2016

Public cloud data auditing with practical key update and zero knowledge privacy

Yong Yu
University of Electronic Science and Technology of China, Guilin University of Electronic Technology, yyong@uow.edu.au

Yannan Li
University of Electronic Science and Technology of China

Man Ho Au
The Hong Kong Polytechnic University, aau@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Kim-Kwang Raymond Choo
University of South Australia

See next page for additional authors

Publication Details

Yu, Y., Li, Y., Au, M. Ho., Susilo, W., Choo, K. & Zhang, X. (2016). Public cloud data auditing with practical key update and zero knowledge privacy. Lecture Notes in Computer Science, 9722 389-405. Melbourne, Australia Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Proceedings

Public cloud data auditing with practical key update and zero knowledge privacy

Abstract
Data integrity is extremely important for cloud based storage services, where cloud users no longer have physical possession of their outsourced files. A number of data auditing mechanisms have been proposed to solve this problem. However, how to update a cloud user's private auditing key (as well as the authenticators those keys are associated with) without the user's re-possession of the data remains an open problem. In this paper, we propose a key-updating and authenticator-evolving mechanism with zero-knowledge privacy of the stored files for secure cloud data auditing, which incorporates zero knowledge proof systems, proxy re-signatures and homomorphic linear authenticators. We instantiate our proposal with the state-of-the-art Shacham-Waters auditing scheme. When the cloud user needs to update his key, instead of downloading the entire file and re-generating all the authenticators, the user can just download and update the authenticators. This approach dramatically reduces the communication and computation cost while maintaining the desirable security. We formalize the security model of zero knowledge data privacy for auditing schemes in the key-updating context and prove the soundness and zero-knowledge privacy of the proposed construction. Finally, we analyze the complexity of communication, computation and storage costs of the improved protocol which demonstrates the practicality of the proposal.

Keywords
knowledge, zero, update, key, practical, auditing, data, cloud, privacy, public

Disciplines
Engineering | Science and Technology Studies

Publication Details
Yu, Y., Li, Y., Au, M. Ho., Susilo, W., Choo, K. & Zhang, X. (2016). Public cloud data auditing with practical key update and zero knowledge privacy. Lecture Notes in Computer Science, 9722 389-405. Melbourne, Australia Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Proceedings

Authors
Yong Yu, Yannan Li, Man Ho Au, Willy Susilo, Kim-Kwang Raymond Choo, and Xinpeng Zhang

Public Cloud Data Auditing with Practical Key Update and Zero Knowledge Privacy

Yong Yu[1,2], Yannan Li[1], Man Ho Au[3], Willy Susilo[4], Kim-Kwang Raymond Choo[5], Xinpeng Zhang[1]

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China.
[2]Guangxi Colleges and Universities Key Laboratory of cloud computing and complex systems, Guilin University of Electronic Technology, Guilin 541004, China.
[3]Department of Computing, The Hong Kong Polytechnic University, Kowloon, China.
[4]Center for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia.
[5]School of Information Technology and Mathematical Sciences, University of South Australia, Adelaide, Australia.
yuyong@uestc.edu.cn

Abstract. Data integrity is extremely important for cloud based storage services, where cloud users no longer have physical possession of their outsourced files. A number of data auditing mechanisms have been proposed to solve this problem. However, how to update a cloud user's private auditing key (as well as the authenticators those keys are associated with) without the user's re-possession of the data remains an open problem. In this paper, we propose a key-updating and authenticator-evolving mechanism with zero-knowledge privacy of the stored files for secure cloud data auditing, which incorporates zero knowledge proof systems, proxy re-signatures and homomorphic linear authenticators. We instantiate our proposal with the state-of-the-art Shacham-Waters auditing scheme. When the cloud user needs to update his key, instead of downloading the entire file and re-generating all the authenticators, the user can just download and update the authenticators. This approach dramatically reduces the communication and computation cost while maintaining the desirable security. We formalize the security model of zero knowledge data privacy for auditing schemes in the key-updating context and prove the soundness and zero-knowledge privacy of the proposed construction. Finally, we analyze the complexity of communication, computation and storage costs of the improved protocol which demonstrates the practicality of the proposal.

1 Introduction

Cloud storage, which enables cloud users to move their data from local storage systems to the cloud, is an important service offered by cloud computing [1]. This kind of new storage service has many advantages such as relieving users' burden

of data management and maintenance, universal data access with independent geographical locations and avoiding capital cost on hardware and software [2]. Currently, an increasing number of users prefer to store their data in the cloud such as Amazon S3, Google cloud storage and Microsoft Skydrive [3].

However, at the meantime, cloud storage also brings a number of challenging security problems [4] despite of its appealing features. Due to losing physical possession and control of their outsourced data, cloud users would worry that their data might be tampered or deleted due to the following reasons [5]. Firstly, data loss could happen in any infrastructure regardless of the reliability measures the service providers would take. Secondly, in cloud storage, the cloud server may not be fully trusted. As a result, the server may discard the data which has been rarely accessed for financial reasons but claims that the data are well stored, or hides data loss incidents to maintain their reputation. Cloud Security Alliance (CSA) conducted a systematic investigation into reported vulnerabilities in cloud computing such as outages, downtimes, and data loss. CSA also released a white paper [6] in 2013 which revealed that the top three threats were "Insecure Interfaces & APIs", "Data Loss & Leakage" and "Hardware Failure". These three threats accounted for 64% of all cloud outage incidents while "Data Loss & Leakage" accounted for 25%. Therefore, guaranteeing the integrity of the data, or data auditing, in cloud is a highly desirable security requirement for secure cloud storage.

