Query Integrity Verification based-on MAC Chain in Cloud Storage

Jun Hong[*]
Software Center, Northeastern University
Shenyang, Liaoning 110819, China[†]
School of Software, North University of China,
Taiyuan,Shanxi 030051,China

Tao Wen
Software Center ,Northeastern University
Shenyang, Liaoning 110819,China
E-mail: wentao@neu.edu.cn
www.neu.edu.cn

Quan Guo
Department of Computer, Science and Technology of Dalian Neusoft University
Dalian, Liaoning, 116023,China
Email:guoquan@neusoft.edu.cn
www.neusoft.edu.cn

Gang Sheng
Software Center, Northeastern University
Shenyang, Liaoning 110819, China[‡]
Email:shenggang@neusoft.edu.cn

Abstract

In order to reduce data maintenance overhead, more and more enterprises outsource their data to the cloud storage, which provides flexible and on-demand storage service for companies. Data outsourcing also brings a lot of security problems, of which query integrity is a critical issue to be resolved. Current research mainly focuses on how to ensure correctness and completeness of the query integrity and pays little attention to freshness verification of query integrity. In this paper, we propose a scheme called MAC Chain for users to verify that their query results are correct, complete and fresh. Experiments show that our scheme does not only have a better performance but also support freshness verification.

Keywords: Data Outsourcing, Cloud Computing, MAC, Digital Signature, MHT, Integrity Verification

[*]School of Software, North University of China, 3 Xueyuan Road, Taiyuan, Shanxi 030051,China

1. Introduction

With the rapid growth of enterprise data, more and more companies outsource their data to the cloud storage to reduce the cost of database maintenance. Cloud computing[1,2] has been envisioned as the next-generation architecture of enterprise IT and has many advantages, for example, it provides flexible and on-demand storage service, fast deployment, scalability, lower costs, etc. However, one obvious defect of data outsourcing is that data is outsourced to the cloud and stored in the semi-trusted cloud server, the data owner (DO) loses control of its own data. Significant security issues[3,4] are associated with this scene, of which query integrity is a key one to be resolved. The integrity of query results in the cloud storage includes three aspects: correctness, completeness and freshness. Correctness means that the query results really originate from the owner's database, and that they have not been tampered with in any way, and no fake data exists in the results; Completeness means that all eligible records are returned to the clients and no one is omitted by the cloud server; Freshness means that each record of the results are the most current version of the DO.

Query integrity verification has great concerns in recent years, Merkle introduced MHT[5](Merkle Hash Tree) which is the first authenticated tree to verify the integrity of query results. authenticated B+ tree[6,7,8,9] is used to replace authenticated binary tree for disk storage and improve I/O efficiency. Digital signature chain[10,11,12] is used to verify the integrity of query results.

Different from digital signature chain, we propose MAC (Message Authentication Code) Chain which is called MACC, a efficient mechanisms, to ensure the correctness and completeness of query results. At the same time, our mechanism ensure the freshness verification to defense the replay attacks from cloud server or attacker. The verification process of our method is implemented on the TTP (Trusted Third Party), and enable the clients only need to receive the final result from TTP, then the clients have no verification cost compared to other methods. Experiments show that MACC is superior to the CES and can effectively defense replay attack.

The rest of this paper is organized as follows. Section 2 introduces the existing work about query integrity. In section 3 we present some preliminary knowledge and definitions. In section 4 we propose our system model. In section 5 we introduce our scheme for query integrity verification in detail. Section 6 shows the experimental results. Finally, section 7 concludes the paper.

