2010 IEEE 30th International Conference on Distributed Computing Systems Workshops

SecCloud: Bridging Secure Storage and Computation in Cloud

Lifei Wei, Haojin Zhu, Zhenfu Cao, Weiwei Jia,
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
Email: {weilifei, zhu-hj, zfcao, jlss}@sjtu.edu.cn

Athanasios V. Vasilakos
Department of Computer and Telecommunications Engineering
University of Western Macedonia
Kozani, Greece
Email: vasilako@ath.forthnet.gr

***Abstract*—Cloud computing becomes a hot research topic in the recent years. In the cloud computing, software applications and databases are moved to the centralized large data centers, which is called *cloud*. In the cloud, due to lack of physical possession of the data and the machine, the data and computation may not be well managed and fully trusted by cloud users. Existing work on cloud security mainly focuses on cloud storage without taking computation security into consideration. In this paper, we propose *SecCloud*, a novel auditing scheme to secure cloud computing based on probabilistic sampling technique as well as designated verifier technique, which aims to consider secure data storage, computation and privacy preserving together. We also discuss how to optimize sampling size to minimize the auditing cost. Detailed analysis and simulations have demonstrated the effectiveness and efficiency of the proposed scheme.**

***Keywords*-Designated verification; Privacy preserving; Batch verification; Data storage secure; Cloud computing.**

I. Introduction

The recent development of cloud computing has shown its potential to reshape the current way IT hardware is designed and purchased. Among numerous benefits, cloud computing offers customers a more flexible way to obtain computation and storage resources "on demand". Rather than owning (and maintaining) a large and expensive IT infrastructure, customers can now rent the necessary resources as soon as, and as long as, they need them. Thus, customers can not only avoid a potentially large up-front investment (which is particularly attractive for small companies and startups), they may also be able to reduce their costs through economies of scale and by paying only for the resources they actually use [1].

Even though cloud computing is envisioned as a promising service platform for the next-generation Internet, security and privacy is one of major challenges which prevent its wide acceptance in practice. Different from the traditional computing model where the users have a full control of data storage and computing, cloud computing means the management of physical data and machine is delegated to the cloud service provider while the users can only retain some control over the virtual machines. Therefore, the correctness of data storage and computation is putting at a risk due to lack of control of data security for data owners. In this study, we further classify the cloud data security into two major classes: *Cloud Storage Security* and *Cloud Computation Security*, where the former is referred to ensuring the integrity of outsourced data stored at untrusted cloud server while the latter refers to checking the result correctness of the outsourced computation performed by untrusted cloud server.

The current research on secure cloud computing still focuses on storage security. However, outsourced computation security receives less attention. For sake of saving computational resources, the service provider may not have performed the necessary computations but claims to have done so. Further, the centralized architecture makes cloud servers more easily to be a single point of failure, which has been witnessed by the recent meltdown of Google gmail system [2]. Under byzantine failure or even external attacks, the cloud may perform unreliable computation operations while choose to hide the computations errors for their benefits. This cheating behavior of cloud providers, if undetected, may render the results useless. Even from the point of accountability, some secure cloud computing mechanism should be in place to meet the needs of deciding whether cloud provider or the users should be responsible for it once there is any problem taking place. Note that, it is quite natural for the provider to initially suspect a problem with the customer's software, and vice versa [3].

Due to the limited computing and communication resources, cloud users cannot afford the cost incurred by computing resulting verification. One promising approach for relieving cloud users from expensive verification cost is introducing a trusted auditor who is behalf of the users to audit cloud computing. Even though public auditability has been proposed in the context of secure storage in cloud [4], [5], public auditability in secure cloud computing receives less attentions. More closely related references are secure remote computation in distributed system [6]. However, none of the proposed schemes are targeted at secure cloud computing. Further, privacy preserving is an critical issue for secure cloud computing while few of existing researches take it into consideration.

To achieve secure computing audit for cloud, one straight-

978-0-7695-4079-5/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDCSW.2010.36

forward method is double-check each computing result. The cloud provider may provide the inputs and overall computing result to auditor, which will follow an identical procedure to compute the same result and then compare it with the one provided by cloud provider. This simple scheme may lead to a waste of I/O and computing resources. Note that data transfer bottlenecks are regarded top ten obstacles which may prevent the overall success of cloud computing [1]. In [7], a Commitment-Based Sampling (CBS) technique is introduced in the conventional grid computing however it does not take the privacy issue into consideration. In this paper, we introduce a privacy preserving CBS technique by integrating CBS with the designated verification technique.

The contributions of this paper can be summarized as follows.

- First, we model the problems in the cloud computing by giving the formal definitions of *uncheatable* and *privacy preserving*.
- Second, we propose an uncheatable framework to achieve both the data storage and computing security.
- Third, we propose an advanced scheme to achieve privacy preserving through the designated verification technique and computational efficiency through batch verification technique.
- Lastly, we discuss how to minimize the auditing cost by choosing the optimal sampling size.

To the best of our knowledge, this work is the first effort towards jointly considering both of cloud storage security and computation security. The remainders of this paper is organized as follows. A review on related work is given in Section II. Section III presents the system model and our design goals. Some preliminary knowledge is given in Section IV. Later, we first provide an overview of our scheme in Section V and then present our scheme with the performance optimization method in Section VI. Section VII gives the security analysis and performance evaluation. Finally, Section VIII concludes this whole paper.

II. Related Work

Security and privacy in cloud computing has received extensive attentions recently. Generally speaking, the research on cloud computing falls into the two cases: *cloud storage security* and *cloud computing security*.

