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

Morteza Noferesti, Simin Ghasemi, Mohammad Ali Hadavi, and Rasool Jalili

Data and Network Security Lab,
Sharif University of Technology, Tehran, Iran
{mnoferesti@ce.,sghasemi@ce.,mhadavi@ce.,jalili@}sharif.edu

Abstract. Correctness verification of query results is an important security concern in data outsourcing scenarios. In previous approaches, correctness verification has been impossible in real applications due to its high overhead. This paper proposes a trust-based approach to reduce the correctness verification overhead relying 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 randomly selects a portion of query result and forwards toward the data owner as a result-proof request. The data owner using a bloom filter, constructed as an abstraction of current state of outsourced data, to response the correctness of result-proof request. The client decides to accept or reject its query result, based on the result-proof response and its trust to the service provider. In terms of performance, our approach outperforms the previous published approaches due to the lack of signature overhead in the verification process which is presented by simulation results. In terms of correctness, the approach is modeled using a transition system and correctness properties is verified by Linear Temporal Logic.

Keywords: data outsourcing; correctness verification; Linear Temporal Logic; bloom filter

1 Introduction

Due to the numerous advantages of cloud computing technology such as flexibility, cost efficiency, location independent resource pooling, and transference of risk, it seems to be the next-generation architecture of IT enterprise. Recent advances in hosted cloud computing and storage technology increase the chance to offer a database service as an outsourced service. In the Database-As-a-Service (DAS) model [6], data and its management is outsourced to a third party service provider. Amazon's RDS and Microsoft's SQL Azure are two well-known cloud database services which are proposed database as a service model [4].

In data outsourcing scenarios, an untrusted 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 [12]. Moreover, while the database management is outsourced to the service provider, it must execute the queries honestly. So, there should be a way to examine whether the returned tuples satisfy the query condition or not. Such kind of problem refers to the correctness assurance or correctness verification of query results which includes integrity, completeness, and freshness of the results. Integrity means the result must be generated solely based on the outsourced data and not to be tampered with. Completeness indicates that all tuples satisfying the query condition are definitely included in the query result. Freshness signifies that the result is generated based on the latest updates on the outsourced data. This paper focuses on the correctness assurance problem and proposes a novel way to efficiently verify the correctness of query results.

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

A desired approach must have acceptable efficiency in practical applications. Utilizing the trust concept, we propose an efficient and probabilistic approach with low overhead, more suitable for real applications. In this approach, C verifies the correctness of a part of returned result instead of the entire result. The volume of this part depends on the previous behavior of SP. According to SP's behavior, a trust value $0 \leq T \leq 1$ 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 return result. Based on T, the query computation and communication overhead is determined.

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

The remainder of paper is organized as follows. Section 2 reviews related works. Section 3 introduces some primitives to describe our approach. The proposed approach is described in Section 4. Section 5 defines the correctness probability and shows how our approach provides probabilistic correctness verification. The approach modelling and a formal proof on the completeness and soundness

property are discussed in Section 6. Empirical evaluation is explained in Section 7. Finally, Section 8 concludes the paper.

2 Related Work

The existing approaches for correctness verification can be divided into three categories, correctness verification based on authentication data structures, correctness verification based on digital signatures, and probabilistic correctness verification.

Merkel Hash Tree (MHT) is an authentication data structure, which has been used in several approaches [5, 7, 10]. MHT is constructed on a data set such that its root maintains the integrity of whole set. Supporting freshness of query results is a disadvantage of using authentication data structures due to the high cost of data update.

In digital signature-based approaches [8, 9, 11], the data O sings data in a granularity in which it should be verified, and outsources the data along with its signatures. After submitting a query, the server generates the result and sends it back with the corresponding signatures. Afterwards, the signatures are used to verify the correctness of the returned result.

Some approaches [13, 14] propose probabilistic verification of query result. In these approach C verifies the query correctness with some probability. Although these approaches does not support certain verification, by lowering the overhead they are more applicable in real environment.

3 Primitives

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

3.1 Infinite counting bloom filter

Bloom filter [3] 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. we propose a new type of bloom filter, namely Infinite Counting Bloom Filter (ICBF), supports *delete* operation without false negative. In ICBF, each entry consists of four bits partitioned into a 3-bit counter and a 1-bit overflow indicator. Whenever the counter overflows, the 1-bit indicator is set and the remainder of the counter is handled in the next instance of the array.

