Correctness Verification in Outsourced Databases: More *Reliable* Fake Tuples Approach

Ganugula Umadevi and Ashutosh Saxena

Infosys Labs, Infosys Limited, Hyderabad, India
{Uma_Ganugula,Ashutosh_Saxena01}@Infosys.Com

Abstract. Enterprises outsource data to storage service providers in order to avail resources at lower costs and for ensuring economy of scale. Prime concerns of organizations are privacy of data and quality of service provided, which include correctness of query results and integrity of data. In earlier works, integrity of data is verified by preserving hash of tuple data as *header* attribute and completeness is assured by inserting and retrieving fake tuples. In this work, we propose a secret sharing based approach for deterministic generation of fake tuples used for verifying completeness and integrity of data, where we eliminate the dependency on header attribute. Our integrity check mechanisms work faster than existing approaches in this direction as depicted in our experimental results. Furthermore, we show that our approach is information theoretically secure, where an adversary without knowledge of underlying security parameters can never be able to break the scheme.

Keywords: Database Outsourcing, Query Integrity, Fake Tuples.

1 Introduction

The necessity for economical solutions for organizations drive them towards options like Storage as a Service (SaaS) [1]. SaaS provides the opportunity to preserve the organization's data in external servers at a reasonable cost for storage. However, the benefits of this facility do not stop the organizations from being skeptical about the security and privacy issues of their outsourced data. To provide security to the data, organizations usually opt for encrypting critical data before it is uploaded to the third party servers. They employ indices to search upon the encrypted data. Another aspect of concern for organizations is the integrity of query results, which means the results are accurate and data is not altered at the server's end. This process of checking for data integrity is referred to as correctness verification of the outsourced databases.

In correctness verification of outsourced databases, three aspects are considered: a) Completeness b) Freshness c) Integrity. Completeness ensures that the query results are correct and complete. Freshness ensures that the query results are up to date. Integrity ensures that the data is not tampered in transit or at rest. In the literature, there are multiple approaches for addressing the problem of verifying correctness of query results in outsourced databases. In [2–4], hash of encrypted data is computed and stored, for checking integrity. Since, any change in the encrypted data is reflected in the hash, changes in original data are detected. In [5], the database is restructured as a Merkle-Hash tree for verification. In [6], hash of the tuple is computed and stored as an extra attribute in each tuple. For verifying correctness, the solution proceeds by insertion of check points in the database. These check points, which are referred to as *fake tuples or artificial tuples*, are created and inserted into the database along with the original data. When query results are retrieved, the number of fake tuples returned are compared against the number of tuples originally inserted; change in the count value of fake tuples indicate adversarial changes to original data [6]. In [7], random and deterministic ways to generate fake tuples are proposed. There are techniques which use secret sharing [8] for generating fake tuples. Hash based techniques are mainly used for integrity check. The fake tuple techniques are used for completeness verification of query results.

The drawbacks of the solutions discussed in literature are (a) approaches discussed in [5, 13] are computationally heavy or (b) fail to ensure integrity for complete query results. In [6], an extra attribute holding hash is used to identify tuple/data integrity. An adversary who has knowledge of this scheme can delete the entire attribute and disturb the integrity check process completely. Approaches for completeness discussed in [7], focus on the verification of the retrieved fake tuples for completeness check and discusses majorly on building trust on the storage server. Approaches discussed in [5] require fake tuples be stored at the organization's desk, which is space-consuming and is conflicting with the idea of using SaaS.

In the data outsourcing scenario, we propose an approach to ensure integrity for the entire query result set. Unlike existing approaches, data for integrity check is not stored as an explicit attribute but is stored as a tuple along with the original data; so that it can not be easily tampered. We ensure information theoretic strength for the approach, when the security parameters are not disclosed to an adversary. The computation and communication overhead of our approach are comparable to existing works [6]. For completeness, we use the fake tuple based approach which can be used to build trust [7]. Unlike the work of [8], we use data available in the database along with the concept of secret sharing to generate fake tuples which can be used for integrity and completeness check; and the proposed approach is storage-efficient.

Paper Organization. In section 2, we explain the notations and well-known techniques that we have used in the work. Section 3 explains the approach for correctness verification, where we discuss how integrity and completeness are checked for outsourced databases. In section 4, we analyze the functionality of the approach, in formal terms for strength and security aspects, cost and computation effort and also in terms of features and limitations with respect to handling database updates. We conclude and discuss future work in section 5.

