SCLPV: Secure Certificateless Public Verification for Cloud Storage in Cyber-physical-social System

Yuan Zhang[a], Chunxiang Xu[a], Shui Yu[b], Hongwei Li[a], Xiaojun Zhang[a]

[a]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*
[b]*the School of Information Technology, Deakin University, Melbourne, Australia*

Abstract

Cyber-physical-social system (CPSS) allows individuals to share personal information collected from not only cyberspace, but also physical space. This has resulted in generating numerous data at a user's local storage. However, it is very expensive for users to store large data sets, and it also causes problems in data management. Therefore, it is of critical importance to outsource the data to cloud servers, which provides users an easy, cost-effective and flexible way to manage data. Whereas, users lose control on their data once outsourcing their data to cloud servers, which poses challenges on integrity of outsourced data. Many mechanisms have been proposed to allow a third-party auditor to verify data integrity using the public keys of users. Most of these mechanisms bear a strong assumption: the auditors are honest and reliable, and thereby are vulnerability in the case that auditors are malicious. Moreover, in most of these approaches, an auditor needs to manage users certificates to choose the correct public keys for verification.

In this paper, we propose a secure certificateless public integrity verification scheme (SCLPV). The SCLPV scheme is the first work that simultaneously supports certificateless public verification and resistance against malicious auditors to verify the integrity of outsourced data in CPSS. A formal and strict security proof proves the correctness and security of our scheme. In addition, an elaborate performance analysis demonstrates that our scheme is efficient and practical. Compared with the best of the existing certificateless public verification scheme (CLPV), the SCLPV provides stronger security guarantees in terms of remedying the security vulnerability of the CLPV and resistance against malicious auditors. At the same time, in comparison with the best of integrity verification scheme achieving resistance

against malicious auditors, the communication cost between the auditor and the cloud server in the SCLPV is independent of the size of the processed data, meanwhile, the auditor in the SCLPV does not need to manage certificates.

Keywords:
Cyber-physical-social System, Cloud storage, Public integrity verification, Certificateless, Resistance against malicious auditors

---

1. Introduction

Cyber-physical-social system (CPSS) has been envisioned as the next phase of computing systems [1]. It combines measured elements of the physical world with manual human input, and seamlessly integrates physical components with traditional social networks [2], Typically, CPSS allows users to store and share information, locations and trajectories collected from personal devices, such as smart phones, sensors and so on. These extra data from physical world, which are numerously generated by CPSS users and collected by enterprised each day, are extraordinarily valuable not only to individuals themselves, but also to enterprises to better understand people's daily activities, social areas and life patterns [3]. From data owners' perspective, including both individuals and enterprises, outsourcing their data to cloud servers is a wise and practical choice, because cloud service provides users an efficient and flexible service to manage data. Such storage service enables users to access the cloud-stored data remotely through the Internet via different devices, without incurring substantial hardware, software and personal costs involved in deploying and maintaining applications in local storage [4], [5], [6], [7].

Although CPSS users enjoy the desirable features brought by the cloud storage service, there are critical security challenging in data outsourcing. Since CPSS users lose control on their data once outsourcing data to cloud servers, the integrity of cloud-stored data faces serious threats. For example, a cloud service provider may hide the fact that the cloud-stored data is corrupted in order to maintain his reputation [8], [9], [10], [11]. Meanwhile, an external adversary motivated by financial or political reward may be interested in distorting the outsourced data, but attempt to convince the users that their data are still maintained intact [12].

The integrity of the outsourced data has become the major concern of

CPSS users. Therefore, it is vital to regularly verify the integrity of cloud-stored data to react as early as possible in the case of data corruption. Some approaches rely on users themselves to perform the verification [8], [10], [11], [13], [14], [15]. However, these approaches are not suitable for CPSS, since verifying the integrity of outsourced data requires additional online burden and computation overhead, which are too expensive for CPSS users. In CPSS, the overhead of using outsourced data should be minimized as much as possible, such that a CPSS user is able to retrieve and use the data without performing too many operations.

To ensure the integrity of the outsourced data and release the CPSS users' computation and online burden, it is an easy and affordable way in practice to employ an external and independent auditor to periodically verify the data integrity on behalf of CPSS users, which is called "public verification" [5], [6], [16], [17]. The existing public verification schemes do not fit CPSS due to the following reasons. First, these schemes assume that the auditor is honest and cannot be corrupted. But this is a very strong assumption as corruption of auditors could happen in practice. Second, most existing public verification mechanisms require the auditor to manage the users' certificates to choose the correct public keys for verification. Nevertheless, certificate management, which includes revocation, storage, distribution and verification, is very costly and cumbersome for applying in CPSS. In short, it is important to study how to verify the integrity of outsourced data while addressing the above problems.

Recently, Wang et al. [17] proposed the first certificateless public integrity verification scheme (CLPV in short). In the scheme, auditor does not need to manage users' certificates. Meanwhile, the CLPV can thwart two types of adversaries, noted as Type I adversary and Type II adversary, respectively. With the CLPV, a Type I adversary, who does not have access to the master-key issued by a Key Generation Center (KGC), but is capable of replacing the public key of any entity with a value of his choice, could not forge legitimate signatures. While a Type II adversary, who is able to access to the master-key but cannot perform public key replacement, could not generate valid signatures either. However, there is another case that is not considered in the CLPV scheme: when an adversary does not have access to the master-key but has the ability to obtain the secret key generated by the user himself, he could impersonate the user and forge valid signatures. As a result, the CLPV faces a serious security threat. We will show an attack example in Appendix A. Moreover, in the CLPV, the auditor is assumed as honest, and hence the

CLPV is vulnerability in the case that auditors are malicious. Therefore, the CLPV is hard to be widely applied in CPSS.

More recently, the first and the only attempt to verify the data integrity against malicious auditor was studied by Armknecht et al. [18] in 2014. An instantiation was proposed and named as Fortress. The Fortress has a great advantage in terms of computation efficiency for verification. However, in the Fortress, to verify the integrity of a set of cloud-stored data, the auditor has to download the entire data set, and for each data block, the auditor has to generate a signature, and finally uploads these signatures to the cloud server. This model implies that the communication cost between the auditor and the cloud server is proportional to the size of the processed data. Besides, in the Fortress, after generating signatures, the auditor needs to convince the user that the signatures computed by the auditor are correct. This procedure uses a zero-knowledge-proof technique and resorts to the root certificate of a certification authority. In other words, the auditor in the Fortress faces the certificate management problem. These barriers have become a bottleneck for the Fortress to be used in CPSS.

In this paper, we propose a secure certificateless public integrity verification scheme (SCLPV in short) against malicious auditors for cloud storage in CPSS. In the SCLPV scheme, a public auditor is able to verify the integrity of outsourced data without retrieving the entire data set and managing the user's certificate. Meanwhile, to fight against malicious auditors, the SCLPV requires CPSS users periodically check their auditors' behavior. Furthermore, we extensively analyze the performance of the SCLPV scheme and demonstrate that the SCLPV scheme is efficient and practical. Compared with the Fortress by experiments and theory, our SCLPV scheme is more practical in terms of communication cost. Furthermore, we also compare the SCLPV scheme with two existing schemes [17], [9], which are vulnerable against malicious auditors. According to the comparison results, we can see that SCLPV provides much more security guarantees while keeping a decent efficiency. Specifically, the contributions of this work are as follows.

- We point out the security vulnerability existing in the CLPV scheme [17]. With the vulnerability, once an adversary obtains a part of the user's signing key, he can impersonate the user and forge legitimate signatures.
- We propose the novel SCLPV scheme that enables a third-party auditor to verify the integrity of cloud-stored data without managing

certificates in CPSS. We prove that the SCLPV can achieve secure certificateless public verification, work against malicious auditors and prevent collusion between any two involved parties through a formal and strict security proof. To the best of our knowledge, the SCLPV is the first mechanism that can resist malicious auditor and support certificateless public verification for cloud storage in CPSS with full proofs of security against arbitrary adversaries in the strongest model, that of Shacham et al. [8], [9].

The rest of the paper is organized as follows. We discuss the related work in Section 2. In Section 3, we present the system model, definition, security model and preliminaries of our work. Then we provide the detailed description of the SCLPV scheme in Section 4. In Section 5, we evaluate our SCLPV scheme in terms of correctness, security and performance. We offer further discussion in Section 6. Finally, we give the concluding remark of the whole paper in Section 7.

2. Related work

2.1. CPS and CPSS

Cyber-physical system (CPS) integrates the cyber world where information is exchanged and transformed, and the physical world we live in. A good many of researchers have studied on CPS in the last decade [19]. A CPS includes physical dynamics, sensors, communication network and computation controllers. The controllers are connected via the communication network to control the physical dynamics and sensors and convey the measurement on the system state.

Recently, researchers added human factors into CPS and presented a new system – cyber-physical-social system (CPSS)[20]. Here, human factors includes human knowledge, mental capabilities, sociocultural elements and so on. Some aspects, such as command and control [20], recommendation [3] and computing [21] in CPSS have been studied. However, there is few study on data integrity in CPSS up to now.

