2015 IEEE 28th International Symposium on Computer-Based Medical Systems

Secure and Private Management of Healthcare Databases for Data Mining

Noman Mohammed*, Samira Barouti†, Dima Alhadidi‡and Rui Chen§
*Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada
†Ericsson Canada Inc., Montreal, Quebec, Canada
‡College of Technological Innovation, Zayed University, Dubai, United Arab Emirates
§Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong

***Abstract*—There has been a tremendous growth in health data collection since the development of Electronic Medical Record (EMR) systems. Such collected data is further shared and analyzed for diverse purposes. Despite many benefits, data collection and sharing have become a big concern as it threatens individual privacy. In this paper, we propose a secure and private data management framework that addresses both the security and privacy issues in the management of medical data in outsourced databases. The proposed framework ensures the security of data by using semantically-secure encryption schemes to keep data encrypted in outsourced databases. The framework also provides a differentially-private query interface that can support a number of SQL queries and complex data mining tasks. We experimentally evaluate the performance of the proposed framework, and the results show that the proposed framework is practical and has low overhead.**

I. Introduction

Healthcare industries store a massive amount of sensitive personal data, such as patient names, dates of birth, and personal medical records. Since healthcare data doubling every year, organizations need to invest in both hardware and software to store and manage large amount of data. In this domain, cloud computing is an effective solution for healthcare companies to handle huge amounts of medical records. However, healthcare organizations face two technical challenges.

First, data outsourcing exposes sensitive healthcare data to untrusted cloud service providers. Unauthorized access to sensitive medical records can have a significant negative impact on healthcare services. To ensure the confidentiality of the medical data stored on the cloud, we should depend on semantically-secure encryption schemes. Using semantically-secure encryption schemes, it must be infeasible for a computationally-bounded adversary to derive significant information about a message when given only the ciphertext and the corresponding public key. In this regard, the challenge is how to ensure data confidentiality while allowing query execution over encrypted data.

Second, driven by mutual benefits and regulations, there is a demand for healthcare organisations to share patient data with various parties for research purposes. Healthcare organization may allow data analysts (e.g., researchers) to execute aggregate queries and perform some data analysis tasks (e.g., classification analysis) on the database. In this regard, the challenge is how to support aggregate queries or complex data mining tasks on encrypted data while preventing inference attacks [1].

There have been a lot of research proposals that separately address these two challenges . Most of the previous proposals on secure outsourced databases suggest encrypting the data before moving it to the cloud [2]. While encryption can provide data confidentiality, it is of little use in deterring inference attacks [1]. Similarly, there is an extensive literature on private data analysis [3]. However, all these proposals require access to unencrypted data to generate privacy-preserving answers and therefore do not satisfy the data confidentiality requirement. This reality demands a new privacy-enhancing technology that can simultaneously provide data confidentiality against an untrusted database server, and prevent inference attacks from data analysts.

In this paper, we propose a general framework for secure and private data management in order to support effective data mining. The contributions of the paper are summarized as follows:

- Based on real-life healthcare scenarios, we identify a new problem of secure and private data management of outsourced databases for data mining purposes (Section II).
- We adopt a new privacy-enhancing protocol that can provide data confidentiality against an untrusted cloud servers by using semantically-secure encryption schemes. The protocol has been proposed by Barouti *et al.* [2] (Section III). We then extend the protocol to support aggregate queries or complex data mining tasks on encrypted data while preventing inference attacks.
- Taking decision tree learning as an example, we show that it is possible to compute a classifier on the encrypted data (Section IV). The computed classifier provides differential privacy guarantee to prevent any inference attack. The proposed private decision tree learning algorithm uses the secure query execution protocol to obtain counts of the encrypted data, adds noises to these counts, and generates probabilistically a differentailly-private classifier.
- We implement the proposed framework and evaluate its performance (Section V). Experimental results on

2372-9198/15 $3.00 © 2015 IEEE
DOI 10.1109/CBMS.2015.54

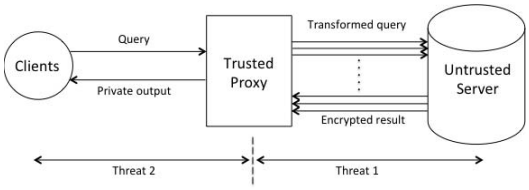


Figure 1. System Overview

real-life data suggest that our solution is effective and scalable.

