Securing the cloud storage audit service: defending against frame and collude attacks of third party auditor

Kun Huang[1], Ming Xian[1], Shaojing Fu[2], Jian Liu[1]

[1]State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, National University of Defense Technology, Changsha 410073, People's Republic of China
[2]School of Computer Science, National University of Defense Technology, Changsha 410073, People's Republic of China
E-mail: khuang_123@163.com

Abstract: Cloud computing has been envisioned as the next generation architecture of the IT enterprise, but there exist many security problems. A significant problem encountered in the context of cloud storage is whether there exists some potential vulnerabilities towards cloud storage system after introducing third parties. Public verification enables a third party auditor (TPA), on behalf of users who lack the resources and expertise, to verify the integrity of the stored data. Many existing auditing schemes always assume TPA is reliable and independent. This work studies the problem what if certain TPAs are semi-trusted or even potentially malicious in some situations. Actually, the authors consider the task of allowing such a TPA to involve in the audit scheme. They propose a feedback-based audit scheme via which users are relaxed from interacting with cloud service provider (CSP) and can check the integrity of stored data by themselves instead of TPA yet. Specifically, TPA generates the feedback through processing the proof from CSP and returns it to user which is yet unforgeable to TPA and checked exclusively by user. Through detailed security and performance analysis, the author's scheme is shown to be more secure and lightweight.

1 Introduction

Cloud computing is a service model that offers users on-demand network access to a large shared pool of computing resources (the cloud). The ever cheaper and more powerful processors, together with the Software as a Service computing architecture, are transforming data centres into pools of computing service on a huge scale. The financial benefits of cloud computing are extensively recognised. From the users' perspective, the ability to utilise and pay for resources on demand and the elasticity of the cloud are strong incentives for migration to the cloud.

Despite these benefits, public clouds are still not widely adopted, especially by enterprises. Most large organisations today run private clouds, in the sense of virtualised and geographically distributed data centres, but rarely rely primarily on externally managed resources; notable exceptions include Twitter and The New York Times, which run on Amazon infrastructure [1]. According to the recent survey, approximately 70 percentages of the current companies do not trust the public auditing services.

Furthermore, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in cloud computing a formidable task, especially for users with constrained computing resources. Since well-known cloud providers experienced temporary lack of loss of personal customer data, most notably the 2011 T-mobile/sidekick incident [4], the importance of ensuring the remote data integrity has been highlighted by the following research works under different system and security models [5–21]. Simultaneously, as a complementary approach, researchers have also proposed distributed protocols [22–27] for ensuring storage correctness and guaranteeing the data availability across multiple servers or peers. Besides, many of them [13, 14, 16, 23] support public verifiable remote integrity checking, which can release users from the computation and online burden for periodical integrity check, especially desirable when the user is equipped with a low end computation device (e.g. smart phone) or is not always connected to the Internet. Nevertheless, without appropriate implementation, public verifiable auditing would impose users a false intuition that their data were intact in the cloud storage. Wang *et al.* [13] first proposed the privacy-preserving public auditing with blind technique, which is based on Shacham's scheme [23]. In their model, the prover masks the proof for a challenge with some randomness so that third party auditor (TPA) cannot abstract the file blocks via solving corresponding linear equation set. Afterwards, Xu [28] prevents the possible exclude attack originating from Wang *et al.* [13] when the selected file blocks have low entropy and enable user to audit the auditor by

the uneven distribution of verification time or denial of availability lasting at least several hours [2, 3] and striking reviewing the TPA's audit process. As stated by Xu, there

are many situations that TPA may abuse public verifications, such as:

- Owing to certain financial benefits, TPA may not always be reliable and independent and may collude with the cloud service provider (CSP) to pass the verification for hiding some CSP's corrupted incident. For instance, some rarely accessed data files are corrupted by CSP and TPA helps CSP conceal this incident for some illegal benefits. Or, TPA may be corrupted by external attackers resulting in seemingly collude attack.
- In contrast to the above case, TPA may frame CSP for their some benefit dispute by means of deliberately inaccurate auditing. For example, some CSP's opponent bribes TPA to falsely audit CSP's stored data just for corrupting the CSP's reputation. Or, TPA may be corrupted by external attackers resulting in frame attack.
- Even worse, some malicious CSPs may create a lot of 'Sybil' identities, who are announcing that they are volunteered to check the cloud storage periodically.
- In order to reduce the overhead of auditing, TPA may also defer some specific audit tasks to sometime when a lot of tasks could be audited in batch simultaneously, resulting in fi service.

Nevertheless, in most cases TPA always behaves well and the aforementioned situations could be handled by law system in real world application once detected. However, we cannot guarantee the non-occurrence of any situations without detected and it would trouble a lot if there is the involvement of law system. Therefore our research addresses the latent threat of public verifiable auditing executed by potentially malicious third party auditor. In this paper, we employ multi-TPAs to take the delegated task since just one TPA cannot be trusted and may be revoked and replaced by others when necessary. We can conclude that the data remotely stored in cloud is intact only if there exist more than one aggregated feedback out of these TPAs verified right by user. Assumption is made here that among all the required TPAs, there exists at least one TPA deserving trust.

1.1 Contribution

Our contribution can be summarised as the following five aspects:

1. The user can authenticate whether any TPA has cheated the date owner or indeed executed the designated computational audit task.
2. The user can verify whether the TPA did corresponding task within the prescriptive time specified by the user, since the adoption of real-time auditing by TPA significantly improve the auditing efficiency and reduce the corresponding overhead.
3. The user could revoke the malicious TPAs via the proposed scheme.
4. The proposed scheme can prohibit the frame and collude attack thoroughly while other existing schemes cannot.
5. The user's data privacy is protected against the malicious TPAs.

