January 2014, Volume 1, Number 1 (pp. 3–14)
http://www.jcomsec.org

Journal of Computing and Security

A Trust-Based Approach for Correctness verification of Query Results in Data Outsourcing

Morteza Noferesti [a,*]
Simin Ghasemi [a]
Mohammad Ali Hadavi [a]
Rasool Jalili [a]
[a] *Data and Network Security Lab,Sharif University of Technology, Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: 15 December 2012
Revised: 30 June 2013
Accepted: 17 July 2013
Published Online: 20 December 2013

Keywords:
Data Outsourcing, Correctness Verification, Linear Temporal Logic, Bloom Filter

ABSTRACT

Correctness verification of query results is an important security concern in data outsourcing scenarios. In previous approaches, the correctness verification was impossible in real applications due to its high overhead. A trust-based approach is proposed here to reduce the correctness verification overhead which relies on the previous positive behavior of service provider. A client maintains a trust value for service provider showing the history of service provider comportment. Considering the trust value, the client selects a portion of query result randomly, and forwards it toward the data owner as a result-proof request. The data owner responses to the correctness of the result-proof request using a bloom filter structure. Based on the result-proof response and the trust in the service provider, the client decides whether to accept or reject its query result. In terms of performance, this approach outperforms previous approaches since it does not contain signature overhead in the verification process (which is presented by simulation results). In terms of correctness, this approach is modeled using a transition system and the correctness properties are verified through the Linear Temporal Logic.

1 Introduction

Due to the numerous advantages of cloud computing technology such as flexibility, cost efficiency, location independent resource pooling, and transference of risk, cloud computing seems to be the next-generation architecture of IT enterprise. Recent advances in hosted cloud computing and storage technology have increased the chance of offering a database service as an outsourced service. In the Database-As-a-Service (DAS) model [1], data and its management are outsourced to a third party service provider. Amazon's RDS and Microsoft's SQL Azure are the two well-known cloud database services proposed DAS model [2].

In data outsourcing scenarios, a not fully trusted third party stores and manages the data. This scenario faces new security challenges including confidentiality and correctness verification. An untrusted service provider may access or distribute unauthorized data and so violate the confidentiality of the sensitive data

* Corresponding author.

Email addresses: `mnoferesti@ce.sharif.edu` (M. Noferesti), `sghasemi@ce.sharif.edu` (S. Ghasemi), `mhadavi@ce.sharif.edu` (M. A. Hadavi), `jalili@sharif.edu` (R. Jalili)

ISSN: 2322-4460 © 2014 JComSec. All rights reserved.

[3]. Moreover, while the database management is outsourced to the service provider, it must execute the queries in a honest manner. So, there should be a way to examine whether the returned tuples satisfy the query condition or not. Such a problem refers to the correctness assurance or correctness verification of query results which include authenticity, completeness, and freshness of the results. Authenticity, here means that the result must be generated solely based on the outsourced data and not to be tampered with. Completeness, here indicates that all tuples satisfying the query condition are definitely included in the query result. Freshness, here signifies that the result is generated based on the latest updated outsourced data. The focus here is on the correctness assurance, where a new way is proposed to verify the correctness of query results in an efficient manner.

A generic model for database outsourcing consists of a data owner O, a service provider SP, some clients Cs, and some users Us which submit their queries to SP through Cs. In most approaches Us and Cs are considered the same. The O outsources its data and its management as well as query execution to SP. The SP has significant resources and expertise in building and managing the distributed cloud storage servers. The SP is considered as fully untrusted hence, a lazy service provider as well as a malicious attacker which has compromised SP, may violate the correctness of the query result. The question here is how does U ensure that the received query result is generated based on O's outsourced data. hence, a database outsourcing scenario is required to make a correctness proof for U. Consequently, the SP sends some information to U as a Verification Object (VO) together with the query results. The VO will be used by U to verify the correctness of the returned results.

A desired approach must have acceptable efficiency in practical applications. Utilizing the trust concept, an efficient and probabilistic approach is proposed here with a low overhead, more suitable for real applications. Here, C verifies the correctness of a portion of returned result instead of the entire result. The volume of this portion depends on the previous SP behaviour. According to it's behaviour, a trust value of $0 \leq T \leq 1$ is maintained by C. If $T = 1$, the query result is correct and there will be a very light verification process; if $T = 0$, C should verify the correctness of the whole returned result. The query computation and communication overhead is determined based on T.

The verification approach should be sound and complete. The soundness means if the return result is a correct result to a query, then the verification approach identifies it as a correct result. The verification approach is complete if SP can generate a VO of a correct result for every query where the approach identifies it as a correct result to the query. To demonstrate the soundness and completeness of this approach, it is modelled through the transition system and verified through the Linear Temporal Logic.

The contributions of this work are threefold: first, an infinite counting bloom filter support delete operation is proposed without false negative; second, the previous behaviour of service provider is employed to calculate the trust value in a service provider which does not rely on cryptographic tools that impose low overhead; and third, a formalism is provided to verify this approach. The linear temporal logic is used to specify and verify the soundness and completeness properties of this approach.The efficiency of this proposed approach is confirmed by the simulation results.

The related works are reviewed in Section 2. Some primitives are introduced in Section 3 to describe this proposed approach. The proposed approach is described in Section 4. The correctness probability is defined in Section 5 where how this approach provides probabilistic correctness verification is presented. The modelling approach and a formal proof on the completeness and soundness properties are discussed in Section 6. Empirical evaluation is explained in Section 7. And article concludes in Section 8.

2 Related Works

The existing approaches for correctness verification are of three categories; correctness verification based on authentication data structures, correctness verification based on digital signatures, and probabilistic correctness verification.

2.1 Correctness verification based on authentication data structure

Merkel Hash Tree (MHT) is an authentication data structure used in several approaches [4–6]. The MHT is constructed on a data set in a manner where its root maintains the authenticity of the whole set. Supporting freshness of query results is a disadvantage of this approach due to the high cost of data update.

