An Improved Privacy Preserving Construction for Data Integrity Verification in Cloud Storage

Yingjie Xia[1, 2], Fubiao Xia[2], Xuejiao Liu[2], Xin Sun[3], Yuncai Liu[2] and Yi Ge[2]
[1] College of Computer Science, Zhejiang University, Hangzhou, 310012 China
[e-mail: xiayingjie@zju.edu.cn]
[2] Institute of Service Engineering, Hangzhou Normal University, Hangzhou, 311121 China
[3] Electric Power Research Institute of State Grid, Zhejiang Electric Power Company, Hangzhou, 310014 China
*Corresponding author: Yingjie Xia

Received March 25, 2014; revised June 4, 2014; revised July 23, 2014; accepted September 10, 2014; published October 31, 2014

Abstract

The increasing demand in promoting cloud computing in either business or other areas requires more security of a cloud storage system. Traditional cloud storage systems fail to protect data integrity information (DII), when the interactive messages between the client and the data storage server are sniffed. To protect DII and support public verifiability, we propose a data integrity verification scheme by deploying a designated confirmer signature DCS as a building block. The DCS scheme strikes the balance between public verifiable signatures and zero-knowledge proofs which can address disputes between the cloud storage server and any user, whoever acting as a malicious player during the two-round verification. In addition, our verification scheme remains blockless and stateless, which is important in conducting a secure and efficient cryptosystem. We perform security analysis and performance evaluation on our scheme, and compared with the existing schemes, the results show that our scheme is more secure and efficient.

Keywords: Data Integrity Verification, Designated Confirmer Signature, Merkle Tree, Public Verifiability.

This research is supported in part by the following funds: National High Technology Research and Development Program of China (863 Program) under grant number 2011AA010101, National Natural Science Foundation of China under grant number 61002009 and 61304188, Key Science and Technology Program of Zhejiang Province of China under grant number 2012C01035-1, and Zhejiang Provincial Natural Science Foundation of China under grant number LZ13F020004 and LR14F020003.

http://dx.doi.org/10.3837/tiis.2014.10.019

1. Introduction

The trends in developing and utilizing cloud computing affect almost every aspects of our society. The ever cheaper and more powerful processors, the increasing network bandwidth, the more reliable and flexible Internet connections [1], together with the "software as a service" (SaaS) computing architecture, promote cloud computing to be the next-generation IT architecture. This promotion moves the software and databases to the centralized data centers, which naturally leads to a new security challenge that the management of the data and services may not be fully trusted.

Comparing with storing data in local, the data uploaded to the cloud will be stored in the remote data centers, which makes the user loss control of his data. So if the cloud experience Byzantine failures occasionally, it will hide the data errors from the users. What's worse, for saving storage cost, the storage server may intentionally delete parts of rarely accessed data files which belongs to an ordinary user. In short, although outsourcing data in cloud reduces the cost of storage, it doesn't offer any guarantee on data integrity. In order to solve this problem, a number of data integrity verification schemes [2, 3, 4, 5] under different cryptographic primitives and security models have been carried out.

Although some of the existing schemes, [3, 5] can support both public verifiability and dynamic data operations, they still have the risk of privacy leakage in the complicated communication environment in cloud storage. In fact, almost all of the data integrity verification schemes are insecure with the existence of a malicious sniffer. The reason is that the essential job to verify the data integrity proof on some challenge information is to verify an ordinary signature. Such a signature can be simply checked by a malicious sniffer if he has captured all transmitted messages. Note one scheme that can escape from the above risk by supporting private verifiability [2]. However, the scheme fails to support public verifiability, which is an important property when a dispute occurs between the prover and the verifier. Meanwhile, considering the tremendous size of stored data in the cloud and resource constraint of clients, another challenge is how can the client find an efficient way to periodically perform integrity verification without the local copy of data files.

Motivated by the problem and challenges in cloud storage, our work and contributions can be summarized in the following three aspects:

(1) By redefining the research problem of data integrity verification in cloud storage, we introduce a new model through both a high-level architecture and a formal syntax. In particular, our model redefines the role of the third party auditor (TPA) to make it more rational than traditional TPAs.

(2) Our scheme is designed to support public verification and data integrity information (DII) for any user. We include a TPA's public key by adding one encryption layer during the proof generation. This work which concerns the so-called designated confirmer signatures (DCS), gives rise to a more flexible solution for ensuring data ingerity in cloud storage.

(3) We give a formal security proof and analyze the performance of our construction by comparing our scheme with the latest schemes.

The rest of the paper is organized as follows. In Section 2 we discuss related work. Section 3 introduces a new model considering a more complicated communication environment in clouds. We introduce some preliminaries of our work and propose a new practical scheme

based on DCS in Section 4. Section 5 gives the security analysis and performance evaluation. Finally, Section 6 concludes the whole paper.

2. Related Work

There has been some work related to data privacy and integrity in cloud storage. Considering the scenario that, Alice uploaded an examination report to cloud storage, and did not want anyone else to know or modify the report, including the cloud administrator. Under such scenario, several access control schemes under cloud computing environments [6, 7] are proposed. Also some cryptographic techniques [8, 9, 10] are used in cloud storage to implement data privacy and integrity. However, as the complexity of the user structure in cloud storage, it is more difficult to combine encrypting data for privacy with verifying integrity of the encrypted data to support both privacy and integrity.

Up to now, many people have proposed a number of methods for data integrity verification. Shah et al. [11] proposed a remote storage auditing method based on pre-computed challenge-response pairs. Ateniese et al. [12] proposed a "provable data possession" (PDP) model for ensuring possessions of files on untrusted stores, and they first introduced "homomorphic authenticators" for auditing outsourced data. Shacham and Waters [2] introduced a "proof of retrievability" (PoR) model, and presented a new scheme to ensure "possession", "retrievability", and "public verifiability" of remote data files. In 2009, Wang et al. [3] introduced another construction that possesses all the features of previous schemes, as well as provides dynamic data operations. As improvement, Wang et al. [5] presented a privacy-preserving public auditing cryptosystem with batch-auditing, which not only achieves privacy-preserving public auditing, but also enables TPA to perform multiple auditing tasks simultaneously. However, all the work above does not support privacy preserving of users' data against external auditors, which may occur serious information leakage.

