Efficient and Privacy Preserving Protocol against Insider Attack for Data Storage in Cloud Computing

Jennifer Batamuliza
University of Rwanda(UR) African Center of Excellence in Data Science
(ACE-DS) PhD Candidate in Data Mining
University of Kigali (UoK)

ABSTRACT
Cloud computing provides remote users a flexible and convenient way to obtain cloud services on demand such as cloud storage service, which has been facing great security and privacy challenges, especially insider attacks. However, most of the previous work on the cloud security focusing on the storage security can't be effective against insider threats. Wei's scheme on the cloud storage, which is based on an ID-based strong designated verifier signature (IBSDVS) protocol, takes privacy and confidentiality into consideration. But it also can't be against insider attacks and its confidentiality exists security flaws. Hence, in this paper, we propose an efficient data storage protocol in the cloud computing, which can be against insider attacks as well as providing privacy preserving and confidentiality. Similar to Wei's scheme, our protocol adopts an IBSDVS scheme that has the secure property of non-delegatability. Then the analysis of security and performance are described in detail.

Keywords
Cloud computing, storage security, designate verifier signature, privacy preserving, and insider attack.

1. INTRODUCTION
CLOUD computing [1] has become the common focus of the industry, academic circle, government and all walks of life concerned in the field of information technology currently. Cloud computing embodies the idea that the network is the computer. It combines large amounts of computing resources, storage resources, and software and hardware resources together to form a huge scale of pool sharing virtual IT resource. Then it can provide remote users a flexible and convenient way to obtain cloud services on demand such as data storage and data computation [2]. Due to its convenience, economy, scalability and other advantages, cloud computing attracts more and more enterprises' sight. They can liberate from the heavy pressure of the IT infrastructure management and maintenance by renting the necessary resources. In the IT industry, cloud computing, which is generally regarded as another an important growth point after the Internet economic boom, has great market prospects [3].

In the same time, with the development of cloud computing, cloud storage has become a hot researching topics in the

filed of information storage. Compared with traditional storage devices, data storage in cloud computing is not just a hardware, but a system constituted with network equipment, storage equipment, servers, software applications, public access interfaces, access networks and clients. Cloud storage provides remote storage service that the local data would store in an available online storage space, which is provided by cloud storage service provider (CSSP). Users who require storage service no longer need to build their own data center, but only apply for storage service to CSSP. Then it plays a role in saving the hardware and software infrastructure investment. Furthermore, when the concept of cloud storage was put forward, it got the supports and attention with numerous manufacturers. Amazon launched simple storage service(S3) with elastic block storage (EBS) technology to support persistent data storage. Google launched the google drive online storage service(GDrive). CDNetworks and Nirvanix formed a strategic partnership to provide an integrated platform of cloud storage service and content delivery service. And Windows Azure was introduced by Microsoft, which has established huge data centres in USA around.

Even if the cloud storage has advantages of low price, deploying conveniently and so on, it faces many key issues that security problem is the first to bear the brunt. With the increasing popularity of cloud storage, the customers worry about security and privacy challenges [4], which has become an important factor to restrict its development, have showed a gradually rising trend. From an investigation conducted by Twinstrata on the new cloud storage service in 2012, only 20% of respondents will put their private data in the cloud storage servers (CSS). In a cloud storage model, once the data is outsourced to a cloud storage service provider, it means that data owners give up the control of their data, because CSSP have the priority access right to cloud users' data or applications [5]. In addition, the existing cloud storage platforms are always having fatal security flaws. Google happened a serious cloud security accident that a large number of users' private documents were leaked in 2009. In the same year, the Simple Storage Service of Amazon happened two interruptions in the single storage service [6]. Thus, it may exist the internal staff dereliction of duty, hacker attacks, system malfunctions leaded to security mechanism failure and a variety of other risks, CSSP don't have enough evidences to convince cloud users that their data is correctly stored in the cloud storage servers. Hence one can see that data security is one of the major obstacles to cloud storage development and cloud storage system has an urgent demand on the security mechanism. The cloud storage security can be divided into two categories: Secure cloud storage and Secure cloud auditing. Secure cloud storage refers to making sure that the cloud users could securely and correctly store their data at mistrusting cloud storage servers, while secure cloud auditing refers to verifying the integrity and correctness of the outsourced data at mistrusting cloud storage servers. Therefore, cloud data integrity also can be split into the integrity of storing and the integrity of using. There have been many researches on the secure cloud auditing, such as [7], [8], [9], [10], [11]. However, in this paper, we only discuss the secure cloud storage in the cloud computing. We involve the designated verifier signature technique with secure properties of non-delegatability, no transferability, unforgeability and privacy of signer's identity to achieve that messages can be stored correctly by means of secure transmission and integrity verifying. Moreover, privacy preserving [12], [13], [14] can't be ignored for secure cloud storage.

In this paper, we propose a new secure cloud storage protocol with designated verifier signature. Then the main research

contents and achievements of this paper are as follow: In next section, we give a brief overview on the related work. Section 3 introduces the secure cloud storage model, secure definitions and necessary preliminary knowledge. We review the existing Sec Cloud scheme in Section 4 and give the secure analysis in Section 5. Section 6 proposes our new secure cloud storage protocol with designated verifier signature. Then Section 7 describes performance analysis and security analysis of our protocol. In the end, we conclude this paper in Section 8.

2. RELATED WORK

Currently, there are many researches on the security of cloud computing. And it is pay more attention to the secure cloud storage while the attention to secure cloud computation is on the contrary. To verify the integrity of a large file deposited at a remote cloud storage server, Deswarte et al.[15] in 2004 first proposed a scheme of validating remote data integrity based on a hash function with RSA. Since this scheme is on account of the public key cryptography, the cost of calculating is great expensive. In 2007, Juels et al.[7] proposed a scheme of proofs of retrievability for large static files, which not only verify the integrity of remote data, but also can retrieve the damage data with a certain probability by using the method of sentinels hidden.