Cloud storage security mainly addresses the secure outsourced storage issue. In [8], Ateniese *et al.* first defined a model for *provable data possession* (PDP), which allowed a client that had data stored at an untrusted server to verify that the server possessed the original data without retrieving it. They utilized RSA-based homomorphic tags for auditing outsourced data, but they did not consider the dynamic data storage. In their later work [9], Ateniese *et al.* proposed a partially dynamic version of the PDP scheme using symmetric key cryptography. However, it did not support public auditability. In [10], a similar partial dynamic data storage in the cloud scenario was proposed and this challenge-response protocol could both determine the data correctness and locate possible errors. Juels *et al.* [11] proposed the definition of *proof of retrievability* (PoR), which used spot-checking and error-correcting codes to ensure both possession and retrievability for data file on archive service system. Wang *et al.* [5] first achieved both public verifiability and dynamic data storage operations employing an Third Party Auditor and improving the Proof of Retrievability model [12] by using classic Merkle Hash Tree [13] construction for BLS [14] based block tag authentication. Later, they proposed a scheme achieving privacy preserving public verifiability as well as the dynamic data storage operations in [4] by utilizing the public key based homomorphic authenticator and uniquely integrate it with random mask technique. The further work explored the technique of bilinear aggregate signature for TPA can verify data auditing in a complexity of $O(n)$. Erway *et al.* [15] proposed the first construction of dynamic provable data possession, which extended the PDP model in [8] to achieve provable updating stored data using rank-based authenticated skip lists.

Compared with secure cloud storage, secure cloud computing still receives less attentions. The most related researches include secure remote computations. Golle and Mironov [16] proposed a ringer scheme in distributed computing where the supervisor sent to the participant some pre-computed results without disclosing the corresponding inputs. A recent research by [6] was dividing sequential tasks into smaller subtasks, permuting them among participants and then enabling the detection of individual and colluding malicious participants. For secure grid computing, Du *at el.* [7] proposed an uncheatable grid computing using a commitment-based sampling technique to detect whether the participant was cheating or not. Their scheme handled generic computations gracefully and also proposed two improvement of their basic scheme.

However, the existing researches discuss secure computation in the general parallel or grid computing case without considering the specific requirement of cloud computing. Further, the privacy issue is not taken into consideration either. In this study, we aim to introduce the concept of secure computation in the context of cloud computing for the first time.

III. Problem Formulation

In this section, we will present our system architecture, model formulation and design goals.

A. Models of Cloud Computing

We consider a general cloud computing model constituted of n cloud computing servers, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$, which may be under the control of one or multiple *cloud service providers*

(**CSPs**). A cloud user (**CU**) could send its service request to CSPs, which allocate cloud computing resources by means of customized Service Level Agreements (**SLAs**). For example, to perform a batch-processing task, by employing the existing programming abstraction techniques such as MapReduce [16] and its open-source counterpart Hadoop [11], CSP could divide such a task into multiple sub-task and allow them parallelly executed across hundreds of Cloud Computing servers.

B. Adversarial Model

We consider an adversary $\mathcal{A}$ that could corrupt a small set of servers and is *Byzantine*, i.e., can behave arbitrarily. Similar to [17], we assume that our adversary controls at most b servers for any given epoch. Obviously, adversary cloud launch attacks to achieve different goals, which are summarized as follows:

- **Storage-Cheating Model:** In this model, the cheating server launches various attacks towards storage functionality of the cloud. For example, the cheating servers might delete rarely access data files to reduce the storage cost (semi-honest case) or arbitrarily modify the stored data to compromise the data integrity (malicious case). In both of cases, the cloud could simply reply the cloud users' storage query with a random number, which represents a great challenge for cloud users due to lack of physical possession of the potentially large size of outsourced data and constrained computing and storage resources.
- **Computation-Cheating Model:** In this model, assume that a computing service F is comprised of sub-tasks $\{f_1, f_2, \ldots, f_n\}$ and each of sub-task may involve the data located at position vector $\vec{p_i}$, and thus the expected computing result is $R = \{f_1(x_{\vec{p_1}}), f_2(x_{\vec{p_2}}), ..., f_n(x_{\vec{p_n}})\}$. The domain X is defined as $x_{\vec{p_1}}, x_{\vec{p_2}}, ..., x_{\vec{p_n}}$. The cheating cloud could cheat the cloud users by the following two ways: (1) the cheating server computes $F' = f_i(x_{\vec{p_i}})$ for some i where $F' \subset F$ and return the cloud users a random number instead, but claims to have completed all the computations; (2) the cloud server chooses $x \in X' \subset X$ to compute correctly and uses different $\hat{x} \notin X$ which has much lower computational cost or just claims to use the correct data x while the original data is missing.
- **Privacy-Cheating Model** In this model, an untrusted cloud server (or a cloud server hacked by attackers) may compromise the cloud users' privacy by leaking their confidential information to other parties, e.g. their business competitors, which may introduce serious consequences. To provide the data confidentiality, one straightforward approach is to encrypt the data before submitting them to the cloud server. However, such an approach may prevent the regular cloud computing from being processed. In this paper, we consider an *illegally private information selling model*: the cheating/hacked cloud servers illegally sell the cloud users' private information to other parities, e.g. their business competitors. However, we argue that, to sell the private information, the cheating servers should provide corresponding proof to demonstrate that the stored data and computing results because any buyers cannot accept unauthentic data. This model is similar to software selling [18] in our daily life. In this case, software vendor may embed digital signature in its products to allow the users to authenticate them as correct, free of viruses, etc. Such an authentication may be strictly limited to paying customers rather than the illegitimate users to avoid software piracy.

The existing studies mainly focus on the first model while leaving the second and third model unexplored. Therefore, the uncheated and yet private-information-selling-thwarted cloud computing is the major interest of this paper. Similar to secure storage auditing, we assume the existence of a *designated agency* (**DA**) in our model, who is responsible for auditing the security of data storage and computation of cloud services for cloud users. DA is expected to have enough computational and storage capability to perform the auditing operations.

