Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.026153

ARTICLE

A Linear Homomorphic Proxy Signature Scheme Based on Blockchain for Internet of Things

Caifen Wang[1,*] **and Bin Wu**[2]

[1]College of Big Data and Internet, Shenzhen Technology University, Shenzhen, 518118, China

[2]School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou, 730070, China

*Corresponding Author: Caifen Wang. Email: wangcfen@126.com

Received: 19 August 2022 Accepted: 14 October 2022

ABSTRACT

The mushroom growth of IoT has been accompanied by the generation of massive amounts of data. Subject to the limited storage and computing capabilities of most IoT devices, a growing number of institutions and organizations outsource their data computing tasks to cloud servers to obtain efficient and accurate computation while avoiding the cost of local data computing. One of the most important challenges facing outsourcing computing is how to ensure the correctness of computation results. Linearly homomorphic proxy signature (LHPS) is a desirable solution to ensure the reliability of outsourcing computing in the case of authorized signing right. Blockchain has the characteristics of tamper-proof and traceability, and is a new technology to solve data security. However, as far as we know, constructions of LHPS have been few and far between. In addition, the existing LHPS scheme does not focus on homomorphic unforgeability and does not use blockchain technology. Herein, we improve the security model of the LHPS scheme, and the usual existential forgery and homomorphic existential forgery of two types of adversaries are considered. Under the new model, we present a blockchain-based LHPS scheme. The security analysis shows that under the adaptive chosen message attack, the unforgeability of the proposed scheme can be reduced to the CDH hard assumption, while achieving the usual and homomorphic existential unforgeability. Moreover, compared with the previous LHPS scheme, the performance analysis shows that our scheme has the same key size and comparable computational overhead, but has higher security.

KEYWORDS

Homomorphic signature; proxy signature; security model; provable security; unforgeability

1 Introduction

In the past decade, the way of data collection and dissemination have inspired the rapid development of the Internet of Things (IoT) [1,2]. The IoT has opened up new avenues for technical support and business upgrades in the fields of industry, healthcare, transportation, military target tracking, smart homes and food traceability, among which mobile healthcare systems (MHSs) and the industrial Internet of Things (IIoT) are the most successful applications.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The IIoT continuously integrates all kinds of acquisition, sensors or controllers with sensing abilities, as well as mobile communication technology, into all aspects of industrial production, to improve production efficiency, reduce costs, and ultimately realize the conversion of traditional industry to smart industry [3]. MHSs provide services and applications [4,5], including mobile telemedicine and electronic monitoring systems based on wireless sensors. In this system, sensors (wearable or implanted) are connected to the bodies of remote patients to collect medical data, including body temperature, blood pressure, pH-value etc., and transmit these data through nodes to a medical server, which distributes the relevant data to professional medical personnel.

The rapid development of IIoT and MHSs, along with the generation of massive medical and industrial data, has led to increasing computing overhead and resource consumption, which makes traditional local computing model (most IoT devices have limited processing and computing power and are not economical to calculate) unable to meet the application requirements. Fortunately, due to the convenience and rapidity of cloud computing, many users migrate local data to cloud servers to meet the above challenge. Fig. 1 shows a typical cloud-based network architecture. Outsourcing data to cloud servers to obtain and accurate computation or analysis results have become preferential.

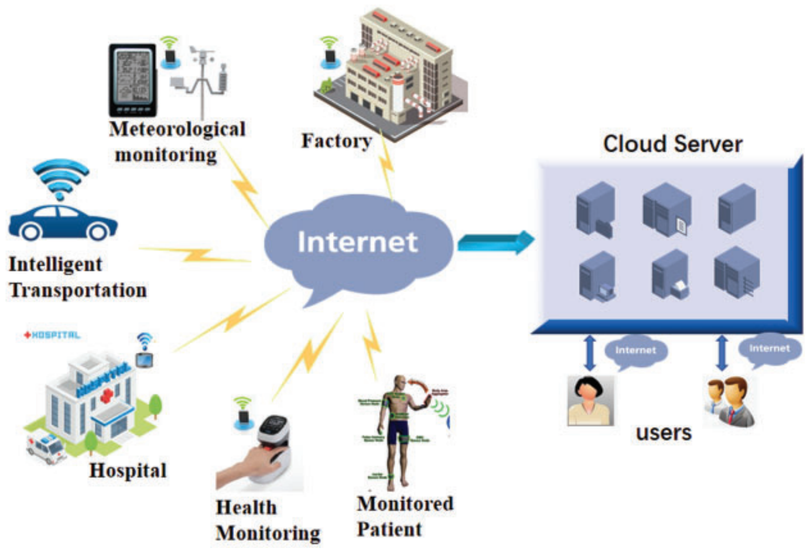

Figure 1: A typical architecture of cloud-based network

However, as an unsecure third party, cloud servers may return incomplete or wrong calculation results due to software and hardware errors and commercial interest inducement. Homomorphic signature (HS) provides a natural method for verifying outsourced computing, which can effectively solve the above problems. HS can enable untrusted servers to run calculations on outsourced data and generate a short signature to ensure the integrity and correctness of the calculation results. In recent years, HS has been extensively studied and developed [6–10].

Furthermore, consider such an application scenario: a hospital authorizes a sensor to sign data and communicate with the cloud server. The cloud server first confirms the authenticity of the sender (i.e., the sensor connected to the bodies of remote patients) and then sends the data calculation result and the derived signature to the hospital or research institution to verify its correctness. In an IIoT environment, an executive officer issues instructions to subordinates (such as the plant manager) to perform communication tasks. Thus, the data signature was often carried out by the subordinate on behalf of the superior organization. To realize this kind of authentication mechanism, it is necessary to authorize the signing right. In order to implement such kind of authentication mechanism, signature rights need to be authorized. Mambo et al. [11] realized this delegation authorization relationship with the concept of proxy signature (PS). In the PS scheme, the original signer (all represented by Alice) delegates its signing power to the proxy signer (all represented by Bob). Bob can generate valid proxy signatures in the name of Alice, and the verifier accepts the authorization protocol.

Naturally, to ensure the reliability of outsourced computing in the case of authorized signing rights, constructing a PS scheme with the homomorphism is interesting, which combines the delegation characteristics and homomorphism in the authentication method. In this type of scheme, Alice can authorize Bob, and Bob can create a proxy signature with homomorphic properties. However, as far as we know, constructions of linearly homomorphic proxy signature (LHPS) have been few and far between; in addition, the existing LHPS scheme [12] is only proven to be usually existential unforgeable (EUF) against adaptive chosen-message attacks (CMA). This security concept only guarantees that an adversary cannot forge a signature of any new message under the unqueried data identifier; it does not ensures that the adversary generates a valid signature for the data sets that have been queried for signatures (i.e., homomorphic existential forgery). In detail, for the security of a signature with homomorphic properties, there are two meanings of existential unforgeability, a verifiable forgery $(f, y', \boldsymbol{\sigma}')$ where f is an admissible function on messages $\boldsymbol{x}$ and $y' \neq y (y = f(\boldsymbol{x}))$ characterizes two facts: First, if $f(\boldsymbol{x}) = \pi_i(\boldsymbol{x}) = x_i$ is a special projection function, then $\boldsymbol{\sigma}'$ is a usual existential forgery (UEF), corresponding to the general concept of signature forgery. Another is that if f is a generally admissible function, then $\boldsymbol{\sigma}'$ is a homomorphic existential forgery (HEF), that is, forgery $\boldsymbol{\sigma}'$ authenticates y as $f(\boldsymbol{x})$, which is not the case. Homomorphic unforgeability is an important property of the LHPS scheme, which can prevent untrusted cloud servers from authenticating the wrong computing or analysis results and returning them to the receiver. Unfortunately, for the two types of adversaries, the existing homomorphic proxy signature security model does not consider the situation in which adversaries output homomorphic existential forgeries; thus, the proposed scheme does not satisfy homomorphic unforgeability in the sense of provable security.

To overcome this security flaw, we improve the security definition for LHPS and construct a new LHPS scheme for IoT environments, and makes the following main contributions in this paper:

1. The security model for LHPS is improved. For the two types of adversaries, considering the situation of an adversary's output of a homomorphic existential forgery, the types of forgeries are more comprehensive, so the model security standard is higher.
2. We construct a blockchain-based LHPS scheme and prove that this scheme is secure against existentially forgery (including the usual existential forgery and homomorphic existential forgery) on adaptive CMA based on the CDH assumption under two types of adversaries.
3. The performance of the new LHPS is analyzed in detail. The discussion shows that our scheme has the same key size and comparable computational overhead as Lin et al.'s LHPS scheme [12], but it has higher security. Therefore, it is feasible to deploy and implement our LHPS scheme in cloud-based IoT environments.

2 Related Works

The concept of HS scheme was first proposed by Goldwasser et al. [13] in 2000. Johnson et al. [14] first introduced the formal definition and overall framework of HS. Until 2009, Boneh et al. [15] proposed the first secure and practical LHS scheme, which can be regarded as a milestone of LHS scheme. Utilizing the k-SIS hardness assumption, reference [9] proposed the first scheme that can against quantum attacks. In order to be independent of certificate management, scholars proposed identity-based LHS schemes [16–19]. Although certificate management is simplified, key escrow issues are introduced. In response to this problem, researchers have successively proposed certificateless LHS [20,21]. Recently, multi-key HS has attracted attention [22–29]. For datasets involving inputs authenticated by different clients, multiple-key support is required, Chen et al. [27,28] successively proposed two multi-key LHS schemes, supporting three-layer and multi-layer routing networks respectively. Lai et al. [29] proposed an unforgeable multi-key HS under internal corruption based on adaptive zero-knowledge non-interactive knowledge demonstration. The public verifiability of HS makes them studied in other application scenarios. For example, the verifiable encryption HS scheme proposed by Seo et al. [30] has been successfully applied to cumulative optimistic fair exchange, and the homomorphic signcryption scheme proposed by Fan et al. [31] has been successfully applied to electronic voting, for voters, the use of homomorphic signcryption can complete the encryption and signature of votes in one step. In 2021, Li et al. [32] proposed a homomorphic signcryption scheme with verifiable public plaintext results, allowing the evaluation of arbitrary functions on signcrypted data, and allowing anyone to publicly test whether a given ciphertext is a signcrypted file for a message under a key.

Since Mambo et al. [11] proposed the concept of PS in 1996, many variants of PS have been proposed by other researchers successively, including proxy multi-signatur, multi-PS, proxy blind signature, certificate-based proxy signature and designated verifier proxy signature [33–38]. In 2003, Lee et al. [39] believed that proxy signature does not necessarily require a secure channel. In terms of the security model, Cao et al. [40] and Wang et al. [41] presented the model of a multi-proxy signature scheme. However, as Schuldt et al. [42] pointed out, the structure of the model is complex and incomplete. In 2012, Boldyreva et al. [43] improved its early security model by clearly defining the security attributes of proxy signatures and formalizing the definition of adversary behavior.

