A Key Management and Public Auditing Scheme for Outsourced Data

Changsheng Wan[1], Juan Zhang[2], Lin Zhou[1], Zhongyuan Qin[1]

[1]School of Information Science and Engineering Southeast University, Nanjing, China
[2]accounting Department Nanjing University, Nanjing, China
Wan.changsheng@163.com

Abstract - Public auditing has vital significance in cloud computing. However, current auditing scheme are costly. This paper brings out an AAA based public auditing architecture for cloud computing, which is much more efficient than current schemes.

Index Terms - Cloud Computing, Publicly Auditable, Proof-of-storage

I. Introduction

Cloud Computing is the long dreamed vision of computing as a utility, where data is centralized or outsourced into the cloud. Its benefits are obvious: relief of the burden of storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, personnel maintenance, and so on [1]. However, the fact that users no longer have physical possession of the large size of outsourced data introduces new security issues.

The first issue is data integrity. Although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they still face a broad range of both internal and external threats to data integrity [2-4].

The second issue is unfaithful cloud server providers (CSP). For benefits of their own, there are various motivations for CSPs to behave unfaithfully toward cloud customers regarding the status of their outsourced data. Examples include CSPs, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed [5], or even hiding data loss incidents to maintain a reputation [6].

As data owners no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data integrity protection cannot be directly adopted [5, 6].

In particular, simply downloading the data for its integrity verification is not a practical solution due to the high cost of input/output (I/O) and transmission across the network.

Moreover, it is often insufficient to detect data corruption only when accessing the data, as it does not give correctness assurance for un-accessed data and might be too late to recover the data loss or damage [7].

In addition, from the system usability point of view, data owners should be able to just use cloud storage as if it is local, without worrying about the need to verify its integrity. Therefore, an external third party auditor (TPA) is required.

Based on the audit result from a TPA, the released audit report would not only help owners to evaluate the risk of their subscribed cloud data services, but also be beneficial for the cloud service provider to improve their cloud based service platform [8].

The reference [7] defines the publicly auditable architecture of cloud data storage service. In the architecture, there are four different entities: data owner, user, cloud server (CS), and TPA. The TPA is a trusted entity that has expertise and capabilities to assess cloud storage security on behalf of a data owner upon request. The data owner may represent either the individual or the enterprise customer. Only the data owner can dynamically interact with the CS to update her stored data, while users just have the privilege of file reading.

Under this architecture, there are a lot of schemes for remotely stored data integrity protection under different systems and security models [9–13]. Those schemes are bilinear map based schemes, which are costly. [11] discussed a non-public key scheme, but it is not a publicly auditable scheme.

This paper analyzes the efficiency and security of those schemes and brings out an AAA based publicly auditable scheme, which is much more efficient than current schemes.

II. Efficiency Analysis of Current Bilinear Map Based Schemes

For analyzing the efficiency of bilinear map based schemes, this paper selects reference [13] as an example. The sketch of [13] is summarized as follows:

Signature process:

The data owner randomly chooses u, x and v ($u \in G_1$, $x \in Z_p, v = g^x$), where p is the prime order, g is the generator of G_2, and G_1, G_2 are groups for bilinear map. Then it signs data as follows: ($\sigma_0 = (u^m)^x$), where m is the message to be signed. Finally, the data owner sends the signed data and the signature to the cloud server.

Auditing process:

(1) The TPA sends a challenge c to the cloud server.

(2) The cloud server chooses a private key (r), computes: ($R = e(u,v)^r$), and hides r in t as follows: $t = r + h(R)cm$. Then it computes: ($\sigma = (\sigma_0)^c$). Finally, the cloud server sends (t, R, σ) to the TPA.

(3) The TPA verifies $R*e(\sigma^{h(R)},g)=e(u^t,v)$ to ensure the data in the cloud server exists and is not modified by other entities.

The computation cost of the bilinear map scheme relies on the counts of modular exponential operations, which is computed in TABLE 1.

