Inference Attack against Encrypted Range Queries on Outsourced Databases

Mohammad Saiful Islam
Computer Science Department
University of Texas at Dallas
saiful@utdallas.edu

Mehmet Kuzu
Computer Science Department
University of Texas at Dallas
mehmet.kuzu@utdallas.edu

Murat Kantarcioglu
Computer Science Department
University of Texas at Dallas
muratk@utdallas.edu

ABSTRACT

To mitigate security concerns of outsourced databases, quite a few protocols have been proposed that outsource data in encrypted format and allow encrypted query execution on the server side. Among the more practical protocols, the "bucketization" approach facilitates query execution at the cost of reduced efficiency by allowing some false positives in the query results. *Precise Query Protocols* (PQPs), on the other hand, enable the server to execute queries without incurring any false positives. Even though these protocols do not reveal the underlying data, they reveal query access pattern to an adversary. In this paper, we introduce a general attack on PQPs based on access pattern disclosure in the context of secure range queries. Our empirical analysis on several real world datasets shows that the proposed attack is able to disclose significant amount of sensitive data with high accuracy provided that the attacker has reasonable amount of background knowledge. We further demonstrate that a slight variation of such an attack can also be used on imprecise protocols (e.g., *bucketization*) to disclose significant amount of sensitive information.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*; H.2.0 [**Database Management**]: General—*Security, integrity, and protection*

Keywords

Encrypted Range Query; Inference Attack; Database-as-a-service

1. INTRODUCTION

Database-as-a-service (DAS) model has been considered as an emerging technology over the last few years. New web services such as *Google Bigquery* [1] are introduced to provide SQL-like capabilities to analyze massive datasets on the cloud. Outsourcing databases to the remote servers paves the way of sharing expertise of database professionals among different organizations at a lower cost. But, executing queries effectively on an encrypted database has been a very challenging issue in the database community. Hacigümüş et al. [15] were the first to propose a notion of encrypted database which supports a limited execution of SQL queries in the remote server without requiring to decrypt the data elements. Since then, a lot of research (e.g., [4–6, 12, 16, 23]) has been conducted in this context to enhance the basic solution proposed in [10, 15]. Most of these protocols, however, focuses only on range queries (e.g., [4–7, 16, 23]).

The basic DAS paradigm proposed in the literature consists of three entities: the users, the clients and the server. Each row in the outsourced database is encrypted individually and called as *etuple*. The remote server in DAS model is assumed to be 'honest but curious'. That is, the server does not deviate from the protocol, however she wants to learn as much knowledge about the encrypted data as possible. A *user* in this model submits queries to the *client*. The *server* is the remote machine that hosts the database physically in encrypted form and handles queries received from various *clients*. Client transforms a user-query to an appropriate encrypted server-query and sends it to the server. The server executes this query on the encrypted database and sends back the appropriate etuples to the client. Upon receiving the set of etuples, the client decrypts them and returns it to the user.

There are two mainstream approaches to implement the DAS model that supports range queries. The *Precise Query Protocols* (PQPs) (e.g., [4–7, 23, 29]) use sophisticated encryption techniques that enables the server to execute range queries without incurring any false positives. That is, for any given encrypted range query, the server returns the exact set of etuples that satisfies the given query. The second approach, known as the *bucketization* approach (e.g., [15, 16]), facilitates range query execution at the cost of reduced efficiency in terms of the number of false positives. In fact, in bucketization approach, each query attribute is divided into some contiguous buckets based on a bucketization strategy. During the execution of queries, all the etuples in a bucket is included in the resultset if this bucket overlaps with the range specified in the query. Needless to mention, this may introduce a reasonable amount of false positives in the resultset depending on the bucket width. The client, in this case, does some post processing to get rid of the false positives before presenting the result to the user.

CODASPY'14, March 3–5, 2014, San Antonio, Texas, USA.
Copyright 2014 ACM 978-1-4503-2278-2/14/03 ...$15.00.
http://dx.doi.org/10.1145/2557547.2557561.