Traditional cryptographic approaches to checking data integrity such as hash functions or digital signatures are not directly applicable for cloud storage since a copy of the original file is required in the verification procedure of these primitives. Downloading entire file from cloud is not economical because it leads to unacceptable communication cost. In comparison with traditional cryptographic techniques for data integrity, secure auditing is much more practical for verifying data storage correctness. In 2007, Ateniese et al. proposed the notion of provable data possession (PDP) [7,8] to check the integrity of remote data. Ateniese et al. [9] also demonstrated how to construct data auditing schemes from homomorphic identification protocols. Juels et al. presented the concept of proof of retrievability (POR) [10], in which error-correcting codes and spot-checking technologies are utilized to achieve the properties of possession and retrievability of data files. Shacham and Waters proposed compact proofs of retrievability [11]. One of their constructions built from the BLS signature and secure in the random oracle model achieves the shortest query and response among all the proof-of-retrievability schemes with public verifiability. Because of the elegant properties of BLS signature [12,13], this construction was widely employed to build auditing schemes with additional properties, such as privacy-preserving public auditing [14–17,20], data dynamic auditing [14,18,19,21,22], efficient user revocation [23–26], etc. Note that Wang et al. [26] proposed a novel public auditing mechanism for the integrity of shared data with efficient user revocation using the similar primitive called proxy re-signatures. While employing similar tools, our idea of making use of proxy re-signature is different from that of Wang et al. In Wang et al.'s scheme [26], it is the cloud server who re-signs the blocks

since the cloud server has the re-signing key. However, in our security model, we assume the cloud server is not fully trusted and the key update is to be conducted by the user. In particular, a user may choose to update his key since it could be compromised, we assume the attacker (which is the cloud server in the model), can obtain the user's old key (say sk). This security requirement makes construction in our model non-trivial. Specifically, it rules out the straightforward solution of releasing a re-signing key defined as $\Delta = sk'/sk$, where sk' is the new user secret key, for many of the discrete logarithm based scheme including the aforementioned schemes from BLS signature The reason is that the cloud server can find out the user's new key (sk') from sk and Δ easily. To date, there are two main mechanisms for data auditing, namely mechanisms based on public key infrastructure (PKI) and those based on pseudorandom functions (PRF). The proposals based on PRF are privately verifiable, which means only the data owner can check the integrity of the outsourced data. In comparison, auditing schemes with public verifiability allow external auditors to verify the data integrity on-demand. In this paper, we consider publicly verifiable auditing schemes only, as they are more practical in many applications. The key technique to realize public verifiability is the homomorphic authenticators introduced by Ateniese et al. [7].

In the PKI system, a certificate authority (CA) is involved, whose primary role is to digitally sign and issue certificates for certified users. Public-key cryptography makes use of certificates to address the issue of impersonation. A certificate is an electronic document used to associate the identity of an individual, a company or any other entity with a public key for a specified validity period. A critical problem in PKI is how to deal with the problem of user secret key exposure or expiration. Key expiration indicates certificate of a user is no longer valid while key exposure brings serious security threat to a valid user. As a consequence, for practical purpose, certificate revocation and re-issuing are critical aspects of maintaining the security of the PKI which underpins secure communication on the internet. A certificate may be worth revoking when it has had its private key compromised, the owner of the certificate no longer controls the domain for which it was issued, or the certificate was mistakenly signed. Without the ability to revoke a certificate, a CA has no direct means of marking a certificate as untrusted before the expiry of the certificate, which could be a few years away. In cloud storage, when a user needs to change his/her public key due to various reasons, a dilemma arises on how to verify the integrity of the outsourced data hosted in the cloud since old authenticators stored in the cloud are not valid any longer under the updated public key. A trivial solution for this problem is downloading all the file blocks and authenticators from the cloud, re-computing the authenticators for each block and uploading the blocks and authenticators to the cloud again. This approach will lead to unacceptable communication cost on both the cloud user and the cloud server and at the same time, bring significant computation cost on the cloud users. Addressing the issue of efficient key-updating and authenticator-evolving in data auditing protocols is highly essential to make cloud storage really practical.

Another important issue in cloud data auditing is privacy, including identity privacy and data privacy. Wang et al. [27] [28] developed ring signatures and group signatures based authenticators to realize that the identity of the signer on each block in shared data is kept private from public verifiers. Data privacy requires that the auditing process should not reveal knowledge of the challenged files to the third party auditor. Note that merely encrypting the data is not a viable solution, since the data on the cloud are usually non-static. Homomorphic encryption schemes may solve this issue in theory but current schemes are far from practical. The early auditing schemes are not privacy preserving. In response to a challenge, the cloud may release $\mu_i = \sum_{(i,v_i)} v_i m_i$ where m_i are the challenged blocks and v_i are the challenge generated by the auditor. Hence, each μ_i reveals partial information of the data block. By challenging those block repetitively, the auditor will learn the data. The first privacy-preserving public auditing scheme was proposed by Wang et al. in 2010 [16]. They modified the response in Shacham-Waters scheme by "blinding" the linear combination of the sampled file blocks, such that the auditor cannot recover the file blocks from the responses sent by the cloud server. Unfortunately, Xu et al. [29] found that the protocol is vulnerable to attacks by a malicious cloud server and an outside attacker. Subsequently, Wang et al. improved their scheme [17] such that it is secure against forgery attack. We argue that in some real world applications, it is not sufficient that the auditor cannot derive the whole file blocks. For example, the auditor can launch an offline guessing attack to learn which file among a number of files is stored on the cloud. Wang et al. proposed the notion of "zero knowledge public auditing" to resist off-line guessing attack. However, as a key point to capture this kind of privacy, a formal security model is missing in their work. Yu et al. [30] recently enhanced the privacy of remote data integrity checking protocols for secure cloud storage, but their model does not cover the key update scenario.

Contributions. In this paper, we formalize the security model of "zero knowledge data privacy" for auditing protocols supporting key update, and propose a technique to settle the key update problem efficiently. To be more specific, we utilize a simple zero-knowledge proof of discrete logarithm to make Shacham-Waters protocol [11] reveal no any knowledge of the outsourced data to the verifier in the context of key update. At the same time, the cloud user does not need to download his data from the cloud when his key changes or expires. Instead, the user downloads only the authenticators, which are much shorter in size compared with the entire file. The user can efficiently evolve authenticators, which reduces the communication and computation overheads drastically. We also prove soundness of the new protocol in the random oracle model under a new and reasonable security model. We finally analyze the complexity of the our construction.

2 Preliminaries

In this section, we review the publicly verifiable cloud storage system model and formalize the security model of data auditing protocols with key update in cloud storage.

