VOL 3 (2019) NO 3

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Achieving Lightweight Verifiable Privacy Preserving Search Over Encrypted Data

Selasi Kwame Ocansey[#], Charles Fynn Oduro[*]

[#] *School of Computer Science and Communication Engineering, Jiangsu University, P.R China*

[*] *Ghana National Petroleum Cooperation, Tema, Ghana*
E-mail: mrocansey@msn.com, fynncharles@gmail.com

***Abstract*— When cloud clients outsource their database to the cloud, they entrust management operations to a cloud service provider who is expected to answer the client's queries on the cloud where database is located. Efficient techniques can ensure critical requirements for outsourced data's integrity and authenticity. A lightweight privacy preserving verifiable scheme for outsourcing database securely is proposed, our scheme encrypts data before outsourcing and returned query results are verified with parameters of correctness and completeness. Our scheme is projected on lightweight homomorphic encryption technique and bloom filter which are efficiently authenticated to guarantee the outsourced database's integrity, authenticity, and confidentiality. An ordering challenge technique is proposed for verifying top-k query results. We conclude by detailing our analysis of security proofs, privacy, verifiability and the performance efficiency of our scheme. Our proposed scheme's proof and evaluation analysis show its security and efficiency for practical deployment. We also evaluate our scheme's performances over two UCI data sets.**

***Keywords*— Verifiable Search, Outsourced Database, Cloud Computing, Privacy Preserving.**

I. INTRODUCTION

Computer systems and its networks' popularity and prevalence have led to the voluminous amount of data being generated by companies, individuals, and organizations. These generated data require protection but the companies lack the necessary resources. A significant method for solving this is by outsourcing the management and storage of the generated data, which relieves companies and individuals from difficulties encountered during data protection. The outsourcing of the data is done on the cloud to benefit the client C in terms of cost saving, high-quality on-demand services, effective computational performances, and quick deployment. The cloud which is semi-trusted is not completely trusted by its C due to a lot of factors. This inhibits C from outsourcing their sensitive data in plaintext form to the cloud. The best solution for these reluctant C is to encrypt their plaintext using traditional encryption techniques before outsourcing.

Security in data outsourcing involves its privacy and integrity [1]. In protecting the privacy of data, cloud service providers CSPs can be ensured by traditional encryption schemes to infer nothing about the data outsourced. Also, data integrity consists of two parts, query integrity, and storage integrity. When returned query results from cloud server CS are able to be verified for correctness and completeness it is referred to as Query integrity [2]. With Storage integrity, data outsourced can be checked to verify which ones are lost or erroneous without recovery, wherein solving this challenge many existing mechanisms such as proofs of retrievability (POR) and provable data possession have been proposed. Correctness here means that the returned result's validity must be able to be verified by its data owner. In terms of the completeness, there should be the inclusion of all valid records from the results query satisfying a set of conditions.

In this era of big data, complex computations cannot be afforded by its Cs due to their constrained resources [3]. This makes computing securely on outsourced data very expensive hence it becomes an obstacle. In relation to storage, every conventional encryption scheme should be able to protect malicious [4] CSPs from inferring on the stored data. In addressing these data privacy and integrity issues, we propose in this chapter a lightweight verifiable searchable scheme which can efficiently verify query result's correctness and completeness and support for resource-constrained devices such as mobile phones. Also, since top-k query results retrieval is a generally used procedure, we propose in furtherance a verification scheme using a random challenge technique to ascertain top-k

keyword query results. Hence, cloud Cs will be able to identify incorrect top-k results with high probability WHP, a probability close to 1.

We structure our scheme using the data structures BF and Embedded Merkle B-tree (EMB-tree) as well as homomorphic encryption. We summarize our contributions as follows:

- A verifiable scheme for outsourced cloud data that achieves integrity and confidentiality.
- Lightweight homomorphic encryption (LHE) is used to encrypt data outsourced that permits the performance of efficient query operations.