Shacham and Waters put forward two improved scheme with compact POR[16] by using homomorphic verifiable tags. The one is based on pseudo random function and does not support the public verification. The other one is based on BLS signature and support the public verification. After doing a lot of research work, Ateniese et al.[8] introduced a protocol of provable data possession at untrusted stores(PDP), which is the first considering the public verification. Two schemes in their paper also uses the technology of homomorphic verifiable tags with RSA and supports privacy protection. But multiple servers can realize collusion attacks, so they are not suitable for multiple copy protocols. Then Ateniese et al. proposed a scalable-PDP[17] scheme again supporting data dynamic operation and it is provable security in the random oracle model. In 2009, Erway et al.[9] proposed 2 kinds of dynamic provable data possession(DPDP) to achieve data updating. One is rank-based authenticated skip lists and the other one is based on the RSA-tree structure. Furthermore, Wang et al.[10] realized the cloud data storage public verification and dynamic operations by applying the third-party auditor. Their protocol uses the technologies of Merkle Hash Tree[18] to build BLS[19] for block tag authentication, while it dose't support privacy preserving. Later, an improved scheme was proposed to support privacy preserving on dynamic remote data storage with public integrity auditing in [20], which makes use of the public homomorphic key and the random mask technique. In subsequent studies, Wang et al. based paper [10] and further discussed the bilinear aggregation signature algorithm[21]. Hence, the third-part auditor can implement multiple integrity auditing tasks with multiple users efficiently. In [22], Zhu et al. put forward a kind of cooperative provable data possession, which can support batch auditing for multiple clouds and also was extended to support the dynamic auditing in [23]. Both [22] and [23] does't support batch auditing for multiple cloud users. However, Zhu's schemes as well as [10], [20], [21] are suffering expensive computational cost for the third-party auditor. Accordingly, Kang et al.[11] brought about the batch auditing for multiple clouds and multiple cloud users. All the schemes above are proposed to achieve the cloud storage security. Contrarily, the comparison is so dramatic between the cloud computation security and the cloud storage security. The secure cloud computation is consist of remote computation auditing and verifiable computation. Golle et al.[24] put forward schemes that prevents cheating by making sure if participants were honest and correctly performing the computations, or allowing participants to prove if they have done most of the computations allocated with high probability. Monrose et al.[25] introduced a framework of remote auditing based on an existing distributed computing model. Their scheme supports efficient approaches for verifying whether a remote host correctly performed the assigned task. In [26], authors gave the concept of veri?able computation. Due to the host with weak computing capability, the host should outsource the computation of a function on difference parameters that were chosen dynamically to the remote servers, which responded the computation results and the proofs to prove the correctness of the computation on the gave parameters. But it is a fully homomorphic encryption scheme, its computational cost on the proof verifying is great. Moveover, [27], [28] were proposed by Wei et al., which introduced auditing schemes to achieve the cloud computation security as well as the cloud storage security and privacy cheating discouragement by the technologies of designated verifier signature, batch auditing and probabilistic sampling technique. In 2011, Canetti et al.[29] researched on verifiable computation with two or more clouds.

3. DATA STORAGE PROTOCOL IN CLOUD COMPUTING

In this section, we first present the definitions of data sharing protocol including the protocol architecture, the security model and the preliminaries of our scheme.

3.1 Data sharing architecture

We construct a general architecture for data sharing, which contains the cloud servers, the cloud users and the verifying agencies, as shown in Fig. 1. In the cloud computing environment, there are a lot of cloud servers managed by one or multiple cloud service providers (CSPs), which have high computational resources and a large of storage space.

CSP further has the abilities of batch auditing and parallel performing a number of sub-tasks divided by a large task, which are allocated to multiple cloud servers. The cloud user may be a computer, a laptop or a mobile phone, which has not enough computational and storage resources compared with the cloud server. Cloud users create data and outsource their data to the cloud servers. When they desire to obtain computational and storage services, cloud users should send requests for applying corresponding service. Our auditing model also involves verifying agencies (VAs) trusted and designated by cloud users. VA has more professional knowledge and capacities to execute the auditing service than cloud users.

Then VAs should be in charge of the auditing results on data storage and computation.

3.2 Definitions of secure data sharing

Definition 1 (Data sharing). A secure data sharing with a ID-based designated verifier signature(IBSDVS) scheme con-sists of Setup, KeyExtract, DataSign, DataTran, VerSign, Simulate.

Setup(1) ! fmpk; mskg : This algorithm takes 1 as input, where is a security parameter, and outputs a system master public key mpk and a system master private key msk.

KeyExtract(msk; A) ! fsk$_A$g : The algorithm takes msk and an identity ID of Alice as input. Then it outputs a secret key sk$_{ID}$.

DataSign(sk$_A$; A; B; mpk; M) ! f g : The algorithm takes Alice's secret key sk$_A$ and identity A as a signer, Bob's identity B as a

verifier, mpk, and a message M = fm$_1$; m$_2$; :::; m$_n$g 2 f0; 1g as input. It outputs a signature for Bob.

DataT ran(M; ; key$_{A;B}$) ! fE$_k$ey$_{A;B}$(M;)g : The algorithm takes the message M, its signature and the section key key$_{A;B}$ as input. It outputs the secret message fE$_k$ey$_{A;B}$(M;)g.

V erSign(M; ; A; B; sk$_B$; mpk) ! f0; 1g: The algo-rithm takes the message M, its signature , the identities of Alice with A and Bob with B, Bob's secret key sk$_B$ and mpk as input. Then it outputs 0 or 1.

Simulate(sk$_B$; A; B; mpk; M) ! f g : The algorithm takes Bob's secret key sk$_B$ and identity B as a signer, Alice's identity A as a verifier, mpk, and a message M as input. It outputs a simulated signature for Alice.

1) Non-Delegatability: Delegatability is that without leak-ing their private key, a signer or a designated verifier can delegate a third-party to produce a valid signature. Hence, we give the following definition of Non-Delegatability.

Definition 2 (Non-Delegatability). A data sharing protocol with an IBSDVS scheme is non-delegatable with (t; t^0; "; "^{0}) if for every algorithm F and its runtime is at most t, it exists a black-box knowledge extractor K satisfying the following conditions:

For every Setup(1) ! fmpk; mskg, every

A; B 2 f0; 1g , every KeyExtract(msk; A) ! sk$_A$,

KeyExtract(msk; B) ! sk$_B$, and every message

M 2 f0; 1g , if F (can be denoted by F$_{A;B;M}$) generates a valid signature on M with respect to A, V with a probability ". And then K extracts secret key sk$_A$ or sk$_B$ on inputting M

and on oracle access to F$_{A;B;M}$ in expected time (t^0 < P$_1$(t)) with a probability ("^{0} > P$_2$(t)), where P$_1$(t) and P$_2$(t) are two polynomial functions.

2) Non-transferability: Non-transferability is that a mes-sage M and its IBSDVS signature are given and there is no possible for an adversary A with secret keys of signer or verifier to distinguish whether the signature is signed by singer or verifier.

Definition 3 (Non-transferability). A data sharing protocol with an IBSDVS scheme is non-transferable if the signature signed by the signer is indistinguishable from ^{0} signed by the designated verifier, i.e.

fDataSign(sk$_A$; A; B; mpk; M)

Simulate(sk$_B$; B; A; mpk; M)g:

If the two distributions are identical, we say that the data sharing protocol with an IBSDVS scheme is perfectly non-transferable.

3) Unforgeability: Without getting a signer's secret key sk$_A$ or a designated verifier's secret key sk$_B$, it is not computationally feasible to calculate the designated verifier signature with message M in a probabilistic polynomial-time. Hence, we can define it by playing the following game between a probabilistic polynomial-time adversary A and a challenger C.

1) Challenger C produces a system master key pair (mpk; msk) by running the Setup algorithm, and in-vokes an adversary A on inputting mpk.

2) Then A issues queries, for many times polynomially, to the following oracles adaptively:

O$_K$: The oracle takes as input a query identity ID, then it computes and returns KeyExtract(msk; ID) ! fsk$_{ID}$g to A.

O$_{DSign}$: A issues a query (A; B; M) to the oracle and it outputs a valid signature to A.

O$_{V\ er}$: A issues a query (A; B; M;) to the oracle and it outputs 1 if is valid or otherwise, outputs 0.

3) Finally, A forms its forgery (A ; B ; M ;). It wins the game if

a. V erSign(A ; B ; sk$_B$; mpk; M ;) ! 1;

b. A did not issue a query to O$_K$ on inputting A and B ;

c. A did not issue a query to O$_{DSign}$ on inputting(A ; B ; M).