2.1 System Model of Data Auditing with Key Update

The cloud data auditing architecture with public verifiability [7,11] is illustrated in Fig 1. Three kinds of entities are involved in the scenario, namely cloud users or data owners, the cloud server and a third party auditor (TPA). A cloud user generates data files and stores large amount of data on the remote cloud server without keeping a local copy. The cloud server has significant storage space and computation resources and provides data storage services for cloud users. TPA can be an organization managed by the government, which has expertise and capabilities that cloud users do not have and is trusted to check the integrity of the hosted data on behalf of cloud users upon request. Each entity has his own obligations and benefits. The cloud server may be self-interested, and for his own benefits, such as to maintain its reputation, the server might even decide to hide data corruption incidents to others. Moreover, in the current time period, the cloud server can obtain the data owner's secret key in previous time periods. However, we assume that the cloud server has no incentives to reveal the hosted data to TPA because of regulations and financial incentives. TPA is responsible for checking the integrity of the cloud data on behalf the cloud users in case that they have no time, resources or feasibility to monitor their data, and returns the auditing report to the cloud user. In an auditing scheme with zero knowledge privacy, the TPA cannot learn any information of the stored data during the auditing process.

2.2 System Components and its Security

A public auditing scheme with key-updating and authenticator-evolving is a collection of four algorithms, namely, CrsGen, KeyGen, AuthGen, AuthUpdate, and an interactive proof system Proof between a prover and a verifier.

CrsGen(1^k): This algorithm takes as input a security parameter k and outputs a common reference string crs, which is an implicit input to all algorithms described below.

KeyGen(crs): On input crs, the algorithm generates a public key pk and a secret key sk for the cloud user. The user publishes pk and keeps sk secret. Note that this algorithm is also used for key update.

AuthGen(sk, F): It takes as input the secret key sk and a file $F = (m_1, \cdots, m_n)$, and outputs a set of authenticators $\{D_i\}$ for this file and a set of public verification parameter ϕ, which will be used for checking the data integrity in the proof phase.

Proof$(\mathcal{P},\mathcal{V})$: This is an interactive protocol between the prover $(\mathcal{P})$ and the verifier $(\mathcal{V})$. The common input to $(\mathcal{P}, \mathcal{V})$ is the public key pk and the public

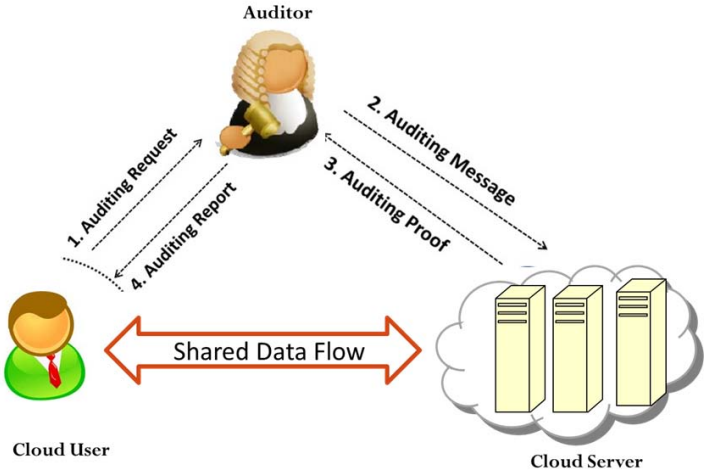

Fig. 1: System model of publicly verifiable data auditing

verification parameter ϕ. $\mathcal{P}$ has additional input the file $F = (m_1, \cdots, m_n)$ and a set of authenticators $\{D_i\}$ of this file. At the end of the protocol, $\mathcal{V}$ outputs a bit 1 or 0 to indicate if the stored file is kept intact or not. For notational convenience, we use $\mathcal{P} \Leftrightarrow \mathcal{V}(pk, \phi) = 1$ to indicate that $\mathcal{V}$ outputs 1 at the end of the interaction with $\mathcal{P}$. We omit the parameters (pk, ϕ) when the context is clear.

AuthUpdate$(sk, pk, sk', pk', \{D_i\}, \phi)$: On input a new key pair (pk', sk') and a set of authenticators $\{D_i\}$ valid under the old key pair (pk, sk), and the public verification parameter ϕ, this algorithm outputs a set of evolved authenticators $\{D_i'\}$ and parameter ϕ' that are valid under the new key pair.

Completeness, soundness and data privacy are three security requirements for the proposed privacy preserving public auditing scheme. Completeness means that when interacting with the cloud server who keeps the data unchanged, the interactive protocol Proof will always result in $\mathcal{P} \Leftrightarrow \mathcal{V} = 1$ when the cloud server and the TPA follow the protocol honestly.

Soundness says that any prover who can convince a verifier it is storing the data file is actually storing that file. Based on the security model due to Shacham and Waters [11], in the following, we define the security model for a public auditing scheme with key-updating and authenticator-evolving against a malicious server, wherein two entities are involved, i.e., an adversary who behaves as the untrusted server and a challenger who represents a data owner or the auditor. The goal of the adversary is to deceive the auditor, that is, generating a valid response without storing the original file. The key difference between our model and that of the previous PDP or PoR models is the capturing of key-updating and authenticator-evolving operations.

Soundness. To capture soundness, we first recap the definition of an ϵ-admissible prover defined in [11]. An algorithm $\mathcal{P}'$ is ϵ-admissible with respect to (pk, ϕ) if $\Pr[\mathcal{P}' \Leftrightarrow \mathcal{V}(pk, \phi) = 1] \geq \epsilon$. That is, it is able to convincingly answer an ϵ fraction of verification challenges from an honest TPA running $\mathcal{V}$.

Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- *Init.* Challenger $\mathcal{C}$ initialises a counter $\mathsf{cnt} = 0$ and an empty table store. It invokes CrsGen on input a security parameter k to obtain crs. It then invokes KeyGen to generate a key pair $(pk_{\mathsf{cnt}}, sk_{\mathsf{cnt}})$. $\mathsf{crs}, pk_{\mathsf{cnt}}$ are given to the adversary $\mathcal{A}$. We assume $\mathcal{A}$ can always retrieve the current value of cnt and store.
- *AuthQuery.* We can safely assume a file F will be unique for each AuthQuery. In case duplicated F is submitted, $\mathcal{C}$ can return the result stored in the table store to $\mathcal{A}$. $\mathcal{A}$ submits a file $F = (m_1, \cdots, m_n)$. $\mathcal{C}$ runs the AuthGen algorithm with input (sk_{cnt}, F) to compute corresponding authenticators $\{D_i\}$ and public verification parameter ϕ_F. $\mathcal{C}$ adds a row $(F, \{D_i\}, \phi_F)$ to the table store. Recall that $\mathcal{A}$ has read access to store and thus returned value is omitted.
- *ProofQuery.*$\mathcal{A}$ can involve itself in a Proof protocol with $\mathcal{C}$. Here, $\mathcal{C}$ will play the role of the TPA ($\mathcal{V}$) with input $(pk_{\mathsf{cnt}}, \phi_F)$. The output of $\mathcal{V}$ is sent to $\mathcal{A}$ upon termination of the protocol.
- *Update.* When $\mathcal{A}$ invokes this query, $\mathcal{C}$ first sets $\mathsf{cnt} = \mathsf{cnt} + 1$. Next, $\mathcal{C}$ invokes KeyGen again to obtain a new key pair $(pk_{\mathsf{cnt}}, sk_{\mathsf{cnt}})$. For each row $(F, \{D_i\}, \phi_F)$ in table store, $\mathcal{C}$ invokes AuthUpdate$(sk_{\mathsf{cnt}-1}, pk_{\mathsf{cnt}-1}, sk_{\mathsf{cnt}}, pk_{\mathsf{cnt}}, \{D_i\}, \phi_F)$ to obtain $\{D'_i\}, \phi'_F$. Update the row to $(F, \{D'_i\}, \phi'_F)$. After that, $\mathcal{C}$ returns $sk_{\mathsf{cnt}-1}$ to $\mathcal{A}$.
- *Challenge.* At some point, $\mathcal{A}$ outputs the description of a prover algorithm $\mathcal{P}^*$.

We first recap the concept of extractor. An extractor Ext is an algorithm which takes as input an ϵ-admissible prover $\mathcal{P}'$, the public key pk and the verification parameter ϕ, outputs a file F. We denote $\mathsf{Ext}(\mathcal{P}', pk, \phi) = F$.

Now we are ready to define soundness. Specifically, we require that for all PPT algorithm $\mathcal{A}$ which outputs an ϵ-admissible prover $\mathcal{P}^*$ with respect to $(pk_{\mathsf{cnt}}, \phi)$, there exists an extractor Ext such that $(\mathsf{Ext}(\mathcal{P}^*, pk_{\mathsf{cnt}}, \phi), pk_{\mathsf{cnt}}, \phi)$ is a row in the table store.

In other words, when an algorithm can pass the Proof protocol with probability greater than ϵ, it stores the original file in some format. The stored file in whatever format can be recovered back to the original file using the extractor algorithm.

Zero Knowledge Data Privacy. This property captures that the TPA learns no knowledge about the content except a random file name of the stored file based on the publicly available information. To further strengthen the property, we allow the TPA to select two equal length files, $F_0 = (m_{0,1}, \ldots, m_{0,n})$ and $F_1 = (m_{1,1}, \ldots, m_{1,n})$ and requires that given a set of meta-data as well as interaction with the cloud server, the TPA cannot obtain any knowledge about the file content of F_0 and F_1. A formal security model is described as follows.

Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- *Init.* Challenger $\mathcal{C}$ initialises a counter $\mathsf{cnt} = 0$, counter $j = 1$ and an empty table store. It invokes CrsGen on input a security parameter k to obtain crs. It then invokes KeyGen to generate a key pair $(pk_{\mathsf{cnt}}, sk_{\mathsf{cnt}})$. $\mathsf{crs}, pk_{\mathsf{cnt}}$ are given to the adversary $\mathcal{A}$. We assume $\mathcal{A}$ can always retrieve the current value of cnt and store. $\mathcal{C}$ initialises an empty row $R^* = (0, \perp, \perp, \perp)$, which is called the challenge row.
- *AuthQuery*(F). We can safely assume F will be unique for each AuthQuery. In case duplicated F is submitted, $\mathcal{C}$ can return the result stored in the table store. $\mathcal{A}$ submits a file $F = (m_1, \cdots, m_n)$. $\mathcal{C}$ runs the AuthGen algorithm with input (sk_{cnt}, F) and obtains corresponding authenticators $\{D_i\}$ and public verification parameter ϕ_F. $\mathcal{C}$ adds a row $(j, F, \{D_i\}, \phi_F)$ to the table store and increases j by one.
- *GenChallenge.* This query can only be issued once. At some point, $\mathcal{A}$ submits two equal-length files F_0 and F_1 (assuming again they are distinct to all inputs to the AuthQuery). $\mathcal{C}$ flips a fair coin $b \in_R \{0, 1\}$ and runs the AuthGen algorithm with input (sk_{cnt}, F_b) and obtains the corresponding authenticators $\{D_i\}$ and public verification parameter ϕ_{F_b}. $\mathcal{C}$ sets the challenge row R^* to be $(0, F_b, \{D_i\}, \phi_{F_b})$. ϕ_{F_b} is returned to $\mathcal{A}$.
- *ProofQuery*$(\hat{j})$. $\mathcal{A}$ engages $\mathcal{C}$ in the Proof protocol. Note that j is specified by $\mathcal{A}$ and refers to the j-th row in the table store or the challenge row if $\hat{j} = 0$ (assume GenChallenge query has been issued). $\mathcal{A}$ plays the role of the TPA and $\mathcal{C}$ plays the role of the prover with input $(pk_{\mathsf{cnt}}, F, \{D_i\}, \phi_F)$ where $(j, F, \{D_i\}, \phi_F)$ is a row in table store if $j > 0$ or $(pk_{\mathsf{cnt}}, F_b, \{D_i\}, \phi_{F_b})$ if $j = 0$ and the GenChallenge query has been issued.
- *Update.* When $\mathcal{A}$ invokes this query, $\mathcal{C}$ first sets $\mathsf{cnt} = \mathsf{cnt}+1$. Next, $\mathcal{C}$ invokes KeyGen algorithm again to obtain a new key pair $(pk_{\mathsf{cnt}}, sk_{\mathsf{cnt}})$. For each row $(F, \{D_i\}, \phi_F)$ in table $\mathsf{store} \cup R^*$, $\mathcal{C}$ invokes AuthUpdate$(sk_{\mathsf{cnt}-1}, pk_{\mathsf{cnt}-1}, sk_{\mathsf{cnt}}, pk_{\mathsf{cnt}}, \{D_i\}, \phi_F)$ to obtain $\{D'_i\}, \phi'_F$. Update the row to $(F, \{D'_i\}, \phi'_F)$. After that, $\mathcal{C}$ returns $sk_{\mathsf{cnt}-1}$ to $\mathcal{A}$.
- *Guess.* At some point, $\mathcal{A}$ outputs a guess bit b'.

