Auditing the Auditor: Secure Delegation of Auditing Operation over Cloud Storage

Jia Xu
National University of Singapore
xujia@comp.nus.edu.sg

ABSTRACT

In cloud storage service, users upload their data together with authentication information to cloud storage server. To ensure the availability and integrity of users' data stored in the cloud storage, users need to verify the cloud storage remotely and periodically, with the help of the pre-stored authentication information and without storing a local copy of the data or retrieving back the data during verification. Public verification enables a third party auditor (TPA), on the behalf of the data owner, to verify the integrity of cloud storage with the data owner's public key. In this paper, we propose a method that allows the data owner to delegate the auditing task to a potentially untrusted third party auditor in a secure manner: (1) The data owner can verify whether the TPA has indeed performed the specified audit task; (2) The data owner can verify whether the TPA did the audit task at the right time specified by the data owner; (3) The confidentiality of the data is protected against the TPA. Our method also enables a TPA to re-outsource the audit task.

Keywords

Authentication, Proof of Retrievability, Secure Cloud Storage, Secure Delegation of Auditing

1. INTRODUCTION

Recently, cloud computing is receiving more and more attentions, from both industrial and academic community. Cloud computing separates usage of IT resources from their management and maintenance, so that users (e.g. individuals, small companies and non-IT companies) can focus on their core business and leave the expensive maintenance of IT services to cloud service provider which has the expertise and knowledge to provide and maintain huge amount of IT resource. Just like a double-bladed sword, cloud computing also brings in many new security challenges on protecting the integrity and privacy of users' data in the cloud.

Since Ateniese [1] and Juels [4], several works aim to provide a method to efficiently verify the integrity of user's data stored in could storage server remotely, without retrieving users' data back and without keeping a local copy of user's data. Various settings are studied, including public verification [1], support of dynamic operation [3], multiple server [2], multiple clients [], and so on.

Public verifiable remote integrity check relaxes 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. Furthermore, several methods even support to verify multiple users' data together in batch.

However, without proper enforcement, public verifiability would give users a *false* impressions that their data were safe in the cloud storage. There are many ways that public verifications can be misused, for example:

- Too many volunteers help to verify a CSP's storage remotely, bringing in too much unnecessary burden on the CSP and possibly resulting in Denail of Service attack to the CSP;
- In contrast to the above case, many volunteers give up in verification, wishfully thinking that someone else will do the verifications;
- The distribution of verification time is uneven, too many verifications for some periods and too few verifications for other periods;
- Even worse, a dishonest CSP may create a lot of sybil identities, who are announcing that they are volunteered to check the cloud storage periodically.

In this paper, we attempt to provide a solution which enables the data owner to securely delegate the auditing task to a potentially untrusted third party auditor (TPA). In other words, our method allows the data owner to audit the auditors of a cloud storage. Our solution also allows the TPA to re-outsource the auditing task.

Contribution

1. Our work is among the first few (if not the first one) that formulate the security requirements with time in concern in delegation of audit tasks over cloud storage service. In our formulation, a delegation of audit is "secure", if
   - the data owner can verify whether TPA has indeed performed the audit task specified by the data owner;

- the data owner can verify whether TPA did perform the audit task at the right time specified by the data owner;
- the confidentiality of the data is protected against the TPA and/or the CSP.

2. We provide a solution that fulfills the security requirements in the formulation.
3. We analyze that our assumption is almost minimum and the performance is efficient.
4. We prove that our solution is secure.

2. PRELIMINARY AND BACKGROUND

We restate the $\mathcal{POR}$ [4, 5] model as below, with slight modifications on notations. We adopt the 1-round verify-prove version in Juels [4] for simplicity.

DEFINITION 1 ($\mathcal{POR}$ [4, 5]). *A* Proof Of Retrievability *($\mathcal{POR}$) scheme consists of four algorithms* KeyGen, DEnc, Prove, Verify*:*

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$. *Given security parameter κ, the randomized key generating algorithm outputs a public-private key pair (pk, sk)*
- $\hat{M} \leftarrow \mathsf{DEnc}(M, sk)$. *Given a data file M and the private key sk, the encoding algorithm* DEnc *produces the encoded file $\hat{M}$.*
- $r \leftarrow \mathsf{Prove}(\hat{M}, c, pk)$*: Given an encoded file $\hat{M}$, a challenge c and the public key pk, the prover algorithm* Prove *produce a response/proof r.*
- $\{\mathtt{accept}, \mathtt{reject}\} \leftarrow \mathsf{Verify}(c, r, pk)$*: Given a challenge c, a response/proof r and the public key, the verifying algorithm* Verify *will output either* accept *or* reject.

