Certificateless Network Coding Scheme from Certificateless Public Auditing Protocol

Genqing Bian

Xi'an University of Architecture and Technology

Mingxuan Song (✉ songmingxuan@xauat.edu.cn)

Xi'an University of Architecture and Technology <https://orcid.org/0000-0002-8061-2212>

Bilin Shao

Xi'an University of Architecture and Technology

Research Article

Keywords: Certificateless network coding, Certificateless public auditing, Homomorphic signature

Posted Date: November 8th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1006227/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Certificateless Network Coding Scheme from Certificateless Public Auditing Protocol

Genqing Bian, Mingxuan Song* and Bilin Shao

Abstract—In recent years, network coding has received extensive attention and been applied to various computer network systems, since it has been mathematically proven to enhance the network robustness and maximize the network throughput. However, it is well-known that it is extremely vulnerable under pollution attacks. Certificateless network coding scheme (CLNS) is a recently-proposed mechanism to defend against pollution attacks for network coding, which avoids the tedious management of certificates and the key-escrow attack. Until now, only a few constructions were presented and more ones should be given in order to enrich this field. In this paper, for the first time, we study the general construction of CLNS from certificateless public auditing protocol (CL-PAP), although the two areas seem to be quite different in their nature and are studied independently. Since there are many candidates of CL-PAPs, we naturally obtain abundant constructions of CLNSs according to our systematic way. In addition, in order to show the power of our general construction, we also present a concrete implementation given a specific CL-PAP. The performance analysis and experimental result show that the implemented CLNS is competitive in the existing network coding schemes.

Index Terms—Certificateless network coding, Certificateless public auditing, Homomorphic signature.

I. INTRODUCTION

THE traditional way of data transmission is based on storing-and-forwarding routing mechanism. For the long time, it has been generally believed that processing the transmitted data on the intermediate node will not produce any benefits. Until the network coding was first proposed by Ahlswede *et al.* [1] in 2000, it completely overturned the traditional view and established its important position in modern network communication research. Network coding (NC) is an information exchange scheme that combines routing and coding, which has the advantages of optimized throughput, stronger robustness and lower power consumption. However, it is highly vulnerable to pollution attacks, because the intermediate nodes will combine polluted packets with other honest packets during the transmission, then destination nodes will waste resources to decode incorrect packets.

Linearly homomorphic signature can effectively solve pollution attacks, intermediate nodes are allowed to verify packets and discard polluted packets in transmission [5]. Boneh *et al.* [2] applied homomorphic signature scheme in network

coding, which can effectively defend against pollution attacks by introducing the CDH assumption, random oracle model and bilinear map. In this mechanism, all nodes have to acquire the public key of the source node rather than share a same private key.

In traditional certificate-based network coding, in order to correlate user's public key to his identity, the most popular method is to generate a user's digital certificate by certificate authority (CA). In order to confirm the relationship between the public key and identity of source nodes, other nodes must verify the validity of the corresponding certificate before using this public key. At present, public key infrastructure (PKI) is the core and foundation of network security construction, it uses digital certificates to manage public keys and binds user's public key to his identity through CA. Because generating and managing many certificates are time-consuming, PKI is costly to deploy and cumbersome to use in practice. Therefore, additional validation of each certificate with CA and PKI in network coding will significantly degrade its performance.

To solve the certificate management problem, Shamir *et al.* [17] introduced identity-based cryptography in 1984, which can be used in identity-based network coding. The solution is utilizing user's information such as email or ID number etc. as his public key, which can eliminate the need for public key certificate. Furthermore, the private key of user is generated from key generation center (KGC) by combining user's public key with KGC's master key.

However, since KGC possesses private keys of all users, which makes identity-based network coding faces key-escrow problems, that is KGC may forge the user's signature. To solve this problem, Al-Riyami *et al.* [20] proposed the theory of certificateless public key cryptography in 2003, which suggested certificateless signature (CLS). Then, for the first time, Chang *et al.* [6] proposed the certificateless network coding scheme in 2020, but there are still few concrete certificateless network coding schemes in current researches. In view of this situation, we consider how to relate network coding with cloud storage in order to discover more concrete certificateless network coding schemes.

In the field of cloud storage, the concept of cloud auditing was put forward by Juels *et al.* [4] and Ateniese *et al.* [29] at same time, they proposed proof of retrievability (PoR) protocol and provable data possession (PDP) protocol respectively, among which PoR protocol can verify whether the cloud service provider (CSP) has complete data of the user. In order to verify the integrity of outsourced data, the data owner can leverage PoR protocol to verify the integrity of outsourced data.

Genqing Bian and Mingxuan Song are with the School of Information and Control Engineering, Xi'an University of Architecture and Technology, Shaanxi, 710055, P.R.China. E-mail: bgq_00@163.com; songmingxuan@xauat.edu.cn.

Bilin Shao is with the School of Management, Xi'an University of Architecture and Technology, Shaanxi, 710055, P.R.China. Email: sbilin0462@sina.com.

Subsequently, Shacham *et al.* [28] came up with a concrete public auditing protocol on the basis of message authentication code (MAC) and digital signature. The public auditing namely the data owner can entrust third party auditors (TPA) to implement the data integrity auditing on his behalf without downloading all data. As mentioned above, KGC is not completely trusted, because data owner's private key is entirely generated by KGC. Hence, the certificateless public auditing protocol (CL-PAP) solved the problems of key escrow. In CL-PAPs, the data owner's private key consists of two parts, one generated by the himself and the other generated by KGC, while the public key can be generated by a public verifier with the data owner's identity such as his email or ID number to ensure that he uses correct public key during the auditing process.

Although network coding and cloud storage are two different fields of the networking, they are both fundamentally concerned on data integrity auditing. Specifically, under what conditions can the public auditing protocol be constructed from the network coding scheme or the public auditing protocol be transformed into the network coding scheme. The relationship between them has been revealed by some researchers. In 2016, Chen *et al.* [8] proposed a cloud storage protocol based on a secure network coding scheme and enhanced it to support public auditing. In 2021, Chang *et al.* [27] presented a general construction from an admissible PoR protocol to secure network coding scheme and they describe some concrete instantiations. This relationship between these two fields can naturally be extended under certificateless public key cryptography.

Our Contributions: In this paper, we propose a certificateless public auditing protocol, and give it some conditions to become an admissible certificateless public auditing protocol. Then, for the first time, we present a general construction in order to transform this admissible protocol into the certificateless network coding scheme. After that, to show the powerful ability of our general construction, we give a concrete admissible CL-PAP and transform it into a CLNS. Finally, we evaluate the performance of the transformed CLNS with other two network coding schemes in terms of communication overheads and computing costs. According to experimental result, we can see that our transformed CLNS is competitive.

Related Works: In the field of network coding, Attrapadung [12] *et al.* proposed a standard model of homomorphic network coding signatures in 2011. Later, Lin *et al.* [18] proposed a linearly homomorphic proxy signature scheme. In order to avoid certificate management, Lin *et al.* [19] and Chang *et al.* [31] introduced different identity-based linearly homomorphic signature schemes, among which Chang's scheme can resist the related-key attack. In addition, Fan *et al.* [13] and Liu *et al.* [25] respectively proposed privacy-preserving signature schemes for network coding. To solve the key-escrow attack, the certificateless linearly homomorphic signature has been applied in network coding, Chang *et al.* [6] introduced certificateless network coding in 2020. In recent works, Wu *et al.* [9] applied certificateless network coding scheme in the IOT.

As for the cloud storage, Chen *et al.* [14] introduced a novel remote data checking for distributed storage systems. Zhu *et*

al [24] designed a audit system which support dynamic data operations. To improve security and performance, Wang *et. al* [15] proposed a privacy-preserving public auditing protocol for shared cloud data and a public auditing protocol with efficient user revocation [16] successively. In 2017, Shen *et al.* [23] considered a public auditing protocol with dynamic structure. In 2021, Wang *et. al* [32] introduced a secure cloud storage auditing protocol which can tolerance small data errors. For certificate management issues, Wang *et. al* [21] proposed an identity-based public auditing protocol for multicloud storage. After the advent of certificateless public key cryptography, many certificateless public auditing protocols have been proposed by researchers [7], [22], [26], [30].

Organizations: This article is organized as follows. In Section 2, we first introduce basic notions and preliminaries including bilinear map, linear network coding and complexity assumptions. Then, in Section 3, we describe several different network coding system models, focusing on the certificateless network coding scheme and its security model. Later, we introduce the certificateless public auditing protocol with security model and construct an admissible certificateless public auditing protocol in Section 4. In Section 5, we propose a general construction for transforming an admissible CL-PAP to a CLNS. We show a concrete instantiation in Section 6. Performance analysis and conclusion are demonstrated in Section 7 and Section 8.

II. NOTIONS AND PRELIMINARIES

In this section, we introduce some basic notations and basic cryptographic tools consist of bilinear map, linear network coding and complexity assumptions.

Basic Notions: We use k denote the security parameter. PPT means probabilistic polynomial time. Given a prime number p , Z_p and Z_p^* denote finite fields $[0, 1, \dots, p-1]$ and $[1, 2, \dots, p-1]$ respectively. Besides, $\mathbf{v}$ denotes a vector and v_i denotes its i -th element.