2.2. Public verification

To ensure the integrity of data stored on an untrusted server, Juels et al. [10] proposed "proof of retrievability" (POR) technique. However, they do not consider the public verification. Ateniese et al. [11] first considered public

verification in their "provable data possession" (PDP) model. Shacham et al. [8] proposed the first compact POR scheme that support public verification. Following up the Shacham et al.'s work, several public verification schemes have been proposed [4], [5], [6], [9]. These schemes build upon a homomorphic signature technique and resort to a fully trusted third-party auditor to verify the integrity of cloud-stored data on behalf of the cloud user without downloading the entire data set. With these schemes, the auditor needs to manage the cloud user's certificate to choose the correct public parameters for verification. To remove the certificate management problem, Wang et al. [17] proposed the first certificateless public integrity auditing scheme. With the scheme, the auditor is still considered as an honest entity and does not need to manage the user's certificate.

All the aforementioned schemes mainly focus on verifying the integrity of cloud-stored data by resorting to a fully trusted auditor. However, it is fair to say that, up to [18], how to resist against a malicious auditor is an open problem.

In this paper, we will design a certificateless public integrity verification mechanism against malicious auditors for CPSS. Our approach can achieve much lower communication cost compared with Fortress [18]. In addition, our approach also can address the security vulnerability existing in the CLPV [17].

3. Formulation and preliminaries

3.1. Modelling SCLPV

In CPSS, the users collect data from cyberspace and physical space, and outsource their data to cloud server. In our SCLPV scheme, we mainly focus on how to efficiently verify the integrity of outsourced data in CPSS. Therefore, we omit the data collection in our SCLPV scheme. There exist four different entities in the SCLPV, as shown in Fig. 1.

1. User. The user is the data owner, who needs flexibly to access his data in the cloud. He has an appeal that checking the integrity of his cloud-stored data. However, due to limited condition, the user cannot check the data integrity by himself. Therefore, the user will resort to a third-party auditor (TPA) to perform the verification work. From the cloud server perspective, the user may upload false information to it.

2. Cloud Server. The cloud server, which is managed by the cloud service provider, provides cloud storage services, it has not only significant storage

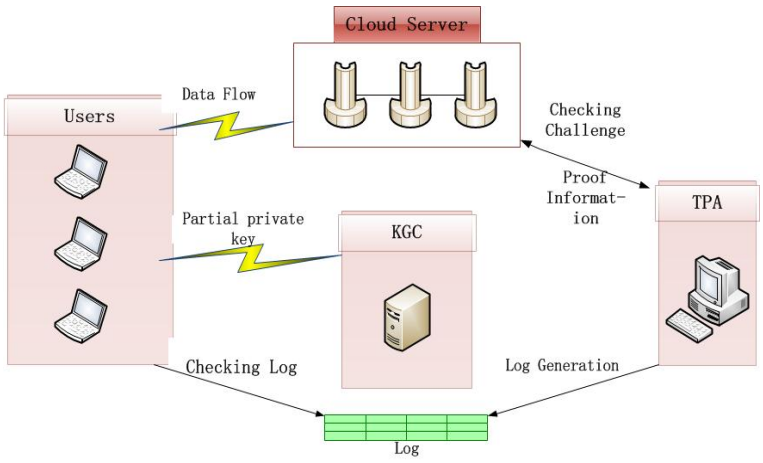

Figure 1: System Model

capability, but also a massive amount of computing power. However, the cloud server may be aborted. This means that the cloud server will follow the prescribed routine except for profit-seeking. We want to further stress that for the cloud service provider, "profit-seeking" means that either reducing the storage overheads or hiding data loss incidents so as to maintain his reputation without increasing the storage overheads.

3. Third-party Auditor (TPA). The TPA, who has expertise and capabilities that the user does not, can verify the integrity of the cloud-stored data when needed. It feeds back the verification result to the user and the cloud server. In the SCLPV, we consider that the TPA may be aborted similar to the cloud server. It means that the TPA may hide an incident that the cloud-stored data has been corrupted to the user. Meanwhile, the TPA may fabricate a verification result to circumvent the cloud server. Therefore, the TPA should be liable for its verification result and it can prove that all of verification work has been executed correctly.

4. Key Generation Center (KGC). The KGC is controlled by an authority, and therefore trusted by the users, the cloud server and the TPA. The KGC generates a system parameter set. And based on the corresponding identity, the KGC further generates a partial private key of the user.

A formal definition of the SCLPV is given in **Definition 1**.

Definition 1: The SCLPV consists of six algorithms, **Setup**, **Store**, **ChalGen**, **ProGen**, **VerPro**, **CheckLog**.

Setup. This randomized algorithm takes the security parameter as input

and outputs a system parameter $SysPara$, the user's signing key $ssk_{\mathcal{U}}$ and the corresponding verification key $spk_{\mathcal{U}}$. The private key is made up of two parts: partial-private key and secret key. The partial-private key is generated by the KGC, and the secret key is generated by the user himself. For simplicity, we consider that any involved entity possesses the public keys of the user at the end of this algorithm.

Store. This randomized algorithm takes the secret parameters of the entities and a file F as input, outputs a file $\widetilde{F}$ to be stored in the cloud. $\widetilde{F}$ includes the file F consisted of n blocks, a file tag τ and signatures of all the data blocks $\{\sigma_i\}_{i\in[1,n]}$. After receiving $\widetilde{F}$ from the user, the cloud server needs to confirm that $\widetilde{F}$ has been uploaded correctly. What we want to emphasize here is that at the end of this algorithm, some necessary parameters and agreements have been determined and established. For example, with the security parameter, the user and the TPA determine the period of verification and the number of data blocks to be verified.

ChalGen. This randomized algorithm takes the security parameter and current time as input and outputs a challenging message. The TPA generates and issues the challenging message to the cloud server to verify the integrity of $\widetilde{F}$. Note that the TPA may be corrupted, the challenging message cannot be determined by its own. Meanwhile, there may be collusion between any two involved parties, the challenging message cannot depend on interactive sampling among two parties. Therefore, the challenging message is determined by a time-based paseudorandom source, which guarantees the randomness and unbiasedness of the challenging message.

ProGen. This randomized algorithm takes the challenging message and $\widetilde{F}$ as input and outputs a proof information. The proof information is generated by the cloud server to prove that $\widetilde{F}$ is still correctly stored and maintained.

VerPro. This deterministic algorithm takes the proof information and public parameters as input and outputs a result of the TPA's verification and a log file Λ. The TPA verifies the validity of the proof information. If fails, the TPA informs the user that his cloud-stored data may be corrupted. Otherwise, for each verification, the TPA needs to create an entry and store it into a log file. This enables the user to audit the TPA.

CheckLog. This deterministic algorithm takes the log file as input and outputs a result of the user's verification. The user checks the validity of Λ. The user considers that the TPA has executed all the algorithms correctly

only if Λ is valid. Here, we want to further stress that the periodicity the user runs **CheckLog** is much longer than the periodicity the TPA runs **VerPro**.

3.2. Security model

The security model in this paper is constructed based on the model in [8], [9], [22] and [18]. In our scheme, we colligate the security of data integrity checking scheme and the security of certificateless aggregate signature. To prove that the signature scheme used for $\{\sigma_i\}_{i\in[1,n]}$ in the SCLPV is existentially unforgeable, we first modify the series of games proposed by [9] to satisfy the certificateless aggregate signature scheme on additive group, and then we prove it. Here, due to space limitations, we omit the formal security definition, and see Section 5 for more details. **Definition 2** gives the security goals that the SCLPV should achieve.

Definition 2: The SCLPV can be said to be secure if and only if all of the following conditions hold:

1. There is no any polynomial time algorithm that can forge a valid signature with nonnegligible probability without the complete and correct signing key.
2. There is no adversary that can deceive the TPA with non-negligible probability.
3. Any involved party cannot deceive the rest parties.
4. Collusion between any two involved parties cannot deceive the rest party.

3.3. Design goals

To enable certificateless public verification for cloud storage under the aforementioned model, the SCLPV should achieve the following objectives.

1. Public certificateless verification: to allow a TPA to verify the integrity of cloud-stored data, without managing the user's certificates.
2. Security: to achieve the security goals presented by **Definition 2**.
3. Efficiency: to allow a TPA to execute verification with minimum communication and computation overheads.

3.4. Bilinear maps

The SCLPV is realized in an efficient bilinear map [23]. Let G_1 be an additive group and G_2 be a multiplicative group, G_1 and G_2 have the same prime order q. A bilinear map e: $G_1 \times G_1 \rightarrow G_2$ has the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1,\ a, b \in Z_q^*$.
2. Non-degeneracy: for $P, Q \in G_1$ and $P \neq Q$, $e(P, Q) \neq 1$.
3. There exists an efficient computable algorithm for computing e.