Blockchain is a new type of decentralized protocol that can securely store Bitcoin transactions or other data. The information cannot be forged or tampered with, and smart contracts can be automatically executed without the audit of any centralized organization [44,45]. Transactions can be digital currencies such as Bitcoin or digital assets such as debt, equity, copyright, etc. Blockchain technology solves the Byzantine general problem, greatly reduces the cost of trust and accounting in the real economy, and redefines the property right system in the Internet era. The introduction of blockchain technology in the proxy signature mechanism can realize the delegation of digital signature rights under the premise of safety and reliability, and support traceability and tamper-proof modification [46]. Additionally, blockchain-based proxy signatures enable anonymity of authentication while ensuring traceability of misconduct [47].

3 Architecture of Sign Delegation and Authentication Computing in a Cloud-Based IoT Environment

In the architecture of a cloud-based IoT environment using delegation and authentication computing, five entities are involved: CA, data owner, cloud server, end user and blockchain, as shown in Fig. 2. The specific functions are as follows (In this paper, Alice/Bob always represents the original/proxy signer):

- **CA:** It generates system parameters and generates certificates for users according to the identity and public key provided by each user.
- **Data owner:** It is composed of industry officials (e.g., system administrators, executive officers, etc.) and intelligent devices (embedded sensors). In our LHPS scheme, the superior is the Alice,

and the subordinate is the Bob. For smart machines, the deployment agency/supervisor acts as the Alice, and the sensor will act as the Bob. Therefore, the superior (or supervisor) delegates its signature right to the subordinate (or smart machine).

- **Cloud server:** The cloud server can use the homomorphic signature to perform various calculations on the authentication data, and then sends the calculation results and the derived signatures to the end user, which can be done with minimal interaction and communication.
- **End user:** The end user may be a hospital, a research institution, or an intelligent machine (depending on the scenario). It receives the calculation result and the corresponding signature and uses the public key of the two signers for verification.
- **Blockchain:** The blockchain mainly stores the warrant from the original signer to the proxy signer so that other entities can download and verify the validity of the warrant.

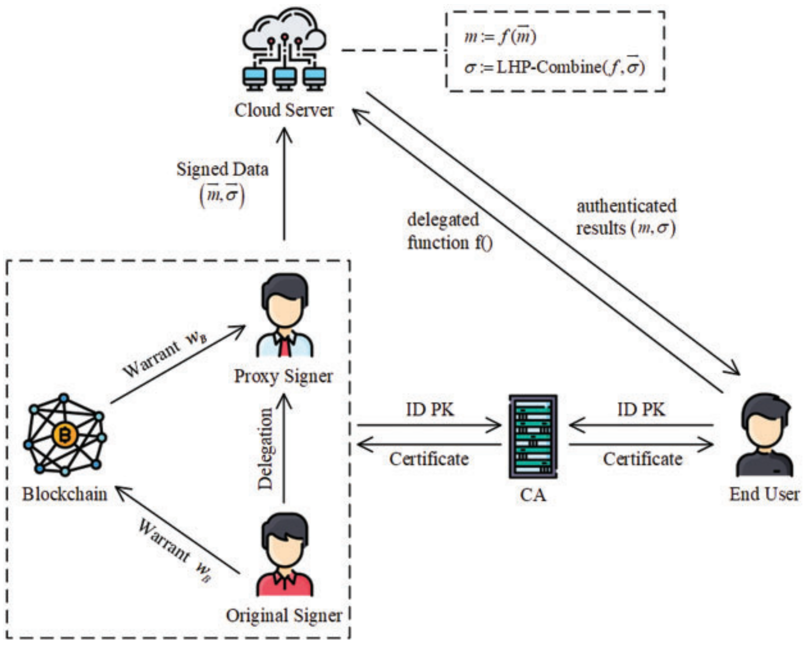

Figure 2: Architecture for IoT environment based on LHPS, blockchain and cloud computing

4 Definitions and Security Model

4.1 Linearly Homomorphic Proxy Signature

A LHPS scheme proposed in this paper includes seven polynomial-time algorithms (**Setup, UserKGen, Delegation, DeleVerify, LHP-Sign, LHP-Combine, LHP-Verify**):

- **Setup:** On input a security parameter λ and a integers l (the maximum data-size) as input, it outputs the system parameters *params*.
- **UserKGen:** On input *params*, it output a public/private key pair (pk_u, sk_u) for a user.
- **Delegation:** Given *params*, the private key sk_A of Alice, the warrant w_B of Bob (it records the identity information of the Alice and Bob, the type of authorization message, and the validity period of the proxy signer), it output delegation information S_{w_B}.

- **DeleVerify:** For the input *params*, public key of Alice, delegation and warrant, it outputs 0 or 1 according to reject or accept.
- **LHP-Sign:** Bob greats a proxy signature $\boldsymbol{\sigma}$ on a message vector $\boldsymbol{v}$, after using *params*, the private key sk_B of Bob, a warrant w_B, delegation information S_{w_B} and a file identifier $\tau \in \{0,1\}^{\lambda}$.
- **LHP-Combine:** Takes a file identifier τ, public keys of both signer, a warrant w_B, and a set of tuples $\{(c_i, \boldsymbol{\sigma}_i)\}_{i=1}^{l}$, where $c_i \in \mathbb{F}_q$, $\boldsymbol{\sigma}_i \leftarrow$ **LHP-Sign** $(params, \tau, sk_B, S_{w_B}, \boldsymbol{v}_i)$, and outputs a signature $\boldsymbol{\sigma}$ on $\boldsymbol{v} = \sum_{i \in [l]} c_i \boldsymbol{v}_i$.
- **LHP-Verify:** Takes as input public keys of both signer, w_B, a file identifier τ, and message/signature pair, outputs either 1 or 0.

Correctness. The requirement is that for any output (pk_A, sk_A), (pk_B, sk_B) of the algorithm **UserKGen**, the following conditions are true:

(1) For any $\tau \in \{0,1\}^{\lambda}$ and message vector $\boldsymbol{v}$, if $\boldsymbol{\sigma} \leftarrow$ **LHP-Sign** $(sk_B, w_B, S_{w_B}, \tau, \boldsymbol{v})$, then **LHP-Verify** $(\tau, pk_A, pk_B, w_B, \boldsymbol{v}, \sigma) = 1$.

(2) For all τ and sets of tuples $\{(c_i, \sigma_i, \boldsymbol{v}_i)\}_{i=1}^{l}$, if **LHP-Verify** $(\tau, pk_A, pk_B, w_B, \boldsymbol{v}_i, \sigma_i) = 1$, then **LHP-Verify** $(\tau, pk_A, pk_B, w_B, \sum_{i=1}^{l} c_i \boldsymbol{v}_i,$ **LHP-Combine** $\{\tau, pk_A, pk_B, w_B, (c_i, \sigma_i)\}_{i=1}^{l}) = 1$.

4.2 Security Models

We consider two type of adversaries, and denoted by $\mathcal{A}_{\mathcal{I}}$, $\mathcal{A}_{\mathcal{II}}$, respectively.

Type-I adversary ($\mathcal{A}_{\mathcal{I}}$): $\mathcal{A}_{\mathcal{I}}$ consists of system parameters, Alice's public-private key pair, and Bob's public key. In fact, $\mathcal{A}_{\mathcal{I}}$ is a malicious original signer.

Type II adversary ($\mathcal{A}_{\mathcal{II}}$): $\mathcal{A}_{\mathcal{II}}$ consists of system parameters, Bob's public-private key pair, and Alice's public key. In fact, $\mathcal{A}_{\mathcal{II}}$ is a malicious proxy signer.

We use the following game between challenger $\mathcal{C}$ and $\mathcal{A}_{\mathcal{I}}$, $\mathcal{A}_{\mathcal{II}}$ to describe the security of the LHPS scheme.

Game 1. (Security against adversary $\mathcal{A}_{\mathcal{I}}$) $\mathcal{A}_{\mathcal{I}}$ attempts to outputs a forged proxy signature while having no private key of the Bob. The following formalized security models are designed for $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{C}$.

- **Setup:** $\mathcal{C}$ performs **Setup** and **UserKGen** oracles, and obtain the system parameters *params*, the key pair (pk_A, sk_A) of Alice as well as key pair (pk_{B^*}, sk_{B^*}) of the targeted proxy signer. $\mathcal{C}$ then gives *params* and (pk_A, sk_A, pk_{B^*}) to $\mathcal{A}_{\mathcal{I}}$.
- **Queries:** Adversary $\mathcal{A}_{\mathcal{I}}$ can performs the following polynomial number of oracle queries.
  - *Key Registration Queries*: Given a key pair (pk_i, sk_i), $\mathcal{C}$ first checks whether (pk_i, sk_i) is valid. If so, $\mathcal{C}$ stores it in a list. Otherwise, $\mathcal{C}$ rejects.
  - *Signing Queries*: Given a tuple $(\tau, pk_A, pk_{B^*}, w_{B^*}, \boldsymbol{v})$, $\mathcal{C}$ outputs a signature σ on $\boldsymbol{v}$.
- **Output:** $\mathcal{A}_{\mathcal{I}}$ outputs a forgery tuple $(pk_A, pk_{B^*}, w_{B^*}, \tau^*, \boldsymbol{v}^*, \sigma^*)$. The adversary wins if **LHP-Verify** $(\tau^*, pk_A, pk_{B^*}, w_{B^*}, \boldsymbol{v}^*, \sigma^*) = 1$, the message vector $\boldsymbol{v}^*$ does not appear in signing queries, and one of the following conditions is satisfied:

  (1) For any τ_i appearing in the signing queries, $\tau^* \neq \tau_i$ holds (**usual existential-forgery–Type 1 forgery**).

(2) For a τ_i appearing in the signing queries, there is $\tau^* = \tau_i$, but $\boldsymbol{v}^* \notin V_i$, where V_i represents the subspace spanned by the vectors $\{\boldsymbol{v}_i\}_{i\in[l]}$ that have been queried by the identifier τ_i (**homomorphic existential-forgery–Type 2 forgery**).

Game 2. (Security against adversary $\mathcal{A}_{\mathcal{II}}$) $\mathcal{A}_{\mathcal{II}}$ attempts to output a forged proxy signature while having no private key of Alice. Now, the following formal model with respect to $\mathcal{A}_{\mathcal{II}}$ and challenger $\mathcal{C}$ is designed.