2. Related Work

Most of the existing methods for query result verification of outsourced data are divided into two categories. The first one is authenticated data structure based on tree. Merkle[5] introduced MHT which sorts records according to the query attribute and uses binary tree to construct the authenticated structure from bottom to top. Each leaf of the authenticated tree contains the hash of a record and the values of non-leaf nodes correspond to the hash of the concatenation of its two children, that is, a node with children n1 and n2 will be assigned the value $h(n1||n2)$. The hash of each tree node is calculated from bottom to top, finally the hash value of the root node is signed and published. When a client sends a query to the cloud server, the cloud server not only responses with the query results but also the VO (verification object) that contains the hashes stored in the siblings of the path from the leaf to the root of the results. Once receives the results and VO, the client iteratively computes the hashes to construct the hash of the root as described above, finally the client checks whether the hash of root computed locally matches the hash in the published signature of the root. B+ tree[6,7,8] is used to construct the authenticated data structure to support disk storage and improve the I/O efficiency. An authenticated structure based-on B+ tree[9] is adopted for append-only data. The defect of authenticated tree structure is that cost of building, storing and updating complex index is expensive and only supports single index search. When integrity verification is performed on more than one attribute, tree should be built on each searchable dimension for query integrity.

The second one is digital signature. DO signs each tuple before storing it in the cloud server. The server stores the tuple signature along with each tuple. When responding a query, the server simply sends the matching tuples and their signatures to prove integrity and authenticity of the results. Pang[10] adopted digital

signature technique to sign each record using equation 1: $Sign(r_i) = s(h(g(r_{i-1}) \| g(r_i) \| g(r_{i+1})))_{SK}$. (1)
Each record's signature consists of its predecessor, successor record and itself. Correctness is guaranteed by signature and completeness is guaranteed by the signature chain, that is, the query results are complete if and only if when the signatures of query results form a chain. Narasimha[11] proposed DSAC (digital signature chain) for correctness and completeness verification of query results, the signature of a record is composed of its immediate predecessor records of every searchable dimension and itself which is shown as equation 2:
$Sign(r) = h(h(r) \| h(IPR_1(r)) \| ...h(IPR_l(r)))_{SK}$. (2)
The cloud server will return all matching tuples, the boundary tuples and their signatures for correctness and completeness verification. Through this signing structure, this mechanism supported multiple dimension integrity verification. Zhang[12] proposed CES to improve the DSAC and replaced the signature content of a tuple from the concatenation of the precursor records to the concatenation of precursor's primary key to lower the numbers of update. The Signature equation is

$$Sign(t_k) = h(t_k.A_1 \| ... \| t_k.A_m \| t_k.C)_{SK} \quad (3).$$

Where $t_k.A_i$ denotes the i^{th} attribute value of t_k, $t_k.C$ denote the precursor primary vector of m orders.
When receives query request from the clients, the cloud server responses with all matching records, boundary records and their corresponding signature. The clients validate the integrity of query results through their corresponding signatures and signature chain.

There are also other integrity verification methods. Trust-Based fake tuples approach[13] inserted fake tuples into the outsourced relation to ensure the integrity. Sheng[14,15] proposed correctness verifying of inner product of vectors in Cloud Computing.

Most of existing work provides correctness and completeness of query results. Few works provide freshness verification which ensures the clients to identify whether the data from the cloud are up to date and defects the relay attacks from cloud. Li7 provided the freshness verification of query result by defining the valid time of data and published the expired signature list. However, no experimental data are given in this paper. Papadopoulos[16] added timestamp of data into both the root node of authenticated subtree and authenticated tree to ensure the freshness of data. Xie [17] inserted fake records to the real data to realize freshness verification based on the assumption that all the clients are trustful. The clients use function to derive the fake record number between query ranges. However, this is unpractical, the clients may collude with the cloud server and then the server can distinguish the real from the fake and makes this method invalid.

3. Preliminary and Definition

Our method depends on standard security technologies ,that is, One-way hash function[18] and Message authentication codes [19].

3.1. *Hash Function*

A one-way hash function, denoted as $h()$, takes an arbitrary length of input and outputs a fixed length binary sequence. It works in one direction. It is easy to compute a hash value $h(m)$ of any data sequence. However, given a y in the image of h, it is hard to find a message m make $h(m) = y$; given m and hash (m) it is hard to find a message $m' \neq m$ make $h(m) = h(m')$. The most commonly used hash functions are MD5 and SHA-1.

3.2. *Message Authentication Codes*

MAC (Message authentication codes) takes a variable length message as input and the result $h(k,m)$ is a fixed length binary sequence. Given a function $h()$ and a message m, it is hard to determine $h(k,m)$ with a successful probability higher than $1/2^n$, even though a great number of pairs $\{m_i, h(k,m_i)\}$ have been leaked to the opponent. The symmetric key used in $h()$ is kept secret to the attackers.

