Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2022.027536
Article

Tech Science Press

Integrity Assurance Method of Multi-Keyword Query for Encrypted Outsourced Data

Ling Wang[1], Shan Ji[2], Zhaokang Wang[3], Xiaowan Wang[4,*], Ghulam Mohiuddin[5] and Yongjun Ren[1]

[1]Engineering Research Center of Digital Forensics, Ministry of Education, School of Computer Science, Nanjing University of Information Science & Technology, Nanjing, 210044, China
[2]Zhengde Polytechnic, Nanjing, 211106, China
[3]Nanjing University of Aeronautics and Astronautics, Nanjing, 210008, China
[4]Xi'an University of Posts & Telecommunications, Xi'an, 710061, China
[5]Department of Cyber Security at VaporVM, Abu Dhabi, 999041, United Arab Emirates
*Corresponding Author: Xiaowan Wang. Email: wangxiaowanxian@yeah.net
Received: 20 January 2022; Accepted: 02 March 2022

Abstract: As the data scale exceeds the petabyte level, more and more users outsource their local databases to third-party service providers to save local storage costs and avoid cumbersome local data management. However, because third-party service providers are not fully trusted, they may leak outsourced data and bring security risks to the personal privacy of data users. The service provider may also return incorrect or incomplete query results to the data user. Therefore, in this paper, we propose a Lightweight and Verifiable Multi-keyword Query Scheme for the integrity verification of multi-keyword query of outsourced data. This scheme supports multi-keyword query integrity verification and does not require a third-party auditor, combining a hash-based accumulator that provides correctness verification and a cuckoo filter that provides integrity verification to form a provably secure composite verification structure, and at the same time, it uses controllable Paillier encryption to support data update operations. This scheme strikes a balance between the cost of verification structure generation, verification object generation, and proof verification cost to enable data users to perform secure multi-keyword query while efficiently verifying the integrity of query results. The security proof and the performance analysis show that the proposed scheme is secure and efficient for practical deployment.

Keywords: Outsourced data; multi-keyword; query integrity

1 Introduction

Due to the rapid growth of data volume, the scale of the data management system exceeds the petabyte (PB) level, and the data management capability lags behind the data collection and storage capability. The user's specific data analysis needs cannot be met, and it is impossible to provide easy-to-deploy data management services [1,2]. With the explosive development of the Internet and the advancement of network technology, a new paradigm called "Database as a Service (DaaS)" has emerged. In the DaaS

paradigm, data owners who have large amounts of data but lack data management resources outsource their data and data analysis to third-party service providers. Third-party service providers provide sufficient hardware, software, technology, and network resources to host the database and perform data management [3–5]. By using these services, data users can achieve large-capacity storage, accelerated processing, and complex data management and analysis at a lower cost [6–8].

While data outsourcing offers many convenient services, it brings many new security issues that do not exist in traditional databases because data is stored on the servers of the outsourcing service provider, the data owner no longer has direct control over the data and the service provider is not fully trusted [9,10]. In order to ensure the confidentiality of outsourced data, it is necessary to encrypt the outsourced data, which makes it difficult to perform multi-keyword query on the encrypted data [11,12]. In addition, service providers driven by their own interests or affected by factors such as software and hardware operating failures, may honestly perform partial query operations and return incorrect or incomplete query results to users [13,14]. It is necessary to provide users with a lightweight and verifiable mechanism to ensure the correctness and completeness of multi-keyword query results.

Therefore, this paper mainly studies the multi-keyword query integrity verification of outsourced data [15,16]. Query integrity refers to the ability to verify the correctness and completeness of the query results returned by the third-party service provider. Specifically, correctness means that data in the query results is true and error-free; completeness means that the query results are not missing and include all valid records that satisfy the query conditions. On the other hand, in order to meet users' efficient data retrieval needs, the server allows users to input multiple keywords at once, and can perform multi-keyword query, that is, given a set of retrieved keywords, retrieve all documents containing these keywords [17,18]. Multi-keyword query can improve query efficiency and save computation and communication overhead.

Many solutions have been proposed in recent years to solve the problem of multi-keyword query integrity verification [19–21]. Wang et al. [22,23] proposed a verifiable audit scheme based on Invertible Bloom Filter, which supported completeness and correctness verification of query results, and supported working in a dynamic environment. However, existing solutions that achieve both correctness and completeness verification usually require a third-party auditor (TPA). However, because TPA is not completely credible, a malicious TPA may return an opposite result to the user even if it calculates the correct authentication result. The integrity of user data will still be threatened, and TPA will also generate additional communication overhead.

In order to solve the above problems, this paper proposes a lightweight and verifiable multi-keyword query scheme for outsourced data. The main contributions of this paper are as follows:

(1) We propose a Lightweight and Verifiable Multi-keyword Query Scheme (*LVMQ*), which supports multi-keyword query integrity verification with low query overhead and high verification efficiency.
(2) A TPA is not needed in the *LVMQ* scheme, and a hash-based accumulator is used to reduce communication overhead and improve system data security.
(3) The *LVMQ* scheme combines a cuckoo filter and a hash-based accumulator to construct a provably secure composite verification structure, which strikes a balance between the verification structure generation cost, the verification object generation cost, and the proof verification cost.

The rest of the paper is organized as follows. Section 2 mainly introduces the existing query integrity verification methods. Section 3 introduces security threats and security requirements of the system. Section 4 introduces the data security model and related technical methods. Section 5 introduces the implementation details of the *LVMQ* scheme. Section 6 analyzes the security of the scheme. Section 7 analyzes the performance of the program. Section 8 gives a final summary.