According to recent research, data integrity verification protocols in cloud storage are required to satisfy three features: data dynamics [3, 13, 14], public verifiability [2, 3, 5, 15], and privacy against verifiers [5]. Hao et al. [16] proposed a protocol that supports public verifiability without help of a third party auditor. Zhu et al. [17] considered the existence of multiple cloud service providers to cooperatively store and maintain the clients' data. They presented a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy. Stefanov et al. [18] presented a practical and authenticated file system by using a large amount of client storage, which supports workloads from large enterprises storing data in the cloud and is resilient against potentially untrustworthy service providers. However, this method requires a large amount of audit cost. The scheme proposed by Cash [19] provided proofs of retrievability for dynamic storage. However, as it employs Oblivious RAM (ORAM) as a black box, it brings increased practical overhead. Elaine et al. [20] proposed a dynamic PoR scheme with constant client storage whose bandwidth cost is comparable to a Merkle hash tree, which outperforms the constructions of the above schemes.

As summary of this section, how to support both public verifiability of data integrity and privacy preserving remains a big problem in cloud storage.

3. A Secure Data Integrity Verification Model for Cloud Storage

Consider such a scenario in financial environment, a trade company uses a public cloud to maintain its business files. Before retrieving a stored file, the company always performs data integrity verification with the cloud. Once such verification fails, which implies the stored data

has been tampered, the company needs some time to deal with the accident and is probably unwilling to reveal the gloomy news. However, in the traditional data integrity verification systems a malicious sniffer by sniffing the communication between the company and the cloud, can simply check the integrity proof since the validation process does not rely on any secret information regarding to the verifier.

To protect DII, it is necessary to seek a secure way to verify a data integrity proof. More specifically, we shall let a verifier with its secret information be able to check the proof which is generated with respect to the particular verifier. However, this again incurs another issue, that is how to handle a dispute between the prover (cloud storage server) and the verifier (the user). In particular, a malicious user can claim *"my file has been tampered!"* although he received a true proof which implies the correctness of the data integrity; Similarly, a dishonest cloud storage server can claim *"your file has never been modified!"* after he tampered the data or the data had unexpectedly been lost, while the user examined the *false* proof. Consequently, the new research problem is, how to design a secure and efficient data integrity verification system for cloud storage with the following two requirements:

(1) It has a flexible protection mechanism towards any user's data integrity information. The DII should be indecipherable without the user's private information, and only the user can see the fact of the DII in general cases.

(2) It supports public verifiability. In some special cases, it allows a TPA, not just the user (data owner), to challenge the cloud server for correctness of data storage. Storage server or a malicious user that disavows the fact of the DII. The TPA may further reveal the fact of the DII by issuing a piece of publicly verifiable information that can be checked by anyone.

In order to solve this problem, we introduce designated confirmer signatures (DCS) to replace the original ordinary signatures in the cloud storage architecture. The concept of DCS was initially proposed by Chaum and van Antwerpen [21]. In a DCS scheme, both the signer and a semi-trusted third party called the designated confirmer can confirm the validity of a signature by running some interactive protocols with a verifier. However, such a verifier cannot further convince other parties of the signature's validity. In addition, the confirmer can selectively convert a designated confirmer signature into an ordinary signature so that it is publicly verifiable. With those special properties of a DCS scheme, we present a secure data integrity verification model which avoids the risk of DII leakage and enables public verifiability. A similar cryptographic primitive called "designated verifier signature" (DVS) can also protect the DII. Since the non-repudiation property is reduced in a DVS scheme where a signer convinces the designated verifier that the message the signer signed is authentic. However, neither the signer nor the designated verifier can convince any other party of the authenticity of that message. Therefore, DVS schemes cannot be used as a building block as it is incompatible with public verifiability in a cloud storage system.

3.1 High-level Description

We give an overview of our improved privacy preserving construction for data integrity verification in cloud storage. A user with mobile devices can create a connection with the cloud storage servers (CSS) to retrieve its stored data files. When receiving a request from the user, a CSS will generate a proof which can only be verified by the specific user or TPA. Whenever a dispute occurs, such a proof can again be sent to TPA, which will examine it and generate *an undeniable proof*. This undeniable proof will reveal the final result and can be checked by other users. Note such a final proof should satisfy these conditions: 1) the proof should be publicly verifiable; 2) the proof must show that the data have not been tampered indeed if a malicious user announces that his stored data files have been modified; 3) the proof

must show that the data have been tampered if dishonest CSS denies occur. **Fig. 1** shows the high-level architecture of a data integrity verification system for cloud storage.

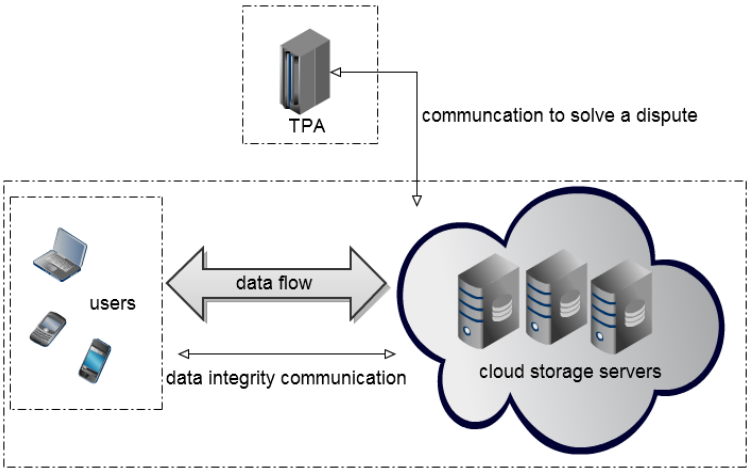

Fig. 1. an architecture of a data integrity verification system

Comparing with previous architectures [3, 5] in cloud storage, we emphasize that the TPA in our model plays a different role, which can be regarded as a judge to mediate disputes between CSS and a user. In fact, any other user except the data owner in the cloud, upon receiving the proof, can verify the proof and know the DII.

3.2 Security Model

We design the security model of a data integrity verification system by extending the model with non-trivial changes. In particular, we design two algorithms, namely "*UndeniableProof*" and "*TPAVerifyProof*", which are both run by TPA. The "*Undeniable*" algorithm aims to resolve the dispute between the user and CSS, while the second algorithm serves as *"Verify Proof"* to check the correctness of a data integrity proof except that the secret input is TPA's private key.

Definition 1 (Syntax). A data integrity verification system involves three roles of parties, i.e., a user *U*, a third party auditor TPA, and a cloud storage server *CSS* consists of the following algorithms:

$KeyGen(1^k) \rightarrow (pk, sk)$. This probabilistic algorithm is run by either the user or TPA. It takes the security parameter 1^k as input, and outputs a public key *pk* and a private key *sk*.