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 the same as Equation 1 where n is the number of inserted elements into the bloom filter up to now, k is the number of hash functions, and m is the array length in bits. The false

negative probability is zero as decrementing a counter happens exactly when a *delete* operation is processed.

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

Fig.1. depicts an ICBF, where three states of the ICBF are demonstrated. phase 1 shows the initial filter; in which the counters are not set. In the second phase the first counter 2 times and the second one, 5 times have been set. Note that the extra bits are still zero. In the third phase, the second counter has been set 10 times and as the counter can show up to 7, the extra bit set to 1 and the rest of values preserved at the second array of counters.

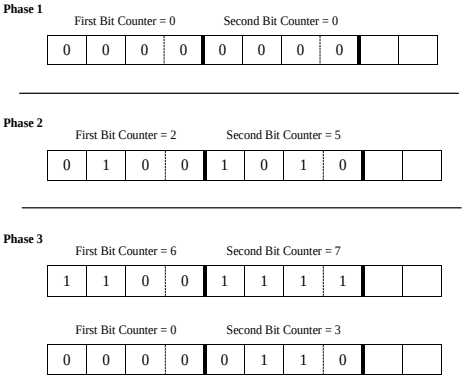

Fig. 1. An ICBF example

3.2 The Transition System

Transition systems (TS) are used to describe the behavior of our 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) [2] is a logical formalism appropriate for specifying linear time properties. An LTL formula over the set AP of atomic propositions 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 in which φ_1 holds and φ_2 holds at all moments until that future moment.

The elementary temporal modalities are presented in most temporal logics which are 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 $i \geq j$ at which any visited states satisfy a.

The semantics of an LTL-formula is explained with respect to a transition system. According to the satisfaction relation for linear time properties, the LTL formula φ holds in 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 fairness constraints are needed to verify the system linear temporal property. Verifying an LTL formulae under fairness constraints can be done. 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. TS satisfies φ under the LTL fairness assumption $fair$ if φ holds for all fair paths that originate from some initial states.

4 System Overview

Fig.2. shows the class diagram of our proposed system. As depicted in the figure, there are four classes in our system; O, SP, C, and U, denoting owner, service provider, client, and user, respectively. O builds metadata on his database and stores it on his own storage, and then outsources the database to SP. This is done 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 waiting for the result.

calculate_result() in SP calls either *generate_incorrect_result*() or *generate_correct_ result*() to generate incorrect or correct 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 our system includes execution of *generate_result_proof*() and $verify\_result()$ functions. With respect to the returned result, C runs *calculate_correctness_probability*() to calculates the correctness probability of the result. If this value goes over 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.

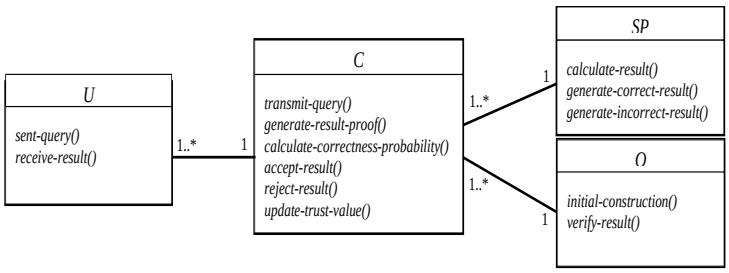

Fig. 2. The proposed approach class diagram

According to the correctness probability of previous returned result, *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 server, the less verification overhead is.

As **Fig.2.** shows, one or more U can be connected to one C. Also one or more C can be connected to one SP and one O.

4.1 System Behavior

We go into depth of our approach with respect to three main steps including initial construction, query processing, and result verification in three following subsections.

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, 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; and $H(a_i)$ is the hash values of a_i. In our approach, a special collision-resistance hash function in used such that for two distinct values m and n, $(m > n \Leftrightarrow H(m) > H(n))$. This function is proposed in [1] by Agrawal et al based on the division of each searchable attribute domain into 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:

$$Hash_t = H(H(t.Prev.a_1)|H(t.a_1)|H(t.Next.a_1)|... \\ |H(t.Prev.a_n)|H(t.a_n)|H(t.Next.a_n)) \quad (3)$$

where | denotes 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 associated relation r' to SP.