Our proposed random challenge technique helps to verify a given query's top-k search result. The C is assured of the top-k result's authenticity WHP. Furthermore, BF and EMB-tree are used to verify and authenticate the search results with correctness and completeness functionalities.

Organization of Paper: The Section 2 of our paper formulates the problems in data outsourcing in our context. Our scheme is presented in Section 3. Our paper's security analysis and discussion are provided in Section 4. Section 5 and 6 evaluates our scheme's performance comparison and evaluation respectively. Review of related work is done in Section 7 and the paper is concluded in Section 8**.**

II. System Overview

This section proposes a lightweight verifiable scheme for outsourced cloud data. The scheme is projected on a lightweight homomorphic encryption technique and BF which are authenticated to assure the outsourced data's authenticity, confidentiality, and integrity. Such a projection is also verifying the correction and completeness of returned query results. The system involves two entities, CS and the C. The C generates the secure index after which its outsourced to CS along with the encrypted data.

We adopt a semi-trusted model where incorrect results may be returned by CS due to minimizing computational cost or system errors. More specifically, CS may complete only partial computational task to minimize the entire cost and as a result, an incorrect result may be returned to C. In our scheme, the C can function as the data owner and data user concurrently and is assumed in the entire process to be always honest. There can also be insecurity in the channel of communication between CS and C. In our scheme, there should be the order of the query results by CS per ranking conditions, which returns the most significant queried results to C [5]. Our scheme for decreasing unwanted communication can request from CS the most important documents termed top-k queries.

III. Lightweight Verifiable Search Scheme

Our verifiable scheme searches on outsourced data and ensures that returned query results are correct and complete in semi-trusted *CS* environments. This is important especially since *CS* can partially return query results for their own benefit. Our main concept leverages BF and EMB-tree to verify query result's correctness and completeness. Additionally, we use LHE to encrypt outsourced data since it provides efficient performance of query operations. Fig 1 shows our proposed scheme's architecture with its five phases.

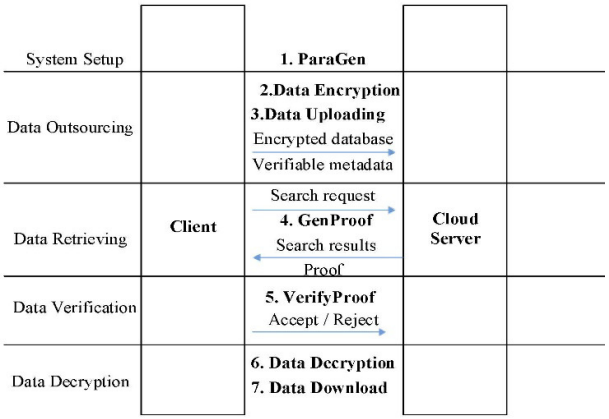

Fig 1. The framework of our scheme

- The first phase is the system setup phase, the *C* generates the encryption keys and BFs. The rest of the phases use the generated keys and BFs.
- In the second phase, data outsourcing takes place, the data are encrypted by the *C* and signatures assigned to them before outsourcing.
- The third phase is for data retrieval, where query results are submitted upon request by the authenticated *C* to *CS*. The query in the encrypted outsourced data are executed by *CS* using the GenProof. The GenProof algorithm is used by *CS* to generate query result's integrity proof and returned to *C*.
- In Data verification phase, the VerifyProof checks the query result's validity. If it is valid, the Accept is returned else Reject is outputted.
- The final phase is the data decryption phase, where upon query result's correctness, *C* decrypts encrypted data.

A. System Setup

The parameters our scheme requires are setup in this phase. The C generates the encryption key K for data d, thus $K(d)$ for three components of a tuple in an outsourced data collection $K(d) = (\beta, \theta, \rho)$. β represents a list of $[(h_1, j_1, m_1) \ldots, (h_d, j_d, m_d)]$ where h_i, j_i and m_i are random real numbers. In ensuring the correct workings of the scheme we have to be certain that $d \geq 4$, $h_i \neq 0$ for $1 \leq i \leq d-1$, $h_d + j_d + m_d \neq 0$ and $m_i \neq 0$ and $1 \leq i \leq d-1$. The component ρ is defined as a list $\rho = (\rho_1, \rho_2, \rho_3)$ where ρ_x is a d dimensional vector $(\rho_{x_1}, \ldots, \rho_{x_d})$ for $1 \leq x \leq 3$. θ represents the two random pairs of real numbers θ_1 and θ_2.