$SigGen(sk_U, F) \rightarrow (\Phi, sig(R))$. This probabilistic algorithm is run by the user. It takes the user's private key sk_U, and an encoded file *F* which is an ordered collection of file blocks $\{m_i\}$ as input, and outputs the signature set Φ, which is a set of signatures of challenging blocks $\{\sigma_i\}$. It also outputs a metadata signature *sig(R)* of the metadata *R* of the encoded file. Note the semantic feature of *R* could be either the file size or other easy-to-store metadata held by the user.

$GenProof(F, \Phi, chal) \rightarrow P$. This algorithm is run by the server. It takes the user's challenging message *chal*, an encoded file *F*, and the signature set Φ as input. It outputs a data integrity proof *P* with respect to the blocks specified by *chal*.

$Verify\Pr oof(chal, P, sk_U) \rightarrow \{TRUE, FALSE\}$. This algorithm is run by the user. It takes the user's private key sk_U, a challenging message *chal*, and a data integrity proof *P* as input. It returns *TRUE* if the integrity of the file is verified to be correct, otherwise it returns *FALSE*.

$Undeinable\Pr oof(chal, P, sk_{TPA}) \rightarrow \pi$. This algorithm is run by TPA. It takes TPA's private key sk_{TPA}, a challenging message *chal*, and a data integrity proof *P* as input. It returns a non-interactive proof π which reveals the correctness of *P*.

$TPAVerify\Pr oof(chal, P, sk_{TPA}) \rightarrow \{TRUE, FALSE\}$. This algorithm is run by the TPA. It takes TPA's private key sk_{TPA}, a challenging message *chal*, and a data integrity proof *P* as input. It returns *TRUE* if the integrity of the file is verified to be correct, otherwise it returns *FALSE*. Note this algorithm can be viewed as a prepositive algorithm of *UndeniableProof*, which is usually invoked before the latter generating a final public evidence.

(Correctness) We say a data integrity verification protocol is correct, if for all key pairs output by *KenGen*, for all files $F = (m_0, \ldots, m_i)$ where $m_i \in Z_q^*$, for all challenge message "*chal*" generated by the user, and for any proof *P* output by an honest prover via *GenProof*, the verification algorithm always accepts:

$$VerifyProof(chal, P, sk_U) = 1 \tag{1}$$

(Soundness) A data integrity verification protocol is sound if any cheating prover can convince the verification algorithm that it is storing a file is actually storing that file. It means that the cheating prover yields up the file to an extractor algorithm that interacts with it using the data integrity verification protocol. Note the formal definition of the soundness of a proof-of-retrievability protocol is presented in [2].

4. The Proposed Scheme

4.1 Preliminaries

We introduce two building blocks used in our construction.

Bilinear Maps a lot of research works [22, 23, 24, 25] have been put in elliptic curve cryptography (ECC) and many cryptosystems [26, 27, 28] have been proposed. Pairings were initially used for attacking the existing cryptosystems. In this context, pairings are cryptanalysis tools which regarded as a function that takes two points on an elliptic curve as input, and outputs an element of some multiplicative group.

Definition 2 (Bilinear Map). Suppose that *G* and G_t be two multiplicative cyclic groups with the same prime order *q*, while *g* is a generator of *G*. A bilinear pairing on (G, G_t) is a bilinear map $e : G \times G \rightarrow G_t$, which satisfies the following properties:

Bilinearity: For all $u, v \in G$, and for all $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$.

Non-degeneracy: $e(g, g) \neq 1$, where 1 is the multiplicative identity of group G_t.

Computability: *e* can be efficiently computed.

Merkle Tree We use a Merkle Tree to maintain all the stored data file. As a well studied authentication structure, a merkle tree intends to prove that a set of elements are undamaged and unaltered efficiently and securely. It is constructed as a binary tree where the leaves are the hashes of authentic data values. In this paper we further employ such a data structure to

authenticate both the values and the positions of data blocks. We treat the leaf nodes as the left-to-right sequence, so any leaf node can be uniquely determined by following this sequence and the way of computing the root node in the Merkle Tree.

4.2 Intuition Behind

A common paradigm of constructing a DCS scheme is "sign-and-encrypt"[29]. For example, to generate a proper DCS, a signer firstly computes an ordinary signature on the target message *m* by using his private key, and then encrypts that signature under the confirmer's public key. Such a ciphertext is usually regarded as a DCS signed on *m*. To verify such a DCS, the signer usually provides some proof show that "the ciphertext is an encryption of his signature under someone's public key". Also the confirmer with its private key can decrypt such a ciphertext, and verify the internal signature.

We propose a new construction based on BLS signatures [30] and ElGamal encryption schemes [31]. In fact, a single DCS in our data integrity verification system is computed by hiding a basic signature with ElGamal encryptions. We use the same construction as in [3] to compute the underlying basic signatures, which are homomorphic authenticators pre-computed by the user, and stored in CSS. To issuing a proof, CSS encrypts those BLS signatures with TPA's public key to generate intermediate DCSs. The responding proof contains aggregated signatures which linearly combines those DCSs of challenging message blocks. Such a proof can only be checked by a verifier with its private key, i.e., either the user or the TPA.

To reduce the complexity of data management, we adopt the same techniques, i.e., a Merkle Tree, as in [3,5] to maintain the authentication structure.

4.3 Our Construction

We assume the client encodes the raw file $\overline{F}$ into *F*, as in Wang et al.'s scheme [3], by using Reed-Solomon codes, and divides *F* into *n* blocks: $m_1, m_2, \ldots, m_n$, where $m_i \in Z_q^*$. Let $e : G \times G \rightarrow G_t$ be a bilinear map where *g* is a generator of *G*. $H : \{0,1\}^* \rightarrow G$ is a full-domain hash function. The procedure of our protocol executions are illustrated as follows.

Setup: Initially, *KeyGen* is invoked to generate the entities' public and private keys. By running *SigGen*, the homomorphic authenticators and the metadata are pre-processed.

$KeyGen(1^k)$: The user *U* chooses a random value $x_U \in_R Z_q^*$ and a random element $u \in_R G$, computes $y_U = g^{x_U}$. The private key of *U* is x_U, and the public key of *U* is (y_U, u). Similarly, the third party auditor TPA sets its private/public key pair as $(x_{TPA}, y_{TPA} = g^{x_{TPA}})$, where $x_{TPA} \in_R Z_q^*$.