II. Preliminaries

In this section, we first present an overview of the system architecture as shown in Figure 1. We then detail the privacy and security models of the proposed system.

A. Overview

The proposed system works by intercepting all queries from clients using a proxy, which executes queries with the help of the database owner. To ensure confidentiality, data is stored in an encrypted format in cloud servers and queries are executed on the encrypted data. The proxy holds the private key, and can encrypt/decrypt any data. It can also change different parameters of a query issued by a client. The cloud server never receives the private key, and therefore, it does not gain access to the sensitive information in the database (illustrated by Threat 1 in Figure 1).

A query could be a simple aggregate query or a complex data mining task such as classification analysis. Queries are issued by clients with different levels of authorization. For example, a physician is different from a data analyst. A data analyst is only allowed to issue aggregate queries and receives differentially-private (see below) answers. Upon receiving a query request from a client, the proxy first checks the client's authorization and then forwards the request to the cloud server. The server returns an encrypted answer, which is decrypted at the proxy. Finally, the proxy either returns the true answer or adds appropriate noise (if the client is a analyst) to provide differential privacy guarantee. This ensures that a curious data analyst cannot launch any inference attack against the database (illustrated by Threat 2 in Figure 1).

B. Security Model

The cloud server is considered malicious and should not have access to the private key. The proxy on the other hand is trusted and managed by the data owner. It holds the private key and does not leak the key to the cloud server server as it is clearly against the interest of the data owner. Real life applications are based on the three-tier architecture, where application and database servers are two different entities. The proxy is collocated with the application server, whereas the database server is outsourced to the cloud that may reside in a different geographical location.

C. Privacy Model

Differential privacy is a recent privacy definition that provides a strong privacy guarantee. It guarantees that an adversary learns nothing more about an individual, regardless of whether her record is present or absent in the data.

Definition 2.1 (ϵ-differential privacy [4]): A randomized algorithm Ag is differentially private if for all data sets D and D' where their symmetric difference contains at most one record (i.e., $|D \triangle D'| \leq 1$), and for all possible anonymized data sets $\hat{D}$,

$$\Pr[Ag(D) = \hat{D}] \leq e^{\epsilon} \times \Pr[Ag(D') = \hat{D}], \tag{1}$$

where the probabilities are over the randomness of the Ag. ■

The parameter $\epsilon > 0$ is public and specified by a data owner. Lower values of ϵ provide stronger privacy guarantee.

A standard mechanism to achieve differential privacy is to add random noise to the true output of a function. The noise is calibrated according to the *sensitivity* of the function. The sensitivity of a function is the maximum difference of its outputs from any two data sets that differ only in one record.

*Definition 2.2 (*Sensitivity*):* For any function $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1 \tag{2}$$

for all D, D' differing in at most one record. ■

Laplace Mechanism. Dwork et al. [5] propose the Laplace mechanism. The mechanism takes a data set D, a function f, and the parameter λ that determines the magnitude of noise as inputs. It first computes the true output $f(D)$, and then perturbs the output by adding noise. The noise is generated according to a Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\exp(-|x|/\lambda)$; its variance is $2\lambda^2$ and mean is 0. The following theorem connects the sensitivity to the magnitude of noise and guarantees that perturbed output $f(\hat{D}) = f(D) + \texttt{Lap}(\lambda)$ satisfies ϵ-differential privacy, where $\texttt{Lap}(\lambda)$ is a random variable sampled from the Laplace distribution.

Theorem 2.1: [5] For any function $f : D \rightarrow \mathbb{R}^d$, the algorithm Ag that adds independently generated noise with distribution $\texttt{Lap}(\Delta f/\epsilon)$ to each of the d outputs satisfies ϵ-differential privacy.