The rest of this paper is organised as follows: Section 2 introduces some preliminaries and background. Then, we provide the detailed description of our scheme and give the performance evaluations in Sections 3 and 4, respectively. Finally, Section 5 concludes the whole paper.

2 Preliminaries and background

2.1. Bilinear pairing

We first briefly introduce some mathematical theory related to both the following schemes and ours.

Let G_1 be a cyclic multiplicative group generated by P, whose order is a prime p, and G_2 a cyclic multiplicative group of the same order p. A bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$ is a map with the following properties:

(1) ***Bilinear:*** $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in G_1$ and $a, b \in Z_p^*$.
(2) ***Non-degenerate:*** $e(P, P)$ is a generator of G_2.
(3) ***Computable:*** There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in G_1$.

The above properties also imply that $e(uv, w) = e(u, w) \cdot e(v, w)$ for all $u, v, w \in G_1$.

2.2 Blind technique for protecting privacy against TPA

Wang *et al.* [13] proposed a blind technique in addition to Shacham and Water's scheme [23], attempting to achieve privacy-preserving third party auditing. With their blind technique, the prover CSP can mask the proof for a challenge with some randomness and the verifier TPA is still able to verify the validity of the masked proof.

We describe their prove-verify algorithms generally as follows:

Blind prove: Upon receiving challenge $Q = (i, v_i)_{i \in I}$ where I is a c-element subset of $[1, n]$ randomly picked by TPA, the CSP generates a response proof of data storage correctness. Specifically, the CSP blinds μ to generate μ' in this way: Choose r at random from Z_p and compute $R = e(u, \upsilon)^r$ and $\gamma = h(R) \in Z_p$ where $h: G_2 \rightarrow Z_p$ is some secure hash function

$$\mu' = r + \gamma\mu (\mathrm{mod}) p \tag{1}$$

The CSP then sends μ', σ, R as the response proof of storage correctness to the TPA.

Blind verify: The TPA performs the following equality check

$$R \cdot e(\sigma^\gamma, g) = e\left(u^{\mu'} \cdot \left(\prod_{(i,v_i) \in Q} H(i)^{v_i}\right)^\gamma, \upsilon\right) \tag{2}$$

Xu [28] proposes a kind of possible attack, say exclude attack, that when the selected file blocks F_i's have low entropy, TPA could be able to find the values of F_i by brute-force search. The issue can be mitigated through compressing the whole data file or just be encrypted before outsourcing. So this attack is out of our focus, and we still borrow the blind technique of the Wang *et al.* to preserve the user's data privacy.

2.3 Limitation of previous work

Wang *et al.* proposed a method to blind their data confidentiality against TPA, but the scheme is based on the assumption that TPA is reliable and trustworthy. Xu [28] presentes a protocol allowing user to audit the TPA, however, his scheme requires users themselves to perform the task of checking the integrated verification equation which brings so much computation and communication burden to user. Take examples as following:

- In Wang *et al.*'s model, the last verification (2) is checked by TPA. Supposing TPA is malicious or corrupted by external attackers, he may oppositely return the final checking result of (2). Thus, CSP may be framed or collude with TPA.
- In Xu's model, although users themselves are able to audit the auditor by retrieving the reply from **Receive Server** and judging some aggregated verification equation, TPA's role in the context of his scheme appears to be relatively meaningless since the majority of computation and communication overhead is imposed on users. What's more, if some TPA is malicious or corrupted, there does exist the threat of frame attack between TPA and CSP. For instance, in the **Execute Plan** (depicted below) TPA may deliberately send random message to receive server, instead of the reply from CSP.

3 Proposed scheme

3.1 Notation and definition of our protocol

We quote a similar definition of Xu's scheme with some modifications of the context of timing audit plan and adapt the blind technique of Wang *et al.*'s scheme to our new audit system.

Let $\mathbb{T}$ be the domain of timestamps w.r.t. a particular minimum time unit (e.g. minute). We define an 'audit' plan as follows:

Definition 1 (audit plan): An audit plan of size m is a pair of vectors $\{\boldsymbol{T}, \boldsymbol{Q}\} \in \mathbb{T}^m \times \mathbb{Q}^m$, where $\mathbb{Q}$ is the set of all possible challenges in our scheme, $\boldsymbol{T} = (t_1, \ldots, t_m) \in \mathbb{T}^m$ is a vector of time points t_i's, such that $t_1 \le t_2, \ldots, \le t_m$, and $\boldsymbol{Q} = (Q_1, \ldots, Q_m) \in \mathbb{Q}^m$ is a vector of m challenges where $Q_\lambda = \{(i, v_i^\lambda) | i \in I\}$. I is a c-element subset of $[1, n]$ and n is the number of the file blocks.

Table 1 Variable and notation used in our scheme description

Variable	Description
$\mathbb{T}$	domain of timestamps
$\boldsymbol{T}$	audit time field which is a vector of time points t_i's
$\boldsymbol{Q}$	vector of m challenges
Q_λ	λth challenge
$\hat{\boldsymbol{Q}}$	challenge matrix transformed from $\boldsymbol{Q}$
$\boldsymbol{Q}_\lambda$	vector of $m-1$ challenges which is from $\boldsymbol{Q}$ missing Q_λ
$\hat{\boldsymbol{Q}}_\lambda$	challenge matrix transformed from $\boldsymbol{Q}_\lambda$
I	c-element subset of $[1, n]$ where c is set to be a fixed constant
v_i^λ	ith random element in Z_p within the λth challenge
n	number of the file blocks
m	length of an audit plan
β_i	secret value used to construct a new novel authenticated tags
p	big prime which is the order of two groups G_1 and G_2
σ_i	constructed authenticated tag
ρ_λ	another secret value pre-computed and permanently stored by user
R	random value to mask the linear combination of sampled blocks μ_λ
θ_λ	feedback calculated by TPA and passed to user for further checking