2.1 Shacham and Waters's Scheme

We describe Shacham and Waters's public verifiable $\mathcal{POR}$ scheme [5] as below.

$\mathsf{KeyGen}(1^\kappa)$: Choose a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where both $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic multiplicative group of prime order p. Choose two random generators g, u of the group $\mathbb{G}$. Choose a private key x at random from $\mathbb{Z}_p$. Set the public key as $(u, v = g^x) \in \mathbb{G}^2$.

$\mathsf{DEnc}(M, sk)$: The file M is divided into blocks $m_1, m_2, \ldots, m_\ell$, $m_i \in \mathbb{Z}_p$, and for each block m_i, generate an authentication tag σ_i as below

$$\sigma_i = \mathsf{Tag}(m_i, i) = (\ h(i)u^{m_i} \ )^x \in \mathbb{G}.$$

Alice sends the encoded file $\hat{M} = \{(m_i, \sigma_i)\}$ to Bob.

$\mathsf{Prove}(\hat{M}, c, pk)$: Upon receiving a challenge $c = \{(i, \nu_i) \in [\ell] \times \mathbb{Z}_p\}$ from the verifier, the prover (i.e. Bob) computes and sends back the response (σ, μ):

$$\sigma \leftarrow \prod_{(i,\nu_i)\in c} \sigma_i^{\nu_i}, \quad \mu \leftarrow \sum_{(i,\nu_i)\in c} \nu_i m_i \mod p. \tag{1}$$

$\mathsf{Verify}(c, r = (\sigma, \mu), pk)$: Verifier outputs `accept` if

$$e(\sigma, g) \stackrel{?}{=} e(u^\mu \prod_{(i,\nu_i)\in c} h(i)^{\nu_i}, \ v) \tag{2}$$

Otherwise, the verifier outputs `reject`.

2.2 Blind Technique

Wang *et al.* [7, 6] proposed a blind technique in addition to Shacham and Water's scheme [5], attempting to achieve privacy-preserving third party auditing. With their blind technique, the prover masks the proof for a challenge with some randomness and the verifier is still able to verify the validity of the masked proof.

The key generation and data encoding algorithms of [7, 6] are the same as Shacham and Water's scheme. We describe their prove-verify algorithms as below:

BlindProve : Upon receiving a challenge $c = \{(i, \nu_i) \in [\ell] \times \mathbb{Z}_p\}$ from the verifier, Bob computes (σ, μ) as in equation (1). Next, Bob blind[1] μ to generate μ' in this way: Choose r at random from $\mathbb{Z}_p$ and compute $R = e(u, v)^r$ and $\gamma = h(R) \in \mathbb{Z}_p$ where $h : \mathbb{G}_T \to \mathbb{Z}_p$ is some secure hash function:

$$\mu' = r + \gamma\mu \mod p \tag{3}$$

Bob sends (σ, μ', R) to the verifier.

BlindVerify : Verifier performs the following equality check:

$$R \cdot e(\sigma^\gamma, g) \stackrel{?}{=} e(u^{\mu'} \cdot \left(\prod_{(i,\nu_i)\in c} h(i)^{\nu_i} \right)^\gamma, v) \tag{4}$$

2.2.1 Exclude-Attack

We found that a curious TPA can verify whether the hidden linear combination μ of file blocks m_i is equal to any particular value, using only information ((μ', R) and public key) that he/she is allowed to access.

TPA has (μ', R, p). Given a guess $\hat{\mu}$, TPA computes $\gamma = h(R)$ and finds a value $\hat{r} \in \mathbb{Z}_p$ by solving the equation

$$\mu' = \hat{r} + \gamma\hat{\mu} \mod p.$$

TPA can confirm that $\hat{\mu} = \mu$, if and only if $e(u, v)^{\hat{r}} = R$. If the selected file blocks m_i's have low entropy, then TPA could be able to find the values of m_i by brute-force search. This violates the authors' claim that their scheme does not assume additional property on the data file.