- Uniformly sample d random real numbers $r_{x_1}, \ldots\ldots, r_{x_d}$. These real numbers can be infinite.
- The following parameters are computed
  - $\rho_{x_1} = h_1 * m_1 * \theta_x + j_1 * r_{x_d} + h_1 * (r_{x_1} - r_{x(d-1)})$ (1)
  - $\rho_{x_i} = h_1 * m_1 * \theta_x + j_1 * r_{x_d} + h_1 * (r_{x_i} - r_{x(d-1)})$ for $2 \leq i \leq d-1$ (2)
  - $\rho_{x_d} = (h_d + j_d + m_d) * r_{x_d}$ (3)

After the generation of K, an initializing of a BF with h hash functions begins. After the initialization, all the outsourced data's attributes values are inserted into BF.

B. Data Outsourcing Phase

Client inserts verification metadata into the data collection that's encrypted before they are outsourced with its corresponding index to the cloud. Supposing the *C* wants to outsource a relational collection of their data $\mathcal{D} = (\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n)$ to *CS*, where $\mathcal{E}_j (1 \leq j \leq n)$ refers to the j^{th} attribute. The index I_d for the whole number of tuples is generated and denoted by n_i, in which the attribute column's value V_i is equivalent to v_i. For each v_i we compute its corresponding hash value as $h(v_i || n_i)$. An EMB-tree is constructed based on the data collection's hash values. Using an LHE scheme the *C* encrypts every element v_i as $l_i = Enc(K(d), v_i)$ and uploads the value which is encrypted with their index for every tuple t as :

$$t_e = (I_1, l_i, h(v_1 || n_1), \ldots, (I_n, l_n, h(v_n || n_n)) \quad (4)$$

Besides the *C* uploading encrypted data, the tree's root signature $Sign$, EMB-tree's $h(r)$ and their corresponding BF are also uploaded.

C. Data Retrieving Phase

In this phase, the query is sent by a legitimate user and the corresponding collection of data from the stored cloud data then returned correspondingly. This can be explained with an example using a SELECT query, viz.,

SELECT * from Tuple where l= v

where v represents values in the encrypted form. The *C* generates the query with all the data records in the encrypted form as shown in the SELECT example above with its data field value encrypted. *CS* receives the query and checks each value. The query returns the equivalent result with its authentication data structures and index. The example using the SELECT query checks entries which match the results The *C* receives query results and uses sign and EMB-tree's $h(r)$ to check its validity.

D. Verification Phase

In this phase, C verifies the query result's integrity. We measure integrity in our scheme by correctness and completeness. Once C receives the query results VerifyProof is used to check the result's integrity.

Correctness: The result must be guaranteed that it hasn't been altered at any point by *CS*. The BF can be used by the *C* to verify the correctness of query results. With EMB-tree *C* performs correctness check if the set of query result contains some data collection tuples. The *C* then checks if any tuple in the search result is genuine, if they are all are genuine the *C* proceeds to verifying the completeness of the result.

Completeness: After the verification of the returned query results by *C* through *CS* the completeness need to be assured. In our scheme completeness can be protected through EMB-tree, thus, by checking the values nearby and verifying whether the query result that *CS* returns is complete or not.

E. Decryption Phase

The C decrypts the data if after successful verification the returned query results are genuine as follows:

The decryption algorithm denoted by $Dec(K(d), (c_1, \ldots, c_d))$ gives the plaintext v for the given ciphertext in the following steps:

i. $Tuple = \sum_{i=1}^{d-1} m_i$
ii. $S = c_d / (h_d + j_d + m_d)$
iii. v= $(\sum_{i=1}^{d-1} (c_i - S * s_i))/\text{Tuple}$

The conditions $d \geq 4, h_i \neq 0 \; for \; 1 \leq i \leq d-1, h_d + j_d + m_d \neq 0$ and $m_i \neq 0 \; for \; 1 \leq i \leq d-1$ validates the decryption scheme above.