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.

Result verification: Consider a query selects 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 together 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 using Equation 3. O using $Bloom(r)$, 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 goes under a predefined threshold, the result will be rejected.

The size of A' is a fraction of size A depends on T. This relation can be linear or nonlinear. For the sake of simplicity, we assume a linear relation between m (size of A') and n (size of A), 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, because a minimum part of returned result even from a fully trusted server must be verified. 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. In this state, untrusted server implies verification of all tuples, when a fully trust server decreases the verification overhead as it decreases the number of m.

As we mentioned, C should maintains the T value. To this purpose, after each verification process, the new trust value in SP is calculated by C. It is the weighted mean of old trust T_{old} and $correctness\_probability$ of returned result. Suppose 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:

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

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.

Definition 1. ***Probabilistic-authenticity***

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

Lemma 1. *Let n is the size of result A, m is the size of result_proof request A', T is the value of trust is SP, and authenticity_ratio is the value of result_proof response. In our approach, the returned result is p-probable authentic where*

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

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

The *authentication_probability* of the result is calculated as follows:

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

authentication_probability shows the probability that the result is authentic, that 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' were not in the bloom filters, the entire result is absolutely rejected. On the other hand, if corresponding bloom filters contain all the members of this subset, from the O's view this part 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 the response of O, C expects that $m * authentication\_ratio$ percent of returned result is authentic. In the other side, due to the definition of T, C expects that $(n-m) * T$ part of returned result is authentic too. In conclusion, With the probability, calculated by Equation 6, C is sure that the returned result is authentic. So, our proposed approach is p-probable authentic where p is equal to *authentication_probability* and is a digit between 0 and 1.

Definition 2. ***Probabilistic-freshness***

The returned result is p-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 is the size of result A, m is the size of $result\_proof$ request A', T is the value of trust is SP, and $authenticity\_ratio$ is the value of $result\_proof$ response. In our approach, the returned result is p-probable fresh where*

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

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

Definition 3. ***Probabilistic-completeness***

The returned result is p-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 is the size of result A, and m is the size of a subset of result A' (which is selected by C to check the completeness of result). In our approach, the returned result is p-probable complete where*

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

Proof. Completeness verification is performed by C as follows:

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

Equation 9 checks that for every tuple t' in A', are there any previous and next tuples with respect to the value of a_i in A or not. If yes, the returned result is probably complete where the $completeness\_probability$ 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 the range $[A_{low}, A_{high}]$, 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, SP sends the hash values of attributes of t'' to C. After verification process, C is sure that t is in outsourced database and the result of query should be empty. Moreover, the $correctness\_probability$ is one.

In conclusion, our proposed approach is p-probably complete where p is equal to $completeness\_probability$.

Definition 4. ***Probabilistic-correctness***

The returned result is p-probable correct, if and only if it was p-probable authentic, p-probable complete, and p-probable fresh.

Theorem 1. *In our approach, the returned result is p-probable correct where* $0 \leq p \leq 1$.

Proof. According to **lemma 1**, **lemma 2**, and **lemma 3** our proposed approach is p'-probable authentic, p''-probable fresh, and p'''-probable complete where p', p'', and p''' are digits between 0 and 1. We can conclude that $p = MIN(p', p'', p''')$ percent of returned result is authentic, fresh, and complete. Where $MIN()$ is a function that return the minimum value of its inputs.

Based on *probabilistic_correctness* definition, our proposed approach is p-probable correct where $0 \leq p \leq 1$ is equal to $MIN(p', p'', p''')$.

6 Modelling Proposed System

The transition system in **Fig.3.** models the proposed system. States are represented by ovals and transitions by labeled edges. State names are depicted inside the ovals. Initial states are indicated by having an incoming arrow without a source.

