Verifiable Search Over Updatable Encrypted Data in Cloud Computing

Selasi Kwame Ocansey[1], Changda Wang[1*], Wolali Ametepe[1], Qinbao Xu[1], Yu Zeng[1]
[1]School of Computer Science and Communication Engineering
Jiangsu University, Zhenjiang, P.R. China

***Abstract*—With all the benefits from cloud computing, there are negative influences for the data trust and integrity since clients lose control over the outsourced data in clouds. We propose a verification scheme that supports keywords based search among the encrypted data which is updatable. During the verification process the outsourced cloud data are protected from being inferred by the cloud server. Additionally, if the cloud server returns wrong or incomplete search results the clients will be able to detect such failures. A novel concept in our scheme is the ability of clients to update their outsourced data and to ensure the data's correctness. With our scheme, the data's update efficiency is high and the client's computational cost is low, which makes our scheme very suitable for resource constrained devices.**

***Keywords*—Cloud computing; verification; outsourced data; update; correctness; search results**

I. Introduction

Cloud storage is one of the most significant cloud computing techniques which offer elastic storage services in a "Pay-Per-Use" mode. A lot more individuals and companies are outsourcing their data to storage providers such as iCloud, MediaMax, Dropdox and Strongspace to reduce storage cost and management. However, practical cloud storage usage [1] is still faced with privacy and security risks as a consequence of cloud users losing control over their data to the cloud. Encrypting data before outsourcing is the most effective way in guaranteeing confidentiality. Outsourced encrypted data also encounter challenges especially in terms of its searchability which severely has an effect on the retrievability of data. In addressing this concern, a cryptographic primitive known as Searchable Symmetric Encryption (SSE) is proposed. SSE permits a Cloud Storage provider (CSP) to return keyword based queries on the encrypted data without any data information as well as the keywords being learnt. Most SSE schemes [2]-[9] require data owners to build searchable indexes at the setup phase so as to make subsequent keyword queries executable in an efficient way.

Due to the mistrust of cloud server (*CS*), it is important to ensure that the contents of the outsourced database will not be tampered with as well as the operations performed on the database are correct. Additionally, a proof should be provided by the *CS* to protect the outsourced data's modification by unauthorized users. Furthermore, the malicious *CS* should not be able to update (add or delete) the data. Finally, the clients should be able to verify the returned search results.

A. Our Contribution

The focus of our paper is to ensure that outsourced data with updatable functionality is authentic. Our main contributions are as follows:

1) We propose a new verifiable scheme where cloud clients can verify every outsourced data's block. Thus, the client with the help from additional block information can verify the authenticity of the stored data in the database.

2) Our proposed scheme also supports data updating. Since our scheme is dynamic every data block can be updated by the cloud client without the data being revealed by the updating process. In furtherance, clients can also verify how many times data in block position p in the database has been updated.

B. Our Paper's Organization

The paper's remaining sections are organized as follows: In Section 2, related works are explained and the related preliminaries are given in Section 3. The system model is presented in Section 4 and Section 5 introduces our proposed scheme. Evaluation of our scheme's security and performance analysis is elaborated in Section 6. We conclude our paper in Section 7.

II. Related Works

Data outsourcing has the advantage of shifting cloud clients' data management burden to CSPs which are honest-but-curious (HBC) so as to either save bandwidth resources or its computation. This problem from malicious CSPs paves way for security threats. In guaranteeing the privacy and integrity of data, various cryptographic protocols and primitives have attempted to fight this challenge. Cloud clients also have the ability to guarantee the correctness of search results and also detect fatal search operation. Lots of research has been done on outsourced database verification and determining the search result's correctness. Most of these works are based on techniques which are fully homomorphic encrypted such as [10], [11]. These schemes however lacked practicality. Benabbas et al. [12] proposed a verifiable database scheme with update and retrieval queries based on composite order bilinear groups. A vector commitment was used by Catalano et al. [13] to generate a verifiable database with efficient update. The vector commitment is used as the scheme's response proof. Another scheme which is used to authenticate outsourced database's query results using signature with Merkles hash tree was proposed by Ma et al.

[14]. However, schemes based on Merkles hash tree require much information in order to verify results. Zheng et al. [15] proposed a verifiable scheme by utilizing attribute based encryption but this scheme is not feasible and practical in large datasets. After this, Sun et al. [16], [17] proposed a verifiable conjunctive keywords search schemes foe static and dynamic database but the flaw with these schemes is their need for a secure channel form.