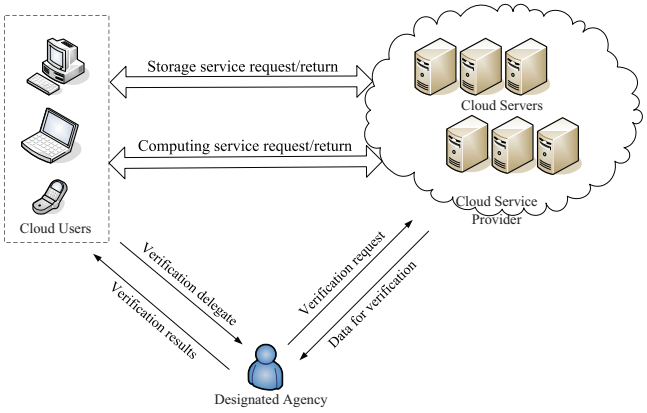

Figure 1. Cloud architecture in our protocol

C. Secure Cloud Computing Model

1) Uncheatable Cloud Computing: To formally define the security model in the cloud computing, we introduce two concepts *Computing Secure Confidence (CSC)* and *Storage Secure Confidence (SSC)* to indicate the trust level of cloud computing and storage security, respectively. Here, CSC is defined as $CSC = |F'|/|F|$ and SSC is formalized as $SSC = |X'|/|X|$. In both cases, cloud computing or

storage is regarded as trustable if CSC (SSC) equals 1 while untrustable if it is 0.

Definition 1: (**Uncheatable cloud computing**) Let `Pr(Cheating Succussfully)` be the probability that a cloud server with both CSC and SSC could successfully cheat without being detected by *sampling based verifiers*. We say the cloud computing is *uncheatable*, if for arbitrary sufficiently small positive number ϵ, there exists a sampling size t such that the following inequation always satisfies

$$\begin{aligned} &\texttt{Pr(Cheating Succussfully)} \\ &= \texttt{Pr(CSC,SSC)} < \epsilon \end{aligned} \tag{1}$$

and

$$t < |X|. \tag{2}$$

2) Privacy-cheating Discouragement: To discourage the CSs from leaking cloud users' data information, we introduce a novel privacy-cheating discouragement model. In this model, it is required that any storage and computation authentication should be under the authorization of the cloud users. In other words, the cloud users could discourage CSP from selling their data or computational results to others by limiting the range of the data/computation verifiers. To achieve this, we introduce the following definition.

Definition 2: (**Privacy preserving**) Let `InfoLeak` denote the event that valid information is leaked by CSP. The cloud computing is *privacy preserving*, if for a sufficiently small positive number ϵ, the following equation holds

$$\texttt{Pr(InfoLeak)} < \epsilon. \tag{3}$$

D. Design goals

The proposed scheme is expected to achieve the following security and performance goals:

1) **Secure data storage**: To ensure the data is stored in cloud securely, the proposed scheme should ensure that the users and DA could audit the stored data effectively.
2) **Secure Cloud Computation**: To enable secure cloud computing, the proposed scheme should ensure that the computation could be audited by users or DA. Considering the fact that the users suffer from computational and transmission constraints, DA's auditing is a promising approach for securing cloud computation.
3) **Privacy Cheating Discouragement**: The proposed scheme should ensure that only designated parties (e.g., CSs or DA) could verify the stored data or computation results, which discourages the CSs from compromising users' privacy, even if the cloud servers are compromised by the attackers.
4) **Efficiency** The computation and transmission overhead of secure cloud computing auditing should be minimized.

IV. Preliminaries and Notation

A. Bilinear pairing

Let $\mathbb{G}_1$ be a cyclic additive group with an operation $(+)$ and $\mathbb{G}_2$ be a cyclic multiplicative group with an operation $(\cdot)$. Both of them have the same prime order q, that is , $|\mathbb{G}_1| = |\mathbb{G}_2| = q$. Let P be a generator of $\mathbb{G}_1$. An efficient admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, with the following properties:

- Bilinear: for all $P \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab}$.
- Non-degenerate: There exists $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1$.
- Computable: there is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

Typically, we can implement the bilinear map using Weil or Tate pairing [19]. Most of the identity based cryptographic schemes [20], [21] are achieved by employing this technique.

B. Designated Verifier Signature

It is a special signature scheme [22]–[27] that the designated verifier can take its private key and the signer's public key to verify the signatures, but it is unable to use this signature to convince any other parties that the message is indeed signed by the original signer, even if the verifier is willing to reveal its private key. This is achieved on that the verifier could take advantage of its private key to generate a fake signature and any other parties are unable to distinguish whether these messages are authentic, i.e. whether they have been signed by the users or not. However, there is few studies on the batch verification of the designated verifier signatures. Another contribution of this paper is designing such a batch verification algorithm for the performance improvement.

C. Merkle hash tree

Merkle hash tree is a well-known authentication structure proposed by Merkle [13], which is constructed as a binary tree where each leaf of the tree is a hash value of authentic data values. It is often used to efficiently and securely to ensure the authencithy and integrity.

D. Discrete logarithm assumption

Discrete logarithms are group-theoretic analogues of ordinary logarithms. In particular, an ordinary logarithm $log_a b$ is a solution of the equation $a^x = b$ over the real or complex numbers. Similarly, if g and h are elements of a finite cyclic group $\mathbb{G}$ then a solution x of the equation $g^x = h$ is called a discrete logarithm to the base g of h in the group $\mathbb{G}$. Discrete logarithm assumption is that given g and h for randomly-chosen $g, h \in \mathbb{G}$, it is difficult to find x such that $g^x = h$.

