Peer-to-Peer Networking and Applications (2021) 14:852–872
https://doi.org/10.1007/s12083-020-01028-8

A certificateless linearly homomorphic signature scheme for network coding and its application in the IoT

Bin Wu[1] · Caifen Wang[2] · Hailong Yao[3]

Received: 17 April 2020 / Accepted: 5 November 2020 / Published online: 7 January 2021

Abstract

Network coding is an effective method to optimize network throughput and improve routing reliability, and has been widely used in a decentralized Internet of Things system. However, the packet-mixing property of network coding renders transmission susceptible to pollution attacks, which may prevent the reconstruction of the original file. A homomorphic signature scheme is a powerful tool that enables network coding to combat pollution attacks. Although a series of homomorphic signature schemes already exists, no construction has been proposed to support both homomorphic network coding signatures and the certificateless characteristic. In this paper, we construct a certificateless linearly homomorphic signature scheme for network coding, thus avoiding the disadvantages of certificate management and key escrow problems. We then prove the security of the scheme in a random oracle model against an adaptively chosen dataset attack under two types of adversaries. Moreover, performance analysis results show that our scheme has a lower communication overhead and enjoys a comparable computation cost with related schemes.

Keywords Homomorphic signature · Certificateless cryptography system · Network coding · Provable security

1 Introduction

Typical Internet of Things (IoT) deployments include hardware technologies, sensing technologies (e.g., radio frequency identification and sensors), actuators, and other smart communication devices that are connected to the Internet. These technologies and equipment facilitate the extensive collection and exchange of information, files, and other real-time content that are shared between more and more smart terminals [1]. According to a report from the International Data Corporation , nearly 28 billion IoT devices will be installed by 2020, and the global economic impact of the IoT is estimated to be 2 trillion [2].

Given the proliferation of shared data and the expanding scale of the IoT, it is very worthwhile to increase throughput in such a large distributed network. The initial motivation of network coding was to improve the throughput of decentralized networks. In fact, this technology is considered to be a good approach to improve the distribution and sharing of digital content in peer-to-peer streaming networks and wireless ad hoc networks [3].

More specifically, unlike the traditional store-and-forward mechanism, in network coding, before the source node transmits a message (file) to the target node, it first divides the file into m packets, and then sends them to the neighboring nodes, thereby allowing the intermediate node (or router) to modify the received data packets and forward them. In linear network coding, the coding packets are regarded as vectors in linear space over some field. The intermediate node calculates the linear combination of these vectors by choosing random coefficients. If the target node receives a certain number of correct data packets, it can recover the original information with high probability. As this technology can optimize network throughput [4, 5], reduce energy consumption, and improve routing reliability [6], it is important to apply network coding technology in the IoT.

Because IoT devices typically interact with third-party applications, in an IoT system with network coding deployed, an important concern is preventing third-party

✉ Bin Wu
wubin1012@outlook.com

1 College of Mathematics and Statistics, Northwest Normal University, Lanzhou, 730070, China

2 College of Big Data and Internet, Shenzhen Technology University, Shenzhen, 518118, China

3 School of Electronic and Information Engineering, Lanzhou City University, Lanzhou, 730070, China

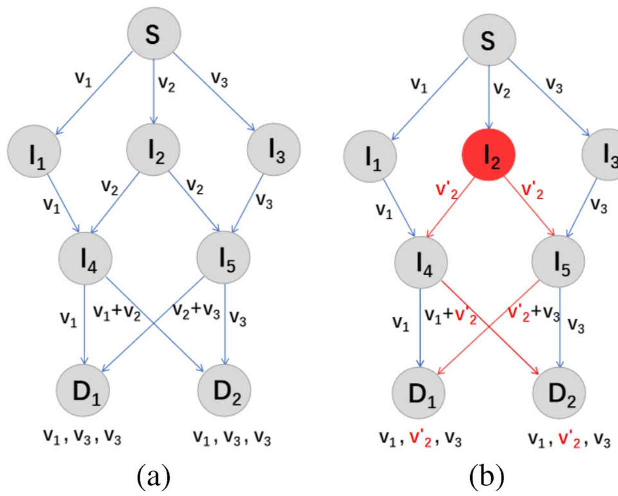

Fig. 1 Network coding and pollution attack on network coding

applications from maliciously modifying data packets, that is, pollution attacks. Specifically, network coding allows nodes to mix data packets to make them more vulnerable to pollution attacks. Errors introduced in only one packet can propagate and generate more invalid packets, which causes them to flow to their destination. A simplified version of the network coding without pollution attack and under pollution attack can be seen in Fig. 1, in which S is the source node, I_1, I_2, I_3, I_4, I_5 are the intermediate nodes, and D_1, D_2 are destination nodes. According to the Fig. 1b, the intermediate node I_2 transmits a invalid packet v'_2 , which affects the outputs of I_4 and I_5. Thus, the adversary can prevents file reconstruction by maliciously modifying only a small number of packets or injecting invalid packets.

To solve this problem, two main solutions have been proposed: information theoretic and cryptographic approaches. For the information theoretic approach, redundancy is introduced into the original package to recover the original files from malicious failures; however, the existing scheme can only passively tolerate the pollution attack at the destination. By contrast, the cryptographic approach does not restrict the adversary's behavior, and the intermediate node can detect and discard invalid data packets in the process of transmission, which can effectively mitigate pollution attacks. Therefore, in recent years, cryptographic solutions have attracted the attention of many scholars. They are divided into public key methods (e.g., homomorphic signature [7–10]) and symmetric key methods (e.g., homomorphic MAC [11–15]). The public key method avoids the problem of key distribution and is very suitable for a network coding environment where senders send multiple files to multiple receivers. In this paper, we focus on homomorphic signatures in public key methods. The main idea is to provide an approach to verify valid vectors. As shown in Fig. 2, after the source node outputs a properly augmented basis and the signatures of the basis, the intermediate node will verify the validity of all the signatures received. If a vector fails to pass the verification, the intermediate node will discard the invalid vector, calculate the linear combination of the remaining valid vectors, and generate a valid signature for the linear combination without the signer's secret key. Finally, the destination node will recover the original file from m linearly independent vectors.

In public key infrastructure, deployment costs are high and management certificates are tricky, particularly in resource-constrained environments, such as the IoT. To mitigate this issue, [16] introduced the concept of identity-based public key cryptography (ID-PKC). The concept is to use the user's identity information (e.g., IP address, driver's license number, and e-mail address) as a public key, and then the trusted key generation center (KGC) is responsible for generating private keys for users. Although ID-PKC simplifies certificate management, it introduces the problem of key escrow. Once the KGC is destroyed, the user's private key will be completely disclosed, making it unsuitable for large-scale network environments.

Al Riyami et al. [28] proposed a certificateless public key cryptosystem (CL-PKC) in which the user's private key is composed of some contributions of KGC, that is, the partial private key and a secret value chosen by the user. Thus, the CL-PKC eliminates the key escrow problem inherent in ID-PKC, while retaining the certificateless property. For different applications, many researchers have proposed encryption schemes [[29–31] and signature schemes [32–35] based on CL-PKC. However, to date, almost all proposed linearly homomorphic signature (LHS) schemes

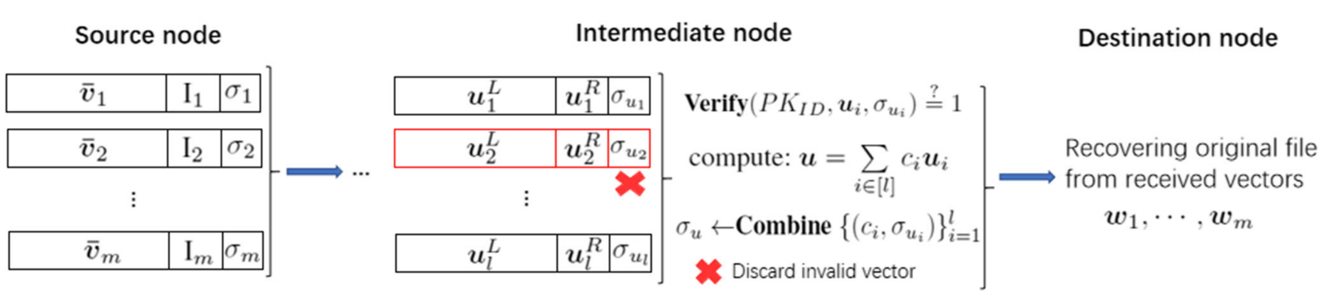

Fig. 2 The principle of homomorphic signature against pollution attack in network coding

have been based on either public key infrastructure [17–23] or identity cryptography [24–27], no construction has been proposed to support both a homomorphic network coding signature and certificateless characteristic. Therefore, to fill this gap in the literature, in this paper we design a certificateless LHS (CL-LHS) scheme for network coding. We prove that our homomorphic signature scheme is unforgettable even in the presence of type I and type II adversaries.

1.1 Our contributions

We summarize our main contributions as follows:

- We introduce the concept of a CL-LHS scheme for network coding, which addresses the issues of certificate management and key escrow while defending against pollution attacks.
- We present a security model for CL-LHS to guarantee the functionality and security of the proposed CL-LHS scheme, which considers two types of adversaries (Type I adversary and Type II adversary) that are capable of forging two types of signatures (Type 1 forgery and Type 2 forgery).
- We construct a concrete CL-LHS scheme and prove that the proposed scheme is secure against an adaptively chosen dataset attack in a random oracle model under the two types of adversaries.
- Compared with related LHS schemes, our CL-LHS scheme has a smaller key size and a shorter signature length and has comparable computation costs. By making the LHS scheme certificateless, our CL-LHS scheme can be deployed and implemented in an IoT environment with limited computing power and storage space.

1.2 Related works

In network coding, intermediate nodes (or routers) are allowed to combine and retransmit received data packets, and the recipient can still obtain the original data. This technology can maximize network throughput and increase robustness. Aiming at addressing the problem that linear network coding is vulnerable to malicious node pollution attacks, a solution based on computational assumptions and cryptographic technology is considered. The main idea here is to provide a method to verify valid vectors by using the network coding signature scheme. These schemes can be constructed from homomorphic hash functions or homomorphic signatures (HSs). Krohn et al. introduced a homomorphic hash function [36] to construct a network coding signature scheme. The main disadvantage of this method is that the authentication information and public key that must be sent with the package are very large, which is not conducive to improving throughput. Using LHS to perform network coding authentication is a more effective method. The work of Boneh et al. [8] is a milestone in LHS. In fact, it is considered the first to provide a practical framework for such a scheme. Attrapadung and Libert [37] showed that the first LHS scheme was secure under the standard model. The earliest RSA-based HS scheme was proposed by Gennaro et al. in 2010 [38], and it is proven that the scheme is unforgeable against a weak adversary in the random oracle model. Boneh and Freeman [39] presents the first scheme that can resist quantum attack, and the hardness assumption exploited is k-SIS. The above schemes are certificate-based cryptosystems. For fine-grained access control, identity-based HS schemes were proposed [24–27], which were proven to be secure in a random oracle model. Previous schemes allowed linear functions to be computed over signed data, while the scheme in [40] can evaluate multivariate polynomials, and [41, 42] proposed fully HS schemes supporting arbitrary functions.

To further extend the utility of HS, multiple-key HS has recently received attention [20–22, 48]. Multiple-key support is necessary for datasets that involve inputs authenticated by different clients, for example, in a distributed network of sensors. Prior to [48], the concept of homomorphic authentication studied only supported executions of computations over data authenticated by a single user. In 2019, Schabhüser et al. proposed the first perfectly context-hiding multiple-key linearly homomorphic authenticator scheme [22]. Lai et al. proposed a generic construction of multiple-key HS with unforgeability under corruption [21].

HS with additional functions applied to specific scenarios is also a popular topic in current research. Quantum-based protocols are being used in homomorphic signature schemes to address quantum network environments. Shang et al. in 2015 treated entanglement swapping as a homomorphic operation and creatively proposed the first quantum HS scheme [43]. However, this scheme only allows one verifier to verify a signature once. To support repeatable verification for general scenarios, Shang et al. proposed a new quantum HS scheme with repeatable verification by using a serial verification model and parallel verification model [44]. Li et al., in 2019, proposed two quantum homomorphic message authentication schemes based on quantum circuits, which can resist pollution attacks initiated by untrusted inside nodes over a general quantum network [45]. In addition, the verifiably encrypted HS scheme proposed by Seo et al. [46] and homomorphic signcryption scheme proposed by Fan et al. [47] have been successfully applied to accumulable optimistic fair exchange and electronic voting, respectively (Table 1).

Table 1 Symbol description

Symbol	Definition
$[N]$	the set $\{1, 2, \cdots, N\}$
$\mathbb{F}_p$	finite field determined by prime number p
I_i	the ith unit vector of dimension m
1^k	the security parameter
V_i	the ith vector space
n	the dimension of each vector to be transmitted
m	the dimension of the unit vector attached to the original vector
$N = n + m$	the upper bound of the size of the signed vectors
$c \xleftarrow{\$} \mathbb{F}_p$	the process of uniformly randomly choosing an element c from $\mathbb{F}_p$

1.3 Organization

The rest of this paper is organized as follows: In Section 2, we present preliminaries, including basic concepts of network coding and complex assumptions. In Section 3, we introduce the notion of the CL-LHS scheme for network coding and give its security model. We construct a concrete CL-LHS scheme and prove its security in Sections 4 and 5 respectively. We present the efficiency comparison in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

2.1 Linear network coding

In the network coding model, three stages are executed to complete the file transmission:

– The file to be transferred is treated as a set of n-dimensional vectors $\bar{\boldsymbol{v}}_1, \cdots, \bar{\boldsymbol{v}}_m \in \mathbb{F}_p^n$, where p is a prime number. Before transmission, the source node augments each of them as

$$\boldsymbol{v}_i = (\bar{\boldsymbol{v}}_i, \overbrace{\underbrace{0, \cdots, 0, 1}_{i}, 0, \cdots, 0}^{m}) \in \mathbb{F}_p^{n+m},$$

In this way, the vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$ form the basis of a subspace $V \subset \mathbb{F}_p^{n+m}$, which is called a properly augmented basis.

– Upon receiving the packets (i.e., vectors) $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_l \in \mathbb{F}_p^{n+m}$ on its incoming edges, an intermediate node computes a linear combination, namely, the vector $\boldsymbol{w} = \sum_{i=1}^{l} c_i \boldsymbol{w}_i$, for $c_i \xleftarrow{\$} \mathbb{F}_p$. Then vector $\boldsymbol{w}$ is transmitted on its outgoing edges.