$SigGen(sk_U, F)$: On the encoded file $F = (m_1, m_2, \ldots, m_n)$, the user *U* generates the authenticators for each blocks by computing *n* signatures as $\sigma_i = (H(m_i) \cdot u^{m_i})^{x_U}$. Let $\Phi = \{\sigma_i\}, 1 \le i \le n$ denote the set of signatures. Then *U* generates a root R based on the construction of a Merkle Tree, where the leave nodes represents an ordered set of hashes of "file tags", i.e. $H(m_i), 1 \le i \le n$. *U* signs the root *R* with its private key x_U as a BLS signature [30]: $\sigma_R = H(R)^{x_U}$. Eventually, *U* sends $\{F, \Phi, \sigma_R\}$ to the storage server *CSS*, and also deletes all values from its local storage.

Default Integrity Verification: The user *U* can check the integrity of the outsourced data *m* by challenging the storage server *CSS*. To generate the challenging message, *U* chooses a random subset $I=(s_1,...,s_c)$, where s_i denotes the index of the message block m_i, and we assume $s_1 \le i \le s_c$. For each $s_i \in I$, *U* picks a random value $r_i \in Z_q^*$. Eventually, U sends the challenging message $chal=\{(s_i,r_i)\}_{s_i \le i \le s_c}$ to *CSS*.

$Gen\,\mathrm{Pr}\,oof(F, \Phi, chal)$: To generate an integrity proof, CSS computes an aggregated DCS for the challenging blocks. Specifically, on receiving the challenge request $chal=\{(s_i,r_i)\}$, *CSS* computes $\varsigma_i = y_{TPA}^{r_i}$, $\mu_i=\sigma_i \cdot g^{r_i}$, $\mu = \Pi_{i=s_1}^{s_c}\mu_i$, $\varsigma=\Pi_{i=s_1}^{s_c}\varsigma_i$, $\theta=\Sigma_i^{s_c} m_i$。 In addition, *CSS* also provides the verifier (*U*) with a small amount of auxiliary information fig $\{\Omega_i\}, s_1 \le i \le s_c$, which denotes the sibling nodes on the path from the leaves $H(m_i), s_i \le i \le s_c$ to the root *R* in the Merkle Tree. Next, *CSS* responds *U* with proof *P*, including the $DCS(\varsigma, \mu)$ and θ, the challenged leaves $\{H(m_i)\}$, and the auxiliary information $\{\Omega_i\}$. We have $P=\{(\varsigma,\mu,\theta),\{H(m_i)\},\{\Omega_i\},\sigma_R\}$, where $s_1 \le i \le s_c$.

$Verify\,\mathrm{Pr}\,oof(chal, P, sk_U)$: On receiving the responses from the prover *CSS*, the verifier *U* first generates the root *R* by using $\{H(m_i)\}$ and $\{\Omega_i\}$, where $s_1 \le i \le s_c$, and authenticates it by checking if the equation $e(\sigma_R, g) \equiv e(H(R), y_U)$ holds. If such a check fails, *U* rejects the proof with an output "*FALSE*". Otherwise, *U* checks the following Equ.2.

$$e(\mu, y_{TPA}) \equiv e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_{TPA})^{x_U} \tag{2}$$

If so, *U* outputs "*TRUE*"; otherwise, *U* outputs "*FALSE*".

Undeniable Integrity Verification: In case of the occurrence of a dispute occurs, that is the cloud storage server *CSS* denies a *false* proof *P* after *U* executing $VerifyProof(chal, P, sk_U)$ with "*FALSE*" as output. The third party auditor TPA can verify the proof *P* and outputs an undeniable proof to reveal the fact that data were tampered.

$TPAVerifyProof(chal, P, sk_{TPA})$.On receiving the proof information from either *CSS* or the user, the TPA first generates the root *R* by using $\{H(m_i)\}$ and $\{\Omega_i\}$, where $s_1 \le i \le s_c$, and authenticates it by checking if the equation $e(\sigma_R, g) \equiv e(H(R), y_U)$ holds. If such a check fails, *U* rejects the proof with an output "*FALSE- 0*". Otherwise, *U* checks the following Equ.3.

$$e(\mu, y_{TPA}) \equiv e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_u)^{x_{TPA}} \tag{3}$$

If Equ.3 holds, U outputs "TRUE"; otherwise, U outputs "FALSE -1".

$Undeinable\,\mathrm{Pr}\,oof(chal, P, sk_{TPA})$: At the beginning, TPA invokes the algorithm *TPAVerifyProof*.

Once a string "*FALSE- 0*" is output, TPA reveals part of the proof information, that is, $\{\{H(m_i),\{\Omega_i\},\sigma_R\}_{s_1 \le i \le s_c}$, so that anyone else can verify it.

Once a string "*TRUE*" is output, TPA can issue a non-interactive zero knowledge proof (NIZK) π_1 to show the correctness of the data storage. And such a NIZK actually proves the

validity of the $DCS(\varsigma, \mu, \theta)$, and is publicly verifiable. Let $W_1 = e(\mu, y_{TPA}) / e(\varsigma, g)$, $W_2 = e(\Pi_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_U)$, a NIZK π_1 is transformed from the zero knowledge proof of knowledge $\pi_1^{'}$: $pk = \{x_{TPA} : W_1 = W_2^{x_{TPA}} \wedge y_{TPA} = g^{x_{TPA}}\}$. In particular, we uses the techniques in [32] to generate π_1 from $\pi_1^{'}$ .TPA picks a random value $v \in Z_q^*$, computes $V = g^v$, $W = W_2^v$, $c = H'(g, W_2, y_{TPA}, W_1, V, W) \in Z_q^*$, and $s = v + cx_{TPA}$, where $H' : \{0,1\}^* \rightarrow Z_q^*$ is a secure cryptographic hash function. TPA issues a NIZK $\pi_1 = (W_1, W_2, s, c)$. To verify such a NIZK proof, anyone with the inputs $\pi_1 = (W_1, W_2, s, c)$, first computes $V = g^s y_{TPA}^{-c}$,and $W = W_2^s, W_1^{-c}$. Then he computes $c' = H'(g, W_2, y_{TPA}, W_1, V, W)$. If c = c´, he accepts such a NIZK that shows the equality of two discrete logarithms, which further implies the validity of the $DCS(\varsigma, \mu)$.

