin{align*}
l_1 & = H_1(ID_0, L, P_k, R_1, v_\ell), \\
T & = H_2(v_\ell, l, P_{pub}), \\
\sigma_1 & = \text{SIGN}_1 \\
& = l_1P_1 + D_1P_1,
\end{align*}
\]

where $\sigma_1$ is the partial signature of $v_\ell$ for $N_1$. Then, $C$ calculates for $N_i$ ($i \in \{1, 2, \ldots, n\}$) relevant to $(v_\ell, \sigma_{i-1})$ as follows:

\[
\begin{align*}
l_j & = H_1(ID_0, L, P_k, R_j, v_\ell), \\
h_j & = H_0(ID_0, R_j), \quad (1 \leq j \leq i - 1), \\
T & = H_2(v_\ell, l, P_{pub}).
\end{align*}
\]

If $e(\sigma_{i-1}, P) = e(T, \sum_{j=1}^{i-1} (l_1P_j + R_j + h_\ell P_{pub}))$ holds, $C$ calculates for $N_i$ as follows: $l_i = H_1(ID_0, L, P_k, R_i, v_\ell), \quad \sigma_i = \sigma_{i-1} + \text{SIGN}_1 = \sigma_{i-1} + (l_1P_i + D_1P_1)$. $C$ calculates $\sigma_n = \sum_{i=1}^{n} \text{SIGN}$, and returns $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_l)$. Combination queries: $A_2$ issues a combination query. For the local coding vector $\alpha = (\alpha_1, \ldots, \alpha_m)$, global vector $\beta = (\beta_1, \ldots, \beta_m)$ and message vector $w = (w_1, w_2, \ldots, w_m)$, $C$ combines the message vector $w = \sum_{j=1}^{m} w_j v_\ell$. Then, the message vector is also denoted as $w = \sum_{j=1}^{m} w_j v_\ell$, and the signature corresponding to $w$ is $\sigma_j = \sum_{i=1}^{m} \sigma_{j,i}$, where $\sigma_{j,i}$ ($1 \leq i \leq m$ and $1 \leq j \leq l$) denotes the $j$-th element of $\sigma_i$. Finally, $C$ outputs a combined signature $\sigma = \prod_{i=1}^{m} \sigma_{j,i}$.

Verification queries: for a verification query, $C$ runs the verification algorithm and returns a result if it is not the $\theta$-th query; otherwise, $C$ calculates $l_i = H_1(ID_0, L, P_k, R_i, v_\ell)$ ($1 \leq i \leq n$) and $T = H_2(v_\ell, l, P_{pub})$. If $e(\sigma_{n, P}) = e(T, \sum_{i=1}^{n} (l_1P_i + R_i + h_\ell P_{pub}))$ holds, $C$ returns $\sigma$ and $\perp$ otherwise.

Finally, $A_2$ outputs a forgery signature $\sigma^*$. In queries, $A_2$ cannot query the secret value of $ID_0$ and $\sigma^*$ is not returned by signature oracle. If it is not the $\theta$-th query, $C$ fails and aborts the game; otherwise, $C$ calls the $H_0, H_1,$ and $H_2$ oracles and then searches the list $L_3$ and then verifies as follows:

\[
e(\sigma_{n, P}) = e\left(T, \sum_{i=1}^{n} (l_1P_i^* + R_i^* + h_\ell P_{pub})\right)
\]

\[
e(\sigma_{n, P}) = e\left(bP, \sum_{i=1}^{n} (l_1P_i^* + R_i^* + h_\ell P)\right)
\]

\[
e\left(\sum_{i=1}^{n}(l_1P_i^* + R_i^* + h_\ell P_{pub}), P\right).
\]

CDH problem solution can be obtained from equation (19):

\[
\sigma_n^* = \sum_{i=1}^{n}(l_1P_i^* + R_i^* + h_\ell P_{pub}),
\]

\[
\Rightarrow abP = \sigma_n^* - \sum_{i=1}^{n}(l_1P_i^* + R_i^* + h_\ell P_{pub}).
\]

6.2. Probability Analysis. Probability that $C$ succeeds in the above game is analyzed as follows. Here, we need to think about three events: $E_3$ is the event that $C$ does not abort the game. In $E_3$, there exists one time not querying the target identity, and then, $Pr[E_3] \geq 1/l_0$, where $l_0$ is the times of secret value query. $E_2$ is the event that $A_2$ successfully forge a signature. $E_2$ denotes $A_2$ wins in the game, then $Pr[E_2|E_3] \geq \epsilon$. $E_3$ is the event that there exists at least one record of nontarget identity in successful forgery case. $E_3$ at least occurs once in $n$ queries, then $Pr[E_3|E_3] \geq 1/n$. Hence, the probability that $C$ solves the CDH problem is

\[
\epsilon' = Pr[E_1 \land E_2 \land E_3]
\]

\[
\epsilon' = Pr[E_1] \cdot Pr[E_2|E_1] \cdot Pr[E_3|E_1 \land E_2]
\]

\[
\geq \frac{\epsilon}{n \cdot l_0}.
\]

Theorem 3. Our NC-CLMSS can prevent the pollution attacks in the multisource network coding environment.