– To recover the original file, a destination node (i.e., receiver) must receive m linearly independent vectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m$ of the form $\boldsymbol{w}_i = (\boldsymbol{w}_i^L, \boldsymbol{w}_i^R)$, where $\boldsymbol{w}_i^L (\boldsymbol{w}_i^R)$ denotes the left-most n (right-most m) positions of the vector. The receiver then computes an $m \times m$ matrix G such that

$$G = \begin{pmatrix} \boldsymbol{w}_1^R \\ \vdots \\ \boldsymbol{w}_m^R \end{pmatrix}^{-1}$$

Finally, the original file can be recovered by computing

$$\begin{pmatrix} \bar{\boldsymbol{v}}_1 \\ \vdots \\ \bar{\boldsymbol{v}}_m \end{pmatrix} = G \cdot \begin{pmatrix} \boldsymbol{w}_1^L \\ \vdots \\ \boldsymbol{w}_m^L \end{pmatrix}.$$

2.2 Bilinear pairing

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be two cyclic groups with the same order p and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a map; e is a bilinear pairing if it has the following three properties:

(1) **Bilinearity**: For any $a, b \in \mathbb{Z}_p^*$ and $g, h \in \mathbb{G}_1$, $e(g^a, h^b) = e(g, h)^{ab}$.
(2) **Non-degeneracy**: There exist $g, h \in \mathbb{G}_1$, such that $e(g, h) \neq 1$.
(3) **Computability**: For any $g, h \in \mathbb{G}_1$, there is an efficient algorithm to compute $e(g, h)$.

2.3 Computational Bilinear Diffie-Hellman problem

Definition 1 (CDH Problem) Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear pairing. Given (g, g^a, g^b), where $a, b \in \mathbb{Z}_p^*$ are unknown numbers, compute the value of g^{ab}.

3 Definitions and security model

3.1 Certificateless linearly homomorphic signature

Definition 2 (Certificateless Linearly Homomorphic Signature Scheme) A certificateless linearly homomorphic signature scheme consists of a tuple of (probabilistic) polynomial-time algorithms (**Setup, Extract-Partial-Private-Key, Set-Secret-Value, Set-Private-Key, Set-Public-Key, CL-HSign, CL-Combine, CL-Verify**) with the following functionality:

– **Setup** $(1^k, N, m)$**:** When a security parameter 1^k and two integers $N, m \geq 1$ are input, this algorithm outputs the system parameters params and a master key m_{sk}.
– **Extract-Partial-Private-Key (params, ID,** m_{sk} **):** This algorithm takes as input m_{sk} and a user's identity ID and outputs the user's partial private key D_{ID}.

- **Set-Secret-Value (params, ID):** This algorithm takes as input params and a user's identity ID and outputs the user's secret value x_{ID}.
- **Set-Private-Key (params, D_{ID}, x_{ID}):** This algorithm takes as input params, the partial private key D_{ID} and the secret value x_{ID}, it generates a user's full private key, referred to as SK_{ID}.
- **Set-Public-Key (params, x_{ID}):** This algorithm takes as input params and a secret value x_{ID}, and it generates the public key PK_{ID} of the identity ID.
- **CL-HSign (params, ID, SK_{ID}, τ, $\boldsymbol{v}$):** For the input params, a user's identity ID, a full private key SK_{ID}, a file identifier $\tau \in \{0,1\}^k$ and a vector $\boldsymbol{v} \in \mathbb{F}_p^N$, this algorithm outputs a signature σ.
- **CL-Combine (params, ID, PK_{ID}, τ, $\{(c_i, \sigma_i)\}_{i=1}^l$):** For the input params, a user's identity ID, a public key PK_{ID}, a file identifier τ, and a set of tuples $\{(c_i, \sigma_i)\}_{i=1}^l$ with $c_i \in \mathbb{F}_p$, where σ_i is a signature on the vector $\boldsymbol{v}_i$, this algorithm outputs a signature σ on the vector $\boldsymbol{v} = \sum_{i \in [l]} c_i \boldsymbol{v}_i$.
- **CL-Verify (params, ID, PK_{ID}, τ, $\boldsymbol{y}$, σ):** For the input params, a user's identity ID, a public key PK_{ID}, a file identifier τ, a vector $\boldsymbol{y} \in \mathbb{F}_p^N$ and a signature σ, this algorithm returns either 1 (accept) or 0 (reject).

Setup and **Extract-Partial-Private-Key** are assumed to be run by the KGC. Once a partial private key generated by the KGC is given to a user via a secure channel, the user performs the **Set-Secret-Value** algorithm.

Correctness. We require that for each key pair (SK_{ID}, PK_{ID}) output by **Setup, Set-Private-Key, Set-Public-Key**, the following hold:

(1) For all $\tau \in \{0,1\}^k$ and $\boldsymbol{v} \in \mathbb{F}_p^N$, if $\sigma \leftarrow$ **CI-HSign** $(ID, SK_{ID}, \tau, \boldsymbol{v})$, then **CL-Verify** $(ID, PK_{ID}, \tau, \boldsymbol{v}, \sigma)$=1.

(2) For all τ and all sets of triples $\{(c_i, \sigma_i, \boldsymbol{v}_i)\}_{i=1}^l$, if **CL-Verify**$(ID, PK_{ID}, \tau, \boldsymbol{v}_i, \sigma_i)$=1 for each $i \in [l]$, then **CL-Verify** $(ID, PK_{ID}, \tau, \sum_{i=1}^l c_i \boldsymbol{v}_i,$ **CL-Combine** $\{ID, \tau, (c_i, \sigma_i)\}_{i=1}^l)$=1.

3.2 System models

Figure 3 shows the system model of the certificateless linearly homomorphic signature scheme for network coding-enabled IoT environments, which consists of three parts: a key generation center (KGC), an IoT device and receivers. The KGC is responsible for the system setup and the calculation of partial private keys for each IoT device. The KGC generates system parameters and sends them to all entities, and partial private keys are sent to each entity through a secure channel. IoT devices with limited computing power and storage space are able to collect data from the physical world. To ensure data integrity and authenticity, each IoT device processes the collected raw data before signing it and then outputs the data, signature, and public key. In the process of file transmission, the certificateless linearly homomorphic signature scheme resists pollution attacks on the network coding-enabled network. Finally, the receivers recover the initial file.

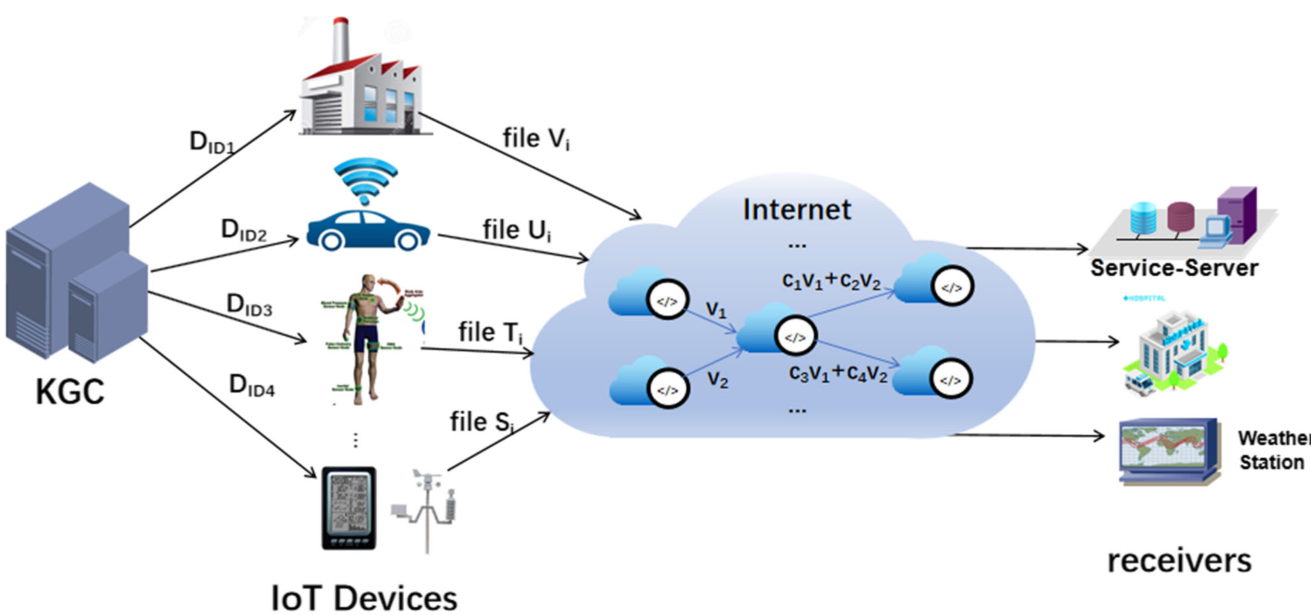

Fig. 3 System model of the CL-LHS for network coding-enabled IoT environments

3.3 Security models

In the security model of certificateless linearly homomorphic signatures, two types of adversaries with different capabilities are considered.

Type I adversary ($\mathcal{A}_{\mathcal{I}}$): This type of adversary cannot access the master key of the system but is allowed to replace the public key of any entity with a value of his choice because of the uncertified nature of the user's public key.

Type II adversary ($\mathcal{A}_{\mathcal{II}}$): This type of adversary can access the system's master key but cannot initiate public key replacement attacks.

The unforgeability of the certificateless linearly homomorphic signature scheme against adaptively chosen dataset attacks can be characterized by the following two games between challengers and adversaries ($\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$).

For adversaries in Game 1, we make the following restrictions:

(1) The adversary cannot extract the full private key for the challenge identity ID^*.
(2) The challenge identity ID^* cannot be one that has been replaced with a public key and had a partial private key extracted.

Game 1 In this game, $\mathcal{A}_{\mathcal{I}}$ interacts with the challenger C.

– **Setup:** The challenger C runs **Setup** $(1^k, N, m)$ to generate the system parameters params and master key m_{sk}. It then gives params to $\mathcal{A}_{\mathcal{I}}$ while keeping m_{sk} secret.
– **Queries:** Adversary $\mathcal{A}_{\mathcal{I}}$ performs the following oracle queries but is subject to the above restrictions.

– *Partial Private Key Extraction*: Given an identity ID, the challenger computes the partial private key D_{ID} and returns it to $\mathcal{A}_{\mathcal{I}}$.

– *Secret Value Extraction* : Given a user's identity ID, the challenger returns the user's secret value x_{ID} to $\mathcal{A}_{\mathcal{I}}$.

– *Public Key Queries* : On receiving such a query with an ID, the challenger computes the corresponding public key PK_{ID} and returns it to $\mathcal{A}_{\mathcal{I}}$.

– *Replace Public Key* : $\mathcal{A}_{\mathcal{I}}$ may replace a public key with a value chosen by him.

– *Signing Queries* : $\mathcal{A}_{\mathcal{I}}$ issues a sequence of queries adaptively for the vector subspaces $V_i \subset \mathbb{F}_p^N$. For each i, the challenger chooses an identifier τ_i uniformly from $\{0,1\}^k$ and returns τ_i and $\sigma_i \leftarrow$ **CL-HSign**$(ID, SK_{ID}, \tau_i, V_i)$ to $\mathcal{A}_{\mathcal{I}}$.
– **Output:**$\mathcal{A}_{\mathcal{I}}$ outputs an identifier τ^*, a nonzero vector $\boldsymbol{v}^* \in \mathbb{F}_p^N$, and a signature σ^* corresponding to a challenge identity ID^* and a public key PK_{ID^*}.

The adversary wins if **CL-Verify** $(ID^*, PK_{ID^*}, \tau^*, \boldsymbol{v}^*, \sigma^*) = 1$ and one of the following conditions is met:

(1) $\tau^* \neq \tau_i$ for all τ_i that appear in the signing queries (Type 1 forgery).
(2) $\tau^* = \tau_i$ for some i but $\boldsymbol{v}^* \notin V_i$ (Type 2 forgery).

The advantage of $\mathcal{A}_{\mathcal{I}}$ winning Game 1 is denoted as $Adv_{\mathcal{A}_{\mathcal{I}}}^{CL-LHS}(k)$.

Game 2 We set the semi-trusted KGC as the adversary in this game.

– **Setup:** The challenger generates the system parameter params and master key m_{sk}, then returns params and m_{sk} to $\mathcal{A}_{\mathcal{II}}$.
– **Queries:** Adversary $\mathcal{A}_{\mathcal{II}}$ initiates a sequence of queries adaptively for polynomial-many times, including *Secret Value Extraction*, *Public Key Queries* and *Signing Queries*. The queries /response method is the same as that in Game 1, except that the adversary is $\mathcal{A}_{\mathcal{II}}$.
– **Output:** $\mathcal{A}_{\mathcal{II}}$ outputs an identifier τ^*, a nonzero vector $\boldsymbol{v}^* \in \mathbb{F}_p^N$, and a signature σ^* corresponding to a challenge identity ID^*.

The adversary wins if **CL-Verify** $(ID^*, PK_{ID^*}, \tau^*, \boldsymbol{v}^*, \sigma^*) = 1$, ID^* has not been issued as a secret value extraction, and one of the following conditions is met:

(1) $\tau^* \neq \tau_i$ for all τ_i that appear in the signing queries (Type 1 forgery).
(2) $\tau^* = \tau_i$ for some i, but $\boldsymbol{v}^* \notin V_i$ (Type 2 forgery).

The advantage of $\mathcal{A}_{\mathcal{II}}$ winning Game 2 is denoted as $Adv_{\mathcal{A}_{\mathcal{II}}}^{CL-LHS}(k)$.

Definition 3 (**Unforgeability**) We say that a certificateless linearly homomorphic signature scheme is unforgeable against adaptively chosen dataset attacks for polynomial-time adversaries $\mathcal{A}_i$ if $Adv_{\mathcal{A}_i}^{CL-LHS}(i = I, II)$ in the above games is negligible

4 Proposed CL-LHS scheme

In this section, we describe the proposed certificateless linearly homomorphic signature scheme, which is composed of eight polynomial time algorithms.

– **Setup:** Taking as input a security parameter 1^k and two positive integers N, m, the algorithm performs the following steps:

(1) Select two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of the same prime order p, and choose a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
(2) Choose a generator $g \in \mathbb{G}_1$, select a random number $s \in \mathbb{F}_p^*$ as the master key m_{sk}, and set $P_{pub} = g^s$.
(3) Choose four different cryptographic hash functions H, H_1, H_2 and H_3, each of which maps $\{0,1\}^*$ to $\mathbb{G}_1$. Then, publish the system parameters params $= (k, \mathbb{G}_1, \mathbb{G}_2, e, p, g, H, H_1, H_2, H_3)$, and keep m_{sk} secret.

- **Extract-Partial-Private-Key:** Taking as inputs params, the master key s, and a user's identity $ID \in \{0,1\}^*$, the KGC executes the following steps:

(1) Compute $Q_{ID} = H(ID)$.
(2) Output the partial private key $D_{ID} = (Q_{ID})^s$.

- **Set-Secret-Value:** Taking as input an identity ID, this algorithm chooses $x_{ID} \in \mathbb{F}_p^*$ at random and sets x_{ID} as the user's secret value.
- **Set-Private-key:** Taking as input params and a user's identity ID, this algorithm outputs the user's full private key $SK_{ID} = (D_{ID}, x_{ID})$.
- **Set-Public-Key:** Taking as input params and a user's secret value x_{ID}, this algorithm generates the user's public key $PK_{ID} = g^{x_{ID}}$.
- **CL-Hsign:** Assume that an initially empty list L has stored all previously returned identifiers τ with the related information (r, U) defined below. Taking as input a signer's private key $SK_{ID} = (D_{ID}, x_{ID})$ and identity ID, an identifier $\tau \in \{0,1\}^k$, and a vector $\boldsymbol{v} = (v_1, \cdots, v_N) \in \mathbb{F}_p^N$, the algorithm responds as follows:

(1) If τ appears in L, the algorithm recovers the associated (r, U) from L.
(2) Otherwise, it selects $r \in \mathbb{F}_p^*$ randomly, sets $U = g^r$ and stores (r, U) into L.