Computation Diffe-Hellman (CDH) problem in G_1: given G_1 and one of its generator P, for any unknown $a, b \in Z_q^*$, given aP and bP; compute abP.

3.5. Bitcoin

Bitcoin is introduced in [24]. In fact, Bitcoin has an attractive property. That is, given a determinate time t, if t is a past or current time, we can easily find a Bitcoin block which is generated in the nearest time of t; but if t is a future time, the Bitcoin block, which is generated in t, is unpredictable. Here, we denote the hash of Bitcoin block which is generated in a past time t as Bl_t. Since Bitcoin has this property, we can consider Bitcoin as a time-based pseudo-randomness source. The output of this source can be computed when the input of the source is a past/current time, otherwise, the output is unpredictable.

We take Bl_t as a seed of pseudorandom bit generator (PRBG) $GetRandomness()$, and get a parameter θ as:

$$\theta = GetRandomness(Bl_t)$$

θ is used to generate a challenging message in the **POR** algorithm.

4. Proposed SCLPV scheme

In this Section, we propose the SCLPV scheme. A user $\mathcal{U}$, a cloud server $\mathcal{C}$ and a third-party auditor (TPA) are involved in the SCLPV scheme.

The specifications of the algorithms are as follows.

Setup. This algorithm consists of the following three steps.

step-1: The KGC generates the system parameters as follows.

- With a security parameter ℓ, choose G_1 and G_2, where G_1 is an additive group generated by a generator P with order q, G_2 is a multiplicative group with the same order. The bilinear map is $e : G_1 \times G_1 \to G_2$.
- Choose a random $\lambda \in Z_q$ as the master key, and compute $P_M = \lambda P$.
- Choose five hash functions $H()$, $H_1()$, $H_2()$, $H_3()$, $H_4()$, where $H_1() \sim H_4() : \{0,1\}^* \to G_1$ are cryptographic hash functions and $H() : \{0,1\}^* \to G_1$ is a secure map-to-point hash function.

The system parameter list is $SysPara = \{G, G_2, e, P, P_M, H(), H_1() \sim H_4()\}$.

step-2: The KGC calculates the partial private key for $\mathcal{U}$ using $\mathcal{U}$'s identity $ID_{\mathcal{U}}$ as follows:

- Compute $Q_{\mathcal{U},0} = H_1(ID_{\mathcal{U}}, 0)$ and $Q_{\mathcal{U},1} = H_1(ID_{\mathcal{U}}, 1)$.
- Compute $D_{\mathcal{U},0} = \lambda Q_{\mathcal{U},0}$ and $D_{\mathcal{U},1} = \lambda Q_{\mathcal{U},1}$.

step-3: $\mathcal{U}$ chooses a random $x_{\mathcal{U}} \in Z_q^*$, and computes $pk_{\mathcal{U}} = x_{\mathcal{U}} P$.

$\mathcal{U}$'s signing key is $ssk_{\mathcal{U}} = \{x_{\mathcal{U}}, D_{\mathcal{U},0}, D_{\mathcal{U},1}\}$ and the corresponding verification key is $spk_{\mathcal{U}} = \{pk_{\mathcal{U}}, Q_{\mathcal{U},0}, Q_{\mathcal{U},1}\}$.

Store. By utilizing an information dispersal algorithm (i.e. erasure code [30]), $\mathcal{U}$ transforms his data F into n blocks: $F = \{m_i\}_{1 \le i \le n}$. $\mathcal{U}$ chooses a random element $name$ for file naming and computes the file tag as $\tau = name||Sig_{ssk_{\mathcal{U}}}(name)$, where $Sig(\cdot)$ is a certificateless signature algorithm.

Then, $\mathcal{U}$ generates a signature for each data block $m_i, i \in [1, n]$ as follows:

- Choose a one-time-use number Δ.
- For each $i \in [1, n]$, choose a random $r_i \in Z_q^*$ and compute $R_i = r_i g$.
- Compute three hash values $T = H_2(\Delta)$, $V = H_3(\Delta)$ and $W = H_4(\Delta)$.
- Compute $S_i = m_i(D_{\mathcal{U},0} + x_{\mathcal{U}} V) + H(i||name)(D_{\mathcal{U},1} + x_{\mathcal{U}} W) + r_i T$
- Output $\sigma_i = \{R_i, S_i\}$ as the signature on m_i

Now, $\mathcal{U}$ outsources $\widetilde{F} = \{F = \{m_i\}_{i \in [1,n]}, \phi = \{\sigma_i\}_{i \in [1,n]}, \tau, \Delta\}$ into $\mathcal{C}$. After $\mathcal{U}$ uploads $\widetilde{F}$ to $\mathcal{C}$, $\mathcal{C}$ needs to confirm that $\widetilde{F}$ has been uploaded correctly by verifying the following equation

$$\begin{aligned} e(\sum_{i=1}^{n} S_i, P) \stackrel{?}{=} & \; e(\sum_{i=1}^{n}(m_i Q_{\mathcal{U},0} + H(i||name) Q_{\mathcal{U},1}), P_M) \\ & \times e(\sum_{i=1}^{n} m_i V + \sum_{i=1}^{n} H(i||name) W, pk_{\mathcal{U}}) \\ & \times e(T, \sum_{i=1}^{n} R_i) \end{aligned}$$

where $Q_{\mathcal{U},0} = H_1(ID_{\mathcal{U}}, 0)$ and $Q_{\mathcal{U},1} = H_1(ID_{\mathcal{U}}, 1)$. If the equation holds, $\mathcal{C}$ accepts $\widetilde{F}$.

ChalGen. The TPA generates a challenging message as follows.

- Acquire Bl_t based on the current time t.
- Initialize the PRBG $GetRandomness$ as $\theta = GetRandomness(Bl_t)$.
- Pick a random subset I of the set $\{1, ..., n\}$ on θ and ℓ.
- For each $i \in I$, choose a random $v_i \in Z_p$ (p is a much smaller prime than q).

Then, the TPA sends the challenging message $\{(i, v_i)\}_{i \in I}$ to $\mathcal{C}$.

ProGen. With the challenging message, $\mathcal{C}$ calculates:

$$S = \sum_{i \in I} v_i S_i,\ R = \sum_{i \in I} v_i R_i,\ \mu = \sum_{i \in I} v_i m_i$$

$\mathcal{C}$ takes $proof = \{S, R, \mu, \Delta\}$ as the proof information and sends it to the TPA.

VerPro. With the proof information, The TPA first retrieves the file tag τ, and checks its validity via $spk_{\mathcal{U}}$. Then, the TPA verifies the following equation:

$$\begin{aligned} e(S, P) \ \stackrel{?}{=}\ & e((\mu Q_{\mathcal{U},0} + \sum_{i \in I} v_i H(i||name) Q_{\mathcal{U},1}), P_M) \\ & \times e((\mu H_3(\Delta) + \sum_{i \in I} v_i H(i||name) H_4(\Delta)), pk_{\mathcal{U}}) \\ & \times e(H_2(\Delta), R) \end{aligned} \tag{1}$$

If the equation does not hold, the TPA takes the verification result as $Reject$.

If the equation holds, the TPA takes the verification result as $Accept$. Next, the TPA creates an entry as follows:

$$(Bl_t, S, R, \mu, \Delta)$$

Finally, the TPA stores the entry into a log file Λ as shown in Table 1.

CheckLog. $\mathcal{U}$ checks the validity of Λ as follows.

- Pick a random subset B of indices of Bitcoin blocks.

Table 1: the log file Λ

Δ			
$Bl_t^{(1)}$	$S^{(1)}$	$R^{(1)}$	$\mu^{(1)}$
$Bl_t^{(2)}$	$S^{(2)}$	$R^{(2)}$	$\mu^{(2)}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$Bl_t^{(l)}$	$S^{(l)}$	$R^{(l)}$	$\mu^{(l)}$

- Generate a set of challenging messages $I^{(B)} = \{\{i^{(1)}, v_i^{(1)}\}_{i\in[I^{(1)}]}, ..., \{i^{(b)}, v_i^{(b)}\}_{i\in[I^{(b)}]}\}$, where b is the size of subset B.
- Send B to the TPA and receive $\Delta, S^{(B)}, R^{(B)}, \mu^{(B)}$ from the TPA, where

$$S^{(B)} = \sum_{j\in I^{(B)}} S^{(j)}$$
$$R^{(B)} = \sum_{j\in I^{(B)}} R^{(j)}$$
$$\mu^B = \sum_{j\in I^{(B)}} \mu^{(j)} = \sum_{j\in I^{(B)}} \sum_{i\in I} v_i^{(j)} m_i^{(j)}$$

- Verify

$$\begin{aligned} e(S^{(B)}, P) \stackrel{?}{=} & \; e((\sum_{j\in I^{(B)}} \mu^{(j)} Q_{\mathcal{U},0} + \\ & \sum_{j\in I^{(B)}} \sum_{i\in I} v_i^{(j)} H(i||name)^{(j)} Q_{\mathcal{U},1}), P_M) \\ & \times e((\sum_{j\in I^{(B)}} \mu^{(j)} H_3(\Delta) + \\ & \sum_{j\in I^{(B)}} \sum_{i\in I} v_i^{(j)} H(i||name)^{(j)} H_4(\Delta)), pk_{\mathcal{U}}) \\ & \times e(H_2(\Delta), \sum_{j\in I^{(B)}} R^{(j)}) \end{aligned} \quad (2)$$

If the verification fails, $\mathcal{U}$ can consider that his cloud-stored data is corrupted, and either/both the TPA or/and $\mathcal{C}$ is/are malicious. When this occurs, $\mathcal{U}$ takes verification result as *Reject*. Otherwise, $\mathcal{U}$ takes verification result as *Accept*.