2 Related Work

The data stored on the outsourcing server may be tampered with or deleted. The role of query integrity verification is to ensure that data users can verify data in the query result is correct and complete. At present, there are many schemes for query integrity verification, and these schemes can be divided into the following three verification methods [24–26].

(1) Verification method based on digital signature

The first method is based on digital signature, mainly using digital signature chains [27,28]. The basic idea of the digital signature chain is to organize data into a chain according to a certain logical relationship, and the signature of each data is determined by itself and the logically related data together. In this way, once a certain data is deleted, the logical relationship is destroyed and the signature verification fails. The signature chain method has serious efficiency problems: signature chains are expensive to establish and maintain.

Pang et al. put forward a scheme for verifying the integrity of relational query results, using digital signature technology to sign each record. The correctness of the scheme was guaranteed by the signature, and the completeness was guaranteed by the signature chain. Narasimha et al. proposed a Digital Signature Aggregation and Chaining (DSAC) scheme, which first sorted all tuples by each attribute, and then generated a chained digital signature for each tuple based on the sorting result, effectively verifying the integrity of non-empty query results. Zhang et al. [29] improved the DSAC method and proposed the chain embedded signature method, which replaced the signature content of the tuple from the predecessor tuple concatenation to the predecessor primary key concatenation, reducing the number of updates.

(2) Verification method based on authenticated data structure

The second method is based on authenticated data structure [30,31]. At present, the most researched one is Merkle hash tree (MHT) and various variants based on MHT. The main idea is to generate a MHT-based index for the entire database, and complete the integrity audit by recalculating the signature on the MHT root. Most MHT-based schemes can guarantee the integrity of query results.

Devanbu et al. [32] first used MHT to study the verification of query results. Even if the cloud service provider (CSP) is dishonest, data users can verify the correctness of the query results. Ma et al. generated a MHT for each tuple, which reduced the communication cost of the verification process and improved the verification efficiency. Li et al. proposed the dynamic authentication index structure of outsourced database by using B+ tree. Wang et al. used MHT, Bloom Filter and Bloom Filter Tree, with tools such as pseudo-random permutation, hash function, etc., regardless of whether the query results returned by CSP are not empty, data users can verify the query integrity. Later, Wang et al. proposed a solution to support dynamic database updates, which fully realized correctness and completeness verification of the query results by data users.

(3) Verification method based on probability

The third method is based on probability [33]. The main idea is that the data owner first inserts some fake tuples into the database. However, an implicit assumption of this method is that all data users must know the fake tuples. A dishonest CSP can collude with any compromised data user and then learn about fake tuples. In addition, this method requires CSP to return all the attributes of the tuple, so it cannot support some common database operations, such as projection.

Sion let the data owner add a fake challenge token to the query request, forcing the CSP to execute all query requests, otherwise the probability of the CSP returning the correct challenge response will be greatly reduced. Xie et al. let the data owner add a small number of fake tuples to the outsourced database and judge whether the CSP's retrieval behavior was honest or not by analyzing the fake tuples in the query results.

In summary, many researchers have carried out extensive research on outsourced data query integrity verification methods. However, the lightweight and verifiable multi-keyword query method for outsourced data still needs to be studied.

3 Problem Statement

3.1 Security Threats

Assuming that the service provider is a semi-honest but friendly server, it may not follow the recommended protocol correctly, may not return a small part of the query results, and only perform a small part of the query operation honestly, that is, it may return incorrect or incomplete query results. Communication channels involving service providers are considered insecure and may be subject to external attacks and internal attacks.

① External attacks: The revoked or unauthorized users try to access data outside the scope of authority, hoping to obtain sensitive information about the outsourced data through public channels.

② Internal attacks: The service provider may not be able to completely send ciphertext data or additional information to the data user due to non-subjective factors such as machine failure, network failure, and hacker intrusion, or it may deliberately send incomplete or incorrect information to the data user due to subjective factors such as cost saving.

3.2 Security Requirements

In order to ensure the security and efficiency of the system, the proposed scheme should meet the following security requirements:

- Query integrity: Data users can verify whether the query results returned by service provider (SP) are correct and complete.
- Dynamic data update: The data owner can perform a data dynamic update operation on the outsourced database.
- Lightweight: The scheme has lower communication and calculation overhead in terms of integrity verification.

4 Preliminaries

4.1 Security Model

Aiming at the problem of outsourced data security, a system security model is constructed, as shown in Fig. 1. Three entities are involved in the system model: data owner, data users and service provider.

- Data Owner: Encrypt the data in the outsourced database, extract keywords at the same time, build a keyword index and encrypt it, outsource the encrypted data and index to the service provider together.
- Data Users: Send a query request to the service provider and use the returned query result and verification object to verify the integrity of the query result.
- Service Provider: Store encrypted data and index, manage data and provide data retrieval services to data users.

The data owner extracts keywords from data files and establishes a corresponding keyword index, then encrypts index files and data files together, and outsources them to the service provider, and shares them with data users. When the data user sends a query request to the service provider for keyword query, the keyword trapdoor is first generated and sent to the service provider. After receiving the data user's query request, the service provider responds to the data user with the corresponding record and proof of integrity query. The

data user then uses the proof to verify the correctness and completeness of the query result. If the query result is correct and complete, the data user decrypts the result and outputs Accept, otherwise outputs Reject.