Then, the algorithm computes

$$\begin{aligned} T_i &= H_1(ID, P_{pub}, \tau, U, i), \\ T_i' &= H_2(ID, PK_{ID}, \tau, i), \\ T &= H_3(ID, PK_{ID}), \\ W &= (D_{ID})^{\sum_{i\in[N]} v_i} \left(\prod_{i\in[N]} T_i^{v_i}\right)^r \left(\prod_{i\in[N]} T_i'^{v_i} \cdot T^{\sum_{i\in[N]} v_i}\right)^{x_{ID}} \end{aligned}$$

and outputs the signature $\sigma = (U, W)$.

- **CL-Combine:** Given an identity ID and corresponding public key $PK_{ID} = g^{x_{ID}}$, an identifier τ, and $\{(c_i, \sigma_i)\}_{i=1}^l$ with $c_i \in \mathbb{F}_p$, where $\sigma_i = (U, W_i)$, this algorithm computes $W = \prod_{i\in[l]} W_i^{c_i}$ and outputs (U, W).
- **CL-Verify:** Given an identity ID and corresponding public key $PK_{ID} = g^{x_{ID}}$, an identifier τ, a signature $\sigma = (U, W)$, and a vector $\boldsymbol{v} = (v_1, \cdots, v_N) \in \mathbb{F}_p^N$, the algorithm accepts the signature if the following equation holds:

$$e(W, g) = e(Q_{ID}^{\sum_{i\in[N]} v_i}, P_{pub}) \cdot e\left(\prod_{i\in[N]} T_i^{v_i}, U\right) \cdot e\left(\prod_{i\in[N]} T_i'^{v_i} \cdot T^{\sum_{i\in[N]} v_i}, PK_{ID}\right)$$

Correctness Given an identity ID and public key PK_{ID}, an identifier τ, a vector $\boldsymbol{v} = (v_1, \cdots, v_N) \in \mathbb{F}_p^N$, and a signature σ, if $\sigma \leftarrow$**CL-HSign** $(ID, SK_{ID}, \tau, \boldsymbol{v})$, the correctness of the scheme can be obtained by the following equation:

$$\begin{aligned} e(W, g) &= e(D_{ID}^{\sum_{i\in[N]} v_i}, g) \cdot e\left(\left(\prod_{i\in[N]} T_i^{v_i}\right)^r, g\right) \cdot e\left(\left(\prod_{i\in[N]} T_i'^{v_i} \cdot T^{\sum_{i\in[N]} v_i}\right)^{x_{ID}}, g\right) \\ &= e(Q_{ID}^{\sum_{i\in[N]} v_i}, P_{pub}) \cdot e\left(\prod_{i\in[N]} T_i^{v_i}, U\right) \cdot e\left(\prod_{i\in[N]} T_i'^{v_i} \cdot T^{\sum_{i\in[N]} v_i}, PK_{ID}\right) \end{aligned}$$

Thus, the verification algorithm on the original signature σ is correct.

Furthermore, given an identity ID and public key PK_{ID}, an identifier τ and a set of triples $\{(c_i, \sigma_i, \boldsymbol{v}_i)\}_{i=1}^l$, where $\sigma_i = (U, W_i)$ and $\boldsymbol{v}_i = (v_{i1}, \cdots, v_{iN})$, if $\sigma_i \leftarrow$ **CL-HSign**$(ID, SK_{ID}, \tau, \boldsymbol{v}_i)$, we need to prove that $\sigma = (U, W = \prod_{i\in[l]} W_i^{c_i})$ is a signature on $\boldsymbol{y} = (y_1, \cdots, y_N) = \sum_{i\in[l]} c_i \boldsymbol{v}_i$. By the correctness of each signature, we have

$$e(W_i, g) = e\left(Q_{ID}^{\sum_{j\in[N]} v_{ij}}, P_{pub}\right) \cdot e\left(\prod_{j\in[N]} T_j^{v_{ij}}, U\right) \cdot e\left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}, PK_{ID}\right)$$

Thus, by the bilinear property, we have

$$\begin{aligned} e(W, g) &= \prod_{i\in[l]} e(W_i, g)^{c_i} \\ &= e\left(Q_{ID}^{\sum_{i\in[l]}\sum_{j\in[N]} c_i v_{ij}}, P_{pub}\right) \cdot e\left(\prod_{j\in[N]} T_j^{\sum_{i\in[l]} c_i v_{ij}}, U\right) \\ &\quad \cdot e\left(\prod_{j\in[N]} T_j'^{\sum_{i\in[l]} c_i v_{ij}} \cdot T^{\sum_{i\in[l]}\sum_{j\in[N]} c_i v_{ij}}, PK_{ID}\right) \\ &= e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} y_j} \cdot e\left(\prod_{j\in[N]} T_j^{y_j}, U\right) \cdot e\left(\prod_{j\in[N]} T_j'^{y_j} \cdot T^{\sum_{j\in[N]} y_j}, PK_{ID}\right). \end{aligned}$$

Therefore, the verification algorithm on a combined signature σ is correct.

5 Security analysis

In this subsection, we analyze the security of the proposed scheme.

Theorem 1 *Our certificateless linearly homomorphic signature scheme is unforgeable in the random oracle model assuming that the CDH problem in $\mathbb{G}_1$ is infeasible.*

This Theorem 1 is derived from the following two lemmas, with Definition 3.

Lemma 1 *For any polynomial-time adversary $\mathcal{A}_\mathcal{I}$, our certificateless linearly homomorphic signature scheme is unforgeable in the random oracle model assuming that the CDH problem in $\mathbb{G}_1$ is infeasible.*

Proof Idea: during the adversary's query process, the challenger assigns g^a and g^b in the random challenge of the CDH problem to some items, so the signature contains the term g^{ab}. In order to ensure that the unknown quantity g^{ab} does not affect the challenger's response to the signing queries, the challenger carefully sets the hash values $T_i (i \in [N])$ and U value in the signing queries, so that the item that bring in the special T_i and U values can eliminate the one containing g^{ab}, while in the output phase, $e(g^{ab}, g)$ can be retained. If an adversary outputs a valid forgery, the value g^{ab} can be solved from the verification algorithm equation by using the non-degeneracy of bilinear pairs. Moreover, it is proved that the probability of aborting simulation is negligible and the simulation process is complete.

Proof Assume that $\mathcal{A}_\mathcal{I}$ represents a third-party attack against the unforgeability of our CL-LHS scheme. We construct a simulator $\mathcal{C}$ that uses $\mathcal{A}_\mathcal{I}$ as a subroutine to solve the CDH problem. According to the definition of Game 1, adversary $\mathcal{A}_\mathcal{I}$ eventually outputs either a Type 1 forgery or a Type 2 forgery. $\mathcal{C}$ guesses the type of forgery that will be output by $\mathcal{A}_\mathcal{I}$ based on the result of flipping a coin randomly. Clearly, $\mathcal{C}$ guesses correctly with a probability of $\frac{1}{2}$.

Case 1 (Type 1 forgery:) In this case, $\mathcal{C}$ has guessed that $\mathcal{A}_\mathcal{I}$ will output a Type 1 forgery. Given a random challenge $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, g^a, g^b)$ of the CDH problem, the goal of $\mathcal{C}$ is to compute the value of g^{ab}. $\mathcal{C}$ interacts with $\mathcal{A}_\mathcal{I}$ as follows:

- **Setup:** $\mathcal{C}$ runs Setup and sets $P_{pub} = g^a$ and params= $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, P_{pub} = g^a)$. It invokes $\mathcal{A}_\mathcal{I}$ on the input params.
- **Queries:** $\mathcal{A}_\mathcal{I}$ can issue queries to the following oracles simulated by $\mathcal{C}$.
  - *H Queries* : Suppose that $\mathcal{A}_\mathcal{I}$ makes at most q_H H queries. A list is maintained by $\mathcal{C}$, referred to as L_H. $\mathcal{C}$ randomly chooses $k \in \{1, 2, \cdots, q_H\}$ and guesses that the k-th identity ID_k of the queries initiated by $\mathcal{A}_\mathcal{I}$ is the challenge identity. When $\mathcal{A}_\mathcal{I}$ sends an H query on identity ID, $\mathcal{C}$ responds as follows:

(1) If this is the k-th query, e.g., $ID = ID_k$, output $Q(ID) = H(ID) = g^b$ and add $< ID, g^b, \perp >$ to L_H.

(2) Otherwise, choose a random number $w_{ID} \in \mathbb{F}_p^*$, output $Q(ID) = H(ID) = g^{w_{ID}}$ and add $< ID, H(ID), w_{ID} >$ to L_H.

  - *Partial Private Key Extraction* : $\mathcal{C}$ maintains a list L_{part} that is initially empty. When $\mathcal{A}_\mathcal{I}$ asks for the partial private key of identity ID, if $ID = ID_k$, $\mathcal{C}$ aborts. Otherwise, it recovers the tuple $< ID, H(ID), w_{ID} >$ from L_H and returns the partial private key $D_{ID} = (g^a)^{w_{ID}}$ to $\mathcal{A}_\mathcal{I}$. Then, it stores $< ID, D_{ID} >$ in the list L_{part}.
  - *Public Key Queries* : $\mathcal{C}$ maintains a list L_{PK} that is initially empty. Given an identity ID, $\mathcal{C}$ randomly chooses $x_{ID} \in \mathbb{F}_p^*$ as the secret value. Then, $\mathcal{C}$ returns the public key $PK_{ID} = g^{x_{ID}}$ to $\mathcal{A}_\mathcal{I}$ and saves $< ID, PK_{ID}, x_{ID} >$ in L_{PK}.
  - *Private Key Extraction* : $\mathcal{C}$ maintains a list L_{SK} that is initially empty. Given an identity ID, $\mathcal{C}$ performs the following actions:

(1) If $ID \neq ID_k$, it recovers the tuple $< ID, H(ID), w_{ID} >$ from L_H and $< ID, PK_{ID}, x_{ID} >$ from L_{PK}. Then, $\mathcal{C}$ returns the secret key $SK_{ID} = ((g^a)^{w_{ID}}, x_{ID})$ to $\mathcal{A}_\mathcal{I}$ and adds $< ID, SK_{ID} >$ to L_{SK}.

(2) Otherwise, $\mathcal{C}$ aborts.

  - *Replace Public Key*: Suppose $\mathcal{A}_\mathcal{I}$ sends a query with the input (ID, PK'_{ID}). If the list L_{PK} contains a tuple $< ID, PK_{ID}, x_{ID} >$, $\mathcal{C}$ sets $PK_{ID} = PK'_{ID}$ and $x_{ID} = \perp$.
  - *H_1 Queries*: Suppose $(ID, P_{pub}, \tau, U, i)$ is submitted to oracle $H_1(\cdot)$. $\mathcal{C}$ first scans for $< (ID, P_{pub}, \tau, U, i), T_i, t_i >$ in the list L_{H_1} to check whether T_i has already been defined. If so, it returns the previously defined value. Otherwise, $\mathcal{C}$ randomly chooses a number $t_i \in \mathbb{F}_p^*$, returns $T_i = g^{t_i}$ to $\mathcal{A}_\mathcal{I}$ as the hash value of $H_1(ID, P_{pub}, \tau, U, i)$, and stores the value in the list L_{H_1}.
  - *H_2 Queries* : Suppose (ID, PK_{ID}, τ, i) is submitted to oracle $H_2(\cdot)$. $\mathcal{C}$ first scans for $< (ID, PK_{ID}, \tau, i), T'_i, t'_i >$ in the list L_{H_2} to check whether T'_i has already been defined. If so, it returns the previously defined value. Otherwise, $\mathcal{C}$ randomly chooses a number

$t'_i \in \mathbb{F}_p^*$, returns $T'_i = g^{t'_i}$ to $\mathcal{A}_{\mathcal{I}}$ as the hash value of $H_2(ID, PK_{ID}, \tau, i)$, and stores the value into the list L_{H_2}.

- H_3 *Queries* : $\mathcal{C}$ maintains a list L_{H_3} containing tuples $< (ID, PK_{ID}), T, t >$. Upon receiving $\mathcal{A}_{\mathcal{I}}$'s query on (ID, PK_{ID}), if it already exists in L_{H_3}, $\mathcal{C}$ returns T. Otherwise, $\mathcal{C}$ chooses at random a number $t \in \mathbb{F}_p^*$, returns $T = g^t$ to $\mathcal{A}_{\mathcal{I}}$ as the hash value of $H_3(ID, PK_{ID})$, and stores the value into the list L_{H_3}.
- *Signing Queries* : Given an identity ID, a vector space $V \subset \mathbb{F}_p^N$ is described by augmented basis vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m \in \mathbb{F}_p^N$, where $\boldsymbol{v}_i = (v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0)$.