2 Notations and Preliminaries

We refer to the SaaS providers as servers S and organizations storing data at S as the client C. Let Q be test query, which is associated with the fake tuples. We denote the result set of query Q as R_Q. $T_1, T_2, \ldots, T_n$ are the tuples returned by Q, where n is a symbolic indication of the number of tuples returned by Q. $A_1, A_2, \ldots, A_\omega$ are the attributes, where A_p is the primary key attribute of the table. Without loss of generality, we assume that values of attributes $A_1, A_2, \ldots, A_\omega \in \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers and the domain for $A_1, A_2, \ldots, A_\omega$. Table 1 gives the list of notations used in rest of the paper.

Table 1. Notations

Notation	Description
PRG	Pseudo random number generator
V_{F_T}	Verifier Fake Tuple
S_{F_T}	Supporting Fake Tuples
L	Set of storage locations of the supporting fake tuples
k	Count of tuples used for V_{F_T} generation
δ	Count of supporting fake tuples inserted into the database

Shamir Secret Sharing: Shamir Secret Sharing(SSS)[11] is a cryptographic technique where a secret value is distributed among a group of players. Each player gets a share which is computed by evaluating a polynomial at the unique player *id*. In SSS, a polynomial $f(.)$ is considered such that, for the threshold of k players (k players can reconstruct the secret), a $k-1$ degree polynomial is constructed such that $f(0) = s$, where s is the secret to be shared amongst the players. When k players come together, the polynomial is reconstructed and secret value is retrieved from the polynomial. SSS is information theoretically secure. The strength of SSS comes from theorem 1. The proof of the theorem discussed in [9, 10] plays a critical role for ensuring unique reconstruction property of Shamir secret sharing scheme [11]. We use the secret sharing technique in our fake tuple generation process discussed in the later sections.

Theorem 1. (Lagrange interpolation theorem). Let $\mathbb{R}$ be a field and $a_0, \ldots, a_n$, $y_0, \ldots, y_n \in \mathbb{R}$ so that all values a_i are distinct. Then there exists only one polynomial f over $\mathbb{R}$ so that $deg(f) \leq n$ and $f(a_i) = y_i$, $(i = 0; \ldots; n)$.

3 Query Integrity Verification: The Approach

The idea is to insert fake tuples along with original data in the database, such that integrity and completeness of result set for a given query (R_Q) is verified. In this section, we discuss the approach for generating the fake tuples for numeric data in a deterministic way. For non-numeric data types like varchar, char, string,

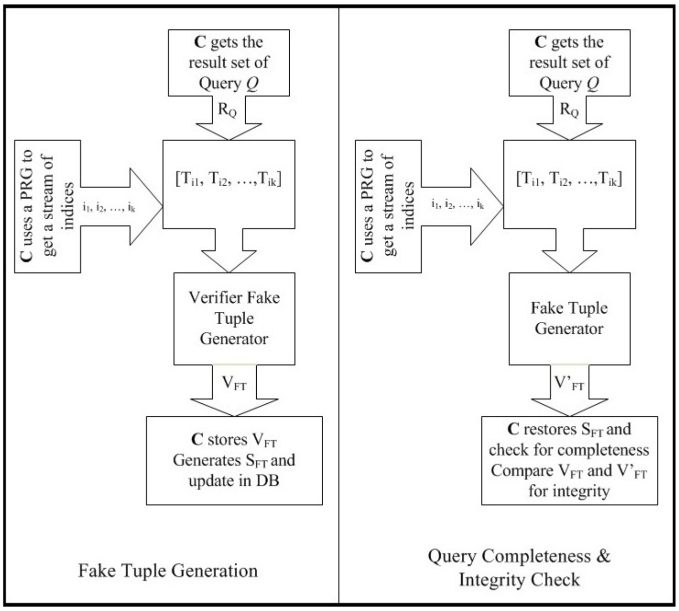
C gets the result set of Query Q
R_Q
C uses a PRG to get a stream of indices
i1, i2, ..., ik
[Ti1, Ti2, ...,Tik]
Verifier Fake Tuple Generator
V_FT
C stores V_FT Generates S_FT and update in DB
Fake Tuple Generation
C gets the result set of Query Q
R_Q
C uses a PRG to get a stream of indices
i1, i2, ..., ik
[Ti1, Ti2, ...,Tik]
Fake Tuple Generator
V'_FT
C restores S_FT and check for completeness Compare V_FT and V'_FT for integrity
Query Completeness & Integrity Check

Fig. 1. Query Integrity and Completeness Verification

blob, they can be converted to numeric by doing a character-wise ascii conversion for char datatypes and by doing base conversion for blob datatypes, based on design requirements.