3.3. *Definitions*

In this paper, we mainly focuses on the relational database, some definitions about it are defined as follows:

Definition 1: relation $R(key, A_1, A_2, \ldots, A_i, \ldots, A_n)$, where A_i denotes the i^{th} attribute and the key denotes the primary key of relation R.

Definition 2: precursor and successor. Relation R can be sorted by any searchable attribute, for example, A_i .

(i) Precursor T_P^i: denotes the T's precursor of the i^{th} attribute and meet the following condition: $T_p.A_i < T.A_i$ and there is no record $T_x \in R$ meets $T_P.A_i < T_x.A_i \le T.A_i$

(ii) Successor T_s^i: denotes the T's successor of the i^{th} order property, coincides with $T.A_i \le T_S.A_i$ and there is no record $T_x \in R$ meets $T_P.A_i \le T_x.A_i < T_S.A_i$.

Definition 3: MAC Chain-MACC

$$MACC(T) = HMAC_{KEY}(h(T) \parallel h(C)) \quad (4),$$

Where $h()$ denotes hash function, "$\parallel$" denotes concatenation,

$$C = hash(T_P^1.key) \parallel \ldots \parallel hash(T_P^l.key) \quad (5)$$

Formula 5 denotes the concatenation of precursor's primary key of each searchable attribute, l is the number of searchable dimensions of the relation, subscript KEY denotes the secret key shared between DO and TTP. If a tuple is the first record of a sorted attribute, then its T_P^i is itself, the last tuple's successor T_s^i is itself as well.

4. System Model

Fig.1 depicts the system model. It consists of four parts: DO (data owner), TTP (trusted third party), cloud service provider and client.

In order to support integrity verification of outsourced data, DO calculate the MAC value of each record according to formula 4 and stores it to the "checksum" field. In order to support dynamic update and freshness verification of outsourced data, DO publishes the FSI (freshness summary information) to the TTP periodically and shares the secret key with TTP to verify the MAC of records from the cloud. The secret key is kept secret from the cloud.

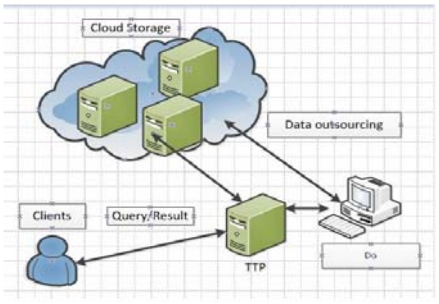

Fig.1. System Model

The cloud server is semi-trusted. The outsourced data in the server may be tampered with or lost. Taking into account the cost, the server may response to the client with a subset of real result set or expired data. In the worst case, the server may be malicious or have been captured by the intruder, and then the server is an opponent of the DO.

TTP is trusted by DO and client, and its security policies can be deployed according to DO's requirements. the main functions of TTP are:

(i) Forwards the query requests from client to cloud server;
(ii) Receives the query result set from the cloud server;
(iii) Receives the FSI from DO;
(iv) Executes integrity verification according to the FSI and MAC in the results.

TTP should be powerful in computation and storage.
The client sends requests to the TTP and receives the correct result from the TTP. There may be a lot of clients sending query requests at the same time. the clients are untrusted to the DO. Therefore, the symmetric key used in MAC is kept secret from the clients.

System assumes that the network channel is secure; the data transmitted in the channel will not be tampered with or lost.

5. MACC Scheme

5.1. *Construction of MACC and FSI*

In order to support query integrity verification, DO need to alter the outsourced relation structure, the modified table structure is shown as follows: $R(A_1, \ldots, A_n, version, precursor, checksum)$, where "version" field stores the update numbers of a record whose value increments one with each update, the default value is zero. "Precursor" field stores the concatenation of precursor's primary key of each searchable attribute. The MAC value of a record calculated based on equation 4 is saved in "checksum" field.