If ID is the challenge identity, i.e., $ID = ID_k$, $\mathcal{C}$ performs the following steps:

(1) Randomly choose an identifier $\tau \leftarrow \{0, 1\}^k$ and numbers $r, u_i \in \mathbb{F}_p^*(i \in [N])$, and set $U = P_{pub}^r = (g^a)^r$.
(2) Define the hash values of $H_1(ID, P_{pub}, \tau, U, i)$ as $T_i = (\frac{g^{u_i}}{Q_{ID}})^{r^{-1}} \in \mathbb{G}_1$. $\mathcal{C}$ aborts if $H_1(ID, P_{pub}, \tau, U, i)$ has already been queried for some $i \in [N]$.
(3) Recover $T'_i(i \in [N])$ and T from L_{H_2} and L_{H_3}, respectively; if there are no such items, $\mathcal{C}$ makes queries to oracles $H_2(\cdot)$ and $H_3(\cdot)$.
(4) Finally, $\mathcal{C}$ computes

$$W_i = (P_{pub})^{\sum_{j\in[N]} u_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + t \sum_{j\in[N]} v_{ij}}$$

Now $\sigma_i = (U, W_i)(i \in [m])$ are returned to $\mathcal{A}_{\mathcal{I}}$; σ_i is a valid signature, since

$$\underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e\left(\prod_{j\in[N]} T_j^{v_{ij}}, U\right)}_{2} \cdot e\left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}, PK_{ID}\right) \quad (1)$$

$$= \underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e\left(\prod_{j\in[N]} (\frac{g^{u_j}}{Q_{ID}})^{r^{-1} v_{ij}}, P_{pub}^r\right)}_{2'} \cdot e\left(g^{\sum_{j\in[N]} t'_j v_{ij} + t \sum_{j\in[N]} v_{ij}}, PK_{ID}\right) \quad (2)$$

$$= \underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e(Q_{ID}, P_{pub})^{-\sum_{j\in[N]} v_{ij}}}_{2'_{(1)}} \cdot \underbrace{e(g^{\sum_{j\in[N]} u_j v_{ij}}, P_{pub})}_{2'_{(2)}}$$
$$\cdot e\left(g^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}, PK_{ID}\right) \quad (3)$$

$$= e\left((P_{pub})^{\sum_{j\in[N]} u_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}, g\right) \quad (4)$$

$$= e(W_i, g) \quad (5)$$

The derivation process of the core part of the above series of equations is shown in note.[1]

Otherwise,[2] $\mathcal{C}$ randomly chooses an identifier $\tau \leftarrow \{0, 1\}^k$ and a number $r \in \mathbb{F}_p^*$, sets $U = g^r$, and computes

$$W_i = (g^a)^{w_{ID} \sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}$$

Now, $\sigma_i = (U, W_i)(i \in [m])$ are returned to $\mathcal{A}_{\mathcal{I}}$; σ_i is a valid signature, since

$$\begin{aligned} & e(W_i, g) \\ = \ & e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}} \cdot e\left(\prod_{j\in[N]} T_j^{v_{ij}}, U\right) \cdot e\left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}, PK_{ID}\right) \\ = \ & e(g^{w_{ID}}, g^a)^{\sum_{j\in[N]} v_{ij}} \cdot e(U^{\sum_{j\in[N]} t_j v_{ij}}, g) \cdot e\left((PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + t \sum_{j\in[N]} v_{ij}}, g\right) \\ = \ & e\left((g^a)^{w_{ID} \sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}, g\right) \end{aligned}$$

- **Output:** Eventually, $\mathcal{A}_{\mathcal{I}}$ outputs a tuple $(ID^*, PK_{ID^*}, \boldsymbol{v}^*, \tau^*, \sigma^*)$, where $\boldsymbol{v}^* = (v_1^*, \cdots, v_N^*)$ and $\sigma^* = (U^*, W^*)$. If $ID^* \neq ID_k$, then $\mathcal{C}$ aborts. Otherwise, for each $i \in [N]$, it retrieves the items T_i^* from L_{H_1}, the items $T_i'^*$ from L_{H_2}, and the item T^* from L_{H_3}; if there are no such items, $\mathcal{C}$ makes queries to the corresponding oracle. If $\mathcal{A}_{\mathcal{I}}$ successfully outputs Type 1 forgery signatures, the file identifier $\tau^* \neq \tau_i$ for all τ_i that appear in signing queries, and note that $T_i^* = g^{t_i^*}$,[3] $T_i'^* = g^{t_i'^*}, T^* = g^{t^*}$, then the following equation holds:

$$\begin{aligned} & e(W^*, g) \\ = \ & e(Q_{ID^*}, P_{pub})^{\sum_{i\in[N]} v_i^*} \cdot e\left(\prod_{i\in[N]} (T_i^*)^{v_i^*}, U^*\right) \cdot e\left(\prod_{i\in[N]} (T_i'^*)^{v_i^*} \cdot (T^*)^{\sum_{i\in[N]} v_i^*}, PK_{ID^*}\right) \\ = \ & e(Q_{ID^*}, P_{pub})^{\sum_{i\in[N]} v_i^*} \cdot e\left((U^*)^{\sum_{i\in[N]} t_i^* v_i^*}, g) \cdot e((PK_{ID^*})^{\sum_{i\in[N]} t_i'^* v_i^* + t^* \sum_{i\in[N]} v_i^*}, g\right). \end{aligned}$$

[1] Since we embed the hard problem in the term 1 of Eq. 1, that is, $Q_{ID} = g^b$, $P_{pub} = g^a$. In order to successfully answer the signing query, our idea is to eliminate item 1 by carefully setting the values of $T_i (i \in [N = n+m])$ and U while ensuring that the values of T_i and U are random ($T_i = (\frac{g^{u_i}}{Q_{ID}})^{r^{-1}}$, $U = P_{pub}^r = (g^a)^r$). The item $2'$ in (2) is further arranged to obtain items $2'_1$ and $2'_2$ in (3). It is not difficult to find that item $2'_1$ can eliminate item 1, because item $2'_1$ and item 1 are inverses of each other in group $\mathbb{G}_2$.

[2] In this case, $ID \neq ID_k$, then $Q_{ID} = g^{w_{ID}}$, where w_{ID} is a known random number, so the required values generated in the process of various queries can be directly brought into the signature algorithm of the proposed scheme to obtain the signature.

[3] Here, the expression of hash value T_i^* is different from that of hash value T_i in signing queries. This is because if an adversary outputs a type 1 forgery, the identifier τ^* never appears in the signing query, so the hash value T_i^* corresponding to the identifier τ^* come from H_1 queries.

Therefore, we have the following equation:

$$e\left(\frac{W^*}{(U^*)^{\sum_{i\in[N]} t_i^* v_i^*}\cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* v_i^* + t^* \sum_{i\in[N]} v_i^*}}, g\right)$$
$$= e(Q_{ID^*}, P_{pub})^{\sum_{i\in[N]} v_i^*}$$
$$= e(g^b, g^a)^{\sum_{i\in[N]} v_i^*}$$
$$= e(g, g)^{ab\cdot \sum_{i\in[N]} v_i^*}$$

Thus, by the nondegenerate property, the value of g^{ab} is the following expression:

$$\left(\frac{W^*}{(U^*)^{\sum_{i\in[N]} t_i^* v_i^*}\cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* v_i^* + \sum_{i\in[N]} t^* v_i^*}}\right)^{\frac{1}{\sum_{i\in[N]} v_i^*}}.$$

Now, we evaluate $\mathcal{C}$'s probability of success.

We first analyze the probability of aborts in handling a signing query. The probability of the event that $\mathcal{C}$ responds to two distinct signature queries by choosing the same identifier τ is at most $\frac{q_s^2}{2^k}$, while the probability of the event that $\mathcal{A_I}$ has already requested the value of $H_1(ID, P_{pub}, \tau, U, i)$ for some i is at most $\frac{q_{H_1}\cdot q_s}{2^k}$.

Then, we can readily check that the probability of not aborting in key extraction queries and in the output stage is $(1-\frac{1}{q_H})^{q_{part}}$ and $\frac{1}{q_H}$, respectively, where q_s, q_H, q_{part} are the numbers of signing queries, H hash queries and partial private key extractions performed by $\mathcal{A_I}$.

Therefore, if $\mathcal{A_I}$ has an advantage $Adv_{\mathcal{A_I}}^{CL-LHS}(k)$ in forging a signature in Game 1, then $\mathcal{C}$ solves the CDH problem with probability

$$\left(\frac{1}{2}Adv_{\mathcal{A_I}}^{CL-LHS}(k) - \frac{q_s^2 + q_{H_1}\cdot q_s}{2^k}\right)\cdot\left(1-\frac{1}{q_H}\right)^{q_{part}}\cdot\frac{1}{q_H}.$$

Case 2 (type 2 forgery:) In this case, $\mathcal{C}$ has guessed that $\mathcal{A_I}$ will output a type 2 forgery. Given a CDH instance, $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, g^a, g^b)$, the goal of $\mathcal{C}$ is to compute the value of g^{ab} by using $\mathcal{A_I}$ as a subroutine. $\mathcal{C}$ interacts with $\mathcal{A_I}$ as follows:

- **Setup:** $\mathcal{C}$ chooses a random number $s \in \mathbb{F}_p^*$ as the master key and sets $P_{pub} = g^s$ and params= $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, P_{pub} = g^s)$. It invokes $\mathcal{A_I}$ on input params.
- **Queries:** $\mathcal{C}$ simulates the oracle queries of $\mathcal{A_I}$ as follows:
  - H $Queries$: $\mathcal{C}$ maintains a list L_H that is initially empty. Given an identity ID, $\mathcal{C}$ chooses a random number $w_{ID} \in \mathbb{F}_p^*$, computes $Q_{ID} = g^{w_{ID}}$ as the value of $H(ID)$, returns it to $\mathcal{A_I}$, and adds $< ID, H(ID), w_{ID} >$ to L_H.
  - $H_1(H_2, H_3)$ $Queries$: Same as in **Case 1**.
  - $Partial\ Private\ Key\ Extraction$: Given an identity ID, $\mathcal{C}$ retrieves the tuple $< ID, H(ID), w_{ID} >$ from L_H and returns the partial private key $D_{ID} = H(ID)^s$ to $\mathcal{A_I}$. Then, it stores $< ID, D_{ID} >$ in a list L_{part} which is initially empty.
  - $Private\ Key\ Extraction$: $\mathcal{C}$ maintains a list L_{SK} of tuples $< ID, SK_{ID} >$. Given an identity ID, $\mathcal{C}$ recovers the tuple $< ID, D_{ID} >$ from L_{part} and chooses a random number $x_{ID} \in \mathbb{F}_p^*$ as the secret value. Then, it returns the secret key $SK_{ID} = (D_{ID}, x_{ID})$ to $\mathcal{A_I}$ and adds $< ID, SK_{ID} >$ to L_{SK}.
  - $Public\ Key\ Queries$: $\mathcal{C}$ maintains a list L_{PK} of tuples $< ID, PK_{ID} >$. Given an identity ID, $\mathcal{C}$ recovers the tuple $< ID, SK_{ID} >$ from L_{SK}, returns the public key $PK_{ID} = g^{x_{ID}}$ to $\mathcal{A_I}$ and saves $< ID, PK_{ID} >$ in L_{PK}.
  - $Replace\ Public\ Key$: Suppose $\mathcal{A_I}$ sends a query with the input (ID, PK'_{ID}). If the list L_{PK} contains the tuple $< ID, PK_{ID} >$, $\mathcal{C}$ sets $PK_{ID} = PK'_{ID}$.
  - $Signing\ Queries$: Given an identity ID and a vector space $V \subset \mathbb{F}_p^N$ described by augmented basis vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m \in \mathbb{F}_p^N$, where $\boldsymbol{v}_i = (v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0)$, $\mathcal{C}$ preforms the following steps:

(1) Randomly choose an identifier $\tau \leftarrow \{0,1\}^k$ and numbers $r, \alpha_1, \cdots, \alpha_n \in \mathbb{F}_p^*$, and set $U = (g^b)^r$.

(2) Set $n = N - m$, and for each $i \in [n]$, compute

$$T_i = H_1(ID, P_{pub}, \tau, U, i) = (g^a)^{\alpha_i}$$

for each $i \in [m]$, compute

$$\beta_i = -\sum_{j\in[n]} \alpha_j v_{ij},$$
$$T_{n+i} = H_1(ID, P_{pub}, \tau, U, n+i) = (g^a)^{\beta_i},$$

and set $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m)$. Now observe that we constructed $\boldsymbol{\alpha}$ so that $\boldsymbol{\alpha} \in V^{\perp}$ (i.e., $\boldsymbol{\alpha}\cdot\boldsymbol{v} = 0$, for all $\boldsymbol{v} \in V = Span\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\}$).[4]

[4]In detail, $\boldsymbol{\alpha}\cdot\boldsymbol{v}_i = (\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m)\cdot(v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0) = \alpha_1 v_{i1} + \cdots + \alpha_n v_{in} + \beta_i = \alpha_1 v_{i1} + \cdots + \alpha_n v_{in} + \left(-\sum_{j=1}^{n}\alpha_j v_{ij}\right) = 0$. In particular, since we set $(T_1, \cdots, T_n, T_{n+1}, \cdots, T_{n+m}) = ((g^a)^{\alpha_1}, (g^a)^{\alpha_n}, (g^a)^{\beta_1}, \cdots, (g^a)^{\beta_m})$, we have $\prod_{j\in[N]} T_j^{v_{ij}} = \prod_{j\in[n]} (g^a)^{\alpha_j v_{ij}}\cdot\prod_{j\in[m]} (g^a)^{\beta_j v_{i,(n+j)}} = (g^a)^{\boldsymbol{\alpha}\cdot\boldsymbol{v}_i} = 1$

$\mathcal{C}$ aborts if $H_1(ID, P_{pub}, \tau, U, i)$ has already been queried for some $i \in [N]$.

(3) Recover T_i', T and SK_{ID} from L_{H_2}, L_{H_3} and L_{SK}, respectively; if there are no such items, $\mathcal{C}$ makes queries on the corresponding oracle.

(4) Finally, compute

$$W_i = (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t_j' v_{ij} + t \sum_{j\in[N]} v_{ij}}$$

Now, $\sigma_i = (U, W_i)(i \in [m])$ are returned to $\mathcal{A}_{\mathcal{I}}$; we show that σ_i is a valid signature, since

$$W_i = (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot \underbrace{\left(\prod_{j\in[N]} T_j^{v_{ij}}\right)^{br}}_{①} \cdot \left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}\right)^{x_{ID}} \tag{6}$$

$$= (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot \underbrace{\left(\prod_{j\in[n]} (g^a)^{\alpha_j v_{ij}} \cdot \prod_{j\in[m]} (g^a)^{\beta_j v_{i,(n+j)}}\right)^{br}}_{②} \cdot \left(\prod_{j\in[N]} g^{t_j' v_{ij}} \cdot g^{t \sum_{j\in[N]} v_{ij}}\right)^{x_{ID}} \tag{7}$$

$$= (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot \underbrace{(g^{ab})^{(\boldsymbol{\alpha}\cdot\boldsymbol{v}_i)r}}_{③} \cdot (PK_{ID})^{\sum_{j\in[N]} t_j' v_{ij} + \sum_{j\in[N]} t v_{ij}} \tag{8}$$

$$= (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t_j' v_{ij} + \sum_{j\in[N]} t v_{ij}} \tag{9}$$

The core part of the derivation process of the above series of equations is is shown in note.[5] The first equation above comes from $U = (g^b)^r$ and the definition of the signature of the proposed scheme, the second equation comes from the introduction of specific expressions of T_j, T_j' and T. After rearrangement, the third equation is obtained. The last equation holds since we constructed $\boldsymbol{\alpha}$ such that $\boldsymbol{\alpha} \cdot \boldsymbol{v} = 0$ for all $\boldsymbol{v} \in V$. Hence, the signatures output by $\mathcal{C}$ in step (4) are valid signatures.

- **Output:** Eventually, $\mathcal{A}_{\mathcal{I}}$ outputs ID^*, PK_{ID^*}, an identifier τ^*, a nonzero vector $\boldsymbol{y} = (y_1, \cdots, y_N)$ and signatures $\sigma_i^* = (U^*, W_i^*), i \in [m]$.