F. Top-k Verification

The interest of *C* is to receive k the most significant query results for her own computational benefits such as saving communication cost [5]. The scheme attains for each similarity score a verifiability excluding the $top-k$ query results. Incorrect $top-k$ results might be returned by *CS* to either save computational cost or energy. An instance is when there is a partial search by *CS* over the data set and then the $top-k$ query results being returned.

i. Achieving Random Challenge

In achieving the verifiability for top-k results the *C*'s randomly selected documents' similarity scores are verified. However, if the scores of the selected documents are smaller than the *top-k* query results that *CS* returns, the question then is: will the *C* be guaranteed that the *k*'s query results is the accurate *top-k* for a given probability?

Assuming C outsources N documents to CS, C is expected to receive the top-k query results having the top scores in comparison to a search Q for some k≪N. M documents are chosen by C as well as requesting for CS's similarity scores. The probability of identifying portions ρ of the top-k results that CS fails to return is computed as following:

$$P_{detect} = 1 - (1 - \frac{\rho k}{N - k})^{\mathcal{M}}$$

When $\rho k \ll N - k$, the *identifying* probability is near to $\frac{\rho k.\mathcal{M}}{N}$. Using k, N and ρ resolution among the probability of identifying P_{detect}. We show in Fig. 2 the P_{detect} relationship with k of $top\ \ k$ retrieval for N=10,000 and $\mathcal{M}$=100. Fig 2 shows that the increase of P_{detect} as k query results number increases. For small k it is possible *CS*'s mischievousness is undetected.

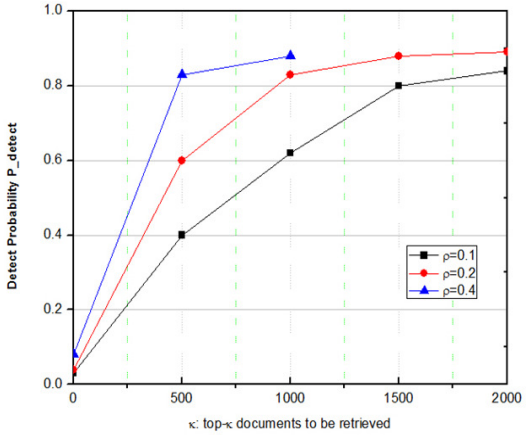

Fig 2. P_{detect} for $\mathcal{M}$=100 and N =10,000

IV. Security Analysis

In this section, the security proofs of the proposed scheme are discussed. Theorems of integrity and confidentiality are proofed as well as the verifiable security also discussed and it has been found that inference or attacks are well protected as explained below.

A. *Integrity*

The integrity of the data as a parameter guarantees that altering or modifying of the data is done solely by authorized *C* (data owner or users). There is the need to prevent *CS* from altering the collection of data records in the cloud computing scenario. A malicious *CS* might in small portions delete the data stored so as to save its storage or computation.

Theorem 1. The proposed scheme achieves the integrity of query results.

Proof. It involves verifying the query result's correctness and completeness. If there exists in the query results some tuples, *C* checks the ciphertext $(c_1, \ldots, c_d)$ in the result tuple's correctness. It will be extremely difficult for an attacker without the encryption key to falsify a ciphertext result which is valid. This is also due to the securing of an encryption scheme against ciphertext-only attacks. It is easier to verify the result's completeness once the correctness has been deduced because of the quantity of tuples n_i is involved in the hash value $h(v_i||n_i)$. With the EMB-tree's help, every tuple can be proven to satisfy query conditions.

B. *Verifiability*

The parameter of verifiability which is achieved by our scheme is important to influence a semi-trusted *CS*, to be honest. For semi-trusted *CS* providers, they try to decrease energy cost while revenue is maximized. Without the keyword query algorithm having a verification mechanism, an incorrect result may be returned by a *CS* provider.