Remark 1: In our protocol, the audit plan is chosen according to some requirements. We first split the $\boldsymbol{Q}$ into a $m \times n$ matrix

$$\hat{\boldsymbol{Q}} = \begin{pmatrix} v_1^1 & v_2^1 & \ldots & v_n^1 \\ v_1^2 & v_2^2 & \ldots & v_n^2 \\ \vdots & \vdots & & \vdots \\ v_1^m & v_2^m & \ldots & v_n^m \end{pmatrix}$$

where we set $v_i^\lambda = 0$, if $(i, v_i^\lambda) \notin Q_\lambda$ and $m < n$. Then we construct a vector $\boldsymbol{Q}_\lambda$ such that $\boldsymbol{Q}_\lambda = (Q_1, \ldots, Q_{\lambda-1}, Q_{\lambda+1}, \ldots, Q_m)$ which is still split into a matrix

$$\hat{\boldsymbol{Q}}_\lambda = \begin{pmatrix} v_1^1 & v_2^1 & \ldots & v_n^1 \\ \vdots & \vdots & & \vdots \\ v_1^{\lambda-1} & v_2^{\lambda-1} & \ldots & v_n^{\lambda-1} \\ v_1^{\lambda+1} & v_2^{\lambda+1} & \ldots & v_n^{\lambda+1} \\ \vdots & \vdots & & \vdots \\ v_1^m & v_2^m & \ldots & v_n^m \end{pmatrix}$$

We require that $rk(\hat{\boldsymbol{Q}}) = rk(\hat{\boldsymbol{Q}}_\lambda) + 1$ for all $1 \le \lambda \le m$ where $rk()$ represents the rank of a matrix. In other words, any row Q_λ of matrix $\hat{\boldsymbol{Q}}$ is linearly independent with the remaining rows.

We quote two server definitions which can exist independently outside our proposed audit system.

Definition 2 [28] (time server): A time-server is associated with a domain $\mathbb{T}$ of timestamps and an IBE public-private key pair (tpk, tsk), where tpk is publicly available and tsk is kept secret by the time server. At each time point $t \in \mathbb{T}$, the time server broadcasts the decryption key w.r.t. the identity t. The time server does nothing else.

Definition 3 [28] (receive server): A receive-server has a large storage. Once receiving a message **Msg** designating for receiver **Rev** from a sender **Snd** at time t, the receive-server will record (t, **Rev**, **Snd**, **Msg**) in his/her storage. The receiver-server also allows the designated receiver to retrieve their message. In real world application, we may adopt a reliable email server to play the role of receive-server.

For clarity, we provide some important used variables in tabular form, see Table 1.

3.2 Detailed scheme

Before giving our main result, we introduce some structural units and some other conceptions. Unlike the previous work on the TPA-based audit system, we construct new authenticated tags by making some modification of the original ones. The original tags are always formulated by $\sigma_i = [H(i)u^{F_i}]^\alpha$. Here, we add a secret value $\beta_i \in Z_p$ into the

original one like $\sigma_i = [H(i) \cdot u^{F_i} \cdot u^{\beta_i}]^{\alpha}$. In order to revoke the malicious TPA and invoke new TPAs when necessary, we also adopt multi-TPA to take the obliged task. Assume at least 1 out of all the TPAs are trustworthy and reliable, but we do not know which one deserves trusting. In a real-world application, this assumption is reasonable, or there is no technical skill to deal with.

- ***Set up phase***: The cloud user runs KeyGen to generate the public and secret parameters. Specifically, the user chooses a random $\alpha \leftarrow Z_p$, a random element $u \leftarrow G_1$ and computes $\upsilon \leftarrow g^{\alpha}$. The secret parameter is $sk = (\alpha, \beta_i)$ and the public parameters are $pk = (v, g, u, e(u, \upsilon))$.

Let $F = (F_1, F_2, \ldots, F_n)$, $F_i \in Z_p$ for each $1 \le i \le n$, be the resulting preprocessed file through encoding and encryption. Then user runs **SigGen** to compute authenticator σ_i for each block F_i

$$\sigma_i \leftarrow (H(i) \cdot u^{F_i} \cdot u^{\beta_i})^{\alpha} \in G_1 \tag{3}$$

where $\beta_i \in Z_p$ is randomly chosen by user. Denote the set of authenticators by $\Phi = \{\sigma_i\}_{1 \le i \le n}$. Finally, the user sends F along with the verification metadata Φ to the server, deletes them from local storage and stores β_i locally temporarily.

- ***Release plan***: The cloud user as required establishes an audit plan $\boldsymbol{P} = \{\boldsymbol{T} = (t_1 \le t_2, \ldots \le t_m), \boldsymbol{Q} = (Q_1, \ldots, Q_m) \in \mathbb{Q}^m)\} \in \mathbb{T}^m \times \mathbb{Q}^m$ where we set $m = 2^a$ explained later. Meanwhile, user computes

$$\rho_\lambda = \sum_{(i,v_i)_\lambda \in Q_\lambda} \beta_i \cdot v_i^{\lambda}, \quad \rho = \sum_{\lambda=1}^{m} \rho_\lambda \tag{4}$$

This phase is always synchronised with setup phase where user deletes β_i once ρ_λ and ρ is obtained and stored by user permanently. For each $\lambda \in [1, m]$, user encrypts Q_λ using the timed-release encryption scheme as in Xu [28]

$$\hat{Q}_\lambda \leftarrow \mathbf{TEnc}(Q_\lambda, t_\lambda) \tag{5}$$

User sends the encrypted audit plan $\hat{\boldsymbol{P}} = \{(t_\lambda, \hat{Q}_\lambda) : 1 \le \lambda \le 2^a\}$ to TPA.