III. PRELIMINARIES

This section introduces some preliminaries that our scheme uses. We also provide our verifiable scheme's algorithms and finally describe the polynomials used.

A. Hash Function

A hash function is a compressive primitive algorithm that accepts arbitrary length inputs (block of data) and outputs a fixed size bit string. Hash function inputs are typically called messages and the outputs as message digests. A cryptographic hash function should be able to stand against the known cryptanalytic attacks. A cryptographic hash function has the following properties:

1) It is computationally infeasible to find two different messages x_1 and x_2 that share the same hash h, such that $h(k, x_1) = h(k, x_2)$, where k is the hash key.

2) Given a hash h value y it should be computationally infeasible to find any message x such that $y = h(x)$

3) Given an input x_1, it should be computationally infeasible to find a different input x_2 such that $h(x_1) = h(x_2)$.

3.2 Verifiable outsourced database algorithm

A verifiable outsourced database is made up of the algorithms below:

a) Setup: Given security parameters, this algorithm outputs secret key SK for the cloud's client and for the database the public key PK.

b) Initialization: This enables clients to perform pre-computation on cloud data and generates a plaintext encryption algorithm as well as updating operations verification algorithm.

c) Query: Index i is inputted after being generated by an algorithm after which the CS returns the encrypted data, the information about verification, the updating times counter and the proof.

d) Verify: Using SK the encrypted data and the number of updating times are verified by the client.

e) Update: After updating the encrypted data, the client verifies the CS generated verification information and then updates the proof.

B. Polynomial Computation of Our Scheme

Client wants to compute a high degree polynomial's value at some point a malicious powerful *CS*'s help. It is assumed ∂_w is an encoded input and ∂_k is an encoded output, the polynomial $\mathcal{P}$ is an encrypted function. The polynomial

$$p(w) = a_0 + a_1 w + a_2 w^2 + \cdots + a_n w^n$$

Where $a_i \in \mathbb{Z}_q, 0 \le i \le n$ is a high degree polynomial that will be outsourced to the *CS*. The function on the value w will be computed by the client. A transformed polynomial $\mathcal{P}(\partial_w)$ is constructed for the secure outsourcing and verification.

1) Client-Side Computation:

Client selects randomly $d \in \mathbb{Z}_q, e \in \mathbb{Z}_q, R \in \mathbb{Z}_q, \ell_0 \in \mathbb{Z}_q, \ell_1 \in \mathbb{Z}_q$ and $f \in \mathbb{Z}_q$ and computes:

$$z_0 = e + a_0$$

The transformed polynomial's coefficient

$$\mathcal{P}(\partial_w) = z_0 + z_1 \partial_w + z_2 \partial_w^2 + \cdots + z_n \partial_w^n$$

is then generated as

$$z_i = a_i d^i - f^i$$

where $1 \le i \le n$. $\mathcal{P}(\partial_w)$ will be outsourced to the *CS*.

Client then computes

$$\tau_0 = g^{\ell_0 + R z_0}$$

$$\tau_1 = g^{\ell_0 + \ell_1^i + R z_i}$$

where i =1, 2, …...n

$\mathcal{P}(\partial_w)$ and τ_i will be sent to the *CS*.

If the client wants to compute the value of $p(w)$ at the point w, the client computes $\partial_w = \frac{w}{d}$ and sends ∂_w to the *CS*. The client then computes

$$Q = \prod_{i=0}^{n} t_1^{\partial_w^i} = g^{\ell_0 \frac{1-(\ell_1 \partial_w)^{n+1}}{1-\ell_1 \partial_w}}$$

2) Cloud Server-Side Computation:

CS returns

$\partial_k = \mathcal{P}(\partial_w)$ and

$$T = \prod_{i=0}^{n} t_1^{\partial_w^i}$$

The CS then sends (∂_k, T) to the client.