- **Setup:** $\mathcal{C}$ performs **Setup** and **UserKGen** oracles, and obtains the parameters *params*, the public/private key pair (pk_A, sk_A) of Alice. $\mathcal{C}$ then gives *params* and pk_A to $\mathcal{A}_{\mathcal{II}}$.
- **Queries:** Adversary $\mathcal{A}_{\mathcal{II}}$ can performs the following polynomial number of oracle queries.
  – *Key Registration Queries*: Given a key pair (pk_i, sk_i), $\mathcal{C}$ first checks whether (pk_i, sk_i) is valid. If so, $\mathcal{C}$ stores it in a list. Otherwise, $\mathcal{C}$ rejects.
  – *Delegate Queries*: Given public key pk_A and registered pk_i, $\mathcal{C}$ returns a warrant w_i and a delegation S_{w_i}.
  – *Signing Queries*: Given public key pk_A and registered public key pk_i, w_i, τ, and a vector $\boldsymbol{v}$, $\mathcal{C}$ returns a signature σ on the message $\boldsymbol{v}$.
- **Output:** $\mathcal{A}_{\mathcal{II}}$ outputs a forgery tuple $(\tau^*, pk_A, pk_{i^*}, w_{Bi^*}, \boldsymbol{v}^*, \sigma^*)$. $\mathcal{A}_{\mathcal{II}}$ wins if **LHP-Verify** $(\tau^*, pk_A, pk_i^*, w_i^*, \boldsymbol{v}^*, \sigma^*) = 1$, and public key pk_{i^*} and message vector $\boldsymbol{v}^*$ do not appear in delegate queries and signing queries, respectively, and one of the following conditions is satisfied:
  (1) For any τ_i appearing in the signing queries, $\tau^* \neq \tau_i$ holds (**usual existential-forgery–Type 1 forgery**).
  (2) For a τ_i appearing in the signing queries, there is $\tau^* = \tau_i$, but $\boldsymbol{v}^* \notin V_i$, where V_i represents the subspace spanned by the vectors $\{\boldsymbol{v}_i\}_{i\in[l]}$ that have been queried by the identifier τ_i (**homomorphic existential-forgery–Type 2 forgery**).

The advantage of adversary $\mathcal{A}_{\mathcal{I}}$ ($\mathcal{A}_{\mathcal{II}}$) is the probability of winning Game 1 (Game 2), which is recorded as $Adv_{\mathcal{A}_{\mathcal{I}}}^{LHPS}(\lambda)$ $(Adv_{\mathcal{A}_{\mathcal{II}}}^{LHPS}(\lambda))$.

Definition 4.1. Our proposed LHPS scheme is said to be unforgeable against the adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$ in the above games, if $Adv_{\mathcal{A}_i}^{LHPS}(\lambda)(i = I, II)$ is negligible.

5 Proposed LHPS Scheme

Our proposed LHPS scheme is described in detail as follows:

- **Setup:** Takes a security parameter λ and two positive integers l, n as inputs, CA generates *params* as follows:
  (1) Choose two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of the same prime order q, with g as the generator of $\mathbb{G}_1$, and select a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
  (2) Selects different hash functions H, H_1, H_2, each of which maps $\{0, 1\}^*$ to $\mathbb{G}_1$.
- **UserKGen:** Each user chooses $x_u \in \mathbb{Z}_q^*$ at random as his private key and computes $pk_u = g^{x_u}$ as public key. Then, the public/private key pair of original signer Alice and proxy signer Bob are $(pk_A = g^{x_A}, x_A)$ and $(pk_B = g^{x_B}, x_B)$, respectively. Subsequently, each user provides its identity and public key to the CA to obtain the corresponding certificate.
- **Delegation:** Alice generates a warrant w_B related to Bob, and computes delegation $S_{w_B} = H(w_B)^{x_A}$, then returns S_{w_B} to Bob. After that, Alice uploads w_B to the blockchain so that other users can query and verify the validity of w_B.

- **DeleVerify:** Bob checks $e(S_{w_B}, g) = e(H(w_B), pk_A)$ and accepts delegation. Then, Bob sets proxy signing key as $S_p = (S_{w_B}, x_B)$.
- **LHP-Sign:** Suppose that this algorithm has stored an initially empty list L containing information about all identifiers τ that have been used. Given a tuple $(S_p, w_B, \tau \in \{0,1\}^\lambda, \boldsymbol{v})$, here the message vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, it performs the following steps:
  (1) Retrieve the relevant (U, r) from L, if τ appears in the list L.
  (2) Otherwise, it chooses a random number $r \in \mathbb{Z}_q^*$, lets $U = g^r$, then stores (U, r) into L.
  (3) Calculate hash values $Q_i = H_1(pk_A, w_B, \tau, U, i), i \in [n]$, $Q = H_2(pk_B, w_B)$.
  (4) Choose a random number $s \in \mathbb{Z}_q^*$, and computes
  $$W = \left(S_{w_B}\right)^{\sum_{i\in[n]} v_i} \left(H(w_B)^s \cdot \prod_{i\in[n]} Q_i^{v_i}\right)^r \left(Q^{\sum_{i\in[n]} v_i}\right)^{x_B}.$$
  (5) Output $\sigma = (U, W, s)$ as signature.
- **LHP-Combine:** On input pk_A, pk_B, τ, w_B, and a set of tuple $\{(c_i, \sigma_i)\}_{i=1}^l$ with $c_i \in \mathbb{Z}_q$, where $\sigma_i = (U, W_i, s_i)$ (note that the data signatures under the same identifier have the same U), this algorithm computes $W = \prod_{i\in[l]} W_i^{c_i}$, $s = \sum_{i\in[l]} c_i s_i$. Then, it outputs (U, W, s).
- **LHP-Verify:** Given a tuple $(\tau, pk_A, pk_B, w_B, \boldsymbol{v}, \sigma)$, where vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, $\sigma = (U, W, s)$, the algorithm considers the proxy signature to be valid by verifying the following equation:
$$e(W, g) = e\left(\left(H(w_B)^{\sum_{i\in[n]} v_i}, pk_A\right) \cdot e\left(H(w_B)^s \cdot \prod_{i\in[n]} Q_i^{v_i}, U\right) \cdot e\left(Q^{\sum_{i\in[n]} v_i}, pk_B\right)\right).$$

Correctness

(1) Given a tuple $(\tau, pk_A, pk_B, \boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n, \sigma)$, if $\sigma \leftarrow$ **LHP-Sign** $(S_p, w_B, \tau, \boldsymbol{v})$, then the correctness of proxy signature verification can be obtained by the following derivation:
$$e(W, g) = e\left(\left(S_{w_B}\right)^{\sum_{i\in[n]} v_i}, g\right) \cdot e\left(\left(H(w_B)^s \cdot \prod_{i\in[n]} Q_i^{v_i}\right)^r, g\right) \cdot e\left(\left(Q^{\sum_{i\in[n]} v_i}\right)^{x_B}, g\right)$$
$$= e\left(\left(H(w_B)^{\sum_{i\in[n]} v_i}, pk_A\right) \cdot e\left(\left(H(w_B)^s \prod_{i\in[n]} Q_i^{v_i}, U\right) \cdot e\left(Q^{\sum_{i\in[n]} v_i}, pk_B\right)\right.\right).$$

(2) Given a tuple $(\tau, w_B, \{(c_i, \sigma_i, \boldsymbol{v}_i)\}_{i=1}^l)$, where $\sigma_i = (U, W_i, s_i)$ and $\boldsymbol{v}_i = (v_{i1}, \ldots, v_{in})$, if $\sigma_i \leftarrow$ **LHP-Sign** $(S_p, w_B, \tau, \boldsymbol{v}_i)$, we have to prove that $\sigma = \left(U, W = \prod_{i\in[l]} W_i^{c_i}, s = \sum_{i\in[l]} c_i s_i\right)$ is a signature on

$\boldsymbol{y} = \sum_{i \in [l]} c_i \boldsymbol{v}_i = (y_1, \ldots, y_n)$. Since for each $i \in [l]$, σ_i is the correct signature of $\boldsymbol{v}_i$, we have

$$e(W_i, g) = e\left(H(w_B)^{\sum_{j\in[n]} v_{ij}}, pk_A\right) \cdot e\left(H(w_B)^{s_i} \prod_{j\in[n]} Q_j^{v_{ij}}, U\right) \cdot e\left(Q^{\sum_{j\in[n]} v_{ij}}, pk_B\right),$$

and further we have

$$\begin{aligned} e(W, g) &= \prod_{i\in[l]} e(W_i, g)^{c_i} \\ &= e\left(H(w_B)^{\sum_{i\in[l]}\sum_{j\in[n]} c_i v_{ij}}, pk_A\right) \cdot e\left(H(w_B)^{\sum_{i\in[l]} c_i s_i} \prod_{j\in[n]} Q_j^{\sum_{i\in[l]} c_i v_{ij}}, U\right) \cdot e\left(Q^{\sum_{i\in[l]}\sum_{j\in[n]} c_i v_{ij}}, pk_B\right) \\ &= e\left(H(w_B)^{\sum_{j\in[n]} y_j}, pk_A\right) \cdot e\left(H(w_B)^{s} \cdot \prod_{j\in[n]} Q_j^{y_j}, U\right) \cdot e\left(Q^{\sum_{j\in[n]} y_j}, pk_B\right). \end{aligned}$$

Therefore, for the **LHP-Combine** algorithm, the **LHP-Verify** algorithm is correct.

6 Security Analysis

This section presents the security analysis of our LHPS scheme. In this section, (g, g^a, g^b) always represents a random instance of the CDH problem.

Theorem 6.1. If there is an adversary $\mathcal{A}_{\mathcal{I}}$ that can forge a valid message/signature pair with an advantage ε within time t, then there is an algorithm $\mathcal{C}$ that can solve the CDH problem with probability ε' within time t'. Among them, q_H refers to the number of querying H oracle, $q_{H_i} (i = 1, 2)$ refers to the number of querying H_i oracles, q_S refers to the number of querying signing oracle.

Proof. For any PPT adversary $\mathcal{A}_{\mathcal{I}}$, we construct a simulator $\mathcal{C}$, which can call $\mathcal{A}_{\mathcal{I}}$ to solve the CDH problem. Since $\mathcal{A}_{\mathcal{I}}$ finally outputs one of the two forgery types, it is obvious that the probability of $\mathcal{C}$ guessing the forgery type outputed eventually by $\mathcal{A}_{\mathcal{I}}$ is $\frac{1}{2}$.

Case 1 (usual existential-forgery.) $\mathcal{C}$ guesses that adversary $\mathcal{A}_{\mathcal{I}}$ will output a Type 1 forgery. $\mathcal{C}$ is invoked on (g, g^a, g^b), and $\mathcal{C}$' goal is to compute g^{ab}. $\mathcal{C}$ interacts with $\mathcal{A}_{\mathcal{I}}$ in this way.

- **Setup:** $\mathcal{C}$ randomly selects $x_A \in \mathbb{Z}_q^*$, computes Alice's public key as $pk_A = g^{x_A}$, and sets Bob's public key as $pk_{B^*} = g^a$ (note that in the proof of Theorem 1, Bob represents the target proxy signer). $\mathcal{C}$ sets the system parameter $params = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, H_1, H_2, q)$. Then, $\mathcal{C}$ sends $params$ and (x_A, pk_A, pk_{B^*}) to $\mathcal{A}_{\mathcal{I}}$.
- **Queries:** $\mathcal{A}_{\mathcal{I}}$ can issue queries to the following oracles simulated by $\mathcal{C}$. $\mathcal{C}$ maintains an initially empty list for each type of query, recording all responses to $\mathcal{A}_{\mathcal{I}}$.
  - *Key Registration Queries*: When $\mathcal{A}_{\mathcal{I}}$ requests to register a new user i by outputting a key pair (pk_i, x_i), $\mathcal{C}$ verifies if it is valid, and then adds (pk_i, x_i) to the list L_K.
  - *H Queries*: On a requested w_i, $\mathcal{C}$ selects a random $z_i \in \mathbb{Z}_q^*$ and sets $H(w_i) = g^{z_i}$. Then, $\mathcal{C}$ updates list $L_H \leftarrow L_H \cup < w_i, g^{z_i}, z_i >$ and responds g^{z_i} to $\mathcal{A}_{\mathcal{I}}$.