5. Evaluation

5.1. Correctness proof

For the equation (1), the correctness is proved as follows:

$$
\begin{aligned}
e(S,P) &= e(\sum_{i\in I} v_i S_i, P) \\
&= e(\sum_{i\in I} v_i(m_i(D_{\mathcal{U},0}+x_{\mathcal{U}}V)+H(i||name)(D_{\mathcal{U},1}+x_{\mathcal{U}}W)+r_iT), P) \\
&= e(\sum_{i\in I} v_i m_i D_{\mathcal{U},0}, P)\cdot e(\sum_{i\in I} v_i m_i x_{\mathcal{U}}V, P)\cdot e(\sum_{i\in I} v_i H(i||name)D_{\mathcal{U},1}, P) \\
&\quad \times e(\sum_{i\in I} v_i H(i||name)x_{\mathcal{U}}W, P)\cdot e(\sum_{i\in I} v_i r_i T, P) \\
&= e(\sum_{i\in I} v_i m_i \lambda Q_{\mathcal{U},0}, P)\cdot e(\sum_{i\in I} v_i m_i H_3(\Delta), x_{\mathcal{U}}P) \\
&\quad \times e(\sum_{i\in I} v_i H(i||name)\lambda Q_{\mathcal{U},1}, P)\cdot e(\sum_{i\in I} v_i H(i||name)H_4(\Delta), x_{\mathcal{U}}P) \\
&\quad \times e(T, \sum_{i\in I} v_i r_i P) \\
&= e(\mu Q_{\mathcal{U},0}, \lambda P)\cdot e(\mu H_3(\Delta), pk_{\mathcal{U}})\cdot e(\sum_{i\in I} v_i H(i||name)Q_{\mathcal{U},1}, \lambda P) \\
&\quad \times e(\sum_{i\in I} v_i H(i||name)H_4(\Delta), x_{\mathcal{U}}P)\cdot e(T,R) \\
&= e((\mu H_1(ID_{\mathcal{U}},0)+\sum_{i\in I} v_i H(i||name)H_1(ID_{\mathcal{U}},1)), P_M) \\
&\quad \times e((\mu H_3(\Delta)+\sum_{i\in I} v_i H(i||name)H_4(\Delta)), pk_{\mathcal{U}}) \\
&\quad \times e(H_2(\Delta), R)
\end{aligned}
$$

And for the equation (2), the correctness is shown as follows:

$$
\begin{aligned}
e(S^{(B)},P) &= e(\sum_{j\in I^{(B)}} S^{(j)}, P) \\
&= e(\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)} S_i^{(j)}, P)
\end{aligned}
$$

$$
\begin{aligned}
&= e(\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}(m_i^{(j)}(\lambda Q_{\mathcal{U},0}+x_{\mathcal{U}}V)+H(i||name)^{(j)}(\lambda Q_{\mathcal{U},1}+x_{\mathcal{U}}W)+r_i^{(j)}T),P)\\
&= e(\sum_{j\in I^{(B)}}\mu^{(j)}Q_{\mathcal{U},0},\lambda P)e(\sum_{j\in I^{(B)}}\mu^{(j)}V,x_{\mathcal{U}}P)e(\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}H(i||name)^{(j)}Q_{\mathcal{U},1},\lambda P)\\
&\quad \times e(\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}H(i||name)^{(j)}W,x_{\mathcal{U}}P)e(T,\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}r_i^{(j)}P)\\
&= e((\sum_{j\in I^{(B)}}\mu^{(j)}H_1(ID_{\mathcal{U}},0)+\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}H(i||name)^{(j)}H_1(ID_{\mathcal{U}},1)),P_M)\\
&\quad \times e((\sum_{j\in I^{(B)}}\mu^{(j)}H_3(\Delta)+\sum_{j\in I^{(B)}}\sum_{i\in I} v_i^{(j)}H(i||name)^{(j)}H_4(\Delta)),pk_{\mathcal{U}})\\
&\quad \times e(H_2(\Delta),\sum_{j\in I^{(B)}}R^{(j)})
\end{aligned}
$$

5.2. Security analysis

In this Section, we provide a formal and strict security proof of SCLPV scheme. More precisely, according to the security requirements presented in Section 3.2, we summarize four theorems and we will prove that our SCLPV scheme satisfies these four theorems one by one.

Theorem 1: For an internal/external adversary, it is computationally infeasible to generate a forge signature of any involved entities with our S-CLPV scheme.

Proof. In SCLPV, the certificateless aggregate signature (here, we name it ICL-ASS) is used to generate the signatures of data blocks. To prove the security of ICL-ASS, we first introduce a lemma which is presented by [26].

Lemma 1: Two signature schemes are called strongly equivalent if the signature of the first scheme can be transformed into signature of the second scheme and vice versa, without knowledge of private key.

Now, we will reduce the security of ICL-ASS to the security of an existing certificateless aggregate signature [22] (short for CL-ASS). We remark that the main difference between the ICL-ASS and the CL-ASS is the signature generation. In the CL-ASS

$$S_i = D_{\mathcal{U},0}+x_{\mathcal{U}}V+h_i(D_{\mathcal{U},1}+x_{\mathcal{U}}W)+r_iT.$$

where $h_i = H_5(m_i||\Delta||ID_{\mathcal{U}}||pk_{\mathcal{U}})$ and $H_5 : \{0,1\}^* \to Z_q^*$ is a cryptographic hash function.

But in the ICL-ASS

$$S_i = m_i(D_{\mathcal{U},0} + x_{\mathcal{U}}V) + H(i||name)(D_{\mathcal{U},1} + x_{\mathcal{U}}W) + r_iT.$$

By **Lemma 1**, if the ICL-ASS is strongly equivalent to the CL-ASS, then we get that the security of ICL-ASS is the same as that of CL-ASS.

Suppose (m, R_i, S_i) is a valid CL-ASS signature, and $(m, R_i, \tilde{S}_i)$ is a valid ICL-ASS signature. Note that for the CL-ASS, we can get

$$e(S_i, P) = e(Q_{\mathcal{U},0} + h_iQ_{\mathcal{U},1}, P_M)e(V + h_iW, pk_{\mathcal{U}})e(T, R_i) \tag{3}$$

and for the ICL-ASS, we can get

$$e(\tilde{S}_i, P) = e(mQ_{\mathcal{U},0} + H_iQ_{\mathcal{U},1}, P_M)e(mV + H_iW, pk_{\mathcal{U}})e(T, R_i) \tag{4}$$

where $H_i = H(i||name)$. With the equation (4), we further get

$$e(mQ_{\mathcal{U},0}, P_M) = \frac{e(\tilde{S}_i, P)}{e(H_iQ_{\mathcal{U},1}, P_M)e(mV + H_iW, pk_{\mathcal{U}})e(T, R_i)}$$

according to the properties of bilinear map, we can get

$$e(Q_{\mathcal{U},0}, P_M)^m = \frac{e(\tilde{S}_i, P)}{e(H_iQ_{\mathcal{U},1}, P_M)e(mV + H_iW, pk_{\mathcal{U}})e(T, R_i)}$$

that is

$$m = \log_{e(Q_{\mathcal{U},0}, P_M)} \frac{e(\tilde{S}_i, P)}{e(H_iQ_{\mathcal{U},1}, P_M)e(mV + H_iW, pk_{\mathcal{U}})e(T, R_i)} \tag{5}$$

Accordingly,

$$h_i = H_5((\log_{e(Q_{\mathcal{U},0}, P_M)} \frac{e(\tilde{S}_i, P)}{e_1e_2e_3})||\Delta||ID||pk_{\mathcal{U}}) \tag{6}$$

where $e_1 = e(H_iQ_{\mathcal{U},1}, P_M)$, $e_2 = e(mV + H_iW, pk_{\mathcal{U}})$ and $e_3 = e(T, R_i)$.

In consideration of clear expression, here, we introduce an operational symbol $e_a^{-1}(b, c)$ which means that given a, b, c, bilinear map e, and $c = e(a, b)$, so that $a = e_a^{-1}(b, c)$.