Although cryptography researchers are taking significant strides in designing efficient Oblivious RAM (e.g., [25, 27]) solutions that allow all the basic operations to be applied directly on outsourced data without revealing any information to the server, the performance overhead of such systems is still significant. In fact, it has been shown in [18] that it is extremely difficult for a remote database server to provide cryptography-style security, and yet be efficient enough to be practical. Therefore, all the practical DAS protocols intentionally allow some information to be leaked to the remote server in exchange of greater efficiency. In essence, all the precise protocols disclose Query Access Pattern (QAP) to an adversary. Informally, we say that a DAS model leaks QAP if *for a given query Q, the adversary can see the list of etuples $\langle e_1, \cdots, e_k \rangle$, but does not learn the content of these encrypted tuples.*

To illustrate how QAP leakage can be exploited to launch an attack, consider two encrypted range queries of the form $[s_1, f_1]$ and $[s_2, f_2]$. Furthermore, let E_1 and E_2 be the respective set of etuples returned as a response to these queries. An attacker can extract valuable information about $[s_1, f_1]$ and $[s_2, f_2]$ by analyzing E_1 and E_2. For example, if $E_1 \subset E_2$, then an adversary can tentatively assume the range $[s_1, f_1]$ is completely contained within the range $[s_2, f_2]$. That is, the condition $(s_1 \geq s_2) \wedge (f_1 \leq f_2)$ holds. In fact, we show that if an attacker has a reasonable estimate of the data distribution of the query attribute (e.g., from publicly available databases), these inferences can lead to very accurate estimates of the range queries, and thereby violates the security of the underlying DAS model.

Since imprecise protocols (i.e., the bucketization approach) do not reveal true access pattern due to the presence of false positives, they are much more resistant to the access pattern based inference attacks. However, bucketization approach reveals the bucket identifiers to an attacker. In fact, we show how a slight variation of our attack can be used to successfully reveal the bucket boundaries in §4.

In recent years, complete database solutions (e.g., Monomi [26], CryptDB [22]) have been proposed in the literature that can run almost any queries on encrypted data. These solutions use the existing practical DAS models outlined earlier and incur a very reasonable amount of added overhead. In fact, the reported median overhead of Monomi is only 1.24 for the TPC-H benchmark [26]. Therefore, it is very likely that there would be commercially available encrypted databases for general use in the near future. The main objective of this paper is to raise awareness to the users before using a database system that leaks QAP. We argue that the DAS models and their vulnerabilities (e.g., QAP leakage) should be thoroughly studied before they can be used in a secure encrypted database system. However, we do not claim that our work presents a worst case exploitation of QAP disclosure. In fact, there is a possibility that future studies may suggest that more information can be inferred exploiting QAP using lesser amount of prior knowledge. In that regard, we consider this work as a stepping stone towards a series of future works in this area of research. The main contributions of this paper are summarized as follows.

1. We present an attack model which exploits access pattern disclosure to reveal range boundaries of encrypted range queries on DAS model.
2. We also propose a variation of such an attack to disclose the bucket boundaries in the bucketization approach.
3. Finally, we empirically demonstrate the efficacy of our model using publicly available real datasets.

The rest of the paper is organized as follows. We present the motivation behind this work in §2. We outline our basic attack model in §3. In §4, we show how a slight variation of our basic model can still be applied to attack imprecise protocols such as "bucketization". We propose a new prediction metric in §5. A thorough empirical evaluation of our attack model is presented in §6. We present a brief survey of the contemporary literature related to this work in §7. Finally, we conclude this paper with §8.

2. MOTIVATION

Access pattern disclosure alone does not pose any significant threat to the overall data security of a DAS model. However, when combined with reasonable amount of background knowledge, access pattern disclosure may cause substantial privacy breach. We like to highlight this claim further by considering the following hypothetical scenario.

Let us assume that a bank is outsourcing its database to a remote server for reduced database management cost. Now, let us consider the account holder table in the outsourced database. Since the names of the account holders may not be considered as sensitive information, we may assume that the remote server (e.g., *Mallory*) has obtained the list of account holder's names through some covert channel. For example, *Mallory* can monitor the recipients of postal mails sent from the bank to find out the customers of this bank. Quite naturally, the bank wants to keep more sensitive account information (e.g., account balance etc.) safe. Therefore, all the tuples in the account table is kept encrypted in the remote database. However, since *Mallory* has the plaintext distribution of names, she can monitor queries and responses to estimate a mapping from the individual names to their corresponding etuples by applying **a similar attack such as ours** with reasonable accuracy. Since the tuples are encrypted, *Mallory* still does not learn any useful information form the etuples yet.