- ***Execute plan***: At time t_λ, $1 \le \lambda \le 2^a$, TPA receives the decryption key K_λ for t_λ from time server, and decrypts $\hat{Q}_\lambda$ to obtain the challenge Q_λ

$$Q_\lambda \leftarrow \mathbf{TDec}(\hat{Q}_\lambda, K_\lambda) \tag{6}$$

TPA interacts with CSP to execute the interactive algorithm **Chal – Pro**: TPA sends $Q_\lambda = \{(i, v_i)_\lambda \in [1, n] \times Z_p\}$ to CSP, the cloud storage server then runs **GenProof** to generate a response proof of data storage correctness. Specifically, the server chooses a random element $r \leftarrow Z_p$, and calculates $R = e(u, \upsilon)^r \in G_2$. Let μ_λ denote the linear combination of sampled blocks specified in chal: $\mu_\lambda = \sum_{(i,v_i)_\lambda \in Q_\lambda} v_i F_i$. To blind μ with r, the CSP computes

$$\mu'_\lambda = r + \gamma \cdot \mu_\lambda (\mathrm{mod}) p, \ \sigma_\lambda = \prod_{(i,v_i)_\lambda \in Q_\lambda} \sigma_i^{v_i} \in G_1 \tag{7}$$

where $\gamma = h(R) \in Z_p$ and h is a hash function. It then sends $R_\lambda = \{\mu'_\lambda, \sigma_\lambda, R\}$ as the response proof of storage correctness to the TPA.

With the response from the server, the TPA runs **ProcessProof** to process the response by computing $\gamma = h(R)$ and θ_λ

$$\begin{aligned}
\theta_\lambda &= \frac{R \cdot e(\sigma_\lambda^{\gamma}, g)}{e\left(u^{\mu'_\lambda} \cdot \left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i}\right)^{\gamma}, \upsilon\right)} \\
&= \frac{R \cdot e(\sigma_\lambda^{\gamma}, g)}{e\left(u^{r} \cdot \left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i} \cdot u^{\mu_\lambda}\right)^{\gamma}, \upsilon\right)} \\
&= \frac{R \cdot e(\sigma_\lambda^{\gamma}, g)}{\left\{e\left(\left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i} \cdot u^{\mu_\lambda}\right)^{\gamma}, \upsilon\right) \cdot e(u, \upsilon)^{r}\right\}} \\
&= \frac{e\left(\prod_{(i,v_i)_\lambda \in Q_\lambda} \left(H(i)^{v_i} \cdot u^{v_i \cdot F_i} \cdot u^{v_i \cdot \beta_i}\right)^{\alpha \cdot \gamma}, g\right)}{e\left(\left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i} \cdot u^{\mu_\lambda}\right)^{\gamma}, \upsilon\right)} \\
&= \frac{e\left(\left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i} \cdot u^{\mu_\lambda}\right)^{\gamma}, \upsilon\right) \cdot e\left(u^{\gamma \cdot \sum_{(i,v_i)_\lambda \in Q_\lambda} \beta_i \cdot v_i}, \upsilon\right)}{e\left(\left(\prod_{(i,v_i)_\lambda \in Q_\lambda} H(i)^{v_i} \cdot u^{\mu_\lambda}\right)^{\gamma}, \upsilon\right)} \\
&= e\left(u^{\gamma \cdot \sum_{(i,v_i)_\lambda \in Q_\lambda} \beta_i \cdot v_i}, \upsilon\right) \\
&= e(u, \upsilon)^{\gamma \cdot \rho_\lambda}
\end{aligned} \tag{8}$$

The protocol is illustrated in Fig. 1.

For each $1 \le \lambda \le m$, TPA sends $\mathbf{Msg} = (\lambda, \theta_\lambda)$ to **Receive Server**. Concurrently, **Receive Server** record (t, **Rev**, **Snd**, **Msg**) where t symbols the time TPA send **Msg** to **Receive Server**.

- ***Review plan***: After $t_m + \Delta$ where Δ indicates the prescriptive time interval, user retrieves the record from **Receive Server** to check whether the recorded time t lie in the domain of specified time. If some time point is beyond the time domain, user revoke the corresponding TPA and invoke a new TPA. Otherwise, user require the TPAs to execute **Aggregate – Feedback – algorithm**: TPA computes an aggregate θ as $\theta = \prod_{\lambda=1}^{\lambda=m} \theta_\lambda$.

TPA sends θ to user. On receiving the aggregate feedback θ, user then check whether the following verification equation is established

$$e(u, \upsilon)^{\gamma \cdot \rho} = \theta \tag{9}$$

- ***TPA-oriented security visibility***: During the auditing task, CSP masks μ_λ to generate μ'_λ and then sends $\{\mu'_\lambda, \sigma_\lambda, R\}$ to TPA. As the $\mu'_\lambda = r + \gamma \cdot \mu_\lambda$ and r is stored by CSP, μ_λ cannot be exposed to TPA. Therefore TPA cannot combine several μ_λ to solve the linear equations. In addition, the original file is encrypted using AES. These two methods assure that the privacy of user's data can be preserved. Interacting with CSP, TPA plays the role of issuing challenges and obtaining proof response from CSP. Henceforth, TPA runs **ProcessProof** to process the response as a computing mechanism and continuously TPA sends the processed data θ_λ to receive server within the specified time domain. After time $t_m + \Delta$, the recorded time