TABLE 1 Counts of modular exponential operations

Data owner	TPA	Cloud server
1	3	2

The TABLE 1 shows that the total computation cost of bilinear map scheme includes 6 modular exponential operations when processing one block of data. This is very costly, especially when there are a variety of data to be audited.

To reduce the auditing cost, this paper designs a symmetric key based scheme, which can be much more efficient than the bilinear map based schemes.

III. Our Scheme

A. Trust Model and Security Goals

Trust Model of the public auditing

There are four entities in our scheme: the cloud server (CS), the third party auditor (TPA), the data owner (DO) and the key server (AAA).

AAA has pre-established trust relationships with CS, TPA and DO respectively. It authenticates and authorizes the DO for the cloud service.

The CS provides cloud service to the DO, after the DO passes the authentication and authorization process. The communications between the CS and DO in our scheme are protected by the secure channel established during authentication. And this secure channel should provide both integrity and confidentiality protection.

The TPA should be authorized by the AAA, before auditing the DO's data in the CS. The TPA has secure channel with the CS. The communications between the TPA and the CS in our scheme are protected by it. And this secure channel should provide both integrity and confidentiality protection.

Security Goals

Our security goal is to ensure that there exists no cheating cloud server that can pass the audit from TPA without indeed storing users' data intact.

The efficiency of this security goal can be evaluated by the following four vectors:

(1) The computation cost should be low.
(2) The communication cost should be low.
(3) The storage cost should be low.
(4) TPA should be stateless, and not need to maintain and update state between audits, since such state is difficult to maintain if the TPA's machine crashes. Statelessness and unbounded use are required for proof-of-storage systems with public verifiability, in which anyone can undertake the role of verifier in the proof-of-storage protocol, not just the user who originally stored the file.

Security Model of Our Scheme

To fulfill our security goals, we define four algorithms in our security model: A_g, S_t, V and P. They are described as follows:

- $A_g(sk,lk,uID)$. It is the algorithm for generating keying materials. (sk) is the keying material generated by this algorithm and will be used by the cloud service. (uID) is the data owner's identity used in the cloud service. (lk) is the long term trust relationship used for generating (sk).

- $S_t(sk,uID,M)$. It is a signature algorithm. (uID) is the data owner's identity. M is the block of data to be signed. The DO uses it for generating signatures from (sk) and M.

- $P(M^*,M,c)$. It is a proving algorithm. The CS generates proof of storages using this algorithm. (M^*) is the output of $S_t(sk,uID,M)$ (e.g. the signature). c is a challenge.

- $V(sk,c,P)$. It is a verification algorithm. The TPA uses it for verifying the proofs from the CS. The verification algorithm uses (sk), and c for verifying whether P is correct.

Based on the four algorithms, our auditing protocol is to check the following equation: ($V(sk,c,P)=True$).

Our security model differs from that defined in [11] mainly in two points:

-In the security model defined in [11], M is not part of P. The CS may delete (M), and only stores M^* to pass the verification. So, we add M to our model to address this issue.

-In the security model defined in [11], c is not part of (P) and (V). The CS may delete (M) and (M^*), and only store $P(t,M^*)$ after a successful auditing process. When the TPA wants to repeat an audit phase, the CS just sends $P(t,M^*)$ back to the TPA. So, the challenge c should be changed every time.

B. Our Scheme

Our scheme includes three parts: the initialization process, the signature process and the auditing process.

Step 1) the initialization process.

When the data owner requests cloud service, the AAA server authenticates it using an authentication mechanism such as the extensible authentication protocol [14, 15].

After authentication, the extensible master session key (EMSK) [14] is generated between the AAA and the DO. Since the EMSK cannot be transported outside the AAA and

the DO [14], the AAA and the DO derives a root key k_{ccr} for data integrity from the EMSK respectively as follow:

$k_{ccr} = PRF(EMSK \mid DOID \mid CSID)$, where PRF has the same meaning as defined in [13] (typically PRF is a hash function), DOID is the identifier of the DO and CSID is the identifier of the cloud server.