Figure 1 shows the outline of the approach. When the client C submits the query Q to the server S, then the server returns the result set R_Q. We create a set $PA = \{A_{p1}, A_{p2}, \ldots, A_{p\tau}\}$, where the elements A_{pi} are the primary key values of of i^{th} tuple of R_Q, and τ indicates the number of tuples in R_Q. We pick a non-empty subset of the result set R_Q using a pseudo random generator PRG to get some indices in PA. We observe that, dependency on PA and A_p can be reduced by associating a dummy id to the chosen tuples and by storing the hash of the tuple for its unique identification from R_Q. In detail, the PRG generates a stream of indices, from which we select first k values $(i_1, i_2, \ldots, i_k)$ which lie in the set PA. From the result set R_Q, the tuples corresponding to these indices are retrieved. These tuples are input to the Fake Tuple Generator, which produces the verifier fake tuple V_{F_T}. V_{F_T} is used to generate the supporting fake tuples S_{F_T}. This V_{F_T} is split into supporting fake tuples S_{F_T} according to a deterministic function. These supporting fake tuples are inserted into the database at locations L. The query Q, the PRG with its seed, the value k, the fake tuple V_{F_T}, location set L and functions associated with S_{F_T} are stored at C.

In the verification phase, V'_{F_T} is generated and checked against the stored values for query completeness and integrity as explained in tables 2,3 and 4. While computing V_{F_T} and S_{F_T}, the queries of generic nature are chosen, such that they can be applied anytime for insert into a given table. An example query is, SELECT * FROM *Table* where $A_i \geq V_1$ *and* $A_i \leq V_2$, where V_1 and V_2 are values in the range of A_i. When an integrity check needs to be performed, a random query can be given to the database along with the clause on A_i. Since the results returned by random query has the data used for V_{F_T} and S_{F_T}, the returned results can be verified. Based on this kind of range queries, for every instance of database insert update, fake tuples can be created and included into the original data before inserting into the original database. C being the owner of the data, decides the queries applicable for integrity check.

Table 2. Verifier Fake Tuple Generation

Method for generating Verifier Fake tuple
1. For each of $T_{i1}, T_{i2}, \ldots, T_{ik}$, C collects the A_x,A_p value pairs to form points $P_{i1}, P_{i2}, \ldots, P_{ik}$; where point P_{ij} = *(attribute* A_p *value of* R_{ij}*, attribute* A_x *value of* R_{ij}*).*
2. $\mathcal{F}_1$ can be explained as: Using a curve fitting algorithm, C interpolates these points to get a $k-1$ degree polynomial.
3. A_x of V_{F_T} is assigned the value of the constant term of the resulting polynomial.

3.1 Fake Tuple Generator

Verifier Fake Tuple Generation. Given the result set of the query Q, R_Q and the chosen indices from the stream of indices generated from the PRG, we explain the deterministic approach used by the client C to generate the fake tuple. Let $T_{i1}, T_{i2}, \ldots, T_{ik}$ be the tuples chosen for creating the fake tuple. As defined earlier, $A_1, A_2, \ldots, A_\omega$ are the attributes of the table, where A_p is the primary key attribute. To generate attribute A_x for V_{F_T}, the function $\mathcal{F}_1 : (\mathbb{R}, \mathbb{R})^k \rightarrow \mathbb{R}$, where $x \in \{1, \ldots, \omega\} \backslash \{p\}$ is an indicator of the location in the list of attributes, the method used by C is given in table 2.

Supporting Fake Tuple Generation. Once the verifier fake tuple is constructed, C uses this V_{F_T} to construct the supporting fake tuples. In table 3, we discuss the supporting fake tuple generation using secret sharing schemes, whereas in table 4, we use *PRG* for generating S_{F_T}. Here $V_{F_T}^{A_x}$ is the A_x attribute of V_{F_T}.