Client computes

$$Q = \prod_{i=0}^{n} g^{\ell_0 \ell_1^i \partial_w^i} = g^{\ell_0 \frac{1-(\ell_1 \partial_w)^{n+1}}{1-\ell_1 \partial_w}}$$

and verifies whether $T = Q g^{R \partial_k}$ holds . If the equation holds, the client can get the final result by computing

$$k = \partial_k - k_1 \bmod q$$

where

$$k_1 = \sum_{i=1}^{n} f^i \partial_w^i - e$$

IV. System Model

Our system model considers a scenario where in order to guarantee outsourced data's authenticity, data will be encrypted before uploading to the *CS*. The cloud's client can query for their outsourced data anytime and in any location. Due to the *CS* being malicious and untrusted the returned query results needs to be verified for its correctness. Also, outsourced data can be updated anytime by clients and a proof information for the verification generated.

A. Security Model

In our scheme the Cloud Clients are fully trusted since they are the owners of the outsourced data. The *CS* is assumed to be malicious, thus, they may strictly follow predefined protocols but may infer on outsourced data. The *CS* will honestly provide the query results to meet the requirements of the security and should also obtain only encrypted data from the clients.

B. Design Objectives

Our proposed scheme has three (3) main objectives as follows:

a) Privacy preserving: During the updating process, the data's plaintext will not be revealed since it's encrypted before outsourcing to the cloud. Information about stored data should not be obtained by the *CS*.

b) Low computational ***cost***: the cost of computation during the verification phase and the updating process should be low.

c) Verification: Clients should be able to verify the returned result's correctness from a query on stored cloud data. The client can also verify the correctness of the number of the times a stored data is updated.

V. Our Proposed Scheme

We introduce our verifiable scheme in details. It is made up of the following subsections: overview of our proposed scheme and our system's initialization.

A. The Model's Overview

The proposed scheme is based on the verifiable polynomial scheme which will be explained in the next subsection. Our scheme considers encrypted outsourced database (ciphertext) which are in the form $(i, w, \beta, \gamma, proof)$, where i is the index, w is the ciphertext form of the data $\mathcal{W}$, β counts the number of times i that has been updated, γ is the verification information of w which helps generate the $proof$.

When i is queried by the client, the *CS* will return the $(i, w, \beta, \gamma, proof)$ tuple. This protocol's security will guarantee to the client the correctness of w and β. The client will also verify whether w, γ and $proof$ are correct. In updating the index i, the client updates β by setting $\beta' = \beta + 1$. The verification information of $\gamma'$is then verified by the client. If γ' is valid, the updated data $w'$is stored with the new proof $proof'$ on the tuple $(i, w', \beta', \gamma', proof')$. Since data may be updated frequently the existing proof will become non-functional and non-usable by the *CS*.

B. The System's Initialization

The algorithms used in this section have been defined in the subsection 2.3 (Polynomial computation of our scheme) of the paper.

Initialization. Two polynomials $p_1(w)$ and $p_2(w)$ are generated by the clients. $p_1(w)$ is a high degree polynomial

$$p_1(w) = a_0 + a_1 w + a_2 w^2 + \cdots + a_n w^n$$

and $p_2(w)$ is a simple polynomial, so that the client can efficiently compute. In protecting the original data in the database the client selects $d \in \mathbb{Z}_q$ and hides the w which is the original data as $\partial_w = \frac{w}{d}\, d$. The client randomly selects $e \in \mathbb{Z}_q, R \in \mathbb{Z}_q, \ell_0 \in \mathbb{Z}_q, \ell_1 \in \mathbb{Z}_q$ and $f \in \mathbb{Z}_q$, after that $p_1(w)$ into $\mathcal{P}(w)$.

$\mathcal{P}(w) = z_0 + z_1 w + z_2 w^2 + \cdots + z_n w^n$ where

$z_0 = e + a_0$

$z_1 = a_1 d - f^1$

$z_2 = a_1 d^2 - f^2$

$z_n = a_n d^n - f^n$

The client then computes $\tau = (\tau_0, \tau_1, \ldots., \tau_n)$ as the public key

$\tau_0 = g^{\ell_0 + R z_0}$

$\tau_1 = g^{\ell_0 \ell_1 + R z_1}$

$\tau_2 = g^{\ell_0 \ell_1^2 + R z_2}$

$\tau_n = g^{\ell_0 \ell_1^n + R z_n}$

The secret key in this system is

$$SK = (d, g, e, R, \ell_1, f, p_1(w), p_2(w))$$

where g is a generator of $\mathbb{Z}_q$ and the public key $PK = (\mathcal{P}(w), \tau)$

Ciphertext Generation. For each data in the database, the client masks the original data w as $\partial_w = \frac{w}{d}$, after the masking client uploads to the *CS*. When *CS* receives the masked data ∂_w for the first time, it records the data's position index and sets the counter $\beta = 1$, and then computes