When the TPA wants to provide auditing process on the DO, it requests k_{ccr} from the AAA over their secure channel.

Therefore, after initialization process, the DO and the TPA share the root key k_{ccr} for auditing.

Using (k_{ccr}), the DO and the TPA can compute $k_{tc} = prf(k_{ccr} \mid DOID \mid CSID \mid MULTIPLICATION)$ and $k_{pro} = prf(k_{ccr} \mid DOID \mid CSID \mid HASH)$ from k_{ccr} respectively, where $DOID$ is the identity of the data owner, $CSID$ is the identity of the cloud server, $MULTIPLICATION$ and $HASH$ are two const strings.

Step 2) the signature process.

In our scheme, the data owner may sign a variety of files, the data owner breaks every file into n blocks: $m_1, m_2, ...m_n \in Z_p$, where p is a prime number. For every block (i), the data owner signs m_i as follows: ($\sigma_i = prf(k_{tc}, i, uID) + k_{pro} m_i \bmod p$), where prf is a keyed hash function.

Then, the DO sends σ_i along with m_i to the CS, and the latter stores them.

Step 3) the auditing process.

During auditing process, the TPA generates a challenge c_i to every block i of every file f_l and sends a subset of $\{c_i, i, f_l\}$ to the CS.

Upon receiving ($\{c_i, i, f_l\}$), the CS computes ($\sigma = \sum c_i \sigma_i \bmod p$) and ($\alpha = \sum c_i m_i \bmod p$). Then it sends $\{\sigma, \alpha\}$ to the TPA.

The TPA verifies $k_{pro}\alpha + \sum c_i prf(k_{tc}, i, uID) - \sigma \equiv 0 \bmod p$

Since

$$\begin{aligned} k_{pro}\alpha + \sum c_i prf(k_{tc}, i, uID) \bmod p &= k_{pro} \sum c_i m_i + \sum c_i prf(k_{tc}, i, uID) \\ &= \sum c_i (k_{pro} m_i + prf(k_{tc}, i, uID)) \\ &= \sum c_i \sigma_i \bmod p \\ &= \sigma \end{aligned}$$

if the CS passed the verification, the TPA can make sure that the CS stores f_l and f_l is not modified by other entities instead of the DO.

IV . Security Analysis of Our Scheme

We evaluate the security of the proposed scheme by analyzing its fulfillment of the security model described in Section 3.1.3. This section is organized as follows: Firstly, we construct security model for our scheme. Then, we prove the security of our scheme based on the security model.

A. Security Model of Our Scheme

Our scheme fulfills the security model defined in 3.1.3:

- $A_g(sk, lk, uID)$. In our scheme, k_{ccr}, k_{tc}, and k_{pro} are sk. Our scheme uses the AAA infrastructure as the cloud computing infrastructure, so the trust relations defined in section 3.1.1 are lk s.

- $S_t(sk, uID, M)$. Similar to [11], we use the linear function defined in step 2) of section 3.2 as our signature algorithm.

- $P(M^*, M, c)$. In our scheme , $\sigma = \sum c_i \sigma_i \bmod p$ and $\alpha = \sum c_i m_i \bmod p$ are our proving algorithms.

- $V(sk, c, P)$. In our scheme , the equation $k_{pro}\alpha + \sum c_i prf(k_{tc}, i, uID) - \sigma \equiv 0$ is our verification algorithm.

B. Security Proof of Our Scheme

For storage correctness goal, we need to prove that the CS cannot generate valid response toward TPA without faithfully storing the data, as captured by Theorem 1.

Theorem 1: There is no attacking algorithm $P'(M^*, c)$ that allows the cloud server to generate a correct proving result ((σ) and (α)), that fulfills with the verification algorithm ($V(sk, c, P)$).