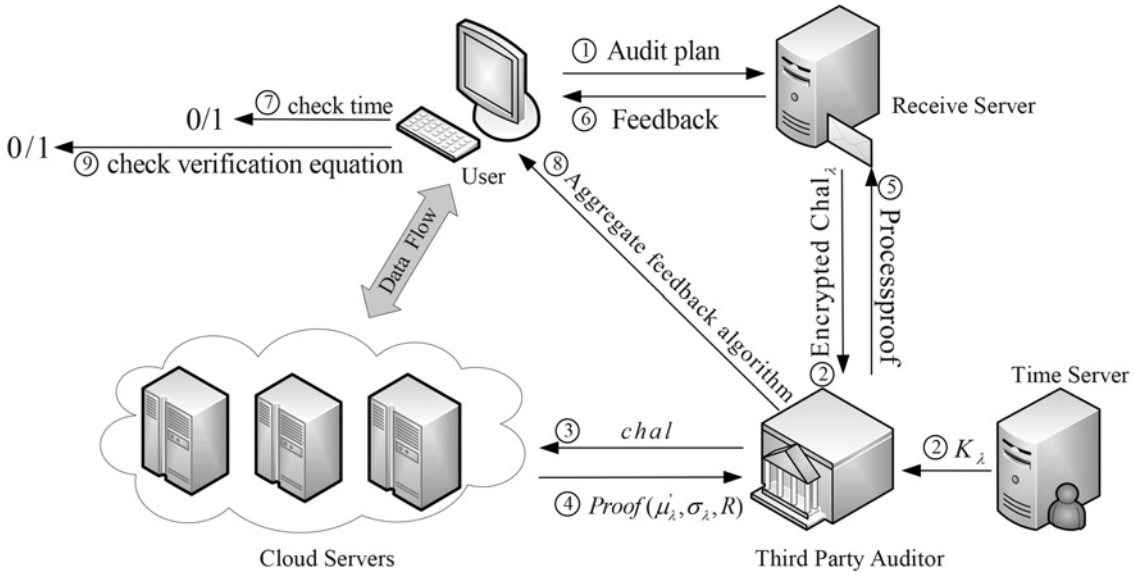

Fig. 1 *Audit protocol diagram*

t retrieved from receive server is utilised to check whether TPA indeed perform the obligated role in the prescriptive time domain. Now it comes to the core part of auditing course. User requires the remaining qualified TPAs to runs **Aggregate – Feedback – algorithm** to aggregate the feedback from time t_1 to t_m. At last, user takes his stored secret value ρ just only to check (9). The security visibility is based on (9) of which the exponent is computed by user blindly against others.

4 Evaluation

4.1. Security analysis

• ***Unforgebility of feedback****:* After sending $\mathbf{Msg} = (\lambda, \theta_\lambda)$ to receive server, TPA may stored it locally. Thus, there may be all the m θ_λ feedbacks possessed by TPA. The following theorem will demonstrate TPA cannot forge a valid feedback without the knowledge of β_i, which guarantees the unforgebility of feedback.

Theorem 1: Based on the above requirement of choosing audit plan and given $(\theta_1, \theta_2, \ldots, \theta_{m-1})$ satisfying $\theta_i = e(u, v)^{\gamma \cdot \rho i}$, forging a feedback θ_m satisfying $\theta_m = e(u, v)^{\rho m}$ is hard.

Proof: In view of the methods of choosing audit plan, we know $\rho_\lambda = \Sigma_{i=1}^{n} \beta_i \cdot \nu_i^\lambda$ where ν_i^λ and γ are known to TPA. As the discrete logarithm problem is hard, one cannot obtain ρ_λ from $\theta_\lambda = e(u, v)^{\gamma \cdot \rho_\lambda}$. However, if ρ_m can be represented in linear combination of $(\rho_1, \ldots, \rho_{m-1})$ such as $\rho_m = k_1 \cdot \rho_1 + s + k_{m-1} \cdot \rho_{m-1}$, then there will exist a kind of attack which is not required to compute the logarithm in group. We list the verification equation set

$$\begin{cases} e(u, v)^{\gamma \cdot \rho_1} = & \theta_1 \\ e(u, v)^{\gamma \cdot \rho_2} = & \theta_2 \\ \vdots & \vdots \\ e(u, v)^{\gamma \cdot \rho_{m-1}} = & \theta_{m-1} \end{cases}$$

□

From the above equation set, we are able to easily obtain

$$e(u, v)^{\gamma \cdot \rho_m} = e(u, v)^{\gamma \cdot (k_1 \cdot \rho_1 + \cdots + k_{m-1} \cdot \rho_{m-1})} = \prod_{i=1}^{m-1} \theta_i^{k_i} \qquad (10)$$

In order to explain the impossibility of the linear combination, we present another linear equation set

$$\begin{cases} \beta_1 \cdot \nu_1^1 + \beta_2 \cdot \nu_2^1 + \cdots + \beta_n \cdot \nu_n^1 = \rho_1 \\ \beta_1 \cdot \nu_1^2 + \beta_2 \cdot \nu_2^2 + \cdots + \beta_n \cdot \nu_n^2 = \rho_2 \\ \vdots \\ \beta_1 \cdot \nu_1^{m-1} + \beta_2 \cdot \nu_2^{m-1} + \cdots + \beta_n \cdot \nu_n^{m-1} = \rho_{m-1} \end{cases}$$

That is $\hat{\boldsymbol{Q}}_{\boldsymbol{m}} \cdot \hat{\beta} = \hat{\rho}$, where $\hat{\boldsymbol{Q}}_{\boldsymbol{m}}$ is defined in Definition 1, $\hat{\beta} = (\beta_1, \ldots, \beta_n)$ and $\hat{\rho} = (\rho_1, \ldots, \rho_{m-1})^{\mathrm{T}}$. Besides, for $m < n$, we cannot solve the equation set. Again because of $rk(\hat{\boldsymbol{Q}}) = rk(\hat{\boldsymbol{Q}}_{\boldsymbol{m}}) + 1$, the row Q_m of matrix $\hat{\boldsymbol{Q}}$ is linearly independent with the remaining rows. In other words, we cannot find $k_1, \ldots, k_m$ such that $\rho_m = k_1 \cdot \rho_1 + , \ldots, + k_{m-1} \cdot \rho_{m-1}$. Thus, we obtain the proof of unforgebility of feedback.

4.2 Performance analysis

4.2.1 Theoretical analysis: We now assess the performance of the proposed storage auditing scheme. We focus on the cost of user's computation as well as TPA's computation including processing the proof and aggregating the feedback.