- H_1 *Queries*: For a requested (Pk_A, w_B, τ, U, i), $\mathcal{C}$ first searches L_{H_1} and if an required entry is found, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ selects a random $t_i \in \mathbb{Z}_q^*$ and computes $Q_i = g^{t_i}$, then updates $L_{H_1} \leftarrow L_{H_1} \cup < (pk_A, w_B, \tau, U, i), Q_i, t_i >$ and sends Q_i to $\mathcal{A}_{\mathcal{I}}$ as response.
- H_2 *Queries*: On a requested (w_B, pk_B), $\mathcal{C}$ checks the list L_{H_2} to determine whether there is a corresponding hash value, and if so, $\mathcal{C}$ returns it to $\mathcal{A}_{\mathcal{I}}$; otherwise, $\mathcal{C}$ chooses a random number $t \in \mathbb{Z}_q^*$ and sets $Q = (g^b)^t$, then updates list $L_{H_2} \leftarrow L_{H_2} \cup < (w_B, pk_B), Q, t >$ and sends Q to $\mathcal{A}_{\mathcal{I}}$ as response.
- *Signing Queries*: Given τ, pk_A, pk_{B^*}, a warrant w_{B^*}, and a vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, $\mathcal{C}$ generates the signature as follows:
  (1) Randomly choose numbers $s, r, u_i \in \mathbb{Z}_q^* (i \in [n])$, and set $U = pk_{B^*}^r = (g^a)^r$.
  (2) Define the hash values of $H_1(pk_A, w_{B^*}, \tau, U, i)$ as $Q_i = \left(\frac{g^{u_i}}{Q}\right)^{r^{-1}} \in \mathbb{G}_1$, where $Q = H_2(w_{B^*}, pk_{B^*}) = (g^b)^{t^*}$. If for some $i \in [n]$, $(pk_A, w_{B^*}, \tau, U, i)$ has already been queried in the H_1 *queries* phase, then $\mathcal{C}$ terminates and gives up.
  (3) Recover $H(w_{B^*}) = g^{z_{B^*}}$ from L_H.
  (4) Finally, $\mathcal{C}$ computes

$$W = (H(w_{B^*}))^{x_A \sum_{i \in [n]} v_i} \cdot (pk_{B^*})^{\sum_{i \in [n]} u_i v_i + srz_{B^*}}.$$

$\mathcal{C}$ returns the signature $\sigma = (U, W, s)$ to $\mathcal{A}_{\mathcal{I}}$. From the following derivation process, it can be known that σ is a valid signature.

$$\begin{aligned}
& e\left(H(w_{B^*})^{\sum_{i \in [n]} v_i}, pk_A\right) \cdot e\left(H(w_{B^*})^s \cdot \prod_{i \in [n]} Q_i^{v_i}, U\right) \cdot e\left(Q^{\sum_{i \in [n]} v_i}, pk_{B^*}\right) \\
&= e\left(H(w_{B^*})^{x_A \sum_{i \in [n]} v_i}, g\right) \cdot e\left(H(w_{B^*})^s \cdot \prod_{i \in [n]} \left(\frac{g^{u_i}}{Q}\right)^{r^{-1} v_i}, pk_{B^*}^r\right) \cdot e\left(Q^{\sum_{i \in [n]} v_i}, pk_{B^*}\right) \\
&= e\left(H(w_{B^*})^{x_A \sum_{i \in [n]} v_i}, g\right) \cdot e\left(H(w_{B^*})^s, pk_{B^*}^r\right) \cdot e\left(g^{\sum_{i \in [n]} u_i v_i}, pk_{B^*}\right) \cdot e\left(Q^{-\sum_{i \in [n]} v_i}, pk_{B^*}\right) \\
&\quad \cdot e\left(Q^{\sum_{i \in [n]} v_i}, pk_{B^*}\right) \\
&= e\left(H(w_{B^*})^{x_A \sum_{i \in [n]} v_i}, g\right) \cdot e\left(H(w_{B^*})^s, pk_{B^*}^r\right) \cdot e\left(g^{\sum_{i \in [n]} u_i v_i}, pk_{B^*}\right) \\
&= e\left(H(w_{B^*})^{x_A \sum_{i \in [n]} v_i} \cdot (pk_{B^*})^{\sum_{i \in [n]} u_i v_i + srz_{B^*}}, g\right) \\
&= e(W, g).
\end{aligned}$$

- **Output:** Finally, $\mathcal{A}_{\mathcal{I}}$ outputs $(pk_A, pk_{B^*}, w_{B^*}, \tau^*, \boldsymbol{v}^*, \sigma^*)$, where $\boldsymbol{v}^* = (v_1^*, \ldots, v_n^*)$, $\sum_{i\in[n]} v_i^* \neq 0$ and $\sigma^* = (U^*, W^*, s^*)$. $\mathcal{C}$ retrieves the items $H(w_{B^*})$ from L_H, the items Q_i from L_{H_1}, and the item Q from L_{H_2}; Note that $H(w_{B^*}) = g^{z_{B^*}}$, $Q_i = g^{t_i^*}$, $Q = (g^b)^{t^*}$. If the $\mathcal{A}_{\mathcal{I}}$ successfully outputs a Type 1 forgery, then $\tau^* \neq \tau_i$, i.e., none of the message vectors identified by τ^* have been queried during the *Signing Queries* phase, and there is a verification equation.

$$\begin{aligned} &e(W^*, g) \\ &= e\left(H(w_{B^*})^{\sum_{i\in[n]} v_i^*}, pk_A\right) \cdot e\left(H(w_{B^*})^{s^*} \prod_{i\in[n]} (Q_i)^{v_i^*}, U^*\right) \cdot e\left((Q)^{\sum_{i\in[n]} v_i^*}, pk_{B^*}\right) \\ &= e\left(H(w_{B^*})^{x_A \sum_{i\in[n]} v_i^*} \cdot (U^*)^{s^* z_{B^*} + \sum_{i\in[n]} t_i^* v_i^*} \cdot \left(g^{ab}\right)^{\sum_{i\in[n]} t^* v_i^*}, g\right). \end{aligned}$$

Therefore, we have

$$W^* = H(w_{B^*})^{x_A \sum_{i\in[n]} v_i^*} \cdot (U^*)^{s^* z_{B^*} + \sum_{i\in[n]} t_i^* v_i^*} \cdot \left(g^{ab}\right)^{\sum_{i\in[n]} t^* v_i^*}.$$

The CDH problem can be solved by calculating the following equation:

$$g^{ab} = \left(\frac{W^*}{H(w_{B^*})^{x_A \sum_{i\in[n]} v_i^*} \cdot (U^*)^{s^* z_{B^*} + \sum_{i\in[n]} t_i^* v_i^*}}\right)^{\frac{1}{\sum_{i\in[n]} t^* v_i^*}}.$$

Next we calculate the probability of $\mathcal{C}$ success.

We only need to analyze the probability of $\mathcal{C}$ aborting the simulation during the signing query phase. When the hash value conflicts on H_1, that is, the same hash object appears in H_1 hash query and signing query, $\mathcal{C}$ aborts. Since the entry in L_{H_1} is not more than $q_{H_1} + q_S$, it is easy to know that the probability of $\mathcal{C}$ aborting in signing query is at most $\frac{q_S(q_{H_1} + q_S)}{2^\lambda}$. Therefore, if the advantage of $\mathcal{A}_{\mathcal{I}}$ forging a signature in Game 1 is ε, $\mathcal{C}$ can solve the CDH problem with a probability of at least $\varepsilon' = \frac{1}{2}\varepsilon - \frac{(q_S)^2 + q_{H_1} q_S}{2^\lambda}$, within time $t' = t + O(q_H + q_{H_1} + q_{H_2} + q_S) \times t_{exp} + q_S \times T_{mul}$, where T_{mul} and t_{exp} represent the multiplication and exponential operation time of group elements, respectively.

Case 2 (homomorphic existential-forgery): $\mathcal{C}$ guesses that adversary $\mathcal{A}_{\mathcal{I}}$ will output a Type 2 forgery. $\mathcal{C}$ is invoked on a tuple (g, g^a, g^b), and is asked to compute value of g^{ab} by using $\mathcal{A}_{\mathcal{I}}$ as a subroutine.

- **Setup:** $\mathcal{C}$ randomly selects $x_A \in \mathbb{Z}_q^*$ and lets $pk_A = g^{x_A}$, and runs $\mathcal{A}_{\mathcal{I}}$ on input $pk_{B^*} = g^a$ as the public key of Bob. $\mathcal{C}$ sets the system parameter $params = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, H_1, H_2, q)$. Then, $\mathcal{C}$ sends $params$ and (x_A, pk_A, pk_{B^*}) to $\mathcal{A}_{\mathcal{I}}$, and responds to $\mathcal{A}_{\mathcal{I}}$ as follows.
- **Queries:** $\mathcal{C}$ maintains a corresponding list in both the key registration queries and the hash queries, recording all responses to the $\mathcal{A}_{\mathcal{I}}$.

- *Key Registration Queries*: When $\mathcal{A}_{\mathcal{I}}$ outputs the key pair (pk_i, x_i) and requests to register a new user i, if $\mathcal{C}$ verifies that it is a valid key pair, it will add (pk_i, x_i) to the list L_K; otherwise, (pk_i, x_i) can be refused by $\mathcal{A}_{\mathcal{I}}$.
- *H Queries*: On a requested $w_i = w_{B^*}$, $\mathcal{C}$ sets $H(w_{B^*}) = g^b$; otherwise, $w_i \neq w_{B^*}$. $\mathcal{C}$ chooses a number $z_i \in \mathbb{Z}_q^*$ randomly, and lets $H(w_i) = g^{z_i}$. Then, $\mathcal{C}$ updates list $L_H \leftarrow L_H \cup < w_i, g^b, \perp >$ or $L_H \leftarrow L_H \cup < w_i, g^{z_i}, z_i >$, and responds $H(w_i)$ to $\mathcal{A}_{\mathcal{I}}$.
- H_1 *Queries*: For a requested (pk_A, w_B, τ, U, i), $\mathcal{C}$ first searches L_{H_1} and if an entry is found, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ chooses randomly $\alpha_i, \beta_i \in \mathbb{Z}_q^*$ and sets $Q_i = (g^b)^{\alpha_i} g^{\beta_i}$, then updates list $L_{H_1} \leftarrow L_{H_1} \cup < (pk_A, w_B, \tau, U, i), Q_i, \alpha_i, \beta_i >$, and returns Q_i to $\mathcal{A}_{\mathcal{I}}$.
- H_2 *Queries*: $\mathcal{A}_{\mathcal{I}}$ submits (w_i, pk_i), if $w_i = w_{B^*}$, $\mathcal{C}$ sets $Q = (g^b)^{t^*}$ with random $t^* \in \mathbb{Z}_q^*$. Otherwise, $\mathcal{C}$ selects a number $t \in \mathbb{Z}_q^*$ randomly and lets $Q = (g^b)^t$, then updates list $L_{H_2} \leftarrow L_{H_2} \cup < (w_{B^*}, pk_{B^*}), Q, t^* >$ or $L_{H_2} \leftarrow L_{H_2} \cup < (w_i, pk_i), Q, t >$, and sends Q to $\mathcal{A}_{\mathcal{I}}$ as response.
- *Signing Queries*: Given τ, pk_A, pk_{B_*}, a warrant w_{B^*}, and a vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, $\mathcal{C}$ generates the signature as follows:
  (1) If list L does not contain τ, $\mathcal{C}$ choose numbers $r \in \mathbb{Z}_q^*$ randomly, set $U = pk_{B^*}^r = (g^a)^r$, and stores (w_{B^*}, τ, U, r) in L. Otherwise, it recovers the required items from L.
  (2) $\mathcal{C}$ computes