***Top-k* Verifiability**. With the random challenge technique with ordering, the P_{detect} can reach close to 1 (WHP) for just tens of challenges, independent of k and the number of documents.

C. *Confidentiality*

In any outsourced data setup, one of the most important parameters is confidentiality, and must always be ensured. Most outsourced cloud data are very sensitive and if leaked can cause identity theft on a large scale. Confidentiality in our context means that CSP cannot obtain plaintext information nor the encryption key for computations and return the query results without knowing the database's content. This makes confidentiality a vital metric in assessing any paradigm for outsourcing data.

Theorem 2 Our proposed scheme is secure on parameters of data confidentiality

Proof. Our proposed scheme uses homomorphic encryption thereby securing it from cipher text-only attacks. The $f_k(.)$ is a pseudorandom function which generates numbers and has a collision probability which is negligible. Linear combinations of $f_k(i)$ and $f_k(x)$ such as $f_k(i) + f_k(x)$ and $f_k(i) - f_k(x)$ which are all pseudorandom that are difficult to be guessed by any *CS*. Hence, our proposed scheme can be said to achieve the corresponding level of confidentiality if the encryption key is firmly limited to the *C* and authorized users only.

V. Performance Comparison

Table 1 compares our scheme with three previous solutions. It can be seen that ours is the only one to accomplish correctness and completeness without Third Party Auditor (TPA) intermediating. In terms of the computational cost, our scheme is higher in cost in terms of the correctness verification than [6] and [1] since it requires two $\mathcal{P}$ operations; in the aspect of completeness verification our scheme is lower than [2]. Additionally, our scheme partially supports partial attribute retrieval, it saves considerable communication cost than the others. In real-world applications our scheme is more appropriate considering efficiency and functionality.

TABLE I
Comparison Of SSE Schemes

Scheme/Property	[2]	[6]	[1]	Our scheme
Correctness	✓	✓	✓	✓
Completeness	✓	✓	✓	✓
Communication	$\geq m$	$N_i \log(n)$	$\geq m$	$\leq n$
Correctness verification	$1\mathcal{D} + 1\mathcal{S}$	$3\mathcal{S}$	$1\mathcal{D} + 1\mathcal{S}$	$2\mathcal{P}$
Completeness verification	KH	$3\mathcal{S}$	KH	$1\mathcal{D} + 2H$
Outsourced database type	Ciphertext	Plaintext	Ciphertext	Ciphertext
Without TPA	×	×	×	✓

VI. PERFROMANCE EVALUATION

We evaluate our proposed scheme's performance in this section. We conduct the simulation on a computer with 2.2GHz, Intel i5 CPU, and 16GB RAM. Our scheme's performance is evaluated on University of California, Irvine (UCI) [7] data sets as shown in Table 2. We divide our results into system setup, data outsourcing, integrity verification, and data retrieval. We also compare our work with the most related work [2] from two aspects: communication cost and integrity verification cost.

TABLE II
UCI BAG-OF-WORDS DATASETS

Name of Data set	Short Name	Number of Documents	Number of Keywords
NY Times news articles	NY Times	200,000	85,000
Enron Emails	Enron	35,660	26,654

A. *System Setup*

It consists of the BF setup and key generation. In this phase, an important fragment of the computational overhead is the calculation time for the hash functions. In our simulation, we have fixed the value k=16 and the bit-array size varied to have the ratio to the tuples number at constant 48. The false probability ($P_f \leq 2^{-20}$) is always maintained to be negligible. The numbers of tuples have been varied from 2,000 to 10,000 for the results. The number of tuples and the computation cost is observed to be linear at all times. The results showed that the setup is acceptable, since for as large as 10,000 tuples, we make it below 400 ms.