E. Bilinear Diffie-Hellman assumption (BDH)

Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP (for unknown randomly chosen $a, b, c \in Z_q^*$), compute $\hat{e}(P,P)^{abc}$. For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_2$ must be chosen so that there is no known algorithm for efficiently solving the Discrete logarithm problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$.

V. The Proposed Basic Schemes

In this section, we propose a basic secure cloud computing scheme based on the identity-based cryptography. Our scheme consists of four steps: "*System initialization*", " *Secure cloud storage*", "*Secure cloud computation*", and "*Commitment verification*". Figure 2 gives an overview of data and service flows in the protocol. These algorithms are constructed as follows.

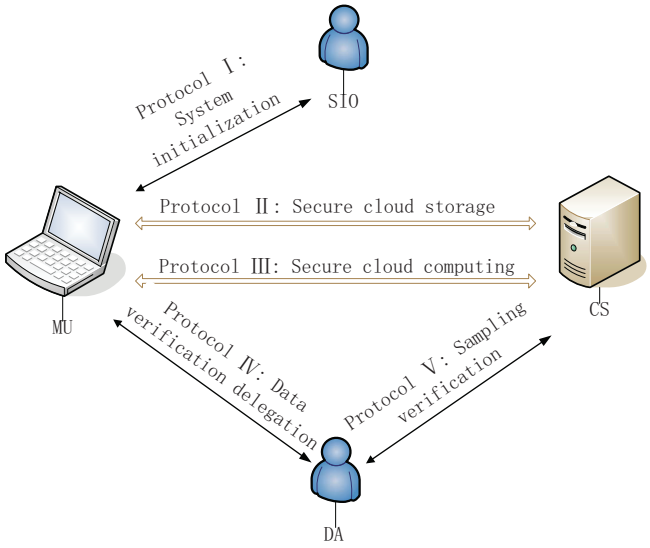

Figure 2. Data and service flow in the proposed protocol

A. System initialization

The *System Initialization Operator* (SIO) [1] runs setup algorithm to generate the system parameters and master secret keys. At the system initialization phase, SIO generates system parameters and public keys. Specifically, SIO selects two cycle groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$; chooses an arbitrary generator $P \in \mathbb{G}_1$. In addition, it chooses three cryptographic hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. After setting the system parameters, SIO picks a random number $s \in \mathbb{Z}_q^*$ as its secret key and sets its public key as $P_{pub} = sP$. Finally, the system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{pub}, H, H_1, H_2)$. The system's secret is s, which is kept safe.

When a cloud user applies for the cloud services, it needs to register to SIO. The user submits its identity ID to SIO and receives system parameters $params$ and a secret key sk_{ID} from SIO in a secure way. Specifically, the procedure is as follows:

$$sk_{ID} = s \cdot Q_{ID} \tag{4}$$

where $Q_{ID} = H_1(ID)$. Note that system initialization and registration step could be performed off-line.

B. Secure Cloud Storage

1) Data Signing: To enable the storage and auditing, the cloud users need to generate a corresponding authentication information for each transmission block $D = \{m_1, m_2, ..., m_n\}$. For each block m_i, the user first generates a identity based signature by (1) selecting a random numbers $r_i \in \mathbb{Z}_q^*$ and computing $U_i = r_i \cdot Q_{ID}$; (2) using the secret key to compute $V_i = (r_i + h_i) \cdot sk_{ID}$ where $h_i = H_2(U_i \| m_i)$. To preserve the privacy, the user then transforms the signature through the idea of designated signature. It computes $\Sigma_i = \hat{e}(V_i, Q_{CS})$ and $\Sigma_i' = \hat{e}(V_i, Q_{DA})$ returns $\{U_i, \Sigma_i, \Sigma_i'\}$ instead of signature (U_i, V_i), where Q_{CS} and Q_{DA} are the identity of the cloud servers and designated agency, respectively. Here, the signature $\sigma_i = (U_i, \Sigma_i, \Sigma_i')$ on each m_i has been generated and is referred as $\Phi = \{\sigma_1, \sigma_2, ..., \sigma_n\}$. Finally, the user sends the data and corresponding signature pairs $\{D, \Phi\}$ to the cloud server and deletes them from local storage.

2) Data Verification: CSs or DA could checks its validity of the stored data by verifying the following equation:

$$\Sigma_i \stackrel{?}{=} \hat{e}(U_i + H_2(U_i \| m_i) Q_{ID_i}, sk_{CS}) \tag{5}$$

If the equation (5) holds, CSs or DA is convinced that users data is securely stored authenticated in the server. Otherwise, the stored data has been compromised.

Note that, privacy cheating is discouraged in the proposed scheme because only CSs or DA could verify the results and the positions of data storage whereas any other parties could not check it since they do not have the cloud servers' private keys. From the definition of our model, privacy preserving is improved.

C. Secure Cloud Computation

The cloud computation auditing protocol is based on Merkle hash tree based commitment scheme, which includes the following three steps: .

[1] In reality, the government or a trusted three party could play the role of SIO. Since the system initialization and registration step could be performed off-line, it will not bring heavy burden to these organization.

1) Computation Request:: The cloud user submits a number of computational service requests $F = \{f_1, f_2, ..., f_n\}$ which could be considered as a set of functions as well as the positions index of data blocks $P = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_n\}$ to cloud server. The functions $f_i \in F$ can be considered as some basic functions such as data sum, data average, data maximum, or other complicated computations based on these functions and on the data which are stored in the positions $\vec{p}_i \in P$.