  If $\mathcal{A}_{\mathcal{I}}$ successfully outputs Type 2 forgery signatures, then τ^* has been used to respond to a vector subspace V in a signature query, but $\boldsymbol{y} \notin V$, so it is known that $U^* = (g^b)^r$, $T_i^* = H_1(ID^*, P_{pub}, \tau^*, U^*, i) = (g^a)^{\alpha_i}(i \in [n])$, $T_{n+i}^* = H_1(ID^*, P_{pub}, \tau^*, U^*, n+i) = (g^a)^{\beta_i}(i \in [m])$, and **Verify** $(ID^*, PK_{ID^*}, \tau^*, \boldsymbol{y}, \sigma^*) = 1$. $\mathcal{C}$ recovers $T_i'^*$ from the list L_{H_2}, T^* from the list L_{H_3} and D_{ID^*} from the list L_{part}, note that $T_i'^* = g^{t_i'^*}$, $T^* = g^{t^*}$; then, the following equation holds:

$$e\left(\prod_{i\in[m]} (W_i^*)^{y_{n+i}}, g\right)$$
$$= e\left(Q_{ID^*}^{\sum_{i\in[N]} y_i}, P_{pub}\right) \cdot e\left(\prod_{i\in[N]} (T_i^*)^{y_i}, U^*\right) \cdot e\left(\prod_{i\in[N]} (T_i'^*)^{y_i} \cdot (T^*)^{\sum_{i\in[N]} y_i}, PK_{ID^*}\right)$$
$$= e\left((D_{ID^*})^{\sum_{i\in[N]} y_i}, g\right) \cdot e\left((g^{ab})^{(\boldsymbol{\alpha}\cdot\boldsymbol{y})r}, g\right) \cdot e\left((PK_{ID^*})^{\sum_{i\in[N]} t_i'^* y_i + t^* \sum_{i\in[N]} y_i}, g\right)$$

The first equation above comes from the verification algorithm of the proposed scheme, and the second equation comes from the concrete expression brought into T_i^*, $T_i'^*$, T^* and U^*.

Therefore, by the nondegenerate property, we have

$$\prod_{i\in[m]} (W_i^*)^{y_{n+i}} = (D_{ID^*})^{\sum_{i\in[N]} y_i} \cdot (g^{ab})^{(\boldsymbol{\alpha}\cdot\boldsymbol{y})r} \cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* y_i + t^* \sum_{i\in[N]} y_i}$$

If $\boldsymbol{\alpha} \cdot \boldsymbol{y} \neq 0$, then $\mathcal{C}$ can compute the value of g^{ab} as follows:

$$\left(\frac{\prod_{i\in[m]} (W_i^*)^{y_{n+i}}}{(D_{ID^*})^{\sum_{i\in[N]} y_i} \cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* y_i + \sum_{i\in[N]} t^* y_i}}\right)^{\frac{1}{(\boldsymbol{\alpha}\cdot\boldsymbol{y})r}}$$

Now, we evaluate $\mathcal{C}$'s probability of success.

As in the case of $\mathcal{A}_{\mathcal{I}}$ forging a Type 1 signature, the probability of $\mathcal{C}$ aborting the signing query is at most $\frac{q_s^2 + q_{H_1} \cdot q_s}{2^k}$.

Since $\mathcal{A}_{\mathcal{I}}$ outputs a Type 2 forgery, $\boldsymbol{y} \notin V$. Note that $\alpha_i(i \in [n])$ are independently and uniformly selected in $\mathbb{F}_p^*$; then, $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m)$ is uniformly random in $V^{\perp}$. Therefore, for any $\boldsymbol{y} \notin V$, $\boldsymbol{\alpha} \cdot \boldsymbol{y}$ is uniform in $\mathbb{F}_p^*$, and we find that $\boldsymbol{\alpha} \cdot \boldsymbol{y} = 0$ with probability $\frac{1}{p}$.

Therefore, if $\mathcal{A}_{\mathcal{I}}$ has an advantage $Adv_{\mathcal{A}_{\mathcal{I}}}^{CL-LHS}(k)$ in forging a signature in Game 1, then $\mathcal{C}$ can solve the CDH problem with probability

$$\left(\frac{1}{2} Adv_{\mathcal{A}_{\mathcal{I}}}^{CL-LHS}(k) - \frac{q_s^2 + q_{H_1} \cdot q_s}{2^k}\right) \cdot \left(1 - \frac{1}{p}\right).$$

□

Lemma 2 *For any polynomial-time adversary $\mathcal{A}_{\mathcal{II}}$, our certificateless linearly homomorphic signature scheme is unforgeable in the random oracle model assuming that the CDH problem in $\mathbb{G}_1$ is infeasible.*

[5] Since we embed the hard problem in the term ① of Eq. 6. In order to successfully answer the signing query, our idea is to carefully set the value of $T_i(i \in [N = n+m])$ such that $(\prod_{j\in[N]} T_j^{v_{ij}})^{br} = 1$, while ensuring that the values of T_i are random. As we know from the previous, the vector $\boldsymbol{\alpha}$ formed by the exponents of $T_i(i \in [N = n+m])$ satisfies $\boldsymbol{\alpha} \in V^{\perp}$, so term ③(= ① = ②) in Eq. 6 is equal to 1, that is, $(g^{ab})^{(\boldsymbol{\alpha}\cdot\boldsymbol{v}_i)r} = 1$.

The proof of Lemma 2 is similar with that of Lemma 1. The difference between the proof of Lemmas 2 and 1 is that the positions of embedding hard problem are different. We omit the proof here for simplicity and show the proof of Lemma 2 in Appendix.

6 Application in IoT environments and performance comparison

6.1 System model of authentication computing using CL-LHS in an IoT environment

In the IoT environment, homomorphic signatures can be used not only to protect applications based on network coding but also to perform the authentication calculation of the linear function of signed data. Although the IoT provides great convenience for production and life, the storage and calculation of massive data is still a major challenge due to limited computing power and storage resources. In recent years, cloud computing technology has developed rapidly, and some common cloud service products have been released and received wide attention. Because of its convenience and rapidity, an increasing number of users choose to upload their data to the cloud server and compute their own data. Of course, the correctness of server computing is a major issue. As the most natural application of homomorphic signatures, server computing can ensure that “correct data” is “correctly operated on” and that “correct results” are obtained in the system assuming that there are some untrusted parties (such as cloud data processors). Suppose the user wants to perform a large computation, but she does not have such a powerful resource. Then, she can use her secret key to sign a large data set and then distribute the signed data to an untrusted cloud server to calculate the data. The cloud server then derives the signature on the calculated results homomorphically. This signature can prove that the data processor outputs the correct calculation result. As shown in Fig. 4, the system model of authentication computing using certificateless linearly homomorphic signatures in the IoT environment consists of three components: a key generation center (KGC), data cloud server and IoT device.

- **KGC:** The KGC is responsible for generating system parameters and calculating partial private keys for each IoT device. Then, these partial private keys are sent to each entity through a secure channel, and system parameters are sent to all entities through a public channel.
- **IoT device:** The KGC generates a unique partial private key for each registered IoT device equipped with sensors. To ensure the integrity and authenticity of the data, each IoT device uses the system parameters and private key to sign the collected original data separately. Then, the IoT device sends the message, corresponding signature and public key to the cloud server.
- **Cloud server:** The cloud server has powerful computing power and storage space to verify the validity of all received signatures. If they are valid, homomorphic

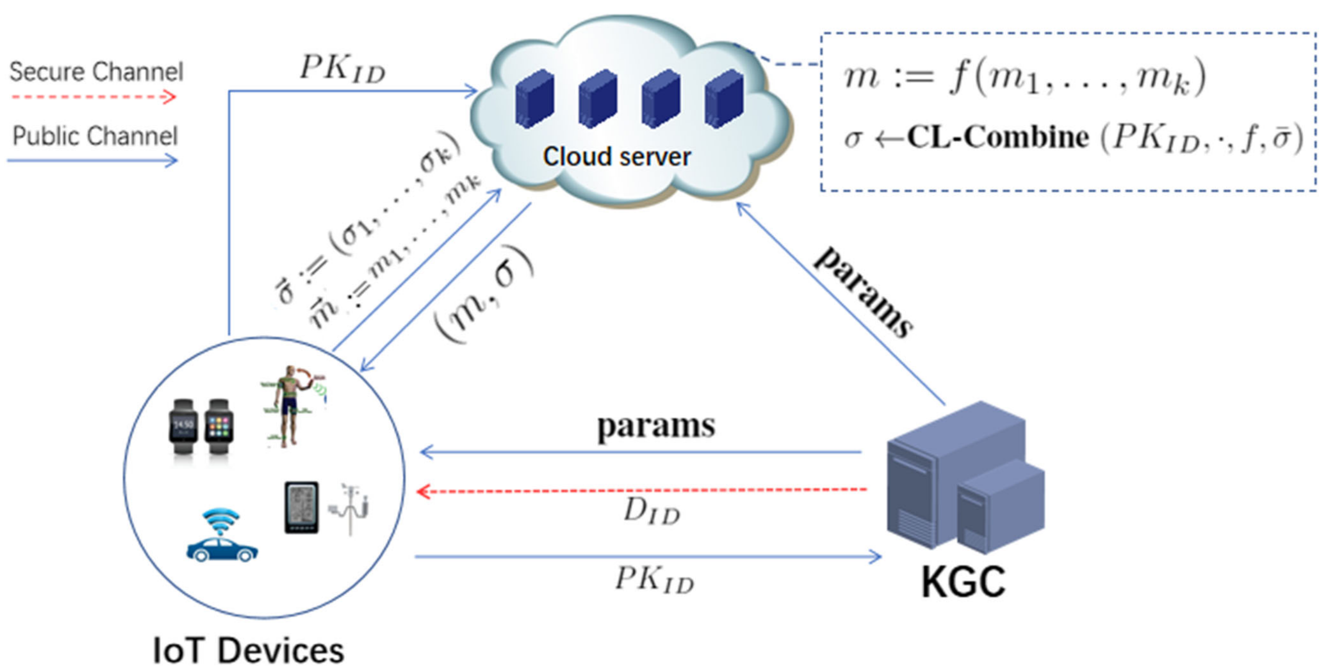

Fig. 4 System model of authentication computing using CL-LHS in an IoT environment

Table 2 A comparison of performance and security

Scheme	SkSize	SigSize	Verify	Type I,II	CL-PKC	ID-PKC	Model	Hardness
Scheme [24]	$\|\mathbb{G}_1\|+\|p\|$	$\|\mathbb{G}_1\|$	$2E$	–	No	Yes	ROM	co-CDH
Scheme $[26]_1$	$2\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|+\|p\|$	$5E$	–	No	Yes	ROM	CDH
Scheme $[26]_2$	$2\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|p\|$	$5E$	–	No	Yes	ROM	CDH
Scheme [23]	$\|\mathbb{G}_1\|+\|p\|$	$\|\mathbb{G}_1\|$	$3E$	–	No	No	ROM	CDH
Scheme $[19]_1$	$2\|p\|$	$2\|\mathbb{G}_1\|+\|p\|$	$4E$	–	No	No	ROM	CDH
Scheme $[27]_1$	$2\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|+\|p\|$	$5E$	–	No	Yes	ROM	CDH
Scheme $[27]_2$	$2\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|p\|$	$5E$	–	No	Yes	ROM	CDH
Our scheme	$\|\mathbb{G}_1\|+\|p\|$	$2\|\mathbb{G}_1\|$	$4E$	Yes	Yes	No	ROM	CDH

signatures are used for various calculations on the data, which can be completed through minimal interaction and communication, including the calculation results and corresponding short signatures sent from the server to the IoT devices.

6.2 Performance analysis

In this subsection, we mainly carry out the performance analysis. Table 2 compares the performance of our CL-LHS scheme with related schemes in the literature, e.g., [19, 23, 24, 26, 27] under random oracles in terms of private key size, signature length, verification cost, and security. Since references [26, 27] both used identity-based signature as module to design identity-based linearly homomorphic signature schemes. Therefore, in the efficiency analysis, we instantiate the module with the identity-based signature schemes proposed by reference [49, 50]. For convenience, the resulting schemes are denoted as schemes $[26]_1$, $[27]_1$ and schemes $[26]_2$, $[27]_2$, respectively. In addition, literature [19] used a general signature scheme as a module to design a linearly homomorphic signature scheme, so we use the BLS short signature to instantiate the module, and record the obtained scheme as $[19]_1$. The SkSize and SigSize columns show the size of the private key and signature, respectively. The **verify** column presents the computational costs of the algorithms **Verify**. Column **Type I, II** lists whether the scheme can resist public key replacement attacks and malicious-but-passive KGC attacks. The CL-PKC (ID-PKC) columns denote whether a scheme is based on a certificateless cryptosystem (identity-based cryptosystem). The **Hardness** columns denote the hardness assumption on which the security of the scheme depends. Let $|p|$, $|\mathbb{G}_1|$ and $|\mathbb{G}_2|$ represent the lengths of elements in $\mathbb{F}_p$, $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Note that the length of the private key affects the storage capacity of IoT devices, and the signature length affects the storage capacity and the communication capability of the IoT device. In addition, the computational cost of the algorithm **Verify** affects the computing power of both IoT devices and cloud servers. According to Table 2, the size of the private key of our scheme is shorter than that of the instantiated schemes $[26]_1$, $[26]_2$, $[27]_1$, $[27]_2$, and is the same as those of the schemes in [23] and [24]. The size of the signature of our scheme is shorter than those of the instantiated schemes of [19, 26, 27] and slightly larger than those of the schemes in [23, 24]. The verification algorithm of our scheme needs four bilinear pairs, which is roughly the same as is needed for the schemes in [23] and the instantiated schemes of [19, 26, 27]. However, our scheme addresses the issues of certificate

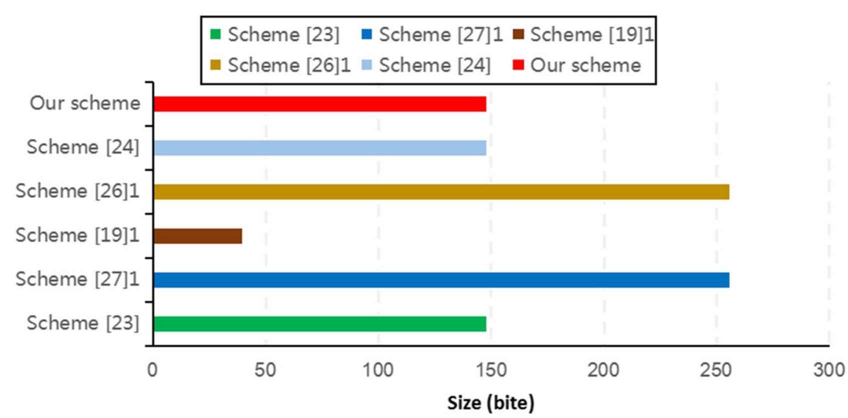

Fig. 5 A comparison of the private key size

management and key escrow and thus provides higher security.

In order to provide numerical results, we implement the proposed CL-LHS scheme and four related schemes, namely [19]$_1$, [23] and [26]$_1$, [27]$_1$, where [19]$_1$ and [23] are certificate-based schemes, while [26, 27] are certificateless schemes. Our implementation was run on a laptop with a 3.10-GHz Intel i5 CPU, 64 GB memory, and the Ubuntu Linux operating system. We chose the Type A curve in the PBC library [51]. The pairing operation is based on the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_p$. The security levels are chosen to be $|p| = 512$ bits.

Because IoT devices must secretly store their private keys, a small-sized private key is applicable in IoT devices with limited storage capacity. According to Fig. 5, the size of the private key in our CL-LHS scheme is 148 bits, which is the same as that in [23] and [24], and is 57.8% of that in [26]$_1$ and [27]$_1$.