3.2 Integrity Check of Database

In the scenario where the database has not undergone any updates, the process for verifying the result set for query Q is explained as follows:

Table 3. Secret Sharing Based S_{F_T} Generation

Method for generating Supporting Fake tuples
1. C chooses a random value δ based on his choice of number of supporting fake tuples. 2. C computes $S_{F_T}^{A_x} = f_{A_x}(i)$, where $f_{A_x}(0) = V_{F_T}^{A_x}$. $S_{F_T}^{A_x}$ are shares generated by SSS using $f_{A_x}(.)$, which is a polynomial of degree $\delta - 1$ defined over $\mathbb{Z}_p$, where p is the prime closer to $max(A_x)$. 3. These $S_{F_{Ti}}$ form the supporting fake tuples, which are inserted into the database. **Reconstructing** V_{F_T}**:** For the chosen query Q, C gets R_Q from the database and using the *secret reconstructing* algorithm, V_{F_T} is reconstructed from δ S_{F_T}s.

1. C submits the query Q and obtains the result set R_Q. C gathers all the A_p values of tuples in the resulting set.
2. Using the PRG definition along with the seed, it generates and gathers k indices falling in the A_p set in a deterministic way. Using the Fake Tuple Generator, it generates the fake tuple V'_{F_T}.
3. Once the original verifier fake tuple V_{F_T} is extracted, it is compared against V'_{F_T}. For each of the attributes, other than A_p, if $V'_{F_T} = V_{F_T}$, it indicates that the integrity of query results is preserved.

3.3 Completeness Check of Database

Completeness of the query results is checked using the supporting fake tuples. In the scenario where the database has not undergone any updates, we explain the completeness verification process for result set of Q.

1. C fires the query Q and obtains the result set R_Q.
2. Using the location information L, C gathers all supporting tuples, $S_{F_{Ti}}$, and reconstructs V'_{F_T} using the techniques discussed in table 3 and table 4.
3. C verifies V'_{F_T} against V_{F_T}, which is stored at his end. For each of the attributes if V_{F_T} and V'_{F_T} has same values, it indicates that the query results are complete.

In both the cases, where some of the tuples are modified or removed at the database end, it can be observed at step 3 of the verification process, where $V'^{A_x}_{F_T}$ generated is different from the stored $V^{A_x}_{F_T}$.

4 Analysis

4.1 Functioning of Proposed Approach

We use the following lemmas to prove that the proposed approach is functionally correct as per the requirements.

Table 4. *PRG* Based S_{F_T} Generation

Method for generating Supporting Fake tuples
1. C chooses a random value δ based on his choice of number of supporting fake tuples.
2. C represents $V_{F_T}^{A_x} = \oplus_{i=1}^{\delta} S_{F_{Ti}}^{A_x}$,where $S_{F_{Ti}}$ are random value chosen from $\mathbb{R}$, for each attribute A_x in V_{F_T}.
3. These $S_{F_{Ti}}$ form the supporting fake tuples, which are inserted into the database.
Reconstructing V_{F_T}**:** For the chosen query Q, C gets R_Q from the database and using the Step 2, V_{F_T} is reconstructed from δ S_{F_T}s.

Lemma 1. *Changes in R_Q will result in changes in V_{F_T}.*

Proof. Assume to the contrary, that even if the tuples in R_Q change, the values in V_{F_T} remain same. This means different points can end up at the same polynomial which contradicts Theorem 1. This proves that changes in R_Q results in changes in V_{F_T}.However, since we consider constant term of the resulting polynomial, there is a negligible probability that the constants remain the same for different unique polynomials. The probability is $\frac{1}{(q-1)(q)^{k-2}}$, for a k-degree polynomial and q is the field size which is $|\mathbb{R}|$ in our case. □

Lemma 2. *When $V_{F_T} = V'_{F_T}$ then the database records preserve integrity.*

Proof. Without chance in notations, V_{F_T} is the verifier fake tuple originally stored at C's end and V'_{F_T} is the fake tuple generated at a later point of time, to verify integrity of database records. In this setup, if $V_{F_T} \neq V'_{F_T}$, it indicates that at least one of the tuples have been tampered with. This indicates that the data is not same as the original data and hence, database records lost their integrity. □

Lemma 3. *When V_{F_T} can be reconstructed from S_{F_T}, R_Q is complete.*

Proof. When secret sharing approach is used, theorem 1 ensures that for a given set of δ S_{F_T}s, V_{F_T} is unique. Hence, with the same success probability of integrity check as discussed in section 4.2, R_Q is complete. □

4.2 Success Probability

In this part of work, we discuss the success probabilities of the integrity check mechanism as well as successful escape probability of the adversary.

Success of Integrity Check: There are two levels at which integrity check success can be discussed: a) query result integrity b) database integrity. In a given instance, when k out of n of R_Q are used for generating V_{F_T}, query results preserve integrity with probability $\frac{k}{n}$. For sure verification, all the n tuples in R_Q need to be used to generate V_{F_T}. In the case, where entire database integrity has to be verified, instead of a single query, multiple queries have to be considered for V_{F_T} generation. Let r be number of tuples covered by all the possible queries in V_{F_T} generation process from y tuples present in the database, then database record integrity is preserved with probability of $\frac{r}{y}$.

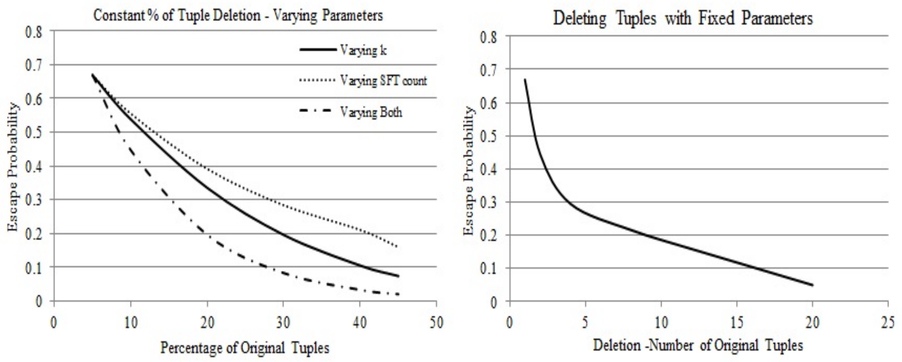

Fig. 2. Escape Probability of Adversary

Escape Probability of Adversary: An adversary can modify(update or delete) some tuples from the database and still get unnoticed. The escape probability of the adversary is the success with which he can delete a tuple from the database which does not belong to the k tuples associated with V_{F_T} generation, and also which do not belong to S_{F_T}s. For a given instance, if Q returns n tuples, which has δ S_{F_T}s and k out of n are used to construct V_{F_T}, then the escape probability of adversary for deleting *one* tuple and being unnoticed is $\frac{n-k}{n+\delta}$. This value decreases exponentially when multiple fake tuples are deleted, depending on both k and δ. Figure. 2 shows how the escape probability reduced to negligible value when $|S_{F_T}|$ varies as a percentage of n. In figure 2(a), the number of fake tuples deleted is fixed and the security parameters are varied; and the decrease in escape probability is observed. It can be observed the when both the security parameters k and δ are set to a small percentage of count of original tuples, the escape probability of adversary becomes negligible. Similarly in figure 2(b), the security parameters are fixed and increasing the number of fake tuple deletions, escape probability is observed.

4.3 Cost Analysis

Here, we analyze the communication and computation cost incurred at C using the proposed approach.

Communication Cost: The cost of communication is the cost incurred in sending a tuple to a outsourced database. If Φ is the cost incurred for sending a tuple to a outsourced database and if C decides to have δ fake tuples for a query Q, then the cost incurred in $\delta.\Phi$.

Computation Cost: At C's end, the following computation costs are involved. Some computations like *generation* are performed once in the query integrity process whereas other computations like *verification* are performed everytime integrity check is done.