When DO modifies a record, it uploads the updated record to the cloud server immediately for the server to response to the requests timely with the most recent version. However, in order to realize the freshness verification, DO needs to upload FIS to the TTP which includes the primary key, version of each updated record. We adopt periodic updates for reducing the cost of network transmission, every ε time, we uploads the FIS which includes the primary key, version of tuple altered in the last ε time. The shorter the time, the higher the data freshness, the larger the network transmission overhead. Negative number or zero in the version field means the record has been deleted in the DO.

Upon TTP receives the FIS, the corresponding record version information will be updated according to the FIS if the tuple's version saved in the TTP less than the corresponding one of FIS. Due to the network congestion , The difference of version between TTP and received FIS may be more than one. The FIS sent firstly may reach TTP later than the FIS sent after the first.

5.2. *Construction of Query Result Set and VO*

Assume that client sends a query request "select * from employee where $salary \geq 3000$ and $salary \leq 5000$" to the TTP. TTP forwards the request to the cloud immediately. Once receives a query request from the TTP, the cloud server constructs the result set according to the query condition. The construction process of query result set is shown below:

Step 1: according to the query condition "$salary \geq 3000$ and $salary \leq 5000$", cloud finds out the result set T_r consisting of all the records matching the query condition. If there are more than one query condition, the cloud constructs the result set according to the logical "and" or "or" between conditions;

Step 2: cloud finds out the result set T_b consisting of the lower and upper boundary records according query searchable dimension, the boundary records will be returned to the client for completeness verification;

Step 3: collects all the precursors' primary key of records in T_r.

The cloud server returns T_r , T_b and the primary key of each record in T_r collected through step3 to the TTP. The cloud server returns T_b if the query result is empty.

5.3. *Freshness Verification*

Upon TTP receives the response, it firstly executes the freshness verification. Correctness and completeness verification will be followed after the freshness verification. TTP checks the freshness of query results according to the following conditions:

Condition 1:
$if\ FIS(r).version \leq 0$ then
record r has been deleted by DO, where r denotes the record checked currently;
Condition 2:
$if\ Q(r).version \geq FIS(r).version$,then record r is up to date;
Condition 3:
$if\ Q(r).version < FIS(r).version$, then record r is expired;

The detail of freshness verification is shown in algorithm 1:

Algorithm 1 Freshness verification algorithm.

```
Input:
   The set of query results, $T_r$ ,including the boundary result set $T_b$;
   The set of Freshness Information Summary, FIS;
Output:
   the correct result set $Q_r$ , the expired set ResExp and the deleted set ResDel;
 1: while $T_r.hasnext$ do
 2: $if\ FIS(t).version \leq 0$ then
 3: $ResDel = ResDel \cup t$
 4:end if
 5: $if\ T_r(t).version \geq FIS(t).version$ then
 6: $Q_r = Q_r \cup t$
 7:end if
 8: $if\ Q(r).version < FIS(r).version$ then
 9: $ResExp = ResExp \cup t$
10:end if
11: end while
12: return ResExp and ResDel;
```

TTP sends the ResExp and ResDel to the DO for further processing .

5.4. *Correctness and Completeness Verification*

After freshness verification, TTP carries out the correctness and completeness verification on Q_r as follows:

Step 1: sorts the query result set Q_r according to query attribute A_i in descending order and check whether the last record's, namely lower boundary record, value of A_i is less than the LOWER , $T_L.A_i < LOWER$ and the first record's ,namely upper boundary, value of A_i is greater than the UPPER, $T_U.A_i > UPPER$, if so, go to the next step, else marks the corresponding boundary records is invalid;

Step 2: Check whether the attribute A_i's value of each tuple in Q_r is within the query range [LOWER, UPPER]. If so, go to the next step, else the record is incorrect ;

Step 3: from the last to the first of result set, TTP reconstructs each tuple's MAC value according to equation 4 and compare it with the "checksum" , if the MAC value computed in the TTP matches the MAC saved in the "checksum" attribute, then the record is correct ; If the result set forms a complete chain from the upper boundary to the lower boundary, then the query results is complete, and then sends Q_r except the boundary records and the integrity state to the client .

The algorithm of correctness and completeness verification is shown as algorithm 2.