Goodrich et al. [4] used the MHT to verify query result correctness. They divided MHT recursively into $O(\log^* n)$ levels. All nodes in these specific levels are signed by the owner. With respect to the received query, SP transmits some of these signatures with associated hash values as VO to the client. This approach provides correctness verifiability of query result for range queries of one-dimension. However, their approach is not efficient to execute $UPDATE$

queries since the imposed overhead depends on the database size. To verify freshness, they proposed an audit mechanism where the owner sends a brief history of database updates to the users. Who will be able to verify the freshness of query result in a batch. One major drawback of this approach is the necessity of the data owner's cooperation in verifying freshness.

In [5], Lie et al. presented a scheme to store MHT in the secondary storage. The main idea is to store a MHT in a B$^+$tree, called Embedded Merkle Hash Tree (EMHT). Similar to MHT, here the owner signs the EMHTs root as well. This scheme only supports range queries and it requires the data owner's cooperation to verify the freshness of the returned result.

A different mechanism based on authentication data structure is proposed by Pang et al in [6]. They proposed the VB-tree concept which is an extension of B-tree type of Merkle hash tree. In VB-tree, each internal node is signed by the owner. While processing a query, the server will identify the smallest subtree that envelopes all tuples in the result and will build a VO according to this subtree. In this mechanism, the VO size is proportional to the size of the query result. The main drawback here is the building of so many signatures that enforce high computation overhead at the owner's side.

2.2 Correctness verification based on digital signature

In digital signature-based approaches [7–9], O signs the data in a granularity where it should be verified, and outsources data along with its signatures. These signatures are used to verify the correctness of the returned result.

Utilizing signature chaining and aggregation, [8] proposed a method to verify the correctness of return results. An MHT authentication structure is constructed over the attribute values of each tuple to provide efficient verification of result authenticity for the projection queries. Timestamps are used to provide freshness verifiability of the results. While this approach supports a wide range of queries, simulation results indicate its efficiency in comparison with some existing methods in terms of communication and computation overhead imposed on executing a query.

In [7], Narasimha and Tsudik proposed an approach to verify authentication and completeness of query result. Here, O signs database tuples to provide correctness assurance in the granularity of a tuple. They proposed a signature chaining scheme where a tuple signature is linked to the immediate predecessor tuple in an associated searchable attribute. This approach supports $SELECT$, $UPDATE$, and SET operation's queries, efficiently; while, the freshness verification of query result is not supported.

Pang et al. [9] represented a scheme where O publishes a certified bitmap summary of recently updated records periodically. These periodic updates are used to maintain a fresh view of database instance. The service provider sends the query result with an aggregated signature as a VO. Although $SELECT$, $UPDATE$, and SET operation queries are supported by this approach, it does not support $JOIN$ and $PROJECT$ queries.

2.3 Probabilistic Correctness Verification

Some approaches by [10–12] propose probabilistic verification of query result, where C verifies the query correctness with some probability. These approaches do not support certain verification, but they are more applicable in lowering the overhead in real environment.

Sion [10] proposed a probabilistic approach to audit the query execution at the server side. This approach uses a challenge-response protocol, supported by a cryptographic proof, which generates a challenge for every query batch. In addition to the result of batch query, the server should determine the desired query to which the challenge is posed. A lazy server may perform query execution in a honest manner and just obtain the challenge response, so the correctness of server's response is probabilistic.

Xie et al. [11][12] proposed a probabilistic approach by adding a number of fake tuples to the outsourced database. Fake tuples are generated through a deterministic function. Based on the fact that all the fake tuples satisfying the query conditions are returned by the server or not. The client verifies the completeness of query results. The authenticity auditing of query results rely on the encryption of outsourced data. To verify the results' freshness, the data owner frequently modifies the outsourced database with some deterministic functions and checks the outsourced database status. If the server modifies the database based on the update queries, the data owner concludes with the probability that the server's results are fresh. This approach adopts the unified client model, where the data owner and clients are the same. Moreover, it only verifies the completeness and freshness of the returned result. The authenticity is verified by the encryption that impose high overhead, e.g. calculate a query over encrypted data.

Ghasemi et al. in [13] proposed an approach where trust value towards service provider in inserting some fake tuples into outsourced database called Trust Based-Fake Tuples (TBFT) is of concern. Unified client uses these fake tuples to verify the freshness and completeness of return result. When a correct result is received the trust value increases additively. On the other hand, wrong answers decreases the trust significantly. The authenticity verification of the return result relies on the encryption of outsource database. When there are some clients, this approach produces high overhead to create and execute cryptographic tools.

Unlike the previous works which are based on cryptographic tools, this proposed approach is based on trust and provides a probabilistic solution, which is more light-weighted. This approach is more realistic as it considers positive/negative past behaviour of SP for an efficient verification; while, in previous approaches SP is considered fully untrusted independent from its past behaviour.

3 Primitives

Some primitives used in describing this proposed approach are introduced in this section.

3.1 Infinite counting bloom filter

Bloom filter [14] is a probabilistic data structure, used to specify whether an element is a member of a set or not. A simple bloom filter does not support the *delete* operation. A new type of bloom filter, namely Infinite Counting Bloom Filter (ICBF) is proposed here that supports *delete* operation without false negative. In ICBF, each entry consists of a 3-bit counter and an initially set null pointer. Whenever the counter overflows, the pointer refers to another instance of 3-bits counter.

Figure 1 illustrates an ICBF, where three states of the ICBF are presented in the following phases: phase 1, the initial filter where the counters are not set and pointers refer to null; phase 2, where the first counter is set on 2 times and the second one is set on 5 times. Note: the pointers still refer to null. And phase 3, where the second counter is set on 10 times and as the maximum value that one counter can show is 7, the pointer refers to another 3-bits counter and the rest of values are preserved at this counter.

To insert an element in ICBF, the relevant counters will be incremented. The element is taken to k hash functions and mapped into k counter positions in ICBF. If all corresponding counter values are non-zero, the element has already been inserted. The false positive probability in ICBF is same as the Equation 1 where n is the number of inserted elements into the bloom filter so far, k is the number of hash functions, and m is the array length in bits. The false negative probability is zero since the decrementing of a counter occurs exactly when a *delete* operation is in process.

$$False\ Positive\ Probability = (1 - [1 - \frac{1}{m}]^{kn})^k \quad (1)$$

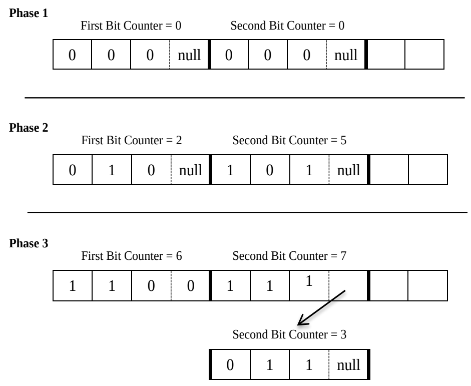


Figure 1. An ICBF example

3.2 The Transition System

Transition systems (TS) are used to describe the behavior of this proposed approach. Each TS is a directed graph where nodes represent states and edges symbolize transitions. A transition system TS is a tuple $(S, Act, \rightarrow, I, AP, L)$ where S is a set of states, Act is a set of actions, $\rightarrow \in S * Act * S$ is a transition relation, $I \subseteq S$ is a set of initial states, AP is a set of atomic propositions, and $L : S \rightarrow 2^{AP}$ is a labeling function.

3.3 Linear Temporal Logic

Linear temporal logic (LTL) [15] is a logical formalism appropriate for specifying the linear time properties. An LTL formula over the set of atomic propositions (AP) is formed according to the grammar

$$\varphi ::= true|a|\varphi_1 \wedge \varphi_2|\neg\varphi| \bigcirc \varphi|\varphi_1 \cup \varphi_2 \quad (2)$$

where, $a \in AP$.

The formula $\bigcirc\varphi$ holds currently, if φ holds in the next step. The $\varphi_1 \cup \varphi_2$ formula holds at the current moment, if there are some future moments where φ_2 holds and φ_1 would holds at all moments until that future moment.

The elementary temporal modalities are presented in most temporal logics, derived as follows:

- $\Diamond\varphi = true \cup \varphi$ *eventually* (eventually in the future)
- $\Box\varphi = \neg\Diamond\neg\varphi$ *always* (now and forever in the future)

By combining $\Diamond$ and $\Box$, $\Box\Diamond a$ (always eventually a) is obtained, stating that at any moment j there is a moment $i \geq j$ where any visited state would satisfy a.

The semantics of an LTL-formula is explained with respect to a transition system. According to the satisfaction relation for the linear time properties, the LTL formula φ holds in a state S if all paths starting in S satisfy φ. The transition system TS satisfies the property φ if all initial paths of TS-paths starting in an initial state $s_0 \in I$ satisfy φ.

In some reactive systems the fairness constraints are needed to verify the system linear temporal property. Verifying an LTL formulae can be devised under fairness constraints. Let φ be a propositional logic formula over AP. An unconditional LTL fairness constraint is an LTL formula of the form $fair = \Box\Diamond\varphi$. A fair path is a path that satisfies a fairness constraint. The TS satisfies φ under the LTL fairness assumption $fair$ if φ holds for all fair paths originating from some initial states.

4 System Overview

The class diagram of this proposed system is shown in Figure 2. In this figure the system contains four classes; O, SP, C, and U, denoting the owner, service provider, client, and user, respectively. O builds a metadata on his database and stores it on his own storage, and then outsources the database to SP. This is accomplished through $initial\_construction()$ function. There are some Us connected to a C sending queries to it by $send\_query()$ function. After sending a query, they call $receive\_result()$ and wait for the result. C calls $transmit\_query()$, function for transmitting the query to SP and waits for the result. The $calculate\_result()$ in SP calls either $generate\_correct\_result()$ or $generate\_incorrect\_result()$ to generate correct or incorrect result to the query and send it to C.

To verify the result, C generates a $result\_proof$ and sends it to O through $generate\_result\_proof()$ function. The $result\_proof$ determines the correctness ratio of the result. O calls $verify\_result()$ function to prepare the answer of $result\_proof$ query based on the metadata stored in his site. Verification process in this proposed system includes execution of the $generate\_result\_proof()$ and $verify\_result()$ functions. With respect to the returned result, C runs $calculate\_correctness\_probability()$ to calculate the correctness probability of the result. If this value passes the desired threshold, the result will be accepted and transferred to U by $result\_accepted()$ function; otherwise, C calls $result\_rejected()$ function, which resubmits the query to SP.

According to the correctness probability of the previous returned result, the $update\_trust\_value()$ maintains the trust value T in SP. Any correct result increases T while an incorrect result decreases it. The value of trust T in SP affects the overhead of verification process. The more trust in the server, the less verification overhead.

One or more U can be connected to one C (see figure 2). One or more C can be connected to one SP and one O as well.

4.1 System Behavior

With respect to the three main steps: initial construction, query processing, and result verification an in depth view of this proposed approach is presented in the following three subsections.

4.1.1 Initial construction:

To outsource a database with a relation schema $r(a_1, ..., a_i, ..., a_n)$, for each a_i as a searchable attribute, the three new attributes $H(Prev.a_i)$, $H(Next.a_i)$, and $H(a_i)$ are inserted to the schema r. $H(Prev.a_i)$ and $H(Next.a_i)$ describe the hash values of the previous and the next values of a_i for each tuple, respectively; where $H(a_i)$ is the hash values of a_i. In the approach, a special collision-resistance hash function is used for two distinct values m and n, $(m > n \Leftrightarrow H(m) > H(n))$. This function is proposed in [16] by Agrawal et al based on the division of each searchable attribute domain in several sub-domains. O sorts tuples based on searchable attributes to specify the previous and next tuples for each one. Then the hash values of searchable attributes of these tuples are inserted in the new table with schema $r'(a_1, ..., a_n, ..., H(Prev.a_i), H(a_i), H(Next.a_i), ...)$. Subsequently, for each tuple t, $Hash_t$ is inserted to the $Bloom(r)$. $Hash_t$ is computed as follows:

$$\begin{aligned} Hash_t = H(&H(t.Prev.a_1)|H(t.a_1)|H(t.Next.a_1)|... \\ &|H(t.Prev.a_n)|H(t.a_n)|H(t.Next.a_n)) \end{aligned} \tag{3}$$

where, $|$ denotes the concatenation function, and $t.k$ is the value of attribute k of tuple t. O stores and maintains the $Bloom(r)$ as a metadata for each relation r and outsources the associated relation r' to SP.

4.1.2 Query processing:

For an arbitrary query $q \in QuerySet$,where $QuerySet$ is the set of valid queries, SP computes a set A of satisfying tuples based on the query conditions as a result set. Then, it sends them to C with the hash values for each tuple as VO.

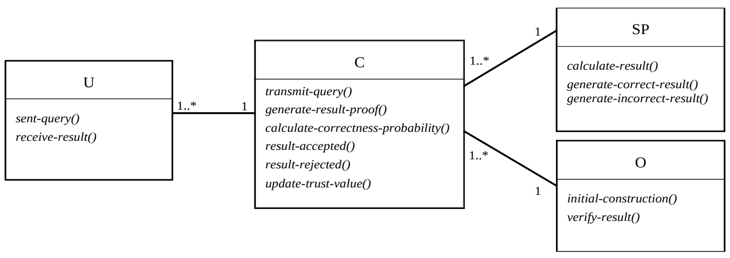


Figure 2. The proposed approach class diagram

4.1.3 Result verification:

Consider a query selects the entire rows of the relation r' where their a_i values are in the range $[A_{low}, A_{high}]$ with A_{low} and A_{high} in domain a_i. SP selects a set of tuples $A = \{r_1, ..., r_n\}$ where, $A = \{t \in r' | A_{low} \leq t.a_i \leq A_{high}\}$.

SP sends A accompanied with its corresponding VO to C. Subsequently, C uses trust value T in SP to determine the subset of A as a $result\_proof$. Assuming A' as the subset of A, for each row in A', C sends its associated hash value to O through Equation 3. By using $Bloom(r)$, O verifies whether the received tuples are correct or not. According to the O's response, the probability of receiving a correct result is calculated and the trust value is updated. If the probability is judged under a predefined threshold, the result will be rejected.

The size of A' is a fraction of size A that depends on T. This relation can be linear or nonlinear. For the sake of simplicity, the assumption is that a linear relation between m (size of A') and n (size of A), is determined by Equation 4 :

$$m = MAX((1 - T) * n, m_{min}) \tag{4}$$

where, $MAX()$ is a function that return the maximum value of its inputs, and m_{min} is the minimum value of m, since a minimum portion of returned result even from a fully trusted server must be verified. Here, m_{min} is applied to emphasis on this point that may be a fully trusted service provider changes his behavior for some reasons, e.g. external attackers or hackers, the m value states constant. So, a portion of the return result has always been verified by the data owner which provides ability to catch the attacks with high probability. Note that trust T in server is a value between zero and one, a fully trusted server gets 1 and an untrusted server gets zero; case consequently the untrusted server implies verification of all tuples, and a fully trusted server decreases the verification overhead as it decreases the number of m.

As mentioned before, C should maintains the T value; therefore, after each verification process, the new trust value in SP is calculated by C. The new trust value is the weighted mean of the old trust T_{old} and $correctness\_probability_c$ of returned result. The $correctness\_probability_c$ is the probability by which the return result for C's query is considered correct. Assume that β, the weight of T_{old}, is a number between 0 and 1. The new trust value, T_{new} can be calculated as follow:

$$T_{new} = \beta * T_{old} + (1 - \beta) * correctness\_probability_c \tag{5}$$

4.2 Database updates

SP and O are involved in an updating operation. This operation requires the maintenance of the owner-side bloom filters. To update an outsource database two approaches are presented here. In the first approach, some metadata is stored in O, which is used to maintain the bloom filters up to date with respect to the last state of the outsourced database. In the second, everything except bloom filters are outsourced to SP and update operations are carried out through a multi-round protocol between SP and O.

In a dynamic environment where data is updated in the outsourced database, the bloom filters at the owner-side should be updated as well. One solution is when O keeps a sorted list of hash values for each searchable attribute. Now, when a value changes, the sorted lists are used to compute $Hash_t$ based on Equation 3 for new inserted and old values. For an update, these values are updated in the owner-side bloom filters. This approach is useful when the updating frequency is high or when the size of outsourced database is much bigger than the size of searchable attributes. Consider applications like Youtube or Netflix. They store a great amount of multimedia data while users query on metadata of the stored multimedia files (for example on the title, keywords, or date of a video recording). In these applications, O can outsource a great amount of data to SP and keep the hash of metadata at its side.

In the second approach, O only keeps the bloom filters. Updating includes inserting the new values and deleting the old values. Consider a table with m

searchable attributes. To insert a tuple, SP calculates the real position of insertion along all m searchable attributes. The pair of adjacent tuples in each searchable attribute is then returned to O for each searchable dimension. Since tuples are linked by the hash values, O uses these pairs of adjacent tuples to verify the position of the new tuple. Inserting a new tuple leads to recomputing the $Hash_t$ of each adjacent tuple. To do this, SP sends the hash of other $m-1$ searchable attributes to the client. To maintain owner-side bloom filter, O uses this proposed result verification process to verify the returned tuples. Then, the adjacent tuples' hash values are used to compute $Hash_t$ of the new tuple. Moreover, the $Hash_t$ for the adjacent tuples are updated, including the new tuple hash value. At last, the new tuple $Hash_t$ is inserted and the adjacent $Hash_t$ tuples are updated in the owner-side bloom filters. Due to lack of space, here no explanation is given on the delete operation which is similar to the insert operation.

5 Probabilistic Correctness

The probabilistic view of correctness in our approach is due to the amount of trust value in SP as well as the false positive of bloom filters. As the correctness includes authenticity, completeness, and freshness, the definition of probabilistic authenticity, probabilistic freshness, and probabilistic completeness must be specified. Based on the definitions, the manner of their support through this approach is presented as follows:

Definition 1. Probabilistic-authenticity

The returned result to client C is p_c-probable authentic, if and only if at least p percent of the returned tuples have been generated based on O's outsourced data and have not been tampered with.

Lemma 1. *Let n be the size of the result A, m be the size of $result\_proof$ request A', T be the value of trust in SP, and $authenticity\_ratio$ be the value of $result\_proof$ response. For client C, the returned result is p_c-probable authentic, where*

$$p_c = \frac{(n-m)*T + m*authenticity\_ratio}{n} \qquad (6)$$

Proof. The bloom filter stored at O shows the authentic state of outsourced database. Let A' be a portion of the entire returned result A. According to this proposed approach C sends A' to O, and O verifies the authenticity of A'.

The $authentication\_probability$ of the result is calculated as follows:

$$authentication\_probability = \frac{the\ portion\ of\ result\ which\ is\ certainly\ authentic}{the\ entire\ result}$$

$authentication\_probability$ indicates the probability of the result's authenticity which is a number between 0 and 1. C sends m tuples to O. O checks the existence of A' in the corresponding bloom filters. If one member of A' is not in the bloom filters, the entire result is rejected. On the other hand, if the corresponding bloom filters contain all the members of this subset, to O, this portion of result is authentic. So the $authenticity\_ratio$, specified by O, is calculated as follows:

$$authenticity\_ratio = \begin{cases} 1-FP & \text{if the result is authentic} \\ 0 & \text{if the result is not authentic} \end{cases}$$

where, FP is the false positive probability of bloom filter.

With respect to $O's$ response, C expects that $m*authentication\_ratio$ percent of returned result to be authentic. Moreover, due to the definition of T, C expects that $(n-m)*T$ portion of returned result to be authentic as well. In conclusion, with the probability obtained through Equation 6, C is sure that the returned result is authentic. With respect to definition 1, this proposed approach is p_c-probable authentic where, p_c is equal to $authentication\_probability$ and it is a digit between 0 and 1. □

To explain more about this lemmas, the following example is presented:

Example 1. Suppose that the number of tuples in the return result is 32, the number of client request is 10, trust in SP is 50%, and false positive ratio of bloom filter is zero. What is the authentication probability of the return result?
Based on this proposed approach the client sends 10 tuples to the data owner. The bloom filter false positive ratio is zero, so if the data owner confirms the authenticity of the request, the client assures that these 10 tuples are absolutely authentic. On the other hand the client's trust in service provider is 50%, based on the history of SP's behavior; 50% of its results are authentic. In conclusion, from the 22 tuples remain, 11 tuples are authentic. Client is sure about 21 tuples (11+10), that is the authentication probability of return result is 21/32. This corresponds to Equation 6.

Definition 2. Probabilistic-freshness

The returned result to client C is p_c-probable fresh, if and only if at least p percent of returned tuples have been generated based on the latest instance of outsourced data.

Lemma 2. *Let n be the size of result A, m be the size of $result\_proof$ request A', T be the value of trust in SP, and $authenticity\_ratio$ be the value of $result\_proof$ response. For client C, the returned result is p_c-probable fresh, where*

$$p_c = \frac{(n - m) * T + m * freshness\_ratio}{n} \tag{7}$$

Proof. ICBF is a bloom filter that supports *delete* and *insert* operations. So, it is used in this approach to store the latest state of outsourced database. Due to this property of ICBF, when O generates the $result\_proof$ response, it not only verifies the authenticity but also the freshness of the result. Based on definition 2, the freshness probability is the same as authentication probability calculated in lemma 1. □

Definition 3. Probabilistic-completeness

The returned result to client C is p_c-probable complete, if and only if at least p percent of tuples satisfying the query conditions are included in the returned result.

Lemma 3. *Let n be the size of result A, and m be the size of a subset of result A' (selected by C to check the completeness of result). For client C, the returned result is p_c-probable complete where,*

$$p_c = \frac{m}{n} \tag{8}$$

Proof. Completeness verification is issued by C as follows:

$$\begin{aligned} \forall t' \in A', [(\exists t \in A, & t'.H(Prev.a_i) = H(t.a_i) \lor \\ & H(t'.Prev.a_i) < A_{low}) \land \\ (\exists t \in A, & t'.H(Next.a_i) = H(t.a_i) \lor \\ & H(t'.Next.a_i) > A_{high})] \end{aligned} \tag{9}$$

Equation 9 checks whether for every tuple t' in A' there exist not any previous and next tuples with respect to the value of a_i in A or not. If yes, the returned result to client C is probably complete, where the $completeness\_probability_c$ is specified by Equation 8

Consider a state where the result A is empty. C cannot select a subset of A as A'. To verify correctness of empty results, SP sends a special tuple to C. Once again suppose that a query selects an entire row of the relation r', where its a_i value is in $[A_{low}, A_{high}]$ range. If the result of query is empty, there is a tuple t'' in r' where, $H(t''.Prev.a_i) < H(A_{low}) < H(A_{high}) < H(t''.Next.a_i)$. Now, when the result is empty, the SP sends the hash values of attributes of t'' to C. After the verification process, C is sure that t is in outsourced database and the result of query should be empty. Moreover, the $correctness\_probability_c$ is one.

In conclusion, for client C this proposed approach is p_c-probably complete (definition 3) where, p_c is equal to $completeness\_probability_c$. □

Definition 4. Probabilistic-correctness

The returned result to client C is p_c-probable correct, if and only if it is p_c-probable authentic, p_c-probable complete, and p_c-probable fresh.

Theorem 1. *For client C, the returned result is p_c-probable correct where, $0 \le p_c \le 1$.*

Proof. According to lemma 1, lemma 2, and lemma 3 this proposed approach for client C is p'_c-probable authentic, p''_c-probable fresh, and p'''_c-probable complete where, p'_c, p''_c, and p'''_c are digits between 0 and 1. It can be deduced that $p_c = MIN(p'_c, p''_c, p'''_c)$ percent of returned result is authentic, fresh, and complete. Where $MIN()$ is a function that returns the minimum value of its inputs.

Based on $probabilistic\_correctness$ definition (definition 4), for client C this proposed approach is p_c-probable correct where, $0 \le p_c \le 1$ is equal $MIN(p'_c, p''_c, p'''_c)$. □

6 Modelling the Proposed System

The soundness and the completeness of this proposed approach is approved in this section. This approach is sound if the verification approach identifies a correct return result as a correct result and is complete if it verifies every correct result. To demonstrate these properties, this approach is modelled using the transition system and is verified trough LTL (see Figure 3).The states are represented by ovals and transitions by labeled edges. The state names are depicted inside the ovals. The initial states are indicated by having an incoming arrow without a source.

The state space is $S = \{wait$, $generate\_query$, $calculate\_result$, $generate\_correct\_result$, $generate\_incorrect\_result$, $generate\_result\_proof$, $verify\_result$, $result\_accepted$, $result\_rejected\}$. The set of the initial states is $I = \{wait\}$. The action set is $Act = \{$ $transmit\_query$, $send\_result$, $send\_proof$, $\tau\}$ where, the client action $transmit\_query$ denotes transferring user query to SP, $send\_result$ is

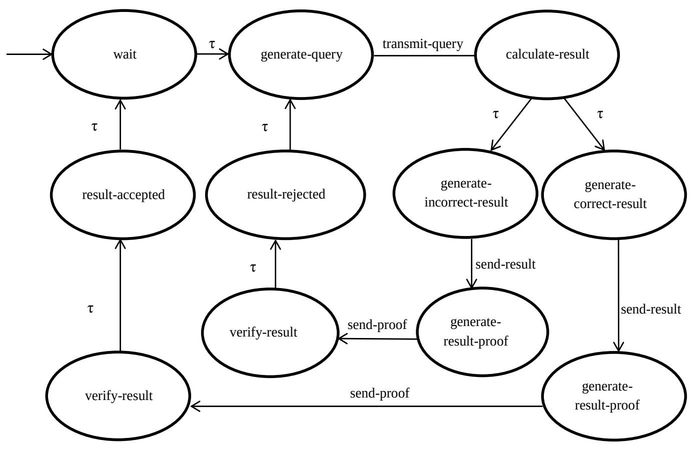


Figure 3. The transition system of proposed approach

an action for sending query result to the client, and *send_proof* is for sending a proof query to the owner. Here the actions with no further interest, e.g. how SP generate correct or incorrect result, are all denoted by the distinguished action symbol τ. Assuming the reliability of communication links, when a message is sent to a destination, it would reach if eventually.

There are 13 transitions in this newly developed system. An example is

$$(generate\_query, send\_query, calculate\_result) \in \rightarrow$$

In this transition system, whenever SP generates a correct result, it will be accepted by C because of the bloom filter function. On the other hand, if SP generates an incorrect result, it will be either accepted or rejected by C, due to the false positive of bloom filter and the newly developed probabilistic approach for correctness verification. The probabilistic approach discussed in Section 5.

The set of propositions AP is always chosen subject to the characteristics of interest. Notations for atomic propositions of the proposed transition system are presented in table 1.

The labeling function L defines a $L(s) \subseteq 2^{AP}$ set of atomic propositions for any states s. Here, $L(s)$ stands for the atomic propositions where $a \in AP$ of which are satisfied by the state s. The labeling function of the transition system shown in Figure 3 is defined as follows:

$$L(s) = \{s\} \cap AP$$

6.1 Completeness and soundness property of the proposed approach

A formalism which can proof the soundness and completeness of the proposed approach is provided here. The transition system is described and a formalism should be selected. Because the soundness and completeness properties are linear in time, LTL, which is appropriate for specifying linear time properties is selected. The properties are expressed by LTL, and indicate that the proposed approach satisfies them.

6.1.1 Completeness property:

This proposed approach is complete if it verifies every p_c-probable C's query result as a p_c-probable correct result. The system response to each query $q \in QuerySet$ and finally enters the *result_accepted* state. The completeness property can be represented by $\Box(q \rightarrow \Diamond result\_accepted)$ formula.

Consider the transition system of the proposed approach presented in Figure 3, there is one path starting with initial state *wait* in the transition system that does not satisfy completeness property. This paths is

$$\begin{aligned}&Path\ 1 : wait \rightarrow generate\_query \rightarrow calculate\_result \rightarrow\\ &generate\_incorrect\_result \rightarrow generate\_result\_proof\\ &\rightarrow verify\_result \rightarrow result\_rejected \rightarrow generate\_query\\ &\rightarrow calculate\_result \rightarrow generate\_incorrect\_result \rightarrow ....\end{aligned}$$

Path 1 is an unrealistic situation where SP always generates incorrect results, not fair in real situation. With respect to *Path 1*, it is concluded that the completeness property $\Box(q \rightarrow \Diamond result\_accepted)$ cannot be established, since SP can always generate an incorrect message.