Definition 4 (Unforgeability). A data sharing protocol with an IBSDVS scheme is unforgeable with (t; q$_K$; q$_{DSign}$; q$_{V\ er}$; ") if no adversary A can runs in time at t, issues at most q$_K$ queries to O$_K$, q$_{DSign}$ queries to O$_{DSign}$ and q$_{V\ er}$ queries to O$_{V\ er}$, and wins the game with probability at least ".

4) Privacy: It is not computationally feasible for an ad-versary A to determine the key pair of signature without knowing a designated verifier's secret key sk$_B$ or a signer's secret key sk$_A$. Then we define PSI of a data sharing protocol with an IBSDVS scheme as follow by describing a game played between the challenger C and a distinguisher D.

1) Challenger C produces a system master key pair (mpk; msk) by running the Setup algorithm, and in-vokes a distinguisher D on inputting mpk.

2) D issues queries, for many times polynomially, to the same oracles adaptively in the unforgeability game.

3) D takes A$_0$, A$_1$, which are two signer identities, and B of a verifier identity and a message M . Then C tosses a coin b 2 f0; 1g and computes

KeyExreact(mpk; A) ! sk$_A$ b b

DataSign(sk ; A ; B ; mpk; M) ! :

Ab b

4) D continues to issue queries.

5) D outputs a bit b^0, which is 1 or 0. It wins the game if

a. b^0 = b;

b. D did not query O$_K$ on inputting B ; and

c. For any d 2 f0; 1g, D did not query O$_{V\ er}$ on inputting (A$_d$; B ; M ;).

Definition 5 (Privacy of signers identity). A data shar-ing protocol with an IBSDVS scheme is PSI security with (t; q$_K$; q$_{DSign}$; q$_{V\ er}$; ") if no distinguisher D runs in time at most t, issues at most q$_K$ queries to O$_K$, q$_{DSign}$ queries to O$_{DSign}$, and q$_{V\ er}$ queries to O$_{V\ er}$ and wins the game with a probability that deviates from one-half by more than ".

3.3 Preliminary knowledge

1) Bilinear pairings: Let G$_1$ and G$_2$ are respectively a cycle additive group and a cycle multiplicative group, which has the same prime order q. Then let P is the generator of G$_1$. Assume that the Discrete Logarithm Problem (DLP) in G$_1$ and G$_2$ is both hard problem. Let e^ : G$_1$ G$_1$! G$_2$ is the bilinear pairings with following properties:

1) Bilinearity: For all P; P ^{0} 2 G_1 and a; b 2 Z, there is e^(aP; bP ^{0}) = e^(P; P ^{0})^{ab}.

2) Non-degeneracy: If there is e^(P; P ^{0}) = 1 for 8P ^{0} 2 G_1,

then P =

3) Computability: For 8P; P ^{0} 2 G_1, there exists an effective algorithm to compute e^(P; P ^{0}).

Generally, we can use the Weil pairing or the reformed Tate pairing[19] to construct bilinear pairings.

2) CBDH Assumption: The Computational Bilinear Diffie-

Hellman (CBDH) assumption is that let $G(1^k)$!

q; ; ; e^ (P; aP; bP; cP) input, where P

h G_1 G_2 i given is as$_{abc}$ 2

G_1, a; b; c 2 Z. Then it computes e^(P; P) 2 G_2 as output.

The difficulty of CBDH problem is based on CBDH assumption. Due k is big enough, an algorithm A has the advantage $Adv_{G;A}(k)$ to solve BDH problem if

Pr[A(q; G_1; G_2; e;^ aP; bP; cP) = e^(P; P)^{abc}] $Adv_{G;A}(k)$ We definite CBDH assumption[?] that $Adv_{G;A}(k)$ can be negligible for all the PPT algorithm A with k. That is,

Pr[CBDH] $Adv_{G;A}(k)$:

3) Designated Verifier Signature: Designated verifier signature [30], [31], [32] refers that only the designated verifier B can verify the validity of the signature signed by A, while B can't convince others that the legitimate signature is really signed by A. Because B can also construct a legitimate signature by himself. Even though an adversary get the private keys of the signer A and the designated verifier B, it is impossible for the adversary to decide whether the legitimate signature is signed by A or B. Strong designated verifier signature is a special designated verifier signature. In the verification process, it should be used the the designated verifier's private key so that only the designated verifier has the capacity to verify the signature.

4. ANALYSIS OF THE SECCLOUD PROTOCOL

To achieve secure cloud computing, the basic SecCloud protocol relying on the identity-based cryptography. Here this paper review their proposed protocol which consists of two phases: system initialization, secure cloud storage.

A. Review of the SecCloud protocol

1) Initialization: The Initialization phase includes two

steps:

Setup(1) ! params. In this step, Initialization System takes 1 as input and chooses a number s 2 Z_q randomly as the system master secret key, then it computes P_{pub} = s P as the system public key. Finally, the outputs of this step is params = (G_1; G_2; q; e;^ P; P_{pub}; H; H_1; H_2; H_3).

UserReg(ID) ! (params; sk_{ID}). In order to register to Initialization System, this step takes a cloud user's identity ID as input to get the cloud services. Then Initialization System outputs params and a user secret key sk_{ID} = s Q_{ID}, and sends them to the cloud user(U), where $Q_{ID} = H_1(ID)$.

2) Cloud storage security: This Cloud storage security phase includes the following four steps:

StorageApp(M) ! I. This step takes a message with n blocks M = m_1; m_2; :::; m_n 2 Z_q as input by U. Then it outputs the space list I = i_1; i_2; :::; i_n distributed by the cloud service provider(CSP) and sent to U.

DataSign(M; I) ! . This step takes n unsign data

M = m_1; m_2; :::; m_n 2 Z_q and space list I = i_1; i_2; :::; i_n as input by U. Then it outputs with n signatures on these data. For each data block m_i, U randomly picks up r_i 2 Z_q and computes a signature $_i$ on the basis of the designated verifier signature technology:

$U_i = r_i$ Q_{ID}

$h_i = H_2(U_i km_i ki_i)$

$V_i = (r_i + h_i)$ sk_{ID}

$_i = e^(V_i; Q_{CS})$

where = f $_i g_{1;:::;n}$ and $_i = (U_i;$ $_i)$.

DataEnca(M; ; $k_{ID;CS}$) ! $E_{kID;CS}$ (fM; g). This step takes the pairs of the message and corresponding signature

fM; g and the session key $k_{ID;CS}$ by U. Then it outputs $E_{kID;CS}$ (fM; g) encrypted by $k_{ID;CS}$ and sent to CSP, where $key_{ID;CS} = H_3(^e(sk_{ID}; Q_{CS}))$.

Similarly, the session key between U and VA is $k_{ID;V\ A} = H_3(^e(sk_{ID}; Q_{V\ A}))$ and returns $E_{kID;V\ A}$ (fM; g) to VA.

DataVer($E_{kID;CS}$ (fM; g); $k_{CS;ID}$; sk_{CS}) ! f0; 1g. This step takes $E_{kID;CS}$ (fM; g), $k_{CS;ID}$ and sk_{CS} by CSP. It decrypts and obtains the pairs fM; g by its own session key

$k_{CS;ID}$, since

$k_{CS;ID} = H_3(^e(sk_{CS}; Q_{ID}))$

= $H_3(^e(s\ Q_{CS}; Q_{ID}))$

= $H_3(^e(Q_{CS}; s\ Q_{ID}))$

= $H_3(^e(Q_{CS}; sk_{ID})) = k_{ID;CS}$

Then CSP takes the secret key sk_{CS} to check the validity of the signatures by the following equation:

?