$\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ is the probability that it wins in the aforementioned game minus 0.5[1]. A public cloud data auditing scheme is called zero knowledge data private if the advantage of any polynomial-time adversary $\mathcal{A}$ is negligible.

2.3 Zero-Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge protocol [**?**] enables a prover ($\mathcal{P}$) to convince a verifier ($\mathcal{V}$) that some statement is true but the verifier learns nothing except the validity of the statement. In the following, we review a simple but powerful zero-knowledge proof of discrete logarithm instantiation introduced by

[1] This is to offset the winning probability based on random guessing.

Schnorr in his identification scheme [**?**], which plays an important role in our protocol. This zero-knowledge proof of knowledge allows $\mathcal{P}$ to prove the knowledge of $x \in Z_p$ such that $y = g^x$ for some $y \in G$ to $\mathcal{V}$.

Commitment. $\mathcal{P}$ picks a random $\rho \in Z_p$, computes $T = g^\rho$ and sends T to $\mathcal{V}$.

Challenge. $\mathcal{V}$ selects a random $c \in \{0,1\}^\lambda$ and sends c back to $\mathcal{P}$.

Response. $\mathcal{P}$ computes $z = \rho - cx \pmod p$ and returns z to $\mathcal{V}$.

Verify. $\mathcal{V}$ accepts the proof if and only if $T = y^c g^z$.

3 Our construction

Our cloud data auditing protocol derives from compact proofs of retrievability due to Shacham and Waters [11]. Their construction utilizes a homomorphic authenticator based on the short signature scheme [12], which leads to the shortest query and response of any proof of retrievability with public verifiability. For the key-changing and authenticator-evolving, we are inspired by the idea of proxy re-signature [31], which enables a semi-trusted proxy to transform Alice's signature on a message m into Bob's signature on the same message m, to evolve the user's authenticators without downloading the file when his/her key changes. Regarding data privacy, we make use of zero knowledge proof of a discrete logarithm [32], such that both the aggregate authenticator σ and the combinations μ_i of the challenged blocks in the response are randomized. As a consequence, the TPA learns no information of the content of the stored file except a file name randomly picked by the data owner[2]. The details of the protocol are as follows.

CrsGen(1^k). On input a security parameter λ, this algorithm outputs a large prime p and G, G_T, two multiplicative cyclic groups of the same order p. g is a generator of G. $e : G \times G \to G_T$ denotes a bilinear map and $H_0, H_1 : \{0,1\}^* \to G$ represent two collision resistant cryptographic hash functions. In addition, this algorithm picks randomly $h, u_1, u_2 \cdots, u_s \in G$ and computes $\eta = e(g, h)$. The common reference string crs is $(k, p, G, G_T, g, e, H_0, H_1, h, u_1, \cdots, u_s, \eta)$.

KeyGen(crs). On input the common reference string crs, a data owner (cloud user) generates a signing key pair (spk, ssk), $spk = g^{ssk}$ and another key pair (α, v) for generating authenticators of file blocks, where $\alpha \in Z_p$ and $v = g^\alpha$. The secret key of the data owner is $sk = (\alpha, ssk)$ and the public key is $pk = (spk, v)$.

For notational convenience, we use η_i to represent $e(u_i, v)$ for $i = 1$ to s. Note that $\eta_1 = e(u_1, v), \cdots, \eta_s = e(u_s, v)$ can be pre-computed by the relevant parties given the public key v and crs.

AuthGen(sk, F). Given a file F, the data owner firstly applies erasure codes such as RS code to obtain a processed file F', and splits F' into n blocks. Each block is further fragmented into s sectors $\{m_{ij}\}_{1 \le i \le n, 1 \le j \le s}$, which is an element of Z_p. The data owner selects a file name Fn from a sufficiently large domain. Let t_0 be $Fn||n$. The data owner computes $t = (H_0(t_0))^{ssk}$ and denotes the file

[2] This is reasonable since the data owner tells TPA which file will be audited in auditing request phase.

tag $ft = t_0||t$. Then for each $i, 1 \leq i \leq n$, the owner computes an authenticator σ_i for block i as

$$\sigma_i = (H_1(Fn||i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha}.$$

Finally, the data owner stores

$$ft||\{m_{ij}\}_{(1\leq i\leq n, 1\leq j\leq s)}||\{\sigma_i\}_{1\leq i\leq n}$$

to cloud. (Note that there exists strict access control policies that who can access the stored files and authenticators.) **Proof**$(P(F, \{\sigma_i\}, ft), V(pk))$. This is a 5-move interactive proof protocol executed between a prover (cloud server) and a verifier (TPA) as follows.