Because of equation (5), we can get

$$S_i = e_{S_i}^{-1}((e(Q_{\mathcal{U},0} + h_i Q_{\mathcal{U},1}, P_M) \cdot e(V + h_i W, pk_{\mathcal{U}}) \cdot e(T, R_i)), P) \quad (7)$$

Then, substituting equation (6) into (7), we obtain equation (8) as follows:

$$\begin{aligned} S_i = e_{S_i}^{-1} \quad ((\quad & e(Q_{\mathcal{U},0} + H_5((\log_{e(Q_{\mathcal{U},0},P_M)} \frac{e(\tilde{S}_i, P)}{e_1 e_2 e_3})||\Delta||ID||pk_{\mathcal{U}})Q_{\mathcal{U},1}, P_M) \\ & \times e(V + H_5((\log_{e(Q_{\mathcal{U},0},P_M)} \frac{e(\tilde{S}_i, P)}{e_1 e_2 e_3})||\Delta||ID||pk_{\mathcal{U}})W, pk_{\mathcal{U}}) \\ & \times e(T, R_i)), P) \end{aligned} \quad (8)$$

From equation (3) to equation (8), we can see that our ICL-ASS can be transformed into CL-ASS, i.e. ICL-ASS $\Rightarrow$ CL-ASS.

By the similar arguments as for ICL-ASS $\Rightarrow$ CL-ASS, the inverse process CL-ASS $\Rightarrow$ ICL-ASS is still feasible, here we omit it. This completes the proof of the **Theorem 1**.

Theorem 2. If the cloud server passes the TPA's verification, it must possess truly the specified data intact.

Proof. We will prove the theorem as a series of games with interleaved analysis. The series of games were first presented by [8] and [9], here, we modify it to satisfy the ICL-ASS on additive group. Here, we want to stress that SCLPV is the first provable secure certificateless public verification scheme under the Shacham et al.'s security model [9].

Game 0. This game is simply the challenge game defined in Section 4.

Game 1. This game is the same as **Game 0**, with the exception of one difference. The adversary is trained to be capable of forging a valid tag τ. The challenger keeps a list of all signed tags which is generated in **Store**. If the adversary ever submits a tag τ during **VerPro** that is a valid signature under $ssk_{\mathcal{U}}$ but is not signed by the challenger, the challenger declares failure and aborts.

Analysis. Clearly, in **Game 1**, if the challenger aborts with nonnegligible probability which is caused by the adversary, we can resort to the adversary to construct a forger against the certificateless signature scheme

used to generate τ. Otherwise, for the adversary, there is no difference between in **Game 0** and in **Game 1**. Since the first thing the TPA does after receiving the proof information, is to check the validity of τ. If failed, the TPA rejects immediately and aborts. Note that if τ with a valid certificateless signature, it might be either generated by the challenger, or forged by the adversary. However, if the latter case occurs, it is in conflict with the definition defined in **Game 1**. Therefore, we can sure that the cloud-stored data, which is issued by the cloud server and will be audited by the TPA, is the "valid" data. Here, "valid" means consistency between the TPA's challenge and the cloud server's proof.

Game 2. This game is the same as **Game 1**, with the exception of one difference. The adversary is trained to be capable of forging a part of the proof information. The challenger keeps a list of its responses of sign which is queried by the adversary. The challenger observes each instance of **ProGen** with the adversary. If in any instances, compared with the expected proof information $\{S, R, \mu, \Delta\}$, the proof information of the adversary has a difference in μ, but the adversary is successful, the challenger declares failure and aborts.

Analysis. As described in **Game 2**, the adversary responds $\{S, R, \mu', \Delta\}$ to the challenger. We now show that if the adversary causes the challenger in **Game 2** to abort with nonnegligible probability, how can we construct a simulator that solve the CDH problem.

For the expected proof information $\{S, R, \mu, \Delta\}$, by the correctness of the scheme, we can get

$$\begin{aligned} e(S,P) &= e((\mu H_1(ID_{\mathcal{U}},0)+\sum_{i\in I} v_i H(i||name)H_1(ID_{\mathcal{U}},1)), P_M) \\ &\quad \times e((\mu H_3(\Delta)+\sum_{i\in I} v_i H(i||name)H_4(\Delta)), pk_{\mathcal{U}}) \\ &\quad \times e(H_2(\Delta), R) \end{aligned}$$

and for the adversary's response $\{S, R, \mu', \Delta\}$, we also get

$$\begin{aligned} e(S,P) &= e((\mu' H_1(ID_{\mathcal{U}},0)+\sum_{i\in I} v_i H(i||name)H_1(ID_{\mathcal{U}},1)), P_M) \\ &\quad \times e((\mu' H_3(\Delta)+\sum_{i\in I} v_i H(i||name)H_4(\Delta)), pk_{\mathcal{U}}) \end{aligned}$$

$$\times e(H_2(\Delta), R)$$

Note that $\mu \neq \mu'$, and if we assume $\bar{\mu} = \mu - \mu'$, so that $\bar{\mu} \neq 0$. There are two cases in this situation.
1. $\mu H_1(ID_{\mathcal{U}}, 0) = \mu' H_1(ID_{\mathcal{U}}, 0)$ and $\mu H_3(\Delta) = \mu' H_3(\Delta)$.
2. $\mu H_1(ID_{\mathcal{U}}, 0) \neq \mu' H_1(ID_{\mathcal{U}}, 0)$ and $\mu H_3(\Delta) \neq \mu' H_3(\Delta)$.

For the first case, clearly, we can learn that

$$\mu H_1(ID_{\mathcal{U}}, 0) = \mu' H_1(ID_{\mathcal{U}}, 0), \quad \bar{\mu} H_1(ID_{\mathcal{U}}, 0) = 0$$

Because G_1 is an additive cyclic group, for two random elements $A, B \in G_1$, there exists $\chi \in Z_q$ so that $A = \chi B$. Similarly, for an arbitrary $H_1(ID_{\mathcal{U}}, 0)$, we can represent it as $H_1(ID_{\mathcal{U}}, 0) = \zeta A + \xi B$. Then, we learn that

$$0 = \bar{\mu} H_1(ID_{\mathcal{U}}, 0) = \bar{\mu}(\zeta A + \xi B)$$

Obviously, we can find a solution of CDH problem with a probability of $1 - (1/p)$. In particularly, given B, $A = \chi B \in G_1$, we can compute

$$A = -\frac{\xi}{\zeta} B, \quad \chi = -\frac{\xi}{\zeta}$$

And for the second case, we can get:

$$e(\mu H_1(ID_{\mathcal{U}}, 0), P_M) e(\mu H_3(\Delta), pk_{\mathcal{U}}) = e(\mu' H_1(ID_{\mathcal{U}}, 0), P_M) e(\mu' H_3(\Delta), pk_{\mathcal{U}})$$

Rearranging terms yields

$$e(\mu H_1(ID_{\mathcal{U}}, 0)\lambda + \mu H_3(\Delta) x_{\mathcal{U}}, P) = e(\mu' H_1(ID_{\mathcal{U}}, 0)\lambda + \mu' H_3(\Delta) x_{\mathcal{U}}, P)$$

that is

$$\mu(H_1(ID_{\mathcal{U}}, 0)\lambda + H_3(\Delta) x_{\mathcal{U}}) = \mu'(H_1(ID_{\mathcal{U}}, 0)\lambda + H_3(\Delta) x_{\mathcal{U}})$$

Here, we set $\omega = H_1(ID_{\mathcal{U}}, 0)\lambda + H_3(\Delta) x_{\mathcal{U}}$, and for an arbitrary ω, we can represent it as $\omega = \varpi C + \varrho D$, where C and D are random elements of G_1. Similarly, there exists $\chi \in Z_q$ so that $C = \chi D$. Next, we learn that

$$0 = \bar{\mu}\omega = \bar{\mu}(\varpi C + \varrho D)$$

Obviously, we can find a solution of CDH problem with a probability of $1-(1/p)$. More precisely, given D, $C=\chi D \in G_1$, we can compute

$$C=-\frac{\varrho}{\varpi}D \qquad \chi=-\frac{\varrho}{\varpi}$$

Therefore, if there is a nonnegligible probability that the adversary causes the challenger abort, we can construct a simulator that use the adversary to solve the CDH problem as needed.

Game 3. This game is the same as **Game 2**, with the exception of one difference. The adversary is trained to be capable of forging a valid aggregate authenticated value. The challenger keeps a list of its responses of sign which is queried by the adversary. The challenger observes each instance of **ProGen** with the adversary. If in any instances, the aggregate signature of the adversary $(S', R') \neq (S, R)$, where $(S=\sum_{i\in I} v_i S_i, R=\sum_{i\in I} v_i R_i)$ is the expected aggregate signature, but the adversary is successful, the challenger declares failure and aborts.