$_i = e^(U_i + H_2(U_i km_i ki_i)Q_{ID}; sk_{CS})$

If the above equation holds, returns 1; Otherwise, returns 0.

B. Insider attacks of the SecCloud protocol

The SecCloud protocol described above enjoys the desirable features of secure storage in cloud computing and achieving privacy cheating discouragement by designated verifier signature. Regarding the security of the data storage protocol in cloud computing, three insider attacks should be extra considered in this protocol.

1) An adversary, who is delegated some computational information rather than the private keys, may simulate to forge valid signatures to demonstrate the authenticity of the stored data to convince others.

2) An adversary may break its confidentiality and then arbitrarily modify the stored data to compromise the data integrity or reveal the confidential data or in both of cases.

3) An adversary may obtain the private information such as a cloud user's identity information.

Therefore, stronger adversaries may exist in the real cloud environment. For example, suppose either a cloud user or

a cloud storage service provider has revealed $e\hat{}(Q_{ID}; sk_{CS})$ (both the cloud user, cloud storage service provider can compute) to an adversary C, then the following insider attacks are possible:

Authentication attack: Assume that a cloud user (U) has sent an authenticated information fM; g to the cloud storage service provider (CSSP) according to the designated verifier signature in the step of Data signing. However, if the value of $e\hat{}(Q_{ID}; sk_{CS})$, obtained by an adversary C in the process of communication between U and CSSP, is insider leaked. After that, C possessed the value of $e\hat{}(Q_{ID}; sk_{CS})$. For each data block m_i with its

signature $_i$, there is

$_i = e\hat{}(U_i + H_2(U_i km_i ki_i)Q_{ID}; sk_{CS})$

$= e\hat{}(U; sk_{CS})\hat{}e(H_2(U_i km_i ki_i)Q_{ID}; sk_{CS})$

$= e\hat{}(Q_{ID}; sk_{CS})^{ri}\, e\hat{}(Q_{ID}; sk_{CS})^{H2(Uikmikii)}$

$= {}_{e\hat{}(QID;\, skCS)}{}^{r_i+H}2^{(U_i k^{m_i} k^{i_i})}$:

Hence, the adversary C, disguising as a legitimate cloud user, can generate the valid signatures that are designated to CSSP on any message M = $fm_1; m_2; :::; m_n g$ 2 Z_q by the following algorithm:

For each data block m_i , C picks up a random number r_i and computes

$U_i = r_i Q_{ID}$;

$_{i = e\hat{}(QID;\, skCS)}{}^{r_i +H}2^{(U_i\, km_i\, ki_i)}$:

Then the signature of each block m_i is $_i = (U_i ; _i)$. Due to = f $_i$ g, the forgery of data-signature pairs are fM ; g.

Therefore, no one, including a cloud user and CSSP, can distinguish this forged signature from the original signature produced by the cloud user U or the adversary C.

Confidentiality attack: In this SecCloud protocol, a cloud user and a cloud storage service provider should pre-compute their own session keys $k_{ID;CS}$ and $k_{CS;ID}$ in order to guaranteing the confidentiality of data trans-mission. However, similar to the Authentication attack above, an adversary C may obtain the insider leaked val-ues of $e\hat{}(Q_{ID}; sk_{CS})$ or $e\hat{}(sk_{ID}; Q_{CS})$ or both of them. Then C can break the confidentiality of data transmission to arbitrarily modify the stored data to compromise the data integrity or reveal the confidential data or in both of cases, since the session keys that

$k_{ID;CS} = H_3(\hat{}e(sk_{ID}; Q_{CS}))$; $k_{CS;ID} = H_3(\hat{}e(sk_{CS}; Q_{ID}))$

$= H_3(\hat{}e(s\, Q_{CS}; Q_{ID}))$

$= H_3(\hat{}e(Q_{CS}; s\, Q_{ID}))$

$= H_3(\hat{}e(Q_{CS}; sk_{ID})) = k_{ID;CS}$:

Therefore, if the adversary C obtains the value of $e\hat{}(Q_{ID}; sk_{CS})$ or $e\hat{}(sk_{ID}; Q_{CS})$ or both of them, C cloud easily calculates their session keys and launches confiden-tiality attack successfully to fetch the data-signature pairs of fM; g.

4.1 The new secure data sharing protocol for cloud storage

In this section, we propose a new secure data shared scheme for storage in cloud computing. Similar to the SecCloud scheme above, our scheme also involves a ID-based strong designated verifier signature (IDSDVS)[33]. However, our auditing protocol with the IDSDVS is more security than the SecCloud scheme, since it has the secure properties of non-delegatability, non-transferability, unforgeability and privacy of signers identity. A. System initialization

The cloud storage system includes a number of cloud servers and cloud users. Cloud users outsource their data to cloud servers, which are controlled by one or more cloud service provider. Here, as figure Fig.1 shows, we consider a cloud service provider(CSP) and a cloud user to describe our secure data sharing scheme for storage in cloud computing with security and privacy considerations. (Table Notions)

Setup( ): The Initialization Procedure, which is a trusted third-party, takes a security parameter as input to generate the system parameters and master secret keys. Let G and G_T be two cyclic groups and the orders of both are q. Then Initialization Procedure makes a bilinear mapping $e\hat{} : G \; G ! G_T$, which is admissible. There are some secure hash functions chosen as follow, H_1 : f0; 1g ! G, H_2 : f0; 1g G_T ! G, H_3 : G ! Z_q, H_4 : G_T ! Z_q. Furthermore, Initialization Procedure randomly chooses a number s 2 Z_q as the system master secret keys and selects g 2 G, which is a random generator. It computes its public key $g_1 = g^s$. Finally, the Initialization Procedure outputs a system master public

key mpk = (G; G_T ; e;^ q; g; g_1; H_1; H_2; H_3; H_4). The system master secret key msk = s is safely kept.

B. Cloud user's secure data storage

In order to obtain the cloud storage service from CSP, in the first, a cloud user should interact with Initialization Procedure in a secure way to extract the system master public key mpk and its secret key. Then the cloud user will transmit its messages to the cloud servers. We consider that the cloud user and the cloud service provider's identities are U, CS and the messages are M = $fm_1; m_2; :::; m_n$ g 2 Z_q. And assume that CSP has allocated storage spaces for the messages M with space index set X = $fx_1; x_2; :::x_n g$.

KeyExtract(msk; U): The cloud user submits its identity U to Initialization Procedure. It takes msk and U as input and computes

$sk_U = Q^s_U$;

where $Q_U = H_1(U)$ is a public key of the cloud user and sk_U is its secret key. Then Initialization Procedure returns the secret key sk_U and the system master public key mpk.

Similarly, the Initialization Procedure takes msk and CSP's identity CS as input. Then it returns mpk and the secret keys $sk_{CS} = Q^s_{CS}$ to the the cloud service provider, where $Q_{CS} = H_1(CS)$.

Note that the two steps of Setup and KeyExtract can be executed off-line.