Proof. In NC-CLMSS, the signature process takes place at the source node and intermediate node. For the source node, the attacker can capture any node in the network and uses it.
to launch the attacks; this node sends the polluted information to the next node, but it is equivalent to solving the elliptic curve discrete logarithm (ECDL) problem for the attacker obtaining the signer private key from the public key. For the intermediate nodes, the attacker captures the signature from source node and tries to forge a signature; then, the attacker must own the user private key, and it is also equivalent to solving the ECDL problem.

NC-CLMSS can resist the pollution attacks in the network-coding environment because solving the ECDL problem is computationally infeasible.

### 7. Performance Analysis

In this section, the performance comparison is made between NC-CLMSS and existing schemes in [19–21] based on the computational complexity. Schemes in [19–21] cannot resist the pollution attacks; the schemes in [20, 21] are not sequential multisignature. Our NC-CLMSS is a sequential multisignature and can resist pollution attacks. Table 2 describes the time complexity of main cryptography operations. Experimental environment for the performance analysis in this section: the processor is Intel (R) Core (TM) i7-6700HQ CPU @2.60GHz; the system type is the 64-bit operating system. Based on this system, we use C programming language, PBC library, and OpenSSL program to obtain the cryptography operation time, as shown in Table 2.

Table 3 describes the computational efficiency of several schemes. From Table 3, the computational complexity of NC-CLMSS is relatively lower than other schemes in [19–21]. Simulation curves of signature time-consuming of comparison schemes are shown in Figure 3. Simulation curves of verification time-consuming comparison are shown in Figure 4. Simulation curves of total algorithm time comparison are shown in Figure 5. Assume the number $n$ of signature members is 10, 20, 30, 40, 50, and 60, respectively. Experiment results show the running time of different schemes increases linearly with the increase of the number of signed members. As shown in Figure 3, in the signature

<table>
<thead>
<tr>
<th>References</th>
<th>Signature time ($ms$)</th>
<th>Verification time ($ms$)</th>
<th>Total time ($ms$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xu and Wang scheme</td>
<td>$3nC_{\text{mtp}} + 2nC_{\text{mul}} + 2nC_{\text{par}}$</td>
<td>$3nC_{\text{mtp}} + 2nC_{\text{par}}$</td>
<td>$6nC_{\text{mtp}} + 2nC_{\text{mul}} + 4nC_{\text{par}}$</td>
</tr>
<tr>
<td>Niu et al. scheme</td>
<td>$(2n + 1)C_{\text{mtp}} + 2nC_{\text{mul}} + 2nC_{\text{par}}$</td>
<td>$(2n + 1)C_{\text{mtp}} + 3nC_{\text{par}}$</td>
<td>$2(2n + 1)C_{\text{mtp}} + 2nC_{\text{mul}} + 5nC_{\text{par}}$</td>
</tr>
<tr>
<td>Tanwar et al. scheme</td>
<td>$3nC_{\text{mtp}} + 4nC_{\text{mul}}$</td>
<td>$3nC_{\text{mtp}} + 2nC_{\text{mul}}$</td>
<td>$6nC_{\text{mtp}} + 6nC_{\text{mul}}$</td>
</tr>
<tr>
<td>NC-CLMSS</td>
<td>$(2n + 1)C_{\text{mtp}} + 2nC_{\text{mul}} + 2nC_{\text{par}}$</td>
<td>$(2n + 1)C_{\text{mtp}} + 2nC_{\text{par}}$</td>
<td>$2(2n + 1)C_{\text{mtp}} + 2nC_{\text{mul}} + 4nC_{\text{par}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cryptographic operations</th>
<th>Operation time ($ms$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to perform an exponential operation: $C_{\text{me}}$</td>
<td>7.5</td>
</tr>
<tr>
<td>Time to perform a scalar multiplication: $C_{\text{mul}}$</td>
<td>1.56</td>
</tr>
<tr>
<td>Time to perform a hash operation: $C_{\text{mtp}}$</td>
<td>17.2</td>
</tr>
<tr>
<td>Time to perform a bilinear operation: $C_{\text{par}}$</td>
<td>19.7</td>
</tr>
</tbody>
</table>

| Table 2: The time complexity of cryptographic operations. |

| Table 3: Computational efficiency comparison of several schemes. |

Figure 3: Comparison of the signature time.

Figure 4: Comparison of the verification time.
phase, the growth rate of NC-CLMSS is relatively slower than other schemes. From Figure 4, the computational efficiency of NC-CLMSS is the highest. In terms of total time in Figure 5, NC-CLMSS takes the least time. Hence, NC-CLMSS is a relatively better cryptography algorithm in several schemes.

8. Summary

Network encoding cryptography has many merits, but there exists the inevitable problem how to resist the pollution attacks and forgery attacks in the message transmission process. By using the techniques of the certificateless multisignature and multisource network coding cryptosystem, we construct a certificateless multisignature scheme suitable for network coding (NC-CLMSS). Under the ECDL and CDH assumptions, this algorithm is proved to satisfy the UF-CMA security and can resist the pollution attacks; its computational complexity is relatively lower.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors’ Contributions

Huifang Yu worked on the security model, instance design, and security proof; Zhewei Qi worked on the instance design and simulation experiment; Danqing Liu worked on the introduction and formal algorithm definition; Ke Yang estimated the probability.

Acknowledgments

This work was supported by the Planning Subject for the 13th Five-Year Plan of National Education Science, under Grant EJA180476.

References