Analysis. After receiving the challenging message, the response of adversary is $\{S', R', \mu', \Delta\}$, while the expected response is $\{S, R, \mu, \Delta\}$. By the correctness of the scheme, we can learn that

$$e(S,P)=e((\mu Q_{\mathcal{U},0}+\sum_{i\in I} v_i H_i Q_{\mathcal{U},1}), P_M) e((\mu V+\sum_{i\in I} v_i H_i W), pk_{\mathcal{U}}) e(T,R)$$

$$e(S',P)=e((\mu' Q_{\mathcal{U},0}+\sum_{i\in I} v_i H_i Q_{\mathcal{U},1}), P_M) e((\mu' V+\sum_{i\in I} v_i H_i W), pk_{\mathcal{U}}) e(T,R')$$

where $H_i = H(i||name)$. Since the adversary causes the challenger to abort, we can get $(S', R') \neq (S, R)$. Obviously, $\mu' \neq \mu$. we define $\bar{\mu} = \mu' - \mu$. We now show that how the challenger constructs a simulator to solve the CDH problem, i.e. given P, sP, P', compute sP'. Here, we want to stress that in this analysis, we consider the user as the challenger, therefore, $x_{\mathcal{U}}$ and r_i can be chosen immediately.

In **Setup**, the simulator sets $P_M = sP$ as the master public key, but the simulator does not know the corresponding master private key s. It controls the random oracle H_1, H_2, H_3 and H_4, and keeps a list of queries and responses to queries consistently. It responds the query of H_j as follows.

The simulator randomly chooses $\gamma, a, b, c \in Z_q$, for the tuple $\{ID_{\mathcal{U}}, m_j\}$, it computes $H_j = -a^{-1}cm_j$ and also sets $\beta = a^{-1}bc$ and $R = \gamma P$. And then,

the simulator computes $Q_{\mathcal{U},0} = cP + \beta P'$ and $Q_{\mathcal{U},1} = aP + bP'$. For random values $r_j, x_{\mathcal{U}}$, the simulator can compute (S_j, R_j),

$$\begin{aligned} R_j &= r_j P \\ S_j &= m_j D_{\mathcal{U},0} + m_j x_{\mathcal{U}} V + H_j \cdot (D_{\mathcal{U},1} + x_{\mathcal{U}} W) + r_j T \\ &= m_j s Q_{\mathcal{U},0} + m_j x_{\mathcal{U}} V + H_j \cdot (s Q_{\mathcal{U},1} + x_{\mathcal{U}} W) + r_j T \\ &= m_j s(cP + \beta P') + m_j x_{\mathcal{U}} V + (-a^{-1} c m_j)(s(aP + bP') + x_{\mathcal{U}} W) + r_j T \\ &= (m_j V - a^{-1} c W) x_{\mathcal{U}} + r_j T \end{aligned}$$

Therefore, the simulator generates an authenticated $(S_j, R_j) = ((m_j V - a^{-1} c W) x_{\mathcal{U}} + r_j T, r_j P)$.

The simulator continues to interact with the adversary until the circumstance defined in **Game 3** occurs: the adversary succeeds in generating (S', R') which is different from (S, R) but can cause the verification equation holds.

Here, because $P_M = sP$, we can learn that

$$\begin{aligned} e(S, P) &= e((\mu Q_{\mathcal{U},0} + \sum_{i \in I} v_i H_i Q_{\mathcal{U},1}), P_M) e((\mu V + \sum_{i \in I} v_i H_i W), pk_{\mathcal{U}}) e(T, R) \\ &= e(s(\mu Q_{\mathcal{U},0} + \sum_{i \in I} v_i H_i Q_{\mathcal{U},1}) + x_{\mathcal{U}}(\mu V + \sum_{i \in I} v_i H_i W) + \sum_{i \in I} v_i r_i T, P) \end{aligned}$$

And since the expected aggregated value also satisfies the above equation, we can learn that

$$e(S' - S, P) = e(s\bar{\mu} Q_{\mathcal{U},0} + x_{\mathcal{U}} \bar{\mu} V, P)$$

Because $Q_{\mathcal{U},0} = cP + \beta P'$, we can further get

$$S' - S = s\bar{\mu} cP + \bar{\mu}\beta s P' + x_{\mathcal{U}} \bar{\mu} V$$

Finally, we find the solution of CDH problem:

$$sP' = \frac{S' - S - s\bar{\mu} cP - x_{\mathcal{U}} \bar{\mu} V}{\bar{\mu}\beta}$$

As we described above, because $\bar{\mu} \neq 0$ and $\beta = 0$ only with probability $1/q$ and naturally can be neglected. Therefore, the challenger can construct a simulator to solve the CDH problem with a nonnegligible probability $1 - (1/q)$.

Finally, we want to stress that Ni et al. proposed an attack model in [12]. In this attack model, an online and active external adversary can invalidate a public integrity verification scheme by tampering with the proof information. Since this attack requires a rigorous implementing condition, and is difficult to perform, we do not consider this attack in our CLP-OPOR scheme. In fact, this attack can be thwarted easily by signing the proof information. This completes the proof of **Theorem 2**.

Theorem 3. In our SCLPV scheme, if any involved party deviates from the prescribed protocol execution, it cannot deceive the rest parties.

Proof. There are three cases in **Theorem 3**. Firstly, if the cloud user is malicious, the only thing he can do is to upload "forge" signatures to the cloud server. Here, "forge" means that the signature which is generated by the cloud user, does not match the corresponding data block. In this case, note that after receiving $\widetilde{F}$, $\mathcal{C}$ first verifies that whether the file has been uploaded correctly. If failed, $\mathcal{C}$ rejects immediately. Therefore, a malicious user cannot deceive the cloud server, let alone to deceive the TPA.

Secondly, if the cloud server is malicious, it may forge a proof information to pass the TPA's verification. However, according to **Theorem 2**, we can see that this way is infeasible.

Finally, there exists a malicious TPA in our SCLPV scheme. We can further divide this situation into two cases. The first case is that the user's data is still well maintained in the cloud but the malicious TPA deceive the cloud server and the cloud user that the data has been corrupted. In this case, the cloud server just need to regenerate the proof information to prove the integrity of cloud-stored data. The other case is contrary to the first one, that is, the cloud-stored data has been corrupted, but the TPA deceive the cloud user that the data is still well maintained in the cloud. In this case, if the proof information generated by $\mathcal{C}$ cannot pass the equation (1), the cloud user can detect it immediately. However, if the proof information passes the equation (1), that means that either $< a >$ the cloud server forge a proof information that can pass the TPA's auditing, or $< b >$ the TPA colludes with $\mathcal{C}$ just to check the data blocks which are not corrupted. However, in **Theorem 2**, we have been proven that it is computationally infeasible to forge the proof information to pass the TPA's auditing. And for $< b >$, note that the indexes of challenging blocks are determined by the time-based psecudu-randomness source, thus $< b >$ is also infeasible. This completes

the proof of the **Theorem 3**.

Theorem 4. Collusion between any two involved parties cannot reduce the security of our SCLPV scheme.

Proof. In fact, if the above three theorems are correct, **Theorem 4** is also correct. We can reduce **Theorem 4** to the first three theorems. Note that collusion between any two involved parties just has two cases, since the only situation where the cloud user may abort is during the initialization phase of **Store** [18]. First one, the cloud server colludes with the TPA. This case has been proven that it cannot deceive the cloud user and naturally cannot reduce the security of our SCLPV scheme. The second case is that the cloud user colludes with the cloud server to circumvent the TPA. In this case, the cloud server just to forge an invalid proof information and sends it to the TPA. Observe that the cloud server cannot "regenerate" this forge proof information without increasing the storage overheads. Therefore, the TPA can prove that it has been used the parameters received from the cloud server and executed correctly executes all protocols indeed in **CheckLog**. Thereby this case also cannot reduce the security of our SCLPV scheme. This completes the proof of the **Theorem 4**.

Summing up, we have proved that our SCLPV scheme satisfies the four security goals indeed. According to our rigorous and formal security proofs, we can say that the proposed scheme is secure.

5.3. Performance analysis

In this Section, we give an elaborate performance analysis of the proposed SCLPV scheme, which proves that the scheme is efficient. Furthermore, we also analyze the performance of [18], and give a comparison for these two schemes in terms of bandwidth overhead and auditing overhead.

All the experiments are tested using a Window 7 system with an Intel Core 2 i5 CPU running at 2.53 GHz and 2 GB DDR 3 of RAM(1.74 GB available). All algorithms are implemented by C language and our code uses the MIRACL library version 5.6.1. The elliptic curve we used is a MNT curve, its base field size is 159 bits and its embedding degree is 6. Accordingly, the security level is chosen to be 80 bits, that is, $|v_i| = 80\ bits$, $|q| = 160\ bits$ and $|p| = 80\ bits$. All the results of experiments are represented the average of 20 trials.