To verify the integrity of stored data, the cloud user should generate authentication blocks by signing every data block m_i of M. Moreover, the cloud user signs each block with a designated verifier signature as follow, which is only verified by the designated CSP.

DataSign(sk_U ; U; CS; mpk; M; X): This algorithm takes the secret key sk_U , the identities U andCS, the sys-tem master public key mpk and the n unsigned da-ta $Mfm_1; m_2; :::; m_ng$ with space index set X =

$fx_1; x_2; :::x_ng$ as input by the cloud user. A cloud user with an identity U sign the messages M for a CSP with an identity CS. For each data block m_i with the space index x_i, the cloud user randomly chooses $r_i; l_i 2 Z_q$ and computes

$a = e\hat{}(g; g^{li});$

$_i \quad 1$

$c_{i2} = H_2(m_ijjUjjx_i; a_i);$

$c_i = c_{i_1} \quad c_{i_2};$

$zi = g^{l_i} =skU^{H3(ci)};$

$_i = H_4(\hat{}e(Q_{CS}; g_1)^{ri});$

$_i = (c_i; z_i; _i):$

Then the cloud user securely transmits = fM; g to CSP by its session key $key_{U;CS}$, where $= f_i g(i = 1; 2; :::; n)$.

In order to guarantee the data confidentiality and ensure data transmission securely, the cloud user performs DataTran algorithm.

DataTran(M; ; $key_{U;CS}$) ! $fE_{keyU;CS}$ (M;)g: This algorithm takes the message M, its signature and the section key $key_{U;CS}$ as input. It outputs the ciphertext message $E_{keyU;CS}$ (M;). After identifying the cloud storage server, the cloud user randomly chooses y 2 Z_q

and calculates the session key $key_{U;CS}$ as following.

$Y \quad = g^y,$

$key_{U;CS} = H_4(\hat{}e(Q_{CS}; g_1)^y),$

where $g_1 = g^s$. Then the cloud user securely encrypts

= fM; g by its session key $key_{U;CS}$ and transmits the ciphertext $E_{keyU;CS}$ (M;) and the secure parameter Y to CSP, where $= f_i g(i = 1; 2; :::; n)$.

At the cloud side, when the cloud storage service provider receives the encrypted packets and secure parameter Y =

g^y, it decrypts $E_{keyU;CS}$ (M;) to extract fM; g by its own session key $key_{CS;U}$ as following.

$key_{CS;U} = H_4(\hat{}e(sk_{CS}; Y))$:

The cloud user and CSP are sharing a pair of symmetrical keys, since

$key_{CS;U} = H_4(\hat{}e(sk_{CS}; Y))$

$= \quad H_4(\hat{}e(Q^s_{CS}; g^y))$

$= \quad H_4(\hat{}e(Q^y_{CS}; g^s))$

$= \quad H_4(\hat{}e(Q_{CS}; g_1)^y)$:

Due to the hardness of Bilinear DiffieCHellman (BDH) problem, the session key is secure against the insider attacks.

Then CSP further needs to verify the valid of fM; g by the VerSign algorithm.

VerSign(U; CS; sk_{CS}; mpk;): The cloud service provider CS executes this algorithm and takes its own secret key sk_{CS}, the system master public key mpk and the message-signature pairs = fM; g from the cloud user U as input. For each message block m_i with its signature $_i = (c_i; z_i; _i)$, it behaves as follow:

$a^0_i = e\hat{}(z_i; g)\hat{}e(Q_U ; g_1)^{H3(ci)};$

$c^0 \quad = H \quad (m\ U \quad x\ ; a^0);$

$^i 2 \quad 2 \quad _i jj\ jj \quad i \quad i$

$c_i{}^0{}_1 \quad = c_i = c_i{}^0{}_2 :$

Finally, CSP checks

$? \quad ; \quad c_i{}^0{}_1$

$_i = H_4(e(sk_{CS} \quad));$

where returns true if correct for all blocks, and otherwise, returns false.

After verifying the valid of fM; g, if the verifying result is true, it means that the messages M are securely and correctly transmitted to CSP. And CSP would store the messages M according the storage space index set $X = fx_1; x_2; :::x_ng$. Otherwise, CSP broadcasts an "error message" to the cloud user to alert the invalid of fM; g.

5 ANALYSIS OF THE PROPOSED SCHEME

In Section 4, we proposed a new secure data sharing protocol with a ID-based strong designated verifier signa-ture (IBSDVS) scheme, which has secure properties of non-delegatability, non-transferability, unforgeability and privacy of signer's identity. Then in this section we provide a detailed analysis of security and performance.

A. Security analysis

1) Against insider attack: Our protocol can resist the insid-er attack due to the fact that the designated verifier signature adopted in our protocol with a non-delegatability property []. The cloud user as a signer implements DataSign algorithm and the cloud storage service provider (CSSP) as a verifier implements Verify algorithm, which both may launch the insider attack by leaking some effective information rather than the secret keys sk_U , sk_{CS}. Hence, an adversary gotten the leaking information may forge a valid signature without the secret keys of the cloud user's sk_U and the CSSP's sk_{CS}. However, DataSign algorithm produced the signature is $_i = (c_i; z_i; _i)$, where

$c_i = c_{i_1} \quad c_{i_2}$

$= \quad g^r i \ \ H_2(m_ijjUjjx_i; e\hat{}(g; g^{sl}i));$

$zi = g^{l_i} =skU^{H3(ci)};$

$_i = H_4(\hat{}e(Q_{CS}; g_1)^{ri})$

$= \quad H_4(\hat{}e(Q^s_{CS}; g^{ri}) = H_4(\hat{}e(sk_{CS}; g^{ri}):$

Our protocol is non-delegatability. On the one hand, even if an adversary has known $e\hat{}(g; g_1)$ and $e\hat{}(Q_{CS}; g_1)$, $r_i; l_i 2 Z_q$ can not be obtained and are chosen randomly for each data block m_i. Furthermore, the key component is $z_i = g^l i =sk_U{}^{H3(ci)}$, due sk_U is unknown unless it can be delegated directly by the signer. Otherwise, it is impossible to compute a valid signature. On the other hand, the simulate algorithm produces

the signature is $_i = (c_i; z_i; _i)$, where $c_i; z_i 2 G$ are selected randomly and

$_i = H_4(\hat{}e(sk_{CS}; c_{i1}))$

$= \quad H_4(\hat{}e(sk_{CS}; c_i = H_2(m_ijjUjjx_i; a_i)))$

= H_4(^e(sk_{CS}; c_i=H_2($m_i$jjU jjx_i; e^(z_i; g)^e(sk_{CS}; g)$^{H3(ci)}$))):

Even though a simulator has delegated e^(sk_{CS}; g) rather than sk_{CS}, it can not simulate a valid signature due c_i ; z_i 2 G chosen randomly by the simulator for each data block m_i are different from c_i; z_i 2 G randomly selected by the CSSP as a verifier. Hence, it must obtain the secret key sk_{CS}, e^(sk_{CS}; c_{i1}) can be simulated successfully.

In summary, unless knowing the secret keys sk_U or sk_{CS}, can it produce the valid signature by a third-party. Com-paring with Wei's scheme[28], our protocol can resist the insider attack by adopting a designated verifier signature that is non-delegatability. Moreover, we show proof of non-delegatability as follow and other secure properties such as non-transferability, unforgeability and privacy in our protocol.