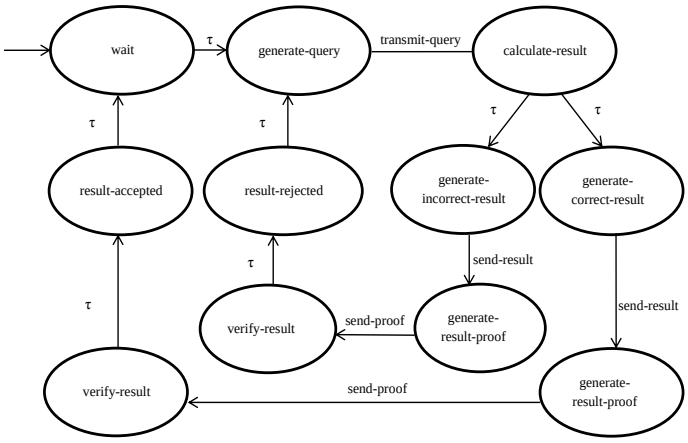

Fig. 3. The transition system of proposed approach

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 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 an action for sending query result to the client, and *send_proof* is for sending a proof query to the owner. The actions we have 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 will be eventually received.

There are 13 transitions in our system. An example one is

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

In the 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 our probabilistic approach for correctness verification. We discussed our probabilistic approach in Section 5.

The set of propositions AP is always chosen depending on the characteristics of interest. Notations for atomic propositions of the proposed transition system are shown in **Table 1**.

Table 1. Notations for atomic proposition of proposed approach

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.

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

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

6.1 Completeness and soundness property of proposed approach

We define two properties, express them by LTL, and show that proposed approach satisfies them.

Completeness property: The system response to each query $q \in QuerySet$ and finally enters the $result\_accepted$ state. The completeness property can be represented by a formula of the form $\Box(q \rightarrow \Diamond result\_accepted)$.

Consider the transition system of the proposed approach depicted in **Fig. 3.**, there is one path started with initial state $wait$ in the transition system of the proposed approach that does not satisfy completeness property. This paths is $Path\,1 : wait \rightarrow generate\_query \rightarrow calculate\_result \rightarrow generate\_incorrect\_resu$

$lt \rightarrow generate\_result\_proof \rightarrow verify\_result \rightarrow result\_rejected \rightarrow generate\_query \rightarrow calculate\_result \rightarrow generate\_incorrect\_result \rightarrow ...$.

We claim that $Path\ 1$ is an unrealistic situation in which SP always generates incorrect results, not fair in real situation. With respect to *Path 1*, we conclude 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 such that both events *generate_correct_result* and *generate_incorrect_result* occur with positive probability, then this unrealistic case can be ignored by means of an unconditional LTL fairness assumption $fair = \Box\Diamond generate\_correct\_result \wedge \Box\Diamond generate\_incorrect\_result$ It is not difficult to check that now

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

Soundness property: Each time the system get the *result_accepted* state, surely the result is p-probable correct. This property can be described by a formula of the type $\Box\neg(result\_accepted \wedge \bigcirc\bigcirc\bigcirc(generate\_incorrect\_result))$. All the paths started in *wait* do satisfy this property. So,

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

7 Implementation and Result

We simulated our approach and compared the results with Goodrich et al.'s MHT-based, Narasimha and Tsudik's signature-based, and Noferesti et al.'s signature-based methods. We consider the table schema $SENSOR(SensorID, Loc, Tmp)$ provided by a weather forecasting company. Each sensor has a unique $SensorID$. It has been located at a specific location Loc and they periodically sample the temperature Tmp. Loc is considered as a searchable attribute. We executed simple $SELECT$ queries which are uniformly distributed in the data space from the $SENSOR$ table which had been filled with 1000000 random records. The weight of T_{old} in 5 is initialized by 0.75. The minimum part 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 proposed approach in three modes, when SP always generates a correct answer or it generates correct result with 75%,50%, or 25% probability.

Figure 4(a) indicates that the execution time of our approach. 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 that are presented in the query condition, but proposed approach does not rely on sort. Result verification in MHT-based approach contains executing several hash and concatenation functions and verifying the root signature. On the contrary, result verification in our approach are not constructed with cryptographic tools which makes our approach more light-weighted. One of the advantages of proposed approach is that its overhead is too low when SP often generates correct answers.