A. Bilinear Map

Suppose G_1 and G_2 are two multiplicative groups which have the same prime order p , g is a generator of G_1 . $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it satisfies the following properties:

- 1) **Computability:** for any $u, v \in G_1$, $e(u, v)$ is efficiently computable.
- 2) **Non-degeneracy:** there exist two elements $u, v \in G_1$ such that $e(u, v) \neq 1$.
- 3) **Bilinearity:** for any $a, b \in Z_p^*$ and $u, v \in G_1$, $e(u^a, v^b) = e(u, v)^{ab}$.

B. Linear Network Coding

There are three steps to complete the file transmission in a linear network coding scheme:

- A file to be transmitted can be regarded as a sequence order of n -dimensional vectors $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m \in Z_p^n$. Before

the transmission starts, the source node augments them to $\mathbf{v}_1, \dots, \mathbf{v}_m$ as

$$\mathbf{v}_i = \left(\underbrace{0, \dots, 0}_i, 1, 0, \dots, 0, \bar{\mathbf{v}}_i \right) \in \mathbb{Z}_p^{m+n}.$$

In this way, $\mathbf{v}_1, \dots, \mathbf{v}_m$ form an augmented basis of a subspace V . The augmented vectors will be transmitted as packets by the source node.

- On receiving packets (i.e., vectors) $\mathbf{w}_1, \dots, \mathbf{w}_l \in \mathbb{Z}_p^{m+n}$ on its l incoming edges, the intermediate node computes the packet (i.e., a linear combination) $\mathbf{w} = \sum_i c_i \mathbf{w}_i$, where c_i is randomly selected from $\mathbb{Z}_p$. Then, each intermediate node sends the vector $\mathbf{w}$ on its outgoing edges.
- For a destination node (i.e., receiver) who wants to recover the original file, he has to receive m linearly independent vectors $\mathbf{w}_1, \dots, \mathbf{w}_m$. Suppose $\mathbf{w}_i^R$ ($\mathbf{w}_i^L$) denotes the right-most n (left-most m) positions of the vector $\mathbf{w}_i$. Then the receiver computes an $m \times m$ matrix Q such that

$$Q = \begin{pmatrix} \mathbf{w}_1^L \\ \vdots \\ \mathbf{w}_m^L \end{pmatrix}^{-1}.$$

Finally, the receiver can recover original file by computing

$$\begin{pmatrix} \bar{\mathbf{v}}_1 \\ \vdots \\ \bar{\mathbf{v}}_m \end{pmatrix} = Q \cdot \begin{pmatrix} \mathbf{w}_1^R \\ \vdots \\ \mathbf{w}_m^R \end{pmatrix}.$$

C. Complexity Assumptions

Definition 1 (CDH problem). $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear map and g is a generator of G_1 . Give (g, g^x, g^y) where x, y are chosen randomly from $\mathbb{Z}_p$. The problem is for any PPT algorithm to compute and output g^{xy} .

Definition 2 (CDH assumption). The advantage for any PPT algorithm $\mathcal{A}$ to solve the CDH Problem is negligible, which can be defined as:

$$\text{Adv}_{G_1}^{\text{CDH}} \mathcal{A} = \Pr \left[\mathcal{A}(g, g^a, g^b) : a, b \xleftarrow{R} \mathbb{Z}_p^* \right].$$

Definition 3 (DL assumption). Suppose $\alpha \in \mathbb{Z}_p^*$ and g is a generator of G_1 , given g and g^α as input. For any PPT algorithm $\mathcal{A}$ to output α is computational infeasible, which can be defined as:

$$\text{Adv}_{G_1}^{\text{DL}} \mathcal{A} = \Pr \left[\mathcal{A}(g, g^\alpha) : \alpha \xleftarrow{R} \mathbb{Z}_p^* \right].$$

III. CERTIFICATELESS NETWORK CODING

In this section, we briefly introduce different network coding with their system models first. Then, we give a certificateless network coding scheme and its security model.

Fig. 1. System model of certificateless network coding

A. System Model

1) *Secure Network Coding*: According to traditional homomorphic signature schemes, there are three nodes in secure network coding: source nodes (S), intermediate nodes (N) and destination nodes (R).

- A source node S chooses or generates a key pair (SK, PK) for signing. The data packets can be regarded as a set of vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$, S computes these vectors' corresponding signatures $\sigma_1, \dots, \sigma_m$. Then, S transmits each tuple $(\mathbf{v}_i, \sigma_i)$ ($1 \leq i \leq m$) to the next intermediate nodes.
- Upon receiving some $(\mathbf{v}_1, \sigma_1), (\mathbf{v}_2, \sigma_2), \dots, (\mathbf{v}_l, \sigma_l)$ as packet-tag pairs, an intermediate node N_i first checks the validity of all pairs and discards the "polluted" ones. For the "unpolluted" pairs, it randomly chooses l coefficients $c_1, c_2, \dots, c_l \in \mathbb{Z}_p$ to compute "combined" vector $\mathbf{v}'$ and its signature σ' as new pair $(\mathbf{v}', \sigma')$, then transmit the new pair to adjacent nodes.
- After the destination node R_j collects enough pairs $(\mathbf{v}'_1, \sigma'_1), (\mathbf{v}'_2, \sigma'_2), \dots, (\mathbf{v}'_m, \sigma'_m)$, R_j also first check the validity of pairs and discard "polluted" ones. Finally, it can recover the original vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$.

2) *Identity-Based Network Coding*: In PKI, CA issues certificate for public key authentication and bind the user's public key to his corresponding identity. But PKI introduces certificate management problems such as certificate distribution, revocation, storage, and computational overhead of certificate validation. To solve the certificate management problem, Shamir *et al.* [17] introduced an identity-based homomorphic signature scheme which can be used for network coding.

There is also a KGC to generate user's private key in identity-based network coding except for the three nodes mentioned above. Concretely, the public key PK of KGC

is unique and universal, it needs not has to execute public key certification of each source node. That is, for intermediate nodes and destination nodes, only the common public key PK of KGC is needed for signature verification. A source node S_i sends his identity ID_i to KGC, then the KGC returns a private key SK_{ID_i} for signing vectors. For intermediate nodes or destination nodes, they use PK verify the validity of vector-signature pairs.

3) *Certificateless Network Coding*: Since KGC knows the private keys of all users, identity-based network coding faces key-escrow problems. Therefore, Al-Riyami *et al.* [20] proposed the concept of certificateless signature (CLS). In CLS scheme, the user's private key is a combination of a *partial* private key and a secret value. The *partial* private key is generated from KGC by using the master key and an identity from the user.

Concretely, in certificateless network coding, after S sends his identity ID to KGC, KGC only returns *partial* private key PP_{ID} of S . As for S , he randomly chooses a secret value s_{ID} , then combine s_{ID} and PP_{ID} to generate his *full* private key SK_{ID} . Meanwhile, public key PK_{ID} of S is also computed by s_{ID} . Therefore, SK_{ID} is used for signing vectors to compute out signatures. Then the intermediate nodes or destination nodes can obtain PK_{ID} and use it verify the vector-signature pairs.

B. Certificateless Network Coding Scheme

A certificateless network coding scheme (CLNS) consists of following PPT algorithms: **Setup**, **Extract**, **SetSecretValue**, **SetPrivateKey**, **SetPublicKey**, **Sign**, **Combine**, **Verify**.

- **Setup**(k) $\rightarrow(msk, params)$: This algorithm runs by KGC, after inputting security parameter k , it outputs a master key msk only known by himself. System parameters $params$ are including the length n of each vector.
- **Extract**(msk, ID) $\rightarrow PP_{ID}$: This algorithm also runs by KGC to produce *partial* private key. It inputs msk and user's identity ID , then outputs his *partial* private key SK_{ID} .
- **SetSecretValue**(ID) $\rightarrow s_{ID}$: This algorithm executes by source node to generate his secret value. For the user with ID , it inputs $params$ and outputs his secret value s_{ID} .
- **SetPrivateKey**(s_{ID}, PP_{ID}) $\rightarrow SK_{ID}$: This algorithm runs by source node to generate his private key. Particularly, it takes the user's secret value s_{ID} and *partial* private key PP_{ID} , then it outputs his *full* private key SK_{ID} .
- **SetPublicKey**(s_{ID}) $\rightarrow PK_{ID}$: This algorithm also runs by source node to generate his public key. Precisely, it takes the user's secret value s_{ID} and outputs public key PK_{ID} .
- **Sign**($ID, SK_{ID}, PK_{ID}, id, V$) $\rightarrow T$: This algorithm is used to signing and run by source node. For the user's identity ID , user's private key SK_{ID} and public key PK_{ID} , a file identifier id and a set V as input, it outputs a signature set T . The set V has m pieces of vectors $\bar{v}_1, \dots, \bar{v}_m \in Z_p^n$, it computes signature t_i for each

augmented vector $\mathbf{v}_i = (\mathbf{v}_{i,1}, \dots, \mathbf{v}_{i,m+n}) \in Z_p^{m+n}$ and the output T has the form of $\{\sigma_1, \sigma_2, \dots, \sigma_m\}$.

- **Combine**($ID, PK_{ID}, id, \{c_i, \mathbf{v}_i, \sigma_i\}_{i=1}^l$) $\rightarrow(\mathbf{v}, \sigma)$: This algorithm runs by intermediate node to process received vector-signature pairs. Upon receiving l vector-signature pairs with same identifier id , the intermediate node randomly chooses l coefficients and outputs a combined vector $\mathbf{v} = \sum_{i=1}^l c_i \mathbf{v}_i$ with its combined signature σ (w.r.t. ID).
- **Verify**($ID, PK_{ID}, id, \mathbf{y}, \sigma$) $\rightarrow 1/0$: This algorithm executes by destination nodes or intermediate nodes to check the received vector-signature pairs if it has validity. It inputs a tuple $(ID, PK_{ID}, id, \mathbf{y}, \sigma)$ and outputs 0 (reject) or 1 (accept).