• ***TPA's computation overhead****:* As discussed, TPA play the role of computing which user may not have. In the **ProcessProof** phase, TPA computes θ_λ by division operation. Given a cyclic multiplicative group, the division arithmetic can be transformed into multiplicative arithmetic via the following lemma:

Lemma 1: In a prime p order group G_2, any element g's inverse g^{-1} can be computed in $O(\ln(p - 1))$ multiplicative

Table 2 Performance comparison of various scheme

Various scheme	User's overhead	TPA's overhead	Anti-frame-attack	Anti-collude-attack
Wang [13]	small	$m{\cdot}c{\cdot}O(\ln(p-1))$ mults + 2 m pairs	no	no
Xu [28]	$m{\cdot}c$ mults + 2 pairs	small	no	yes
ours	$m{\cdot}c$ mults	$m{\cdot}c{\cdot}O(\ln(p-1))$ mults + 2 m pairs	yes	yes

operations. In the same way, one exponentiation operation can be computed in $O(\ln(p-1))$ multiplicative operations.

Proof: As known to all, any element g's inverse in a prime order group is unique and equal to g^{p-1}. So a trivial computation of g^{-1} is through the method of g's $(p-1)$th power. Another efficient way to compute g^{-1} is splitting $p-1$ in binary such as

$$p-1=2^{k_1}+2^{k_2}+\cdots+2^{k_q},\ k_q<k_{q-1}<\cdots<k_1 \quad (11)$$

□

We set $k_q=1$, as $p-1$ is even. And from the above equation, it is easy to see $k_1\le\ln(p-1)\le k_1+1$. Thus, we can obtain an algorithm: (i) compute the square of g, (i.e. g^2), (ii) continue to compute $(((g^2)^2)\ldots)^2$ until the number of square operations of g^2 reach to $k_{q-1}-1$; then we obtain $g^{2^{k_{q-1}}}$ and (iii) after obtaining $g^{2^{k_1}}$, we multiply all the g's powers and finally we obtain the g^{p-1}.

In complexity analysis, we only need to repeat k_1 times square operations of element g, that is we should carry out k_1 multiplicative operations. Adding the final multiplication of all the g's powers, the overall computation cost is (k_1+q-1) which is the same order of magnitude with $\ln(p-1)$.

Hence, within an entire audit plan, the **ProcessProof** phase totally costs TPA $2m$ pair operations, m $(c+3)$ exponentiations, m hash operations, m $(c+1)$ multiplicative operations and m division operations. In the **Aggregate – feedback** phase, TPA aggregates all the θ_λ from $\lambda=1$ to m. This phase costs TPA $(m-1)$ multiplicative operations. During an audit plan, through operation transformation TPA needs to perform $2m$ pair operations, m hash operations and $m\{(c+4){\cdot}O(\ln(p-1))+c+1\}$ multiplicative operations which is approximately $m{\cdot}c{\cdot}O(\ln(p-1))$ multiplications. In addition, hash operations comparatively cost the least.

- ***User's computation overhead:*** In the **setup** phase, user computes the signatures of every file blocks F_i, $1\le i\le n$. This cost is out of our focus as it is similar to other audit protocols. User just needs to compute a_i through F_i, secret key α, and the secret value β_i stored temporarily. In the **Release plan** phase, user establishes an audit plan according to certain requirements. Thereafter, user computes ρ and $\rho_\lambda, 1\le\lambda\le m$ like (4) and simultaneously deletes β_i. This cost user $c{\cdot}m$ multiplicative operations. In the **Review plan** phase, user only need to perform one exponentiation operation which can be transformed into multiplication as in Lemma 1. So, compared to Wang *et al.* [13], our scheme costs user $\{m{\cdot}c+O(\ln(p-1))\}$ more multiplicative operations which is nearly the same magnitude with $m{\cdot}c$ multiplications.

By amortising the computation overhead to TPA, user is relaxed from the heavy burden of processing feedback and meanwhile ensures the security of auditing.

- ***Cloud server's computation overhead:*** As server in our scheme performs the same operations with server in Wang *et al.* [13], namely generating proof response and taking mask technique. Here, we omit the comparison between ours and Wang *et al.* [13].

For comparison simplicity, we let Wang *et al.* [13] be the benchmark. Table 2 presents the performance comparison between various schemes where Mults means multiplications and pairs means pair operations.