However, suppose the bank is investigating a case of money laundering and keeps a list of suspects incorporated in the account table. Naturally, the information whether an account holder belongs to this list is considered to be highly sensitive and disclosure of such information may very well lead to lawsuits. Now, further assume that an investigating authority is making a query to the remote database to pull the list of suspects. However, if *Mallory* observes the set of etuples accessed by the investigating authority, she can ascertain this sensitive list with reasonable accuracy by using her estimate of the *name-to-etuple* mapping.

Quite clearly, neither the list of the account holder's names nor the encrypted tuples are detrimental to the overall data security. However, it is the association between these two sets of information that can lead to a significant privacy breach. For example, the information that 'John has an account in this bank' does not seem sensitive. Again, an adversary (e.g., the remote database server) may see all the etuples in the encrypted table and still learns no useful information. However, as soon as the adversary has a way to link the name 'John' with a particular etuple, she can use

Table 1: Notations

Notation	Meaning
$\|x\|$	The cardinality of the set x.
R	The relation on which queries are executed.
A	An attribute on *Relation* R.
$\mathcal{D}$	A positive, discrete, ordered domain of A.
$\mathcal{D}_{low}$	The lowest value of the domain $\mathcal{D}$.
$\mathcal{D}_{high}$	The highest value of the domain $\mathcal{D}$.
n	The number of distinct elements in $\mathcal{D}$. That is, $n = \|\mathcal{D}\|$.
m	The number of queries in the query set.
$[s, f]$	A query with left end s and right end f. That is, $[s, f] = \{x \| (x \geq s) \wedge (x \leq f)\}$.
β, γ	*Real* valued constants.
B_{size}	Individual batch size.

this to gain sensitive information. For example, she can easily find out whether the person named 'John' is a suspect of the money laundering scheme or not. Therefore, attack such as ours may cause significant privacy breach by establishing a mapping from attribute values to their corresponding etuples.

In this paper, we show that if a DAS model reveals access pattern, an adversary can use plaintext data distribution to ascertain the encrypted queries. This ultimately reveals the association of the range query attribute with their corresponding etuples. Quite naturally, using such DAS models naively may cause significant security breach. Therefore, we strongly argue that these protocols should only be used in controlled settings where an adversary is guaranteed not to possess any background knowledge on the plaintext data distribution.

3. THE PROPOSED MODEL

In this section, we describe an attack model that exploits query access pattern to identify sensitive data. For the sake of simplicity, we focus our attention to one dimensional range queries. *Range Queries* are perhaps one of the most important queries and most of the existing techniques supports range queries on one dimension anyway. Since any *equality* query can be equivalently replaced by a range query, all the discussions in this paper is implicitly applicable to *equality queries* as well. Let us assume R is a relation with a sensitive attribute A. We assume A has a finite ordered discrete domain (e.g., the set of all the non-negative integers less than t, where $t \in \mathcal{N}$). We denote the domain of A by $\mathcal{D}$ such that $|\mathcal{D}| = n$. That is, the domain $\mathcal{D}$ has n discrete elements[1]. A comprehensive list of notations used in this paper has been presented by *Tab.* 1. We define most of these notations as needed, but we encourage the readers to use *Tab.* 1 as a reference throughout the paper.

3.1 Threat Model

In our attack model, we assume an attacker (e.g., *Mallory*) has access to a sequence of m range queries of the form $\mathcal{Q} = \langle[s_1, f_1], [s_2, f_2], \cdots [s_m, f_m]\rangle$ and their corresponding result sets $\langle E_1, E_2, \cdots E_m\rangle$. Ideally, we assume an 'honest but curious' server as our adversary. Furthermore, we assume that *Mallory* has access to the following background knowledge on data distribution to facilitate an attack.

1. The attacker knows the plaintext distribution (exact or approximate) of the column A in the original database. That is, for any domain value x, the attacker knows how many tuples in the table contains x as the attribute value of A. Let, $S_{i,j}$ denotes the set of elements in A that falls into the range $[i, j]$. Therefore, we can assume that the adversary knows how to calculate $S_{i,j}$ values for any value of i and j.