Due to the limited battery power and communication bandwidth of IoT devices, signature size is the key factor affecting communication costs, so one of the tasks of our CL-LHS scheme is to reduce the communication overhead of devices in the IoT. As shown in Fig. 6, the signature size of our CL-LHS scheme is 256 bits, compared with [19]$_1$, [26]$_1$ and [27]$_1$, the signature size of our proposed scheme is reduced by 33.35%, 36.63% and 51.87%, respectively. Although the signature size of our CL-LHS scheme is larger than that of the schemes in [24] and [23], the literature [24] lacks the security proof for the identity-based homomorphic signature scheme proposed, and [23] is faced with a thorny certificate management issue. Hence, the proposed CL-LHS scheme has a lower communication overhead.

We compare the private key extraction cost of our scheme with the only three ID-LHS schemes based on bilinear pairing. As shown in Fig. 7, our extraction algorithm is faster than that of schemes [26]$_1$ and [27]$_1$ and slower than that of [24], but the scheme [24] lacks security proof. Figures 8 and 9 show the running time of signature generation and verification algorithms of the schemes. The x-axis is the dimension of the vector to be signed, and the y-axis is the time required by the corresponding algorithm. Overall, our CL-LHS scheme is less computationally efficient than but still comparable with the four related schemes, but it eliminates the problems of certificate management and key escrow, provides stronger security guarantees and better protects the privacy of users.

7 Conclusions

We constructed the first CL-LHS for network coding, which not only supports the authentication calculation of the linear function of the signed data to effectively mitigate pollution attacks in network coding but also solves the problems of certificate management and key escrow. To summarize, the scheme combines the properties of LHS and a certificateless signature. We proved that the scheme is secure against an adaptively chosen dataset attack under the random oracle model, even in the presence of type 1 and type 2 adversaries. Furthermore, compared to related schemes, our CL-LHS scheme has a smaller key size and a shorter signature length, and has comparable computation cost.

This work presents some interesting possibilities for future study. Since our scheme is unforgeable against adaptively chosen dataset attacks, it would be interesting to construct a CL-LHS scheme that is secure in a stronger security model that allows fully adaptive queries at the message level. As the CL-LHS scheme provides the credentials of the results calculated by a given function on a dataset, which are calculated by untrusted parties (e.g., the cloud), CL-LHS is very suitable for application in the cloud computing environment, such as in a smart grid, an e-voting system, or electronic health records. Proposing such applications is also the goal of our future work.

Acknowledgements The authors thank for the help of reviewers and editors. This work was supported by the Characteristic innovation project of general colleges and universities in Guangdong Province, Department of education of Guangdong Province (2020KTSCX126).

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

Appendix: Proof of Lemma 2

Proof Assume that $\mathcal{A}_{II}$ represents a malicious key generation center against the unforgeability of our CL-LHS scheme. We construct a simulator $\mathcal{C}$ that uses $\mathcal{A}_{II}$ as a subroutine to solve the CDH problem. According to the definition of Game 2, adversary $\mathcal{A}_{II}$ eventually outputs either a Type 1 forgery or a Type 2 forgery. $\mathcal{C}$ guesses the type of forgery to be output by $\mathcal{A}_{II}$ based on the result of flipping a coin randomly. Clearly, $\mathcal{C}$ guesses correctly with a probability of $\frac{1}{2}$.

Case 1 (Type 1 forgery:) In this case, $\mathcal{C}$ has guessed that $\mathcal{A}_{II}$ will output a Type 1 forgery. Given a random instance $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, g^a, g^b)$ of the CDH problem, $\mathcal{C}$ interacts with $\mathcal{A}_{II}$ as follows:

- **Setup:** $\mathcal{C}$ runs the setup, randomly chooses $s \in \mathbb{F}_p^*$ as the master key, and then initializes $\mathcal{A}_{II}$ with the master key s and params= $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, P_{pub} = g^s)$.
- **Queries:** $\mathcal{A}_{II}$ can issue queries to the following oracles, and $\mathcal{C}$ responds to $\mathcal{A}_{II}$ as follows:

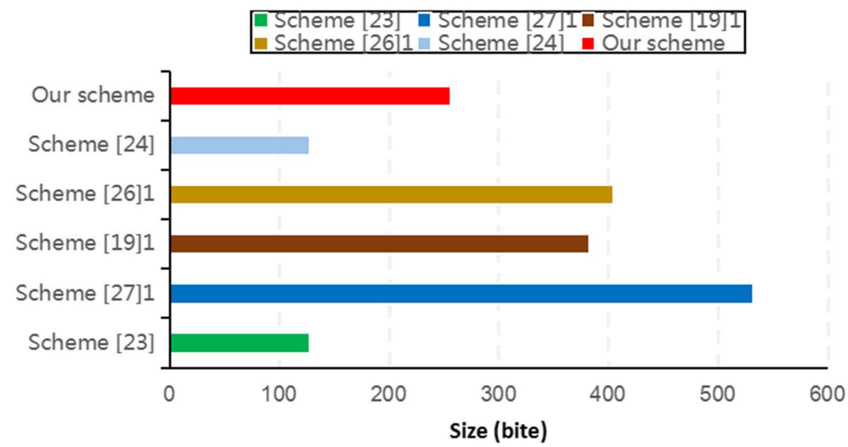

Fig. 6 A comparison of the communication cost

- H $Queries$: $\mathcal{C}$ maintains a list referred to as L_H. Suppose that $\mathcal{A_{II}}$ makes at most q_H queries. $\mathcal{C}$ randomly chooses $k \in \{1, 2, \cdots, q_H\}$ and guesses that the k-th identity ID_k submitted by $\mathcal{A_{II}}$ is the challenge identity. When $\mathcal{A_{II}}$ makes an H query on identity ID, $\mathcal{C}$ picks a random number $w_{ID} \in \mathbb{F}_p^*$, outputs $Q(ID) = H(ID) = g^{w_{ID}}$, and adds $< ID, H(ID), w_{ID} >$ to L_H.
- $Public\ Key\ Queries$: $\mathcal{C}$ maintains a list L_{PK} that is initially empty. When an identity ID is submitted for this query, $\mathcal{C}$ responds as follows:

(1) If $ID = ID_k$, $\mathcal{C}$ outputs the public key $PK_{ID} = g^a$ and adds $< ID_k, g^a, \perp >$ to L_{PK}.

(2) Otherwise, $\mathcal{C}$ randomly chooses $x_{ID} \in \mathbb{F}_p^*$ as the secret value. Then, $\mathcal{C}$ returns the public key $PK_{ID} = g^{x_{ID}}$ to $\mathcal{A_I}$ and saves $< ID, PK_{ID}, x_{ID} >$ in L_{PK}.

- $Private\ Key\ Extraction$: $\mathcal{C}$ maintains a list L_{SK} that is initially empty. Given an identity ID, $\mathcal{C}$ performs the following actions:

(1) If $ID \neq ID_k$, it recovers the tuple $< ID, H(ID), w_{ID} >$ from L_H and $< ID, PK_{ID}, x_{ID} >$ from L_{PK}. Then, $\mathcal{C}$ returns the secret key $SK_{ID} = ((g^{w_{ID}})^s, x_{ID})$ to $\mathcal{A_{II}}$ and adds $< ID, SK_{ID} >$ to L_{SK}.

(2) Otherwise, $\mathcal{C}$ aborts.

- H_1 $Queries$: Suppose $(ID, P_{pub}, \tau, U, i)$ is submitted to oracle $H_1(\cdot)$. $\mathcal{C}$ first scans $< (ID, P_{pub}, \tau, U, i), T_i, t_i >$ from the list L_{H_1} to check whether T_i has already been defined. If so, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ randomly chooses a number $t_i \in \mathbb{F}_p^*$, returns $T_i = g^{t_i}$ to $\mathcal{A_{II}}$ as the hash value of

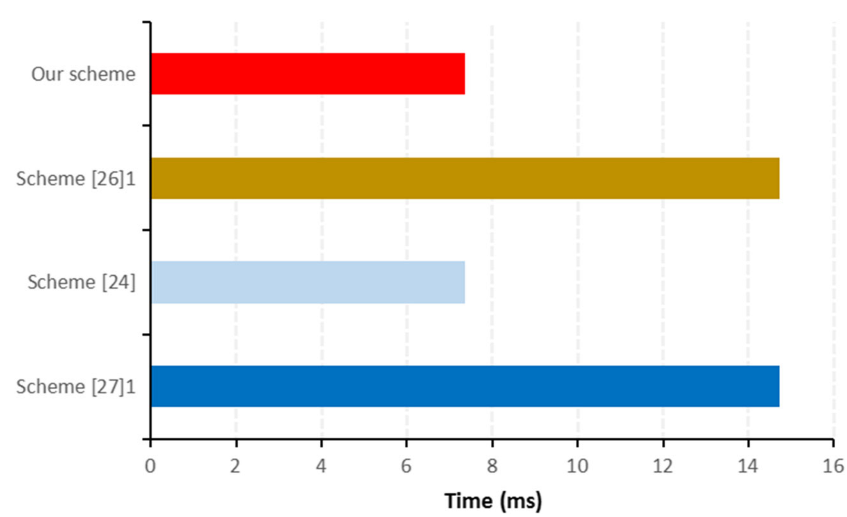

Fig. 7 A comparison of the *Extract* cost

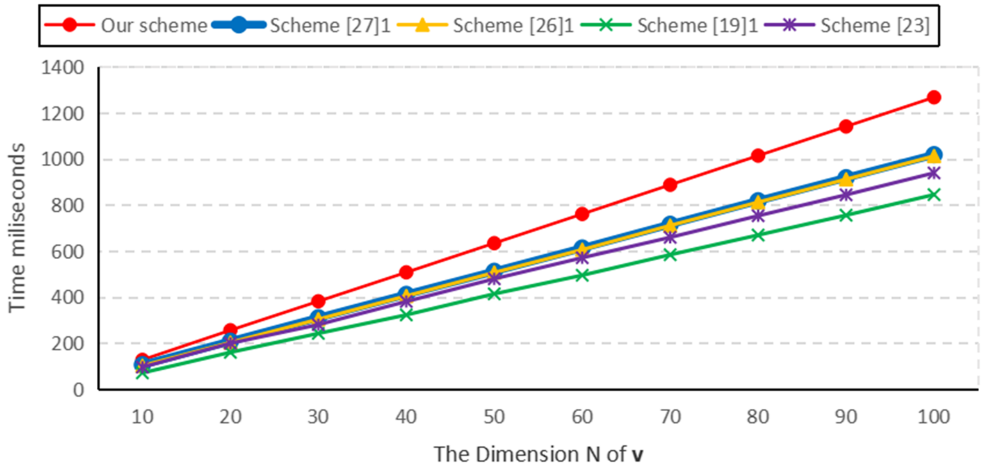

Fig. 8 A comparison of the signature generation cost

$H_1(ID, P_{pub}, \tau, U, i)$, and stores the value in the list L_{H_1}.

– H_2 *Queries* : Suppose (ID, PK_{ID}, τ, i) is submitted to oracle $H_2(\cdot)$. $\mathcal{C}$ first scans $< (ID, PK_{ID}, \tau, i), T_i', t_i' >$ from the list L_{H_2} to check whether T_i' has already been defined. If so, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ chooses a random number $t_i' \in \mathbb{F}_p^*$, returns $T_i' = g^{t_i'}$ to $\mathcal{A}_{\mathcal{II}}$ as the hash value of $H_2(ID, PK_{ID}, \tau, i)$, and stores the value in the list L_{H_2}.

– H_3 *Queries* : $\mathcal{C}$ maintains a list L_{H_3} containing tuples $< (ID, PK_{ID}), T, t >$. Upon receiving $\mathcal{A}_{\mathcal{II}}$'s query on (ID, PK_{ID}), if it already exists in L_{H_3}, $\mathcal{C}$ returns T. Otherwise, $\mathcal{C}$ chooses a random number $t \in \mathbb{F}_p^*$, returns $T = (g^b)^t$ to $\mathcal{A}_{\mathcal{II}}$ as the hash value of $H_3(ID, PK_{ID})$, and saves the value in the list L_{H_3}.

– *Signing Queries*: Given an identity ID and a vector space $V \subset \mathbb{F}_p^N$ described by augmented basis vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m \in \mathbb{F}_p^N$, where $\boldsymbol{v}_i = (v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0)$, if ID is the challenge identity (e.g., $ID = ID_k$), $\mathcal{C}$ preforms the following steps:

(1) Randomly choose an identifier $\tau \leftarrow \{0, 1\}^k$ and numbers $r, u_i \in \mathbb{F}_p^*(i \in [N])$, and set $U = PK_{ID}{}^r$.

(2) Define the hash values of $H_1(ID, P_{pub}, \tau, U, i)$ as $T_i = (\frac{g^{u_i}}{T})^{r^{-1}} \in \mathbb{G}_1$, where $T = H_3(ID, PK_{ID}) = (g^b)^t$. Abort if $H_1(ID, P_{pub}, \tau, U, i)$ has already been queried for some $i \in [N]$.

(3) Recover $T_i'(i \in [N])$ and Q_{ID} from L_{H_2} and L_H, respectively. If there are no such items, $\mathcal{C}$ makes queries on oracles $H_2(\cdot)$ and $H(\cdot)$.

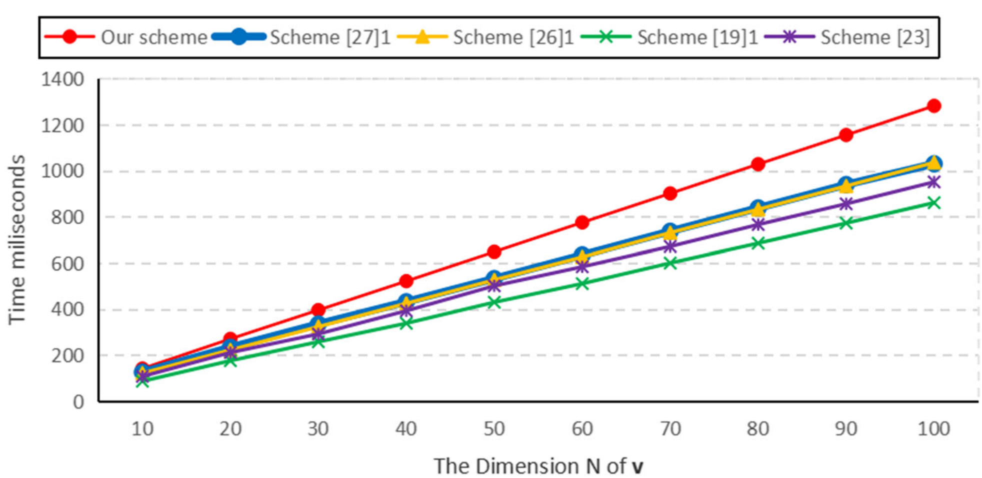

Fig. 9 A comparison of the signature verification cost

(4) Finally, compute

$$W_i = (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} u_j v_{ij} + \sum_{j\in[N]} t'_j v_{ij}}$$

Now, $\sigma_i = (U, W_i)(i \in [m])$ are returned to $\mathcal{A}_{\mathcal{II}}$. Each σ_i is a valid signature, since

$$\underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e\left(\prod_{j\in[N]} T_j^{v_{ij}}, U\right)}_{2} \cdot e\left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}, PK_{ID}\right) \tag{6}$$

$$= \underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e\left(\prod_{j\in[N]} (\frac{g^{u_j}}{Q_{ID}})^{r^{-1} v_{ij}}, P_{pub}^r\right)}_{2'} \cdot e\left(g^{\sum_{j\in[N]} t'_j v_{ij} + t\sum_{j\in[N]} v_{ij}}, PK_{ID}\right)$$

$$= \underbrace{e(Q_{ID}, P_{pub})^{\sum_{j\in[N]} v_{ij}}}_{1} \cdot \underbrace{e(Q_{ID}, P_{pub})^{-\sum_{j\in[N]} v_{ij}}}_{2'_{(1)}} \cdot \underbrace{e\left(g^{\sum_{j\in[N]} u_j v_{ij}}, P_{pub}\right)}_{2'_{(2)}} \tag{7}$$

$$\cdot e\left(g^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}, PK_{ID}\right) \tag{8}$$

$$= e\left((P_{pub})^{\sum_{j\in[N]} u_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij} + \sum_{j\in[N]} t v_{ij}}, g\right) \tag{9}$$

$$= e(W_i, g) \tag{10}$$

The derivation process of the core part of the above series of equations is shown as follows.[6]

Otherwise,[7] $\mathcal{C}$ randomly chooses an identifier $\tau \leftarrow \{0,1\}^k$ and a number $r \in \mathbb{F}_p^*$, sets $U = g^r$, and computes

$$W_i = (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t'_j v_{ij}} \cdot (g^b)^{t x_{ID} \sum_{j\in[N]} t v_{ij}}$$

The verification of the validity of the above signature is straightforward and is omitted here.