3. FORMULATION

3.1 Motivations and Observations

Integrity verification and data accessing cannot be completely separated.. Before accessing data, the user has to verify the downloaded data locally, even if he/she has performed the auditing task periodically or delegated the audit task to a third party. This is in contrast with the claim in [TPA, infocom 10]. Without local checking, a malicious CSP can inevitably cheat and provide the user altered data with non-negligible probability, no matter what remote integrity check schemes are deployed. Suppose the data owner/verifier will initial $poly(\kappa)$ number of interactions with the CSP, and each interaction is either for verification or data retrieval. A

[1]Note that the newer eprint paper [6] gives an improved version of blind technique than the INFOCOM conference paper [7]. Here we presents the eprint version. In the conference paper version, in equation (3), the positions of r and μ are swapped and thus the verification is different accordingly.

malicious CSP may cheat in this way: Choose a random i from interval $[1, poly(\kappa)]$, forge a wrong response in i-th interaction and follow the protocol honestly in all other sessions. Such CSP will always win by feeding data owner with wrong data, with non-negligible probability $N/poly(\kappa)$ where N is the number of retrievals. Furthermore, if CSP is able to distinguish verification from retrival, he/she will win with even higher probability.

Timing is essential in remote integrity check of cloud storage. If data are corrupt and no longer "retrievialbe" in the cloud server, the data owner should know this at soon as possible. So that he/she can take counter-measure in time, to minimize the loss. Without timing concern, users can always download the (partial) data from CSP, verify the integrity of data locally, before using the data. If data owner eventually does not retrieval the data, it does not matter whether the data in the cloud is intact or not. A dishonest TPA may have incentive to perform all audit tasks specified by the data owner within a short period, to decrease his/her Internet connection time.

Auditors themselves should be audited. Auditors may have incentive to execute a partial audit task specified by the data owner, or execute the audit task at time which fit their own interests. Furthermore, the CSP may collude with the auditors or just create some sybil identites, who are vulunteer to be the auditors. In order to ensure they faithfully accomplish their promise on auditing the cloud storage at right time, auditors themselves should be audited.

Scope. : DOS (Denial of Service) attack is out of range of this paper (1) If the DOS attack is launched by the Cloud Storage Server or the designated third party auditor, there is no technical way to resolve it. In real world applications, this could be handled by law system. (2) If the DOS attack is launched by outside attackers, besides plenty of existing solutions against DOS attack, our system may have a simple solution: the Cloud Storage Server only responds to auditors who are authorized by the data owner.

Frame attack between Cloud Storage Server and Third Party Auditor: We will deal with it.

Frame attack by data owner can be dealed with.

3.2 Limitation of Previous work

Wang *et al.* [7] proposed to a method to protect data confidentiality against the TPA. However, their security model is weak:

- In their model, both CSP and TPA are semi-trusted. Precisely, they trust TPA in auditing and trust CSP in data confidentiality; they do not trust TPA in data confidentiality and do not trust CSP in maintaining data integrity.
- Their privacy protection is also weak. Although they showed that their blind technique prevents TPA from recovering the original data through auditing process (Theorem 2 of their paper), they did not analyze whether their blind technique reveals any *partial* information about the original data. We found that, in their scheme, the TPA is able to verify that whether the original data equal to any particular value, with only information that he/she is allowed to access. Such ability to evaluate equality predicate over blinded data could be a serious vulnerability when the entropy of some data blocks is very low. Although this issue can be mitigated by compressing the whole data file before outsourcing, this potential weakness may suggest that a stronger privacy requirement is desired in such applications.

3.3 Definitions

System Model. Our system consists of three parties: data owner Alice who has a large amount of data to backup, cloud service provider (CSP) Bob who provides cloud storage service, third party auditor Charlie who promises to help Alice to audit the integrity of Bob's cloud storage periodically. Typically, Alice is equipped with a low end computer device (e.g. a smart phone), Bob and Charlie have more powerful computation resource.

Trust mode and Security goal. We trust in Alice, and do not trust in Bob or Charlie. We aim to verify whether Bob and Charlie have provided specified services to Alice as they promised, and provide privacy protection on the outsourced data against TPA Charlie or/and CSP Bob.