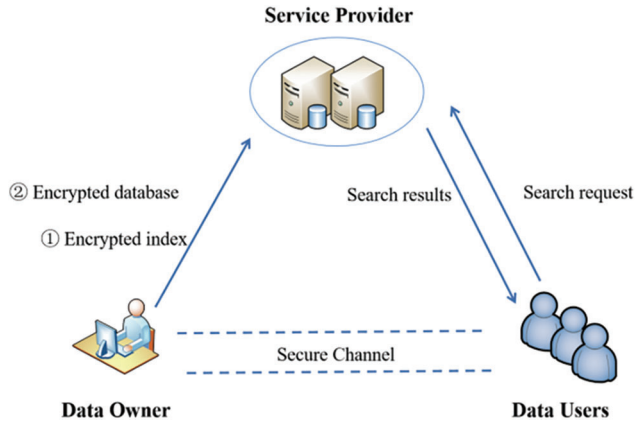

Figure 1: System model

4.2 Cuckoo Filter

Cuckoo filter [34] is a filter based on the cuckoo hash algorithm. It hashes the storage item, and stores its hash value in an array with a certain number of bits. When querying, determine whether a stored item exists by determining whether the hash of equal bits is in the array. Cuckoo filter supports dynamic deletion of items, provides better lookup performance than Bloom filter, has better space efficiency and is easier to implement.

The following describes the insert, lookup and delete operations of the cuckoo filter respectively.

(1) Insert operation

When inserting a new item x in the hash table, first calculate the fingerprint $f \leftarrow fingerprint(x)$ of x and two optional candidate buckets $i_1 \leftarrow hash(x)$ and $i_2 \leftarrow hash(f) \oplus i_1$. Then read i_1 and i_2, if there is an empty cell e in i_1 or i_2, put f in the first available empty cell e. If two candidate buckets are not empty, one of the buckets will be randomly selected, and the selected existing item y will be kicked out, and the item y is placed in a spare position. If there is still item z in the alternate position, then the item z is kicked out, and so on, until the number of kicks n reaches a maximum threshold *MaxNumKicks*. Once the hash table is full, perform a refresh hash table operation.

(2) Lookup operation

When looking up whether a given item x belongs to the set, calculate the fingerprint f of x and two candidate buckets i_1 and i_2, and then read i_1 and i_2. If f matches the fingerprint of any one of the buckets i_1 or i_2, it returns *Success*, otherwise it returns *Fail*.

(3) Delete operation

When deleting a given item x, calculate the fingerprint f of x and the two candidate buckets i_1 and i_2, then read i_1 and i_2, if f matches the fingerprint of any one of the buckets i_1 or i_2, then delete the matching fingerprint from the corresponding candidate bucket.

4.3 Hash-Based Accumulator

The cryptographic accumulator is a one-way member hash function, which allows the user to prove that a potential element is a member of a certain set without revealing the individual members of the set. The cryptographic accumulator scheme allows elements in the finite set $X = \{x_1, \ldots, x_n\}$ to be accumulated into a compact and constant-sized value acc_x. The accumulator is defined as:

$$acc_x = h(h(h(\ldots h(h(h(g, x_1), x_2), x_3), \ldots, x_{n-2}), x_{n-1}), x_n) \quad (1)$$

By calculating the witness wit_{x_i} of each element $x_i \in X$ in the set and verifying $h(wit_{x_i}, x_i) = acc_x$, the membership of element x_i can be effectively proved.

The traditional accumulator will introduce a trusted third party-the accumulator manager, he can delete elements in the accumulator, and at the same time create a membership witness for any element. The only known accumulator that can be safely used in the untrusted system is the accumulator based on Merkle trees. This is because the Merkle tree does not require trapdoor values at any node during its operation, and the implementation of Rivest Shamir Adleman (RSA) algorithm and bilinear map accumulators is limited to trusted accumulator manager settings because they rely on the secret trapdoor values used for initialization and constant time operations. Hash-based dynamic accumulators have no trusted settings or manager requirements and use a perfect Merkle tree forest that allows elements to be efficiently removed from the accumulator.

5 Proposed Scheme

5.1 Overview

We propose a Lightweight and Verifiable Multi-keyword Query Scheme (*LVMQ*). The main idea is to choose a hash-based accumulator and combine it with a cuckoo filter to construct a provably secure composite verification structure to verify the correctness and completeness of query results without a third-party auditor. In addition, combined with controllable Paillier encryption to effectively support dynamic data update operations. As shown in Fig. 2: The proposed scheme includes the following five stages: system setup, data outsourcing, data retrieval, data verification and data update. The program is a tuple consisting of eight polynomial time algorithms (***KeyGen***, ***SigGen***, ***BuildCF***, ***BuildACC***, ***BuildIndex***, ***Proof***, ***Verify***, ***Update***). The algorithm is specifically described as follows.