Verifier Fake tuple Generation: For a given query, the computing for V_{F_T} is equivalent to getting the constant term for the polynomial coming from a set of points. Getting the constant term involves evaluating a simple expression, which involves mere substitution of x and y values.

$$constant = \sum_{i=1}^{k} \frac{(-1)^k . \prod_j^{k;j\neq i} x_j}{\prod_j^{k;j\neq i}(x_i - x_j)} * y_i$$

Supporting Fake tuple Generation: For a given query Q, the number of fake tuples inserted depends on the client's demand for robustness. For a given query, computing δ fake tuples as discussed in table 3 and table 4 involves two types of computations. Using the first technique involves triggering $\delta - 1$ PRGs with their respective seeds to produce $\delta - 1$ random values, identifying the remaining S_{F_T} value involves simple *xor* of generated random values with corresponding value in V_{F_T}. This is also a constant time operation. Using the second technique of using *secret sharing* polynomial, generating supporting fake tuples involves evaluating various polynomials at different values. Evaluating a polynomial is also a constant time operation.

Verifier Fake Tuple Verification: For any given query Q, cost of retrieving R_Q is a communication cost. After retrieving R_Q, the fake tuple generation involves constant time operation as explained in fake tuple generation process.

Supporting Fake Tuple Verification: After retrieving R_Q which has the supporting fake tuples, using the same operations discussed in supporting fake tuple generation process, verification is done. This verification involves performing ex-or or constructing V_{F_T}, which are constant time operations.

4.4 Security Analysis

In this section, we prove the security of our system in an *ideal* setting. The data outsourcing scheme is considered broken if an adversary can reconstruct the polynomial used either in the generation of the verifier fake tuple or in the generation of supporting fake tuples.

Lemma 4. *The scheme is information-theoretically secure, that is, an adversary with unlimited computing power also, can not break the scheme.*

Proof. It is well known that the secret sharing schemes such as Shamir's are information theoretically secure, in that, less than the requisite number of shares of the secret provide no information about the secret or equivalently the polynomial used in the generation of the shares. As the number of shares required for reconstructing the polynomial used either in the verifier fake tuple/supporting fake tuples is unknown to the adversary (both k and δ are secret), even an adversary with unlimited computational power cannot break the scheme. Even if we assume that k and δ are known to the adversary, he cannot distinguish whether the value of an attribute in a tuple is a share or any arbitrary value in the domain of that attribute. Therefore, our scheme is information-theoretically secure. □

In the case of outsourced databases, the possible adversary is a curious server who would like to differentiate between the fake tuples and normal data. Since the server could not differentiate the tuples without the knowledge of δ and k, the S_{F_T} tuples are indistinguishable from the normal data and hence, the server can not treat the fake tuples seperately. For a new query, when the fake tuples are inserted along with the normal data, following this theorem, the server can not differentiate between a normal data insert and insert with fake tuples.

4.5 Empirical Evaluation

We used a PC with AMD Athlon IIX2 processor and 4GB RAM to act as C and S. The approach is simulated using Java language and Microsoft SQL Server DBMS is used to store the outsourced tuples. A table with medical records is used as test table with five attributes namely Record-ID, BP-Observed, Sugar-Observed, Weight and Height. C fires the queries to S, which hosts the table having 1 million tuples. Fake tuple sets of different sizes are constructed with respect to different *range* queries and inserted into the database. We implemented our approach and Hash Attribute approach discussed in [6] and observed the difference in processing time at C's end for completeness check which is seen in Figure 3(b). Our approach has comparable results for processing time at client. We also analyzed the efforts for generating supporting fake tuples at the client using the two deterministic approaches discussed in table 3 and table 4, results on computation effort at C are shown in Figure 3(a).

4.6 Features and Limitations

Here, we discuss the limitations and features of our approach.

Handling Database Updates: Updates to a database include insert, delete and drop. The client has to ensure that the appropriate update is reflected at the server. We discuss how each update affects the query integrity process and how to ensure integrity despite running updates on database.

Insert Update. In insert operation, new tuples are inserted into the database. This process may not effect the existing integrity related tuples. However, the process discussed in section 2 , should be applied again for these tuples as well, if there is a noticeable amount of data inserted into the database. If the insert changes will be reflected in the result set R_Q of a query Q which is covered by C, then supporting fake tuples can be inserted according to technique (a) or (b) discussed in generating supporting fake tuples. Since this leads to a change from δ to δ_1, C updates the new δ_1 along with Q.