$$W = \left(g^b\right)^{x_A \sum_{i\in[n]} v_i} \cdot \left(g^a\right)^{r \sum_{i\in[n]} \beta_i v_i}, s = -\sum_{i\in[n]} \left(\frac{t^*}{r} + \alpha_i\right) v_i.$$

and returns the signature $\sigma = (U, W, s)$ to $\mathcal{A}_{\mathcal{I}}$. The following analysis shows that σ is valid since $s = -\sum_{i\in[n]} \left(\frac{t^*}{r} + \alpha_i\right) v_i$, so $rs + r\sum_{i\in[n]} \alpha_i v_i + t^* \sum_{i\in[n]} v_i = 0$. Hence,

$$\begin{aligned}
&\left(S_{w_{B^*}}\right)^{\sum_{i\in[n]} v_i} \cdot \left(H\left(w_{B^*}\right)^s \cdot \prod_{i\in[n]} Q_i^{v_i}\right)^{ra} \cdot \left(Q^{\sum_{i\in[n]} v_i}\right)^{x_{B^*}} \\
&= \left(g^b\right)^{x_A \sum_{i\in[n]} v_i} \cdot \left(g^{bs} \cdot g^{b \sum_{i\in[n]} \alpha_i v_i} \cdot g^{\sum_{i\in[n]} \beta_i v_i}\right)^{ra} \cdot \left(g^{ab}\right)^{t^* \sum_{i\in[n]} v_i} \\
&= \left(g^b\right)^{x_A \sum_{i\in[n]} v_i} \cdot g^{ab\left(rs + r\sum_{i\in[n]} \alpha_i v_i + t^* \sum_{i\in[n]} v_i\right)} \cdot g^{ra \sum_{i\in[n]} \beta_i v_i} \\
&= \left(g^b\right)^{x_A \sum_{i\in[n]} v_i} \cdot \left(g^a\right)^{r \sum_{i\in[n]} \beta_i v_i} \\
&= W.
\end{aligned}$$

- **Output:** Finally, $\mathcal{A}_{\mathcal{I}}$ outputs $(pk_A, pk_{B^*}, w_{B^*}, \tau^*, \boldsymbol{v}^*, \sigma^*)$, where $\boldsymbol{v}^* = (v_1^*, \ldots, v_n^*)$, $\sum_{i\in[n]} v_i^* \neq 0$ and $\sigma^* = (U^*, W^*, s^*)$. $\mathcal{C}$ retrieves the items $H(w_{B_*})$, T_i and T from L_H, L_{H_1}, and L_{H_2}, respectively. Note that $H(w_{B_*}) = g^b$, $Q_i = (g^b)^{\alpha_i} g^{\beta_i}$, $Q = (g^b)^{t^*}$. If $\mathcal{A}_{\mathcal{I}}$ successfully outputs a Type 2 forgery, it is not hard to see that $U^* = pk_{B^*}^r = (g^a)^r$, and the following equation holds:

$$\begin{aligned}
&e(W^*, g)\\
&= e\left(H\left(w_{B_*}\right)^{\sum_{i\in[n]} v_i^*}, pk_A\right)\cdot e\left(H\left(w_{B_*}\right)^{s^*}\prod_{i\in[n]}(Q_i^*)^{v_i^*}, U^*\right)\cdot e\left((Q^*)^{\sum_{i\in[n]} v_i^*}, pk_{B^*}\right)\\
&= e\left(\left(g^b\right)^{x_A\sum_{i\in[n]} v_i^*}, g\right)\cdot e\left(g^{bs^*+b\sum_{i\in[n]}\alpha_i v_i^*}\cdot g^{\sum_{i\in[n]}\beta_i v_i^*}, g^{ar}\right)\cdot e\left(g^{bt^*\sum_{i\in[n]} v_i^*}, g^a\right)\\
&= e\left(g^{bx_A\sum_{i\in[n]} v_i^*}\cdot g^{ab\left(rs^*+r\sum_{i\in[n]}\alpha_i v_i^* + t^*\sum_{i\in[n]} v_i^*\right)}\cdot g^{ar\sum_{i\in[n]}\beta_i v_i^*}, g\right).
\end{aligned}$$

Therefore, we have

$$W^* = \left(g^b\right)^{x_A\sum_{i\in[n]} v_i^*}\cdot\left(g^{ab}\right)^{rs^*+\sum_{i\in[n]}\left(t^*+r\alpha_i\right)v_i^*}\cdot g^{ar\sum_{i\in[n]}\beta_i v_i^*}.$$

If $s \neq -\sum_{i\in[n]}\left(\frac{t^*}{r}+\alpha_i\right)v_i^*$, then it holds that $rs^* + \sum_{i\in[n]}\left(t^*+r\alpha_i\right)v_i^* \neq 0$. So $\mathcal{C}$ can compute

$$g^{ab} = \left(\frac{W^*}{g^{bx_A\sum_{i\in[n]} v_i^*}\cdot g^{ar\sum_{i\in[n]}\beta_i v_i^*}}\right)^{\frac{1}{rs^*+\sum_{i\in[n]}\left(t^*+r\alpha_i\right)v_i^*}}.$$

Now, we show that $s = -\sum_{i\in[n]}\left(\frac{t^*}{r}+\alpha_i\right)v_i^*$ with probability $\frac{1}{q}$. Since for each τ, $\mathcal{A}_{\mathcal{I}}$ queries signatures on independent vectors, and all signed messages are n-dimensional vectors, we can assume without loss of generality that $\mathcal{A}_{\mathcal{I}}$ performs at most signing queries for the file identifier τ^* and obtains $\sigma_i = (U^*, W_i, s_i)$ for $\boldsymbol{v}_i = (v_{i1}, \ldots, v_{in}), i \in [n-1]$. $\mathcal{C}$ recovers the corresponding r, t^* from L and L_{H_2} respectively. From the above simulation process, we can see that the $n-1$ value s_i satisfies the equation $s_i = -\sum_{j\in[n]}\left(\frac{t^*}{r}+\alpha_j\right)v_{ij}$ for $i \in [n-1]$. So we have

$$\begin{cases}
-s_1 = \left(\frac{t^*}{r}+\alpha_1\right)v_{11} + \cdots + \left(\frac{t^*}{r}+\alpha_n\right)v_{1n},\\
\qquad\qquad\qquad\vdots\\
-s_{n-1} = \left(\frac{t^*}{r}+\alpha_1\right)v_{n-1,1} + \cdots + \left(\frac{t^*}{r}+\alpha_n\right)v_{n-1,n}.
\end{cases}$$

Moreover, according to the definition of type 2 forgery, $\boldsymbol{v}^* \notin Span(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1})$. Therefore, the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}, \boldsymbol{v}^*$ identified by τ^* are linearly independent vectors.

Assume $\mathcal{A}_{\mathcal{I}}$ outputs a forgery $\sigma^* = (U^*, W^*, s^*)$ with respect to $\boldsymbol{v}^*$ such that $s^* = -\sum_{i\in[n]} \left(\frac{t^*}{r} + \alpha_i\right) v_i^*$, then combined with the above equations, we have

$$\begin{cases} -s_1 = \left(\frac{t^*}{r} + \alpha_1\right) v_{11} + \cdots + \left(\frac{t^*}{r} + \alpha_n\right) v_{1n}, \\ \qquad\qquad\vdots \\ -s_{n-1} = \left(\frac{t^*}{r} + \alpha_1\right) v_{n-1,1} + \cdots + \left(\frac{t^*}{r} + \alpha_n\right) v_{n-1,n}, \\ -s^* = \left(\frac{t^*}{r} + \alpha_1\right) v_1^* + \cdots + \left(\frac{t^*}{r} + \alpha_n\right) v_n^*. \end{cases}$$

Note that $\boldsymbol{v_1}, \ldots, \boldsymbol{v_{n-1}}, \boldsymbol{v}^*$ are linearly independent vectors, so

$$\begin{vmatrix} v_{11} & \cdots & v_{1n} \\ & \vdots & \\ v_{n-1,1} & \cdots & v_{n-1,n} \\ v_1^* & \cdots & v_n^* \end{vmatrix} \neq 0.$$

From Gramer criterion, $\mathcal{A}_{\mathcal{I}}$ can obtain $\left\{\frac{t^*}{r} + \alpha_i\right\}_{i\in[n]}$ and further obtain the random numbers $\{\alpha_i\}_{i\in[n]}$. However, as $\{\alpha_i\}_{i\in[n]}$ are independent of $\mathcal{A}_{\mathcal{I}}$'s view, it can be concluded that $s = -\sum_{i\in[n]} \left(\frac{t^*}{r} + \alpha_i\right) v_i^*$ holds randomly with a probability of $\frac{1}{q}$. Therefore, if the advantage of $\mathcal{A}_{\mathcal{I}}$ forging a signature in Game 1 is ε, $\mathcal{C}$ can solve the CDH problem with a probability of at least $\frac{1}{2}\varepsilon\left(1 - \frac{1}{q}\right)$ within time $t' = t + O(q_H + q_{H_1} + q_{H_2} + q_S) \times t_{exp} + 2q_S \times T_{mul}$.

Theorem 6.2. If there is an adversary $\mathcal{A}_{\mathcal{II}}$ that can forge a valid message/signature pair with an advantage ε within time t, then there is an algorithm $\mathcal{C}$ that can solve the CDH problem with probability ε' within time t'. Among them, q_H refers to the number of querying H oracle, $q_{H_i}(i = 1, 2)$ refers to the number of querying H_i oracles, q_D refers to the number of querying delegation oracle, q_S refers to the number of querying signing oracle.

Proof. For any PPT adversary $\mathcal{A}_{\mathcal{II}}$, which represents a malicious proxy signer, we construct a simulator $\mathcal{C}$, which can call $\mathcal{A}_{\mathcal{II}}$ to solve the CDH problem. As in Theorem 1, the probability of $\mathcal{C}$ guessing the forgery type outputed eventually by $\mathcal{A}_{\mathcal{II}}$ is $\frac{1}{2}$.