Proof: Our proof includes two steps. Firstly, we prove that the CS cannot generate correct proving result from ($P'(M^*, c)$). Secondly, we prove that without a correct proving result, the verification algorithm $V(sk, c, P)$ will fail. The two steps are described as follows:

Step 1) in our scheme, assuming the CS possesses (σ_i), but does not possess (m_i). Then It can generate σ using (σ_i), but it will not be able to generate α from (σ_i), since α is computed from (m_i), and m_i cannot be computed from σ_i without (k_{tc}) and (k_{pro}). Therefore, using an attacking

algorithm ($P'(M^*, c)$), the attacker can only generate a correct (σ). But it cannot generate (α).

Step 2) we need to prove that without a correct (σ) and (α). The attacker will not pass the verification algorithm ($V(sk, c, P)$). In our scheme, the verification algorithm is the equation ($k_{pro}\alpha + \sum c_i prf(k_{tc}, i, uID) - \sigma \equiv 0$). This is a linear equation. The CS knows: (c_i), (i), uID and σ in the equation. To pass the verification process, the attacker must be able to compute $\alpha' = (\sigma - \sum c_i prf(k_{tc}, i, uID)) / k_{pro}$, and sends α' along with σ to the TPA. Since it does not know (k_{tc}) and (k_{pro}), it cannot compute α' from the verification algorithm.

From step 1 and step 2 above, we can see that our scheme fulfills the theorem 1 (i.e. the storage correctness goal).

C. *Security Strength of Our Scheme*

In our scheme, the prime number p determines the security strength of our signature and auditing algorithm, while the key length of k_{tc} and k_{pro} should be equal to that of (p). Typically, the security strength of a 128-bit p is equal to the widely used AES algorithm.

V . Efficiency Analysis of Our Scheme

We evaluate the efficiency of the proposed scheme by analyzing its fulfillment of the design goals described in Section 3.1.2.

Firstly , we analyze the computation cost of our scheme . In our scheme , the signature cost on the DO is one hash operation and one multiplication operation . The proving cost on the CS is two multiplication operations . The verification cost on the TPA is one hash operation and two multiplication operations . The computation cost of our scheme can be evaluated using the term CPU cycles. This paper takes 128-bit symmetric key cryptographies as an example for evaluating the computation cost of our scheme . Referring to [16], using the Montgomery algorithm , the number of CPU cycles for 128-bit modular multiplication on 32-bit CPU is around 512 . This paper chooses SHA1 as the hash function used in our scheme . Referring to [17] and [18] , the computation cost of hash function in our scheme is $32 + (2+3)*1110 = 5582 cpucycles$ .Referring to [19], modular exponentiation with about 2100 bits will have about the same resistance against attack to that of 128-bit symmetric key cryptographies. The computation cost for 2100-bit RSA on 32-bit CPU is around 450,000 CPU cycles.

Therefore, the computation cost of our scheme compared with the bilinear map based schemes is listed in TABLE 2.

TABLE 2 Computation Cost (unit: CPU cycles)

	Data owner	TPA	Cloud server
Bilinear map based schemes	450,000	1,350,000	900,000
Our scheme	6094	6606	1024

Table 2 shows that the computation cost of our scheme is around $10^{-2} - 10^{-3}$ to that of bilinear map based schemes.

Secondly, we analyze the communication cost of our scheme. There are two messages in our scheme during auditing phase, which is equal to that of bilinear map based schemes. There are five parameters included in the auditing messages of our scheme (e.g. ($\{c_i, i, f_l, \sigma, \alpha\}$)), the total length of the messages is around 128*5=640. The bilinear map based schemes transporting similar parameters. However, as 2048-bit public key cryptographies, the total length of the messages is around 1024*5=5120. The length of messages in our scheme is around 10^{-1} to that of bilinear map based schemes.