If SP is assumed to be fair enough to *generate_correct_result* and *generate_incorrect_result* with positive probability, this unrealistic case can

Table 1. Notations for atomic proposition

Notation	Atomic proposition
$generate\_query$	C generates a query.
$calculate\_result$	SP calculates the result of query.
$generate\_correct\_result$	SP generates a p-probable correct result.
$generate\_incorrect\_result$	SP generates a result which is not p-probable correct.
$generate\_result\_proof$	C generates a proof request for the result.
$verify\_result$	O verifies the result.
$result\_accepted$	The result is accepted.
$result\_rejected$	The result is rejected.

be ignored by the means of an unconditional LTL fairness assumption where,

$$\begin{aligned} fair =& \Box\Diamond generate\_correct\_result \wedge \\ & \Box\Diamond generate\_incorrect\_result. \end{aligned}$$

It is not difficult to check this state since now

$$TS \models_{fair} \Box(q \rightarrow \Diamond result\_accepted) \tag{10}$$

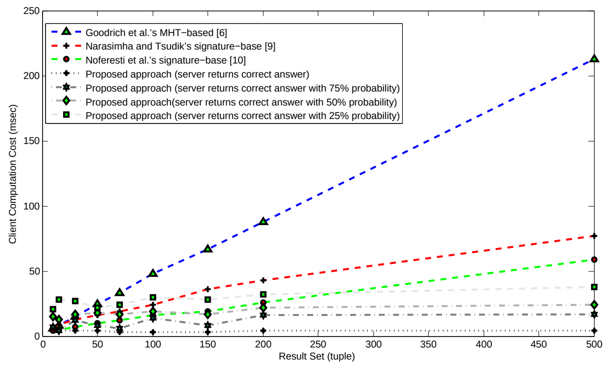
6.1.2 Soundness property:

This proposed approach is sound if it identifies a p_c-probable C's query result as a p_c-probable correct result. Whenever the system gets the $result\_accepted$ state, the result is bound to be p-probable correct. This property can be described by a formula of $\Box\neg(result\_accepted \wedge \bigcirc\bigcirc\bigcirc(generate\_incorrect\_result))$ type. All the paths beginning in $wait$ do satisfy this property. So,

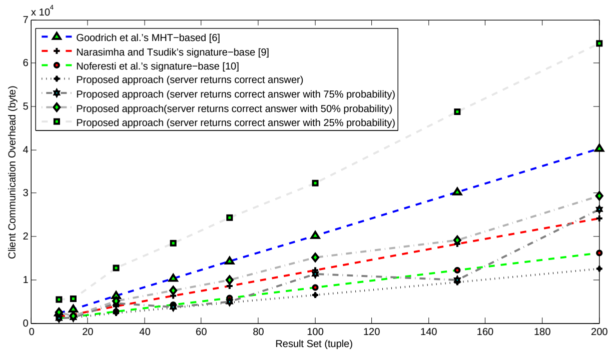
$$\begin{aligned} TS \models \Box\neg(& result\_accepted \wedge \\ & \bigcirc\bigcirc\bigcirc(generate\_incorrect\_result)) \end{aligned} \tag{11}$$

7 Implementation and Result

This approach is simulated and compared with Goodrich et al.'s MHT-based, Narasimha and Tsudik's signature-based, and Noferesti et al.'s signature-based methods' result. The table schema $SENSOR(SensorID, Loc, Tmp)$ provided by a weather forecasting company is considered here. Each sensor has a unique $SensorID$ located at a specific location Loc with periodic sampling of the temperature Tmp. Loc is considered as a searchable attribute. The simple $SELECT$ queries which are uniformly distributed in the data space from the $SENSOR$ table filled with 1000000 random records



(a) Client computation cost



(b) Communication overhead

Figure 4. Client computation cost and communication overhead of different approaches.

is executed here. The weight of T_{old} in 5 is initialized by 0.75. The minimum portion of returned result which is always verified is initialized by 0.1 of returned result. These values should be initialized based on environment specifications. We evaluate this proposed approach in three modes, when SP always generates a correct answer or correct result with 75%,50%, or 25% probability.

The execution time of this approach is presented in Figure 4a. In the Narasimha and Tsudiks approach,

SP sorts database per each searchable attribute in the table schema; in Noferesti et al.'s approach database is sorted according to the searchable attributes presented in the query condition, with no reliance on sort. The result verification in MHT-based approach contains executing several hash and concatenation functions along with verifying the root signature. On the contrary, the result verification in this approach is not constructed through the cryptographic tools; hence making this approach more light-weighted. One of the advantages is that when SP generates correct answers its overhead is too low.

The communication overhead of different approaches are compared in Figure 4b. Communication overhead refers to the volume of data transferred from or to a user except query result size. In the Narasimha and Tsudiks and Noferesti et al.'s signature based approach VO contains two boundary values for each tuple in the result set. VO in the MHT-based approach contains its co-path to the root and the root signature, so its size is big. In our approach, when U receives an incorrect result, the query is resubmitted to SP and the result will be received afterwards. This fact indicates that when SP generates incorrect result the communication overhead is increased.

8 Conclusion

Database outsourcing scenario is a noteworthy approach to outsource data management to a service provider. In this trend, correctness assurance of query results is an important security issue to concerned about. A new approach which verifies the correctness of returned results efficiently is proposed here.

This proposed approach utilizes the trade off between security and efficiency. The previous behavior of service provider is applied to measure a trust value in SP. The amount of trust specifies the verification process overhead.

Client sends a portion of result to the data owner to verify the correctness of entire result. The size of this portion is determined by the trust value in SP. Here, O maintains some bloom filters showing the current state of outsourced database. With respect to these bloom filters, O verifies the portion of returned result which was chosen by C. If this portion is not in the bloom filters, the entire returned result will be rejected. Otherwise, the returned result is accepted by a probability, ejected by bloom filter's false positive and the trust value in SP.

A transition system is defined to describe the behavior of the proposed system. Linear temporal logic (LTL) is used as a logical formalism suited for specifying linear time properties. The soundness and completeness properties of the verification algorithm are described by LTL. The transition system of the proposed approach satisfies these properties is expressed here.