Correctness: The correctness of CLNS requires each key pair (SK_{ID}, PK_{ID}) generated by **Setup**, **Extract**, **SetSecretValue**, **SetPrivateKey**, **SetPublicKey**, it holds that:

- 1) if $T \leftarrow \text{Sign}(ID, SK_{ID}, PK_{ID}, id, V)$, then for all id and $\mathbf{v}_i \in Z_p^{m+n}$, $1 \leftarrow \text{Verify}(ID, PK_{ID}, id, \mathbf{v}, \sigma)$; and
- 2) if for all id and i ,

$$1 \leftarrow \text{Verify}(ID, PK_{ID}, id, \mathbf{v}_i, \sigma_i),$$

and for $1 \leq i \leq l$, then

$$1 \leftarrow \text{Verify}\left(ID, PK_{ID}, id, \sum c_i \mathbf{v}_i, \text{Combine}(ID, PK_{ID}, id, \{c_i, \sigma_i, \mathbf{v}_i\}_{i=1}^l)\right).$$

C. Security Model

In a CLNS, there are two types of adversaries to be considered. Note that in CLNS the algorithm **Combine** doesn't need to input private key, therefore, the adversaries can obtain the results after running it and do other things they don't need. In addition, we need to focus on a valid forgery, because the adversaries can use several vector-signature pairs to generate a new pair by combination algorithm. According to our analysis, we consider two adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$. The former one $\mathcal{A}_I$ is an usual adversary who can not access the system's master key but can replace the public key with a value. The latter one $\mathcal{A}_{II}$ can access master key but can not replace the public key who can be considered as a malicious KGC.

The security of CLNS can be characterized by two games **Game-1** and **Game-2**, in which $\mathcal{A}_I$ interacts with its challenger CH_I and $\mathcal{A}_{II}$ interacts with its challenger CH_{II} respectively. These games are based on the difficulty of the CDH problem. In particular, for the adversary in **Game-1** we give it some following restrictions. First, the challenge identity ID^* has not been replaced by a public key and its *partial* private key has not been extracted. Then, the adversary $\mathcal{A}_I$ can not access its *full* private key.

Game-1: $\mathcal{A}_I$ is an usual adversary plays with his challenger.

- **Setup-1**: CH_I runs **Setup**(k) $\rightarrow(msk, params)$, it keeps msk secret and gives $params$ to $\mathcal{A}_I$.
- **Queries-1**: $\mathcal{A}_I$ can adaptively perform the following queries but comply with above restrictions.
 - *Partial-Private-Key-Extract*: Given user's ID , the challenger runs **Extract**(msk, ID) $\rightarrow PP_{ID}$ and returns PP_{ID} to $\mathcal{A}_I$.

- *Private-Key-Extract*: The challenger uses ID to run **SetSecretValue** $(ID) \rightarrow s_{ID}$ and outputs secret value s_{ID} . Then, it continues to runs

SetPrivateKey $(s_{ID}, PP_{ID}) \rightarrow SK_{ID}$

and outputs SK_{ID} , then returns SK_{ID} to $\mathcal{A}_I$.

- *Public-Key-Query*: Upon receiving a query with the ID , the challenger runs **SetPublicKey** $(s_{ID}) \rightarrow PK_{ID}$ and returns PK_{ID} to $\mathcal{A}_I$.
- *Public-Key-Replace*: $\mathcal{A}_I$ can replace the public key PK_{ID} with anotehr value PK'_{ID} and note that $\mathcal{A}_I$ does not need to provide the secret value corresponding to PK'_{ID} .
- *Signing-Queries*: When $\mathcal{A}_I$ queries a vector $\mathbf{v}_i \in Z_p^n$, the challenger chooses a file identifier id and returns it to $\mathcal{A}_I$. Then, the challenger runs

Sign $(ID, SK_{ID}, PK_{ID}, id, V) \rightarrow T$,

then returns id and T to $\mathcal{A}_I$.

- **Output-1**: $\mathcal{A}_I$ outputs a file identifier id^* and a nonzero vector $\mathbf{v}^*$ with its signature σ^* all respect to ID^* and PK_{ID^*} . In this case, the adversary $\mathcal{A}_I$ wins the **Game-1** if **Verify** $(ID^*, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*) \rightarrow 1$ and one of the following two conditions holds:

- 1) When $\mathbf{v}^* \neq 0$ and $id^* \neq id_i$ to any id_i appeared in signing queries. (Type-1 forgery)
- 2) Suppose $id^* = id_i$ for some i and $\mathbf{v}^* \notin S_i$, where S_i is a subspace spanned by vectors corresponding to id_i . (Type-2 forgery)

We denote $Adv_{\mathcal{A}_I}^{CLNS}(k)$ as the advantage or probability of $\mathcal{A}_I$ winning the **Game-1**.

Game-2: $\mathcal{A}_{II}$ is an adversary like a malicious KGC interacts with its challenger in this game.

- **Setup-2**: $\mathcal{CH}_{II}$ also runs **Setup** $(k) \rightarrow (msk, params)$, then give msk and $params$ to $\mathcal{A}_{II}$.
- **Queries-2**: $\mathcal{A}_{II}$ can adaptively make queries including *Private-Key-Extract*, *Public-Key-Query* and *Signing-Queries* which are same as ones in **Game-1**.
- **Output-2**: $\mathcal{A}_{II}$ outputs a file identifier id^* and a nonzero vector $\mathbf{v}^*$ with its signature σ^* all respect to ID^* and PK_{ID^*} , here ID^* has not been issued as a Private-Key-Extract query. On this occasion, the adversary $\mathcal{A}_{II}$ wins the **Game-2** if **Verify** $(ID^*, PK_{ID^*}, id^*, \mathbf{v}^*, \sigma^*) \rightarrow 1$ and one of the following two conditions holds:
 - 1) When $\mathbf{v}^* \neq 0$ and $id^* \neq id_i$ to any id_i appeared in signing queries. (Type-1 forgery)
 - 2) Suppose $id^* = id_i$ for some i and $\mathbf{v}^* \notin S_i$, where S_i is a subspace spanned by vectors corresponding to id_i . (Type-2 forgery)

We denote $Adv_{\mathcal{A}_{II}}^{CLNS}(k)$ as the advantage or probability of $\mathcal{A}_{II}$ winning the **Game-2**.

The CLNS is secure under the adaptive chosen message attack if for any PPT adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$, their advantages $Adv_{\mathcal{A}_I}^{CLNS}(k)$ and $Adv_{\mathcal{A}_{II}}^{CLNS}(k)$ of winning above games are negligible.

Fig. 2. System model of certificateless public auditing

IV. CERTIFICATELESS PUBLIC AUDITING

In this section, we introduce the certificateless public auditing protocol including its system model and security model. Then, we make some constraints as conditions on this protocol in order to transform it into a certificateless network coding scheme.

A. System Model

In a certificateless public auditing protocol (CL-PAP), there are four entities including KGC, CSP, data owner and TPA. The main functions of each entity are described as follows:

- KGC is completely trusted which generates master key, public parameters and data owner's *partial* private key based on data owner's identity (i.e., email and name).
- CSP is semi-trusted, it provides sufficient storage space and retrieval capabilities. With multiple motivations, the CSP may modified or delete user's storage file.
- Data owner is the user who owns the data, he outsources the data to the CSP. In general, he is able to divide the data file into a number of blocks to modify the data blocks efficiently.
- TPA is completely trusted. According to the public parameters, data owner's identity and auditing challenge, the TPA or a public verifier checks the correctness of the cloud data belongs to data owner without managing certificates, he can not access any specific data information in auditing process.

Generally, a certificateless public auditing protocol considers a data owner wants to store a file F to CSP, he is able to divide the file into m blocks and each of them $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m \in Z_p^{m+n}$. In order to audit the cloud data, the data owner can use some key to generate an authenticated signature σ_i corresponding to a block $\mathbf{v}_i$ for $1 \leq i \leq m$. Then, the data file F and its authenticated signatures set T will be stored together by CSP. During the auditing process, the TPA launches a challenge $chal$ to CSP, then the CSP returns a proof Γ based on challenge and data user's cloud data. Finally, the TPA checks the validity of the proof. Note that the data owner and TPA can both perform the audit process by themselves.

B. Certificateless Public Auditing Protocol

A certificateless public auditing protocol CL-PAP needs requirements of public auditability, privacy preserving and unforgeability, it consists of seven PPT algorithms: **Setup'**, **Extract'**, **KeyGen'**, **Outsource'**, **Chal'**, **ProofGen'** and **Verify'**.