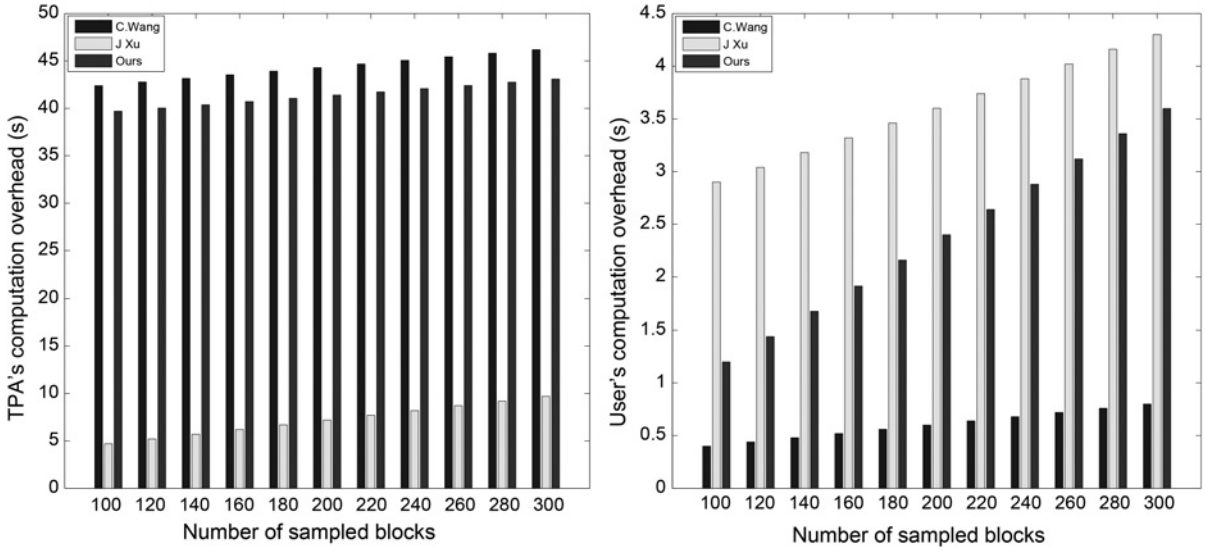

Fig. 2 *Comparison on TPA's and user's overhead between three schemes when m = 100, n = 1000*

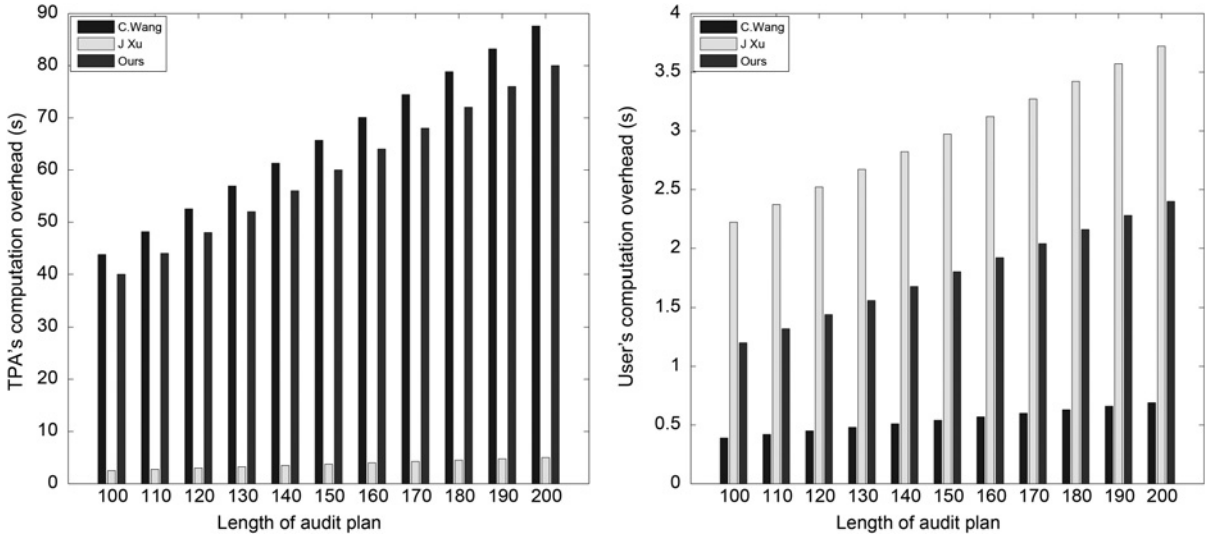

Fig. 3 *Comparison on TPA's and user's overhead between three schemes when c = 100, n = 1000*

4.2.2 Running time analysis: We will focus on the cost of user's and TPA's overhead among various schemes. Our experiment is conducted using C++ and Compiler Version G++ (4.7.1) on a Linux OpenSUSE (12.5) with Intel Core (i5–2450 M) running at 2.5 GHz CPU and 8 G (DDR3) Memory. Algorithms are implemented using open source library Pairing-Based Cryptography library version 0.4.19. The elliptic curve utilised in the experiment is a MNT curve, with base field size of 160 bits and the embedding degree 12. The security level is chosen to be 80 bit, which means $|p| = 160$. By default, we set $m = 100$, $c = 100$, $n = 1000$, $|p| = 160$ and the block size $|F_i| = 1$ kB.

We first investigate the effect of number of sampled blocks c on the running time of the TPA and user. We vary this parameter c, while fixing the other parameters at default value. Fig. 2 shows that it costs Wang *et al.* [13] and ours significantly more time than Xu [28] in terms of TPA's overhead, when c changes from 100 to 300. It can be easily explained that TPA in Wang *et al.* [13] and ours involves in each audit task, while it does not in Xu [28]. In terms of user's overhead, Fig. 2 also illustrates that user in Xu [28] and ours need more time to perform the verification task than Wang *et al.* [13] and ours outweighs Xu [28]. As the majority of computation overhead in ours is amortised to TPA while user in Xu [28] takes this burden.

We then investigate the effect of length of audit plan m on the running time of the TPA and user. We vary this parameter m, while fixing the other parameters at default value. Fig. 3 shows approximately the same as Fig. 2, which thus we omit here. Seeing Table 2, Figs. 2 and 3, we can conclude that our proposed scheme outweighs Wang *et al.* [13] and Xu [28], meaning ours can simultaneously defend against frame and collude attack and performance overhead also lies in the midst between the other two schemes.

5 Conclusion

In this paper, we investigate the problem of TPA-oriented security visibility in cloud data storage, which significantly impacts the adoption of cloud computing. To achieve the assurances of cloud data integrity and availability and avoid the frame (or collude) attack from malicious TPA, we proposed an effective and lightweight protocol where user himself executes the final verification task and TPA plays the role of processing proof and aggregating feedbacks. In addition, we adopt multi-TPAs to implement the same computational audit and let user check the final verification equation for preventing frame attack and collude attack, respectively. Through detailed security and performance analysis, we show that our scheme is more secure than other related schemes and lightweight under the assumption that TPA is malicious in some situations.

6 Acknowledgment

The work is supported by the Natural Science Foundation of China nos. 60372039 and 61103191.