5.3.1. Bandwidth overhead analysis

We first give an analysis of bandwidth overheads between the cloud server and the TPA of our SCLPV scheme, and also give a comparison with the Fortress [18]. For our SCLPV scheme, in **ChalGen** and **ProGen**, the TPA sends the challenging message to the cloud server and the cloud server responds to the TPA with the proof information $proof$. However, for the Fortress, besides same interaction operations as our SCLPV scheme, before the TPA starts to generate a challenging message, the TPA needs to download the entire data from the cloud server, and upload all the signatures to the cloud server. It means that Fortress [18] has an additional bandwidth overhead in downloading the entire cloud-stored data and uploading the corresponding signatures. Fig.2 and Fig.3 show the total communication traffic between the TPA and the cloud server of the Fortress and our SCLPV respectively. Because the total communication traffic between the TPA and the cloud server of private-key POR scheme (short for PSW) in [8], [9], public key POR scheme (BLS SW POR) in [8], [9] and [17] are similar to our SCLPV scheme, here we do not provide the bandwidth overhead analysis of these schemes.

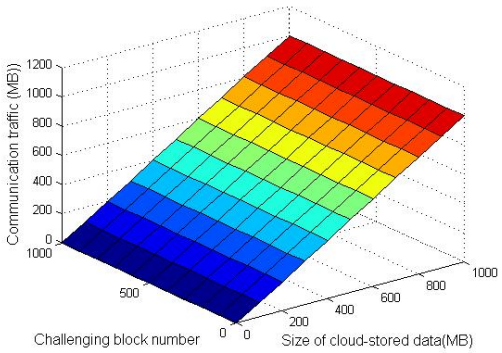

Figure 2: Communication traffic of Fortress [18]

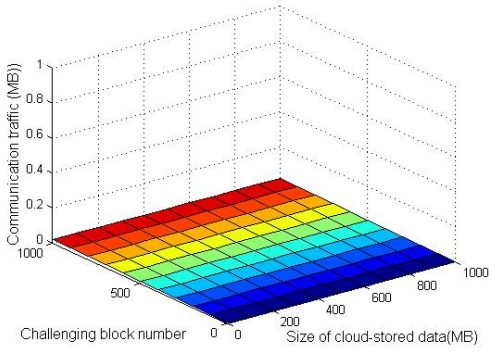

Figure 3: Communication traffic of SCLPV

According to the comparison between Fig.2 and Fig.3, we can say that our SCLPV scheme has obvious advantage in bandwidth overhead between the TPA and the cloud server. In our SCLPV scheme, the TPA does not need to bear the huge communication traffic occurred in the Fortress [18].

5.3.2. Verification overhead analysis

We then give an analysis of verification overhead. For our SCLPV scheme, in **VerPro**, the TPA verifies the integrity of cloud-stored data, and in **CheckLog**, the cloud user audits the correct behavior of the TPA. Firstly, we specify some notations represent the computation of corresponding operation (refer to Table 2).

Table 2: Notation of Operations

Symbol	Corresponding Operation
$Mult_G$	multiplication in additive group G
$Hash_G$	hash a value into G
Add_G	addition in additive group G
$Pair_{G_2}$	computing pairing $pair = e(u, v)$ where u,$v \in G$ and $pair \in G_2$

In **VerPro**, based on the proof information received from $\mathcal{C}$, the TPA verifies the equation (1) to check the integrity of cloud-stored data. The corresponding computation cost is

$$4Pair_{G_2} + (2c + 5)Hash_G + (2c + 4)Mult_G + (2c + 2)Add_G$$

where c is the total number of challenging data blocks.

Compared with the certificateless public auditing scheme proposed by [17], our SCLPV scheme has a slightly larger verification overhead in the TPA side. However, this additional verification overhead guarantees that the security vulnerability of [17] is remedied successfully, and naturally, this is surely sacrifices well worth making. Also compared with BLS SW POR, our SCLPV scheme requires more verification cost in the TPA side. But this extra cost is exactly the guarantee to avoid the certificate management problem. A detailed comparison of verification overhead of the TPA in different challenging blocks in BLS SW POR, [17] and our SCLPV scheme is described in Fig.4.

In **CheckLog**, $\mathcal{U}$ verifies the validity of log entry and the correctness of E_L. Next, $\mathcal{U}$ checks the equation (1). The corresponding computation cost is

$$4Pair_{G_2} + (2c + 5)Hash_G + (2c + 4)Mult_G + (2c + 2)Add_G$$

Here, note that $\mathcal{U}$ needs to make some additional calculations, but it can resist the malicious TPA. Moreover, as we described in Section 3, the period that $\mathcal{U}$ runs **CheckLog** is much more than the period that the TPA runs

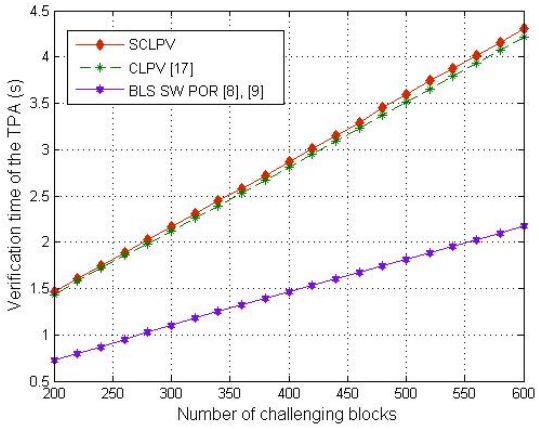

Figure 4: The performance of TPA in different challenging blocks

VerPro. Therefore, this additional calculations in the cloud user side can be, to a large extent, tolerated in the practical cloud environment.

Comparison with the Fortress [18]: The Fortress utilizes the private POR technique of [8]. Its biggest advantage is the high-efficiency of computational overhead. More precisely, as described in [18], the verification in Fortress is almost 2000 times faster than the BLS SW POR. It means that verification in Fortress is more than 2000 times faster than our SCLPV scheme. As we shown in Fig.4, the time of verification in the SCLPV is a second computation, and consequently, the time of Fortress's verification is a millisecond computation. In a practical CPSS, these two computation cost can be tolerated for the TPA and the user. However, in the SCLPV, the TPA need not to download the entire outsourced data, generate signatures for each of the data blocks and convince the user that it correctly computed parameters. Therefore, in that sense, the SCLPV is more practical than the Fortress.

6. Further discussion

To the best of our knowledge, our SCLPV scheme is the first certificateless public verification mechanism against malicious auditors. As a prior research, there are many issues to be further investigated and improved. We list some of them here based on our understanding.

First, the storage overhead can be further improved. As presented by [9],

the data block can be further divided into s sectors to give a tradeoff between storage overhead and the length of proof information. In practice, the cloud user can choose an appropriate s as needed.

Second, the optimization of computational overhead in the TPA side is expected. As we discussed in Section 5, the verification time of the Fortress [18] is a millisecond computation, but our SCLPV's verification time is a second computation. How to further optimize the verification time is our further work.

Finally, our SCLPV just considers the case of single CPSS user. In a practical CPSS, how to perform multiple verification tasks from different users simultaneously is well worth researching.

7. Conclusion and future work

In this paper, we first point out the vulnerability of the best certificateless public verification scheme. Then we propose the first secure certificateless public integrity verification scheme (SCLPV) for cloud storage in cyber-physical-social system with full proofs of security against malicious auditor and arbitrary adversaries in the Shacham et al.'s security model. With our SCLPV scheme, an auditor does not need to manage certificates. Meanwhile, a malicious auditor/CPSS user cannot impact the security of our scheme. A formal security proof proves the security of our SCLPV scheme. Performance analysis demonstrates that our SCLPV scheme is efficient and practical.

In regards to future work, we will first further investigate the optimization of computation cost in the TPA side. This means that we have to design a more efficient certificateless aggregate signature scheme that can be used in our system. Second, we also will research the case of multiple CPSS users and multiple cloud servers, e.g., how a single TPA can simultaneously handle multiple verification tasks from different CPSS users and different cloud servers, even if they are in various CPSSs. We believe these works are deserve us to study further.

8. Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.61370203, No.61472065 and No.61350110238), the Science and Technology on Communication Security Laboratory Foundation (Grant No.

9140C110301110C1103), the International Science and Technology Cooperation and Exchange Program of Sichuan Province, China under Grant 2014HH0029, and China Postdoctoral Science Foundation funded project under Grant 2014M552336.

References

[1] R. R. Raj, I. Lee, L. Sha and J. Stankovic. "Cyber-physical Systems: The Next Computing Revolution." Proceedings of the 2010 Design Automation Conference. ACM, 2010, pp. 731-736.

[2] R. K. Ganti, Y. Tsai, and T. F. Abdelzaher. "Senseworld: Towards Cyber-physical Social Networks." Proceedings of the 2008 International conference on Information processing in sensor networks. IEEE Computer Society, 2008, pp. 563-564.

[3] X. Yu, A. Pan, L. Tang, Z. Li and J. Han. "Geo-friends Recommendation in GPS-based Cyber-physical Social Network." Proceedings of the 2011 International conference on Advances in Social Networks Analysis and Mining, IEEE, 2011, pp. 361-368.

[4] Q. Wang, C. Wang, J. Li, K. Ren and W. Lou. "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing." Computer Security – ESORICS 2009. Springer Berlin Heidelberg, 2009, pp. 355-370.