Once a string "FALSE- 1" is output, TPA can issue a NIZK proof π_2 to show the incorrectness of the data storage. And such a NIZK actually proves the invalidity of the $DCS(\varsigma, \mu)$, and is publicly verifiable. Let $W_1 = e(\mu, y_{TPA}) / e(\varsigma, g)$, $W_2 = e(\Pi_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_U)$, a NIZK π_2 is transformed from the zero knowledge proof of knowledge $\pi_2^{'}$: $pk = \{x_{TPA} : W_1 \neq W_2^{x_{TPA}} \wedge y_{TPA} = g^{x_{TPA}}\}$. In particular, we use the Fiat-Shamir heuristics [33] to transform the honest verifier zero-knowledge proof for showing the inequality of two discrete logarithms in [34], into a non-interactive zero-knowledge proof. The detailed techniques can be found in **Fig. 2**.

Common Input: W_1, W_2, g, y_{TPA}		
prover CSS (with private input x_{TPA})		verifier
$c_2, d_1^{'}, d_2^{'}, r, \alpha, \beta \in_R Z_q^*$,		
$A' \in_R G$, $A = (W_1^{x_{TPA}} / W_2)^r$		
$z_1^{'} = y_{TPA}^{d_1^{'}} / (W_2^{d_2^{'}} A^{'c_2})$,		Output Accept if
$z_2^{'} = g^{d_1^{'}} W_2^{d_2^{'}}$,		$c_1 + c_2 \equiv H(A, A', z_1, z_2, z_1', z_2') \bmod q$,
$z_1 = W_1^{\alpha} / W_2^{\beta}$, $z_2 = g^{\alpha} / y_{TPA}^{\beta}$;	$(A, A', z_1, z_2, z_1^{'}, z_2^{'}, c_1, c_2, d_1, d_2, d_1^{'}, d_2^{'})$	$W_1^{d_1} / W_2^{d_2} = z_1 A^{c_1}$
$c_1 = H(A, A', z_1, z_2, z_1', z_2') - c_2 \bmod q$	$\longrightarrow$	$g^{d_1} / y_{TPA}^{d_2} = z_2$
$d_1 = \alpha + c_1 x_{TPA} r \bmod q$,		$y_{TPA}^{d_1^{'}} / W_2^{d_2} = z_1' A'^{c_2}$
$d_2 = \beta + c_1 r \bmod q$		$g^{d_1^{'}} / W_1^{d_2^{'}} = z_2^{'}$

Fig. 2. A NIZK proof in the algorithm UndeniableProof

Remark 1. It is noticed that our construction supports blockless verification. That is to say, for both efficiency and security concerns, no challenged file blocks should be retrieved by the verifier or TPA during the verification process. In addition, the user or TPA in the proposed

scheme is fully stateless, which means that the verifier do not need to maintain the state information between audits throughout the long term of data storage.

Remark 2. Our *UndeniableProof* algorithm actually relies on two NIZK protocols, which are also a DH-tuple witness hiding (WH) protocol and a non Diffie-Hellman(DH)-tuple WH protocol respectively. The concept of WH was introduced by Feige an3d Shamir [35]. Informally a two-party protocol between a prover and a verifier is witness hiding if at the end of the protocol the verifier cannot compute any new witness which he did not know before the protocol.

5. Evaluation

We give formal security analysis and performance analysis for our DCS-based data integrity verification scheme. The results below show that our scheme is both secure and efficient. On the other hand, our scheme achieves high flexibility for balancing the public and private verifiability by using the means of designated confirmer signature.

5.1 Security Analysis

We analyze our construction in two dimensions, i.e., we shall guarantee that our data integrity verification protocol is both correct and sound.

As shown in the Equ.4, we simply show the correctness of our construction by showing the correctness of Equ.2.

$$\begin{aligned}(\mu, y_{TPA}) &= e(\prod_{i=s_1}^{s_c} \sigma_i \cdot g^{r_i}, y_{TPA}) \\ &= e(\prod_{i=s_1}^{s_c} \sigma_i, y_{TPA}) \cdot e(\prod_{i=s_1}^{s_c} g^{r_i}, y_{TPA}) \\ &= e(\prod_{i=s_1}^{s_c} \sigma_i, y_{TPA}) \cdot e(\prod_{i=s_1}^{s_c} g^{r_i}, g)^{TPA} \\ &= e(\prod_{i=s_1}^{s_c} \sigma_i, y_{TPA}) \cdot e(\varsigma, g) \\ &= e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} (H(m_i) \cdot u^{m_i}), y_{TPA})^{x_U} \\ &= e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} (H(m_i) \cdot u^{\theta}), y_{TPA})^{x_U}\end{aligned} \tag{4}$$

The correctness of Equ.3 analogously derives from the above analysis by replacing (y_{TPA}, x_U) with (y_U, x_{TPA}).

Soundness requires that, if any cheating prover that convinces the verification algorithm that storing a file *F* is actually storing that file, which we define that it yields the file *F* to an extractor algorithm that interacts with it using the data integrity verification protocol. We prove the soundness of our protocol in three theorems, while proofs for the second theorem and the third theorem are actually similar as the proofs in [3], and therefore we give the proof sketch of the second and the third theorems.

Theorem 1. If the underlying basic signature scheme is existentially unforgeable and the computational Diffie-Hellman problem is hard in bilinear groups, then in the random oracle

model, except with negligible probability, no adversary against the soundness of our public-verification scheme ever causes the verifier to accept in a data integrity verification protocol instance, except by responding with *correctly computed values*.

Proof: Our proof consists of two steps. Firstly, we show the unforgeability of the underlying basic signature. Consider the underlying basic signature $\sigma_0 = (H(m_0) \cdot u^{m_0})^{x_U}$ on any encoded message m_0. Since we use the same construction as [2], the unforgeability is naturally satisfied due to the previous paragraph.

Secondly, we prove the theorem by using a sequence of games. The first game, Game 0, is simply the challenge game, which is similar to the *setup* game between the adversary *A* and the challenger (environment) *C* in [2] (in Section 2). Game 1 is almost the same as Game 0 with only one difference. The challenger *C* keeps a list of all signed tags ever issued as part of a store-protocol query. If *A* ever submits a tag either in initiating a data integrity verification protocol or as the challenge tag, the challenger will abort if it is a valid tag that has never been signed by the challenger. Based on the definition of Game 0 and Game 1, it is obviously that we can use the adversary to construct a forger against the signature scheme, if there is a difference in the adversary's success probability between Game 0 and Game 1.