Exponential Mechanism. McSherry and Talwar [6] propose the exponential mechanism that can choose an output $t \in \mathcal{T}$ that is close to the optimum with respect to a utility function while preserving differential privacy. The exponential mechanism takes as inputs a data set D, output range $\mathcal{T}$, privacy parameter ϵ, and a utility function $u : (D \times \mathcal{T}) \rightarrow \mathbb{R}$ that assigns a real valued score to every output $t \in \mathcal{T}$, where a higher score means better utility.

The mechanism induces a probability distribution over the range $\mathcal{T}$ and then samples an output t. Let $\Delta u = \max_{\forall t,D,D'} |u(D,t) - u(D',t)|$ be the sensitivity of the utility function. The probability associated with each output

is proportional to $\exp(\frac{\epsilon u(D,t)}{2\Delta u})$; that is, the output with a higher score is exponentially more likely to be chosen.

III. Secure Data Management

In this section, we present a quick overview of the protocol proposed by Barouti *et al.* [2] to protect database confidentiality and enable a client to execute SQL queries over the encrypted database without decryption. The proposed protocol handles aggregate queries over encrypted databases stored on the cloud. The protocol uses Fischlin's protocol [7] that allows comparing two ciphertexts encrypted with the Goldwasser-Micali (GM) crytosystem [8] using the same public key. Query predicates are evaluated using the GM cryptosystem and Fischlin's protocol whereas query aggregate functions are computed using the Paillier cryptosystem [9]. In the remaining of this section, we describe the basic steps of our protocol (see [2] for details).

1) Tree Construction: The data owner organizes the records of the database as a left-balancing kd-tree. Left-balancing binary tree makes it possible to store a binary tree structure in an array [10]. Using this scheme, the root node would be stored in the position 1 in the array. In addition, if a node is stored in the position n in the array, the left child node and the right child node are stored in the positions $2n$ and $2n + 1$, respectively. The resulted tree is then traversed and stored in an index-based table, namely A. The outsourced table is then generated by encrypting the existing columns using the GM and Paillier cryptosystems by the public key of the data owner. The GM and Paillier keys (private and public) are stored on the trusted proxy of the data owner.

2) Oblivious Tree Traversal: In order to execute SQL queries, the data analyst sends the query to the proxy. The proxy executes the query by traversing the tree stored on the cloud server. Traversing a tree is performed by retrieving the root's record, evaluating the query against the root's record and updating the query result if the record satisfies the query conditions, and then terminating the search or determining a new root. The records in the database are encrypted by the GM cryptosystem; therefore the query evaluation must be performed on ciphertexts without the need of decryption. For this purpose, the proxy encrypts the constants in the query predicate by the public key of the GM cryptosystem.

The transformed query is then sent to the cloud server by the proxy. The cloud server uses the encrypted constants in the queries and the GM-encrypted columns in the encrypted table as inputs to Fischlin's protocol. The outputs are ciphertext sequences Δ and c, which are required to compare two ciphertexts and evaluate the query predicate. These sequences are stored in a table, namely T. The columns of the table T depend on the type of the query. If an attribute appears in either an interval matching or an exact matching predicate, a column is added to T that contains the ciphertext sequence Δ generated by Fischlin's protocol. In the case of exact matching, another column that contains the ciphertext sequence c is added. In case of *sum* and *avg* queries, the Paillier encryption of the column, targeted by the aggregate function, is added to the database T. In case of *max* and *min* queries, the GM encryption of the column, targeted by the aggregate function, is added to the database T.

Afterwards, the table T is made available to the proxy. The proxy initiates the search by retrieving the content of the tree root at index 1 of the table T using Private Information Retrieval (PIR) [2]; it then extracts Δ and c from the retrieved item. Decryption of these sequences using the decryption keys enables the proxy to evaluate the query conditions. Based on the comparison result, the search is continued on the left or on the right subtree by updating the root index. The search will be performed on the left and/or the right subtree in order to find all the records that satisfy the query conditions. If the left (resp. right) subtree satisfies the condition, it is returned as the query result. If the left (resp. right) subtree intersects with the query range, the search will continue on that subtree. If the current root is at index i in table T, the root index of the left (resp. right) subtree is $2i$ (resp. $2i + 1$).

Updating query result depends on the type of the query:

- *count*: The query result is initialized by zero. If a record that satisfies the query condition(s) is found, the query resulted is incremented by one.
- *sum*: The query result is initialized by the Paillier encryption of 0 using the data owner's public key. If a record that satisfies the query condition(s) is found, its Paillier encrypted column will be multiplied by the query result.
- *max/min*: In the case of *max* (resp. *min*), the query result is encrypted with the GM encryption of a big negative (resp. big positive) number. If a record that satisfies the query condition(s) is found, the query result and the GM encryption of the column targeted in the query are used as inputs to Fischlin's protocol. The query result is then updated based on the output of the ciphertexts comparison.

3) Query Result Decryption: The proxy, at this point, obtains the encrypted query result. The decryption of this ciphertext by the proxy will give the query result.

IV. Private Data Mining

This section first presents how to compute differentially private query answers. We then present a private version of the decision tree induction algorithm [11] using the secure query mechanism.

A. Computing Private Queries

Differentially private query results can be obtained easily by using the Laplace mechanism. Using the secure query execution technique, the proxy first obtains the query result.

Algorithm 1 PDA - *P*rivate *D*ecision tree *A*lgorithm

Input: D - a set of records; $\mathcal{A}$ - a set of predictor attributes; C - class attribute; h - maximum tree depth; ϵ - privacy budget
Output: A decision tree

1: create a node T;
2: **if** $\mathcal{A}$ is empty *or* $h = 0$ **then**
3: return a leaf node with class value $argmax_{c_i \in \Omega(C)}$ $(|D_{c_i}| + Lap(\Delta f/\epsilon'))$, where $\epsilon' = \frac{\epsilon}{(h+1)}$;
4: **else**
5: determine the best attribute $A_i \in \mathcal{A}$ with probability $\propto \exp(\frac{\epsilon'}{2\Delta \mathrm{u}}\mathrm{u}(D, A_i))$, where $\epsilon' = \frac{\epsilon}{(h+1)}$;
6: partition D into $D_{a_1}, \cdots, D_{a_m}$ such that each record in D_{a_i} contains the value a_i, where $a_i \in \Omega(A_i)$;
7: **for** each value a_i **do**
8: attach the node returned by PDA($D_{a_i}, \mathcal{A} - A_i, C, h-1, \epsilon$) to node T;
9: **end for**
10: **end if**
11: **return** T;

The proxy then adds noise to the true count to guarantee differential privacy. The noise is proportional to the sensitivity of the query (i.e., function). The sensitivity Δf of the count function is 1 because count can differ by at most 1 due to the addition or removal of a single record. Therefore, the proxy returns the noisy answer $\hat{f(D)} = f(D) + \mathtt{Lap}(1/\epsilon)$ to the client, which satisfies ϵ-differential privacy.

B. Private Decision Tree Learning

We choose classification analysis as an example of a data mining task as it has diverse applications in many domains. Classification analysis is a two-step process: *learning step* and *classification step*. In the learning step, a classification algorithm builds a classifier by taking a data table as an input, where each row is a record and each column is an attribute. One of the attributes is the *class* attribute with a set of possible values. In the classification step, the classifier predicts the class values of new records.

Notations. We use the following notations. Let $D = \{r_1, \ldots, r_n\}$ be a multiset of records, where each record r_i is composed of d predictor attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ and a *class* attribute C. We represent the data set D in a tabular form and use the terms "database", "data set", or "data table" interchangeably. We assume that each attribute A_i has a finite domain, denoted by $\Omega(A_i)$. Let D_a denote the set of records in D generalized to the value a and $|D_{a,c}|$ denote the number of records in D_a having the class value $c \in \Omega(C)$.

Private Algorithm. Decision tree induction is a learning algorithm for building a classifier, which is known as a decision tree. A decision tree is a tree-like structure containing nodes and edges. Each internal node, starting from the root node, examines an attribute of a record and redirects it to a child node based on the value of that attribute. Finally, the leaf node of the tree contains the expected class value for that record.