2) Computation Commitment Generation:: When the cloud server receives the computing requests $\{F, P\}$, it first inputs the data in the position P, computes each function as $y_i = f_i(x_{\vec{p}_i})$ honestly and then builds a Merkle hash tree to make a commitment. The cloud server constructs n leaves with the values $\{v_i = H(y_i \| \vec{p}_i)\}$ where each $\vec{p}_i$ is the position of the data in the cloud. Then the cloud server builds the complete Merkle tree using these leaf values from bottom to top. The value Ω of the internal tree node is defined as follows:

$$\Omega(V) = H(\Omega(V_{leftchild}) \| \Omega(V_{rightchild})) \tag{6}$$

where V denotes an internal tree node and $V_{leftchild}$ and $V_{rightchild}$ are $V's$ two child nodes. Figure 3 shows an example to construct a commitment tree with data computing results. We denote R as the root value of the Merkle hash tree. The cloud server signs the root R and obtains the signature $Sig(R)$. Finally, the cloud server returns the results $Y = \{y_i | \ (1 \leq i \leq n)\}$ as well as $Sig(R)$ to the cloud user or DA for auditing.

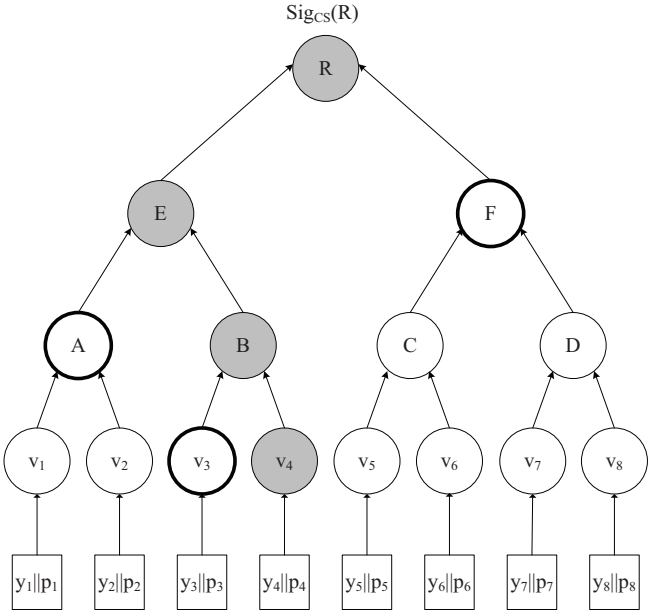

Figure 3. Construct Merkle hash tree based commitment scheme

D. Commitment Verification

In this subsection, the cloud users or DA could perform commitment verification to check the security of the cloud computation. If using DA verification, the cloud user need delegate the verification right to DA, which is proceeded as follows: it sends $\{F, P, Y\}$ as well as a *warrant* include the identity of the delegatee and the expired time to DA for auditing.

Note that, DA not only checks if the data has been appropriately stored at the cloud server but also ensures if the computation process has been well performed. Even though DA is expected to have more powerful computational and transmission capability than the cloud users, it is not cost-effective for DA to re-compute each $f_i(\cdot)$ and to check if the cloud storage and computation is well proceeded. Thus, the probabilistic sampling technique is adopted here to reduce the overall verification cost. The detailed protocol includes the following steps:

1) Audit Challenge Step: When DA starts to audit the data storage and check the computational results, it picks a random subset $S = \{c_1, c_2, ..., c_t\}$ from the domain $[1, n]$. $c_1, c_2, ..., c_t$ are the samples for auditor the verifying the data results. Then DA sends this challenge request to cloud server as well as the *warrant*.

2) Audit Response Step: When the cloud server receives the audit challenge, it first verifies the *warrant* to check whether it is expired. Then for each $c_i \in S$, the cloud server finds in the Merkle hash tree a path ϕ_{c_i} from the leaf to the root. For each node on this path ϕ_{c_i}, cloud server sends the sibling sets to DA. Take Figure 3 for an example. The challenge on $f_4(x_4)$ needs to compute a path ϕ_4 with the vertices $\{v_4, B, E, R\}$. To perform this computation, each node's sibling vertices is required to compute the root R. This means that the cloud server need to return the data x_4, its signature σ_4, as well as the value set $\{v_3, A, F\}$ back to challenger. We show them in black in the Figure 3.

3) Response Verify Step: When DA gets the values from the cloud server, it needs to verify two kinds of correctness. As shown in Algorithm 1, firstly, DA needs to check whether the cloud server uses the data in the request position, not other positions. It verifies

the signature σ_i using the equation

$$\Sigma_i' \stackrel{?}{=} \hat{e}(U_i + H_2(U_i \| m_i) Q_{ID_i}, sk_{DA}). \tag{7}$$

If the signature is correct, it is convinced that the cloud server uses the right position data. Otherwise, the cloud server's cheating behave is detected. Secondly, it checks the correctness of the result $y_{c_i}^*$. If the equation $y_{c_i}^* = f_{c_i}(x_{\vec{p}_{c_i}})$ is incorrect, the cloud server's cheating behave is caught at once. If the $y_{c_i}^*$ is correct, the verifier uses the commit information R to ensure that each y_{c_i} is used at the beginning time of building the Merkle hash tree: the verifier uses the correct $f_i(x_{\vec{p}_i}) \| \vec{p}_i$ as one leaf and its sibling to reconstruct

Algorithm 1: The Probabilistic Sampling Cloud Computation Auditing Protocol

```
Set retValue=valid;
for A sampling index τ do
    Fetch the sampling data and its signature;
    if IsSignatureWrong(τ) then
        retValue=invalid;
    end if
    if IsComputingWrong(τ) then
        retValue=invalid;
    end if
    Fetch the sibling data;
    Reconstruct the root value R(τ);
    if IsRootWrong(R(τ)) then
        retValue=invalid;
    end if
end for
return retValue;
```

the root R^* of the tree by equation (6). Only if $R^* = R$, the verifier can trust that all of the $f_{c_i}(x_{\vec{p}_{c_i}})$ had been computed before the tree was built.