2) Proofs of secure properties in our protocol: Let $_g$, $_e$, $_t$ respectively on behalf of the runtime of a multiplication in group G, the runtime of a bilinear pairing evaluating, the runtime of a exponential operation.

Property 1. (Non-delegatability) Imaging that the KEA-BDH assumption holds with a probability (1 "^{00}). Then the our data sharing protocol adopted the IBSDVS scheme is non-

delegatable with (t; t^0; "; "^{0}) in the context of Definition 4, in which "^{0} > (1 "^{00})=("=q_{H2} 1=q) and t^0 < t+(q_{H1} +q_{H2}) $_g$.

Proof. Assuming that mpk = (G; G_T ; e;^ q; g; g_1), where g_1 = g , and Q_{CS} = g . An extractor K existed that extracts the secret key of either a signer or a designated verifier by inputting the signature $_i$ and accessing a black-box oracle of an algorithm F.

Let $F_{U;CS;M}$ be a simulator token (U; CS; M) as input. For each data block m_i, K supplies hash oracles H_2, H_3 and H_4 and maintains tables of T H_1; T H_2; T H_3; T H_4 to avoid issuing repeated queries to the oracles.

H_1 Query: Given a query ID, K selects randomly $!_{ID}$ 2 Z_q. If $!_{ID}$ has been in the table T H_1, choosing an another number. Then it tosses a coin b_{ID} so that

Pr[b_{ID} = 1] = , which is to be determined later.

If b_{ID} = 0, K sets Q_{ID} = (g)$^{!ID}$; otherwise, it sets Q_{ID} = g$^{!ID}$. In either case, K stores the tuple (ID; Q_{ID}; b_{ID}; $!_{ID}$) into table T H_1 and returns ^{Q}ID ^{to} $F_{U;CS;M}$.

H_2 Query: Given a query ($m_i$jjU jjx_i; a_i), K randomly chooses d_i 2 Z_q. If d has been in the table T H_2, choosing an another number. Then it returns Q_i = g^{d_i} and stores ($m_i$jjU jjx_i; a_i; d_i; Q_i) into T H_2.

H_3 Query: Given a query c_i, K randomly chooses c_l 2 Z_q. If c_l has been in the table T H_3, choosing an another number and storing (c_i; c_l) into T H_3. Then it returns c_l.

H_4 Query: Given a query T_i, K randomly chooses $_i$ 2 Z_q. If has been in the table T H_4, choosing an another number and storing (T_i;) into T H_4. Then it returns $_i$.

KeyExtract Query: Given an identity ID, K retrieves the tuple (ID; Q_{ID}; b_{ID}; $!_{ID}$) from table T H_1. If b_{ID} =

1, K calculates the cloud user's secret key sk_{ID} = (g)$^{!ID}$ and returns to $F_{U;CS;M}$. Otherwise, b_{ID} = 0, K aborts.

K implements $F_{U;CS;M}$ to generate a signature $_i$ = (c_i; z_i; $_i$). Then K implements $F_{U;CS;M}$ again to gener-ate a forgery $_i^0$ = (c^0_i; z_i^0; $_i^0$) with fixed coins. When ($m_i$jjU jjx_i; a_i), where a_i = e^(z_i; g)^e(Q_U ; g)$^{H}3^{(c_i)}$, is issued to oracle H_2. The answers of the oracle H_2 for the first and the second run of $F_{U;CS;M}$ are different. Q^0_i = c_i=g $^{!}$ is the answer and ! is chosen by K.

There are further two cases:

If (H_3(c^0_i); z_i^0) 6= (H_3(c_i); z_i), K produces the cloud user's secret key sk_U as in the simulation of Extract oracle.

Otherwise, K recovers T_i^0 = (^e(Q_{CS}; R_i^0)) from hash table H_4, where R_i^0 = c^0_i=Q^0_i = c_i=Q^0_i = g $^{!}$. Then we regard $F_{U;CS;M}$ as a function by inputting (g_1; Q_{CS}), and it outputs (R_i^0; e^(g; g) $^{2!}$).

Furthermore, we take the KEACBDH assumption to claim that the event happens with a probability (1 "^{00})

that there is an algorithm $F_{U;CS;M}$ on the same random coins, which produces an output (!; R^0; e^(g; g) $^{2!}$). With the value !, K computes and sk_{CS} = g_1 . Then the private key of the cloud storage service provider is sk_{CS}.

Assume that (H_3(c^0_i); z_i^0) 6= (H_3(c_i); z_i) with a probability

2 [0; 1]. Then the probability of a successful extraction of a cloud users private key is "$_0$ = "("=q_{H2} 1=q) according to the general forking lemma [26], Then the prob-ability of a successful extraction of a CSSPs private key is "$_1$ = (1)(1 "^{00})"("=q_{H2} 1=q). Hence, the probability of a successful extraction of a private key is

" = "$_0$ + "$_1$ > (1 "^{00})"("=q_{H2} 1=q):

And the runtime is about

t^0 < t + (q_{H1} + q_{H2}) $_g$:

Property 2. (Non-transferable) Our proposed data sharing protocol with an IBSDVS scheme is satisfying the Definition 3 and non-transferable perfectly.

Proof. The designated verifier cloud storage service provider (CSSP) can simulate to generate a signature on the original message M as follow: For every data block m_i, CSSP ran-domly chooses c^0_i; z_i^0 2 G and computes

a^0_i = e^(z_i^0; g)^e(Q_U ; g_1)$^{H}3^{(c0}i^{)}$;

c^0_{i2} = H_2($m_i$jjU jjx_i; a^0_i);

c^0_{i1} = c^0_i=c^0_{i2} ;

$_i^0$ = H_4(^e(sk_S; c^0_{i1}));

$_i^0$ = (c^0_i; z_i^0; $_i^0$):

Then the simulate signature is= f $_i^0$g$_{i2(1;2;:::;n)}$.

signature $_i$ = (c_i; z_i; $_i$) on m_i is determined by the random numbers r_i; l_i 2 Z_q, while the simulate signature of m_i is $_i$ = (c^0_i; z_i^0; $_i^0$), which is determined by the random numbers c_i; z_i 2 G. So a random pair (r_i; l_i) maps a random pair (c_i; z_i) and for any pair (c_i; z_i), there is a corresponding pair (r_i; l_i). Because of both (r_i; l_i) and (c_i; z_i) randomly

chosen, the signatures $_i$ = (c_i; z_i; $_i$) and $_i$ = (c^0_i; z_i^0; $_i^0$) are indistinguishable. Hence, even if an adversary A has secret

keys of the cloud user, the CSSP or both, it is impossible to distinguish signatures from ^{0}.