Case 1 (usual existential-forgery): $\mathcal{C}$ is invoked on a triple (g, g^a, g^b) of the CDH problem, and $\mathcal{C}$' goal is to compute g^{ab}.

- **Setup:** $\mathcal{C}$ set Alice's public key $pk_A = g^a$, and returns pk_A and systems $params = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, H_1, H_2, q)$ to $\mathcal{A}_{\mathcal{II}}$, then responds to $\mathcal{A}_{\mathcal{II}}$ as follows:
- **Queries:** $\mathcal{C}$ maintains an initially empty list for each type of query, recording all responses to $\mathcal{A}_{\mathcal{II}}$.
  - *Key Registration Queries*: As in the query/response in the proof of case 1 in Theorem 1, it is omitted here.

    Suppose that $\mathcal{A}_{\mathcal{II}}$ gets the warrants $w_i(i \in [q_H])$ from $\mathcal{C}$ before this queries. $\mathcal{C}$ randomly selects $k \in \{1, \ldots, q_H\}$, and guesses that the k-th warrant w_k submitted by $\mathcal{A}_{\mathcal{II}}$ is the targeted warrant. On a requested w_i, if $i = k$, $\mathcal{C}$ outputs the value $H(w_k) = g^b$ and adds $< w_k, g^b, \perp >$

to list L_H. Otherwise, $\mathcal{C}$ randomly chooses $z_i \in \mathbb{Z}_q^*$ and returns the value $H(w_i) = g^{z_i}$, and adds $< w_i, g^{z_i}, z_i >$ to L_H.

- H_1 *Queries*: For a requested (Pk_A, w_B, τ, U, i), $\mathcal{C}$ first searches L_{H_1} and if an entry is found, $\mathcal{C}$ returns it. Otherwise, $\mathcal{C}$ selects a random $t_i \in \mathbb{Z}_q^*$ and lets $Q_i = g^{t_i}$, then updates list $L_{H_1} \leftarrow L_{H_1} \cup < (pk_A, w_B, \tau, U, i), Q_i, t_i >$ and sends Q_i to $\mathcal{A}_{\mathcal{II}}$ as response.
- H_2 *Queries*: On a requested (w_B, pk_B), $\mathcal{C}$ checks the list L_{H_2} to determine whether there is a corresponding hash value, if so, return it to $\mathcal{A}_{\mathcal{I}}$; Otherwise, $\mathcal{C}$ chooses a random number $t \in \mathbb{Z}_q^*$ and sets $Q = g^t$, then updates list $L_{H_2} \leftarrow L_{H_2} \cup < (w_B, pk_B), Q, t >$ and sends Q to $\mathcal{A}_{\mathcal{II}}$ as response.
- *Delegation Queries*: Given a warrant w_i, if $w_i \neq w_k$, $\mathcal{C}$ recovers the tuple $< w_i, g^{z_i}, z_i >$ from L_H, and returns the delegation $S_{w_i} = (pk_A)^{z_i}$ to $\mathcal{A}_{\mathcal{II}}$ and adds $< w_i, S_{w_i} >$ to list L_D. Otherwise, $\mathcal{C}$ aborts.
- *Signing Queries*: Given τ, pk_A, pk_i, a warrant w_i, and a vector $v = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, $\mathcal{C}$ checks the lists defined above and responses as follows:
  (1) If $w_i \neq w_k$, $\mathcal{C}$ chooses random numbers $s, r \in \mathbb{Z}_q^n$, sets $U = g^r$, and computes the signature

$$W = \left(S_{w_i}\right)^{\sum\limits_{i\in[n]} v_i} \left(H\left(w_i\right)^s \cdot \prod_{i\in[n]} Q_i^{v_i}\right)^r \left(Q^{\sum\limits_{i\in[n]} v_i}\right)^{x_i}$$
$$= (pk_A)^{z_i \sum\limits_{i\in[n]} v_i} \cdot \left(g^{z_i s} \cdot g^{\sum\limits_{i\in[n]} t_i v_i}\right)^r \cdot g^{t x_i \sum\limits_{i\in[n]} v_i}.$$

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

- **Output:** $\mathcal{A}_{\mathcal{II}}$ outputs a tuple $(pk_A, pk_{B^*}, w_{B^*}, \tau^*, \boldsymbol{v}^*, \sigma^*)$, where $\boldsymbol{v}^* = (v_1^*, \ldots, v_n^*)$, $\sum_{i\in[n]} v_i^* \neq 0$ and $\sigma^* = (U^*, W^*, s^*)$. If $w_{B^*} \neq w_k$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves the required hash values $H(w_{B_*})$, Q_i and Q from the lists L_H, L_{H_1} and L_{H_2}, respectively, note that $H(w_{B^*}) = g^b$, $Q_i^* = g^{t_i^*}$, $Q^* = g^{t^*}$. If the $\mathcal{A}_{\mathcal{II}}$ successfully outputs a Type 1 forgery, then $\tau^* \neq \tau_i$, i.e., none of the message vectors identified by τ^* have been queried during the *Signing Queries* phase, and there is a verification equation.

$$e(W^*, g)$$
$$= e\left(H\left(w_{B_*}\right)^{\sum\limits_{i\in[n]} v_i^*}, pk_A\right) \cdot e\left(H\left(w_{B_*}\right)^{s^*} \prod_{i\in[n]} (Q_i^*)^{v_i^*}, U^*\right) \cdot e\left((Q^*)^{\sum\limits_{i\in[n]} v_i^*}, pk_{B^*}\right)$$
$$= e\left(g^{ab}, g\right)^{\sum\limits_{i\in[n]} v_i^*} \cdot e\left(\left(g^b\right)^{rs^*} \cdot (U^*)^{\sum\limits_{i\in[n]} t_i^* v_i^*}, g\right) \cdot e\left(pk_{B^*}^{t^* \sum\limits_{i\in[n]} v_i^*}, g\right).$$

Further, we have

$$W^* = \left(g^{ab}\right)^{\sum\limits_{i\in[n]} v_i^*} \cdot \left(g^b\right)^{rs^*} \cdot (U^*)^{\sum\limits_{i\in[n]} t_i^* v_i^*} \cdot (pk_{B^*})^{t^* \sum\limits_{i\in[n]} v_i^*}.$$

Next, $\mathcal{C}$ solves the CDH problem by calculating the following equation:

$$g^{ab} = \left(\frac{W^*}{(g^b)^{rs^*} \cdot (U^*)^{\sum_{i\in[n]} t_i^* v_i^*} \cdot (pk_{B^*})^{t^* \sum_{i\in[n]} v_i^*}} \right)^{\frac{1}{\sum_{i\in[n]} v_i^*}}.$$

Then the CDH problem has been solved.

It is not difficult to see that the probability of not aborting in delegation queries, signing queries and forgery stage is at least $1 - \frac{q_D}{q_H}$, $1 - \frac{q_S}{q_H}$, and $\frac{1}{q_H}$. Thus, if the advantage of $\mathcal{A}_{\mathcal{II}}$ forging a signature is ε, $\mathcal{C}$ can solve the CDH problem with a probability of at least $\frac{1}{2}\left(1 - \frac{q_D}{q_H}\right) \cdot \left(1 - \frac{q_S}{q_H}\right) \cdot \frac{1}{q_H}\varepsilon$ within time $t' = t + O(qH + qH_1 + qH_2 + qD + qS) \times t_{exp} + q_S \times T_{mul}$.

Case 2 (homomorphic existential-forgery): $\mathcal{C}$ guesses that adversary $\mathcal{A}_{\mathcal{II}}$ will output a Type 2 forgery. $\mathcal{C}$ is invoked on a triple (g, g^a, g^b), and is asked to compute value of g^{ab} by using $\mathcal{A}_{\mathcal{II}}$ as a subroutine.

- **Setup:** $\mathcal{C}$ invokes $\mathcal{A}_{\mathcal{II}}$ on input $pk_A = g^a$ as the public key of Alice, and returns pk_A and systems $params = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, H_1, H_2, q)$ to $\mathcal{A}_{\mathcal{II}}$. Then, $\mathcal{C}$ responds to $\mathcal{A}_{\mathcal{II}}$ as follows:
- **Queries:** $\mathcal{C}$ maintains an initially empty list for each type of query, recording all responses to $\mathcal{A}_{\mathcal{II}}$.
  - *Key Registration Queries*, *H* (H_2) *Queries*: As in the query/response in the proof of case 1 in Theorem 2, it is omitted here.
  - H_1 *Queries*: $\mathcal{A}_{\mathcal{II}}$ submits (pk_A, w_B, τ, U, i). $\mathcal{C}$ checks L_{H_1} to find out whether the required hash value exists, If so, $\mathcal{C}$ sends it to $\mathcal{A}_{\mathcal{II}}$; otherwise, $\mathcal{C}$ selects randomly $\alpha_i, \beta_i \in \mathbb{Z}_q^*$, let $Q_i = (g^b)^{\alpha_i} g^{\beta_i}$, then updates list $L_{H_1} \leftarrow L_{H_1} \cup < (pk_A, w_B, \tau, U, i), Q_i, \alpha_i, \beta_i >$, and sends Q_i to $\mathcal{A}_{\mathcal{II}}$.
  - *Delegation Queries*: Given a warrant w_i, if $w_i \neq w_k$, $\mathcal{C}$ recovers the tuple $< w_i, g^{z_i}, z_i >$ from L_H, and returns the delegation $S_{w_i} = (pk_A)^{z_i} = (g^a)^{z_i}$ to $\mathcal{A}_{\mathcal{II}}$ and adds $< w_i, S_{w_i} >$ to L_D. Otherwise, $\mathcal{C}$ aborts.
  - *Signing Queries*: Given pk_A, pk_B, τ, a warrant w_B, and a vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$. Assuming w.l.o.g $\mathcal{A}_{\mathcal{II}}$ has inquired the corresponding hash queries and delegation queries, and retrieve the required items by searching the list $L_H, L_{H_1}, L_{H_2}, L_D$, then $\mathcal{C}$ performs the following steps:
    (1) If $w_B \neq w_k$, $\mathcal{C}$ can sign the message vector in the same way as the real **LHP-Sign** algorithm, which is omitted here.
    (2) If $w_B = w_k$, $\mathcal{C}$ first checks whether there is a record (w_B, τ, U, r) in list L, if it exists, $\mathcal{C}$ restores (U, R), otherwise, it selects $r \in \mathbb{Z}_q^*$ randomly, sets $U = (pk_A)^r = (g^a)^r$, and stores (w_B, τ, U, r) in L. Then, $\mathcal{C}$ compute

$$W = (g^a)^{r\sum_{i\in[n]} \beta_i v_i} \cdot g^{tx_k \sum_{i\in[n]} v_i}, s = -\sum_{i\in[n]} \left(\frac{1}{r} + \alpha_i\right) v_i.$$