4) Return Step: DA returns true if the cloud server's cheating behave is not caught in all of the challenges. The cloud user is convinced that the cloud server has not cheated with a high probability. Otherwise, returns false. The cloud user drops the computing results and starts a new computing request instead.

VI. Improving Auditing Efficiency with Batch Verification

Considering the fact that the major communication and computation overhead comes from verification of the commitment tree root signature, in this section, we introduce several advanced protocols to further reduce the computational and communication overhead.

The basic idea comes from aggregate signatures. By using identity based aggregate signatures, our scheme can achieve almost constant computation overhead for DA and cloud server. Cloud servers can concurrently handle the multiple verification request not only from one user but also from the different cloud users. Assume that a set of users $\{u_i|1 \leq i \leq k\}$, each of which generates signature $\{\sigma_{ij}|1 \leq j \leq n_i\}$ on message $\{m_{ij}|1 \leq j \leq n_i\}$. The cloud servers do as follows: $\Sigma_A = \prod_{i=1}^{k}\prod_{j=1}^{n_i}\Sigma_{ij}$, $U_A = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(U_{ij} + H(U_{ij}\|m_{ij})Q_{ID_i})$. Then the cloud servers use their secret keys to verify

$$\hat{e}(U_A, sk_{CS}) \stackrel{?}{=} \Sigma_A \tag{8}$$

The correctness is as follows:

$$\begin{aligned}\Sigma_A &= \prod_{i=1}^{k}\prod_{j=1}^{n_i}\hat{e}(V_{ij}, Q_{CS}) = \hat{e}(\sum_{i=1}^{k}\sum_{j=1}^{n_i}V_{ij}, Q_{CS}) \\ &= \hat{e}(\sum_{i=1}^{k}\sum_{j=1}^{n_i}(r_{ij} + H_2(U_{ij}\|m_{ij}))sk_{ID_i}, Q_{CS}) \\ &= \hat{e}(\sum_{i=1}^{k}\sum_{j=1}^{n_i}(U_{ij} + H_2(U_{ij}\|m_{ij})Q_{ID_i}), sk_{CS}) \\ &= \hat{e}(U_A, sk_{CS}). \end{aligned} \tag{9}$$

The computational complexity of our scheme is analyzed as follows. Note that the signature combination can be performed incrementally and the computational cost are almost measured by the expensive pairing operations. It is obvious that given ξ unauthenticated signatures, in the batch verification, the major computational cost is bounded by 2 pairings while it costs 2ξ pairings by individual verification, which is a significant improvement on computational efficiency.

If the verification succeeds for all the sample set S, the verifier is convinced that the cloud has not cheated as a high probability.

VII. Security Analysis and Performance Evaluation

A. Uncheatability analysis

To analyze our uncheatable cloud computing, in this section, we evaluate the sampling performance of the proposed scheme in terms of the number of sampling blocks that needs to be retrieved.

We first define the `FCS` as the event that the cloud server could successfully cheat through guessing the results of functions. Let R be the range of function f. Thus, the probability that a cloud server can randomly guess the correct result of $f(x)$ is $\frac{1}{R}$. For most f of interest, $|R|$ would be quite large. Besides, we can adapt the concept of Computing Secure Confidence in the previous. Therefore, the cloud server could successfully cheat a t time-sampling scheme with the probability as follows.

$$\texttt{Pr[FCS]} = \left(CSC + (1 - CSC)\frac{1}{R}\right)^t. \tag{10}$$

Since the $R > 1$, `Pr[FCS]` goes to 0 when the sampling size t is large enough. Thus, we have

$$\texttt{Pr[FCS]} < \epsilon. \tag{11}$$

We also define `PCS` is the event that the cloud server could successfully cheat through using the data on the different positions. We also use the concept of Storage Secure Confidence. In t time-sampling scheme, the probability the

cloud can successfully using the data on the invalid positions is as follows.

$$\texttt{Pr[PCS]} = \Big(SSC + (1-SSC)\texttt{Pr[SigForge]}\Big)^t \quad (12)$$

where Pr[SigForge] is the probability that the cloud server could forge the digital signatures, which is very small in our model. Pr[PCS] will go to 0 when the sampling size is large enough. We have

$$\texttt{Pr[PCS]} < \epsilon. \quad (13)$$

From our definition 1, we can define the cloud server cheats successfully if the event FCS or event PCS takes place. Here, we consider the assumption that FCS and PCS are independent. Thus, the probability that the cloud server can cheat successfully is computed as follows.

$$\begin{aligned}\texttt{Pr[Cheating Successful]} &= \texttt{Pr}[\texttt{FCS} \cup \texttt{PCS}] \\ &= \texttt{Pr[FCS]} + \texttt{Pr[PCS]} \quad (14)\end{aligned}$$