$\partial_k = \mathcal{P}(w)$

$$T = \prod_{i=0}^{n} t_j^{\partial_w^j}$$

The *CS* sends ∂_w, ∂_k and T to the client. Client computes $\mathbb{Z}$ with ∂_w.

$$\mathbb{Z} = g^{\ell_0 \frac{1-(\ell_1 \partial_w)^{n+1}}{1-\ell_1 \partial_w}}$$

The equation below is verified by the client to ascertain if it holds $T = \mathbb{Z} g^{R \partial_k}$, Else, client outputs $\perp$. Otherwise, client computes $\gamma = p_2(\partial_k)$, and generates the proof $\mathcal{H} = h(pos || g || \beta)$ where $h(\cdot)$is a non-collision hash function and

pos is the position of the index. The proof $\mathcal{H}$ is sent to the *CS* by the client. $\mathcal{C}$ is the ciphertext that is stored in the outsourced database

$$\mathcal{C} = (i, \partial_w, \partial_k, \beta, T, \mathcal{H})$$

Query. The data in the position i is queried by the client. The *CS* returns to the client the ciphertext $\mathcal{C}_i = (i, \partial_{w_i}, \partial_{k_i}, \beta_i, T_i, \mathcal{H}_i)$.

Verify. Client computes $\mathbb{Z}_i$ with ∂_{w_i},

$$\mathbb{Z}_i = \prod_{j=0}^{n} g^{\ell_0 \ell_1^j \partial_i^j} = g^{\ell_0 \frac{1-(\ell_1 \partial_i)^{n+1}}{1-\ell_1 \partial_i}}$$

And then verifies whether the following equation holds

$$T_i = \mathbb{Z}_i g^{R\partial_{k_i}}$$

If it does not hold, client outputs $\perp$. Otherwise, clients compute $\gamma_i = \mathcal{P}_2(\partial_{k_i})$ and then verifies

$$h(i||g^{\gamma_i}||\beta) \stackrel{?}{=} \mathcal{H}_i$$

If this equation does not hold, client outputs $\perp$.

Update. If the data in position i is to be updated to w_i', the client masks w_i' as $\partial_{w_i'}$ and sends it to the CS.

Cloud server computes

$$\partial_{k_i'} = \mathcal{P}(\partial_{w_i'})$$

$$T_i' = \prod_{j=0}^{n} t_j^{\partial_{w_i'}^j}$$

$\partial_{k_i'}$ and T_i' are returned to the client.

Client then computes $\mathbb{Z}_i'$ and $\partial_{w_i'}$ after which it's verified whether the following equations hold

$$T_i' = \mathbb{Z}_i' g^{R\partial_{k_i}}$$

If it does not hold, client outputs $\perp$. Else, client computes $\gamma_i' = \mathcal{P}_2\left(\partial_{k_i'}\right)$ and generates a new proof

$$\mathcal{H}_i' = h(i \left|\left| g^{\gamma_i'} \right|\right| \beta + 1)$$

and then sends the proof to the *CS*. The *CS* then updates the $\mathcal{H}_i'$ (proof) and sets β. The new ciphertext is

$$\mathcal{C}_i' = (i, \partial_{w_i'}, \partial_{k_i'}, \beta_i', T_i', \mathcal{H}_i')$$

Where $\beta_i' = \beta_i + 1$.

VI. Security Evaluation

This section evaluates our scheme's security with theorems and proofs. We also evaluate our scheme's performance based on the cost of query, verification and update.

A. Security Analysis

Theorem 1. The proposed scheme secures the ciphertext from integrity.

Proof of Theorem 1. A new ciphertext has to be generated by the challenger if the ∂_{w_i} (stored data) is changed to $\partial_{w_i'}$,thus, $\mathcal{C}_i' = (i, \partial_{w_i'}, \partial_{k_i'}, \beta_i', T_i', \mathcal{H}_i')$. The challenger's advantage is defined as:

$$Adv_c = Pr\{\mathcal{C}_i' \; is \; valid\}$$

If the $\mathcal{C}_i'$ (new ciphertext) is valid, the probability is

$$Pr_{\mathcal{C}_i'=1} = Pr\{V\left(\partial_{w_i'}, \partial_{k_i'}, \mathcal{H}_i'\right) = 1\}$$

$$= Pr\{V\left(\partial_{w_i'}, \partial_{k_i'}\right) = 1 \wedge (\mathcal{H}_i') = 1\}$$

$$= Pr\{V\left(\partial_{w_i'}, \partial_{k_i'}\right) = 1\} Pr\{V(\mathcal{H}_i') = 1\}$$

where $V(\cdot)$ is the verification function; $V(\cdot) = 1$ implies that data verification has been passed.