$\mathcal{C}$ returns the signature $\sigma = (U, W, s)$ to $\mathcal{A}_{\mathcal{II}}$. The following analysis shows that σ is valid since $s = -\sum_{i\in[n]}\left(\frac{1}{r}+\alpha_i\right)v_i$, so $rs + \sum_{i\in[n]} v_i + r\sum_{i\in[n]}\alpha_i v_i = rs + \sum_{i\in[n]}(1+r\alpha_i)\,v_i = 0$. Hence,

$$\begin{aligned}
&\left(S_{w_k}\right)^{\sum_{i\in[n]} v_i}\cdot\left(H(w_k)^s\cdot\prod_{i\in[n]} Q_i^{v_i}\right)^{ar}\cdot\left(Q^{\sum_{i\in[n]} v_i}\right)^{x_k}\\
&=\left(g^{ab}\right)^{\sum_{i\in[n]} v_i}\cdot\left(g^{bs}\cdot g^{b\sum_{i\in[n]}\alpha_i v_i}\cdot g^{\sum_{i\in[n]}\beta_i v_i}\right)^{ar}\cdot\left(g^{t\sum_{i\in[n]} v_i}\right)^{x_k}\\
&=\left(g^{ab}\right)^{\left(rs+\sum_{i\in[n]} v_i + r\sum_{i\in[n]}\alpha_i v_i\right)}\cdot g^{ar\sum_{i\in[n]}\beta_i v_i}\cdot g^{tx_k\sum_{i\in[n]} v_i}\\
&=\left(g^{a}\right)^{r\sum_{i\in[n]}\beta_i v_i}\cdot g^{tx_k\sum_{i\in[n]} v_i}\\
&= W.
\end{aligned}$$

- **Output:** Finally, $\mathcal{A}_{\mathcal{II}}$ outputs $(pk_A, pk_{B^*}, w_{B^*}, \tau^*, \boldsymbol{v}^*, \sigma^*)$, where $\boldsymbol{v}^* = (v_1^*, \ldots, v_n^*)$, $\sum_{i\in[n]} v_i^* \neq 0$ and $\sigma^* = (U^*, W^*, s^*)$. If $w_{B^*} \neq w_k$, $\mathcal{C}$ aborts. Otherwise, it retrieves the items $H(w_{B^*})$, Q_i and Q from L_H, L_{H_1}, and L_{H_2}, respectively. Note that $H(w_{B^*}) = g^b$, $Q_i = (g^b)^{\alpha_i} g^{\beta_i}$, $Q = g^{t^*}$. If $\mathcal{A}_{\mathcal{II}}$ outputs Type 2 forgery signatures successlly, it is not hard to see that $U^* = pk_A^r = (g^a)^r$, and there are the following series of equations:

$$\begin{aligned}
&e(W^*, g)\\
&= e\left(H\left(w_{B_*}\right)^{\sum_{i\in[n]} v_i^*}, pk_A\right)\cdot e\left(H\left(w_{B_*}\right)^{s^*}\cdot\prod_{i\in[n]}(Q_i)^{v_i^*}, U^*\right)\cdot e\left(Q^{\sum_{i\in[n]} v_i^*}, pk_{B_*}\right)\\
&= e\left(\left(g^{ab}\right)^{\sum_{i\in[n]} v_i^*}, g\right)\cdot e\left(g^{bs^*+b\sum_{i\in[n]}\alpha_i v_i^*}\cdot g^{\sum_{i\in[n]}\beta_i v_i^*}, g^{ar}\right)\cdot e\left(\left(pk_{B_*}\right)^{t^*\sum_{i\in[n]} v_i^*}, g\right)\\
&= e\left(g^{ab\left(rs^*+\sum_{i\in[n]}(1+r\alpha_i)\,v_i^*\right)}\cdot (U^*)^{\sum_{i\in[n]}\beta_i v_i^*}\cdot\left(pk_{B_*}\right)^{t^*\sum_{i\in[n]} v_i^*}, g\right).
\end{aligned}$$

Therefore, we have

$$W^* = g^{ab\left(rs^*+\sum_{i\in[n]}(1+r\alpha_i)\,v_i^*\right)}\cdot (U^*)^{\sum_{i\in[n]}\beta_i v_i^*}\cdot\left(pk_{B_*}\right)^{t^*\sum_{i\in[n]} v_i^*}.$$

If $s^* \neq -\sum_{i\in[n]} \left(\frac{1}{r} + \alpha_i\right) v_i^*$, then it holds that $rs^* + \sum_{i\in[n]} (1 + r\alpha_i) v_i^* \neq 0$. So $\mathcal{C}$ can compute the value of g^{ab} as follows:

$$g^{ab} = \left(\frac{W^*}{(U^*)^{\sum_{i\in[n]} \beta_i v_i^*} \cdot (pk_{B_*})^{t^* \sum_{i\in[n]} v_i^*}} \right)^{\frac{1}{rs^* + \sum_{i\in[n]} (1 + r\alpha_i) v_i^*}}.$$

Now, we need to show that $s^* = -\sum_{i\in[n]} \left(\frac{1}{r} + \alpha_i\right) v_i^*$ with probability $\frac{1}{q}$, the proof method is the same as in case 2 of Theorem 1, so it is omitted here. Therefore, if the advantage of $\mathcal{A}_{\mathcal{II}}$ forging a signature is ε, $\mathcal{C}$ can solve the CDH problem with a probability of at least $\frac{1}{2}\varepsilon \left(1 - \frac{1}{q}\right) \cdot \frac{1}{q_H}$ within time $t' = t + O(qH + qH_1 + qH_2 + qS) \times t_{exp} + 2q_S \times T_{mul}$.

7 Analysis of Security and Efficiency

We compare our scheme with the only LHPS scheme proposed by Lin et al. [12]. For convenience, we abbreviate this scheme as Lin-LHPS. Specifically, Tables 1 and 2 compare the proposed LHPS scheme with the Lin-LHPS scheme in terms of security, communication overhead and delegation overhead. Two main operations are listed: "E" represents the exponential operation on $\mathbb{G}_1$, and "P" represents bilinear pairing operation. $|\mathbb{G}_1|$ and $|q|$ represent the bit length of the elements in a group $\mathbb{G}_1$ and a field $\mathbb{F}_q$, respectively.

Table 1: Comparison of security

	Type-1 adversary		Type-2 adversary	
Scheme	U-EUF	H-EUF	U-EUF	H-EUF
Lin-LHPS [12]	√	×	√	×
Proposed scheme	√	√	√	√

Note: U-EUF: Usual Existential-Unforgeability; H-EUF: Homomorphic Existential-Unforgeability.

Table 2: Comparison of communication overheads and computational complexity

	Communication overhead		Delegation overhead	
Scheme	DeleSize	SigSize	DeleGen	DeleVer
Lin-LHPS [12]	$\mathbb{G}_1$	$\mathbb{G}_1$	E	$2P$
Proposed scheme	$\mathbb{G}_1$	$2\mathbb{G}_1 + \lvert q\rvert$	E	$2P$

Note: DeleSize: the size of delegation; SigSize: the size of the proxy signature; DeleGen: the computational cost of generating the delegation signature; DeleVer: the computational cost of delegation verification.

As seen in Table 1, our scheme satisfies both the usual and homomorphic existential unforgeability, while the Lin-LHPS scheme only satisfies the usual existential unforgeability. In the outsourced computation of authentication data, homomorphic existential unforgeability can prevent untrusted cloud servers from authenticating the wrong computing or analysis results and returning them to the receiver. In Table 2, DeleSize and SigSize represent the size of the delegation and the proxy signature, respectively. The two columns, DeleGen and DeleVer, show the computational cost of generating the delegation signature and delegation verification, respectively. Table 2 shows that the signature size of the proposed scheme is larger than that of the Lin-LHPS scheme; whereas the size of delegation, the computational cost of generating the delegation signature and delegation verification in our scheme are the same as those in the Lin-LHPS scheme. However, our scheme provides stronger security. We implement our LHPS scheme and Lin-LHPS in the experiments. The experiment was run on a laptop equipped with a 3.10-GHz Intel i5 CPU, 128 GB memory, and the Ubuntu Linux operating system.

As shown in Figs. 3 and 4, the computational cost of signature generation and signature verification in our scheme is slightly higher than that in the Lin-LHPS. It is a natural result because our scheme provides provably secure homomorphic existential unforgeability.

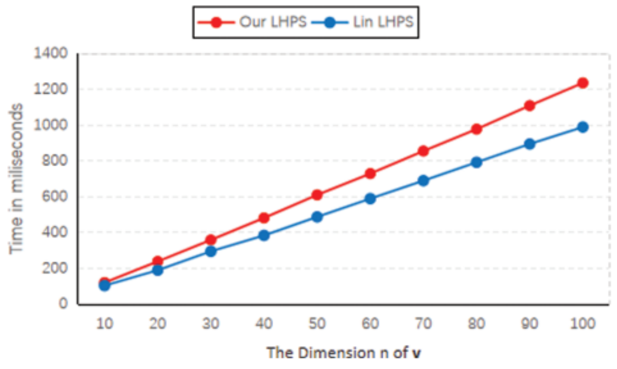

Figure 3: A comparison of the signature generation cost

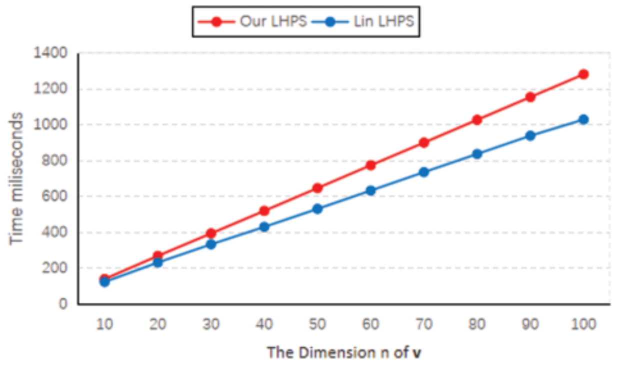

Figure 4: A comparison of the signature verification cost

8 Conclusions

In this paper, we improve the security model of the LHPS for the two types of adversaries, considering the situation in which adversaries output homomorphic existential forgeries. Under the new model, we present a blockchain-based LHPS scheme and prove that this scheme is secure against existential forgery (including the usual existential forgery and homomorphic existential forgery) under adaptive CMA based on the CDH assumption. The tamper-proof modification of the blockchain ensures the validity of the original signer's warrant, which prevents problems such as the abuse of proxy signing rights. Moreover, the performance analysis shows that this new scheme has the same key size and comparable computing cost as Lin et al.'s LHPS scheme [12], and it has higher security. Therefore, our LHPS scheme is suitable for deployment and implementation in cloud-based IoT environments. However, the proposed scheme cannot resist quantum computing attacks. The next task is to design a secure and efficient LHPS scheme on lattice.