2. Finally, the attacker knows the true ranges of k queries in $\mathcal{Q}$, where k can be 0. Let $\langle\langle a_1, b_1\rangle \cdots \langle a_m, b_m\rangle\rangle$ be the sequence of true values for the observed sequence of range queries. That is, $\forall i, (s_i = a_i) \wedge (f_i = b_i)$ holds. Then, for some sequence of k values s.t. $k < m$, $\langle c_1 \cdots c_k\rangle$, the adversary knows the right assignments for each of the c_i queries. That is, for all $i \in [1, k]$, the adversary knows the sequence $\langle\langle a_{c_1}, b_{c_1}\rangle \cdots \langle a_{c_k}, b_{c_k}\rangle\rangle$.

The first assumption has been adopted in some notable previous works in this area (e.g., [8,10]). Also, many different data distributions can be obtained from publicly available databases. For example, there is a publicly available database, namely the *North Carolina Voter Database* [21] that presents many key personnel information about the voters in the state of *North Carolina*. Again, in a bank account database, the list of account holder's names can be fully known beforehand [10]. The *ZIP* codes of the account holders, on the other hand, can be approximated from population data [10]. In fact, we empirically show that an attacker can obtain a very reasonable amount of sensitive information by attacking range queries on *NC Voter Dataset* using the age statistics of NC census data [2] in §6.2.4. Therefore, we argue that the first assumption is reasonable in this context. Finally, we see it appropriate to add the second assumption for the following reasons. First of all, we believe that revelation of a few queries should not compromise the secrecy of other queries in a secure system (e.g., [14]). Secondly, this assumption may be true for scenarios where an inside attacker has restricted permission to execute a very few queries, and therefore can use his limited privilege to compromise many other queries. Finally, the required number of known queries is extremely small. In fact, our empirical analysis indicates that a very few known queries, e.g., 2, is sufficient for a large query set of over hundred queries.

3.2 The Attack Model

We formalize the attack model to a *Constrained Optimization* problem. The adversary has a sequence of range queries $\mathcal{Q}$ of the form $\langle[s_1, f_1], [s_2, f_2], \cdots [s_m, f_m]\rangle$ and their corresponding result set $\langle E_1, E_2, \cdots E_m\rangle$. The goal of the adversary is to estimate the values of $s_1, \cdots, s_m, f_1, \cdots, f_m$ from the domain $\mathcal{D}$ as accurately as possible with the help of her prior knowledge. We model this problem as a *constrained optimization* problem in the following way.

$$\underset{\langle[s_1,f_1],\cdots[s_m,f_m]\rangle}{\text{argmin}} \sum_{[s_i,f_i],[s_j,f_j]} \left(\left|S_{s_i,f_i} \cap S_{s_j,f_j}\right| - |E_i \cap E_j|\right)^2 \tag{1}$$

$$\textbf{Subject To}: T$$

[1]This assumption does not limit the applicability of our approach. An adversary can still apply our attack model to a continuous domain by discretizing the domain.

Here, T is the set of constraints. It should be noted that this optimization problem trivially incorporates individual cardinalities of the etuple sets when $i = j$. We describe the constraint generation process in great detail in the following section.

3.3 Generating Constraints.

The first set of trivial constraints that the optimization problem has to respect is that the known queries has to be assigned to their known values. Therefore, we have k such trivial constraints. But, the set of *etuples* reveal many other constraints in the following way.

Let us assume that $[s_1, f_1]$ and $[s_2, f_2]$ be any two range queries. Also, let E_1 and E_2 be the set of corresponding etuples for these queries. Regardless of the underlying data distribution, one of the following four scenarios should hold for these two queries.

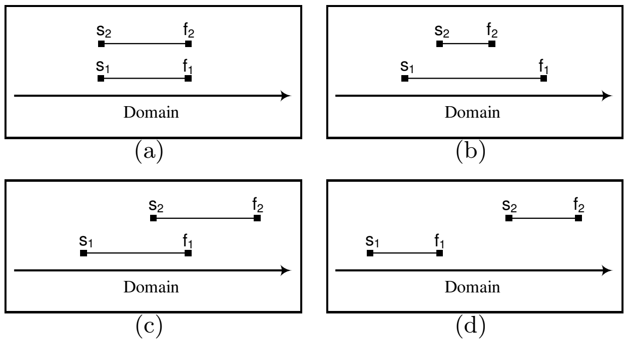


Figure 1: Possible relationships between two *range queries*. (a) Two queries are equal. (b) One of the queries is a proper subset of the other. (c) Queries are overlapping. (d) Queries are mutually exclusive.

1. These two queries represent the same range.
2. One of the ranges is completely subsumed by the other.
3. The ranges are overlapping.
4. The ranges are non-overlapping, i.e., they are disjoint.

These four different scenarios are pictorially depicted in Fig. 1. It should be noted that in Fig. 1(b), it has been arbitrarily assumed that the first range contains the second one. But, the other possibility can also occur where the second range contains the first one. Similarly, alternate considerations also exist for cases (c) & (d).

Interestingly enough, an attacker can tentatively ascertain which one of the above mentioned four scenarios holds for a given pair of queries by using their corresponding set of etuples E_1 and E_2. For example, *if* $E_1 = E_2$, *then* the given queries are assumed to represent the same range and hence falls into the first scenario. This fact can be further expressed into a succinct logical expression of the form: $(s_1 = s_2) \wedge (f_1 = f_2)$. An attacker can use this expression as a constraint to the optimization problem presented in (1) to limit the search space of feasible solutions. Similarly, if $E_2 \subset E_1$, then case (b) holds and the expression $(s_2 \geq s_1) \wedge (f_1 \geq f_2)$ can be used as a constraint.

Therefore, if *Mallory* observes a sequence of m queries and their respective set of *etuples* $\langle E_1, E_2, \cdots E_m \rangle$, she can take any two set of *etuples* E_i and E_j, and generate a new constraint involving the corresponding start and end variables, namely, s_i, f_i, s_j and f_j. Thus, an adversary can generate $\binom{m}{2} = \frac{1}{2}m(m-1)$ different non-trivial constraints from m different range queries. Additionally, any given query $[s_i, f_i]$ trivially satisfies the constraint $s_i \leq f_i$. So, given the query response of m queries, an adversary can generate $m + \frac{1}{2}m(m-1) + k$ constraints to the optimization problem presented in (1).

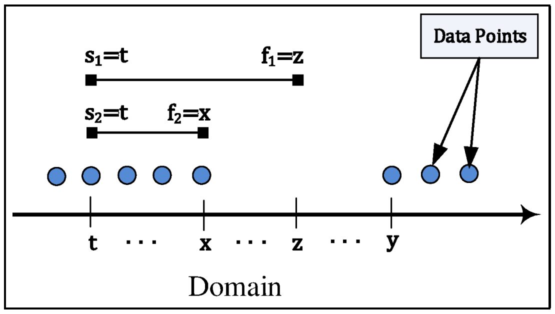


Figure 2: Imperfect constraints due to absence of data points.

However, caution must be exercised in generating constraints when some of the domain values of the attribute A are devoid of any data points. To further illustrate this point, let us consider the scenario depicted in Fig. 2. Here, no tuple in the relation R contains an element from the range (x, y) as a value of the attribute A. Furthermore, let z be a point such that $z \in (x, y)$ holds. It is quite apparent from the figure that the etuple sets for the queries $[s_1, f_1]$ and $[s_2, f_2]$ contain the same set of etuples. Therefore, an attacker may erroneously conclude that these two queries fall to the first category and thus generate the constraint $(s_1 = s_2) \wedge (f_1 = f_2)$. But, even a cursory examination of the figure tells us that these queries belong to the second category and thus the afore-mentioned constraint is wrong. The main reason behind this error is that there is no data point between x and z. Even the existence of a single data point between x and z would make E_2 a proper subset of E_1. In that case, the attacker could successfully identify that these queries fall into the second scenario and generate the correct constraint: $(s_2 \geq s_1) \wedge (f_1 \geq f_2)$.

Definition 1. For a given domain $\mathcal{D}$ of an attribute A and an instance distribution D, τ is defined to be the maximum number of consecutive domain values for which there is no data point in D.