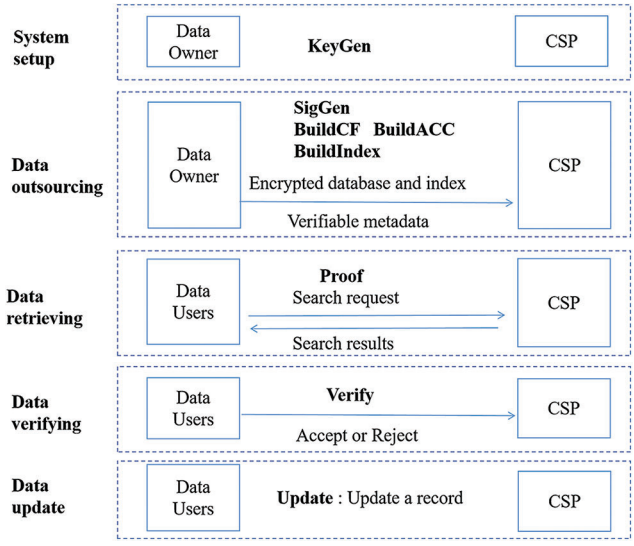

Figure 2: Scheme process

1) ***KeyGen***$(\boldsymbol{1}^{\lambda}) \rightarrow (\boldsymbol{sk}, \boldsymbol{pk})$**:** Key generation algorithm. The data owner initializes the system parameters. Input the security parameter λ, and output secret key sk and public key pk.
2) ***SigGen*****(*pk*, *sk*, *R*) → *SC*:** Record generation algorithm. The data owner generates the signature of the record. Input public key pk, secret key sk, database R, and output metadata SC composed of ciphertext c_{ij} and signature σ_{ij}.
3) ***BuildCF*****(*pk*, *sk*, *R*) → *CF*:** Filter generation algorithm. The data owner constructs a cuckoo filter for each attribute. Input public key pk, secret key sk, database R, and output filter CF.
4) ***BuildACC*****(*pk*, *sk*, *R*) → *ACC*:** Accumulator generation algorithm. The data owner constructs a hash-based accumulator for each attribute. Input public key pk, secret key sk, database R, and output accumulator ACC.
5) ***BuildIndex*****(*pk*, *sk*, *R*, *W*) → *Index*:** Index generation algorithm. The data owner extracts keywords and builds a query index. Input public key pk, secret key sk, database R, multi-keyword set W, and output security index structure $Index$.
6) ***Proof*** $(\boldsymbol{pk}, \boldsymbol{T}, \boldsymbol{SC}, \boldsymbol{CF}, \boldsymbol{ACC}) \rightarrow \boldsymbol{P}$**:** Proof generation algorithm. The service provider generates a proof and returns it to data users. Input public key pk, query request T, metadata SC, filter CF, accumulator ACC, and output proof P.
7) ***Verify*** $(\boldsymbol{sk}, \boldsymbol{pk}, \boldsymbol{I}, \boldsymbol{J}, \boldsymbol{P}) \rightarrow \{\boldsymbol{Accept}, \boldsymbol{Reject}\}$**:** Verification algorithm. Data users verify the integrity of query results. Input public key pk, secret key sk, set I and J, proof P, and output the verification result $Accept$ or $Reject$.
8) ***Update*** $(\{\boldsymbol{c_j}||\sigma_j\}_{j=1}^{n}, \{\boldsymbol{f_j}\}_{j=1}^{n}, \{\boldsymbol{f_j'}\}_{j=1}^{n}) \rightarrow (\boldsymbol{SC}, \boldsymbol{CF})$**:** Update algorithm. Input ciphertext c_{ij}, signature σ_{ij}, new fingerprint f, old fingerprint f', and output updated SC and CF.

5.2 Proposed Scheme

This section introduces the proposed scheme in detail. The *LVMQ* scheme includes five stages: system setup, data outsourcing, data retrieval, data verification and data update.

Suppose G_1, G_2 and G_T are three multiplicative cyclic groups of prime number p, which g_1 is generator of G_1 and g_2 is generator of G_2. Let e:$G_1 \times G_1 \rightarrow G_T$ is a bilinear mapping. $\Pi_0 = (KeyGen_0, Enc_0, Trapdoor_0, Search_0, Dec_0)$ is a secure SSE, $\Pi_1 = (KeyGen_1, Enc_1, Dec_1)$ is a controllable Paillier encryption. Let $h_1 : \{0, 1\}^* \rightarrow G_1$ denotes a hash function that is securely mapped to points, and h_2:$G_1 \rightarrow Z_p$ denotes a function that uniformly maps G_1 group elements to Z_p.

5.2.1 System Setup

In the system setup stage, the data owner initializes the system necessary parameters. The data owner generates secret key $sk = (k_0, sk_1, x)$ and public key $pk = (\lambda, pk_1, g_1, g_2, u, v, h_1, h_2)$ according to algorithm ***KeyGen***. The data owner then discloses pk and shares sk with the authenticated user.

5.2.2 Data Outsourcing

Suppose the data owner wants to outsource the relational database $R = (A_1, A_2, \ldots, A_n)$ to the service provider, where $A_j (1 \leq j \leq n)$ is the name of the j-th attribute, and a_{ij} is the attribute value of the attribute column A_j belonging to the i-th $(1 \leq i \leq m)$ record.

- Database encryption

First, according to algorithm ***SigGen***, the data owner uses an encryption algorithm to encrypt each element a_{ij} in the database R, generates ciphertext c_{ij}:

$$c_{ij} \leftarrow Enc_0(k_0, a_{ij}) \tag{2}$$

and sign σ_{ij}:

$$\sigma_{ij} \leftarrow (h_1(i||j) \cdot g_1^{c_{ij}})^x \tag{3}$$

Finally, the metadata SC composed of ciphertext $\{c_{ij}\}_{1\le i\le m,1\le j\le n}$ and signature $\{\sigma_{ij}\}_{1\le i\le m,1\le j\le n}$ is output.

- Build cuckoo filter CF

According to algorithm ***BuildCF***, the data owner constructs a cuckoo filter CF_j for the j-th attribute column. The owner calculates and generates the fingerprint f:$f \leftarrow fingerprint(a)$, selects three random numbers r_{j_1}, r_{j_2} and r_{j_3}, which satisfying

$$n_j(a) \leftarrow r_{j_1} + r_{j_2} - r_{j_3} \tag{4}$$

Encrypt these three random numbers respectively: $cr_{j_1} \leftarrow Enc_1(pk_1, r_{j_1})$, $cr_{j_2} \leftarrow Enc_1(pk_1, r_{j_2})$, $cr_{j_3} \leftarrow Enc_1(pk_1, r_{j_3})$ and generate auxiliary information $cr_{j_1}||cr_{j_2}||cr_{j_3}$. The fingerprint and auxiliary information are stored in the candidate buckets of the cuckoo filter hash table, so that the equation

$$Dec_1(cr_{j_1})Dec_1(cr_{j_2}) - Dec_1(cr_{j_3}) = |n_j(a)| \tag{5}$$

is established, where $n_j(a)$ is the total number of records whose attribute A_j value is equal to a. Finally, output filter $CF = \{CF_j\}_{1\le j\le n}$.