Funding Statement: This research is funded by the Special Innovation Project for General Colleges and Universities in Guangdong Province (Grant No. 2020KTSCX126).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Atzori, L., Iera, A., Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks, 54,* 2787–2805. DOI 10.1016/j.comnet.2010.05.010.
2. Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University-Computer and Information Sciences, 30(3),* 291–319. DOI 10.1016/j.jksuci.2016.10.003.
3. Xu, L., He, W., Li, S. (2014). Internet of Things in industries: A survey. *IEEE Transactions on Industrial Informatics, 10(4),* 2233–2243. DOI 10.1109/TII.2014.2300753.
4. Doukas, C., Pliakas, T., Maglogiannis, I. (2010). Mobile healthcare information management utilizing cloud computing and android OS. *Proceedings of the 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, Buenos, Aires, Argentina. DOI 10.1109/IEMBS.2010.5628061.
5. Li, X., Huang, X., Li, C., Yu, R., Shu, L. (2019). EdgeCare: Leveraging edge computing for collaborative data management in mobile healthcare systems. *IEEE Access, 7,* 22011–22025. DOI 10.1109/ACCESS.2019.2898265.
6. Hu, X., Zheng, S., Gong, J., Cheng, G., Zhang, G. et al. (2019). Enabling linearly homomorphic signatures in network coding-based named data networking. *Proceedings of the 14th International Conference on Future Internet Technologies*, New York, USA. DOI 10.1145/3341188.3341191.
7. Schabhüser, L., Buchmann, J., Struck, P. (2017). A linearly homomorphic signature scheme from weaker assumptions. *IMA International Conference on Cryptography and Coding*, Oxford, UK. DOI 10.1007/978-3-319-71045-7_14.
8. Luo, F., Wang, F., Wang, K., Chen, K. (2019). A more efficient leveled strongly-unforgeable fully homomorphic signature scheme. *Information Sciences, 480,* 70–89. DOI 10.1016/j.ins.2018.12.025.
9. Boneh, D., Freeman, D. M. (2011). Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. *International Workshop on Public Key Cryptography*, Taormina, Italy. DOI 10.1007/978-3-642-19379-8_1.
10. Wang, F., Shi, S., Wang, C. (2019). Leveled lattice-based linearly homomorphic signature scheme in the standard model for network coding. *International Conference on Frontiers in Cyber Security*, Xi'an, China. DOI 10.1007/978-981-15-0818-9_6.
11. Mambo, M., Usuda, K., Okamoto, E. (1996). Proxy signatures: Delegation of the power to sign messages. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 79(9),* 1338–1354. CRID 1570009752558013440.

12. Lin, Q., Li, J., Huang, Z., Chen, W., Shen, J. (2018). A short linearly homomorphic proxy signature scheme. *IEEE Access, 6,* 12966–12972. DOI 10.1109/ACCESS.2018.2809684.

13. Goldwasser, S., Micali, S., Rivest, R. L. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing, 17(2),* 281–308. DOI 10.1137/0217017.

14. Johnson, R., Molnar, D., Song, D., Wagner, D. (2002). Homomorphic signature schemes. *Cryptographersâ Track at the RSA Conference*, San Jose, CA, USA. DOI 10.1007/3-540-45760-7_17.

15. Boneh, D., Freeman, D., Katz, J., Waters, B. (2009). Signing a linear subspace: Signature schemes for network coding. *Proceedings of International Workshop on Public Key Cryptography*, Irvine, CA, USA. DOI 10.1007/978-3-642-00468-1_5.

16. Zhang, Y., Jiang, Y., Li, B., Zhang, M. (2017). An efficient identity-based homomorphic signature scheme for network coding. *International Conference on Emerging Internetworking, Data and Web Technologies*, Wuhan, China. DOI 10.1007/978-3-319-59463-7_52.

17. SadrHaghighi, S., Khorsandi, S. (2016). An identity-based digital signature scheme to detect pollution attacks in intra-session network coding. *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology*, Tehran, Iran. DOI 10.1109/iscisc.2016.7736444.

18. Lin, Q., Yan, H., Huang, Z., Chen, W., Shen, J. et al. (2018). An ID-based linearly homomorphic signature scheme and its application in blockchain. *IEEE Access, 6,* 20632–20640. DOI 10.1109/ACCESS.2018.2809426.

19. Chang, J., Ma, H., Zhang, A., Xu, M., Xue, R. (2019). RKA security of identity-based homomorphic signature scheme. *IEEE Access, 7,* 50858–50868. DOI 10.1109/ACCESS.2019.2908244.

20. Chang, J., Ji, Y., Shao, B., Xu, M., Xue, R. (2020). Certificateless homomorphic signature scheme for network coding. *IEEE/ACM Transactions on Networking, 28(6),* 2615–2628. DOI 10.1109/TNET.2020.3013902.

21. Li, Y., Zhang, F., Sun, Y. (2021). Lightweight certificateless linearly homomorphic network coding signature scheme for electronic health system. *IET Information Security, 15(1),* 131–146. DOI 10.1049/ise2.12011.

22. Wang, F., Wang, K., Li, B., Gao, Y. (2015). Leveled strongly-unforgeable identity-based fully homomorphic signatures. *International Conference on Information Security*, Trondheim, Norway. DOI 10.1007/978-3-319-23318-5_3.

23. Fiore, D., Mitrokotsa, A., Nizzardo, L., Pagnin, E. (2016). Multi-key homomorphic authenticators. *International Conference on the Theory and Application of Cryptology and Information Security*, Hanoi, Vietnam. DOI 10.1007/978-3-662-53890-6_17.

24. Samarin, S. D., Fiore, D., Venturi, D., Amini, M. (2021). A compiler for multi-key homomorphic signatures for turing machines. *Theoretical Computer Science, 889,* 145–170. DOI 10.1016/j.tcs.2021.08.002.

25. Aranha, D. F., Pagnin, E. (2019). The simplest multi-key linearly homomorphic signature scheme. *International Conference on Cryptology and Information Security in Latin America*, Bogota, Colombia. DOI 10.1007/978-3-030-30530-7_114.

26. Schabhüser, L., Butin, D., Buchmann, J. (2019). Context hiding multi-key linearly homomorphic authenticators. *Cryptographersâ Track at the RSA Conference*, San Francisco, CA, USA. DOI 10.1007/978-3-030-12612-4_25.

27. Chen, W., Lei, H., Li, J., Gao, C., Li, F. et al. (2017). A multi-source homomorphic network coding signature in the standard model. *International Conference on Green, Pervasive, and Cloud Computing*, Xi'an, China. DOI 10.1007/978-3-319-57186-7_6.

28. Li, T., Chen, W., Tang, Y., Yan, H. (2018). A homomorphic network coding signature scheme for multiple sources and its application in IoT. *Security and Communication Networks, 2018,* 9641273. DOI 10.1155/2018/9641273.

29. Lai, R. W., Tai, R. K., Wong, H. W., Chow, S. S. (2018). Multi-key homomorphic signatures unforgeable under insider corruption. *International Conference on the Theory and Application of Cryptology and Information Security*, Brisbane, QLD, Australia. DOI 10.1007/978-3-030-03329-3_16.

30. Seo, J. H., Emura, K., Xagawa, K., Yoneyama, K. (2018). Accumulable optimistic fair exchange from verifiably encrypted homomorphic signatures. *International Journal of Information Security, 17(2),* 193–220. DOI 10.1007/s10207-017-0367-z.
31. Fan, X., Wu, T., Zheng, Q., Chen, Y., Alam, M. et al. (2020). HSE-voting: A secure high-efficiency electronic voting scheme based on homomorphic signcryption. *Future Generation Computer Systems, 111,* 754–762. DOI 10.1016/j.future.2019.10.016.
32. Li, S., Liang, B., Mitrokotsa, A., Xue, R. (2021). Homomorphic signcryption with public plaintext-result checkability. *IET Information Security, 15(5),* 333–350. DOI 10.1049/ise2.12026.
33. Gu, K., Jia, W., Li, C., Chen, R. (2013). Identity-based group proxy signature scheme in the standard model. *Journal of Computer Research and Development, 40(7),* 1370–1386. DOI 10.1016/j.mejo.2013.01.001.
34. Chen, M., Yuan, S. (2016). Provably secure identity-based multi-proxy signature scheme in standard model. *Journal of Computer Research and Development, 54(8),* 1879–1892. DOI 10.7544/issn1000-1239.2016.20150197.
35. Verma, G. K., Singh, B. B. (2017). Efficient message recovery proxy blind signature scheme from pairings. *Transaction on Emerging Telecommunication Technologies, 28(11),* e3167. DOI 10.1002/ett.3167.
36. Verma, G. K., Singh, B. B., Singh, H. (2018). Bandwidth efficient designated verifier proxy signature scheme for healthcare wireless sensor networks. *Ad Hoc Networks, 81,* 100–108. DOI 10.1016/j.adhoc.2018.07.026.
37. Verma, G. K., Singh, B. B., Singh, H. (2018). Provably secure certificate-based proxy blind signature scheme from pairings. *Information Sciences, 468,* 1–13. DOI 10.1016/j.ins.2018.08.031.
38. Verma, G. K., Singh, B. B., Kumar, N., Obaidat, M. S., He, D. et al. (2020). An efficient and provable certificate-based proxy signature scheme for IIoT environment. *Information Sciences, 518,* 142–156. DOI 10.1016/j.ins.2020.01.006.
39. Lee, J. Y., Cheon, J. H., Kim, S. (2003). An analysis of proxy signatures: Is a secure channel necessary? *Cryptographersâ Track at the RSA Conference*, San Francisco, CA, USA. DOI 10.1007/3-540-36563-X_5.
40. Cao, F., Cao, Z. (2009). A secure identity-based multi-proxy signature scheme. *Computers and Electrical Engineering, 35(1),* 86–95. DOI 10.1016/j.compeleceng.2008.05.005.
41. Wang, Q., Cao, Z., Wang, S. (2005). Formalized security model of multi-proxy signature schemes. *The Fifth International Conference on Computer and Information Technology*, Shanghai, China. DOI 10.1109/CIT.2005.119.
42. Schuldt, J. C., Matsuura, K., Paterson, K. G. (2008). Proxy signatures secure against proxy key exposure. *International Workshop on Public Key Cryptography*, Barcelona, Spain. DOI 10.1007/978-3-540-78440-1_9.
43. Boldyreva, A., Palacio, A., Warinschi, B. (2012). Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology, 25(1),* 57–115. DOI 10.1007/s00145-010-9082-x.
44. Han, D., Chen, J., Zhang, L., Shen, Y., Gao, Y. et al. (2021). A deletable and modifiable blockchain scheme based on record verification trees and the multisignature mechanism. *Computer Modeling in Engineering & Sciences, 128(1),* 223–245. DOI 10.32604/cmes.2021.016000.
45. Hu, N., Teng, Y., Zhao, Y., Yin, S., Zhao, Y. (2021). IDV: Internet domain name verification based on blockchain. *Computer Modeling in Engineering & Sciences, 129(1),* 299–322. DOI 10.32604/cmes.2021.016839.
46. Wang, Y., Qiu, W., Dong, L., Zhou, W., Pei, Y. et al. (2020). Proxy signature-based management model of sharing energy storage in blockchain environment. *Applied Sciences, 10(21),* 7502. DOI 10.3390/app10217502.
47. Zou, H., Liu, X., Ren, W., Zhu, T. (2022). A decentralized electronic reporting scheme with privacy protection based on proxy signature and blockchain. *Security and Communication Networks, 2022,* 5424395. DOI 10.1155/2022/5424395.