B. *Data Outsourcing*

For data outsourcing, every created EMB-Tree and tuple should be encrypted by the *C*. Also, the EMB-Tree can reduce the number of hashes but the signature of each verification object needs to be completed which accounts for n signatures. Hence, the outsourcing phase becomes slow compared with that of retrieval, it should be noted that it's a process which is one-off and can be accepted. It can be seen from our graph in Fig. 3 that we managed to keep the cost less than 35s for 10,000 tuples, which is quite decent.

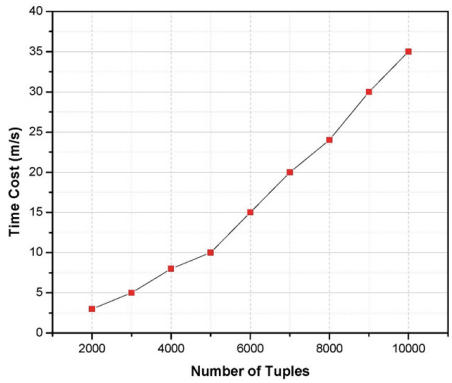

Fig. 3 Time cost for data outsourcing

C. *Integrity Verification*

The verifiability of our scheme is vital to influencing semi-trusted *CS* operating honestly. *CS* can decide to return incorrect results if no verification mechanism exists in the algorithms for keyword query search. Verification consists of two aspects: Completeness and correctness. In our simulation, timers are set for obtaining the time cost of the two parts and sum all as the integrity verifying time. We vary the number of tuple in query results from 1000 to 5000; we set the number of attributes that should be returned as |J|= 2, 5,10,15. The hash table's size is initialized to the total number of tuples stated above. We plot the bar chart between the integrity verification time and the number of tuples, grouped by the number of returned attributes. Fig. 4 shows the integrity verification results. It can be deduced that the time cost of our proposal and [2] increases linearly with the number of attributes returned, but our scheme has lower cost.

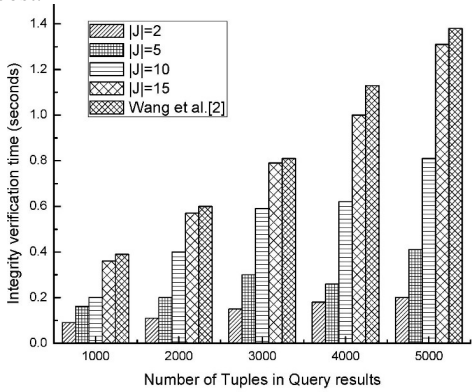

Fig. 4 Integrity Verification Time

D. *Communication Cost*

The communication cost in aspect of data retrieval is evaluated in this section. The communication cost is categorized into two parts: the cost of query result and the cost of trapdoor. The cost of the scheme's trapdoor depends on the number of attributes that should be returned, namely |J|. The number of attributes that should be returned is set as |J|= 2, 5, 10, 15. The size of our scheme's trapdoor are about 0.85 KB when |J|= 5 and 0.37 KB when |J|= 2. In [2] the trapdoor size depends on the attribution number of tables, and per our experiment is about 1.8KB when the attribution number is about 15. Also, in evaluating the query result's cost, we vary the tuple numbers in query result from 1000 to 5000; we also initialize the size of the hash tables to the total number of tuples stated earlier. We then plot the bar chart between the sizes of the query results and the tuple numbers, grouped by the number of returned attributes. The results are demonstrated in Fig.5 and it shows our scheme's communication cost lower than that of [2].

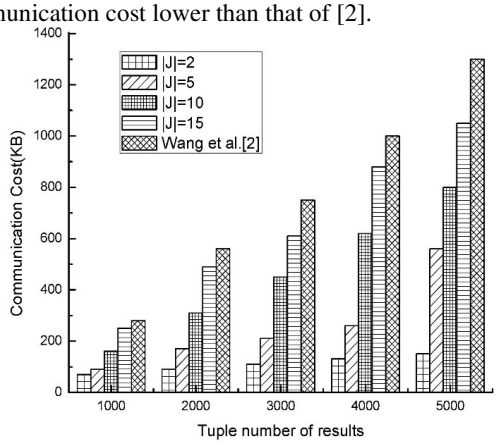

Fig. 5 Communication Size

VII. Related Works

To accomplish an outsourced data which is verifiable, some conventional cryptographic schemes can be leveraged to guarantee data's integrity. These cryptographic primitives are however costly computationally, have extensive loopholes and are not practically deployable. A lot of practical deployable schemes have been in proposed in the past and these solutions can be classified into three (3) according to their approaches.