We assume the communication channels are secure. As we have discussed, both DOS attack and frame attack between Bob and Charlie are out of our focus. We remark that, these are just for simplicity of presentation, instead of limitation of our solution.

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 below.

Definition 2 (Audit Plan). *An audit plan of size m w.r.t. a $\mathcal{POR}$ scheme $\mathcal{E}$ is a pair of vectors $\{\mathbf{T}, \mathbf{C}\} \in \mathbb{T}^m \times \mathbb{C}^m$, where $\mathbb{C}$ is the set of all possible chanllenges in $\mathcal{E}$, $\mathbf{T} = (t_1, \ldots, t_m) \in \mathbb{T}^m$ is a vector of time points t_i's, such that $t_1 \leq t_2 \ldots \leq t_m$, and $\mathbf{C} = (c_1, \ldots, c_m) \in \mathbb{C}^m$ is a vector of m $\mathcal{E}$-challenges.*

In addition to a $\mathcal{POR}$ scheme, we define an "audit protocol". Let Δ be a time interval aggreed by all of the three parties, such that one audit task should be complished within Δ time, including all computation time and network transimission time and with reseasonable safe margin.

Definition 3 (Audit Protocol). *An* audit protocol *w.r.t. a $\mathcal{POR}$ scheme $\mathcal{E}$ consists of three phases* (ReleasePlan, ExecutePlan, ReviewPlan)*, and each phase is described as below.*

- ReleasePlan*: The data owner Alice chooses an audit plan $\mathbf{P} = \{\mathbf{T} = (t_1, \ldots, t_m), \mathbf{C} = (c_1, \ldots, c_m)\} \in \mathbb{T}^m \times \mathbb{C}^m$ of size m, and sends an encoded plan $\hat{\mathbf{P}}$ of the audit plan $\mathbf{P}$ to TPA Charlie.*
- ExecutePlan*: At each time t_i, TPA Charlie is able to recover the corresponding $\mathcal{E}$-challenge c_i from $\hat{\mathbf{P}}$. TPA Charlie sends c_i to CSP Bob and Bob replys with response $r_i \leftarrow \mathsf{Prove}(\hat{M}, c_i, pk)$, where the encoded file $\hat{M}$ is generated by Alice and stored by Bob. Charlie commits r_i in the allowed time interval $[t_i, t_i + \Delta]$.*
- ReviewPlan*: After time $t_m + \Delta$, Alice may verify whether Charlie has indeed performed the i-th audit task in the*

plan $\mathbf{P}$ *during the specified time interval* $[t_i, t_i + \Delta]$. *Possibly, Alice may interact with CSP Bob. If the verification succeeds, Alice outputs* `accept`*; otherwise outputs* `reject`.

It is required that, the data owner Alice is only involved in $O(1)$ *rounds of interactions during all of the three phases.*

DEFINITION 4 (TIMED-SECURE). *Let* $\mathcal{P} =$ (ReleasePlan, ExecutePlan, ReviewPlan) *be a audit protocol.*

- $\mathcal{P}$ *is Backward-Timed-Secure: If there exists some* $j \in [m]$*, such that TPA Charlie performs the* j*-th audit task (more precisely, TPA can obtain the challenge* c_j*) before time* t_j*, then with overwhelming high probability, Alice will reject in the* ReviewPlan *phase.*
- $\mathcal{P}$ *is Forward-Timed-Secure: If there exists some* $j \in [m]$*, such that TPA Charlie performs the* j*-th audit task (more precisely, TPA obtained the response* r_j *for challenge* c_j*) after time* $t_j + \Delta$*, then with overwhelming high probability, Alice will reject in the* ReviewPlan *phase.*

DEFINITION 5. *Let* $\mathcal{E}$ *be a secure* $\mathcal{POR}$ *scheme. An audit protocol* (ReleasePlan, ExecutePlan, ReviewPlan) *w.r.t.* $\mathcal{E}$ *is verifiable, if the following conditions are satisfied:*