Property 3. (Unforgeability) If the weak GBDH assumption (t; ") holds, the data sharing protocol with an IBSDVS scheme is unforgeable with (t^0; q_{H1} ; q_{H2} ; q_K ; q_{DSign}; $q_{V\,er}$; q_{Sim}; "^{0}), where

t < $2t^0$ + 2(q_{H1} + q_{H2} + q_K + 2q_{DSign}) $_g$ + 2(2q_{DSign} +

3q_{Sim} + 3$q_{V\,er}$) $_e$ + 2(q_{DSign} + q_{Sim} + $q_{V\,er}$) $_t$, and

" > "^{0}=[2(q_K)^2]("^{0}=[2(q_K)$^2q_{H2}$] 1=q):

Proof. Given the data sharing protocol with an IBSDVS scheme is unforgeability, then by using it an algorithm B is built to solve the weak GBDH problem. Given (g; g ; g ; g) and the oracle O , that a random instance of the weak GB-DH problem, the algorithm B purposes to obtain e^(g; g) . Furthermore, it works as follow:

Setup: B invokes an adversary A on inputting mkp = (G; G_T ; e;^ q; g; g_1), where g_1 = g . Note that the master secret key is msk = that is unknown to B. It then simulates oracles for A as below.

Query: B simulates and issues random oracles H_1; H_2; H_3; H_4; KeyExtract in the same way as K dose in Property 1 and also including data signing oracle and verification oracle.

DataSign Query: Given a query (U; CS; M), B retrieves the tuple (U; Q_U ; b_U ; !$_U$) from T H_1. Here we need to distinguish two cases:

1) If b_U = 1, produces the cloud user's private key sk_U as in the simulation of KeyExtract oracle in Property 1 and then calculates the signature $_i$ by taking the DataSign algorithm for each data block with inputting (sk_U ; U; CS; mpk; m_i) by accessing to random oracles H_2; H_3 and H_4.

2) If b_U = 0, B chooses randomly z_i 2 G and c_{ij} 2 Z, then computes c_i = $g^{c_{ij}}$, $_{ai}$ = e^(z_i; g)^e(Q_U ; g_1)$^{H3(ci)}$, Q_i = H_2($m_i$jjUjjx$_i$; a_i), ^ $^{c_{ij}}$ d_i R = c_i=Q_i and $_i$ = H_4((Q_V ; g_1)) , where d_i is in the entry ($m_i$jjUjjx$_i$; a_i; d_i; Q_i) of table T H_2 for the query ($m_i$jjUjjx$_i$; a_i). B asks random oracles H_2; H_3 and H_4 for related hashing operations.

Note that since the scheme is perfectly non-transferable (Property 2), the Sign oracle is per-fectly simulated.

Simulate Query: This can be answered by B in a similar way with the simulation of DataSign oracle. The differ-ence is that now B generates the signature from the point of a designated verifier.

Verify Query: Given a query (m_i; U; CS; $_i$), where $_i$ = (c_i; z_i; $_i$), B retrieves the tuple (CS; Q_{CS}; b_{CS}; !$_{CS}$) from T H_1. Again, there are two cases:

1) If b_{CS} = 1, B generates the user secret key sk_{CS} as in the simulation of KeyExtract oracle, and then

verifies the signature $_i$ by running the Verify algorithm on inputting (m_i; U; CS; sk_{CS}; mpk; $_i$).

B looks up hash tables H_2; H_3 and H_4 for the cor-responding hashing operation. If there is no match,

B simply returns 0 indicating that the signature is invalid. As we assume that A would not use a value before asking the corresponding random oracle on the input, the event will not happen that the signature is valid but B returns 0 when A luckily guesses a hashing result before the hashing query is issued.

2) If such a tuple is not found, B looks up the hash table T H_4 indexed by $_i$. If there is no match, it simply returns 0. Else B re-trieves the match tuple (T_i; $_i$) in T H_4. It com-putes a_i^0 = e^(z_i; g)^e(Q_U ; g_1)^{H}3$^{(c_i)}$, and Q_i^0 = H (m U x ; a^0), R^0 = c =Q^0 . It looks up hash 2 $_i$jj jj i i i i i tables T H_2 and T H_3 for the corresponding hashing operation. If there is no match, it simply returns 0. 0$^!$CS Else B asks the oracle O on inputting ($R_{i!}$; T_i) to obtain a decision bit. If T_i 6= e^(g; R_i^0 CS) , B returns 0 indicating that the signature is invalid; otherwise it returns 1.

A i i Forge: generates its forgery, (U ; CS ; m ;) for each data block m_i , where $_i$ = (c_i ; z_i ; $_i$). B then checks the va-lidity of the forgery as in the simulation of Verify. If invalid, it aborts; otherwise, it retrieves the two tuples (U ; Q_U ; b_U ; !$_U$) and (CS ; Q_{CS}; b_{CS}; !$_{CS}$) from table T H_1. If b_U = 1 or b_{CS} = 1, B aborts. The validity of guarantees that there is a tuple (T_i ; $_i$) in T H_4, a tuple (c_i ; c_{ij}) in T H_3 and a tuple (m_i ; a_i ; d_i ; Q_i) in T H_2 under our assumptions about random oracles. B rewinds A to the status of querying oracle H_2 on inputting (m_i ; a_i). It sets Q^0 $_i$ = c_i =g , and answers A with Q^0 $_i$. It then continues to simulate oracles for A as in the Query phase. Suppose that, A produces a successful forgery once again, say (U^0 ; CS^0 ; m^0 $_i$; $_i^0$) where $_i^0$ = (c^0 $_i$; z_i^0 ; $_i^0$). If (U^0 ; CS^0 ; m^0 $_i$; $_i^0$) 6= (U ; CS ; m_i ; $_i$), B aborts. Other-wise, there are two cases:

1) If (H (c^0); z^0) = (H (c); z), we have that 3i i 6 3i i 0 z_i $^{skU^{H3(c_i)}}$ = g^{l_i} = $g^{l_i^0}$ = z_i0 $^{skU^{H3(c_i)}}$. Hence, recovers sk by computing sk = B 0 1^U U $_{(zi=zi^0)}$ $^{(H3(ci}$) H_3 (c_i)).

Note that sk_U = ((g)$^!$U) . B can compute 1 e^(g; g) = e^($sk_U^{!U}$; g).

2) If (H_3(c^0 $_i$); z_i^0) = (H_3(c_i); z_i), we have that R_i^0 = c^0 $_i$=Q^0 $_i$ = c_i =Q^0 $_i$ = g . B retrieves the tuple (T_i^0 ; $_i^0$) from table T H_4. Note that T_i^0 = e^(Q_{CS} ; R_i^0) and Q_{CS} = (g)$^!$CS . Hence, B can compute e^(g; g) = $_{(Ti^0)^{!CS^1}}$.

In either case, B solves the given instance of the weak GBDH problem.

Therefore, the probability of a successful forgery of (U ; CS ; m_i ; $_i$) is

> "^{0}=[2(q_K)^2]("^{0}=[2(q_K)$^2q_{H2}$] 1=q):

Fig. 1. Performance comparison.

And the runtime is about

t $< 2t^0 + 2(q_{H1} + q_{H2} + q_K + 2q_{DSign})$ g

\+ $2(2q_{DSign} + 3q_{Sim} + 3q_{V er})$ e

\+ $^{2(q}$DSign $^{+ q}$Sim $^{+ q}$V er$^{)}$ t

considering only the runtime of g, e and t.

Property 4. (Privacy) Our proposed data sharing protocol with an IBSDVS scheme is satisfying the Definition 5 and privacy perfectly.