An applicable and efficient data outsourcing approach is desired; therefore, the overhead of approach and supporting a wide range of queries are important and mandatory. Data structures' types instead of ICBF which can be used to support the correctness verifiability of other more advanced query types are planned to investigate deeply in the future.

References

[1] Hakan Hacigümüs, Sharad Mehrotra, and Balakrishna R. Iyer. Providing database as a service. In Rakesh Agrawal and Klaus R. Dittrich, editors, *ICDE*, pages 29–38. IEEE Computer Society, 2002. ISBN 0-7695-1531-2.

[2] Carlo Curino, Evan P. C. Jones, Raluca A. Popa, Nirmesh Malviya, Eugene Wu, Samuel Madden, Hari Balakrishnan, and Nickolai Zeldovich. Relational cloud: a database service for the cloud. In *CIDR*, pages 235–240. www.cidrdb.org, 2011.

[3] Pierangela Samarati and Sabrina De Capitani di Vimercati. Data protection in outsourcing scenarios: issues and directions. In Dengguo Feng, David A. Basin, and Peng Liu, editors, *ASIACCS*, pages 1–14. ACM, 2010. ISBN 978-1-60558-936-7.

[4] Michael T. Goodrich, Roberto Tamassia, and Nikos Triandopoulos. Super-efficient verification of dynamic outsourced databases. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 407–424. Springer, 2008. ISBN 978-3-540-79262-8.

[5] Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Dynamic authenticated index structures for outsourced databases. In Surajit Chaudhuri, Vagelis Hristidis, and Neoklis Polyzotis, editors, *SIGMOD Conference*, pages 121–132. ACM, 2006. ISBN 1-59593-256-9.

[6] HweeHwa Pang and Kian-Lee Tan. Authenticating query results in edge computing. In Z. Meral Özsoyoglu and Stanley B. Zdonik, editors, *ICDE*, pages 560–571. IEEE Computer Society, 2004. ISBN 0-7695-2065-0.

[7] Maithili Narasimha and Gene Tsudik. Dsac: An approach to ensure integrity of outsourced databases using signature aggregation and chaining. *IACR Cryptology ePrint Archive*, 2005: 297, 2005.

[8] Morteza Noferesti, Mohammad Ali Hadavi, and Rasool Jalili. A signature-based approach of cor-

rectness assurance in data outsourcing scenarios. In Sushil Jajodia and Chandan Mazumdar, editors, *ICISS*, volume 7093 of *Lecture Notes in Computer Science*, pages 374–378. Springer, 2011. ISBN 978-3-642-25559-5.

[9] HweeHwa Pang, Jilian Zhang, and Kyriakos Mouratidis. Scalable verification for outsourced dynamic databases. *PVLDB*, 2(1):802–813, 2009.

[10] Radu Sion. Query execution assurance for outsourced databases. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *VLDB*, pages 601–612. ACM, 2005. ISBN 1-59593-154-6, 1-59593-177-5.

[11] Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Integrity auditing of outsourced data. In Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold, editors, *VLDB*, pages 782–793. ACM, 2007. ISBN 978-1-59593-649-3.

[12] Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Providing freshness guarantees for outsourced databases. In Alfons Kemper, Patrick Valduriez, Noureddine Mouaddib, Jens Teubner, Mokrane Bouzeghoub, Volker Markl, Laurent Amsaleg, and Ioana Manolescu, editors, *EDBT*, volume 261 of *ACM International Conference Proceeding Series*, pages 323–332. ACM, 2008. ISBN 978-1-59593-926-5.

[13] Simin Ghasemi, Morteza Noferesti, Mohammad Ali Hadavi, Sadegh Dorri Nogoorani, and Rasool Jalili. Correctness verification in database outsourcing: A trust-based fake tuples approach. In Venkat Venkatakrishnan and Diganta Goswami, editors, *ICISS*, volume 7671 of *Lecture Notes in Computer Science*, pages 343–351. Springer, 2012. ISBN 978-3-642-35129-7, 978-3-642-35130-3.

[14] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[15] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[16] Divyakant Agrawal, Amr El Abbadi, Fatih Emekçi, and Ahmed Metwally. Database management as a service: Challenges and opportunities. In Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng, editors, *ICDE*, pages 1709–1716. IEEE, 2009. ISBN 978-0-7695-3545-6.

Morteza Noferesti received his bachelor's degree in information technology from Shiraz University of Technology, Shiraz, Iran, in 2009. He also received his master's degree from Sharif University of Technology, Tehran, Iran in 2011. He is now a Ph.D. student at Sharif University of Technology. His research interests include big data security, database security, and security formal models.

Simin Ghasemi graduated on B.Sc. in 2010 from Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran. She continued her M.Sc. in Sharif University of Technology, Tehran, Iran, on Network Security in September 2010. She graduated in September 2012. She started to work from November 2012 and now she is faculty in Payam Noor University of Zanjan.

Mohammad Ali Hadavi received his B.Sc. degree in software engineering from Ferdowsi University of Mashhad, Mashhad, Iran, in 2002. He also received his M.Sc. degree in software engineering from Amirkabir University of Technology, Tehran, Iran, in 2004. He is now a Ph.D. student at Sharif University of Technology, Tehran, Iran. His research interests include software security, database security, and security aspects of data outsourcing.

Rasool Jalili was born in Iran in 1961. He received his bachelor's degree in Computer Science from Ferdowsi University of Mashhad in 1985, and his master's degree in Computer Engineering from Sharif University of Technology in 1989. He received his Ph.D. in Computer Science from The University of Sydney, Australia, in 1995. He then joined the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, in 1995. He has published more than 130 papers in Computer Security and Pervasive Computing in international journals and conferences proceedings. He is now an associate professor. His research interests include such areas as access control, vulnerability analysis, and database security-which he conducts at his network security laboratory (DNSL, dnsl.sharif.ir).