- **Setup'**(k) $\rightarrow$ ($msk, params$): This algorithm runs by KGC and outputs system parameters and master key. After inputting a security parameter k , it outputs $params$ and a master key msk . Note that $params$ is public and KGC keeps the msk secret.
- **Extract'**(msk, ID) $\rightarrow PP_{ID}$: This algorithm runs by KGC to generate data owner's partial private key. Concretely, given the data owner's identity ID and KGC's msk , it outputs partial private key PP_{ID} , KGC sends PP_{ID} to data owner with ID through a secure channel.
- **KeyGen'**(ID, PP_{ID}) $\rightarrow (s_{ID}, SK_{ID}, PK_{ID})$: This is a key generation algorithm runs by data owner. In particular, it takes PP_{ID} and data owner randomly chooses a secret value s_{ID} as input. Then output user's corresponding full private key SK_{ID} and public key PK_{ID} .
- **Outsource'**(SK_{ID}, PK_{ID}, F, id) $\rightarrow F'$: This algorithm runs by data owner, it generates an authenticated data file which will be stored on CSP. Upon inputting data file F , private SK_{ID} and public key PK_{ID} , the algorithm first divides F into m blocks and for each block v_i has the form of $(m+n)$ -dimensional vector over Z_p . Then, the algorithm generates an authenticated signature for each block. Finally, it randomly chooses a file identifier id and outputs an authenticated file F' including id , the original file F and signatures set T , that is $F' = \{id, F, T\}$.
- **Chal'**(l) $\rightarrow chal$: This algorithm runs by data owner or TPA. It generates a challenge message denoted as $chal$, which has l elements and submits the challenge to CSP.
- **ProofGen'**($chal, F'$) $\rightarrow \Gamma$: This algorithm runs by CSP. It inputs received challenge message $chal$ and an authenticated data file F' , then computes and outputs an integrity proof Γ which will be returned to the data owner or TPA.
- **Verify'**($ID, PK_{ID}, id, chal, \Gamma$) $\rightarrow 1/0$: This algorithm executes by data owner or TPA, after receiving a proof Γ from CSP, they check its validity. It inputs a public key PK_{ID} , data owner's identity ID , file identifier id , challenge message $chal$ and proof Γ , then output 1 or 0 (accept or reject).

Correctness: The correctness of CL-PAP requires for all (SK_{ID}, PK_{ID}) generated by **Setup'**, **Extract'**, **KeyGen'**, any $F' \leftarrow \text{Outsource}'(SK_{ID}, PK_{ID}, F, id)$, any $chal \leftarrow \text{Chal}'(l)$, and an honest proof Γ generated by **ProofGen'**($chal, F'$), it holds that

$$1 \leftarrow \text{Verify}'(PK_{ID}, ID, id, chal, \Gamma).$$

C. Security Model

There are three types of adversaries $\mathcal{A}'_I$, $\mathcal{A}'_{II}$ and $\mathcal{A}'_{III}$ in CL-PAPs, abilities of $\mathcal{A}'_I$ and $\mathcal{A}'_{II}$ are same as $\mathcal{A}_I$ and $\mathcal{A}_{II}$. For the adversary $\mathcal{A}'_{III}$, he is regard as a malicious CSP

and can forge the integrity proof without actual data, and the advantages of three adversaries are denoted as $Adv_{\mathcal{A}'_I}^{\text{CL-PAP}}(k)$, $Adv_{\mathcal{A}'_{II}}^{\text{CL-PAP}}(k)$ and $Adv_{\mathcal{A}'_{III}}^{\text{CL-PAP}}(k)$. Here, we define three following games **Game-1'**, **Game-2'** and **Game-3'** played by the defined adversaries with their challengers CH'_I , CH'_{II} and CH'_{III} .

Game-1': $\mathcal{A}'_I$ can not access the system's master key but can replace the public key with a value.

- **Setup-1'**: CH'_I runs **Setup'**(k) $\rightarrow$ ($msk, params$), it keeps msk secret and gives $params$ to $\mathcal{A}'_I$.
- **Queries-1'**: $\mathcal{A}'_I$ can adaptively perform the following queries.
 - **Partial-Private-Key-Extract'**: Given a user's ID , the challenger runs **Extract'**(msk, ID) $\rightarrow PP_{ID}$ and returns PP_{ID} to $\mathcal{A}'_I$.
 - **Secret-Value-Extract'**: $\mathcal{A}'_I$ sends an identity ID to challenger CH'_I , the challenger returns a secret value s_{ID} to $\mathcal{A}'_I$.
 - **Public-Key-Query'**: Upon receiving an identity ID from $\mathcal{A}'_I$, the challenger CH'_I returns PK_{ID} to $\mathcal{A}'_I$ after performing **KeyGen'**(ID, PP_{ID}).
 - **Public-Key-Replace'**: $\mathcal{A}'_I$ may replace the public key PP_{ID} with another value PP'_{ID} .
 - **Signing-Queries'**: $\mathcal{A}'_I$ submits ID and F , the challenger runs

$$\text{Outsource}'(SK_{ID}, PK_{ID}, F, id) \rightarrow F'$$

and returns F' to $\mathcal{A}'_I$.

- **Output-1'**: $\mathcal{A}'_I$ outputs a signatures set T^* on all blocks with ID^* and PK_{ID^*} . The adversary $\mathcal{A}'_I$ wins the game if the following conditions holds:
 - 1) For the data file F^* with ID^* and PK_{ID^*} , the signature T^* is valid.
 - 2) $\mathcal{A}'_I$ does not query the private key SK_{ID^*} and the partial private key PP_{ID^*} with the identity ID^* , and he replaces the public key with PK_{ID^*} .
 - 3) $\mathcal{A}'_I$ does not query signature T^* for ID^* and F^* .

The CL-PAP is secure against signature forgery under the public key replacement attack if for the PPT adversary $\mathcal{A}'_I$, his advantage $Adv_{\mathcal{A}'_I}^{\text{CL-PAP}}(k)$ of winning the game is negligible.

Game-2': $\mathcal{A}'_{II}$ is a malicious KGC plays game with his challenger.

- **Setup-2'**: CH'_{II} runs **Setup'**(k) $\rightarrow$ ($msk, params$), then give msk and $params$ to $\mathcal{A}'_{II}$.
- **Queries-2'**: $\mathcal{A}'_{II}$ can adaptively perform queries including **Secret-Value-Extract'**, **Public-Key-Query'** and **Signing-Queries'** which are same as ones in **Game-1'**.
- **Output-2'**: $\mathcal{A}'_{II}$ outputs a signatures set T^* on all blocks with ID^* and PK_{ID^*} . The adversary $\mathcal{A}'_{II}$ wins the game if the following conditions holds:
 - 1) For the data file F^* with ID^* and PK_{ID^*} , the signature T^* is valid.
 - 2) $\mathcal{A}'_{II}$ does not query the secret value s_{ID^*} and signature T^* for ID^* and F^* .

The CL-PAP is secure against signature forgery under the malicious KGC attack if for the PPT adversary $\mathcal{A}'_{II}$, his advantage $Adv_{\mathcal{A}'_{II}}^{\text{CL-PAP}}(k)$ of winning the game is negligible.

Game-3': $\mathcal{A}'_{III}$ is a semi-trusted CSP who can forge the proof without actual data.

- **Setup-3'**: CH'_{III} runs $\text{Setup}'(k) \rightarrow (msk, params)$ and

$$\text{KeyGen}'(ID, PP_{ID}) \rightarrow (s_{ID}, SK_{ID}, PK_{ID}),$$

then it keeps msk , SK_{ID} secret and gives $params$ to $\mathcal{A}'_{III}$.

- **Queries-3'**: $\mathcal{A}'_{III}$ can adaptively perform queries including *Secret-Value-Extract'*, *Public-Key-Query'* and *Signing-Queries'* which are same as ones in **Game-1'**.
- **Output-3'**: CH'_{III} randomly sends a challenge message $chal$ to $\mathcal{A}'_{III}$, then $\mathcal{A}'_{III}$ returns a forged proof Γ^* as respond. The adversary $\mathcal{A}'_{II}$ wins the game if $chal$ contains the index which has been queried in *Signing-Queries'* and $1 \leftarrow \text{Verify}'(PK_{ID}, ID, id, chal, \Gamma^*)$ holds.

The CL-PAP is secure against proof forgery under the semi-trusted CSP attack if for the PPT adversary $\mathcal{A}'_{III}$, his advantage $Adv_{\mathcal{A}'_{III}}^{\text{CL-PAP}}(k)$ of winning the game is negligible.

D. Admissible CL-PAP

We make some slight modifications on CL-PAP in order to transform it into a certificateless network coding scheme and therefore we give the following constraints as conditions.

- The challenge message $chal$ has the form of index and coefficient. That is, the j -th element of $chal$ has the form of (μ_j, ν_j) , where μ_j is the index of challenge block's position and ν_j is a random coefficient on Z_p . In most existing CL-PAPs, this kind of challenge message is suitable for instantiating CLNSs.
- The proof Γ has the form of linear combination. In other words, a proof Γ which is generated by **ProofGen'** should have the form of $(\mathbf{u}, \sigma)$, where $\mathbf{u}$ is a linear combination of packets with index and coefficient in $chal$, and σ is signature of $\mathbf{u}$. Since in CLNSs the transmitted pairs have the form of (package, tag), for the CSP it should return the proof Γ like this form.
- The Proof Γ can be aggregated. There exists an algorithm **Aggr'** which can aggregate some proofs with same id into a new proof Γ . Concretely, input s proof-coefficient pairs $(\Gamma_1, c_1), \dots, (\Gamma_s, c_s)$ and id , the algorithm **Aggr'** can output a new combined proof $\Gamma = (\mathbf{w}, \sigma)$, where $\mathbf{w}$ is a linear combination of vectors with coefficient $c_1, \dots, c_s$ in all Γ_i and σ is the tag of $\mathbf{w}$. This property is designed to describe the **Combine** algorithm in the CLNS.

Definition 4 (Admissible CL-PAP). A certificateless public auditing protocol is called admissible CL-PAP if it has properties of above three conditions.

V. CONSTRUCT CLNS FROM ADMISSIBLE CL-PAP

In this section, we give a general construction of a CLNS from an admissible CL-PAP, then we give the security analysis.

A. General Construction

According to our introduced admissible CL-PAP = (**Setup'**, **Extract'**, **KeyGen'**, **Outsource'**, **Chal'**, **ProofGen'**, **Verify'**, **Aggr'**), a CLNS = (**Setup**, **Extract**, **SetSecretValue**, **SetPrivateKey**, **SetPublicKey**, **Sign**, **Combine**, **Verify**) can be constructed as follows.

- **Setup**(k) $\rightarrow (msk, params)$: For the security parameter k as input, this algorithm runs **Setup'**(k) $\rightarrow (msk, params)$ and outputs them.
- **Extract**(msk, ID) $\rightarrow PP_{ID}$: For the inputs msk and ID , this algorithm runs **Extract'**(msk, ID) $\rightarrow PP_{ID}$ and outputs *partial* private key PP_{ID} .
- **SetSecretValue**(ID) $\rightarrow s_{ID}$: For the input ID , this algorithm runs **KeyGen'**(ID, PP_{ID}) and outputs secret value s_{ID} .
- **SetPrivateKey**(s_{ID}, PP_{ID}) $\rightarrow SK_{ID}$: This algorithm also runs **KeyGen'**(ID, PP_{ID}). It inputs s_{ID} and PP_{ID} and outputs *full* private key SK_{ID} .
- **SetPublicKey**(s_{ID}) $\rightarrow PK_{ID}$: This algorithm also runs **KeyGen'**(ID, PP_{ID}) and outputs public key PK_{ID} .
- **Sign**($ID, SK_{ID}, PK_{ID}, id, V$) $\rightarrow T$: For the inputs ID, SK_{ID}, PK_{ID}, id and a set V has m pieces of vectors $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m \in Z_p^n$, this algorithm first augments each vector as $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,m+n}) \in Z_p^{m+n}$ and define $F = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m) \in Z_p^{m \times (m+n)}$. Then, it runs

$$\text{Outsource}'(SK_{ID}, PK_{ID}, F, id) \rightarrow F'.$$

Finally, split $F' = \{id, F, T\}$ and output $T = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$.

- **Combine**($ID, PK_{ID}, id, \{c_i, \sigma_i, \mathbf{v}_i\}_{i=1}^l$) $\rightarrow (\mathbf{v}, \sigma)$: For the inputs ID, PK_{ID} and id , upon receiving tuples $(c_1, \mathbf{v}_1, \sigma_1), \dots, (c_l, \mathbf{v}_l, \sigma_l)$ with same file identifier id , define each $\Gamma_i = (\mathbf{v}_i, \sigma_i)$ according to the condition that "The proof Γ has the form of linear combination", where $1 \leq i \leq l$. Then, aggregate $\Gamma_1, \dots, \Gamma_l$ into a new Γ based on the condition that "The Proof Γ can be aggregated", that is

$$\text{Aggr}'(id, (\Gamma_i, c_i)_{i=1}^l) \rightarrow \Gamma = (\mathbf{v}, \sigma).$$

Finally, output proof Γ .

- **Verify**($ID, PK_{ID}, id, \mathbf{y}, \sigma$) $\rightarrow 1/0$: For the inputs ID, PK_{ID}, id , a vector $\mathbf{y} \in Z_p^{m+n}$ and a signature σ , this algorithm first divides $\mathbf{y} = (y_1, y_2, \dots, y_{m+n})$, then sets the challenge message $chal = \{(i, y_i)_{i=1}^m\}$ according to the condition that "The challenge message $chal$ has the form of index and coefficient" and sets $\Gamma = (\mathbf{y}, \sigma)$. Finally, it runs **Verify'**($ID, PK_{ID}, id, chal, \Gamma$) and outputs 0 (reject) or 1 (accept).

B. Security Analysis

The security of transformed CLNS will be depicted as follows. Note that there are only two types of adversaries, because there is no CSP as entity in network coding.

Theorem 1. The transformed CLNS is secure if the admissible CL-PAP is secure.

Concretely, suppose there exist two adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$ attack on the CLNS. Meanwhile, there are other two adversaries $\mathcal{B}_I$ and $\mathcal{B}_{II}$ attack the admissible CL-PAP, they regard $\mathcal{A}_I$ and $\mathcal{A}_{II}$ as its own subroutine respectively. Therefore, we divide the security proof of our construction into two parts: *Proof 1* and *Proof 2*.

Proof 1: $\mathcal{A}_I$ attacks the CLNS and $\mathcal{B}_I$ simulates environment for $\mathcal{A}_I$ as following steps.

Setup: $\mathcal{B}_I$ obtains $(msk, params)$, then gives $params$ to $\mathcal{A}_I$.

Queries: $\mathcal{A}_I$ can adaptively perform the following queries and $\mathcal{B}_I$ answers these queries.

- *Private-Key-Extract:* $\mathcal{B}_I$ uses ID to choose a secret value s_{ID} . Then, it continues to runs $\mathbf{KeyGen}'(ID, PP_{ID})$ and outputs SK_{ID} , then returns SK_{ID} to $\mathcal{A}_I$.
- *Public-Key-Query:* $\mathcal{B}_I$ also runs $\mathbf{KeyGen}'(ID, PP_{ID})$ and returns PK_{ID} to $\mathcal{A}_I$.
- *Public-Key-Replace:* $\mathcal{A}_I$ replaces the public key PK_{ID} with another value PK_{ID}^* and $\mathcal{A}_I$ does not need to provide the secret value corresponding to PK_{ID}^* .
- *Signing-Queries:* For the file $V \subset Z_p^{m \times n}$ will be queried, it can be split into n -dimensional vectors $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m \in Z_p^n$, $\mathcal{B}_I$ augments them to $\mathbf{v}_1, \dots, \mathbf{v}_m \in Z_p^{m+n}$ and defines $F = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$. Then, $\mathcal{B}_I$ runs $\mathbf{Outsource}'$ algorithm and obtains F' , whereafter $\mathcal{B}_I$ parses $F' = \{id, F, T\}$ and returns (id, T) to $\mathcal{A}_I$.
- *Combining-Queries:* For the tuples $(id, \{c_i, \sigma_i, \mathbf{v}_i\}_{i=1}^l)$ will be queried, $\mathcal{B}_I$ defines each $\Gamma_i = (\mathbf{v}_i, \sigma_i)$ and aggregates $\Gamma_1, \dots, \Gamma_l$ into a new Γ based on the condition that “The Proof Γ can be aggregated”, that is

$$\mathbf{Aggr}'(id, (\Gamma_i, c_i)_{i=1}^l) \rightarrow \Gamma = (\mathbf{v}, \sigma).$$

Finally, $\mathcal{B}_I$ returns Γ to $\mathcal{A}_I$.

Output: When $\mathcal{A}_I$ outputs a file identifier id^* and a nonzero vector $\mathbf{y}^*$ with its signature σ^* all respect to ID^* and PK_{ID}^* . $\mathcal{B}_I$ parses $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_{m+n}^*)$ and sets $chal^* = \{(i, y_i^*)_{i=1}^m\}$, $\Gamma = (\mathbf{y}^*, \sigma^*)$. $\mathcal{A}_I$ outputs $(ID^*, PK_{ID}^*, id^*, chal^*, \Gamma^*)$ as forgery.

In this case, if

$$\mathbf{Verify}'(ID^*, PK_{ID}^*, id^*, chal^*, \Gamma^*) \rightarrow 1,$$

and satisfy one of the following conditions:

- 1) $\mathbf{v}^* \neq 0$ and $id^* \neq id_i$ to any id_i appeared in *Signing-Queries*.
- 2) id^* equals some queried file V_i 's identifier id_i but $y^* \notin S_i$.

Then, $\mathcal{A}_I$ uses $(ID^*, PK_{ID}^*, id^*, \Gamma^*)$ to forge corresponding $chal^*$ successfully. The *Proof 1* ends.

Proof 2: $\mathcal{A}_{II}$ attacks the CLNS and $\mathcal{B}_{II}$ simulates environment for $\mathcal{A}_{II}$ as follows.

Setup: $\mathcal{B}_{II}$ obtains $(msk, params)$, then gives msk and $params$ to $\mathcal{A}_{II}$.

Queries: $\mathcal{A}_{II}$ can adaptively perform the following queries and $\mathcal{B}_{II}$ answers corresponding queries.

- *Private-Key-Extract:* $\mathcal{B}_{II}$ chooses a secret value s_{ID} based on an ID . Then, it runs $\mathbf{KeyGen}'(ID, PP_{ID})$ and outputs SK_{ID} , then returns SK_{ID} to $\mathcal{A}_{II}$.