Algorithm 2 completeness verification algorithm.

```
Input:
  The set of query results $Q_r$ ,including the boundary records;
Output:
  the final result set $Q$ , integrity_state
 1: Sorts $Q_r$ order by $A_i$ desc
    integrity_state=true   // the integrity state of results
 2:gets and removes the first and last record from $Q$
 3:check whether $T_U.A_i > UPPER$ and $T_L.A_i < LOWER$
 4: while $Q_r$.hashnext do
 5: if $r.A_i$ between LOWER and UPPER
    and $r.A_i \geq T_P^i.A_i$ then
 6: compute the MAC value according to equation 4
 7: if $MAC = r.checksum$ then
 8: $Q = Q \cup r$
  9: else integrity_state=false
10:end if
11:end if
12:end while
13:return $Q$ , integrity_state
```

6. Analysis

In this section ,we analyze the query type which this schema supports and the storage cost, VO cost. for convenience, the symbols used are shown in table 1.

Table 1 Symbol List

Symbol	meaning
num	Total num of records in the relation table
$\|Q\|$	Total num of records in the result
A_{Num}	Numbers of attribute in the relation
l	Numbers of searchable attributes
T_h	Time of one hash computation
$\|hash\|$	The size of hash (size of MD5 is 20 bytes)

6.1. *Query Type Analysis*

The above schema can comply with all types of range queries, such as normal criteria query, top k query, etc. The cloud server will return all the matching records and boundary records . However, for projections of relation R , the query clients only need a subset of attributes. According to equation 4, the cloud server has to return all the matching records including all the attributes of the relation to calculate the hash of a tuple. In order to support project operation of relation and reduce the data transformation overhead, $hash(T)$ included in equation 4 is modified as equation 6 .

$$hash(T) = hash(T.A_1) \| ...hash(T.A_n) \quad (6)$$

In this way, the cloud server can reply with the attributes the clients focus on and the hashes of the other filtered attributes , instead of actual plaintext values.

6.2. *Construction cost*

The main cost of DO consists of three part: calculation of records' MAC value, the dynamic update of records and the periodical update of FIS. A MAC of a tuple is calculated according to equation 4, so the calculation cost includes sorting each searchable attribute and calculating the hash of each tuple. The extra storage includes the added fields: “version”, “precursor” and “checksum”. The whole extra storage of the relation is num*(|version|+|precursor|+|checksum|),where |attribute name| denotes attribute byte size.

6.3. *Dynamic update cost*

Dynamic update means the modification, insertion and deletion. The outsourced tuples may only stored in the cloud server or keep a copy locally. In the second case, When modifying a record, if each precursor's key of the record keeps no change, the value of version increments by one, else the “precursor” field needs to be recalculated, finally the recalculated MAC is saved in the “checksum” and the record is uploaded to the cloud server immediately. In the first case, DO has to send the modified record to the cloud server firstly ,then the cloud server calculates the actual position of the modified record along l searchable attributes and return all the precursors’ primary key and successor records to the DO. After successfully verifying the correctness of the precursors , DO computes the hash of the updated record according to equation 4 and also recalculates the hash of successor records and sends at most $l+1$ new hashes to the cloud server. The worst case of updating a searchable attribute will lead to recalculate three records' MAC which is same with the CES, but the computation overhead of MAC is lower than the CES. When inserting a record, the MAC value of the inserted tuple and all its successors should be recalculated. In the worst case, there are $l+1$ records will be updated. When a tuple is deleted, it’s all successors need recalculate the MAC value. In the worst case, there are l records' MAC will be updated. In summary , the time cost of updating a record is at most $(l+1)*T_h$.

6.4. *Cloud Construction Cost*

When the cloud server receives a query request, the normal cost is constructing the query result set T_r . While the extra cost includes: construction of the boundary result set T_b and obtaining all the hash value of immediate precursor's primary key in each searchable dimension of records in set T_r and T_b. The size of VO is $|Q| * l * |key|$ for completeness verification, where |key| denotes the byte size of primary key.

6.5. *Verification Cost*

The integrity verification is implemented in the TTP. The cost consists of the hash and MAC calculation, check if the tuple in T_r and the boundary records in T_b satisfied corresponding condition. In order to implement freshness verification, an integer field “version” field which take four bytes respectively, are added to the primary table. Finally, the storage cost is at most num*4 bytes. Though this leads to more storage space, it can defense replay attacks effectively. The time cost of hash computation is $T_h*(|Q|+2)$.