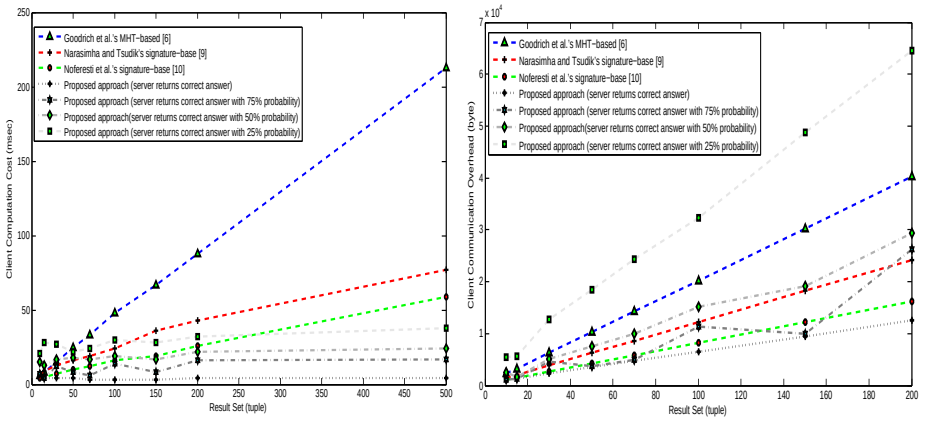

Fig. 4. Client computation cost(a) and communication overhead(b) of different approaches.

Figure 4(b) compares the communication overhead of different approaches. 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 value 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 high. In our approach when U receive an incorrect result, the query is resubmitted to SP and the result will be received. So 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 care. We proposed a novel approach which verifies the correctness of returned results efficiently.

The 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 part is determined by the trust value in SP. In the proposed approach, 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 had been chosen by C. If this part is not in the bloom filters, the entire returned result will be rejected. Otherwise, the returned result is accepted by a probability, comes up 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 spec-

ifying linear time properties. The soundness and completeness properties of the verification algorithm are described by LTL. Finally, we show that the transition system of the proposed approach satisfies these properties.

An applicable and efficient data outsourcing approach is desired. For this, not only the overhead of approach is important but also it must support a wide range of queries. So, we plan to study another data structures types instead of ICBF which can be used to support the correctness verifiability of other more advance query types.

References

1. Agrawal, D., Abbadi, A.E., Emeki, F., Metwally, A.: Database management as a service: Challenges and opportunities. In: ICDE. pp. 1709–1716. IEEE (2009)
2. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. vol. 13, pp. 422–426 (1970)
4. Curino, C., Jones, E., Popa, R.A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N.: Relational Cloud: A Database Service for the Cloud. In: 5th Biennial Conference on Innovative Data Systems Research. Asilomar, CA (January 2011)
5. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-efficient verification of dynamic outsourced databases. In: CT-RSA. Lecture Notes in Computer Science, vol. 4964, pp. 407–424. Springer (2008)
6. H. Hacigumus, B.I., Mehrotra, S.: providing database as a service. In: International conference on data engineering. Lecture Notes in Computer Science, vol. 7093, pp. 29–38. Springer (2002)
7. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD Conference. pp. 121–132. ACM (2006)
8. Narasimha, M., Tsudik, G.: Dsac: An approach to ensure integrity of outsourced databases using signature aggregation and chaining. IACR Cryptology ePrint Archive 2005, 297 (2005)
9. Noferesti, M., Hadavi, M.A., Jalili, R.: A signature-based approach of correctness assurance in data outsourcing scenarios. In: ICISS. Lecture Notes in Computer Science, vol. 7093, pp. 374–378. Springer (2011)
10. Pang, H., Tan, K.L.: Authenticating query results in edge computing. In: ICDE. pp. 560–571. IEEE Computer Society (2004)
11. Pang, H., Zhang, J., Mouratidis, K.: Scalable verification for outsourced dynamic databases. PVLDB 2, 802–813 (2009)
12. Samarati, P., di Vimercati, S.D.C.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. pp. 1–14. ASIACCS '10, ACM, New York, NY, USA (2010)
13. Sion, R.: Query execution assurance for outsourced database. pp. 601–612. VLDB Endowment (2005), 148051
14. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of the 33rd international conference on Very large data bases. pp. 782–793. VLDB '07, VLDB Endowment (2007)