- Build a hash-based accumulator ACC

According to algorithm ***BuildACC***, the data owner constructs an accumulator ACC_j: $ACC_j = g\prod_{i=1}^{n} a \bmod \varphi(N)$ for the j-th attribute. For the accumulative tree in ACC, the accumulative value acc and the root $leaf$ are stored in each tree. Finally, the accumulator $ACC = \{ACC_j\}_{1\le j\le n}$ is output.

- Build multi-keyword index $Index$

According to algorithm ***BuildIndex***, the data owner first performs word segmentation processing on the document to obtain a keyword set $W = \{w_1, w_2, \ldots, w_m\}$. Then build an index for each data file according to the keywords, and then use a symmetric encryption algorithm to encrypt the index to form an encrypted multi-keyword index $Index$.

- Finally, the data owner outsources the metadata SC, filter CF, accumulator ACC, and multi-keyword index $Index$ to the service provider.

5.2.3 Data Retrieving

In the data retrieval phase, the authenticated user submits a query request to the SP to obtain the corresponding record from the outsourced database under the premise of ensuring correctness and completeness. The SP performs the query in the encrypted database, generates proofs of integrity of the query result, and sends them together to the data user.

Suppose a data user wants to query for records that meet $A_q = a$. The following detailed process should be performed:

- First, the data user selects a random value $j \in J (J = \{j_1, j_2, \ldots, j_t\})$ for each element. Then, calculate the fingerprint $f \leftarrow fingerprint(a)$ and retrieve the mark $T_w \leftarrow Trapdoor_0(a)$. Finally, send a query request $T = (q||T_w||J||f||\{y_j\}_{j\in J})$ to the service provider.
- When receiving the request T, the service provider first uses algorithm $Search_0(t)$ on the q-th attribute column. The service provider then generates a proof $P = (\mu||\sigma||R||Aux)$. Finally, the corresponding result $(\{c_{ij}\}_{i\in I,j\in J}, P)$ is sent to the data user. Query in the hash table of the filter CF, if the fingerprint f is in the

table, obtain the additional information $Aux = cr_{q_1}||cr_{q_2}||cr_{q_3}$, otherwise $Aux = \phi$. Finally output the proof $P = (\mu||\sigma||R||Aux)$.

5.2.4 Data Verifying

When receiving the query result from the SP, the data user checks the integrity of the result. If the returned result is correct and complete, the data user decrypts $\{c_{ij}\}_{i\in I, j\in J}$ through $Dec_0(\{c_{ij}\}_{i\in I, j\in J})$ and accepts them, otherwise rejects them.

Check the validity of the following formula.

$$e(\sigma \cdot R^{h_2(R)},\ g_2) \stackrel{?}{=} e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} h_1(i||j)^{y_j} \cdot g_1^{\mu},\ v\right) \tag{6}$$

If invalid, output *Reject*; otherwise, if $Aux \neq \phi$, decrypt $r_{q_1} = Dec_1(sk_1,\ cr_{q_1})$, $r_{q_2} = Dec_1(sk_1,\ cr_{q_2})$, $r_{q_3} = Dec_1(sk_1,\ cr_{q_3})$. If $r_{q_1} + r_{q_2} - r_{q_3} \neq |I|$, it returns *Reject*, otherwise it returns *Accept*.

5.2.5 Data Update

The data owner can update records in the encrypted database. Assume that the data owner wants to update a record $(a_1, a_2, \ldots, a_n)$, where a_j is the j-th attribute value. According to algorithm ***Update***, for the different value a in the j-th attribute, the data owner first calculates ciphertext c_j, signature σ_j, fingerprint f_j of the new record, and fingerprint f_j' of the old record of the record to be updated. Then, the owner sends these to the SP together. After receiving them, SP first obtains $cr_{j_1}||cr_{j_2}||cr_{j_3}$ from the candidate bucket corresponding to CF_j through fingerprint f_j. Then calculate a new $cr_{j_2}{:}cr_{j_2} \leftarrow cr_{j_2} \cdot Enc_1(pk_1,\ 2)$ and a new $cr_{j_3}{:}cr_{j_3} \leftarrow cr_{j_3} \cdot Enc_1(pk_1,\ 3)$. The SP stores the new $cr_{j_1}||cr_{j_2}||cr_{j_3}$ to the correct position of CF_j and inserts $\{c_j||\sigma_j\}_{1\le j\le n}$ into the SC. Finally output $SC = \{c_j||\sigma_j\}_{1\le j\le n}$ and $CF = \{CF_j\}_{1\le j\le n}$.

6 Security Analysis

6.1 Correctness

Analyze the correctness of the query results from two aspects.