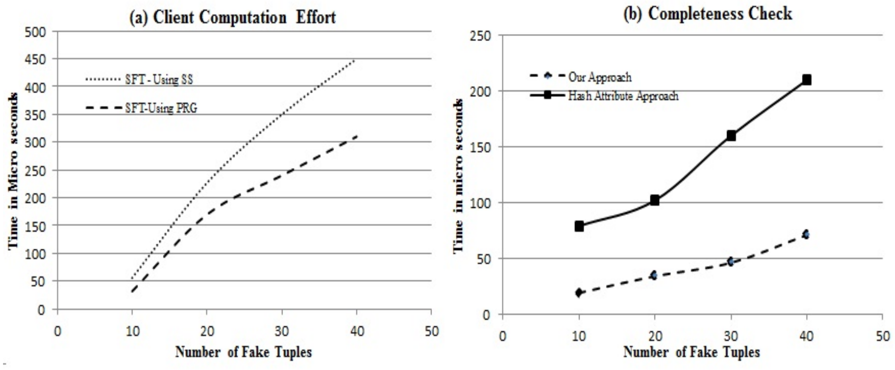

Fig. 3. Computation Effort at Client

Delete Update. Delete is a complex operation to incorporate in the existing setup, because delete could lead to deletion of S_{F_T} tuple associated with V_{F_T} at C. Since locations associated with S_{F_T} are known to C, it can verify the delete query before executing and exclude deletion of fake tuples.

5 Conclusion and Future Work

Earlier works in completeness verification in outsourced databases ensured query integrity by adding an extra attribute to the table. However, deleting this attribute hinders integrity check. In this work, we eliminated the dependency on adding new attribute. Integrity check and also, completeness check are ensured by using fake tuples. We proposed two deterministic ways to generate fake tuples. These generated fake tuples are used for completeness check. Our integrity check mechanism works faster than existing approaches, as depicted in our empirical evaluation. And also discussed the aspects of cost and computation complexity of the approach along with its features and limitations as well. We showed that our approach is information theoretically secure, where an adversary without knowledge of the underlying security parameters can never be able to break the scheme. Future work involves utilization of the proposed approach for public auditing in cloud.

References

1. Hacigumus, H., Iyer, B.: Providing Database as a Service. In: International Conference of Data Engineering (2002)
2. Narasimha, M., Tsudik, G.: Authentication of Outsourced Databases Using Signature Aggregation and Chaining. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 420–436. Springer, Heidelberg (2006)
3. Noferesti, M., Hadavi, M.A., Jalili, R.: A Signature-Based Approach of Correctness Assurance in Data Outsourcing Scenarios. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2011. LNCS, vol. 7093, pp. 374–378. Springer, Heidelberg (2011)
4. Zhu, Y., Wang, H., Hu, Z., Ahn, G.-J., Hu, H., Yau, S.S.: Dynamic audit services for integrity verification of outsourced storages in clouds. In: Proceedings of the ACM Symposium on Applied Computing (SAC 2011), pp. 1550–1557. ACM, New York (2011)
5. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-Efficient Verification of Dynamic Outsourced Databases. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 407–424. Springer, Heidelberg (2008)
6. Xie, M., Wang, H., Yin, J.: Integrity Auditing of Outsourced Data. In: Conference on Very Large Databases, VLDB (2007)
7. Ghasemi, S., Noferesti, M., Hadavi, M.A., Dorri Nogoorani, S., Jalili, R.: Correctness Verification in Database Outsourcing: A Trust-Based Fake Tuples Approach. In: Venkatakrishnan, V., Goswami, D. (eds.) ICISS 2012. LNCS, vol. 7671, pp. 343–351. Springer, Heidelberg (2012)
8. Hadavi, M.A., Jalili, R.: Secure Data Outsourcing Based on Threshold Secret Sharing; Towards a More Practical Solution. In: Proceeding of VLDB PhD Workshop, pp. 54–59 (2010)
9. Waring, E.: Problems concerning interpolations. Philosophical Transactions of Royal Society 69, 59–67 (1779)
10. Lagrange, J.L.: Leons lmentaires sur les mathamatiques donnes a lcole normale. In: Serret, J. (ed.) OEuvres de Lagrange, Paris, France, vol. 7, pp. 183–287 (1877)
11. Shamir, A.: How to share a secret. Communications of the ACM 22, 612–613 (1979)
12. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
13. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In: Proceedings of INFOCOM, pp. 1–9 (2010)