The challenger can generate the valid pair $(\partial_{w_i'}, \partial_{k_i'})$. Thus, $Pr\{V\left(\partial_{w_i'}, \partial_{k_i'}\right) = 1\}$

However, γ_i is generated by ∂_{k_i}

$$\gamma_i = \mathcal{P}_2(\partial_{k_i})$$

$\mathcal{P}_2(w)$ is the secret key and cannot be acquired by the challenger. If ∂_k is changed to $\partial_{k_i'}$, it is difficult to generate γ_i which equals to $\partial_{k_i'}$. Thus, it is hard for the challenger to forge the proof $\mathcal{H}_i'$.

$Pr\{\mathcal{H}_i' = h(i||g^{(\gamma)}||\beta)\}$ is negligible, thus, $Pr\{V(\mathcal{H}_i') = 1\}$ is negligible.

Hence,

$Pr_{\mathcal{C}_i'=1} = P\,r\left\{V\left(\partial_{w_i'}, \partial_{k_i'}\right) = 1\right\} P\,r\{V(\mathcal{H}_i') = 1\}$ is negligible. Thus making

Adv_c negligible. The stored data ∂_{w_i} is therefore unforgeable.

Theorem 2. The update's counter cannot be falsified.

Proof of Theorem 2. If β is changed to β_i', the challenger has to generate $\mathcal{H}_i'$ (new proof) to pass the verification. The proof $\mathcal{H}_i' = h(i||g^{(\gamma)}||\beta_i')$ is generated by γ_i. Nonetheless, the challenger can obtain the index i 's position and the updates counter β from the ciphertext. Though β_i' is generated by ∂_{w_i} , which can be obtained from the ciphertext, the client's secret key is the polynomial $\mathcal{P}_2(w)$. Hence, the challenger cannot obtain β_i' from ∂_{k_i}.

With a given hash it is also impossible to generate a message. Given k, $Pr\{w|h(w) = h(k)\}$ is negligible. Thus, making it difficult to generate a new proof $\mathcal{H}_i' = h(i||g^{(\gamma)}||\beta_i')$.

Theorem 3. Data in a position cannot be substituted with another from a different position.

Proof of Theorem 3. When queries are made for a data in position i, the CS may replace the data in the position i by the data in position j. The ciphertext $\mathcal{C}_i = (i, \partial_{w_i}, \partial_{k_i}, \beta_i, T_i, \mathcal{H}_i)$ will be changed into $\mathcal{C}_i' = (i, \partial_{w_j}, \partial_{k_j}, \beta_j, T_j, \mathcal{H}_j)$.

The $(\partial_{w_i}, \partial_{k_i})$ will pass the verification for the new ciphertext $\mathcal{C}_i'$. The client will then obtain $\beta_i = \wp_2(\partial_{k_j})$. When the client queries the index i, the proof is

$$\mathcal{H}_i' = h\left(i||g^{(\gamma)}||\beta_j\right)$$

$$i = h(i||g)$$

However, when

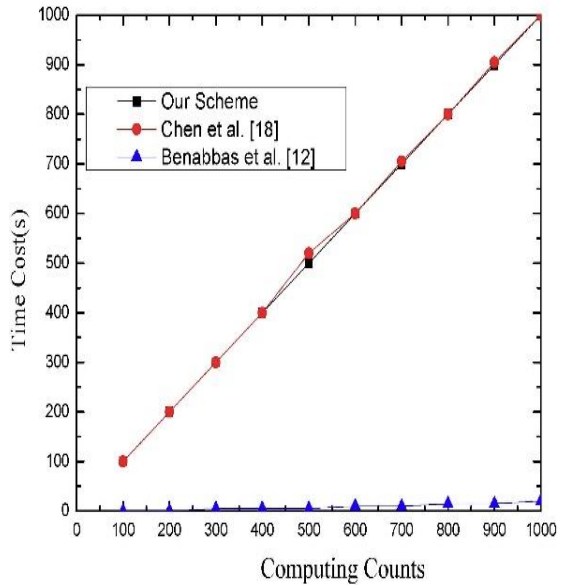
$$\mathcal{H}_j' = h\left(j||g^{(\gamma)}||\beta_j\right)$$

the CS cannot obtain $g^{(\gamma)}$, where the client's secret key is used to generate γ and g is also a secret. Due to the non-collision hash function property it is impossible for $\mathcal{H}_j'$ to generate $\mathcal{H}_i'$. Hence, the conclusion that data in position i cannot be substituted by data in position j.