- *Public-Key-Query:* $\mathcal{B}_{II}$ also runs $\mathbf{KeyGen}'(ID, PP_{ID})$ and returns PK_{ID} to $\mathcal{A}_{II}$.
- *Signing-Queries:* Given the queried file $V \subset Z_p^{m \times n}$, it can be split as $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m \in Z_p^n$. $\mathcal{B}_{II}$ augments them to $\mathbf{v}_1, \dots, \mathbf{v}_m \in Z_p^{m+n}$ and defines $F = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$. Then, $\mathcal{B}_{II}$ runs $\mathbf{Outsource}'$ algorithm and obtains F' , whereafter $\mathcal{B}_{II}$ parses $F' = \{id, F, T\}$ and returns (id, T) to $\mathcal{A}_{II}$.
- *Combining-Queries:* Given tuples $(id, \{c_i, \sigma_i, \mathbf{v}_i\}_{i=1}^l)$ will be queried, $\mathcal{B}_{II}$ defines each $\Gamma_i = (\mathbf{v}_i, \sigma_i)$ and aggregates $\Gamma_1, \dots, \Gamma_l$ into a new Γ by the algorithm

$$\mathbf{Aggr}'(id, (\Gamma_i, c_i)_{i=1}^l) \rightarrow \Gamma = (\mathbf{v}, \sigma).$$

Finally, $\mathcal{B}_{II}$ returns Γ to $\mathcal{A}_{II}$.

Output: $\mathcal{A}_{II}$ outputs a file identifier id^* , a nonzero vector $\mathbf{y}^*$ and a signature σ^* , all of them corresponding to ID^* and PK_{ID}^* . $\mathcal{B}_{II}$ parses $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_{m+n}^*)$ and sets $chal^* = \{(i, y_i^*)_{i=1}^m\}$, $\Gamma = (\mathbf{y}^*, \sigma^*)$. $\mathcal{A}_{II}$ outputs $(ID^*, PK_{ID}^*, id^*, chal^*, \Gamma^*)$ as forgery.

In this situation, if

$$\mathbf{Verify}'(ID^*, PK_{ID}^*, id^*, chal^*, \Gamma^*) \rightarrow 1,$$

and one of the following conditions holds:

- 1) $\mathbf{v}^* \neq 0$ and $id^* \neq id_i$ to any id_i appeared in *Signing-Queries*.
- 2) id^* equals some queried file V_i 's identifier id_i but $y^* \notin S_i$.

In this way, $\mathcal{A}_{II}$ utilizes msk and $(ID^*, PK_{ID}^*, id^*, \Gamma^*)$ successfully forges $chal$. The *Proof 2* ends.

Combining above *Proof 1* and *Proof 2*, we can derive that our general construction from admissible CL-PAP to CLNS is secure. The proof of **Theorem 1** ends.

VI. OUR CONCRETE INSTANTIATION

In this section, to show the power of our general construction in practice, we construct a concrete certificateless network coding scheme from an admissible certificateless public auditing protocol.

A. Scheme Details

We note that the certificateless public auditing protocol in [7] satisfies three conditions that have mentioned before. First, we propose an concrete admissible CL-PAP = (**Setup'**, **Extract'**, **KeyGen'**, **Outsource'**, **Chal'**, **ProofGen'**, **Verify'**, **Aggr'**) as follows.

- **Setup'**(k) $\rightarrow$ ($msk, params$): Given a security parameter k , KGC chooses two cyclic groups G_1, G_2 with same order p and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, we set g is the generator of G_1 . Then, KGC randomly chooses a $\lambda \in Z_p^*$ as his master key and defines $h = g^\lambda$. The KGC also chooses $m + n$ elements $(g_1, g_2, \dots, g_{m+n}) \in G_1$ randomly and defines $H_1 : \{0, 1\}^* \rightarrow G_1$ and $H_2 : \{0, 1\}^* \rightarrow G_1$ are two hash functions. The public system parameters $params$ are $(G_1, G_2, e, p, g, h, H_1, H_2, g_1, g_2, \dots, g_{m+n})$.

- **Extract'**(msk, ID) $\rightarrow PP_{ID}$: Given an identity ID of data owner, KGC computes *partial* private key $PP_{ID} = H_1(ID)^\lambda \in G_1$ and returns PP_{ID} to data owner.
- **KeyGen'**(ID, PP_{ID}) $\rightarrow (s_{ID}, SK_{ID}, PK_{ID})$: The data owner randomly chooses $s_{ID} \in Z_p^*$ as secret value, then he generates his *full* private key $SK_{ID} = (PP_{ID}, s_{ID})$ and public key $PK_{ID} = g^{s_{ID}}$.
- **Outsource'**(SK_{ID}, PK_{ID}, F, id) $\rightarrow F'$: The data owner chooses a random $id \in \{0, 1\}^k$ and splits data file F into m blocks $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$, each block has the form of $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \dots, v_{i,m+n}) \in Z_p^{m+n}$. Then, the data owner computes signature as

$$\sigma'_i = H_1(ID)^\lambda \times \left(H_2(ID || PK_{ID} || id || i) \cdot \prod_{j=1}^{m+n} g_j^{v_{i,j}} \right)^{s_{ID}} \in G_1.$$

Define $\mathbf{g} = \{g_1, g_2, \dots, g_{m+n}\}$, $\sigma_i = (\mathbf{g}, \sigma'_i)$ and $T = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$, where $1 \leq i \leq m$. Finally, output $F' = \{id, F, T\}$.

- **Chal'**(l) $\rightarrow chal$: Given input l , this algorithm chooses random $1 \leq \mu_1 < \mu_2 < \dots < \mu_l \leq m$ and l elements $\nu_1, \nu_2, \dots, \nu_l \in Z_p$, then output $chal = \{(\mu_j, \nu_j)_{j=1}^l\}$.
- **ProofGen'**($chal, F'$) $\rightarrow \Gamma$: For the inputs $chal$ and authenticated data file F' , this algorithm first parses $chal = \{(\mu_j, \nu_j)_{j=1}^l\}$ and $F' = \{id, F, T\} = \{id, (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m), (\sigma_1, \sigma_2, \dots, \sigma_m)\}$, where $\sigma_i = (\mathbf{g}, \sigma'_i)$. Then, for $1 \leq i \leq m$ and $1 \leq j \leq m+n$, this algorithm computes

$$M_j = \sum_{i=1}^l \nu_i v_{\mu_i, j} \in Z_p, \text{ and } \sigma' = \prod_{i=1}^l (\sigma'_{\mu_i})^{\nu_i} \in G_1.$$

Define proof $\Gamma = (M, \sigma) = ((M_1, \dots, M_{m+n}), (\mathbf{g}, \sigma'))$ and output the proof Γ .

- **Verify'**($ID, PK_{ID}, id, chal, \Gamma$) $\rightarrow 1/0$: For the inputs data owner's ID , the public key PK_{ID} , the file identifier id , the challenge message $chal$ and the returned proof $\Gamma = (\mathbf{y}, \sigma)$, this algorithm parses $\sigma = (\mathbf{g}, \sigma')$ first and checks whether

$$e(\sigma', g) = e \left(\prod_{i=1}^l H_1(ID)^{\nu_i}, h \right) \cdot e \left(\prod_{i=1}^l H_2(ID || PK_{ID} || id || \mu_i)^{\nu_i} \cdot \prod_{j=1}^{m+n} g_j^{\nu_j}, PK_{ID} \right).$$

If it equals, then output 1. If not, output 0.

The conditions of “The challenge message $chal$ has the form of index and coefficient” and “The proof Γ has the form of linear combination” are obvious for above CL-PAP. As for the condition of “The Proof Γ can be aggregated”, in the following we design an aggregation algorithm **Aggr'**.

- **Aggr'**($id, \{c_i, \Gamma_i\}_{i=1}^s$) $\rightarrow \Gamma$: Given the file identifier id and s proof-coefficient tuples $(c_1, \Gamma_1), \dots, (c_s, \Gamma_s)$ for $1 \leq$

$i \leq s$. This algorithm parses $\Gamma_i = (\mathbf{v}_i, \sigma_i) = (\mathbf{v}_i, (\mathbf{g}, \sigma'_i))$. Then, it computes,

$$\mathbf{v} = \sum_{i=1}^s c_i \mathbf{v}_i, \text{ and } \sigma' = \prod_{i=1}^s (\sigma'_i)^{c_i} \in G_1.$$

Finally, output the proof $\Gamma = (\mathbf{v}, \sigma) = (\mathbf{v}, (\mathbf{g}, \sigma'))$ which has been aggregated.

Then, based on our general construction, the transformed CLNS from above admissible CL-PAP can be depicted as follows.

- **Setup**(k) $\rightarrow (msk, params)$: For the security parameter k as input, this algorithm performs **Setup'**(k) $\rightarrow (msk, params)$. Output $params$ and the KGC keeps msk secret.
- **Extract**(msk, ID) $\rightarrow PP_{ID}$: For the inputs msk and ID , this algorithm runs **Extract'**(msk, ID) $\rightarrow PP_{ID}$. Output *partial* private key $PP_{ID} = H_1(ID)^\lambda$.
- **SetSecretValue**(ID) $\rightarrow s_{ID}$: Given the identity ID , this algorithm runs **KeyGen'**(ID, PP_{ID}) $\rightarrow s_{ID}$. Output s_{ID} as the secret value.
- **SetPrivateKey**(s_{ID}, PP_{ID}) $\rightarrow SK_{ID}$: For the inputs s_{ID} and ID , this algorithm performs

$$\mathbf{KeyGen}'(ID, PP_{ID}) \rightarrow SK_{ID}.$$

Output $SK_{ID} = (PP_{ID}, s_{ID})$ as the *full* private key.