Game 2 is almost the same as Game 1, except that in Game 2 the challenger keeps a list of its responses to queries from the adversary. Now the challenger observes each instance of the data integrity verification protocol with the adversary. Let $P = \{(\varsigma, \mu, \theta), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$ where $s_1 \le i \le s_c$ is the expected response that would have been obtained from an honest prover. The correctness of $H(m_i)$ can be verified through $\{H(m_i), \Omega_i\}_{s_1 \le i \le s_c}$ and σ_R. The correctness of the proof can be verified based on the equation $e(\mu, y_{TPA}) \equiv e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_u)^{x_{TPA}}$. Assume the adversary's response is P′. Because of the authentication in Merkle Tree, the second part in P′ should be the same as $\{H(m_i), \Omega_i\}_{s_1 \le i \le s_c}$ and σ_R. Suppose $P' = \{(\varsigma', \mu', \theta'), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$ where $s_1 \le i \le s_c$ is the adversary's response. The verification of $(\varsigma', \mu', \theta')$ is $e(\mu', y_{TPA}) \equiv e(\varsigma', g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta'}, y_{TPA})^{x_u}$. Obviously, $\varsigma' = \varsigma = \prod_{i=s_1}^{s_c} y_{TPA}^{r_i}$. And also we have $\theta' \neq \theta$, otherwise $\mu' = \mu$, which contradicts our assumption in this game.

Define $\Delta\theta = \theta' - \theta$. With this adversary, the simulator could break the challenge Computational Diffie-Hellman (CDH) instance as follows: Given $(g, g^a, h) \in G$, a CDH-adversary B is asked to output h^a. Initially, by picking a random value $b \in Z_q^*$ and a random element $y_{TPA} \in G$, *B* sets $y_U = g^a$, and $u = gh^b$, which implies the private key of the user *U* is $x_U = a$ and is not known by *B*. *B* could answer all signature queries with the similar method in [1], by manipulating the random oracle: $H(m_i) = g^{r_i} h^{-m_i}$. Eventually, *A* outputs $P' = \{(\varsigma', \mu', \theta'), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$ where $s_1 \le i \le s_c$. We obtain $e(\mu'/\mu, y_{TPA}) \equiv e(u^{\Delta\theta}, y_{TPA})^a$.

From this equation, we have $e(\mu'\mu^{-1}, y_{TPA}) \equiv e((gh^b)^{\Delta\theta}, y_{TPA})^a$. Therefore, $h^a = \mu' \mu^{-1} V^{\Delta\theta})^{\frac{1}{b\Delta\theta}}$. Unless evaluating the exponent $b\Delta\theta$ causes a divide-by-zero. However, because *b* is chosen by the *B* and hidden from the adversary *A*, the probability that $b\Delta\theta = 0 \bmod q$ will be only $1/q$, which is negligible.

Game 3 is the same as Game 2, with the following difference: as before, the challenger *C* observes the data integrity verification protocol instances. Suppose the file that causes the abort is that the signatures are $\{\sigma_i\}$.

Suppose (i, v_i) where $s_1 \le i \le s_c$ is the query that causes the challenger to abort, and that *A*'s response to that query was $P' = \{(\varsigma', \mu', \theta'), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$, where $s_1 \le i \le s_c$. Let $P = \{(\varsigma, \mu, \theta), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$ where $s_1 \le i \le s_c$ is the expected response obtained from an honest prover. We have proved in Game 2 that $\mu' = \mu$. It is only the values θ and θ' that can differ. Define $\Delta\theta = \theta' - \theta$. Finally, the adversary outputs a forged signature $P' = \{(\varsigma', \mu', \theta'), \{H(m_i)\}, \{\Omega_i\}, \sigma_R\}$ where $s_1 \le i \le s_c$. Now we can have:

$$e(\mu', y_{TPA}) = e(\varsigma', g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta'}, y_{TPA})^{x_U}$$

$$= e(\mu, y_{TPA}) = e(\varsigma, g) \cdot e(\prod_{i=s_1}^{s_c} H(m_i) \cdot u^{\theta}, y_{TPA})^{x_U} \tag{5}$$

In case of $\varsigma' = \varsigma$, from Equ.5, we have $1 = u^{\Delta\theta}$. In this case, $\Delta\theta = 0 \bmod q$. Therefore, we have $\theta = \theta' \bmod q$. As we analyzed above, there is only negligible difference probability between these games. This completes the proof.

Theorem 2. Suppose a cheating prover on an *n*-block file *F* is well-behaved in the sense above, and that it is ε -admissible. Let $\omega = 1/\# B + (pn)^l / (n - c + 1)^c$. Then, provided that $\varepsilon - \omega$ is positive and non-negligible, it is possible to recover a ρ -fraction of the encoded file blocks in $O(n/(\varepsilon - \rho))$ interactions with the cheating prover and in $O(n^2 + (1 + \varepsilon n^2)(n)/(\varepsilon - \omega))$ time overall.

Proof sketch: The verification process is similar with [3]. Therefore, we omit the details of the proof here. We can also prove that extraction always succeeds against a well-behaved cheating prover, with the same probability analysis given in [3]. For detailed discussion, we refer to the proof of Theorem 2 in [3] or Section 4.2 in [2].

Theorem 3. Given a fraction of the *n*-blocks of an encoded file *F*, it is possible to recover the entire original file *F* with all but negligible probability.

Proof sketch: By adopting the rate- ρ Reed-Solomon codes, this result can be easily achieved, since any ρ -fraction of encoded file blocks suffices for decoding. Again, we refer to Section 4.3 in [2].

5.2 Performance Analysis

Recall *c* is the number of challenged data blocks. Let *pr* and *ex* denote the time for computing a pairing and an exponentiation in G and G_t , respectively. We compare our DCS-based data integrity verification scheme with three schemes. The first one is the public verifiable version in [2] (see the section 3.3 of [2]), the second one is the proposition in [3], and

the last scheme is the privacy-preserving public auditing scheme in [5] (see in scheme C). All compared schemes are based on elliptic curves, and are derived from the same origin, i.e., the framework of proofs of retrievability in [2]. We simply call these schemes, including our scheme, as "PoR"-type schemes. Also we mainly concern about the security and time complexity among four schemes, because our scheme adds one more layer comparing to the other schemes while process the remaining communicated messages under the same construction. In addition, we add two algorithms for supporting public verifiability, which slightly increases the computational and communication costs if invoking a TPA. **Table 1** gives the comparison of these schemes within several dimensions. Note that private verifiability means only the data owner can check the proof to know the data integrity information, while no other parties can know that fact. For the schemes in [3] and [5], we suppose their TPAs are including the user itself because there is no key-pair for the TPA, and we think they support private verifiability in that case. We remark that in our scheme, a TPA is a semi-trusted party with its own key-pair, as we assume that it only runs the algorithms *TPAVerifyProof* and *Undeniableproof* to check and further publish the integrity information if a dispute occurs.