7 References

1 Amazon S3 Versioning 2010. Available at: http://www.doc.s3.amazonaws.com/betadesign/Versioning.html [accessed 02.11.10]
2 Helft, M.: 'Google confirms problems with reaching its services' (The New York Times, 2009), http://www.developmentguruji.com/news/99/Googleconfirms-problems-with-reaching-its-services.html
3 Stern, A.: 'Update from amazon regarding friday S3 downtime' (CenterNetworks, 2008), http://www.centernetworks.com/amazon-s3-downtime-update
4 Wingfield, N., Microsoft.: 'T-Mobile Stumble with Sidekick Glitch' (The Wall Street Journal, 2009), http://www.online.wsj.com/article/SB10001424052748703790404574467431941990194.html
5 Ateniese, G., Burns, R., Curtmola, R., *et al.*: 'Provable data possession at untrusted stores'. Proc. 14th ACM Conf. on Computer and Communications Security, New York, Alexandria, VA, October 2007, pp. 598–609
6 Erway, C., Kupcu, A., Papamanthou, C., Tamassia, R.: 'Dynamic provable data possession'. Proc. 16th ACM Conf. on Computer and Communications Security, New York, Chicago, November 2009, pp. 213–222
7 Curtmola, R., Khan, O., Burns, R., Ateniese, G.: 'MR-PDP: multiple-replica provable data possession'. Proc. ICDCS'08, 2008, pp. 411–420
8 Curtmola, R., Khan, O., Burns, R.: 'Robust remote data checking'. Proc. Fourth ACM Int. Workshop on Storage Security and Survivability, StorageSS, 2008, pp. 63–68
9 Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: 'Scalable and efficient provable data possession'. Proc. Securecomm, 2008, pp. 1–10
10 Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: 'Enabling public verifiability and data dynamics for storage security in cloud computing'. Proc. 14th European Symp., Research in Computer Security (ESORICS 09), 2009, pp. 355–370
11 Chen, B., Curtmola, R., Ateniese, G., Burns, R.: 'Remote data checking for network coding-based distributed storage systems'. Proc. 2010 ACM

Workshop on Cloud Computing Security Workshop (CCSW'10), 2010, pp. 31–42
12 Wang, C., Wang, Q., Ren, K., Lou, W.: 'Ensuring data storage security in cloud computing'. Proc. 17th Int. Workshop Quality of Service (IWQoS'09), July 2009, pp. 1–9
13 Wang, C., Wang, Q., Ren, K., Lou, W.: 'Privacy-preserving public auditing for data storage security in cloud computing'. InfoCom2010, IEEE, March 2010, pp. 525–533
14 Wang, C., Ren, K., Lou, W., Li, J.: 'Towards publicly auditable secure cloud data storage services', *IEEE Netw. Mag.*, 2010, **24**, (4), pp. 19–24
15 Wang, C., Wang, Q., Ren, K., Cao, N., Lou, W.: 'Toward secure and dependable storage services in cloud computing', *IEEE Trans. Serv. Comput.*, 2012, **5**, (2), pp. 220–232
16 Wang, C., Chow, S.S.W, Wang, Q., Ren, K., Lou, W.: 'Privacy-preserving public auditing for secure cloud storage', *IEEE Trans. Comput.*, 2013, **62**, (2), pp. 362–375
17 Hao, Z., Zhong, S., Yu, N.: 'A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability', *IEEE Trans. Knowl. Data Eng.*, 2011, **23**, pp. 1432–1437
18 Zhu, Y., Wang, H., Hu, Z., Ahn, G.J., Hu, H., Yau, S.S.: 'Dynamic audit services for integrity verification of outsourced storages in clouds'. Proc. 2011 ACM Symp. on Applied Computing (SAC'11), 2011, pp. 1550–1557
19 Zhu, Y., Hu, H., Ahn, G., Yu, M.: 'Cooperative provable data possession for integrity verification in multi-cloud storage', *IEEE Trans. Parallel Distrib. Syst.*, 2012, **23**, (12), pp. 2231–2244
20 Sebe, F., Domingo, J.F., Martinez, A.B., Deswarte, Y., Quisquater, J.: 'Efficient remote data possession checking in critical information infrastructures', *IEEE Trans. Knowl. Data Eng.*, 2007, **20**, (8), pp. 1034–1038
21 Yang, K., Jia, X.: 'An efficient and secure dynamic auditing protocol for data storage in cloud computing', *IEEE Trans. Parallel Distrib. Syst.*, 2013, **24**, (9), pp. 1717–1726
22 Juels, A., Kaliski, B.S., Pors, Jr.: 'Proofs of retrievability for large files'. Proc. 14th ACM Conf. on Computer and Communications Security (CCS'07), 2007, pp. 584–597
23 Shacham, H., Waters, B.: 'Compact proofs of retrievability'. Proc. 14th Int. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT'08), 2008, pp. 90–107
24 Dodis, Y., Vadhan, S., Wichs, D.: 'Proofs of retrievability via hardness amplification'. Proc. Sixth Theory of Cryptography Conf. on Theory of Cryptography (TCC'09), 2009, pp. 109–127
25 Bowers, K.D., Juels, A., Oprea, A.: 'Hail: a high-availability and integrity layer for cloud storage'. ACM Conf. on Computer and Communications Security, 2009, pp. 187–198
26 Bowers, K.D., Juels, A., Oprea, A.: 'Proofs of retrievability: theory and implementation'. Proc. 2009 ACM Workshop on Cloud Computing Security (CCSW'09), 2009, pp. 43–54
27 Zheng, Q., Xu, S.: 'Fair and dynamic proofs of retrievability'. Proc. first ACM Conf. on Data and Application Security and Privacy (CODASPY'11), 2011, pp. 237–248
28 Xu, J.: 'Auditing the auditor: secure delegation of auditing operation over cloud storage'. Proc. IACR Cryptology ePrint Archive, 2011, p. 304