Thirdly, we analyze the storage cost of our scheme. Similar to bilinear map schemes, our scheme generates signatures along with files, which will double the storage costs on the CS. Our scheme does not save storage cost compared to that of bilinear map based schemes. However, the DO and the TPA need only store 3 128-bit symmetric keys in our scheme, while bilinear map based schemes needs to store 1024-bit long keys. Therefore, our scheme saves the keying material storage.

Fourthly, our scheme is stateless. In our scheme, the TPA, the CS and the DO need not store previous state.

The experiment is conducted using C on two Linux systems with an Intel Core 2 processor running at 3.06 GHz, 1024MB of RAM. One computer acts as the CS and the other acts as the TPA. The two computers are in one local area network. The result is shown in TABLE 3. To improve the precision, we repeat the auditing messages 10,000 times continuously, and use the average as the results.

TABLE 3 Experimental Results

	Our scheme	[13]
Auditing phase(ms)	0.87	132.14
Signature (ms)	0.16	68.52

The experiment result matches our theoretical analysis.

VI . Conclusion

In this paper, we propose a light weight publicly auditable Proof-of-storage Scheme for cloud computing. Its security is analyzed, and it is much more efficient than current bilinear map based publicly auditable Proof-of-storage Scheme.

Acknowledgement

This paper is supported by the NSFC (No. 61101088), the Chinese 863 plan (No.2013AA014001), the Fundamental Research Funds for the Central Universities, and the project "security detecting service for wireless intelligent terminal".

References

[1] M. Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing," Univ. California, Berkeley, Tech. Rep. UCBEECS-2009-28, Feb. 2009.

[2] Amazon.com, "Amazon s3 Availability Event: July 20, 2008," July 2008, http://status.aws.amazon.com/s3-20080720.html

[3] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," Dec. 2006, http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-massemail-deletions/

[4] M. Krigsman, "Apple's Mobile Me Experiences Post-Launch Pain," July 2008, http://blogs.zdnet.com/projectfailures/?p=908

[5] A. Juels, J. Burton, and S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. ACM CCS '07, Oct. 2007, pp. 584–97.

[6] G.Ateniese et Al., "Provable Data Possession at Untrusted Stores," Proc. ACM CCS '07, Oct. 2007, pp. 598–609.

[7] C. Wang Et.Al., "Toward Publicly Auditable Secure Cloud Data Storage Services", IEEE network, July. 2010, pp. 19–24.

[8] M. A. Shah et al., "Auditing to keep Online Storage Services Honest," Proc. USENIX HotOS '07, May 2007.

[9] G.Ateniese et al., "Provable Data Possession at Untrusted Stores," Proc. ACM CCS '07, Oct. 2007, pp. 598–609.

[10] M. A. Shah et al., "Auditing to keep Online Storage Services Honest," Proc. USENIX HotOS '07, May 2007.

[11] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. Asia-Crypt '08, LNCS, vol. 5350, Dec. 2008, pp. 90–107.

[12] Q. Wang et al., "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. ESORICS '09, Sept. 2009, pp. 355–70.

[13] C. Wang et al., "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing", Proc. IEEE INFOCOM '10, Mar. 2010.

[14] B. Aboba, et al., "Extensible Authentication Protocol (EAP)", RFC3748, IETF, June 2004.

[15] P. Eronen et al., "Diameter Extensible Authentication Protocol (EAP) Application", RFC4072, IETF, August 2005.

[16] Peter L Montgomery. Modular multiplication without trial division . Mathematics of Computation, 1985, 44 (170) : 19- 521.

[17] O. Elkeelany et al., "Performance Analysis of IPSec Protocol: Encryption and Authentication", proc. of IEEE Communications Conference, New York, USA, pp.1164–1168, 2002.

[18] C. Xenakis et al., "A generic characterization of the overheads imposed by IPSEC and associated cryptographic algorithms", The International Journal of Computer and Telecommunications Networking,Vol.50, No.17,pp.3225-3241,2006.

[19] IETF RFC3766:2004, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys"