Cooperative Query Answer Authentication Scheme over Anonymous Sensing Data

Liangmin Wang, Member, IEEE Qingqing Xie, and Hong Zhong

Abstract—In cloud service over crowd-sensing data, the data owner (DO) publishes the sensing data through the cloud server, so that the user can obtain the information of interest on demand. But the cloud service providers (CSP) are often untrustworthy. The privacy and security concerns emerge over the authenticity of the query answer and the leakage of the DO identity. To solve these issues, many researchers study the query answer authentication scheme for cloud service system. The traditional technique is providing DO’s signature for the published data. But the signature would always reveal DO’s identity. To deal with this disadvantage, this paper proposes a cooperative query answer authentication scheme, based on the ring signature, the Merkle hash tree (MHT) and the non-repudiable service protocol. Through the cooperation among the entities in cloud service system, the proposed scheme could not only verify the query answer but also protect the DO’s identity. First, it picks up the internal nodes of MHT to sign, as well as the root node. Thus, the verification computation complexity could be significantly reduced from \( O(\log_2 N) \) to \( O(\log_2 N^{0.5}) \) in the best case. Then it improves an existing ring signature to sign the selected nodes. Furthermore, the proposed scheme employs the non-repudiation protocol during the transmission of query answer and verification object (VO) to protect trading behavior between the CSP and users. The security and performance analysis prove the security and feasibility of the proposed scheme. Extensive experimental results demonstrate its superiority of verification efficiency and communication overhead.

Index Terms—Cooperation, query answer authentication, identity privacy, non-repudiation protocol, crowd-sensing data.

1 INTRODUCTION

With the advances of wireless sensor networks and Internet of things, crowd-sensing big data is collected by scattering sensors over a vast field. As time goes by, the fast-growing data volumes make it hard for the sensors to store due to their weak storage and computing resources. It becomes a problem that how to store these crowd-sensing data economically, as well as perform queries on it efficiently. Considering the flexible, on-demand and low-cost usage of cloud storage resources [1], [2], [3], the enterprises and individuals, i.e., data owners (DO), outsource their data to the cloud server. Thus, the users can get the information of interest by asking the cloud service provider (CSP) for searching the outsourced data [3], [4], [5], [6]. Such a cloud service system based on crowd-sensing data comes into being, as shown in Fig. 1.

![Fig. 1. The cloud service environment over crowd-sensing data.](image)

Observing the service model in Fig. 1, there are three entities in the system: DO, user, and CSP. The crowd-sensing data is provided by many data owners. More users and CSP would join in the system for utilizing these data. Due to the collaborative operation among DO, user and CSP, multiple security and privacy problems have to be taken into consideration. The security and privacy requirements include:

- In demand for privacy preservation, the DO tends to outsource the data anonymously. Thus in some specific application scenarios, the DO is also called as the anonymous data provider.
- The CSP provides the paid service for users. Hence in pursuit of commercial profits, the CSP requires that users can not deny having been served by the CSP if the CSP has sent the proper query answers to the users.
- Since the CSP is often untrustworthy, the users desire urgently for an efficient query answer authentication scheme.

In brief, there are three aspects of requirements: the anonymity of DO identity, the efficient verification for the users’ query answers and the non-repudiation of query transaction for the CSP.

At present, there have been some researches related to the query answer authentication over outsourced data [7], [8], [9], [10], [11], [12], [13], [14], [15]. Nevertheless, the existing research works can not satisfy the aforementioned requirements simultaneously. Moreover, the existing research scenarios are far away from the above big-data environment based on crowd-sensing: multiple DOs with anonymity requirement, a very complicated user base who may be dishonest, and an untrustworthy CSP. Furthermore, the anonymity requirement of DO conflicts with the trusti-
ness authentication for data sources. Hence, the challenge is how to satisfy the aforementioned security and privacy requirements simultaneously, in such a complex cloud service environment.

To overcome the challenges above, we propose a novel cooperative query answer authentication scheme. The DO, CSP and users collaboratively store, search and verify the data. In the proposed scheme, we make the following contributions:

1) We select some internal nodes and root node of MHT as key nodes (KN). The KNs account for \(1/2^{\log_2 \sqrt{N} - 0.5} + 1/n\) of the total data records, which greatly reduces the computational and storage overhead, thereby improves the verification efficiency.

2) We adapt the linkable ring signature scheme to sign the KNs. It’s worth noting that this step paves the perfect way for anonymous authentication.

3) We construct a non-repudiation protocol based on verification object (VO) to protect the secure interaction between the CSP and the user.

Here assume users issue selection queries of the form as

\[
\text{SELECT * FROM stream WHERE } l_i < A_i < u_i
\]

where \((l_i, u_i)\) are the selection ranges over attribute \(A_i\).

The rest of this paper is structured as follows. In Section 2, we discuss the related works. Section 3 presents some preliminaries. Section 4 describes our system model and the scheme framework. Section 5 presents our proposed cooperative query answer authentication scheme in detail. In Section 6, we analyze the security and performance of the proposed scheme in theory. In Section 7, we evaluate the performance of our scheme by experiment. Finally, Section 8 concludes this paper.

2 RELATED WORKS

This paper presents a suite of cooperative query answer authentication scheme over anonymous sensing data. The related works include anonymous data publishing, query answer authentication, and non-repudiation service.

2.1 Anonymous Data Publishing

The anonymous data publishing techniques are aiming to protect individual identification. The research on anonymous data publishing originated from the \(K\)-anonymity model proposed by Samarati P. and Sweeney L. in 1998 [16], later amended and supplemented in 2002 [17], [18]. Subsequently, some new models appeared, such as \(l\)-diversity model [19], \(t\)-closeness model [20], uncertain data models with anonymity [21], [22], etc. The existing anonymous methods include generalization [16], [18], suppression [16], [18], clustering [23], microaggregation [24], anatomy [25], permutation [26], and so on.

In the above data publishing techniques, there is a common request that the data publisher must be trusted. As we all know, it cannot be guaranteed in cloud environment, since the CSP plays the role of data publisher. So we introduce ring signature scheme. It is good at cutting off the correlation between the data and the data signer (i.e., DO). Moreover, the trustiness of data source is well guaranteed. The seminal construction of ring signature scheme was proposed by Rivest, Shamir and Tauman in 2001 [27]. Subsequently, there exist many constructions and variants [28], [29], [30], [31], [32]. In ring signature scheme, DO could sign the message anonymously, and user can check the signature trustiness without knowing the signer [33], [34], [35], [36]. However, if applying the ring signature scheme directly, DO needs to sign all the data records one by one. It is prohibitively impractical when facing the crowd-sensing data in cloud environment.

2.2 Query Answer Authentication

The query answer authentication schemes are used to serve the query user to verify the downloaded data. The straightforward solution for verifying a set of \(N\) values is to generate \(N\) digital signatures. An improvement on this solution is based on the MHT [10]. Its basic idea is exactly to replace signatures with the much cheaper hashes. The MHT is a binary tree where each leaf contains the hash of a data value, and each internal node contains the hash of the concatenation of its two children. Verification of data values is based on the fact that the hash value of the tree root is authentically published using a digital signature \(s\). To prove the authenticity of any value, DO provides the user with the data value itself and the hash values of the siblings of the nodes that lie in the path that connects the root of the tree with this value. The user, by iteratively computing and concatenating the appropriate hashes, can recompute the hash of the root and verify its correctness using \(s\). Correctness is guaranteed by the security of the publicly key digital signature for the hash value of the root node, as well as the collision resistance of the hash functions. By hashing a given node, it becomes computationally infeasible for an adversary to modify the node in a way that ultimately preserves the hash value of the root. MHT is mainly used in query authentication over outsourced data [37], [38], [39]. F. Li, K. Yi, M. Hadjieletheriou and G. Kolios proposed an authentication of sliding window queries on streams on the basis of MHT later on [8], [12]. D. Wu, B. Choi, J. Xu and C. S. Jensen proposed an authentication of moving top-\(k\) spatial keyword queries by extending the MHT [39].

Nevertheless, when facing big-data environment, there exist several disadvantages in MHT. First, the constructed MHT is always extremely high. When executing verification, the user cannot get the hash value of the root until completing computing many hash values of internal nodes from the bottom up. Moreover, if the corresponding signature of the root is tampered, these hashing calculating will be meaningless. Thus it has low authentication efficiency and weak security. Second, multiple DOs are not well supported.

2.3 Non-Repudiation Service

Non-repudiation services are aiming to collect, maintain, make available, and validate irrefutable evidence regarding a transaction. They protect the parties involved in a transaction against the other party denying that a particular event or action took place, and collect irrefutable evidence to support the resolution of any such disagreement. The basic non-repudiation services have two properties, i.e.,
non-repudiation of origin and non-repudiation of receipt. They are composed of four distinct phases: evidence generation, evidence transfer and storage, evidence verification, and dispute resolution. The fair non-repudiation protocol, proposed by Zhou J. and Gollmann D. in 1996, was the most widely studied [40], [41]. Subsequently, a variety of formal verification and multiple protocol variants appear [42], [43], [44], [45], [46], [47], [48], [49]. So far, the non-repudiation protocols have used various forms of encryption technique, such as encryption technique based on hash function, block encryption technique based on the Chinese Remainder Theorem, multiple blocks encryption technique, encryption technique and secret sharing technique, etc.

There exist several models of non-repudiation protocols. According to whether a trusted authority (TA) is included, they can be divided into two types: non-repudiation protocol with TA and non-repudiation protocol with non-TA. Non-repudiation protocol with non-TA exchanges information step by step [50], [51]. It can achieve just probabilistic security. Its high requirement for computing power of the communicating parties, however, makes it a bottleneck to apply into cloud service model successfully. Non-repudiation protocol with TA can protect the security of non-repudiation services well. Considering the frequent transactions between CSP and user in cloud, we apply a non-repudiation protocol with offline TA [49] to ensure the interests of both sides, where TA does not intervene in the protocol while both the CSP and the user have no incorrect behavior, and no network error occurs.

3 PRELIMINARIES

In this section, we provide the preliminary background of the collision-resistant hash functions and the public-key digital signature schemes.

Collision-resistant hash functions. A hash function \( H \) is an efficiently computable function that takes a variable-length input \( x \) to a fixed-length output \( y = H(x) \). Collision resistance states that it is computationally infeasible to find two different inputs, \( x_1 \neq x_2 \), such that \( H(x_1) = H(x_2) \). Collision-resistant hash functions can be built provably based on various cryptographic assumptions, such as hardness of discrete logarithms. Reference [11] has tested the time cost for hashing. One hashing operation takes approximately 2 to 3 us, which is around 50 times faster than modular multiplication.

Public-key digital signature schemes. A public-key digital signature scheme is used to authenticate the integrity and ownership of the signed message. Firstly, the signer generates a pair of secret and public keys, denoted as \((SK, PK)\), where the secret key \( SK \) is kept secret, and the public key \( PK \) associated with his identity is published. Subsequently, for any message \( m \) that he sends, he computes a signature \( s_m \) by: \( s_m = S(SK, m) \). The recipient of \( s_m \) and \( m \) can verify the validity of \( s_m \) via \( V(PK, m, s_m) \). If \( s_m = V(PK, m, s_m) \), \( s_m \) is a valid signature and the message \( m \) has not been changed.

4 SYSTEM MODEL AND FRAMEWORK

4.1 System Model

In our system, there are four entities: CSP, DO, user, and TA, as shown in Fig. 2. The TA is introduced to achieve non-repudiation service between the CSP and user.

![Fig. 2. The cloud system model over crowd-sensing data.](image)

We assume that the CSP is untrustworthy. The DO loses the direct control over the outsourced data. The outsourced data may be tampered, lost, and forged by the CSP, or the attackers. Hence, users suspect whether the query answers provided by the CSP are authentic, complete and trusted.

There are multiple DOs in practical cloud service systems. DO, as the only prover, can provide convincing evidence, such as signature, to verify the stored data in cloud servers. But, the signature is always associated with DO’s identity. DO worries that the signature will expose identity privacy to a significant risk. Assume there is a ring signature group consisting of such \( n \) DOs.

User is an important and complex role in cloud service systems. He should pay to CSP for the service. But there exist some users who deny having been served by CSP with the intention of avoiding payment. Here, we do not take the data privacy and query privacy into consideration, which is orthogonal to our paper, and can be guaranteed by the existing searchable encryption [52], [53] and order-preserving encryption schemes [54], [55].

TA is assumed to be trustworthy, who are supervised by the government offices. It can ensure the fair transaction between CSP and user, by providing non-repudiable evidence for possible happening denial behavior.

We propose a cooperative query answer authentication scheme, to satisfy the aforementioned three aspects of requirements in Section 1, including the identity privacy preservation for the DO, the efficient verification of query answers for the users, the non-repudiation service between the CSP and users. We will present the scheme framework next. To make our discussion clear, we first explain some notations, as shown in Table 1.

4.2 Framework

Our scheme is a tuple of five algorithms:

- **Setup(1^\lambda) \rightarrow PA**: This algorithm takes as input the security parameter \( 1^\lambda \), and outputs the public system parameters \( PA \).
- **KeyGen(PA) \rightarrow (sk_i, pk_i)**: This algorithm outputs a private/public key pair \((sk_i, pk_i)\) for \( DO_i \).
- **Sign(Data) \rightarrow (sk_i, pk_i)**: This algorithm takes as input the outsourced data set \( A_{data} \), a certain private key \( sk_{i, n} \), the number \( n \) of DO members in the signature group, and the corresponding set \( Y \) of the \( n \) DOs’ public keys,
TABLE 1
Notations for system construction.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_{data} = {a_j}_{j=1,2,...,N})</td>
<td>a data set, where (N) is the number of data</td>
</tr>
<tr>
<td>(Y = {pk_1, pk_2, ..., pk_n})</td>
<td>a set of (n) DOs’ public keys</td>
</tr>
<tr>
<td>(rs)</td>
<td>a range selection query</td>
</tr>
<tr>
<td>(R)</td>
<td>a query answer</td>
</tr>
<tr>
<td>(VO)</td>
<td>a verification object</td>
</tr>
<tr>
<td>(H_1 : {0,1}^* \rightarrow G)</td>
<td>a hash function, where (G) is a group of prime order (p)</td>
</tr>
<tr>
<td>(H_2 : {0,1}^* \rightarrow Z_p)</td>
<td>a collision-resistant hash function</td>
</tr>
<tr>
<td>(X \rightarrow W)</td>
<td>transmission from entity (X) to entity (W)</td>
</tr>
<tr>
<td>(E_K(\cdot))</td>
<td>a symmetric-key encryption function under key (K)</td>
</tr>
<tr>
<td>(D_K(\cdot))</td>
<td>a symmetric-key decryption function under key (K)</td>
</tr>
<tr>
<td>(E_X(\cdot))</td>
<td>a public-key encryption function under entity (X)’s public key</td>
</tr>
<tr>
<td>(D_X(\cdot))</td>
<td>a public-key decryption function under entity (X)’s private key</td>
</tr>
<tr>
<td>(Sign_x(\cdot))</td>
<td>the signature function of entity (X)</td>
</tr>
<tr>
<td>(I)</td>
<td>a flag indicating the purpose of a message</td>
</tr>
<tr>
<td>(EOO = Sign_{CSP}(f_{EOO}, User, TA, L, H_2(E_K(R))))</td>
<td>a label that in conjunction with ((CSP, User)) uniquely identifies a protocol run</td>
</tr>
<tr>
<td>(EOR = Sign_{User}(f_{EOR}, CSP, TA, L, H_2(E_K(R))))</td>
<td>the evidence of origin for the ciphered answer (R)</td>
</tr>
<tr>
<td>(Sub = Sign_{CSP}(f_{Sub}, User, L, E_{TA}(K</td>
<td></td>
</tr>
<tr>
<td>(EEOO_{K</td>
<td></td>
</tr>
<tr>
<td>(EOR_{K</td>
<td></td>
</tr>
<tr>
<td>(Rec_X = Sign_x(f_{Rec_X}(Y, L)))</td>
<td>the evidence of receipt for (K) and (VO)</td>
</tr>
<tr>
<td>(Con_{VO} = Sign_{CSP}(f_{Con_{VO}}, User, L, K, VO))</td>
<td>the recovery request, where if (X) is CSP then (Y) is (User), else (Y) is CSP</td>
</tr>
<tr>
<td>(Abort = Sign_{CSP}(f_{Abort}, User, L))</td>
<td>the confirmation evidence for (K) and (VO)</td>
</tr>
<tr>
<td>(Con_A = Sign_{TA}(f_{Con_{A}}, CSP, User, L))</td>
<td>the abort confirmation evidence</td>
</tr>
</tbody>
</table>

then outputs a signature set \(\{s\}\), where \(sk_i\)’s public key \(pk_i\) belongs to \(Y\), i.e. \(pk_i \in Y\).

- \(ServiceTransaction(rs) \rightarrow (R, VO, E)\): According to a query request \(rs\), this protocol computes and transmits the corresponding query answer \(R\) and the verification object \(VO\). During the transaction, the evidence \(E\) of non-repudiation service are generated for solving some transaction disputes.

- \(ServiceVerify(n, Y, R, VO) \rightarrow \text{accept/reject}\): This algorithm takes as input a public keys set \(Y\), a query answer \(R\), and the corresponding verification object \(VO\), then verifies the query answer \(R\).

5 Our Cooperative Query Answer Authentication Scheme

In this section, we will present the design details of our scheme. Assume the data owner \(DO_i\) will store a set \(A_{data} = \{a_j\}_{j=1,2,...,N}\) of data to the cloud server. We will take it as an example to present our scheme design.

5.1 Setup(1^k) \rightarrow PA

Let \(G\) be a group of prime order \(p\) where the underlying discrete logarithm problem is intractable. Let \(H_1 : \{0,1\}^* \rightarrow G\) and \(H_2 : \{0,1\}^* \rightarrow Z_p\) be two hash functions. Let \(g = H_1(\text{"GENERATOR"})\) and \(h = H_1(\text{"GENERATOR"})\). The public system parameters are \(PA = (G, g, h, p, H_1, H_2, \text{"GENERATOR"}, g^h, \text{"GENERATOR"})\).

5.2 KeyGenerate(PA) \rightarrow (sk_i, pk_i)

\(DO_i\) randomly chooses \(x_i, y_i \in \mathbb{Z}_p\) as his secret key \(sk_i = (x_i, y_i)\), then computes his public key \(pk_i = g^{x_i}h^{y_i}\).

5.3 \(Sign(A_{data}, sk_i, n, Y) \rightarrow \{s\}\)

\(DO_i\) executes this algorithm to sign the data set \(A_{data}\) before uploading it. The procedure includes two steps:

Step 1: Construct a MHT, and choose some key nodes (KN).

Definition 5.1. A key node is defined as the node whose hash value will be signed.

Assume that \(A_{data}\) has been sorted on query attribute. Let us construct a MHT whose each leaf contains the hash of a data value, and each internal node contains the hash of the concatenation of its two children [10]. Assume \(N=16\), as shown in Fig. 3, computing \(N_1 = H(a_1)\), \(N_2 = H(a_2)\), \(N_3 = H(a_3)\), \(N_4 = H(a_4)\), \ldots, \(N_{1,2} = H(N_1||N_2)\), \(N_{3,4} = H(N_3||N_4)\), \ldots, in a similar way, \(N_{1,4} = H(N_{1,2}||N_{3,4})\), \(N_{1,8} = H(N_{1,4}||N_{5,8})\). At last the hash of root node is \(N_{1,16} = H(N_{1,8}||N_{9,16})\).

Then we will pick the root node and the internal nodes on the middlemost level \(l = \lceil \log_2 \sqrt{N} + 0.5 \rfloor\) (From down to up, the levels are defined as level 0, level 1, \ldots, hence the root-node level as level \log_2 N + 1) as KNS, marked by solid black nodes in Fig. 3. Then the DO will sign the hash values of these KNS, as described in Step 2.
**Theorem 5.1.** When selecting each node on the middlemost level \( \lceil \log_2 \sqrt{N + 0.5} \rceil \) and the root node as KNs, the total number of signed nodes is \( \frac{N}{2} + 1 \).

**Proof.** Let's prove by mathematical induction. Assume there are \( \frac{N}{2} \) nodes on level \( i \) (\( 1 \leq i \leq \log_2 N + 1 \)).

- **Base case:** on level 1, i.e., the leaf level, there are \( \frac{N}{2} \) nodes. The assumption holds for level 1. Suppose the assumption also holds for level \( i \). Since the number of nodes on level \( i \) is the double of that on level \( i+1 \), there are \( \frac{N}{2} \) nodes on level \( i+1 \). The above induction shows that the assumption holds. In consequence, there are \( \frac{N}{2} + 1 \) nodes on level \( \lceil \log_2 \sqrt{N + 0.5} \rceil \). We can claim that total number of signed nodes is \( \frac{N}{2} + 1 \), together with the root node.

- **Step 2:** Sign the hash values of the KNs one by one.

We adapt the linkable ring signature scheme to sign the hash values of these KNs one by one. Algorithm 1 shows the signing procedure, where there are \( n \) DOs to participate in signing hash value collaboratively, instead of just DO, hence hiding the real data provider DO. Moreover, the user still can verify the signature without knowing the true data provider DO. The detailed verification procedure will be shown in Section 5.5.

**Algorithm 1 AnonySign**

**Input:**
- \( hashvalue \): the hash value \( hashvalue \) of one KN;
- \( sk_i = (x_i, y_i) \): the private key of the DO, \( 1 \leq i \leq n \);
- \( Y = \{pk_1, pk_2, \ldots, pk_n\} \): the public key list of \( n \) DOs in the ring;

**Output:**
- \( s \): the signature of \( hashvalue \);
- \( r_1, r_2, c_1, c_2, \ldots, c_n \in RZ_p \) randomly;
- \( c_i \): the sub-key for each node.

1. Compute \( K = g^{rs} h^s \prod_{j=1,j\neq i}^n c_j^{c_j} \)
2. Find \( c_i \) satisfying \( c_i + \ldots + c_n \mod p = H_2 (Y || hashvalue || K) \), where \( H_2 : \{0, 1\}^* \rightarrow Z_p^* \)
3. Compute \( \tilde{x} = r_x - c_i x_i \mod p, \tilde{y} = r_y - c_i y_i \mod p \)
4. Return \( s = (\tilde{x}, \tilde{y}, c_1, \ldots, c_n) \)

After signing all the hash values of KNs, DO uploads the data set \( A_{data} \) and all the signatures to the cloud server. Next, we discuss how to conduct service transaction between the CSP and the user.

**5.4 ServiceTransact**

In this procedure, we will introduce the TA to achieve fair service transaction between the CSP and the user. When the user submits a range selection query \( rs \) to the CSP, the CSP first computes query answer \( R \) and verification object \( VO \), then sends them to the user following a predetermined protocol. This interaction procedure includes the following two main steps.

- **Step 1:** The CSP computes the \( R \) and \( VO \).

  - When receiving a query request \( rs \) from the user, the CSP reconstructs the MHT, and traverses it to find out the query answer \( R \), and simultaneously generates \( VO \) by Algorithm 2, where the \( VO \) is used to verify \( R \).

**Algorithm 2 VOgenerator**

**Input:**
- \( R = \{a_k, a_{k+1}, \ldots, a_j\} \): the query answer;
- \( s \): the signature set generated in Section 5.3;

**Output:**
- \( VO \): the verification object for \( R \);

1. If \( k > 1 \)
2. \( VO = \{a_{k-1}\}, k' = k - 1 \); // the left boundary data
3. Else
4. \( VO = \{null\}, k' = k \);
5. End if
6. If \( j < n \)
7. \( VO = VO \cup \{a_{j+1}\}, j' = j + 1 \); // the right boundary data
8. Else
9. \( VO = VO \cup \{null\}, j' = j \);
10. End if
11. Put all the KNs into a KN set, denoted as \( S_{KN} \), where the root node is denoted as root;
12. Find the KN from down to up in the reconstructed MHT, denoted as \( kn \in S_{KN} \), which is the first parent to cover all the hash values of \( \{a_k, a_{k+1}, \ldots, a_j\} \);
13. \( VO = VO \cup \{h_{kn}, s(h_{kn})\} \), where \( s(h_{kn}) \) is the signature of hash value of \( kn \).
14. Let \( A(a_k, kn) \) represents the set of nodes in the path from the leaf node containing \( a_k \) to the KN \( kn \);
15. For each node \( \alpha \in A(a_k, kn) \)
16. If there exists a left sibling of the node \( \alpha \), denoted as \( \alpha_{left} \)
17. \( VO = VO \cup \{hashvalue of \alpha_{left}\} \);
18. End if
19. End for
20. Let \( B(a_k', kn) \) represents the set of nodes in the path from the leaf node containing \( a_k' \) to the KN \( kn \);
21. For each node \( \beta \in B(a_k', kn) \)
22. If there exists a right sibling of the node \( \beta \), denoted as \( \beta_{right} \)
23. \( VO = VO \cup \{hashvalue of \beta_{right}\} \);
24. End if
25. End for

Return \( VO \); // the first two elements are the left and right boundary data of \( R \), the next one is a signature together with the signed hash value, the rest are the necessary hash values.

Let us take Fig. 4 for instance to explain Algorithm 2. The nodes marked by dashed circles make up \( VO \). Assume the query answer \( R = \{a_2, a_3\} \), whose left and right boundary data values are \( \{a_1, a_4\} \). From down to up, the first KN covering the union section of query answer and two boundary data values, i.e., \( \{a_1, a_2, a_3, a_4\} \), is the node \( N_{1.4} \). There is no left (right) sibling nodes in the path from \( a_1(a_4) \) to \( N_{1.4} \). Hence \( VO = \{a_1, a_4, N_{1.4}, s_{N_{1.4}}\} \), where \( s_{N_{1.4}} \) is the signature of \( N_{1.4} \). While for the query answer \( \{a_3, a_4\} \), the KN included in \( VO \) is the root node.

**Step 2:** The CSP transmits the \( R \) and \( VO \) to the user.

**Definition 5.2 (Resilient channel).** A resilient channel is a channel that delivers data correctly after a finite, but unknown amount of time.
Definition 5.3 (Unreliable channel). An unreliable channel is a channel that may loss data.

Here assume that the communication channels between the TA and both the CSP and user are resilient, while the communication channels between the CSP and user or user and TA may be unreliable. Inspired by [49], we will apply a fair non-repudiation protocol with off-line TA into transmitting R and VO from the CSP to the user. Here the TA is off-line, because it does not intervene in the protocol while both the CSP and user have no incorrect behavior, and no network error occurs.

Main protocol:
1) CSP → User: \( f_{EOO}, f_{Sub}, User, TA, L, E_K(R), E_{TA}(K||VO), EOO, Sub \)
2) User → CSP: \( f_{EOR}, CSP, TA, L, EOR \)
   if CSP times out then Abort
3) CSP → User: \( f_{EOO_K||VO}, User, L, K, VO, EOO_K||VO \)
   if User times out then Recovery
4) User → CSP: \( f_{EOR_K||VO}, CSP, L, EOR_K||VO \)
   if CSP times out then Recovery

Abort Protocol:
1) CSP → User: \( f_{Abort}, User, Abort \)
   if aborted or recovered then stop
   else aborted = true
2) TA → CSP: \( f_{Con_A}, CSP, User, L, Con_A \)
3) TA → User: \( f_{Con_A}, CSP, User, L, Con_A \)

Recovery Protocol:
1) \( X \rightarrow TA: f_{Rec_X}, f_{Sub}, L, \ h(E_K(R)), E_{TA}(K||VO), Rec_X, Sub, EOR, EOO \)
   if aborted or recovered then stop
   else recovered = true
2) TA → CSP: \( f_{Con_K||VO}, CSP, User, L, K, VO, Con_K||VO, EOR \)

3) TA → User: \( f_{Con_K||VO}, CSP, User, L, K, VO, Con_K||VO, EOR \)

Our protocol is divided into three sub-protocols, a Main Protocol, an Abort Protocol, and a Recovery Protocol, as shown in above three protocols, respectively. The TA does not intervene in the main protocol. The main protocol consists of two parts. The first part is the exchange of the ciphertext of \( R \) under the CSP’s key \( K \), and the evidence of origin for the cipher against an evidence of receipt for this cipher. The second part consists of the exchange of the key \( K \), VO and the corresponding evidence of origin against the evidence of receipt for the key \( K \) and VO. If the second message in Main protocol does not arrive to the CSP, the CSP executes an Abort protocol. If the third or fourth messages in Main protocol does not arrive, the user and CSP, respectively, can launch a Recovery protocol. The Recovery protocol aims to provide the CSP with the possibly missing evidence of receipt for the cipher (EOR), as well as a substitution (\( Con_K||VO \)) for the evidence of receipt of \( K, VO \) and user with a substitution (\( Con_K||VO \)) of the missing evidence of origin for \( K, VO \), as well as \( K, VO \) themselves.

Table 2 shows the evidences owned by the CSP and user, respectively, after executing the above protocols. In practice, most of the time no problem will occur, only the Main Protocol is launched. In this case, the evidence of non-repudiation receipt is \( \{EOR, EOR_K||VO\} \), saved by CSP. The evidence of non-repudiation origin is \( \{EOO, Sub, EOO_K||VO\} \), saved by user.

According to [49], it is known that our interaction protocol can guarantee that after transmitting the \( R \) and VO, both the CSP and user can receive all the expected non-repudiation evidences related to query behavior, fairly and timely. In Section 6, we will prove the interaction traceability between the CSP and user by utilizing these non-repudiation evidences.

5.5 ServiceVerify\((n, Y, R, VO) \rightarrow \text{accept/reject}\)

With the public key of CSP, the user first verifies the evidence \( EOO_K||VO \) to obtain \( K \) and VO, next verifies the evidence \( EOO \) to obtain the \( E_K(R) \). He finally decrypts the \( E_K(R) \) with the key \( K \). Now, the user gets the query answer \( R \) and verification object VO, and could verify the \( R \) according to Algorithm 3.

It is worth noting that what differs from existing query answer authentication schemes is that, the signature provided for user is always accompanied with the corresponding signed hash value. The purpose is to check the signature’s trustiness before verifying the authenticity and completeness of the query answer. Because if the signature is tampered or forged, the subsequent query answer authentication processing (from the 5-th line to the last line in Algorithm 3) is meaningless. The first four lines of code in Algorithm 3 can guarantee that query authentication work is done on the basis of the trusted signature. To some extent, it reduces meaningless verification computation cost. In addition, nobody can identify whom the query data and signature truly come from. Thus our scheme realizes anonymous authentication.
Table 2: Evidence Notations for our non-repudiation protocol with offline TA.

<table>
<thead>
<tr>
<th>Evidence owned by CSP</th>
<th>No problem occurs</th>
<th>CSP launches abort protocol</th>
<th>CSP or User launches recovery protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOR, EOR_{K</td>
<td></td>
<td>VO}</td>
<td>Con_A</td>
</tr>
<tr>
<td>EOO, Sub_EOR_{K</td>
<td></td>
<td>VO}</td>
<td>EOO, Sub, Con_A</td>
</tr>
</tbody>
</table>

Algorithm 3 Verification

Input:
- R: the query answer;
- Y = \{pk_1, pk_2, \ldots, pk_n\}: a set of public keys;
- VO: the verification object;

Output:
- accept or reject;
1. obtain the KN’s signature and hash value from the VO, denoted as s = (x, y, c_1, \ldots, c_n) and hashvalue, respectively;
2. compute c_0 = H_2 \left( Y \parallel \text{hashvalue} \parallel g^x h^y \prod_{j=1}^{n} pk_j^{c_j} \right);
3. if \( \sum_{j=1}^{n} c_j \mod p \neq c_0 \) then return reject;
4. end if
5. reconstruct the KN’s hash value, denoted as hashvalue’, according to R and VO;
6. if hashvalue’ = hashvalue then return accept;
7. else return reject;
8. end if

6 Security and Performance Analysis

In this section, we first prove the security of our scheme in terms of: the unforgeability and anonymity of the signature, the completeness, authenticity, and trustiness of the query answer, the interaction traceability between CSP and user. Then we present the performance analysis of our scheme.

6.1 Security Analysis

The security of our proposed scheme is mainly based on the discrete logarithm assumption (DLA) hard.

Definition 6.1 (DLA). For any probabilistic polynomial time (PPT) algorithm A, the probability that \( Pr[A(g, g^a) = a] \) is negligible, where \( g, g^a \in R \). G.

This Computational Assumption is reasonable, since DLP in large number field is widely considered to be intractable \cite{56, 57, 58}. Therefore a is not deducible from \( g^a \) even if g is publicly known. In this paper, the field \( G \) is large enough to ensure the security of our scheme.

Definition 6.2 (Unforgeability). A signature scheme is unforgeable if for all PPT adversary A, the probability that he can construct a forged signature \( s_M \) which satisfies \( V(Y, M, s_M) = \text{accept} \), denoted as \( \text{Adv}^\text{unf}_A = Pr[A \text{ forges a valid signature}] \) is negligible, where \( M \) is a message, and \( s_M \) is the signature on \( M \).

Definition 6.3 (Anonymity). A signature scheme is anonymous if for any PPT adversary A, the probability that A can guess the signer, denoted as \( \text{Adv}^\text{anon}_A = Pr[A \text{ infers the signer’s public key}] = \frac{1}{n} \), is negligible.

Next we will analyze the security of our scheme from seven aspects, as shown in the following seven theorems.

Theorem 6.1 (Unforgeability). Our signature scheme is unforgeable.

Proof: Let’s prove by contradiction. Assume a hash value \( hv \), whose signature is denoted as \( s = (x, y, c_1, \ldots, c_n) \), provided by DO. From Algorithm 3, we can get

\[ c_0 = H_2 \left( Y \parallel hv \parallel g^x h^y \prod_{j=1}^{n} pk_j^{c_j} \right) \]

where \( \sum_{j=1}^{n} c_j \mod p = c_0 \). Now, an PPT adversary forges a signature, denoted as \( s’ = (x’, y’, c_1’, \ldots, c_n’) \), where there exists at least one element is not equal to the counterpart in s, i.e. \( x’ \neq x \), or \( y’ \neq y \), or \( c_j’ \neq c_j, j = 1, \ldots, n \). \( s’ \) is assumed to satisfy the following equation:

\[ c_0’ = H_2 \left( Y \parallel hv \parallel g^{x’} h^{y’} \prod_{j=1}^{n} pk_j^{c_j’} \right) \]

where \( \sum_{j=1}^{n} c_j’ \mod p = c_0’ \).

1) \( c_0’ = c_0 \). Since the collision-resistance of hash function \( H_2 \), we can deduce

\[ g^{x’} h^{y’} \prod_{j=1}^{n} pk_j^{c_j’} = g^x h^y \prod_{j=1}^{n} pk_j^{c_j} \]

\[ x’ + \sum_{j=1}^{n} c_j’ x_j = x + \sum_{j=1}^{n} c_j x_j \]

\[ \Rightarrow g^{x’} h^{y’} \prod_{j=1}^{n} pk_j^{c_j’} = g^x h^y \prod_{j=1}^{n} pk_j^{c_j} \]

\[ y’ + \sum_{j=1}^{n} c_j y_j = y + \sum_{j=1}^{n} c_j y_j \]

\[ \Rightarrow y’ = y + \sum_{j=1}^{n} (c_j’ - c_j) y_j = 0 \]

\[ \Rightarrow a \]

a) One obvious solution to the above equation is \( x’ = x, y’ = y, c_j’ = c_j, j = 1, \ldots, n \), which contradicts the assumption that \( x’ \neq x \), or \( y’ \neq y \), or \( c_j’ \neq c_j, j = 1, \ldots, n \).

b) Other solutions are hard to calculate, since \( x_j, y_j, j = 1, \ldots, n \) are kept secret.

2) \( c_j’ \neq c_0 \). It is obviously hard to compute \( x’, y’, c_1’, \ldots, c_n’ \) such that \( \sum_{j=1}^{n} c_j’ = H_2 \left( Y \parallel hv \parallel g^{x’} h^{y’} \prod_{j=1}^{n} pk_j^{c_j’} \right) \mod p \), since the one-way hash function and the hard DLP.

Hence, our signature algorithm proves unforgeable.
Theorem 6.2 (Anonymity). Our scheme can guarantee DO’s identity anonymity.

Proof. Here we assume only public key \( pk_i \) can reveal the identity of \( DO_i \). All the information the adversary can get is the signature \( s = \{ \tilde{x}, \tilde{y}, c_1, \ldots, c_n \} \), public key list \( Y \), and public parameter \( PA \). These information holds the following equation:

\[
\sum_{j=1}^{n} c_j = H_2 \left( \|Y\|hv \| g^x h^y \prod_{j=1}^{n} pk_j \right) \mod p
\]

where the prefix \( Y|hv \) can tell nothing about the identity of \( DO_i \). We just consider the suffix, i.e., \( g^x h^y \prod_{j=1}^{n} pk_j \). We denote \( B = \prod_{j=1}^{n} pk_j \), and \( A = g^x h^y B \). From Algorithm 1, we can obtain

\[
\tilde{x} = r_x - c_i x_i \mod p \\
\tilde{y} = r_y - c_i y_i \mod p
\]

Hence the following equation holds:

\[
A = g^x e^{c_i x_i} r_y e^{-c_i y_i} B = g^{x + c_i x_i} r_y B
\]

\[
\Rightarrow pk_i = \frac{B}{A} g^{x + r_y}
\]

As described in Algorithm 1, \( r_x, r_y \) are random values and kept secret. Hence we can not compute the public key \( pk_i \) of \( DO_i \).

Hence, our scheme can guarantee DO’s anonymity.

Compared with the scheme in [9], the sign computation cost and verification computation cost are reduced from \( E + 2M \) and \( 2M \) to \( E + M \) and \( M \), respectively, where \( E \) represents an exponentiation, \( M \) represents a multi-bases exponentiation which is equal to the cost of approximate 1.3 exponentiation. We achieve this by discarding its linkability.

Theorem 6.3 (Completeness). If hash function is collision-resistant, the query answer and VO are all authentic, then our scheme can verify the completeness of query answer.

Proof. Assume the query answer is \( \{a_k, a_{k+1}, \ldots, a_j\} \). In general, the first two elements of VO, as the output of Algorithm 2, are the left and right boundary data. When they are not null, we denote them as \( a_k \) and \( a_j \), respectively. \( a_k \) and \( a_j \) surely dissatisfy the query claims. If we can prove that \( a_k\) is left (right) adjacent tightly to \( a_k(a_j) \), then our scheme can verify the completeness of \( \{a_k, a_{k+1}, \ldots, a_j\} \). Let’s prove by contradiction. Assume that \( a_k\) is not left (right) adjacent to \( a_k(a_j) \). That is to say, there are other data records between \( a_k(a_j) \) and \( a_k(a_j) \). When executing Algorithm 3, the verification result is \( \text{hashvalue}' \neq \text{hashvalue} \) and failing, since hash function is collision-resistant. In consequence, only when \( a_k\) is left (right) adjacent tightly to \( a_k(a_j) \), will the authentication succeed. So far the proof is completed. Our scheme can verify the completeness of query answer.

Theorem 6.4 (Authenticity). If hash function is collision-resistant, and the query answer is complete, our scheme can verify the authenticity of query answer.

Proof. Let’s prove by contradiction. If the query answer is tampered or forged, hash value of the KN, reconstructed using unauthentic query answer and hash values of VO, will differ with the hash value computed from the signature of the same KN. Algorithm 3 will output \( \text{reject} \). In consequence, only when \( \{a_k, a_{k+1}, \ldots, a_j\} \) is authentic, will the verification succeed. The proof is completed. Our scheme can verify the authenticity of query answer.

Theorem 6.5 (Trustiness). If hash function is collision-resistant, our scheme can verify the trustiness of the query answer.

Proof. Theorems 6.3 and 6.4 have proven our scheme can verify the completeness and authenticity of query answer. As for the trustiness, it relies completely on the trustiness of the signature in VO. Assume the signature is denoted as \( s = \{ \tilde{x}, \tilde{y}, c_1, \ldots, c_n \} \). From the signing procedure in Algorithm 1, we can obtain

\[
c_1 + \ldots + c_n \mod p = H_2 \left( \|\text{hashvalue}\| K \right)
\]

\[
= H_2 \left( \|\text{hashvalue}\| g^{x + c_i x_i} h^y + c_i y_i \prod_{j=1, j \neq i}^{n} pk_j \right)
\]

\[
= H_2 \left( \|\text{hashvalue}\| g^x h^y \prod_{j=1, j \neq i}^{n} pk_j \right)
\]

Algorithm 3 tells us that

\[
c_0 = H_2 \left( \|\text{hashvalue}\| g^x h^y \prod_{j=1}^{n} pk_j \right)
\]

So \( \sum_{j=1}^{n} c_j \mod p = c_0 \) holds. Till now, the trustiness of the signature is proved. Hence, our scheme can verify the trustiness of the query answer.

Theorem 6.6 (Traceability). Our scheme can guarantee inter-transaction traceability between CSP and user.

Proof. When the transaction between CSP and user is successful, either only the Main Protocol, or both the Main Protocol and Recovery Protocol, are launched. In each case, we can prove the interaction traceability of our scheme.

1. Only the Main Protocol is launched.

CSP has the evidence of non-repudiation receipt \( \{EOR, EOR_{K||VO}\} \), and user has the evidence of non-repudiation origin \( \{EEO, Sub, EORK||VO\} \).

After verifying \( R \) is correct through Algorithm 3, if user denies receipt of \( R \), CSP can prove his receipt by presenting \( R, E_K(R), K, VO, L \) and \( \{EOR, EORK||VO\} \) to TA. TA executes three checks: 1) \( EORK||VO \) is user’s signature on \( \{EOR_{K||VO}, CSP, L, K, VO, O\}; 2) EOR is user’s signature on \( \{EOR, CSP, TA, L, H_2(E_K(R))\}; 3) R = D_K(E_K(R)). If the above three checks are positive, TA will conclude user received \( R \).

When CSP denies the origin of incorrect \( R \), user has to present \( R, E_K(R), K, VO, L \) and \( \{EEO, Sub, EORK||VO\} \) to TA. Similarly, TA executes four checks: 1) \( EEO_{K||VO} \) is CSP’s signature on \( \{EEO, Cient, L, K, VO\}; 2) EEO is CSP’s signature on \( \{EEO, User, TA, L, H_2(E_K(R))\}; 3) Sub is CSP’s signature on \( \{Sub, User, L, ET_A(K||VO)\}; 4)
To reduce the hashing computation overhead at user, our computation cost is still high when facing the big data. Theorem 6.7 (High Efficiency).

6.2 Performance Analysis

**Theorem 6.7 (High Efficiency).** The high efficiency of our scheme is mainly manifested in ServiceVerify from two aspects: on the one hand, it reconstructs the subtree rooted in the KN of VO; on the other hand, it first checks the trustiness of signature in VO.

**Proof.** Even though one hashing operation is around 50 times faster than modular multiplication [11], the hashing computation cost is still high when facing the big data. To reduce the hashing computation overhead at user, our scheme improves the traditional MHT-based verification scheme from the following two aspects.

On the one hand, our scheme signs the hash values of the root node, as well as the nodes on level \( \log_2 \sqrt{N} + 0.5 \). So if there is one KN on internal level covering the query answer and the two boundary values, our scheme just needs to reconstruct the subtree rooted in such a KN, while the traditional schemes always reconstruct the whole MHT.

On the other hand, when verifying query answer, previous schemes firstly reconstruct the hash value of the root node using a lot of hash computation, then compare it with the hash value computed from the signature. If the signature is not trusted, not only much hash computation is wasted, but also the query answer will prove to be false all the same even though it is correct. However, such problem can be avoided in our scheme by concatenating the hash values with their signatures, then checking trustiness before verifying the completeness and authenticity.

Next we will compare our scheme performance with those in [9] and [7]. Our scheme is improved from both [9] and [7]. The scheme proposed in [9] signs the values one by one. Chapter 3 in [7] describes an authentication scheme for selection queries on the basis of MHT. Assume the query answer \( R \) contains \( l \) data values. The comparison and analysis results are shown in Table 3. In Table 3, our scheme has two cases about verification computation cost and VO size, respectively. It is caused by the KN in VO. If the KN is the root node, both verification computation cost and VO size are the same as those in [7]. If the KN is an internal node, both verification computation cost and VO size are much smaller. In essence, our scheme sacrifices slightly higher sign computation cost to improve the verification efficiency and reduce the communication cost. Hence the sign computation cost is a little higher than that in [7], but far lower than that in [9], which will be proved by experiments in Section 7.

### 7 EXPERIMENT

All algorithms are implemented using Visual C++ 6.0 on a Windows 8.1 system with Intel CORE i7-4500U CPU @ 1.80GHz and 8.00G RAM. We implement the proposed scheme and evaluate the performance over multiple groups of random data. The selection queries are of the form as

\[
\text{SELECT * FROM stream WHERE } l < A_i < u_i.
\]

In order to compare with other schemes fairly, there are different numbers of random data in each group. Moreover, 1000 random queries are processed. We compute averages of sign computation time, VO generation time, verification computation time, VO size and tamper detection time, respectively, as experimental results shown in Figs. 5–9.

Figs. 5–8 show the cost comparison results of sign computation, VO generation, verification computation and VO storage, respectively. Firstly, it is evident that all these costs, except the VO generation cost, as shown in the lines marked by circles, increase quickly. The scheme proposed in [9] signs all the data values one by one. As a consequence, the CSP need do nothing for VO generation, and just takes the signatures as the VO. In brief, the signature cost is proportional to the whole database size, and the VO generation cost is trivial. In our scenario, the volume of the data is very large. Hence the scheme [9] isn’t applicable. By the way, the verification computation cost and VO size in [9] are experimentally proportional to the number of data in a query answer. But for simplicity, we present all these performance in one figure.

Next, let’s see the comparison results between our scheme and the scheme proposed in [7], shown in lines marked by triangles and rectangles, respectively. We can see that our scheme has slightly higher sign computation cost, but the lower verification cost and the smaller VO size. Our scheme sacrifices slightly higher sign computation cost to improve the verification efficiency and reduce the communication cost, as analyzed in Theorem 6.7.

Finally, we compare the efficiency in detecting whether the signature is valid or not. Fig. 9 shows our scheme achieves the highest detection efficiency. The detection time cost of the schemes in [7], [9] is almost the same as the verification computation cost in Fig. 7. It is because that the signature tampering can not be detected until the whole verify process is completed. In our scheme, the signature is always accompanied with the corresponding signed hash value, which can help us to detect the signature tampering efficiently by Algorithm 3.

All the experimental results meet the theoretical analysis in Table 3.

### 8 CONCLUSION

Since there are multiple data providers and a wide range of users in cloud service systems, it is hard to take full advantage of cloud data to serve people well on the premise of not infringing upon the interests of others. In this paper, it is the first time to propose a cooperative query answer authentication scheme which applies to cloud. This scheme can not only verify the trustiness, completeness, authenticity of the query answers efficiently, but also satisfy DO’s requirement for anonymity and guarantee non-repudiation service between CSP and user. Firstly, the proposed scheme
chooses and signs the KN in the MHT based on the ring signature scheme, which can both verify the correct of query result when keeping DO anonymous, and supports multiple DOs. Secondly, we introduce a non-repudiation protocol based on VO to solve the repudiable behaviors of CSP and user. Finally, the experimental results show our proposed scheme is of higher efficiency and lower communication cost than others.

**ACKNOWLEDGMENTS**

The work was supported by the National Natural Science Foundation of China (61472001, 61472283, U1405255), and the scientific research construction fee of Anhui University.

**REFERENCES**


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2676008, IEEE Access


Liangmin Wang received his B. S. degree in Computational Mathematics in Jilin University, Changchun, China, in 1999, and the Ph.D degree in Cryptology from XiDian University, Xi’an, China, in 2007. He is a full professor in the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. He has been honored as a “Wan-Jiang Scholar” of Anhui Province since Nov. 2013. Now his research interests include security protocols for wireless networks and Privacy & Security of Big Data.

He has published over 60 technical papers at premium international journals and conferences, like IEEE Transactions on Information Forensics and Security, IEEE Transactions on Vehicular Technology, IEEE GlobeCOM, IEEE WCNC. Dr WANG has been served as the TPC of many IEEE conferences, such as IEEE ICC, IEEE HPCC, IEEE TrustCOM. Now he is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of Chinese Computer Federation.
Qingqing Xie received the B. Eng. degree in computer science and technology from Anhui University, PRC in 2012. She is currently a Ph.D. candidate in Department of Computer Science and Technology at Anhui University. She is also a research visitor in Department of Computer Science at Boise State University. Her research interest includes data security, cloud computing, and applied cryptography.

Hong Zhong received her B. S. degree in applied mathematics in Anhui University, China, in 1986, and the Ph.D degree in computer science and technology from University of Science and Technology of China (USTC), China, in 2005. Now she is a professor and Phd Advisor of Anhui University. Her research interests include security protocols and wireless sensor networks.