6.6. *Schema Optimization*

The above schema can comply with all types of queries, such as normal criteria query, top k query and aggregate query, etc. However, for any type of query, the cloud server have to return each eligible tuple’s hash and its precursor’s primary key for integrity verification, the VO size will become lager with the size of the query result set, Even if a hash size of MD5 is only 20 bytes.
In order to reduce network transmission cost of VO between TTP and cloud server and the hash computation burden of TTP. We can use another hash algorithm for C in equation 4, and the hash is public to the cloud ,then the cloud server can return one hash for each record to the TTP for integrity verification. incremental hashing[20] also can be adapted to our schema which computes the hash of a message or a file by breaking the message into smaller blocks and combine the hashes of each block by using a “compression function". An incremental hash, instead of replacement, may operate by merely hashing the new contents of the block and keeping all other block hashes unchanged. The scene is the same as our schema which the hash value of a record is composed of all the attributes and its precursors’ primary key. By utilizing its XOR schema , instead of sending back all the precursor’s primary key , we return only one hash for each record by compressing them for integrity verification.

7. Experiment Analysis

The experiment is implemented on a PC with 2.2 GHz CPU and 2G memory, using mysql as database. We implement the experiment on three and six searchable attributes respectively and the experiment results used are the average of ten times experiments. We show the time cost of constructing MACC , signature chain and the freshness verification; analyze the detection successful ratio under different update cycle of FIS and different proportions of outdated data.

Fig.2 shows the comparison of construction time of MACC and CES on three and six searchable attributes, where the horizontal axis denotes the numbers of tuple, vertical axis denotes construction time of all records. The construction time of MACC includes sorting and getting precursor of each searchable attribute of every tuple, the connection of database and calculation of MAC.

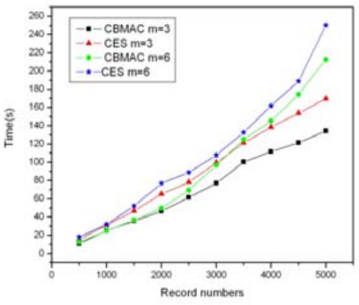

Fig. 2. Construction time of MACC and CES

The main reason of difference lies in the computation overhead of MAC and signature. The calculation of MAC is based on the hash and symmetric key while the CES is based on the digital signature and public key, under the same condition, the MACC is superior to CES. Fig.2 illustrates that the more the record numbers and the sorted fields, the longer the construction time.

Fig.3 presents the freshness verification successful ratio with different update interval and the ratio of expired data, where the horizontal axis denotes the update interval of FIS, vertical axis denotes success ratio. Fig.3 demonstrates that the shorter the update interval, the higher the success rate, in the same update interval, the more the expired data, the higher the successful rate.

Fig.4 indicates the cost of freshness verification, where the horizontal axis denotes the numbers of record in result set, vertical axis denotes the time of verification. Experiment result shows that the more the number of record in the result, the longer the verification time.