1. The TPA picks a random integer c and $k, \psi \in Z_p$, computes $\Psi = g^k h^{\psi}$. For $1 \leq i \leq c$, the TPA selects a random $v_i \in Z_p$. The commitment Ψ and the challenge $chal = \{i, v_i\}_{1\leq i\leq c}$, which locates the positions of the challenged blocks in this auditing process, are sent to the cloud server.
2. Upon receiving $(chal, \Psi)$, the cloud server firstly chooses $r, \rho_r, \rho_1, \cdots, \rho_s \in Z_p$ randomly, then computes

$$\Pi = \prod_{(i,v_i)\in chal} \sigma_i^{v_i} \cdot h^r, T = \eta^{\rho_r} \eta_1^{\rho_1} \cdots \eta_s^{\rho_s}$$

   and forwards (T, Π) to the TPA.
3. The TPA sends (k, ψ) to the server.
4. The server checks if $\Psi \stackrel{?}{=} g^k h^{\psi}$. If the equation does not hold, the server aborts. Otherwise, it computes $z_r = \rho_r - kr$, $\mu_i = \sum_{(i,v_i)\in chal} v_i m_{ij}$, $z_i = \rho_i - k\mu_i (1 \leq i \leq s)$ and sends $(z_r, z_1, \cdots, z_s)$ to the TPA.
5. The TPA verifies the file tag ft firstly by checking if the following equation holds,

$$e(g, t) = e(spk, H_0(t_0)).$$

   If the verification fails, reject by emitting *False*. Otherwise, the TPA verifies if

$$(\frac{e(\Pi, g)}{e(\prod_{(i,v_i)\in chal} H_1(Fn||i)^{v_i}, v)})^k \stackrel{?}{=} \frac{T}{\eta^{z_r} \eta_1^{z_1} \cdots \eta_s^{z_s}}.$$

KeyUpdate(pk, sk). The data owner can change his/her key pair (sk, pk) by generating a new key and as a consequence, the data owner has a updated public key $pk' = (spk', v')$ and secret key $sk' = (\alpha', ssk')$ where $spk' = g^{ssk'}$ and $v' = g^{\alpha'}$.

AuthEvolve$(pk, sk, pk', sk', ft, \sigma_i)$. The data owner downloads $ft||\{\sigma_i\}_{1\leq i\leq n}$ from the cloud and evolves the file tag and block authenticators as follows.

1. Compute $t' = t^{\frac{ssk'}{ssk}}$ and let $ft' = t_0||t'$.
2. Compute $\sigma_i' = \sigma_i^{\frac{\alpha'}{\alpha}}$ for $1 \leq i \leq n$.
3. Upload $ft'||\{\sigma_i'\}_{1\leq i\leq n}$ to the cloud.

4 Security of the new protocol

In this section, we show the proposed protocol achieves the properties of completeness, soundness and zero-knowledge privacy. The key update and authenticator evolving are valid as well. Completeness shows the correctness of the protocol and soundness guarantees the security against an untrusted server.

4.1 Completeness

The correctness of the scheme before key update is straightforward to verify with the properties of bilinear pairings. Below we demonstrate the verification still works after the key pair changes. If both the data owner and the server are honest, we have

$$\begin{aligned} t' &= t^{\frac{ssk'}{ssk}} \\ &= ((H_0(t_0))^{ssk})^{\frac{ssk'}{ssk}} \\ &= H_0(t_0)^{ssk'} \end{aligned}$$

Thus, $e(g, t') = e(spk', H_0(t_0))$ holds.

Regarding the evolved authenticators,

$$\begin{aligned} \sigma_i' &= \sigma_i^{\frac{\alpha'}{\alpha}} \\ &= ((H_1(Fn||i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha})^{\frac{\alpha'}{\alpha}} \\ &= (H_1(Fn||i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha'} \end{aligned}$$

Accordingly,

$$e(\sigma', g) = e(\prod_{(i,v_i) \in chal} H_1(Fn||i)^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, v')$$

holds as well.

4.2 Soundness.

An auditing scheme has the property of soundness if any cheating prover who can convince a verifier that it is storing a file F is actually storing that file. In other words, from the cheating prover, there exists an extractor algorithm to extract the file blocks. The soundness proof of our protocol depends on the soundness of the protocol from Shacham and Waters. Specifically, we prove that if there is an adversary who can violate the soundness of our construction, we can construct another algorithm to break the soundness of the Shacham and Waters scheme [11] (referred to as SW scheme hereafter).

Proof. (Sketch) Suppose there is an adversary $\mathcal{A}$ that can break the soundness of our scheme, we construct a simulator $\mathcal{B}$ that can break the soundness of SW scheme.

- $\mathcal{B}$ is given a public key of the SW scheme. The value h will be generated by $\mathcal{B}$ in such a way that the discrete logarithm of h to the base g, an element of the public key of the SW scheme, is known. h and the corresponding elements in pk^* is treated as the crs. Only the component (v, ssk) in the public key of the SW scheme is treated as pk^*, Let n be the number of update query made by $\mathcal{A}$. $\mathcal{B}$ picks a random index $\hat{i} \in \{1, \ldots, n\}$ and set $pk_{\hat{i}} = pk^*$. For $i \in \{1, \ldots, n\} \setminus \{\hat{i}\}$, $\mathcal{B}$ picks a random sk_i and computes $pk_i = g^{sk_i}$. pk_1 is given to $\mathcal{A}$ as the first public key of the system.
- (Queries at period other than $\hat{i}$). For period not equal to $\hat{i}$, $\mathcal{B}$ is in possession of the private key and can thus answer all the queries from the adversary.
- (Update query from $cnt = \hat{i} - 1$). When $\mathcal{A}$ invokes the update query when $cnt = \hat{i} - 1$, $\mathcal{B}$ does not have the corresponding secret key $sk_{\hat{i}}$ and will thus answer the query by re-computing all the authenticators using the Auth-Query to the underlying SW scheme. Note that the resulting authenticators are distributed as if it is computed from the update query.
- (Update query from $cnt = \hat{i}$). When $\mathcal{A}$ invokes the update query when $cnt = \hat{i}$, $\mathcal{B}$ answers the query by re-computing all the authenticators using the secret key $sk_{\hat{i}+1}$. The resulting authenticators are distributed as if it is computed from the update query.
- (Output). Finally, $\mathcal{A}$ outputs a prover $P*$. With probability $1/n$, $P*$ is output during the period $\hat{i}$. $\mathcal{B}$ aborts otherwise. Now we show how $\mathcal{B}$ outputs a prover P' to the underlying SW scheme.
- (Construction of P'). In an execution of $P*$, $\mathcal{B}$ obtains $(T, \Pi, k, z_r, z_1, \ldots, z_s)$. $\mathcal{B}$ rewinds $P*$ to the third step and replies with a different value k' (this is possible since $\mathcal{B}$ knows the discrete logarithm of h to base g and can thus provide the corresponding value ψ'). Now the protocol finishes with a different transcript $(T, \Pi, k', z'_r, z'_1, \ldots, z'_s)$. From these two set of equations, $\mathcal{B}$ can compute the underlying values $(r, \mu_1, \ldots, \mu_s)$. $\mathcal{B}$ computes $\sigma = \Pi / h^r$ and outputs $(\sigma, \mu_1, \ldots, \mu_s)$ on behalf of the prover P'. Note that if $P*$ is ϵ-admissible, P' is ϵ^2-admissible to the underlying SW scheme. In other words, if SW scheme is sound, our scheme is sound too.