- **Output:** Eventually, $\mathcal{A}_{\mathcal{II}}$ outputs a tuple $(ID^*, PK_{ID^*}, \boldsymbol{y}, \tau^*, \sigma^*)$, where $\boldsymbol{v} = (v_1^*, \cdots, v_N^*)$, $\sigma^* = (U^*, W^*)$. If $ID^* \neq ID_k$, then $\mathcal{C}$ aborts. Otherwise, for each $i \in [N]$, it retrieves the items T_i^* from L_{H_1}, the items $T_i'^*$ from L_{H_2}, and the item T^* from L_{H_3}. Note that $T_i^* = g^{t_i^*}$, $T_i'^* = g^{t_i'^*}$, $T^* = (g^b)^{t^*}$. If $\mathcal{A}_{\mathcal{I}}$ successfully outputs Type 1 forgery signatures, the file identifier $\tau^* \neq \tau_i$ for all τ_i appears in signing queries, and the following equation holds:

$$e(W^*, g) = e(Q_{ID^*}, P_{pub})^{\sum_{i\in[N]} v_i^*} \cdot e(\prod_{i\in[N]} (T_i^*)^{v_i^*}, U^*) \cdot e(\prod_{i\in[N]} (T_i'^*)^{v_i^*} \cdot T^{\sum_{i\in[N]} v_i^*}, PK_{ID^*})$$

$$= e((Q_{ID^*})^{s\sum_{i\in[N]} v_i^*} \cdot (U^*)^{\sum_{i\in[N]} t_i^* v_i^*} \cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* v_i^*} \cdot (g^{ab})^{t^* \sum_{i\in[N]} v_i^*}, g)$$

Therefore, by the nondegenerate property, we have the solution of the CDH problem as follows:

$$\left(\frac{W^*}{(Q_{ID^*})^{s\sum_{i\in[N]} v_i^*} \cdot (U^*)^{\sum_{i\in[N]} t_i^* v_i^*} \cdot (PK_{ID^*})^{\sum_{i\in[N]} t_i'^* v_i^*}}\right)^{\frac{1}{t^* \sum_{i\in[N]} v_i^*}}.$$

Now, we evaluate $\mathcal{C}$'s probability of success.

We first analyze the probability of aborting in performing a signing query. The probability of the event that $\mathcal{C}$ responds to two distinct signature queries by choosing the same identifier τ is at most $\frac{q_s^2}{2^k}$, while the probability of the event that $\mathcal{A}_{\mathcal{II}}$ has already requested the value of $H_1(ID, P_{pub}, \tau, U, i)$ for some i is at most $\frac{q_{H_1} \cdot q_s}{2^k}$.

It is not hard to see that the probability of not aborting in key extraction queries is $(1 - \frac{1}{q_H})^{q_{sk}}$, and the probability of not aborting in the output stage is $\frac{1}{q_H}$, where q_s, q_H, q_{sk} are the number of signing queries, H is the number of hash queries and private key extraction is performed by $\mathcal{A}_{\mathcal{II}}$.

Thus, if $\mathcal{A}_{\mathcal{II}}$ has an advantage $Adv_{\mathcal{A}_{\mathcal{II}}}^{CL-LHS}(k)$ in forging a signature in Game 2, then $\mathcal{C}$ can solve the CDH problem with probability

$$\left(\frac{1}{2} Adv_{\mathcal{A}_{\mathcal{II}}}^{CL-LHS}(k) - \frac{q_s^2 + q_{H_1} \cdot q_s}{2^k}\right) \cdot \left(1 - \frac{1}{q_H}\right)^{q_{sk}} \cdot \frac{1}{q_H}$$

Case 2 (Type 2 forgery:) In this case, $\mathcal{C}$ has guessed that $\mathcal{A}_{\mathcal{II}}$ will output a Type 2 forgery. Given a CDH instance $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, g^a, g^b)$, the goal of $\mathcal{C}$ is to compute the value of g^{ab} by using $\mathcal{A}_{\mathcal{II}}$ as a subroutine. $\mathcal{C}$ interacts with $\mathcal{A}_{\mathcal{II}}$ as follows:

- **Setup:** $\mathcal{C}$ chooses a random number $s \in \mathbb{F}_p^*$ as the master key and sets $P_{pub} = g^s$ and params= $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, P_{pub} = g^s)$. It invokes $\mathcal{A}_{\mathcal{II}}$ on the input params and master key s.
- **Queries:** $\mathcal{C}$ simulates the oracle queries of $\mathcal{A}_{\mathcal{II}}$ as follows:
  - *H Queries* : $\mathcal{C}$ maintains a list L_H that is initially empty. Suppose that $\mathcal{A}_{\mathcal{II}}$ makes at most q_H queries. $\mathcal{C}$ randomly chooses $k \in \{1, 2, \cdots, q_H\}$ and guesses that the k-th identity ID_k submitted by $\mathcal{A}_{\mathcal{II}}$ is the challenge identity. When $\mathcal{A}_{\mathcal{II}}$ makes an H query on identity ID, $\mathcal{C}$ picks a random number $w_{ID} \in \mathbb{F}_p^*$, outputs $Q(ID) = H(ID) = g^{w_{ID}}$, and adds $< ID, H(ID), w_{ID} >$ to L_H.
  - *Public Key Queries* : $\mathcal{C}$ maintains a list referred to as L_{PK}. Given an identity ID, $\mathcal{C}$ responds as follows:

(1) If $ID = ID_k$, $\mathcal{C}$ outputs the public key $PK_{ID} = g^a$ and adds $< ID_k, g^a, \perp >$ to L_{PK}.

(2) Otherwise, $\mathcal{C}$ randomly chooses $x_{ID} \in \mathbb{F}_p^*$ as the secret value. Then, $\mathcal{C}$ returns the public

[6]Since we embed the hard problem in the term 1 of Eq. 6, that is, $T = (g^b)^t$, $PK_{ID} = g^a$. In order to successfully answer the signing queries, our idea is to eliminate item 1 by carefully setting the values of $T_j (j \in [N = n + m])$ and U while ensuring that the values of T_j and U are random ($T_j = (\frac{g^{u_i}}{T})^{r^{-1}}$, $U = PK_{ID}^r = (g^a)^r$). The item $2'$ in (11) is further arranged to obtain items $2'_1$ and $2'_2$ in (12). It is not difficult to find that item $2'_2$ can eliminate item 1, because item $2'_2$ and item 1 are inverses of each other in group $\mathbb{G}_2$.

[7]In this case, $ID \neq ID_k$, then $PK_{ID} = g^{x_{ID}}$, where x_{ID} is a known random number, so the required values generated in the process of various queries can be directly brought into the signature algorithm of the proposed scheme to obtain the signature.

key $PK_{ID} = g^{x_{ID}}$ to $\mathcal{A}_{\mathcal{II}}$ and saves $< ID, PK_{ID}, x_{ID} >$ in L_{PK}.

– *Private Key Extraction* : $\mathcal{C}$ maintains a list L_{SK} containing tuples $< ID, SK_{ID} >$. Given an identity ID, $\mathcal{C}$ performs the following actions:

(1) If $ID \neq ID_k$, it recovers the tuple $< ID, H(ID), w_{ID} >$ from L_H and $< ID, PK_{ID}, x_{ID} >$ from L_{PK}. Then, $\mathcal{C}$ returns the secret key $SK_{ID} = ((g^{w_{ID}})^s, x_{ID})$ to $\mathcal{A}_{\mathcal{II}}$ and adds $< ID, SK_{ID} >$ to L_{SK}.
(2) Otherwise, $\mathcal{C}$ aborts.

– *H_1 Queries*: Suppose $(ID, P_{pub}, \tau, U, i)$ is submitted to oracle $H_1(\cdot)$. $\mathcal{C}$ first scans for $< (ID, P_{pub}, \tau, U, i), T_i, t_i >$ in the list L_{H_1} to check whether T_i has already been defined. If so, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ chooses a random number $t_i \in \mathbb{F}_p^*$, returns $T_i = g^{t_i}$ to $\mathcal{A}_{\mathcal{I}}$ as the hash value of $H_1(ID, P_{pub}, \tau, U, i)$, and stores the value in the list L_{H_1}.
– *H_2 Queries* : Suppose (ID, PK_{ID}, τ, i) is submitted to oracle $H_2(\cdot)$. $\mathcal{C}$ first scans for $< (ID, PK_{ID}, \tau, i), T_i', t_i' >$ in the list L_{H_2} to check whether T_i' has already been defined. If so, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ selects $t_i' \in \mathbb{F}_p^*$ at random, returns $T_i' = g^{t_i'}$ to $\mathcal{A}_{\mathcal{I}}$ as the hash value of $H_2(ID, PK_{ID}, \tau, i)$, and stores the value in the list L_{H_2}.
– *H_3 Queries* : $\mathcal{C}$ maintains a list L_{H_3} containing tuples $< (ID, PK_{ID}), T, t >$. Taking (ID, PK_{ID}) as input, if it already exists in L_{H_3}, $\mathcal{C}$ returns T. Otherwise, $\mathcal{C}$ randomly chooses $t \in \mathbb{F}_p^*$, returns $H_3(ID, PK_{ID}) = g^t$ to $\mathcal{A}_{\mathcal{I}}$, and saves $< (ID, PK_{ID}), T, t >$ in L_{H_3}.
– *Signing Queries*: Given an identity ID and a vector space $V \subset \mathbb{F}_p^N$ described by augmented basis vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m \in \mathbb{F}_p^N$, where $\boldsymbol{v}_i = (v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0)$, $\mathcal{C}$ preforms the following steps:

(1) Randomly choose an identifier $\tau \leftarrow \{0,1\}^k$ and numbers $r, \alpha_1, \cdots, \alpha_n \in \mathbb{F}_p^*$, and set $U = g^r$.
(2) Set $n = N - m$, and for each $i \in [n]$, compute

$$T_i' = H_2(ID, PK_{ID}, \tau, i) = (g^b)^{\alpha_i}$$

For each $i \in [m]$, compute

$$\beta_i = -\sum_{j \in [n]} \alpha_j v_{ij}$$

$$T_{n+i}' = H_2(ID, PK_{ID}, \tau, n+i) = (g^b)^{\beta i}$$

and set $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m)$. Now observe that we constructed $\boldsymbol{\alpha}$ so that $\boldsymbol{\alpha} \in V^{\perp}$ (i.e., $\boldsymbol{\alpha} \cdot \boldsymbol{v} = 0$, for all $\boldsymbol{v} \in V = Span\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\}$).[8] $\mathcal{C}$ aborts if $H_2(ID, PK_{ID}, \tau, i)$ has already been queried for some $i \in [N]$.
(3) Recover T_i, T and SK_{ID} from L_{H_1}, L_{H_3} and L_{SK}, respectively. If there are no such items, $\mathcal{C}$ makes queries on the corresponding oracle.
(4) Compute

$$W_i = (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (PK_{ID})^{t\sum_{j\in[N]} v_{ij}}$$

(5) Return τ and $\sigma = (\sigma_1, \cdots, \sigma_m)$; here, $\sigma_i = (U, W_i)$.

Now, we show that the signatures σ_i are valid signatures, since

$$\begin{aligned} W_i &= (D_{ID})^{\sum_{j\in[N]} v_{ij}} \cdot \left(\prod_{j\in[N]} T_j^{v_{ij}}\right)^r \cdot \left(\prod_{j\in[N]} T_j'^{v_{ij}} \cdot T^{\sum_{j\in[N]} v_{ij}}\right)^{x_{ID}} \\ &= (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot \left(g^{\sum_{j\in[N]} t_j v_{ij}}\right)^r \cdot \left(\prod_{j\in[n]} (g^b)^{\alpha_j v_{ij}} \cdot \prod_{j\in[m]} (g^b)^{\beta_j v_{i,(n+j)}}\right)^a \cdot \left(g^{t\sum_{j\in[N]} v_{ij}}\right)^a \\ &= (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (g^{ab})^{\boldsymbol{\alpha}\cdot\boldsymbol{v}_i} \cdot (PK_{ID})^{\sum_{j\in[N]} t v_{ij}} \\ &= (Q_{ID})^{s\sum_{j\in[N]} v_{ij}} \cdot U^{\sum_{j\in[N]} t_j v_{ij}} \cdot (PK_{ID})^{\sum_{j\in[N]} t v_{ij}} \end{aligned}$$

Since we constructed $\boldsymbol{\alpha}$ such that $\boldsymbol{\alpha} \cdot \boldsymbol{v} = 0$ for all $\boldsymbol{v} \in V$, the signatures output by $\mathcal{C}$ in step (5) of signing queries are valid signatures.

- **Output:** Eventually, $\mathcal{A}_{\mathcal{II}}$ outputs ID^*, PK_{ID^*}, an identifier τ^*, a nonzero vector $\boldsymbol{y} = (y_1, \cdots, y_N)$ and signatures $\sigma_i^* = (U^*, W_i^*), i \in [m]$. If $ID^* \neq ID_k$, then $\mathcal{C}$ aborts.