From the inequation (11) and (13), (14) can be reduced to

$$\texttt{Pr[Cheating Successful]} < \epsilon. \quad (15)$$

To get more accurate results, we evaluate the sampling size

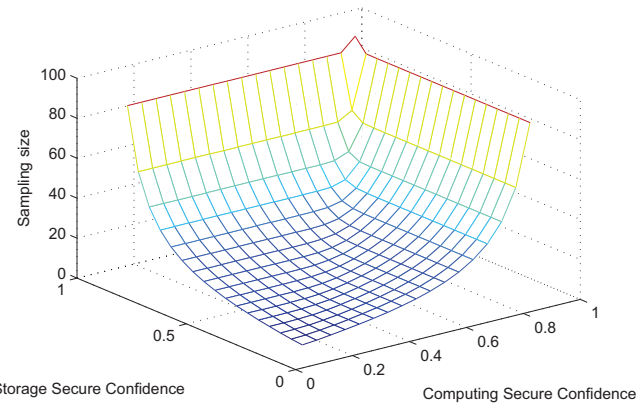

Figure 4. Required sample size achieving uncheatable cloud computing given $\epsilon = 0.0001$)

by numerical method. Given $\epsilon = 0.0001$, Figure 4 shows that how large t should be for the different CSC and SSC. From the figure, we can find that the probability can be below ϵ when t is larger than a certain number. When we consider such a situation that the cloud server has computing with half CSC and half SSC of the task, the range of the domain is $R = 2$, we need at least 33 samples to ensure the probability of successful cheating to be below $\epsilon = 0.0001$. When R is large enough, (i.e., $R \to \infty$, it is almost impossible to make a correct guess on $f(x)$ without computing it), we only need 15 samples. Therefore, the cloud server can successfully cheat with a neglect probability in our cloud computing.

Theorem 1: From the discussion above, our cloud computing is *uncheatable*.

B. Privacy analysis

To analyze our privacy level, we compare the probability with cryptographic assumptions in the section IV. In our privacy enhanced protocol, our designated verifier signature scheme could be proved secure based on the assumption BDH in the random oracle model. Therefore, the cloud servers can confirm the users' identity since any other users can not forge these data signature pairs. Besides, even the cloud servers are compromised by attackers, it can not convince others the users' data since the cloud server can generate the signature of themselves in our model.

The only way for the cloud servers is to forge the original signature of the cloud users, which leads to solve the the hard problem of the assumption. We assume that the probability of solve a hard problem is ϵ for any algorithms existing. From the discussion above, we have

$$\texttt{Pr[InfoLeak]} \approx \texttt{Pr[SigForge]} < \epsilon. \quad (16)$$

From the definition 2, our cloud computing is *privacy preserving* with a neglect probability. We omit the cryptographic proving procedure for space limited.

Theorem 2: From the discussion above, our cloud computing is *privacy preserving*.

C. Optimal sample set size to minimize total cost

To generally estimate the total cost for our sampling algorithm in the protocol, we assume that it can be divided into three kinds of cost. Let C_{trans} and C_{comp} be the transmission cost and the computational cost for each sampling message-signature pairs, respectively. We also define the cost C_{cheat} that is caused by the undetected cheating attacks. The total cost C_{total} for a sample set of t is as follows:

$$C_{total} = a_1 \cdot t \cdot C_{trans} + a_2 \cdot C_{comp} + a_3 \cdot C_{cheat} \cdot q^t \quad (17)$$

where q is the probability of cheating successful and a_1, a_2, and a_3 are coefficients for these costs, respectively. The following theorem gives the optimal sample set t to achieve a minimal total cost.

Theorem 3: Given the transmission cost C_{trans}, computational cost C_{comp} and successfully cheating cost C_{cheat}, and the probability of cheating successful q, the optimal sample set t for achieving the minimal cost is

$$t = \left\lceil \ln(-\frac{a_1 \cdot C_{trans}}{a_3 \cdot C_{cheat} \cdot \ln q}) / \ln q \right\rceil \quad (18)$$

Proof: Since $C_{total} = a_1 \cdot t \cdot C_{trans} + a_2 \cdot C_{comp} + a_3 \cdot C_{cheat} \cdot q^t$, to minimize the total cost C_{total}, we have

$$\frac{dC}{dt} = a_1 \cdot C_{trans} + a_3 \cdot C_{cheat} q^t \ln q \quad (19)$$

It is easy to check that the derivative is 0 when $t = \ln(-\frac{a_1 \cdot C_{trans}}{a_3 \cdot C_{cheat} \cdot \ln q}) / \ln q$. Note that, t must be an integer in practice.

In order to optimize the cost in practice, we need the detail cost information such as C_{trans}, C_{comp}, C_{cheat}, a_1, a_2, and a_3. We evaluate them through a history learning process.

D. Performance analysis

To analyze the performance of our scheme, we experiment in the environment consisted of Intel Core 2 Duo E6550 with 2 GB RAM to evaluate the delay of cryptographic operations based on cryptographic library MIRACL [28], which is shown in Table I. We also implement our algorithm in Matlab 2009a, and substitute various values for t and R to gain an insight into the behavior of our protocol.

	Descriptions	Execution Time
T_{pmul}	Time for one point multiplication	0.86 ms
T_{pair}	Time for one pairing operation	4.14 ms

Table I
CRYPTOGRAPHIC OPERATION'S EXECUTION TIME.

Compared with other data auditing schemes in [4], [5], we mainly consider the verification operation cost in protocol since it is most expensive cost operation at both cloud server and verifier side. The computation costs are measured by pairing and point multiplication operation time. Figure 5 plots the computation cost of verification when the number of cloud user ranges from 1 to 50. Our protocol is much more efficient than the previous one since pairing times are constant in ours while linear in theirs.

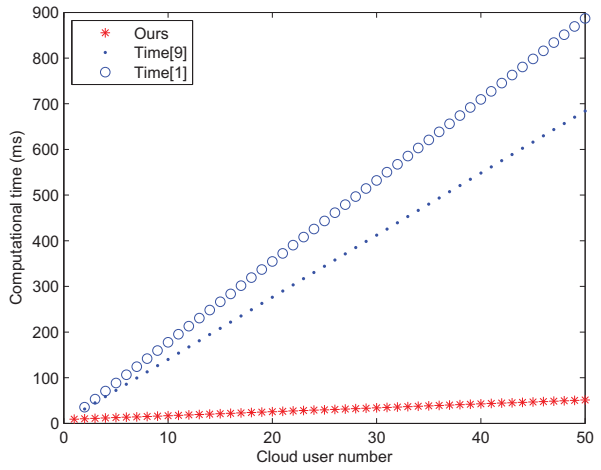

Figure 5. Comparison of the computational cost among relative schemes

To further demonstrate the suitability of the proposed scheme, we analyze the computational cost for different signatures are summarized in Table II , which need to handle the batch size of η.

scheme	individual Verify	Batch Verify
RSA	$\eta \cdot T_{RSA}$	NA
ECDSA	$\eta \cdot T_{ECDSA}$	NA
BGLS [29]	$2\eta \cdot T_{pair}$	$(\eta + 1) \cdot T_{pair}$
Our scheme	$2\eta \cdot T_{pair}$	$2 \cdot T_{pair}$

Table II
COMPARISON OF VARIOUS SIGNATURE SCHEME.

VIII. CONCLUSION

In this paper, we propose a privacy preserving designated verification protocol for data computing security in the cloud. To the best of our knowledge, it is the first paper that jointly considers both of the data computing and data storage security. We define the concept of uncheatable cloud computing and model the attacks in this concept. Besides, we introduce a novel data privacy cheating discouragement concept. To improve the efficiency, the designated verifiers can concurrently handle multiple sessions from different users' verifying requests. By using extensive security and performance analysis, it is showed that our protocol is secure and efficient for achieving a secure cloud computing.

ACKNOWLEDGE

This paper is supported by National Natural Science Foundation of China grants. 60972034, 60970110, 60773086, and National 973 Program No. 2007CB311201. We would also like to thank anonymous people who helped us in writing this paper.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A berkeley view of cloud computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[2] Marinos and A. Gerard Briscoe, "Community Cloud Computing," in *Proceedings of Cloud Computing: First International Conference (CloudCom 2009)*, Beijing, China, December 1-4, 2009.

[3] A. Haeberlen, "A Case for the Accountable Cloud," in *3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware*, Big Sky Resort, Big Sky, MT, October 10-11, 2009.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in *29th IEEE Conference on Computer Communications (INFOCOM'10)*, San Diego, California, USA, March 14-19, 2010.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *14th European Symposium on Research in Computer Security (ESORICS'09)*, Saint Malo, France, September 21-23, 2009.

[6] G. Karame, M. Strasser, and S. Capkun, "Secure Remote Execution of Sequential Computations," in *11th International Conference on Information and Communications Security (ICICS'09)*, Beijing, China, December 14-17, 2009.

[7] W. Du, J. Jia, M. Mangal, and M. Murugesan, "Uncheatable Grid Computing," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Hachioji, Tokyo, Japan, March 24-26, 2004.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security (CCS'07)*, Alexandria, Virginia, USA, October 28-31, 2007.

[9] G. Ateniese, R. Di Pietro, L. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, Istanbul, Turkey, September 22-26, 2008.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *17th IEEE International Workshop on Quality of Service (IWQoS'09)*, Charleston, South Carolina, USA, July 13-15, 2009.

[11] A. Juels and B. Kaliski Jr, "PORs: Proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security (CCS'07)*, Alexandria, Virginia, USA, October 28-31, 2007.

[12] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Asiacrypt, The 14th Annual International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, December 7-11, 2008.

[13] R. Merkle, "Protocols for public key cryptosystems," in *IEEE Symposium on Security and Privacy*, Oakland, California, USA, April, 1980.

[14] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[15] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM conference on Computer and communications security (CCS'09)*, Chicago, Illinois, USA, November 9-13, 2009.

[16] P. Golle and I. Mironov, "Uncheatable distributed computations," in *The Cryptographers' Track at RSA Conference 2001*, San Francisco, CA, USA, April 8-12, 2001.

[17] K. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," in *Proceedings of the 16th ACM conference on Computer and communications security (CCS'09)*, Chicago, Illinois, USA, November 9-13, 2009.

[18] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Advances in Cryptology - EUROCRYPT96*, 1996.

[19] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[20] J. Cha and J. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, Miami, Florida, USA, January 6-8, 2003.

[21] H. Zhu, X. Lin, R. Lu, X. Shen, D. Xing, and Z. Cao, "An Opportunistic Batch Bundle Authentication Scheme for Energy Constrained DTNs," in *29th IEEE Conference on Computer Communications (INFOCOM'10)*, San Diego, California, USA, March 14-19, 2010.

[22] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk, "Universal designated-verifier signatures," in *Advances in Cryptology-ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security*, Taipei, Taiwan, 30 November - 4 December 2003.

[23] F. Zhang, W. Susilo, Y. Mu, and X. Chen, "Identity-based universal designated verifier signatures," *Lecture notes in computer science*, vol. 3823, pp. 825–834, 2005.

[24] J. Zhang and J. Mao, "A novel ID-based designated verifier signature scheme," *Information sciences*, vol. 178, no. 3, pp. 766–773, 2008.

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

[26] F. Cao and Z. Cao, "An identity based universal designated verifier signature scheme secure in the standard model," *Journal of Systems & Software*, vol. 82, no. 4, pp. 643–649, 2009.

[27] A. Bender, J. Katz, and R. Morselli, "Ring signatures: Stronger definitions, and constructions without random oracles," *Journal of Cryptology*, vol. 22, no. 1, pp. 114–138, 2009.

[28] M. Scott, "Implementing cryptographic pairings," in *The first International Conference on Pairing-based Cryptography (Pairing'07)*, Tokyo, Japan, July 2-4, 2007.

[29] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, May 4-8, 2003.