(1) If the service provider does not modify the data, the result must be such that the formula $e(\sigma \cdot R^{h_2(R)},\ g_2) \stackrel{?}{=} e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} h_1(i||j)^{y_j} \cdot g_1^{\mu},\ v\right)$ holds. In this case, correctness can be explained as follows:

$$\begin{aligned} e(\sigma \cdot R^{h_2(R)},\ g_2) &= e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} \sigma_{i,j}^{y_j}(g_1^{xr})^{h_2(R)},\ g_2\right) \\ &= e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} (h_1(i||j)g_1^{c_{i,j}})^{xy_j}(g_1^{rh_2(R)})^{x},\ g_2\right) \\ &= e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} h_1(i||j)^{y_j} g_1^{\sum_{i=i_1}^{i_s}\sum_{j=j_1}^{j_t} c_{i,j}y_j + rh_2(R)},\ g_2^{x}\right) \\ &= e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} h_1(i||j)^{y_j} \cdot g_1^{\mu},\ v\right) \end{aligned} \tag{7}$$

In which, the randomly selected $\{y_j\}_{j=1}^{n}$ ensures the correctness of a single element in the database.

(2) If the service provider maliciously modifies the data, the probability of the service provider forging $(\mu,\ \sigma,\ R)$ through the formula $e(\sigma \cdot R^{h_2(R)},\ g_2) \stackrel{?}{=} e\left(\prod_{i=i_1}^{i_s}\prod_{j=j_1}^{j_t} h_1(i||j)^{y_j} \cdot g_1^{\mu},\ v\right)$ should be negligible.

6.2 Completeness

In the context of controllable Paillier encryption, a service provider cannot construct a valid integrity-validated ciphertext without the help of a data owner or a data user authenticated by **Theorem 1**.

Theorem 1. If the Paillier encryption scheme $\Pi = (KeyGen, Enc, Dec)$ is *IND-CPA* secure and H is $(t, q, L, \varepsilon) - MAC$ secure, there is no adversary who can construct a valid ciphertext with a non-negligible probability.

Proof: Use contradictions to prove this theorem. Assuming that there is an adversary A who can construct an effective ciphertext with a non-negligible advantage, whose goal is to break the $(t, q, L, \varepsilon) - MAC$ security of hash H. First, B uses $(pk_1,\ sk_1) \leftarrow \Pi.KeyGen(1^{\lambda})$ and $sk_0 \leftarrow \{0,\ 1\}^{\lambda}$ to set the game parameters. Then A can perform q queries at most. In each query, B randomly selects r_i, calculates c_i, and sends $r_i||sk_0||c_i$ to H. Then B obtains the hash value h_i from H and sends (r_i, h_i, c_i) as ciphertext to A. After the query, A outputs a ciphertext (r, h, c) to B and B sends $r||sk_0||c$ and h to H. Finally, H checks the validity of $h = H(r||sk_0||c)$. If the above equation is valid, then B succeeds, otherwise B fails. Suppose $\Pr[(r, h, c) \text{ is valid}] = \varepsilon$, and $\Pr[B \text{ success}] = \varepsilon$, where ε cannot be ignored. Then B breaches $(t, q, L, \varepsilon) - MAC$ security of H, which is a contradiction.

Therefore, controllable Paillier encryption can prevent dishonest service providers from returning a small portion of the query results.

7 Performance Evaluation

The experiment uses Pairing-based cryptography (PBC) library and OpenSSL library. Perform 100 times for each test and take the average. In order to accurately measure the overhead of the server and the client, all experiments are performed on the same machine, using Intel Xeon (R) E5-1620 3.50 GHz CPU and 16 gb RAM to run Linux. We conduct a comprehensive experimental evaluation of the proposed scheme, including the construction time of the cuckoo filter *CF* and the accumulator *ACC* in the data outsourcing phase, as well as the communication cost and the integrity verification cost.

7.1 Cost of CF Building Time

In the experiment, a timer is set to obtain the time for constructing the cuckoo filter *CF* for each different attribute, and then the average value is taken. Change the number of tuples in the database from 5000 to 40000 and initialize the size of the hash table to the total number of tuples. Fig. 3 plots the bar graph between the number of tuples and the construction time of the cuckoo filter *CF*. It can be found that the construction time increases as the total number of tuples increases. If you insert 4000 items into the hash table, the cost of construction time is 566.7 milliseconds, which is a very small overhead.

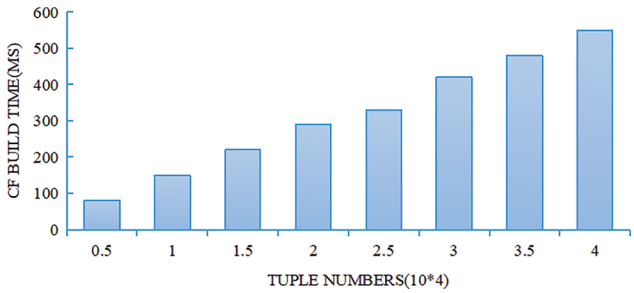

Figure 3: The building time of *CF*

7.2 Cost of ACC Building Time

In the experiment, a timer is set to obtain the time for constructing the accumulator *ACC* for each different attribute, and then the average value is taken. Fig. 4 plots the bar graph between the number of tuples and the build time of the accumulator *ACC*. It can be found that the construction time increases as the total number of tuples increases.

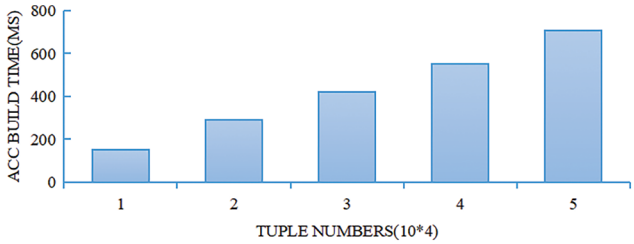

Figure 4: The building time of *ACC*

7.3 Cost of Communication

The communication cost in the data retrieval phase includes two parts: trapdoor cost and retrieval result cost. The cost of the trapdoor depends on the number of attributes that should be returned, namely $|J|$. Set the number of attributes that should be returned to $|J| = 5, 10, 15$. In order to evaluate the cost of the query result, the number of tuples is changed from 1000 to 5000; the size of the hash table is also initialized to the total number of tuples. Finally, a bar graph between the number of tuples and the size of the query result is drawn and grouped according to the number of returned attributes. Fig. 5 shows the experimental results of the communication payload size, and compared with Wang's scheme, the communication cost of our proposed scheme is lower.