If $\mathcal{A}_{\mathcal{I}}$ successfully outputs Type 2 forgery signatures σ^*, then τ^* has been used to answer a vector subspace V under a signature query, but $\boldsymbol{y} \notin V$; it is known that $T_i'^* = (g^b)^{a_i} (i \in [n])$ and $T_{n+i}'^* = (g^b)^{\beta i} (i \in [m])$. $\mathcal{C}$ recovers T_i^* from list L_{H_1}, T^* from list L_{H_3} and D_{ID^*} from list L_{SK}; then, the following equation holds:

$$\begin{aligned} & e\left(\prod_{i\in[m]} (W_i^*)^{y_{n+i}}, g\right) \\ =\ & e\left(Q_{ID^*}^{\sum_{i\in[N]} y_i}, P_{pub}\right) \cdot e\left(\prod_{i\in[N]} (T_i^*)^{y_i}, U^*\right) \cdot e\left(\prod_{i\in[N]} (T_i'^*)^{y_i} \cdot (T^*)^{\sum_{i\in[N]} y_i}, PK_{ID^*}\right) \\ =\ & e\left((Q_{ID})^{s\sum_{i\in[N]} y_i}, g\right) \cdot e\left((U^*)^{\sum_{i\in[N]} t_i^* y_i}, g\right) \cdot e\left((g^{ab})^{(\boldsymbol{\alpha}\cdot\boldsymbol{y})}, g\right) \cdot e\left((PK_{ID^*})^{t^*\sum_{i\in[N]} y_i}, g\right) \end{aligned}$$

[8]In detail, $\boldsymbol{\alpha} \cdot \boldsymbol{v}_i = (\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m) \cdot (v_{i1}, \cdots, v_{in}, \underbrace{0, \cdots, 1}_{i}, \cdots, 0) = \alpha_1 v_{i1} + \cdots + \alpha_n v_{in} + \beta_i = \alpha_1 v_{i1} + \cdots + \alpha_n v_{in} + (-\sum_{j=1}^{n} \alpha_j v_{ij}) = 0$. In particular, since we set $(T_1', \cdots, T_n', T_{n+1}', \cdots, T_{n+m}') = ((g^b)^{\alpha_1}, (g^b)^{\alpha_n}, (g^b)^{\beta_1}, \cdots, (g^b)^{\beta_m})$, we have $\prod_{j\in[N]} (T_j')^{v_{ij}} = \prod_{j\in[n]} (g^b)^{\alpha_j v_{ij}} \cdot \prod_{j\in[m]} (g^b)^{\beta_j v_{i,(n+j)}} = (g^b)^{\boldsymbol{\alpha}\cdot\boldsymbol{v}_i} = 1$

If $\boldsymbol{\alpha} \cdot \boldsymbol{y} \neq 0$, by the nondegenerate property, we obtain the value of g^{ab} as follows:

$$\left(\frac{\prod_{i \in [m]} (W_i^*)^{y_{n+i}}}{(Q_{ID})^{\sum_{i \in [N]} s y_i} \cdot (U^*)^{\sum_{i \in [N]} t_i^* y_i} \cdot (PK_{ID^*})^{\sum_{i \in [N]} t^* y_i}} \right)^{\frac{1}{(\boldsymbol{\alpha} \cdot \boldsymbol{y})}}$$

Now, we evaluate $\mathcal{C}$'s probability of success.

As before, obviously, the probability of $\mathcal{C}$ aborting in the signing query is at most $\frac{q_s^2 + q_{H_2} \cdot q_s}{2^k}$, the probability of not aborting in the output stage is $\frac{1}{q_H}$ and $\boldsymbol{\alpha} \cdot \boldsymbol{y} = 0$ with probability $\frac{1}{p}$, where q_s, q_H, q_{H_2} are the numbers of signing queries and H and H_2 are the numbers of hash queries made by $\mathcal{A_{II}}$.

Therefore, if $\mathcal{A_{II}}$ has an advantage $Adv_{\mathcal{A_{II}}}^{CL-LHS}(k)$ in forging a signature in Game 2, then $\mathcal{C}$ can solve the CDH problem with probability

$$\left(\frac{1}{2} Adv_{\mathcal{A_{II}}}^{CL-LHS}(k) - \frac{q_s^2 + q_{H_1} \cdot q_s}{2^k} \right) \cdot \left(1 - \frac{1}{p} \right) \cdot \frac{1}{q_H} \quad \square$$

References

1. Atzori L, Iera A, Morabito G. (2010) The internet of things: A survey. Comput Netw 54:2787–2805
2. Alaybeyi SB (2016) Pragmatic strategies to improve industrial IoT Security. tech rep Gartner
3. Ren H, Li H, Dai Y, Yang K, Lin X. (2018) Querying in internet of things with privacy preserving: Challenges, solutions and opportunities. IEEE Netw 32(6):144–151
4. Krohn M, Freedman M, Mazieres D (2004) On the-fly verification of rateless erasure codes for efficient content distribution. In: Proceedings of IEEE symposium on security and privacy Berkeley, CA, USA, pp 226–240
5. Li S-YR, Yeung R, Cai N. (2003) Linear network coding. IEEE Trans Inform Theory 49:371–381
6. Jin J-Q, Ho T, Viswanathan H (2006) Comparision of network coding and 1198: non-network coding schemes for multi-hop wireless networks. In: Proceedings of 2006 IEEE international symposium on information theory (ISIT 2006), Seattle, WA, USA, pp 197–201
7. Lun D, Medard M, Koetter R, Effros M (2005) Further results on coding for reliable communication over packet networks. In: Proceedings of international symposium on information theory (ISIT 2005), Adelaide, SA, Australia, pp 1848–1852
8. Boneh D, Freeman D, Katz J, Waters J (2009) Signing a linear subspace: Signature schemes for network coding. In: Proceedings of international workshop on public key cryptography (PKC 2009), vol 5443. Springer, Berlin, pp 68–87
9. Liu X, Huang J, Wu Y, Zong G. (2019) A privacy-preserving signature scheme for network coding. IEEE Access 7:109739–109750
10. Li T, Chen W, Tang Y, Yan H. (2018) A homomorphic network coding signature scheme for multiple sources and its application in IoT. Secur Commun Netw 2018:1–6
11. Agrawal S, MACs BonehD. (2009) Homomorphic MAC-based integrity for network coding. In: Proceedings of international conference on applied cryptography and network security (ACNS 2009), vol 5536. Springer, Berlin, pp 292–305
12. Chang J, Ji Y, Xu M, Xue R. (2019) General transformations from single-generation to multi-generation for homomorphic message authentication schemes in network coding. Future Gener Comp Sy 91:426–425
13. Esfahani A, Mantas G, Rodriguez J (2016) An efficient null space-based homomorphic MAC scheme against tag pollution attacks in RLNC. IEEE Commun Lett 20(5):918–921
14. Esfahani A, Yang D, Mantas G, Nascimento A, Rodriguez J. (2015) Dual-homomorphic message authentication code scheme for network codingenable wireless sensor networks. Int J Distrib Sensor Netw 11(7):1–10
15. Cheng C, Lee J, Jiang T, Takagi T. (2016) Security analysis and improvements on two homomorphic authentication schemes for network coding. IEEE Trans Inf Forensics Secur 15(5):993–1002
16. Shamir A (1984) Identity-based cryptosystems and signature schemes. In: Proceedings of the CRYPTO 1984, Santa Barbara, CA, USA, pp 47–53
17. Hu X, Zheng S, Gong J et al (2019) Enabling linearly homomorphic signatures in network coding-based named data networking. In: Proceedings of the 14th international conference on future internet technologies (CFI 2019). ACM, New York, pp 1–4
18. Liu X, Huang J, Zong G (2018) Public auditing for network coding based secure cloud storage. In: 2018 17th IEEE international conference on trust, security and privacy in computing and communications/ 12th ieee international conference on big data science and engineering (TrustCom/BigDataSE 2018) New York, NY, USA, pp 713–720
19. Schabhuser L, Buchmann J, Struck P (2017) A linearly homomorphic signature scheme from weaker assumption. In: IMA international conference on cryptography and coding (IMACC 2017), vol 10655. Springer, Cham, pp 261–279
20. Fiore D, Matrioska PE (2018) A compiler for multi-key homomorphic signatures. In: Proceedings of international conference on security and cryptography for networks (SCN 2018), vol 11035. Springer, Cham, pp 43–62
21. Lai RWF, Tai RKH, Wong HWH et al (2018) Multi-key homomorphic signatures unforgeable under insider corruption. In: Proceedings of international conference on the theory and application of cryptology and information security (ASIACRYPT 2018), Lecture notes in computer science, vol 11273. Springer, Cham, pp 465–492
22. Schabhüser L, Butin D, Buchmann J (2019) Context hiding multi-key linearly homomorphic authenticators. In: Proceedings of cryptographers' track at the RSA conference (CT-RSA 2019), vol 11405. Springer, Cham, pp 493–513
23. Lin Q, Li J, Huang Z, Chen W, Shen J. (2018) A short linearly homomorphic proxy signature scheme. IEEE Access 6:12966–12972
24. Zhang Y, Jiang Y, Li B, Zhang M (2017) An efficient identity-based homomorphic signature scheme for network coding. In: Proceedings of international conference on emerging internetworking, data and web technologies (EIDWT 2017), vol 6. Springer, Cham, pp 524–531
25. Sadrhaghighi S, Khorsandi S (2016) An identity-based digital signature scheme to detect pollution attacks in intra-session network coding. In: Proceedings of 13th international iranian society of cryptology conference on information security and cryptology (ISCISC 2016) Tehran, Iran, pp 7–12
26. Lin Q, Yan H, Huang Z, Chen W, Shen J, Tang Y. (2018) An ID-based linearly homomorphic signature scheme and its application in blockchain. IEEE Access 6:20632–20640
27. Chang J, Ma H, Zhang A, Xu M, Xue R. (2019) RKA security of identity-based homomorphic signature scheme. IEEE Access 7:50858–50868

28. Al-Riyami SS, Paterso KG (2003) Certificateless public key cryptography. In: Proceedings of 13th international Iranian society of cryptology conference on information security and international conference on the theory and application of cryptology and information security (ASIACRYPT 2003), vol 2894. Springer, Berlin, pp 452–473
29. Islam SH, Biswas G. (2014) Certificateless short sequential and broadcast multisignature schemes using elliptic curve bilinear pairings. J King Saud Univ Comp Info Sci 26(1):89–97
30. Wu L, Zhang Y, Ma MM et al (2019) Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical Internet of Things. Ann Telecommun 74:423–434
31. Wu T, Chen C, Wang K. (2019) Security analysis and enhancement of a certificateless searchable public key encryption scheme for IIot environments. IEEE Access 7:49232–49239
32. Yang XD, Pei XZ, Chen GL, Li T, Wang MD, Wang C. F. (2019) A strongly unforgeable certificateless signature scheme and its application in IOT environments. Sensors 19(12):1–27
33. Zhang Y, Deng H, Zheng D. et al (2019) Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial IoT. IEEE T Ind Inform 15(9):5099–5108
34. Karati A, Islam SH, Karuppiah M. et al (2019) Provably secure and lightweight certificateless signature scheme for IIoT environments. IEEE T Ind Inform 14(9):3701–3711
35. Yeh K-H, Su C, Choo KR, Chiu W. (2017) A novel certificateless signature scheme for smart objects in the internet-of-things. Sensors 17(5):1–17
36. Krohn MN, Freedman MJ, Mazi'eres D (2004) On-the fly verification of rateless erasure codes for efficient content distribution. In: Proceedings of IEEE symposium on security and privacy (SECPRI 2004), Berkeley, CA, USA, USA, pp 226–239
37. Attrapadung N, Libert B (2011) Homomorphic network coding signatures in the standard model. In: Proceedings of international workshop on public key cryptography (PKC 2011), vol 6571. Springer, Berlin, pp 17–34
38. Gennaro R, Katz J, Krawczyk H, Rabin T (2010) Secure network coding over the integers. In: Proceedings of international workshop on public key cryptography (PKC 2010), vol 6056. Springer, Berlin, pp 142–160
39. Boneh D, Freeman D (2011) Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Proceedings of international workshop on public key cryptography (PKC 2011), vol 6571. Springer, Berlin, pp 1–16
40. Boneh D, Freeman D (2011) Homomorphic signatures for polynomial functions. In: Proceedings of annual international conference on the theory and applications of cryptographic techniques (EUROCRYPT 2011), vol 6632. Springer, Berlin, pp 149–168
41. Gorbunov S, Vaikuntanathan V, Wichs D (2015). In: Proceedings of the forty-seventh annual ACM symposium on theory of computing (STOC New York, NY, USA, pp 469–477
42. Luo F, Wang F, Wang K, Chen K. (2019) A more efficient leveled strongly-unforgeable fully homomorphic signature scheme. Inf Sci 480:70–89
43. Shang F, Zhao X, Wang C, Liu J. (2015) Quantum homomorphic signature. Quantum Inf Process 14:393–410
44. Shang T, Pei Z, Chen R, Liu G. (2019) Quantum homomorphic signature with repeatable verification. CMC-Comput Mater Con 159(1):149–165
45. Li Z, Xu G, Chen L, Yang Y. (2019) Secure quantum network coding based on quantum homomorphicmessage authentication. Quantum Inf Process 18:1–21
46. Seo J, Emura K, Xagawa K, Yoneyama K. (2018) Accumulable optimistic fair exchange from verifiably encrypted homomorphic signatures. Int J Inf Secur 17:193–220
47. Fan X, Wu T, Zheng Q. (2019) HSE-Voting: A secure high-efficiency electronic voting scheme based on homomorphic signcryption. Future Gener Comp Sy 1–31. https://doi.org/10.1016/j.future.2019.10.016
48. Fiore JD, Mitrokotsa A, Nizzardo L et al (2016) Multi-key homomorphic authenticators. In: Proceedings of international conference on the theory and application of cryptology and information security (ASIACRYPT 2016), vol 10032. Springer, Berlin, pp 1–41
49. Choon JC, Cheon JH (2003) An identity-based signature from gap Diffie-Hellman groups. In: Proceedings of international workshop on public key cryptography (PKC 2003), vol 2567. Springer, Berlin, pp 18–30
50. Hess F (2002) Efficient identity based signature schemes based on pairings. In: Proceedings of International Workshop on Selected Areas in Cryptography (SAC 2002), vol 2595. Springer, Berlin, pp 1–15
51. Lynn B et al (2013) Pairing-based crytography library. https://crypto.stanford.edu/pbc/

Bin Wu was born in Baiyin City, Gansu Province. She received a B.E. degree in mathematics and applied mathematics in 2008 from Northwest Normal University, China and an M.S. degree in Homology Algebra in 2018 from Northwest Normal University. She is currently a PhD student at Northwest Normal University. The focus of her research has been on cryptography and information security.

Caifen Wang received her Ph.D. degree in cryptography from the School of Communication Engineering, Xidian University in 2003. She is a professor at Shenzhen Technology University. She has been selected as the director of the China Cryptography Society and a member of the Special Committee of Cryptography Algorithms. Her main research interests include cryptography and information security, in particular, applied cryptography and security in cloud computing.

Hailong Yao received an M.S. degree in communication and information systems from the School of Electronic & Information Engineering, Lanzhou Jiaotong University in 2013. He is currently a PhD student at Northwest Normal University. His research interests include cryptography and privacy security in distributed systems.