Algorithm 1 presents a private decision tree learning algorithm, which is executed by the proxy using the secure count queries. A decision tree is constructed recursively in a top-down fashion. Initially, all the attributes are considered with respect to a quality function (to be discussed below) that measures how well an attribute classifies the records. The best attribute is then chosen as a root node (Line 5) and the records are partitioned into child nodes according to the possible values of the chosen attribute (Line 6).

To ensure differential privacy, the best attribute is chosen using the exponential mechanism (see Section II). Given the scores of all the candidate attributes, the exponential mechanism selects an attribute A_i with the following probability,

$$\frac{\exp(\frac{\epsilon'}{2\Delta \mathrm{u}}\mathrm{u}(D, A_i))}{\sum_{A_i \in \mathcal{A}} \exp(\frac{\epsilon'}{2\Delta \mathrm{u}}\mathrm{u}(D, A_i))}, \tag{3}$$

where $\mathrm{u}(D, A_i)$ is either $\mathtt{InfoGain}(D|A)$ or $\mathtt{Max}(D|A)$ and the sensitivity of the function Δu is $\log_2 |\Omega(C)|$ and 1, respectively (see below). The beauty of the exponential mechanism is that while it ensures privacy, it also exponentially favours a candidate with a high score.

The decision tree learning algorithm is then recursively invoked on each child node without considering the attribute previously partitioned (Line 8). The algorithm terminates either when there is no remaining candidate attribute or the tree has reached the maximal tree depth set by the data analyst (Line 2). Finally, the algorithm returns the leaf node with a class value $c_i \in \Omega(C)$ that has the highest noisy count $|D_{c_i}| + \mathtt{Lap}(\Delta f/\epsilon')$ (Line 3).

Quality Function. We define two quality functions to calculate the score of each candidate $A_i \in \mathcal{A}$. The proxy can compute these functions using the adopted secure count queries. The first utility function is *information gain*:

$$\mathtt{InfoGain}(D|A) = H_C(D) - H_C(D|A) \tag{4}$$

Here, $H_C(D) = -\sum_{c_i \in \Omega(C)} \frac{|D_{c_i}|}{|D|} \times \log_2 \frac{|D_{c_i}|}{|D|}$ is the *entropy* of the data table D with respect to the class attribute C. Entropy denotes the information needed to identify a class value in D. $H(D|A) = \sum_{a_i \in \Omega(A)} \frac{|D_{a_i}|}{|D|} H_C(D_{a_i})$ is the conditional entropy given an attribute A is specialized.

The sensitivity of $\mathtt{InfoGain}(D|A)$ is $\log_2 |\Omega(C)|$, where $|\Omega(C)|$ is the domain size of the class attribute C. It is because the value of the entropy $H_C(D)$ must be between 0 and $\log_2 |\Omega(C)|$. And, the value of the conditional entropy $H_C(D|A)$ lies between 0 and $H_C(D)$. Therefore, the maximum change of $\mathtt{InfoGain}(D|A)$ due to the addition or removal of a record is bounded by $\log_2 |\Omega(C)|$. The second utility function is *Max*:

$$\mathtt{Max}(D|A) = \sum_{a_i \in \Omega(A)} (\max_{c_i \in \Omega(C)} (|D_{a,c_i}|)). \tag{5}$$

$\mathtt{Max}(D|A)$ is the summation of the highest class frequencies over all values of attribute A. The sensitivity of this function is 1.

C. Analysis

We now show that Algorithm 1 provides differential privacy guarantee. The proposed algorithm differs from a non-private version in two different ways. In Line 3, a non-private algorithm would use a deterministic procedure to return a leaf node with the majority class value. Similarly, in Line 5, a non-private algorithm would select an attribute with the highest score value. A differentially-private algorithm cannot use deterministic procedures; therefore, we use the Laplace and the exponential mechanisms in these two steps. We use composition properties of differential privacy to guarantee that the proposed algorithm satisfies ϵ-differential privacy as a whole.

Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition* [12]. However, if the sequence of computations is conducted on *disjoint* data sets, the privacy cost does not accumulate but depends only on the worst guarantee of all computations. This is known as *parallel composition* [12].

Theorem 4.1: Algorithm 1 is ϵ-differentially private.