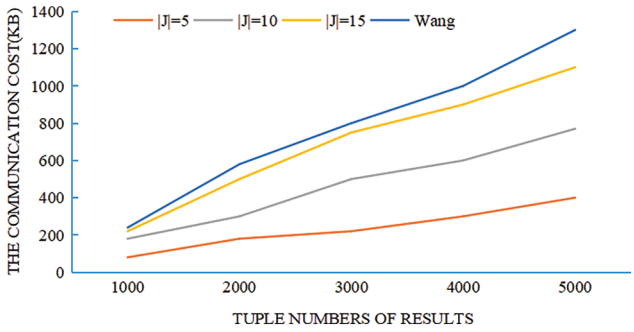

Figure 5: The size of communication payload

7.4 Cost of Integrity Verifying

Integrity verification consists of two parts: correctness and completeness. In the experiment, a timer is set to obtain the time overhead of these two parts and add them up as the time for integrity verification. Change the number of tuples from 1000 to 5000 and set the number of attributes that should be returned to $|J| = 5, 10, 15$. The size of the hash table is also initialized to the total number of tuples. Finally, a bar graph between the number of tuples and the integrity verification time is drawn, grouped according to the number of returned attributes. Fig. 6 shows the experimental results of the integrity verification. It can be found that the time cost of the proposed scheme increases linearly with the number of returned attributes, and the cost is lower than that of Wang's scheme.

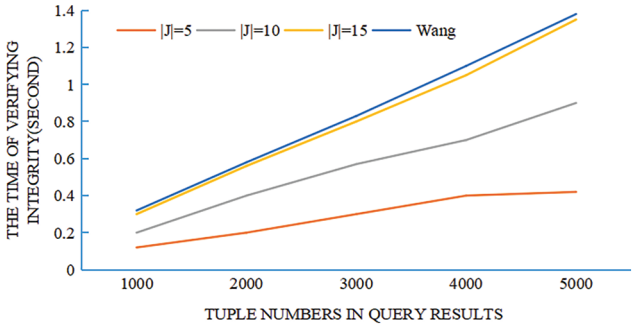

Figure 6: The time of integrity verifying

8 Conclusion

Aiming at the problem of high communication overhead and low efficiency in multi-keyword query integrity verification of outsourced data, this paper proposes a Lightweight and Verifiable Multi-keyword Query Scheme, which does not require a third-party auditor and chooses a hash-based accumulator and combined with a cuckoo filter to construct a provably secure composite verification structure. A cuckoo filter is constructed for different attribute columns to achieve integrity verification and a hash-based accumulator is constructed to achieve correctness verification. A controllable Paillier encryption is also proposed to support dynamic data update. This method strikes a balance between the cost of verification structure generation, verification object generation and proof verification cost. In addition, we also theoretically prove the security and performance of the scheme. Finally, our experimental results demonstrate the effectiveness and efficiency of our scheme.

Funding Statement: This work was supported by the National Key R&D Program of China (Grant No. 2021YFB3101104), Zhaokang Wang received the grant and the URLs to sponsors' websites is https://service.most.gov.cn/. This work was supported by the National Natural Science Foundation of China (Grant No. 62072249), Yongjun Ren received the grant and the URLs to sponsors' websites is https://www.nsfc.gov.cn/.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study. Xiaowan Wang and Yongjun Ren are the co-corresponding authors.

References

[1] Y. J. Ren, J. Qi, Y. P. Cheng, J. Wang and O. Alfarraj, "Digital continuity guarantee approach of electronic record based on data quality theory," *Computers, Materials & Continua*, vol. 63, no. 3, pp. 1471–1483, 2020.

[2] C. P. Ge, W. Susilo, Z. Liu, J. Y. Xia, L. M. Fang *et al.,* "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2787–2800, 2021.

[3] L. M. Fang, M. H. Li, Z. Liu, C. T. Lin, S. L. Ji *et al.,* "A secure and authenticated mobile payment protocol against off-site attack strategy," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 1–12, 2021.

[4] S. R. Ausekar and S. K. Pasupuleti, "Dynamic verifiable outsourced database with freshness in cloud computing," *Procedia Computer Science*, vol. 143, pp. 367–377, 2018.

[5] T. Li, N. P. Li, Q. Qian, W. Xu, Y. Ren *et al.,* "Inversion of temperature and humidity profile of microwave radiometer based on bp network," *Intelligent Automation & Soft Computing*, vol. 29, no. 3, pp. 741–755, 2021.

[6] Y. J. Ren, Y. Leng, J. Qi, K. S. Pradip, J. Wang *et al.,* "Multiple cloud storage mechanism based on blockchain in smart homes," *Future Generation Computer Systems*, vol. 115, pp. 304–313, 2021.

[7] J. Y. Hu, K. L. Li, C. B. Liu and K. Q. Li, "A game-based price bidding algorithm for multi-attribute cloud resource provision," *IEEE Transactions on Services Computing*, vol. 21, no. 8, pp. 345–356, 2021.