Table 1. A comparison between four "PoR"-type schemes

	Our scheme	**Scheme in [2]**	**Scheme II in [3]**	**Scheme C in [5]**
DII protection	YES	NO	NO	NO
Public verifiability	YES	YES	YES	YES
Private verifiability	YES	NO	YES	YES
Time A Proof generate time (by CSS)	$2c \cdot ex$	$c \cdot ex$	$c \cdot ex$	$(1+c) \cdot ex$
Time B Proofverify time (by U)	$5pr+2ex$	$2pr+2\ c \cdot ex$	$4pr+(1+c) \cdot ex$	$2pr+(2+c) \cdot ex$
Time C Proof verify time (by TPA)	$5pr+2ex$	N/A	$4pr+(1+c) \cdot ex$	$2pr+(2+c) \cdot ex$
Time D Undeniable Proof generate time (by TPA)	$2ex$ or $10ex$ (when data tampered)	N/A	N/A	N/A
Time E Undeniable Proof verify time (by others)	$4ex$ or $10ex$ (when data tampered)	N/A	N/A	N/A

Time A= the time of proof generation by cloud storage server, or the time of the proof verification by the prover in the scheme in [2];

Time B=the time of proof verification by the user, or the time of the proof verification by the verifier in the scheme in [2];

Time C= the time of proof verification by the third party auditor;

Time D=the time of undeniable proof generation by TPA;

Time E=the time of undeniable proof verification by other parties. Note the time cost in each cell is approximately evaluated by omitting minor factors such as multiplications in bilinear groups.

From the table, we can see that our protocols benefit from the lowest time cost in the verification phases, i.e., Time B and Time C. Our scheme takes only constant exponentiations, even though a bit more pairing-computation. Note our proof generation time is roughly twice as long as the other competitors'. This is because the listed three schemes adopt a variant of "aggregated BLS" signatures to hide the randomnesses in the challenge. However, our scheme employs an advanced version, by adding one more layer to hide the randomnesses via an Elgamal encryption, and thus it results in more time cost. In addition, we stress that our scheme is the first proposition that can effectively protect the data integrity information according to the first row.

6. Conclusion and Future Works

To ensure cloud data storage security, we propose a new architecture of data integrity verification system which limits the public verifiability of data integrity proofs in a public cloud. By deploying designated confirmer signatures, we construct a new practical scheme to resist the risk of data integrity information leakage. On one hand, our scheme achieves a more flexible verification mechanism compared with the schemes in [2]. Note that all these schemes supporting "public verifiability" actually reveal the data integrity information for any user. Our scheme maintains two-levels, i.e., a private verification with regard to a specific user, and a public verification with regard to a TPA. On the other hand, our scheme still benefits from the features of being blockless and stateless. In fact, no local copies of the challenged data blocks in CSS will be retrieved by the verifier (or a TPA) during the verification process, and there is no need to maintain information in a single verification process between different audits throughout the long term of data storage.

In future work, we will improve the construction to support data dynamics. In particular, the scheme which can support block modification, insertion, and deletion is a significant step towards practicality. Moreover, we plan to further prove the efficiency and effectiveness of our scheme for data integrity verification in cloud storage.

References

[1] Y. Xia, M. Zhu, Y. Li, "Towards topology-and-trust-aware P2P grid," *Journal of Computers*, Vol.5, No.9, pp. 1315–1321, 2010. Article (CrossRef Link)

[2] H. Shacham, B. Waters, "Compact proofs of retrievability," in *Proc. of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90–107, 2008. Article (CrossRef Link)

[3] Q. Wang, C. Wang, J. Li, K. Ren, W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of the 14th European Symposium on Research in Computer Security*, pp. 355–370, 2009. Article (CrossRef Link)

[4] A. Juels, B. S. Kaliski Jr, Pors, "Proofs of retrievability for large files," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 584–597, 2007. Article (CrossRef Link)

[5] C. Wang, Q. Wang, K. Ren, W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. of the 29th Conference on Information Communications*, pp. 525–533, 2010. Article (CrossRef Link)

[6] Y. Xia, L. Kuang, M. Zhu, "A hierarchical access control scheme in cloud using HHECC," *Information Technology Journal*, Vol.9, No.8, pp. 1598–1606, 2010. Article (CrossRef Link)

[7] X. Liu, Y. Xia, Sh. Jiang, F. Xia, "Hierarchical attribute-based access control with authentication to outsourced data in cloud computing," in *Proc. of the 12th IEEE International Conference on*

Trust, Security and Privacy in Computing and Communications, pp. 477–484, 2013. Article (CrossRef Link)

[8] Y. Yoon, J. Oh, B. Lee. "The Establishment of Security Strategies for Introducing Cloud Computing," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 4, pp. 860–877, 2013. Article (CrossRef Link)

[9] L. Zhang, Y. Hu. "New Constructions of Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Computing," *KSII Transactions on Internet and Information Systems*, vol.7, no.5, pp. 1343–1356, 2013. Article (CrossRef Link)

[10] J. Kim, C. Park, J. Hwang H. Kim. "Privacy Level Indicating Data Leakage Prevention System," *KSII Transactions on Internet and Information Systems,* vol. 7, no.3, pp. 558–575, 2013. Article (CrossRef Link)

[11] M. A. Shah, M. Baker, J. C. Mogul, R. Swaminathan, et al., "Auditing to keep online storage services honest," in *Proc. of the 11th USENIX Workshop on Hot Topics in Operating Systems*, pp. 1–6, 2007. Article (CrossRef Link)

[12] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable data possession at untrusted stores," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, 2007. Article (CrossRef Link)

[13] G. Ateniese, R. Di Pietro, L. V. Mancini, G. Tsudik, "Scalable and efficient provable data possession," in *Proc. of the 4th International Conference on Security and Privacy in Communication Networks*, No. 9, 2008. Article (CrossRef Link)

[14] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, "Dynamic provable data possession," in *Proc. of the 16th ACM Conference on Computer and Communications Security*, pp. 213–222, 2009. Article (CrossRef Link)

[15] K. D. Bowers, A. Juels, A. Oprea, Proofs of retrievability: "Theory and implementation," in *Proc. of the 16th ACM Workshop on Cloud Computing Security,* pp. 43–54, 2009. Article (CrossRef Link)