- **SetPublicKey**(s_{ID}) $\rightarrow PK_{ID}$: For the input s_{ID} , this algorithm also runs **KeyGen'**(ID, PP_{ID}) $\rightarrow PK_{ID}$. Output $PK_{ID} = g^{s_{ID}}$ as the public key.
- **Sign**($ID, SK_{ID}, PK_{ID}, id, V$) $\rightarrow T$: For the inputs ID, SK_{ID}, PK_{ID}, id and a set V has m pieces of vectors $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m \in Z_p^n$, this algorithm first augments these vectors as $\mathbf{v}_i \in Z_p^{m+n}$ where $1 \leq i \leq m$ and define $F = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m) \in Z_p^{m \times (m+n)}$. Then, run

$$\mathbf{Outsource}'(SK_{ID}, PK_{ID}, F, id) \rightarrow F'.$$

In particular, this algorithm chooses a random id from $\{0, 1\}^k$, for each $\mathbf{v}_i$ where $1 \leq i \leq m$, it computes

$$\sigma'_i = H_1(ID)^\lambda \times \left(H_2(ID || PK_{ID} || id || i) \cdot \prod_{j=1}^{m+n} g_j^{v_{i,j}} \right)^{s_{ID}} \in G_1.$$

Define $\mathbf{g} = \{g_1, g_2, \dots, g_{m+n}\}$, $\sigma_i = (\mathbf{g}, \sigma'_i)$ where $1 \leq i \leq m$ and $T = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$. Finally, output T .

- **Combine**($ID, PK_{ID}, id, \{c_i, \mathbf{v}_i, \sigma_i\}_{i=1}^l$) $\rightarrow (\mathbf{v}, \sigma)$: For the inputs ID, PK_{ID} and id , upon receiving tuples $(c_1, \mathbf{v}_1, \sigma_1), \dots, (c_l, \mathbf{v}_l, \sigma_l)$ with the file identifier id , define each $\Gamma_i = (\mathbf{v}_i, \sigma_i)$, where $1 \leq i \leq l$. Then, it runs

$$\mathbf{Aggr}'(id, (\Gamma_i, c_i)_{i=1}^l) \rightarrow \Gamma = (\mathbf{v}, \sigma).$$

Output the aggregated proof Γ .

- **Verify**($ID, PK_{ID}, id, \mathbf{y}, \sigma$) $\rightarrow 1/0$: For the inputs ID, PK_{ID}, id , a vector $\mathbf{y} \in Z_p^{m+n}$ and a signature σ , this algorithm first divides $\mathbf{y} = (y_1, y_2, \dots, y_{m+n})$, then

sets the challenge message $chal = \{(i, y_i)_{i=1}^m\}$. This algorithm runs

$$\text{Verify}'(ID, PK_{ID}, id, chal, \Gamma) \rightarrow 1/0.$$

That is to check whether

$$e(\sigma', g) = e\left(\prod_{i=1}^m H_1(ID)^{y_i}, h\right) \cdot e\left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{y_j}, PK_{ID}\right).$$

If it equals, then output 1. If not, output 0.

Correctness: Given a data owner's identity ID , a full private key SK_{ID} , a public key PK_{ID} , a file identifier id , a set $V = (\bar{v}_1, \dots, \bar{v}_m) \in Z_p^n$ and a signature set $T \leftarrow \text{Sign}(ID, SK_{ID}, PK_{ID}, id, V)$, if the admissible CL-PAP is secure, then the correctness of verification equation in transformed CLNS can be checked as follows.

$$\begin{aligned} e(\sigma', g) &= e\left(\prod_{i=1}^m (\sigma'_i)^{y_i}, g\right) \\ &= e\left(\prod_{i=1}^m (H_1(ID)^{\lambda} \cdot \left(H_2(ID || PK_{ID} || id || i) \cdot \prod_{j=1}^{m+n} g_j^{v_{i,j}}\right)^{s_{ID}})^{y_i}, g\right) \\ &= e\left(\prod_{i=1}^m H_1(ID)^{\lambda y_i} \cdot \prod_{i=1}^m \left(H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{y_i v_{i,j}}\right)^{s_{ID}}, g\right) \\ &= e\left(\prod_{i=1}^m H_1(ID)^{\lambda y_i} \cdot \left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{i=1}^m \prod_{j=1}^{m+n} g_j^{y_i v_{i,j}}\right)^{s_{ID}}, g\right) \\ &= e\left(\prod_{i=1}^m H_1(ID)^{\lambda y_i} \cdot \left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{\sum_{i=1}^m y_i v_{i,j}}\right)^{s_{ID}}, g\right) \\ &= e\left(\prod_{i=1}^m H_1(ID)^{\lambda y_i} \cdot \left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{y_j}\right)^{s_{ID}}, g\right) \\ &= e\left(\prod_{i=1}^m H_1(ID)^{y_i}, h\right) \cdot e\left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{y_j}, PK_{ID}\right). \end{aligned}$$

As for the security of the transformed CLNS, we give a inference of Theorem 1.

Theorem 2. If the CDH assumption on G_1 holds, at the same time the two hash functions H_1 and H_2 are modeled as the random racle, then the transformed CLNS is secure.

TABLE I
TOTAL COMPARISONS OF COMMUNICATION OVERHEADS

Scheme	PP_{ID} Size	SigSize	SK_{ID} Size
[3]	$ Z_p + 2 G_1 $	$4 G_1 + 2 Z_p $	–
[9]	$ G_1 $	$2 G_1 $	$ G_1 + Z_p $
Our CLNS	$ G_1 $	$ G_1 $	$ G_1 + Z_p $

VII. PERFORMANCE ANALYSIS

This section we first compare the communication overheads and computational costs of our transformed CLNS with other similar network coding schemes in [3] and [9]. Then we evaluate their performance by experimental analysis.

A. Communication Overheads

First of all, we consider the communication overheads for KGC to users. In our proposed CLNS, a data owner sends his ID to KGC and KGC returns a *partial* private key PP_{ID} , therefore, the communication overheads for KGC to users is $|G_1|$, that is the size of PP_{ID} . In the same way, we can get the overhead for KGC to users in [3] and [9] is $|Z_p| + 2|G_1|$ and $|G_1|$. We use $|G_1|$ and $|Z_p|$ denote the length of G_1 and Z_p .

Second, we consider the communication overheads for source nodes. Since the source node transmits the all vectors to others, we compute the signature size for single vector to denote its overhead. In our scheme, the size of a single signature σ_i equals to $|G_1|$. In the similar way, we obtain that the overhead for source node is $4|G_1| + 2|Z_p|$ and $2|G_1|$ in [3] and [9].

Next, we compare the size of full private key SK_{ID} . The size of SK_{ID} in our scheme equals to $|G_1| + |Z_p|$. From this, we can figure out the size of SK_{ID} in [9] equals to $|G_1| + |Z_p|$. Note that in [3] the SK_{ID} is equivalent to its PP_{ID} , hence, this overhead for this item doesn't exist. Here, we use “–” represent the item is null.

The detailed comparisons of communication overheads are listed in TABLE I. According to comparisons, the size of PP_{ID} , SK_{ID} and signature(for single vector) in our transformed CLNS are competitive.

B. Computation Costs

We represent T_{bp} , T_{exp} , T_{mul} , T_{lc} as the execution time of a bilinear pairing operation, a modular exponentiation operation on G_1 , a multiplication operation on G_1 and a linear combination of l vectors. For convenience, we omit the computation like pseudorandom permutation, hash operation, addition on Z_p , multiplication on Z_p and so on, because their computation costs are negligible. Suppose there is a source node will

transmit a file in network, that is he transmits the packets $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m$ and generates signatures for them. Particularly, for the augmented package $\mathbf{v}_i$, its generated signature is

$$\sigma'_i = H_1(ID)^\lambda \times \left(H_2(ID || PK_{ID} || id || i) \cdot \prod_{j=1}^{m+n} g_j^{v_{i,j}} \right)^{s_{ID}} \in G_1.$$

Therefore, the computation cost for the σ'_i is $(n+2) \cdot (T_{exp} + T_{mul})$. As for the intermediate node or destination node, he needs to execute the **Verify** algorithm and **Combine** algorithm.

Then, he verifies each vector-signature pair $(\mathbf{y}, \sigma')$ by checking whether

$$e(\sigma', g) = e \left(\prod_{i=1}^m H_1(ID)^{y_i}, h \right) \cdot e \left(\prod_{i=1}^m H_2(ID || PK_{ID} || id || i)^{y_i} \cdot \prod_{j=1}^{m+n} g_j^{y_j}, PK_{ID} \right),$$

its cost time equals to $2 \cdot T_{bp} + (3m+n) \cdot T_{exp} + (3m+n-3) \cdot T_{mul}$. Moreover, given l vector-signature pairs $\{\mathbf{v}_i, \sigma_i\}_{i=1}^l$, the destination node computes

$$\mathbf{v} = \sum_{i=1}^s c_i \mathbf{v}_i, \text{ and } \sigma' = \prod_{i=1}^s (\sigma'_i)^{c_i} \in G_1,$$

which outputs a combined pair $(\mathbf{v}, \sigma)$. Hence, the total time spent takes $T_{lc} + l \cdot T_{exp} + (l-1) \cdot T_{mul}$.

Likewise, we can obtain the computation costs of [3] and [9]. The detailed comparisons of computation costs are listed in TABLE II. Based on computation costs analysis, we can find the efficiency of our scheme is comparable and our general generation from admissible CL-PAP to CLNS is feasible.

Fig. 3. Time consumption of signature generation

C. Experimental Analysis

In this subsection, we evaluate the computation costs of our transformed CLNS and schemes in [3] and [9] by experiments. According to “Charm” [10] framework, we choose the 512-bit SS elliptic curve from PBC library [11] and implement

Fig. 4. Time consumption of signature verification

all experiments on a Lenovo laptop with Intel Core i5-8300H CPU @2.30GHz, 8GB RAM, Ubuntu 20.04 LTS 64-bit and Python 3.8.