[8] C. B. Liu, K. L. Li, K. Q. Li and R. Buyya, "A new service mechanism for profit optimizations of a cloud provider and its users," *IEEE Transactions on Cloud Computing*, vol. 17, no. 4, pp. 156–168, 2017.

[9] Y. J. Ren, Y. Leng, Y. P. Cheng and J. Wang, "Secure data storage based on blockchain and coding in edge computing," *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 1874–1892, 2019.

[10] J. Wang, H. Han, H. Li, S. M. He, P. K. Sharma *et al.,* "Multiple strategies differential privacy on sparse tensor factorization for network traffic analysis in 5G," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1939–1948, 2022.

[11] Y. J. Ren, F. J. Zhu, S. P. Kumar, T. Wang, J. Wang *et al.,* "Data query mechanism based on hash computing power of blockchain in internet of things," *Sensors*, vol. 20, no. 1, pp. 1–22, 2020.

[12] X. Zhou, K. L. Li, Y. T. Zhou and K. Q. Li, "Adaptive processing for distributed skyline queries over uncertain data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 28, no. 2, pp. 371–384, 2016.

[13] Y. J. Ren, F. Zhu, J. Wang, P. Sharma and U. Ghosh, "Novel vote scheme for decision-making feedback based on blockchain in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1639–1648, 2022.

[14] C. P. Ge, W. Susilo, J. Baek, Z. Liu, J. Y. Xia *et al.,* "Revocable attribute-based encryption with data integrity in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, pp. 1, 2021.

[15] J. Wang, C. Y. Jin, Q. Tang, N. X. Xiong and G. Srivastava, "Intelligent ubiquitous network accessibility for wireless-powered MEC in UAV-assisted B5G," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 2801–2813, 2021.

[16] C. P. Ge, W. Susilo, J. Baek, Z. Liu, J. Y. Xia *et al.,* "A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 7, pp. 1–12, 2021.

[17] L. Ren, J. Hu, M. Li, L. Zhang and J. Xia, "Structured graded lung rehabilitation for children with mechanical ventilation," *Computer Systems Science & Engineering*, vol. 40, no. 1, pp. 139–150, 2022.

[18] C. P. Ge, Z. Liu, J. Y. Xia and L. M. Fang, "Revocable identity-based broadcast proxy re-encryption for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1214–1226, 2021.

[19] Z. Fan, Y. Peng, B. Choi, J. Xu and S. S. Bhowmick, "Towards efficient authenticated subgraph query service in outsourced graph databases," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 696–713, 2014.

[20] J. Wang and X. Chen, "Efficient and secure storage for outsourced data: A survey," *Data Science and Engineering*, vol. 1, no. 3, pp. 178–188, 2016.

[21] T. Xiang, X. Li, F. Chen, Y. Yang and S. Zhang, "Achieving verifiable, dynamic and efficient auditing for outsourced database in cloud," *Journal of Parallel and Distributed Computing*, vol. 112, pp. 97–107, 2018.

[22] J. F. Wang, X. F. Chen, X. Y. Huang, I. You and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3293–3303, 2015.

[23] J. F. Wang, X. F. Chen, J. Li, J. L. Zhao and J. Shen, "Towards achieving flexible and verifiable search for outsourced database in cloud computing," *Future Generation Computer Systems*, vol. 67, pp. 266–275, 2017.

[24] S. Hraiz, G. Al-Naymat and A. Awajan, "A novel method to verify the search results of database queries on cloud computing," in *2020 21st Int. Arab Conf. on Information Technology (ACIT)*, 6th of October City, Cairo, Egypt, pp. 1–7, 2020.

[25] M. Rady, T. Abdelkader and R. Ismail, "Integrity and confidentiality in cloud outsourced data," *Ain Shams Engineering Journal*, vol. 10, no. 2, pp. 275–285, 2019.

[26] H. Yin, Z. Qin, J. Zhang, L. Ou and K. Li, "Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 27–39, 2017.

[27] Y. J. Ren, K. Zhu, Y. Q. Gao, J. Y. Xia, S. Zhou *et al.,* "Long-term preservation of electronic record based on digital continuity in smart cities," *Computers, Materials & Continua*, vol. 66, no. 3, pp. 3271–3287, 2021.

[28] B. Zhang, B. Dong and W. H. Wang, "Integrity authentication for sql query evaluation on outsourced databases: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 287–298, 2019.

[29] M. Zhang, C. Hong and C. Chen, "Server transparent query authentication of outsourced database," *Journal of Computer Research and Development*, vol. 47, no. 1, pp. 182–190, 2010.

[30] Y. J. Ren, J. Qi, Y. P. Liu, J. Wang and G. Kim, "Integrity verification mechanism of sensor data based on bilinear map accumulator," *ACM Transactions on Internet Technology*, vol. 21, no. 1, pp. 1–20, 2021.

[31] X. R. Zhang, W. F. Zhang, W. Sun, X. M. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.

[32] P. Devanbu, M. Gertz, C. Martel and S. G. Stubblebine, "Authentic data publication over the internet," *Journal of Computer Security*, vol. 11, no. 3, pp. 291–314, 2003.

[33] X. R. Zhang, X. Sun, W. Sun, T. Xu and P. P. Wang, "Deformation expression of soft tissue based on BP neural network," *Intelligent Automation & Soft Computing*, vol. 32, no. 2, pp. 1041–1053, 2022.

[34] B. Fan, D. G. Andersen, M. Kaminsky and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proc. of the 10th ACM Int. on Conf. on Emerging Networking Experiments and Technologies*, New York, NY, USA, vol. 14, no. 2, pp. 75–88, 2014.