The initial approach is founded on authenticated data structures such as MHT and EMB-Tree to provide the integrity of query results [2, 8]. The primary idea is for the index to be generated established on MHT for the collection of data and verifying integrity by re-computing the MHT's root signature. The security of MHT is founded on the hash function's collision resistance. In achieving every single tuple's verifiability, a set of log(n) child nodes must be computed to the root and returned to the C, which in computational complexity is extensive. In furtherance, most schemes based on MHT can ensure the completeness of query results.

The second approach is constructed using signature aggregation [9]. Computation and communication overheads for query verification are reduced using this approach. The aggregated signature was extended by Mykletun et al [10] to an unchangeable form, inhibiting the challenger from computing a fresh aggregated signature whether or not there has been some earlier possession of these signatures. Yuan et al. [9] also proposed a new aggregated verifiable scheme for querying data collection outsourced. In [9] an authentication tag based on polynomials of each tuple is assigned. These polynomials can check query result's integrity for definite aggregation queries. However, their scheme did not attain completeness.

Integrity verifications which are probabilistic were used in the final approach [6]. The idea is for the C to insert false tuples from the data collection earlier. However, the false tuples per the idea of this approach are to be known by all of the users. A drawback that may occur is when a dishonest *CS* can connive with any malicious user and then gain information about on the false tuples. Moreover, this third approach requires *CS* to output all the attributes of the tuples and thus cannot support operations on the outsourced data such as projection.

VIII. Conclusions

Cloud computing provides important opportunities for its users to outsource data and applications. However, a major issue in this cloud computing setting is data integrity. In his chapter, we presented a lightweight verifiable search scheme that allows users to securely and efficiently verify the integrity of returned search results. We implemented such a scheme by applying lightweight homomorphic encryption scheme and Embedded Merkle hash B-tree to support the secure search on the cloud. Our scheme is also based on authenticated data structures and also provides a random challenge verification technique which orders *top-k* query results.

The security of our scheme is analyzed and showed to achieve the requirements of efficiency, integrity and verifiability. Furthermore, the performance of the proposed scheme is shown theoretically and the evaluation results demonstrates the efficiency and effectiveness of the scheme

References

[1] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," IEEE transactions on computers, no. 1, pp. 1-1, 2015.

[2] [2] J. Wang, X. Chen, J. Li, J. 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.

[3] [3] Z. Xu et al., "Crowdsourcing based description of urban emergency events using social media big data," IEEE Transactions on Cloud Computing, vol. 10, p. 1109, 2016.

[4] [4] M. Kumar and S. K. Pasupuleti, "Lightweight Verifiable Auditing for Outsourced Database in Cloud Computing," in International Conference on Distributed Computing and Internet Technology, 2018: Springer, pp. 280-287.

[5] [5] Z. Wan and R. H. Deng, "VPSearch: achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," IEEE Transactions on Dependable and Secure Computing, 2016.

[6] [6] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan, "Verifying completeness of relational query results in data publishing," in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005: ACM, pp. 407-418.

[7] [7] M. Lichman, "UCI machine learning repository," ed: Irvine, CA, 2013.

[8] [8] Y. Zhang, J. Katz, and C. Papamanthou, "IntegriDB: Verifiable SQL for outsourced databases," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015: ACM, pp. 1480-1491.

[9] [9] J. Yuan and S. Yu, "Flexible and publicly verifiable aggregation query for outsourced databases in cloud," in Communications and network security (cns), IEEE, pp. 520-524.

[10] [10]E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," ACM Transactions on Storage (TOS), vol. 2, no. 2, pp. 107-138, 2006.