PROOF. (Sketch) Algorithm 1 first determines an attribute for partitioning using the exponential mechanism (Line 5) and thus, attribute selection step guarantees ϵ'-differential privacy for each iteration. Since the partitions in the same level contain disjoint records, attribute selection step can use the full privacy budget ϵ' for each partition due to the parallel composition property. Note that this step does not take place in the terminating iteration. In the terminating iteration, the algorithm returns the class value of each node (Line 3) using the Laplace mechanism and guarantees ϵ'-differential privacy. Therefore, due to the sequential composition property, Algorithm 1 is ϵ-differentially private, where $\epsilon' = \frac{\epsilon}{(h+1)}$. ■

V. EXPERIMENTAL RESULTS

In this section, we calculate the computation cost for Algorithm 1 and then we present the experimental results.

A. Computational Cost

Let N be the number of records, n the number of the reported nodes that satisfy the query predicates, d the total number of attributes, and s the number of attributes in the query predicates. The computation complexity of the server for exact matching, partial matching and interval matching are $O(N \log^2 N)$, $O\big(N \log N(n + N^{1-s/d})\big)$ and $O\big(N \log N(n + N^{1-1/d})\big)$, respectively [13]. The computation costs of the adopted protocol is presented in Table I. We compare the adopted protocol with the naive approach, in which the proxy performs a linear search on the encrypted table by engaging in Fischlin's protocol to evaluate queries at each table entry. This approach incurs excessive computation overheads, linear to the number of records in the database. The comparison of the protocol with the naive approach, presented in Table I, shows that the adopted protocol achieves better computation complexity on the proxy.

Algorithm	Query Type	Computation on the proxy	Computation on the server
Protocol [13]	Exact matching	$O(\log^2 N)$	$O(N \log^2 N)$
	Partial matching	$O\big(\log N(n + N^{1-s/d})\big)$	$O\big(N \log N(n + N^{1-s/d})\big)$
	Range matching	$O\big(\log N(n + N^{1-1/d})\big)$	$O\big(N \log N(n + N^{1-1/d})\big)$
Naive approach	All types of queries	$O(sN)$	$O(sN)$

Table I
COMPARISON OF THE PROPOSED QUERY EXECUTION PROTOCOL WITH THE NAIVE APPROACH

For each internal node, Algorithm 1 executes a constant number of exact-matching queries for each attribute. Therefore, the number of queries that are executed on the encrypted table is $O(|\mathcal{A}|bh)$ where b indicates the branching factor, h denotes the number of iterations and $|\mathcal{A}|$ is the cardinality of the set of predicator attributes. Therefore, the computation cost of Algorithm 1 is $O(|\mathcal{A}|bh. \log^2 N)$ on the proxy and $O(|\mathcal{A}|bh.N \log^2 N)$ on the server.

B. Experiments

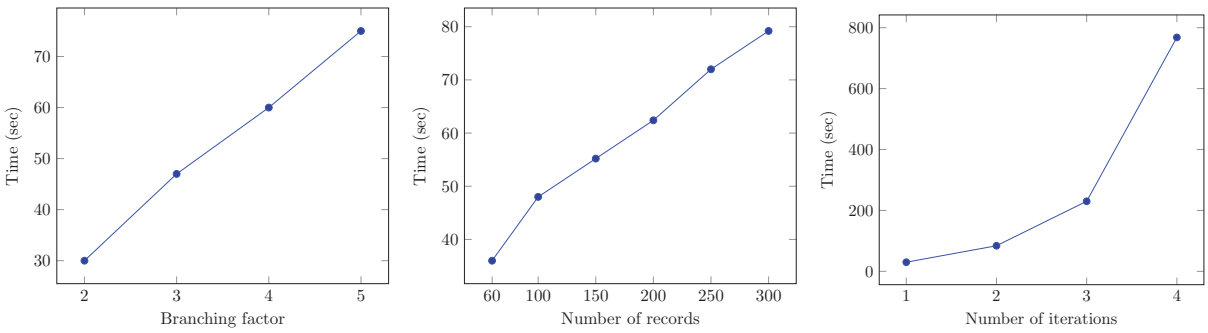
To evaluate the performance of the proposed classification analysis algorithm, we implemented a prototype in Java 1.6 using the BigInteger class provided by the Java standard API. The secret shares of the GM cryptosystem are 256-bit long. All experiments were conducted on an Intel dual Core i5 2.3GHz Notebook with 4GB RAM.