[16] Z. Hao, S. Zhong, N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering* vol.23, no.9, pp.1432–1437, 2011. Article (CrossRef Link)

[17] Y. Zhu, H. Hu, G. Ahn, M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage," 2012. Article (CrossRef Link)

[18] E. Stefanov, M. van Dijk, A. Juels, A. Oprea, Iris, "A scalable cloud file system with efficient integrity checks," in *Proc. of the 28th Annual Computer Security Applications Conference*, pp. 229–238, 2012. Article (CrossRef Link)

[19] D. Cash, A. Küpçü, D.Wichs, "Dynamic proofs of retrievability via oblivious ram," *Advances in Cryptology–EUROCRYPT*, pp. 279–295, 2013. Article (CrossRef Link)

[20] E. Shi, E. Stefanov, C. Papamanthou, "Practical dynamic proofs of retrievability," in *Proc. of the 2013 ACM SIGSAC conference on Computer & Communications Security*, pp. 325–336, 2013. Article (CrossRef Link)

[21] D. Chaum, "Designated confirmer signatures," *Advances in Cryptology*, pp. 86–91, 1995. Article (CrossRef Link)

[22] G.-J. Lay, H. G. Zimmer, "Constructing elliptic curves with given group order over large finite fields," in *Proc. of the 1st International Symposium on Algorithmic Number Theory*, pp. 250–263, 1994. Article (CrossRef Link)

[23] A. J. Menezes, T. Okamoto, S. A. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, vol.39, no.5, pp.1639–1646, 1993. Article (CrossRef Link)

[24] I. Semaev, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic *p,*" *Mathematics of Computation of the American Mathematical Society,* vol.67, no. 221, pp. 353–356, 1998. Article (CrossRef Link)

[25] N. P. Smart, "The discrete logarithm problem on elliptic curves of trace one," *Journal of Cryptology*, vol. 12, no. 3, pp. 193–196, 1999. Article (CrossRef Link)

[26] S. D. Galbraith, N. P. Smart, "A cryptographic application of weil descent," in *Proc. of 7th*

Institute of Mathematics and its Applications International Conference, pp. 191–200, 1999. Article (CrossRef Link)

[27] L. Zhang, Q. Wu, Y. Hu. "New Constructions of Identity-based Broadcast Encryption without Random Oracles," *KSII Transactions on Internet and Information Systems*, Vol. 5, No. 2, pp. 247–276, 2011. Article (CrossRef Link)

[28] Y. Hitchcock, E. Dawson, A. Clark, P. Montague, "Implementing an efficient elliptic curve cryptosystem over *gf(p)* on a smart card," *ANZIAM Journal* 44, pp. 354–377, 2003. Article (CrossRef Link)

[29] G. Wang, F. Xia, Y. Zhao, "Designated confirmer signatures with unified verification," Cryptography and Coding, pp. 469–495, 2011. Article (CrossRef Link)

[30] D. Boneh, B. Lynn, H. Shacham, "Short signatures from the weil pairing," in *Proc. of Annual Cryptographical and Cryptoanalytic Conference*, pp. 514–532, 2001. Article (CrossRef Link)

[31] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Advances in Cryptology*, pp. 10–18, 1985. Article (CrossRef Link)

[32] E. Goh, S. Jarecki, "A signature scheme as secure as the diffie-hellman problem," in *Proc. of Annual International Conference on the Theory and Applications of Cryptographic Techniques* ,Lecture Notes in Computer Science, pp. 401–415, 2003. Article (CrossRef Link)

[33] A. Fiat, A. Shamir, "How to prove yourself: practical solutions to identification and signature problems,lecture notes in computer science," in *Proc. of Annual Cryptographical and Cryptoanalytic Conference*, pp. 186–194, 1987. Article (CrossRef Link)

[34] K. Kurosawa, S.-H. Heng, "3-move undeniable signature scheme," in *Proc. of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 181–197, 2005. Article (CrossRef Link)

[35] U. Feige, A. Shamir, "Witness indistinguishable and witness hiding protocols," in *Proc. of the 22th Annual ACM Symposium on Theory of Computing*, pp. 416–426, 1990. Article (CrossRef Link)

Yingjie Xia received his Ph.D. Degree in the College of Computer Science, Zhejiang University in 2009. He was affiliated as a research scientist in National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign (UIUC), USA. Now he is an associate professor in Zhejiang University, Hangzhou, Zhejiang, P. R. China. He founded Intelligent Transportation and Information Security Lab (ITIS) in Hangzhou Normal University and serves as its director. His research interests cover distributed computing, high performance computing, data mining, and their applications in intelligent transportation systems. Prof. Xia has published more than 80 research papers.

Xuejiao Liu received her BSc in computer science and technology from Huazhong Normal University, Wuhan, in 2006; Ph.D. degree in network security from Huazhong Normal University, Wuhan, in 2011.She is currently a lecturer in Intelligent Transportation and Information Security Lab at Hangzhou Normal University. Her research interests include network security, applied cryptography and cloud storage security.

Fubiao Xia obtained his BSc in Software Engineering in Fudan Univerisity, and received his MSc in Computer Security, and Ph.D. in Computer Science in Univerisity of Birmingham, G.B. He is presently employed as a research engineer at Shanghai Fudan Microelectronics Group .co .Ltd. His research areas include applied cryptography, software security, embedded software system and security topics in cloud computing.

Xin Sun received his BSc and MSc in Zhejiang University. Now he is an Engineer in Electric Power Research Institute of State Grid, Zhejiang Electric Power Company, Hangzhou, Zhejiang, China. His research interests are in application security testing a nd penetration testing.

Yuncai Liu received the Ph.D. degree in electrical and computer science engineering from the University of Illinois at Urbana-Champaign in 1990. He worked as an associate researcher at the Beckman Institute of Science and Technology from 1990 to 1991. Since 1991, he had been a system consultant and then a chief consultant of research in Sumitomo Electric Industries Ltd., Japan. In October 2000, he joined the Shanghai Jiao Tong University, China, as a Distinguished Professor. His research interests are in image processing and computer vision, especially in motion estimation, feature detection and matching, and image registration. He also made many progresses in the research of intelligent transportation systems.

Yi Ge is an undergraduate student in school of foreign language, Hangzhou Normal University. She is currently a research student in Intelligent Transportation and Information Security Lab at Hangzhou Normal University. Her research interest is cloud storage security.