Proof. In this paper, our protocol mainly achieves the privacy of the cloud user and the CSSP's identities. In Property 1, without delegating the secret keys sk_U and sk_{CS}, an adversary can not produce a valid signature and also can not verify the signature successfully. And in Property 2, the signature $_i$ signed by the cloud user is indistinguishable from $_i$ signed by the designated verifier CSSP, even if given the secret keys sk_U and sk_{CS}. Hence, It is not computationally feasible for an adversary to determine the key pair of signature $_i$ without knowing a designated verifier CSSPs secret key sk_{CS} or a signer the cloud users secret key sk_U .

B. Performance analysis

1.1

Let g, e, t respectively on behalf of the runtime of a multiplication in group G, the runtime of a bilinear pairing evaluating, the runtime of a exponential operation.

To analyze the performance of our data sharing protocol, we compare our protocol with current IBSDVS schemes about computation cost and signature size in Table 1, where G Z^2_q means that this signature algorithm contains three elements that one is an element in G and two are in Z_q. Note that, for each data block m_i, the computation of g^li and g^ri in those signature algorithms is taken as one scalar multiplication since they can be computed sequentially when they are sorted. Finally, from the Table 1, we obtain the fact that the IBSDVS scheme adopted in our data sharing protocol for cloud storage is more efficient than other IBSDVS scheme on both signature size and computation cost.

Table 1. Performance comparison.

Scheme	Signature-size	Sign-cost	V
[1]	$\mathbb{G}^2 \times \mathbb{Z}_q^4$	$4\tau_t + 4\tau_e + \tau_p$	4
[13]	$\mathbb{G}_T^2$	$2\tau_e + 2\tau_p$	2
[17]	$\mathbb{G} \times \mathbb{Z}_q^2$	$\tau_e + 2\tau_p$	2
[19]	$\mathbb{G}^3$	$3\tau_p$	3
[20]	$\mathbb{Z}_q$	τ_e	1
Our protocol	$\mathbb{G}^2 \times \mathbb{Z}_q$	$\tau_t + 2\tau_e + 2\tau_p$	3

6 CONCLUSION

In this paper, we construct a cloud storage environment to propose a novel secure data sharing protocol that can resist the insider attack due to the fact that the designated verifier signature adopted in our protocol with a non-delegatability property. We have defined the concepts of non-delegatability, unforgeability,non-transferable and privacy of signer's identity and proposed a novel data sharing protocol to achieve the security goals. By the extensive security analysis and performance analysis, our data sharing protocol is proved that not only can resist the insider attack with enjoying all secure properties, but also can be more efficient

7 REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[3] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud com-puting: Vision, hype, and reality for delivering it services as computing utilities," in High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on. Ieee, 2008, pp. 5–13.

[4] K. Popovic and Z. Hocenski, "Cloud computing security issues and challenges," in MIPRO, 2010 proceedings of the 33rd international convention. IEEE, 2010, pp. 344–349.

[5] D.-G. Feng, M. Zhang, Y. Zhang, and Z. Xu, "Study on cloud computing security," Journal of Software, vol. 22, no. 1, pp. 71–83, 2011.

[6] Y. Chen, V. Paxson, and R. H. Katz, "Whats new about cloud computing security," University of California, Berkeley Report No. UCB/EECS-2010-5 January, vol. 20, no. 2010, pp. 2010–5, 2010.

[7] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007, pp. 584–597.

[8] A. Giuseppe, B. Randal, C. Reza et al., "Provable data possession at untrusted stores," Proceedings of CCS, vol. 10, pp. 598–609, 2007.

[9] C. Erway, A. Kupc¨¸u,¨ C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009, pp. 213–222.

[10] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Computer Security–ESORICS 2009. Springer, 2009, pp. 355–370.

[11] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," Parallel and Distributed Systems, IEEE Transactions on, vol. 24, no. 9, pp. 1717–1726, 2013.

[12] M. Mowbray and S. Pearson, "A client-based privacy manager for cloud computing," in Proceedings of the fourth international ICST conference on COMmunication system softWAre and middlewaRE. ACM, 2009, p. 5.

[13] E. Bertino, F. Paci, R. Ferrini, and N. Shang, "Privacy-preserving digital identity management for cloud computing." IEEE Data Eng. Bull., vol. 32, no. 1, pp. 21–27, 2009.

[14] S. Pearson, Y. Shen, and M. Mowbray, "A privacy manager for cloud computing," in Cloud Computing. Springer, 2009, pp. 90–106.

[15] Y. Deswarte, J.-J. Quisquater, and A. Sa¨ıdane, "Remote integrity check-ing," in Integrity and Internal Control in Information Systems VI. Springer, 2004, pp. 1–11.

[16] H. Shacham and B. Waters, "Compact proofs of retrievability," in Advances in Cryptology-ASIACRYPT 2008. Springer, 2008, pp. 90– 107.

[17] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication netowrks. ACM, 2008, p. 9.

[18] R. C. Merkle, "Protocols for public key cryptosystems," in 2012 IEEE Symposium on Security and Privacy. IEEE Computer Society, 1980, pp. 122–122.

[19] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in Advances in CryptologyłASIACRYPT 2001. Springer, 2001, pp. 514–532.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in INFOCOM, 2010 Proceedings IEEE. Ieee, 2010, pp. 1–9.

[21] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 5, pp. 847–859, 2011.

[22] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," Parallel and Distributed Systems, IEEE Transactions on, vol. 23, no. 12, pp. 2231– 2244, 2012.

[23] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in Proceedings of the 2011 ACM Symposium on Applied Computing. ACM, 2011, pp. 1550–1557.

[24] P. Golle and I. Mironov, "Uncheatable distributed computations," in Topics in CryptologyłCT-RSA 2001. Springer, 2001, pp. 425–440.

[25] F. Monrose, P. Wyckoff, and A. D. Rubin, "Distributed execution with remote audit," in NDSS, vol. 99, 1999, pp. 3–5.

[26] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable com-puting: Outsourcing computation to untrusted workers," in Advances in Cryptology–CRYPTO 2010. Springer, 2010, pp. 465–482.

[27] L. Wei, H. Zhu, Z. Cao, W. Jia, and A. V. Vasilakos, "Seccloud: Bridging secure storage and computation in cloud," in Distributed Computing Sys-tems Workshops (ICDCSW), 2010 IEEE 30th International Conference on. IEEE, 2010, pp. 52–61.

[28] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," Information Sciences, vol. 258, pp. 371–386, 2014.

[29] R. Canetti, B. Riva, and G. Rothblum, "Verifiable computation with two or more clouds," in Workshop on Cryptography and Security in Clouds, 2011.

[30] S. Saeednia, S. Kremer, and O. Markowitch, "An efficient strong designated verifier signature scheme," in Information Security and Cryptology-ICISC 2003. Springer, 2004, pp. 40–54.

[31] W. Susilo, F. Zhang, and Y. Mu, "Identity-based strong designated veri-fier signature schemes," in Information Security and Privacy. Springer, 2004, pp. 313–324.

[32] B. Kang, C. Boyd, and E. Dawson, "A novel identity-based strong designated verifier signature scheme," Journal of Systems and Software, vol. 82, no. 2, pp. 270–273, 2009.

[33] H. Tian, X. Chen, F. Zhang, B. Wei, Z. Jiang, and Y. Liu, "A non-delegatable strong designated verifier signature in id-based setting for mobile environment," Mathematical and Computer Modelling, vol. 58, no. 5, pp. 1289–1300, 2013.