First, we evaluated the performance of Algorithm 1 on synthetic data. Fig. 2 shows how the time required by our algorithm depends on several parameters of the decision-tree learning problem: the branching factor, the number of iterations, and the number of records. Runtime increases linearly with respect to the branching factor and the number of records. However, it increases quite rapidly with an increase in the number of iterations. This is because the number of leaf nodes grows quickly with the increase of the number of iterations. Finally, we observe that the runtime is independent of the number of attributes. This is consistent with our computational cost analysis in Section V-A.

We also evaluated our algorithm with the publicly available Breast Cancer dataset [14]. Breast Cancer dataset has 286 records with 9 attributes, and a branching factor of 4. We chose to build a tree with 2 attributes and 4 iterations. The time consumed by the server for the table generation and the tree traversal are 40s and 0s, respectively, whereas the time consumed by the proxy for the corresponding operations are 0s and 35s, respectively. These experiments demonstrate that the proposed algorithm is practical and can be successfully applied to problem instances of realistic size.

VI. RELATED WORK

Private data analysis techniques can be broadly classified into interactive and non-interactive [3]. In an interactive framework, data analysts pose aggregate queries through a private mechanism and the data owner outputs aggregate answers in response. In a non-interactive framework, a data owner first anonymizes the raw data and then releases the anonymized version for data analysis. The privacy mechanism of this paper is based on the interactive framework. Many algorithms have been proposed to preserve privacy, but only a few have considered the goal for classification



(a) (b) (c)

Figure 2. Performance of the prototype implementation on synthetic data

analysis. Lindell and Pinkas propose a distributed algorithm for computing a decision tree using cryptographic techniques, where the data is partitioned horizontally among the parties [15]. However, the computed decision tree provides no privacy guarantee. Zhu and Du show that publishing decision trees without formal guarantee threatens individual privacy [16]. Recently, a few differentially private classifiers have been proposed [17]. All these approaches, however, do not work on encrypted data, which is the basic requirement of our problem.

VII. Conclusion

The paper has proposed a secure and private data management framework for data mining. The proposed solution executes queries on encrypted data and returns differentially-private answers. Encrypted data provides confidentiality against an untrusted database server while differential privacy thwarts possible inference attacks from a data analyst. Experimental results show that the proposed solution incurs reasonably small communication and computation overhead.

References

[1] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2003.

[2] S. Barouti, D. Alhadidi, and M. Debbabi, "Symmetrically-private database search in cloud computing," in *Cloud Computing Technology and Science (CloudCom), International Conference on*, vol. 1. IEEE, 2013, pp. 671–678.

[3] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys*, vol. 42, no. 4, pp. 1–53, June 2010.

[4] C. Dwork, "Differential privacy," in *Proceedings of the International Conference on Automata, Languages and Programming (ICALP)*, 2006.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the 3rd conference on Theory of Cryptography (TCC)*, 2006.

[6] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[7] M. Fischlin, "A cost-effective pay-per-multiplication comparison method for millionaires," in *Proceedings of the Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, 2001.

[8] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of The ACM Symposium on Theory of Computing (STOC)*, 1982.

[9] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 1999.

[10] J. A. Bærentzen, "On left-balancing binary trees," August 2003, http://www2.imm.dtu.dk/pubdb/p.php?2535.

[11] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann Publishers Inc., 2011.

[12] F. McSherry, "Privacy integrated queries," in *Proceedings of the 35th ACM International Conference on Management of Data (SIGMOD)*, 2009.

[13] M. D. Berg, O. Cheong, M. V. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer-Verlag TELOS, 2008.

[14] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[15] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2002.

[16] Z. Zhu and W. Du, "Understanding privacy risk of publishing decision trees," in *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec)*, 2010.

[17] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," *Trans. Data Privacy*, vol. 5, no. 1, pp. 273–295, Apr. 2012.