For the sake of simplicity, we set $m = n$. Particularly, we consider the time consumption of signature generation and verification for all vectors and change parameter m (or n) from 10 to 50. After choosing 50 times different random data packets and performing the experiments, we list the average results in Fig. 3 and Fig. 4. From the experimental result, we can see that our transformed CLNS is better than the other two schemes in terms of signature generation. As for the signature verification, our transformed CLNS is still comparable.

VIII. CONCLUSION

In this paper, we consider how to construct more concrete certificateless network coding schemes. First of all, we introduce an admissible certificateless public auditing protocol. Thereafter, we propose a general construction of certificateless network coding scheme from admissible certificateless public auditing protocol. Then, we give a concrete implementation in order to show the power of our general construction. In addition, we evaluate the performance of our transformed CLNS with other previous schemes. The experimental result shows our transformed CLNS is comparable and more efficient.

ACKNOWLEDGMENT

The first author was supported by National Natural Science Foundation of China (No.61872284), and the third author was supported by Shaanxi Provincial Natural Science Basic Research Project (No.2021JLM-16).

REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” IEEE Trans. Inf. Theory, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
- [2] D. Boneh et al., “Signing a linear subspace: Signature schemes for network coding,” in Public Key Cryptography, vol. 5443. Berlin, Germany: Springer, 2009, pp. 68–87.
- [3] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen and Y. Tang, “An ID-Based Linearly Homomorphic Signature Scheme and Its Application in Blockchain,” in IEEE Access, vol. 6, pp. 20632–20640, 2018.

TABLE II
TOTAL COMPARISONS OF COMPUTATION COSTS

Scheme	Sig-Generation	Sig-Verification	Combination
[3]	$(n+5) \cdot T_{exp} + (n+2) \cdot T_{mul}$	$3 \cdot T_{bp} + (m+n+2) \cdot T_{exp} + (m+n+1) \cdot T_{mul}$	$T_{lc} + l \cdot T_{exp} + (l-1) \cdot T_{mul}$
[9]	$(2n+7) \cdot T_{exp} + (2n+1) \cdot T_{mul}$	$4 \cdot T_{bp} + (2m+n+1) \cdot T_{exp} + (m+n+2) \cdot T_{mul}$	$T_{lc} + l \cdot T_{exp} + (l-1) \cdot T_{mul}$
Our CLNS	$(n+3) \cdot T_{exp} + (n+2) \cdot T_{mul}$	$3 \cdot T_{bp} + (2m+n+1) \cdot T_{exp} + (2m+n) \cdot T_{mul}$	$T_{lc} + l \cdot T_{exp} + (l-1) \cdot T_{mul}$

- [4] A. Juels and B. Kaliski Jr, "Pors: Proofs of retrievability for large files," in ACM Conference on Computer and Communications Security (SP), 2007, pp. 584–597.
- [5] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in Topics in Cryptology (Lecture Notes in Computer Science), vol. 2271. New York, NY, USA: Springer, 2002, pp. 244–262.
- [6] J. Chang, Y. Ji, B. Shao, M. Xu and R. Xue, "Certificateless Homomorphic Signature Scheme for Network Coding," in IEEE/ACM Transactions on Networking, vol. 28, no. 6, pp. 2615–2628, Dec. 2020.
- [7] B. Wang, B. Li, H. Li and F. Li, "Certificateless public auditing for data integrity in the cloud," 2013 IEEE Conference on Communications and Network Security (CNS), 2013, pp. 136–144.
- [8] F. Chen, T. Xiang, Y. Yang and S. S. M. Chow, "Secure Cloud Storage Meets with Secure Network Coding," in IEEE Transactions on Computers, vol. 65, no. 6, pp. 1936–1948, 1 June 2016.
- [9] B. Wu, C. Wang and H. Yao, "A certificateless linearly homomorphic signature scheme for network coding and its application in the IoT," Peer-to-Peer Netw. Appl. 14, 852–872, 2021.
- [10] J. A. Akinyele et al., "Charm: A framework for rapidly prototyping cryptosystems," J. Cryptograph. Eng., vol. 3, no. 2, pp. 111–128, Jun. 2013.
- [11] B. Lynn. The Standard Pairing Based Crypto Library. pbc-0.5.10. Accessed: Oct. 23, 2010. [Online]. Available: <http://crypto.stanford.edu>
- [12] N. Attrapadung and B. Libert, "Homomorphic network coding signatures in the standard model," in Public Key Cryptography—PKC (Lecture Notes in Computer Science), vol. 6571. Berlin, Germany: Springer, 2011, pp. 17–34.
- [13] Y. Fan, Y. Jiang, H. Zhu, J. Chen, and X. S. Shen, "Network coding based privacy preservation against traffic analysis in multi-hop wireless networks," IEEE Trans. Wireless Commun., vol. 10, no. 3, pp. 834–843, Mar. 2010.
- [14] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-based Distributed Storage Systems," in the Proceedings of ACM CCSW 2010, 2010, pp. 31–42.
- [15] B. Wang, H. Li and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," 2013 IEEE International Conference on Communications (ICC), 2013, pp. 1946–1950.
- [16] B. Wang, B. Li, and H. Li, "Public Auditing for Shared Data with Efficient User Revocation in the Cloud," in the Proceedings of IEEE INFOCOM 2013, 2013.
- [17] A. Shamir et al., "Identity-based cryptosystems and signature schemes," in Crypto, vol. 84. Berlin, Germany: Springer, 1984, pp. 47–53.
- [18] Q. Lin, J. Li, Z. Huang, W. Chen, and J. Shen, "A short linearly homomorphic proxy signature scheme," IEEE Access, vol. 6, pp. 12966–12972, 2018.
- [19] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen and Y. Tang, "An ID-Based Linearly Homomorphic Signature Scheme and Its Application in Blockchain," in IEEE Access, vol. 6, pp. 20632–20640, 2018.
- [20] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," in Proc. ASIACRYPT, 2003, pp. 452–473.
- [21] H. Wang, "Identity-Based Distributed Provable Data Possession in Multicloud Storage," in IEEE Transactions on Services Computing, vol. 8, no. 2, pp. 328–340, March–April 2015.
- [22] R. Zhou, M. He and Z. Chen, "Certificateless Public Auditing Scheme with Data Privacy Preserving for Cloud Storage," 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), 2021, pp. 675–682.
- [23] J. Shen, J. Shen, X. Chen, X. Huang and W. Susilo, "An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
- [24] Y. Zhu, G. Ahn, H. Hu, S. S. Yau, H. G. An and C. Hu, "Dynamic Audit Services for Outsourced Storages in Clouds," in IEEE Transactions on Services Computing, vol. 6, no. 2, pp. 227–238, April–June 2013.
- [25] X. Liu, J. Huang, Y. Wu and G. Zong, "A Privacy-Preserving Signature Scheme for Network Coding," in IEEE Access, vol. 7, pp. 109739–109750, 2019.
- [26] J. Li, H. Yan and Y. Zhang, "Certificateless Public Integrity Checking of Group Shared Data on Cloud Storage," in IEEE Transactions on Services Computing, vol. 14, no. 1, pp. 71–81, 1 Jan.–Feb. 2021.
- [27] J. Chang, B. Shao, Y. Ji, M. Xu and R. Xue, "Secure network coding from secure proof of retrievability," SCIENCE CHINA Information Sciences, 2021, 64(12): 229301.
- [28] Shacham H, Waters B. "Compact proofs of retrievability," Journal of Cryptology, 2013, 26: 442–483.
- [29] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores," in ACM Conference on Computer and Communications Security (CCS), 2007, pp. 598–609.
- [30] D. He, S. Zeadally and L. Wu, "Certificateless Public Auditing Scheme for Cloud-Assisted Wireless Body Area Networks," in IEEE Systems Journal, vol. 12, no. 1, pp. 64–73, March 2018.
- [31] J. Chang, H. Ma, A. Zhang, M. Xu and R. Xue, "RKA Security of Identity-Based Homomorphic Signature Scheme," in IEEE Access, vol. 7, pp. 50858–50868, 2019.
- [32] H. Wang, L. Feng, Y. Ji, B. Shao and R. Xue, "Toward Usable Cloud Storage Auditing, Revisited," in IEEE Systems Journal, doi: 10.1109/JSYST.2021.3055021.

Genqing Bian received his Ph.D. degree from the School of Management, Xi'an University of Architecture and Technology (XAUAT), Shaanxi, China. He is currently a Professor with the College of Information and Control Engineering, XAUAT. He is also a member of China Computer Federation (CCF) and Association for Computing Machinery (ACM). His research mainly includes information security, cloud computing security and data analysis. Email: bgq_00@163.com

Mingxuan Song received his B.S. degree from the School of Science, XAUAT, Shaanxi, China. He is currently pursuing the M.S. degree with the College of Information and Control Engineering, XAUAT. His research interests mainly include cryptography and information security. Email: songmingxuan@xauat.edu.cn

Bilin Shao received his B.S. degree from the School of Management, XAUAT, Shaanxi, China. He is currently a Professor with XAUAT. He is also a member of the CCF. His research mainly includes information security, information management technology, cloud computing security and VANETS security. Email: sbilin0462@sina.com

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [AuthorBiography.pdf](#)