- *Correctness: If both Bob and Charlie follow the protocol exactly and Bob's storage is intact, then Alice will always accept in the* ReviewPlan *phase.*
- *Soundness:*
  - *The audit protocol is both Backward-Timed-Secure and Forward-Timed-Secure.*
  - *If for some* $j \in [m]$*, the response* r_j *presented by TPA for challenge* c_j *is not valid (i.e. cannot pass the* $\mathcal{POR}$ *verification), then with overwhelming high probabiity, Alice will reject in the* ReviewPlan *phase.*

DEFINITION 6 (SEMANTIC-SECURE). *A* $\mathcal{POR}$ *scheme* $\mathcal{E} =$ (KeyGen, DEnc, Prove, Verify) *is* semantic secure *against the TPA, if: Alice runs the key generating algorithm* KeyGen *to produce a public-private key pair* (pk, sk)*. Given the public key* pk*, adversary* $\mathcal{A}$ *chooses two equal length data files* M_0, M_1*. Alice chooses a random bit* $b \in \{0, 1\}$ *and encodes* M_b *to obtain* M^* *using the data encoding algorithm* DEnc*. Alice sends the encoded data file* M^* *to CSP Bob. The adversary checks for integrity of Bob's storage remotely, using* $\mathcal{POR}$*-verification. Eventually, adversary outputs a guess* $b' \in \{0, 1\}$*. We have* $\mathsf{Pr}[b = b'] <= 1/2 + negl(\kappa)$.

4. OUR SCHEME

Background on Brent and Shacham Scheme ReleasePlan, ExecutePlan, ReviewPlan

TIMESERVER,RECEIVESERVER

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.

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 $(t, \texttt{Rev}, \texttt{Snd}, \texttt{Msg})$ 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.

We remark that, both time-server and receive-server are "independent" on our scheme, in the sense that: (1) both servers are not set-uped by our scheme; (2) both servers provide generic service and are not exclusive or dedicated to our scheme; (3) both servers are always unaware of the existence of our scheme.

Let $\mathcal{E} =$ (KeyGen, DEnc, $\langle \mathcal{P}, \mathcal{V} \rangle$

- Setup: Alice runs key generating algorithm $\mathcal{E}$.KeyGen to generate public-private key pair (pk, sk). Given a data file, Alice preprocesses the data file in two steps:
  - encode the data file using error-correcting code (e.g. Reed Soloman Code)
  - divide the data file into blocks and encrypt each block using a semantic secure public key encryption scheme (or a symmetric encryption scheme, e.g. AES).

  Let $X = (x_1, x_2, \ldots, x_\ell)$, $x_i \in \mathbb{Z}_p$ for each $1 \leq i \leq \ell$, be the resulting file. Then Alice runs data encoding algorithm $\mathcal{E}$.DEnc to generate authentication tags $\Phi = \{\sigma_i\}$. Alice sends $\{(i, x_i, \sigma_i)\}$ to Bob.
- ReleasePlan: Alice choose m time points $t_1 \leq t_2 \ldots \leq t_m$ from $\mathbb{T}$, and chooses m $\mathcal{E}$-challenges $(c_1, \ldots, c_m)$ at random. For each $\lambda \in [m]$, Alice encrypts c_λ using the Timed-Release Encryption scheme:

  $$C_\lambda \leftarrow \mathsf{TEnc}(c_\lambda, t_\lambda)$$

  Alice sends the coded audit plan $\{(t_\lambda, C_\lambda) : 1 \leq \lambda \leq m\}$ to TPA Charlie.
- ExecutePlan: At time t_λ, $1 \leq \lambda \leq m$, TPA Charlie receives the decryption key K_λ for t_λ, and decrypts C_λ to obtain the challenge c_λ:

  $$c_\lambda \leftarrow \mathsf{TDec}(C_\lambda, K_\lambda)$$

  TPA Charlie, who is taking the role of verifer, interacts with the CSP Bob, who is taking the role of prover, to execute the interactive algorithm $\langle \mathcal{P}(c_\lambda), \mathcal{V}(X, \{\sigma_i\}) \rangle$. TPA Charlie obtains reply $r_\lambda = (\Sigma_\lambda, \mu_\lambda)$ for the challenge c_i from CSP Bob. TPA Charlie sends $\texttt{Msg}_\lambda = (\lambda, r_\lambda)$ to Receive Server.