4.3 Zero Knowledge Privacy

The new protocol achieves zero knowledge privacy. That is, the data auditing process does not leak any information of the outsourced data. To prove this property, we construct a simulator $\mathcal{S}$ for the interaction between the data owner and the cloud server.

Proof. (Sketch) We show that the probability that $\mathcal{A}$ can output the guess bit correctly with probability negligibly close to 1/2. The idea is to demonstrating how $\mathcal{A}$ is given a challenge that is independent of the underlying file F_0 or F_1.

Recall that the two files are of the same length and will be using the same name fn since the number of blocks and the filename are supposed to be known to the TPA.

To simulate a proof query related to the challenge file, $\mathcal{S}$ follows the protocol honestly until step 3 upon receiving the values (k, ψ). $\mathcal{S}$ rewinds $\mathcal{A}$ to step 2 and picks a random Π, $z_r, z_1, \ldots, z_s$ and computes

$$T = \left(\frac{e(\Pi, g)}{e\left(\prod_{(i,v_i)\in chal} H_1(Fn||i)^{v_i}, v\right)}\right)^k \eta^{z_r} \eta_1^{z_1} \cdots \eta_s^{z_s}.$$

$\mathcal{S}$ sends T, Π to $\mathcal{A}$. Now $\mathcal{A}$ returns with the same (k, ψ) since this pair is fixed by its first message Ψ. $\mathcal{S}$ replies with $(z_r, z_1, \ldots, z_s)$.

It can be seen that the value passes the verification and is distributed correctly. In addition, the view of $\mathcal{A}$ is independent to F_0 and F_1 and thus the probability that $\mathcal{A}$ answers correctly must be $1/2$.

5 Complexity analysis

In this section, we report the complexity analysis of communication, computation and storage costs of the improved protocol.

5.1 Parameter selection

The typical selection of the security parameter λ is 80. Due to the public verification of the proposed protocol, p should be a $2k = 160$-bit prime, and the elliptic curve should be chosen so that discrete logarithm is 2^k-secure. For values of λ up to 128, pairing-friendly elliptic curves of prime order due to Barreto and Naehrig [33] can be employed. $n \gg k$ denotes the number of blocks in a file. Regarding the choice of erasure codes, we follow the suggestions of the SW scheme. That is, traditional RS style erasure codes can be applied to our construction, but the encoding the decoding procedures take $O(n^2)$ time. For the server that is not malicious, a system code, in which the first m blocks of the encoded file are the file itself, can be used for much more efficient public retrievability.

5.2 Performance analysis

Communication cost. In the Proof phase, the TPA sends Φ and *chal* to the cloud server, which is of binary length $\log_2 c + (c+1)\log_2 p$. We can shorten this challenge dramatically by selecting a pseudo-random permutation to compute the locations i of the challenged blocks and a pseudo-random function to calculate the random challenge values v_i. In this circumstance, the TPA sends only keys of the pseudo-random permutation and pseudo-random function, which is of $\log_2 p$ bits each. In the second step, the cloud server returns two points of elliptic curves, T and Π, to the TPA, which is of 320 bits. In the third step, the TPA sends (k, ϕ) to the server, which is of binary length $2\log_2 p$. In the next

step, the cloud server sends $(z_r, z_1, \cdots, z_s)$ to the TPA, which is of binary length $(s+1)\log_2 p$.

Storage cost. In terms of the storage cost of the protocol, both the files and the corresponding metadata including the file tag and block authenticators need to be stored on cloud server due to public verifiability. Our scheme enjoys the advantage of flexible tradeoff between storage and communication in [11]. That is, a parameter s is used to give a tradeoff between response length and storage overhead. Each block is composed of s elements of Z_p called sectors. Each block instead of each sector has an authenticator, reducing the storage overhead to $(1+\frac{1}{s})\times$. The data owner only needs to store the public key $pk = (spk, v)$, the private key $sk = (\alpha, ssk)$, so the storage cost of the data owner is almost $2\log_2 p + 320$ bits. The TPA needs to store the public key of a user, which is of binary length 320 bits. During the auditing process, the TPA stores $k, \phi, chal, \Phi, T, z_r, z_1, \cdots, z_s$ for validating a response from the server, which is of binary length $(c+s+3)\log_2 p + 320$ bits in total.

Computation cost. We report the computation cost from the viewpoint of the data owner, the cloud server and the TPA. We only consider the expensive operations including bilinear maps, exponentiations and multiplications in G and G_T. T_{pair} denotes the time cost of computing a bilinear map of two points of an elliptic curve, and ignore some highly efficient operations, say, computing a hash function. T_{expg} and T_{expgt} stand for the time cost of an exponentiation in G and G_T respectively. T_{mulg} and T_{mulgt} denote the time overhead of a multiplication in G and G_T respectively.

The dominating computation of the data owner is generating authenticators for file blocks as $\sigma_i = (H_1(Fn||i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha}$, in which the time cost is $(s+1)T_{expg} + sT_{mulp}$ for one authenticator. Regarding the authenticators evolving, the data owner needs to perform $n+1$ exponentiations in G, and the time cost is $(n+1)T_{expg}$. As a consequence, the main computation cost of the data owner during the protocol is $(sn+2n+1)T_{expg}+nsT_{mulp}$. To generate and verify a proof, the TPA needs to compute Φ and validate the file tag and the response, and thus the total computation cost of the TPA is $(4T_{pair} + (c+2)T_{expg} + (s+2)T_{expgt})$. To produce a proof, the cloud server has to compute Π, T and $z_r, z_1, \cdots, z_s$, the primary computation cost of the server is $(c+2)T_{expg} + (c+1)T_{mulg} + (s+1)T_{expgt} + sT_{mulgt}$.