[5] C. Wang, Q. Wang, K. Ren and W. Lou. "Privacy-preserving Public Auditing for Data Storage Security in Cloud Computing. Proceedings of the 2010 IEEE Conference on Computer Communication, 2010, pp. 1-9.

[6] C. Wang, S. S. Chow, Q. Wang, K. Ren and W. Lou. "Privacy-preserving Public Auditing for Secure Cloud Storage." IEEE Transactions on Computers, vol. 62, no. 2, pp. 362-375, February, 2013.

[7] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. U. Khan, R. Buyya and A. Y. Zomaya "Remote Data Auditing in Cloud Computing Environments: A Survey, Taxonomy, and Open Issues." ACM Computing Surveys, vol. 47, no.4, Article 65.

[8] H. Shacham and B. Waters. “Compact Proofs of Retrievability.” Advances in Cryptology – ASIACRYPT 2008. Springer Berlin Heidelberg, 2008, pp. 90-107.

[9] H. Shacham and B. Waters. “Compact Proofs of Retrievability.” Journal of cryptology 26.3 (2013): 442-483.

[10] A. Juels and B. S. K. Jr. “PORs: Proofs of Retrievability for Large Files.” Proceedings of the 2007 ACM conference on Computer and Communications Security. ACM, 2007, pp. 584-597.

[11] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song. “Provable Data Possession at Untrusted Stores. “Proceedings of the 2007 ACM conference on Computer and Communications Security. ACM, 2007, pp. 598-609.

[12] J. Ni, Y. Yu, Y. Mu and Q. Xia. “On the Security of an Efficient Dynamic Auditing Protocol in Cloud Storage.” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 10, pp. 2760-2761, October, 2013.

[13] E. Shi, S. Emil and P. Charalampos. “Practical Dynamic Proofs of Retrievability.” Proceedings of the 2013 ACM conference on Computer and Communications Security. ACM, 2013, pp.325-336.

[14] J. Yu, K. Ren, C. Wang and V. Varadharajan. “Enabling Cloud Storage Auditing With Key-Exposure Resistance” IEEE Transactions on Information Forensics and Security, vol. 10, no. 6, pp. 1167-1179, June, 2015.

[15] D. Cash, A. Küpçü and D. Wichs. ”Dynamic Proofs of Retrievability via Oblivious RAM.” Advances in Cryptology – EUROCRYPT 2013. Springer Berlin Heidelberg, 2013, pp. 279-295.

[16] H. Wang, Q. Wu, B. Qin and J. Domingo-Ferrer. “Identity-based Remote Data Possession Checking in Public Clouds.” IET Information Security, vol. 8, no. 2, pp. 114-121, 2014.

[17] B, Wang, B, Li, H. Li and F, Li. “Certificateless Public Auditing for Data Integrity in the Cloud.” Proceedings of 2013 IEEE conference on Communications and Network Security, IEEE, 2013, pp. 136-144.

[18] F. Armknecht, J. Bohli, G. O. Karame, Z. Liu and C. A. Reuter. "Outsourced Proofs of Retrievability." Proceedings of the 2014 ACM conference on Computer and Communications Security. ACM, 2014, pp. 831-843.

[19] H. Li, A. Dimitrovski, J. B. Song, Z. Han and L. Qian. "Communication Infrastructure Design in Cyber Physical Systems with Applications in Smart Grids: A Hybrid System Framework." IEEE Communications Survey and Tutorials, vol. 16, no. 3, pp. 1689-1708, Third quarter, 2014.

[20] Z. Liu, D. Yang, D. Wen, W. Z and W. Mao. "Cyber-Physical-Social Systems for Command and Control." IEEE Intelligent Systems, vol. 26, no. 4, 2011, pp. 92-96.

[21] A. Sheth, P. Anantharam and C. Henson. "Physical-Cyber-Social Computing: An Early 21st Century Approach." IEEE Intelligent Systems, vol. 28, no. 1, 2013, pp. 78-82.

[22] L. Zhang, B. Qin, Q. Wu and F. Zhang. "Efficient Many-to-one Authentication with Certificateless Aggregate Signatures." Computer Networks, 2010, vol. 54, no. 14, pp. 1167-1179, October, 2010.

[23] D. Boneh, B. Lynn and H. Shacham. "Short Signatures from the Weil Pairing." Journal of Cryptology 17.4 (2004): 297-319.

[24] Nakamoto, Satoshi. "Bitcoin: A Peer-to-peer Electronic Cash System." [Online]. Available: http://www.cryptovest.co.uk/resources/Bitcoin

[25] Z. Zhang, D. S. Wong, J. Xu and D. Feng. "Certificateless Public-key Signature: Security Model and Efficient Construction." Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2006, pp. 293-308.

[26] K. Nyberg and R. A. Rueppel. "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem." Advances in Cryptology – EUROCRYPT 1994. Springer Berlin Heidelberg, 1995, pp. 182-193.

[27] S. S. Al-Riyami and K. G. Paterson. "Certificateless Public Key Cryptography." Advances in Cryptology – ASIACRYPT 2003. Springer Berlin Heidelberg, 2003. pp. 452-473.

[28] W. Mao. "Modern Cryptography: Theory and Practice." Prentice Hall Professional Technical Reference, 2003.

[29] H. Li, X. Lin, H. Yang, X. Liang, R. Lu and X. Shen. "EPPDR: An Efficient Privacy-Preserving Demand Response Scheme with Adaptive Key Evolution in Smart Grid." IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 8, pp. 2053 - 2064, August, 2014.

[30] I. S. Reed and G. Solomon. "Polynomial Codes over Certain Finite Fields." Journal of the Society for Industrial & Applied Mathematics, vol. 8, no. 2, pp. 300-304, June, 1960.

[31] H. Li, R. Lu, L. Zhou, B. Yang, and X. Shen. "An Efficient Merkle Tree Based Authentication Scheme for Smart Grid." IEEE SYSTEMS Journal, vol. 8, no.2, pp. 655 - 663. June, 2014.

Appendix A. On the vulnerability of the CLPV [17]

As pointed out in Section 1, the only certificateless public integrity auditing mechanism — CLPV [17] — has security vulnerability and naturally fails to provide the strong security guarantees. To illustrate this, we show in how to invalidate the CLPV by utilizing this security vulnerability.

Since the CLPV [17] is based on a homomorphic certificateless authenticable signature technique (short for HA-CLS), which also has same security issue. Now, we first briefly review the HA-CLS, then we give a concrete to show how to invalidate it.

In the HA-CLS, the KGC first generates the system parameters $(G_1, G_2, e, g, P_1, H_1, H_2, P_T)$, where G_1 and G_2 are two multiplicative groups of prime order q, $e : G_1 \times G_1 \to G_2$, g is a generator of G_1, P_1 is a random number, H_1 and H_2 are two cryptographic hash functions, $P_T = g^\lambda$ and λ is the master key. Then, the KGC generates the partial private key for the user with identity ID as $D_s = H_1(ID)^\lambda$. The user randomly chooses $x \in Z_q*$ as the secret key and corresponding public key is $P_s = g^x$. Now, the signing secret key is (x, D_s), and the corresponding verifying public key is $(P_s, H_1(ID))$. Given data block $m \in Z_q$ and block identifier $id \in \{0,1\}^*$, the user generates a signature by using signing secret key as:

(1) Compute $V = H_2(ID||P_s||id) \cdot P_1^m$.

(2) Output a signature σ on block m and block identifier id as $\sigma = V^x \cdot D_s$.

Given signature σ, system parameters, verifying public key, user's identity

ID, data block m and block identifier ID, a verifier checks the integrity of this block as:

(1) Compute $V = H_2(ID||P_s||id) \cdot P_1^m$

(2) Verify

$$e(\sigma, g) \stackrel{?}{=} e(H_1(ID), P_T) \cdot e(V, P_s)$$

If the equation holds, output *Accept*, otherwise, output $Reject$.

As described in [28], a secure signature algorithm should satisfy a property, that is, without the correct signing secret key, an adversary cannot create valid signature from given data block. Here, "correct" means "correctness" and "completeness". Unfortunately, the HA-CLS proposed by [17] fails to meet this security property.

More precisely, in the HA-CLS, the signing secret key is (x, D_s). To create a forge signature, first, an adversary gets a valid signature σ generated by the user ID earlier, he also gets correspond data block m and identifier id. Then, the adversary obtains x and computes $V = H_2(ID||P_s||id) \cdot P_1^m$, and he also calculates V^x. Next, the adversary computes the inverse element of V^x. Finally, the adversary computes the partial private key D_s as

$$D_s = \sigma \cdot (V^x)^{-1}$$

Now, the adversary has the user's signing secret key, and he can create valid signature from given data block. By this way, the adversary invalidates the HA-CLS proposed by [17]. The same security vulnerability still exists in the public auditing mechanism presented by [17], and here we omit it. However, it is easy to verify that our SCLPV scheme can thwart this attack. Due to space limitations, we will not give the formal proof.