  An authorized verifier may interact with CSP Bob by running algorithm P, V to audit the integrity of Alice's data in Bob's storage.
- ReviewPlan:
  - Alice retrieves $\{(T_\lambda, \texttt{Rev} = Alice, \texttt{Snd} = Charlie, \texttt{Msg}_\lambda = (\lambda, r_\lambda)) : 1 \leq \lambda \leq m\}$ from the Receive Server.
  - Alice chooses a vector $\boldsymbol{w} = (w_1, \ldots, w_m)$ of size m at random from $(\mathbb{Z}_p)^m$. Construct a matrix $\boldsymbol{M} = (m_{\lambda,j})_{m \times \ell}$, such that $m_{\lambda,j} = \nu_j$ if $(j, \nu_j) \in c_\lambda$, otherwise $m_{\lambda,j} = 0$. Let the vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_\ell) = \boldsymbol{w} \times M$ be the multplication of vector $\boldsymbol{w}$ and matrix M.

– Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$. Compute $\mu^* = \langle \boldsymbol{w},\ \boldsymbol{\mu} \rangle$ and $\Sigma^* = \prod_{\lambda=1}^{m} \Sigma_\lambda^{w_\lambda}$.

– Alice verifies (Σ^*, μ^*) with $c^* = \{(i, \gamma_i) : 1 \leq i \leq \ell\}$ as chanllenge, using $\mathcal{E}.\mathsf{Verify}$ algorithm:

$$e(\Sigma^*, g) \stackrel{?}{=} e(u^{\mu^*} \prod_{(i,\gamma_i) \in c^*} h(i)^{\gamma_i},\ v)$$

5. ANALYSIS

Claim 1. *Timed-Released Encryption is necessary and sufficient for [backward-security] After receiving the coded audit-plan from data owner Alice, TPA Charlie must not obtain the challenge c_i before time t_i. Otherwise, Charlie may send c_i to CSP Bob to perform the i-th audit task earlier than specific time t_i, which violates the [backward-security]. After time t_i, an honest Charlie should be able to obtain c_i, so that she can perform the i-th audit task as required by Alice. Thus a timed-released encryption which allows a ciphertext to be decrypted only after a time point which is specified during encryption, is necessary. The other direction ...*

Claim 2. *Timed-Released Encryption is insufficient for [forward-security] At the first glance, one may think [forward-security] can be achieved similarly as [backward-security]. Time-Released Encryption can only control when TPA Charlie can access certain information. A dishonest TPA Charlie could always obtain the required information (i.e. the challenge) at the right time. After obtaining the required information, Charlie could perform the actual audit action at any time.*

5.1 Optimal:

Claim 3. *The coded audit plan contains at least m ciphertexts encrypted using Time-Released Encryption scheme.*

Claim 4. *TPA Charlie has to commit at least m times.*

5.2 Security

Theorem 1. *Our audit protocol is veririable (as defined in Definition 5) and semantic secure (as defined in Definition 6).*

Due to the exclude-attack in Section 2.2.1, Wang *et al.* [7] is not semantic secure.

6. CONCLUSION

We proposed a solution that allows the owner of data stored in a cloud storage to delegate the auditing task to a potentially untrusted third party verification in a secure way. That is, the data owner can verify whether the TPA did perform the audit task at the right time as specified by the data owner. In another words, we provide a method allowing the data owner to audit to the auditor.

7. REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *CCS '07: ACM conference on Computer and communications security*, pages 598–609, 2007.

[2] R. Curtmola, O. Khan, R. Burns, and G. Ateniese. MR-PDP: Multiple-Replica Provable Data Possession. In *ICDCS '08: International Conference on Distributed Computing Systems*, pages 411–420, 2008.

[3] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. In *CCS '09: ACM conference on Computer and communications security*, pages 213–222, 2009.

[4] A. Juels and B. S. Kaliski, Jr. Pors: proofs of retrievability for large files. In *CCS '07: ACM conference on Computer and communications security*, pages 584–597, 2007.

[5] H. Shacham and B. Waters. Compact Proofs of Retrievability. In *ASIACRYPT*, pages 90–107, 2008.

[6] C. Wang, S. S.-M. Chow, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for secure cloud storage. Cryptology ePrint Archive, Report 2009/579, 2009. `http://eprint.iacr.org/`.

[7] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In *IEEE INFOCOM*, pages 525–533, 2010.