6 Conclusion

In this paper, we investigate two important issues of secure cloud data auditing to make cloud storage more practical: (1) how to evolve the authenticators of outsourced files efficiently when a cloud user's key changes, and (2) how to preserve the privacy of the stored files in auditing protocols with key update. We formalized the security model of soundness and zero knowledge data privacy for cloud data auditing process supporting key update. We also provided a concrete construction by cooperating the well-known Shacham-Waters scheme [11] and several novel cryptographic techniques. We proved the security including sound-

ness and zero knowledge privacy of the proposed scheme in the new security model. The performance analysis shows that the new scheme is efficient and can be used in practice.

Acknowledgements. This work is supported by the Fundamental Research Funds for the Central Universities under Grant ZYGX2015J059.

References

1. P. Mell, and T. Grance, "Draft NIST working definition of cloud computing," Reference on June. 3rd, 2009. http://csrc.nist.gov/groups/SNC/cloud-computing/index.html.
2. M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity auditing of outsourced data," in VLDB '07: Proceedings of the 33rd International Conference on Very Large Databases, pp. 782-793, 2007.
3. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generat. Comput. Syst., vol. 25, no. 6, pp. 599–616, Jun. 2009.
4. Cloud Security Alliance. "Top threats to cloud computing," http://www.cloudsecurityalliance.org., 2010.
5. K. Yang, and X. H. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," World Wide Web, vol. 15, no. 4, pp. 409-428, 2012.
6. Cloud Security Alliance, online: `http://www.cert.uy/wps/wcm/connect/975494804fdf89eaabbdab1805790cc9/Cloud_Computing_Vulnerability_Incidents.pdf/?MOD=AJPERES.`, 2010.
7. G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *Proc. of ACM Conference on Computer and Communications Security* 2007: 598-609, 2007.
8. G. Ateniese, R. C.Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. N. J. Peterson, and D. Song, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*,14, 1–34, 2011.
9. G. Ateniese, S. Kamara, J. Katz, "Proofs of storage from homomorphic identification protocols", Proc. of *ASIACRYPT 2009*, pp. 319-333, 2009.
10. A. Juels, and B. S. K. Jr. Pors, "Proofs of retrievability for large files," in *Proc of CCS 2007*, Alexandria, VA, USA, November, 2007, pp. 584-597, ACM.
11. H. Shacham, and B. Waters, "Compact proofs of retrievability," in *Proc of Cryptology-ASIACRYPT 2008*, Melbourne, Australia, Lecture Notes in Computer Science Volume 5350, pp. 90-107, Springer, 2008.
12. D. Boneh , B. Lynn, and H. Shacham, "Short signatures from the weil pairing", In *Proc. of Asiacrypt 2001*, 514-532, 2001.
13. D. Boneh , B. Lynn, and H. Shacham, Short signatures from the weil pairing. *J. Cryptology*, 17, 297-319, 2004.
14. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS 2009*, Saint-Malo, France, September, 2009, Lecture Notes in Computer Science Volume 5789, pp. 355-370, 2009.
15. H. Cui, Y.Mu, M. H. Au, "Proof of retrievability with public verifiability resilient against related-key attacks," IET Information Security, vol. 9, no. 1, pp. 43-49, 2015.

16. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc of IEEE INFOCOM 2010*, San Diego, CA, March, 525-533, 2010.
17. C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. on Computers*, 62, 362-375, 2013.
18. Y. Zhu, G. J. Ahn, H. X. Hu, S. S. Yau, H. G. An, C. J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Services Computing*, vol. 6, no. 2, 227-238, 2013.
19. K.Yang, and X. Jia. "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. on Parallel and Distributed Systems, vol 24, no. 9, 1717-1726, 2013.
20. H. Wang, "Proxy Provable Data Possession in Public Clouds,?" IEEE Trans. Services Computing, Vol. 6, no. 4, pp. 551-559, 2013.
21. A. F. Barsoum, and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," IEEE Transactions on Information Forensics and Security, vol. 10, no. 3, pp. 485-497, 2015.
22. E. Shi, E. Stefanov, C. Papamanthou, "Practical dynamic proofs of retrievability", ACM CCS 2013, 325-336, 2013.
23. J. Yuan, S. Yu, "Public integrity auditing for dynamic data sharing with multi-user modification", IEEE Trans. on Information Forensics and Security, 10(8): 1717-1726, 2015.
24. B.Y. Wang, B.C. Li, H. Li, "Public auditing for shared data with efficient user revocation in the cloud", INFOCOM 2013: 2904–2912, 2013.
25. J.W. Yuan, S.C. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification", IEEE INFOCOM 2014, 2121-2129, 2014.
26. B.Y. Wang, B.H. Li, H. Li, "Panda: public auditing for shared data with efficient user revocation in the cloud", IEEE Trans. Services Computing 8(1): 92–106, 2015.
27. B.Y. Wang, B.C. Li, H. Li, "Oruta: privacy-preserving public auditing for shared data in the cloud", IEEE Trans. on Cloud Computing, 2(1): 43-56, 2014.
28. B.Y. Wang, B.C. Li, H. Li, "Knox: privacy-preserving auditing for shared data with large groups in the cloud", Proc. of the 10th International Conference on Applied Cryptography and Network Security (ACNS 2012), pp.507-525, 2012.
29. C. X. Xu, X. H. He, and D. A. Weldemariam, "Cryptanalysis of wang's auditing protocol for data storage security in cloud computing," Prof. of ICICA (2), 422-428, 2012.
30. Y. Yu, M H Au, Y. Mu, S. Tang, J. Ren, W. Susilo, and L. Dong, "Enhanced privacy of a remote data integrity checking protocol for secure cloud storage," *International Journal of Information Security*, 14(4): 307-318, 2015.
31. G. Ateniese, S. Hohenberger, "Proxy re-signatures: new definitions, algorithms, and applications," in Proc. of *ACM Conference on Computer and Communications Security*, pp. 310-319, 2005.
32. J. Camenisch, and M. Stadler, "Efficient group signature schemes for large groups (Extended Abstract)," in Proc. of *Crypto 1997*, LNCS 1294, pp. 410-424, 1997.
33. P. Barreto, and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. of SAC 2005*, LNCS 3897, pp. 319-331, 2006.