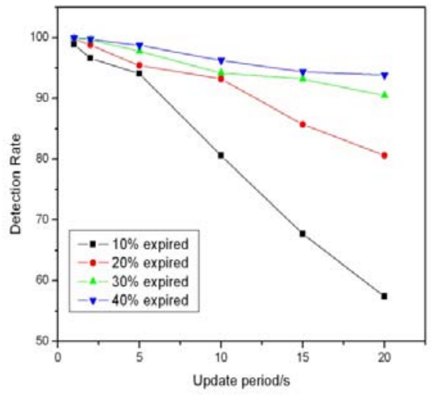

Fig.3. Successful rate of freshness verification

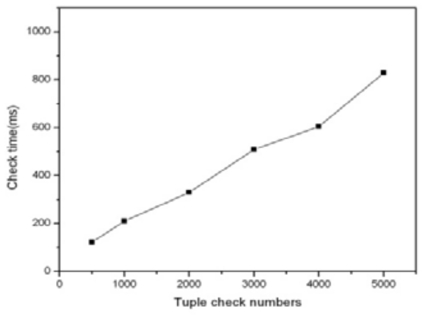

Fig.4. Freshness verification cost

8. Conclusion

This paper mainly studies the integrity verification of query result of cloud storage. With the help of TTP, we can use MAC to verify the correctness and completeness of query result and reduce the computation cost compared with the signature. Meanwhile, freshness verification is realized through FIS updating periodically. All three aspects of integrity verification of cloud storage are included in our scheme. In the future, the cost of computation and transmission of freshness verification will be taken into account, data compression aforementioned in subsection 6.6 will be deeply studied and will be tested in practice in order to improve the efficiency of communication and make the VO size smaller.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 61170169, 61170168,11271226 and 61100028, and by the Natural Science Foundation of Shandong Province under Grants ZR2012AL07 and ZR2013AM013.

1. Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM. 53(4) (2010) 50-58.
2. S Subashini and V Kavitha. A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications. 34(1) (2011) 1–11.
3. Deng-Guo Feng, Min Zhang, Yan Zhang, and Zhen Xu. Study on cloud computing security. Journal of Software. 22(1) (2011) 71–83.
4. S Ramgovind, Mariki M Eloff, and E Smith. The management of security in cloud computing. In Information Security for South Africa (ISSA). 2010, pp. 1–7.
5. Ralph C Merkle. A certified digital signature. In Advances in Cryptologył CRYPTO89 Proceedings. Springer. (1990) 218–238.
6. HweeHwa Pang and K-L Tan. Authenticating query results in edge computing. In Data Engineering, 2004. Proceedings. 20th International Conference. pp.560–571.
7. Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Dynamic authenticated index structures for outsourced databases. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data. (2006) pp. 121–132.
8. Goodrich, Michael T., Roberto Tamassia, and Nikos Triandopoulos. Super-efficient verification of dynamic outsourced databases. Topics in Cryptology–CT-RSA 2008. Springer Berlin Heidelberg.(2008). 407-424.
9. Tao Wen, Gang Sheng, Quan Guo, and Sheng Guo-Jun. Query results authentication of outsourced append-only databases. Journal of Computer Research and Development, 49(10) (2012) 2077–2085.
10. HweeHwa Pang, Arpit Jain, Krithi Ramamritham, and Kian-Lee Tan.Verifying completeness of relational query results in data publishing.In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 407–418. ACM, 2005
11. Narasimha, M., & Tsudik, G. DSAC: integrity for outsourced databases with signature aggregation and chaining. In Proceedings of the 14th ACM international conference on Information and knowledge management (ACM). pp. 235-236.
12. Zhang Min, Hong Cheng, and Chen Chi. Server transparent query authentication of outsourced database [J]. Journal of Computer Research and Development. (2010) 1:028
13. Ghasemi, Simin Noferesti, Morteza, et al. Correctness verification in database outsourcing: A trust-based fake tuples approach. Information Systems Security.(Springer) pp. 343-351.
14. Gang Sheng, Tao Wen, Quan Guo and Ying Yin. Verifying Correctness of Inner Product of Vectors in Cloud Computing. Proceedings of the 2013 international workshop on Security in cloud computing.(ACM, 2013)PP.61-68.
15. Gang Sheng, Tao Wen, Quan Guo, and Ying Yin. Privacy Preserving Inner Product of Vectors in Cloud Computing. International Journal of Distributed Sensor Networks, 2014.
16. Stavros Papadopoulos, Yin Yang, and Dimitris Papadias. Cads: Continuous authentication on data streams. In Proceedings of the 33rd international conference on Very large data bases.(VLDB Endowment, 2007) pp. 135–146.
17. Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Providing freshness guarantees for outsourced databases. In Proceedings of the 11th international conference on Extending database technology:Advances in database technology,(ACM, 2008.) pp 323–332.
18. Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac: Keyed-hashing for message authentication. 1997.
19. Bart Preneel. Cryptographic hash functions. European Transactions on Telecommunications. 5(4) (1994) 431-448.
20. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography and application to virus protection. In 27th Annual Symposium of Theory of Computing, 1995.