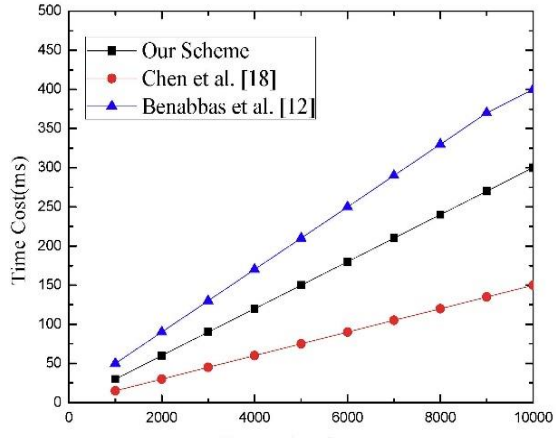
B. Performance Evaluation

We provide a detailed experimental evaluation of our proposed scheme. Our experiments are based on the Pairing based cryptography (PBC) [19] library. We performed 20 runs for each test and the average was taken. All experiments were performed on a computer with 16GB RAM, Intel i7-4600 2.7 GHz CPU with Linux Operating system. This specification of the computer will help to measure precisely both the cloud server and the client server's overhead precisely. We provide in our experiment the time cost simulation for our scheme when n=50 and the related schemes [12], [18]. The time cost in the query, verification and update phases are shown in Fig. 1(a), 1(b) and 1(c), respectively.

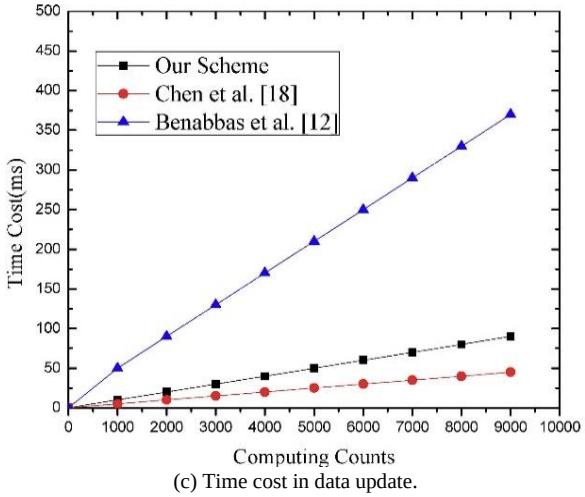
The *CS* on receiving query request from its client performs a search over the indexes and computes on the queried data. Fig. 1(a) shows our scheme's computational overhead and that of [12], [18] is significant and linearly grows with the computing counts. We however argue that the query's computational overhead is mainly performed by the *CS* rather than the resource constrained client. In a real world scenario, the result indicates that the time is acceptable. Data verification's efficiency comparison among the three schemes is shown in Fig. 1(b). It shows that our scheme can achieve data verification with nearly the computation overhead over the scheme [18]. However, when compared with [12] our scheme shows an increase in computational overhead. Our claim is that our scheme can achieve both verifiability and updating functionalities. Notably, in our scheme the data verification's computational cost can be reduced by the client using the secret key. Simulation results shown in Fig. 1(c) show our scheme being more efficient than that of [12] in the update algorithms by the clients. All these make our scheme most suitable for real world applications.



(a)Time cost in data query.



(b)Time cost in data verification.



(c) Time cost in data update.

Fig. 1. Efficiency comparison for data query, verification and update.

VII. Conclusion and Future Work

In this era of Big Data, end users are faced with many challenges. In dealing with these huge amounts of data, end users need high storage capacities and powerful devices which can perform complex computations. The insurgence of cloud computing has provided solutions to these problems. Cloud computing provides services which includes providing clients with huge storage space and also performing powerful computational operations on these stored data. Pertaining to this paper, we construct a new verifiable scheme which supports updated data on the cloud. During the updating process the original data will be protected from malicious adversaries which includes the CSPs. Our scheme's computational cost is low during the verification, update and query phases.

One disadvantage of the proposed scheme is that when clients continuously insert data into the same database index the number of the level in hierarchical commitment increases. The storage and computational overheads of the *CS* will linearly increase thereby reducing the verifiable scheme's efficiency. Future works should thence try to solve the problem of how to construct a verifiable scheme which is efficient and supports updatable operations regardless of the type of insertion.

Acknowledgments

This work was supported by the National Science Foundation of China under grant 61172269 and the Jiangsu Provincial Science and Technology Project under grant BA2015161.

References

[1] Armbrust M., Fox A., Griffith R., Joseph A. D., Katz R., Konwinski A., Lee G.,. Patterson D, Rabkin A., Stoica I., and Zaharia M. A view of cloud computing. Commun. ACM, 53(4):50–58, 2010.

[2] Cash D., Jaeger J., Jarecki S., Jutla C., Krawczyk H., Rosu M.-C., and Steiner M. Dynamic searchable encryption in very large databases: Data structures and implementation. In Network and Distributed System Security Symposium, NDSS'14, 2014.

[3] Cash D., Jarecki S., Jutla C., Krawczyk H., Rosu M.-C, and Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries. In Advances in Cryptology - CRYPTO'13, pages 353–373. 2013.

[4] Curtmola R., Garay J., Kamara S., and Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions. In Proceedings of the 2006 ACM Conference on Computer and Communications Security, CCS'06, pages 79–88, 2006.

[5] Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In Proceedings of the 2013 ACM conference on Computer and Communications Security, CCS'13, pages 875–888, 2013.

[6] Kamara S. and Papamanthou C. Parallel and dynamic searchable symmetric encryption. In Financial Cryptography and Data Security, pages 258–274. 2013.

[7] Kamara S., Papamanthou C., and Roeder T. Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS'12, pages 965–976, 2012

[8] Naveed M., Prabhakaran M., and Gunter C. A. Dynamic searchable encryption via blind storage. In IEEE Symposium on Security and Privacy, SP'14, 2014.

[9] Stefanov E., Papamanthou C., and Shi E. Practical dynamic searchable encryption with small leakage. In Network and Distributed System Security Symposium, NDSS'14, 2014.

[10] Backes M., Fiore D. and Reischuk R.M. Verifiable delegation of computation on outsourced data. In: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4–8, 2013, pp. 863–874, 2013.

[11] Boneh D. and Waters B. Conjunctive, subset, and range queries on encrypted data. In: Theory of Cryptography, Proceedings of the 4th Theory of Cryptography Conference, TCC 2007, pp. 535–554. 2007

[12] Benabbas S., Gennaro R. and Vahlis Y. Verifiable delegation of computation over large datasets. In: Advances in Cryptology -CRYPTO 2011—Proceedings of the 31st Annual Cryptology Conference, pp. 111–131, 2011

[13] Catalano D. and Fiore D. Vector commitments and their applications. In: Public-Key Cryptography—PKC 2013—Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography, pp. 55–72, 2013

[14] Ma D., Deng R.H., Pang H. and Zhou J. Authenticating query results in data publishing. In: Information and Communications Security, Proceedings of the 7th Inter-national Conference, ICICS 2005, pp. 376–388, 2005

[15] Zheng Q., Shouhuai X. and Ateniese G. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: Proceedings 33th IEEE international conference on computer communications, 522-530, 2014

[16] Sun W.H., Wang B., Cao N., Li M., Lou W.J., Hou Y.T. and H. Li. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans Parallel Distrib Syst 25(11):3025–3035, 2014

[17] Sun W.H., Liu X.F., Lou W.J., Hou Y.T. and Li H. Catch you if you lie to me: efficient verifiable conjunctive keywords keyword search over large dynamic encrypted cloud data. In: Proc. 34th IEEE international conference on computer communications, 2015

[18] Chen X., Li J., Huang X., Ma J.and Lou W. New publicly verifiable databases with efficient updates, IEEE Trans. Dependable Secure Comput. 12 (5), 546–556, 2015

[19] Lynn B. The pairing-based cryptography library, http://crypto.stanford.edu/pbc/, 2006