Example 1. (τ) Let us consider a domain $\mathcal{D} = [1, 10]$ and a given distribution $\{1, 1, 3, 7, 8, 10\}$. Here, the maximum number of consecutive domain values where there is no data points is 3 (no data points in $4, 5$ and 6). Therefore, the value of τ for this distribution is 3.

It should be noted that an attacker can efficiently calculate τ from a given data distribution with an $O(n)$ algorithm. An attacker can use τ to model the amount of error in the constraints. In essence, an attacker just relaxes the constraints by τ to compensate the errors generated by the lack of data points in some domain values. *Lemma* $1 - 4$ outline how much relaxation is necessary under different scenarios.

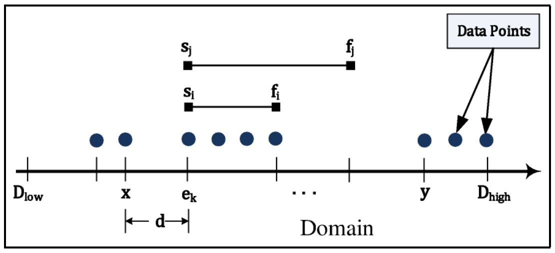


Figure 3: Generating constraints when $E_i = E_j$.

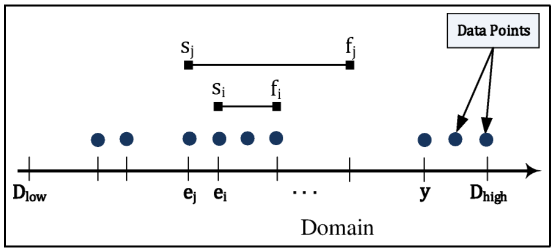


Figure 4: Illustrating case $e_j < e_i$ when generating constraints for $E_i \subset E_j$.

Throughout the proofs of Lemma $1 - 4$, we assume an ordered discrete attribute A on which the range queries are performed. We further assume $\mathcal{D} = [\mathcal{D}_{low}, \mathcal{D}_{high}]$ be the domain of A and D be an instance data distribution. Furthermore, throughout the proofs, we adopt some notational abuses for the sake of simplicity. That is, we informally use encrypted tuples as arguments of comparison operators (e.g., $=, <, >, \geq, \leq$) and arithmetic operators (e.g., $+, -$) to mean that the actual operation is to be performed on the attribute value of A of the argument etuples. For example, we simply write $e_1 = e_2$ to mean that both these etuples has the same attribute value of A. It should be noted that this slight abuse simplifies the notations in the proofs significantly.

Lemma 1. *If $[s_i, f_i]$ and $[s_j, f_j]$ be a given pair of queries and E_i and E_j be their corresponding set of* etuples *s.t. $E_i = E_j$, then the following constraint holds.*

$$(|s_i - s_j| \leq \tau) \wedge (|f_i - f_j| \leq \tau)$$

Proof. Let us assume, without loss of generality, $e_k \in E_i, E_j$ be the lowest data point in E_i and E_j. If D is the data distribution of attribute A, let us assume $x \in D$ be the maximum data point s.t. $x < e_k$ if such x exists, $x = \mathcal{D}_{low}$ otherwise. This scenario is depicted in Fig. 3. Since e_k is the lowest data point in E_i and E_j, x can not be a member of either E_i and E_j, i.e., $x \notin E_i, E_j$. Since both E_i, E_j contains e_k, but not x, it is obvious that the condition $x \leq s_i, s_j \leq e_k$ holds.

Therefore, if $d = e_k - x$ is the distance between x and e_k, then the inequality $|s_i - s_j| \leq d$ holds. But, by the definition of τ, we have $d \leq \tau$. Therefore, the inequality $|s_i - s_j| \leq d \leq \tau$ holds. That is, $|s_i - s_j| \leq \tau$ holds. By a similar way, we can show that the inequality $|f_i - f_j| \leq \tau$ holds. This concludes the proof. $\square$

Lemma 2. *If $[s_i, f_i]$ and $[s_j, f_j]$ be a given pair of queries and E_i and E_j be their corresponding set of* etuples *s.t. E_i is a proper subset of E_j, i.e., $(E_i \subset E_j)$, then the following constraint holds.*

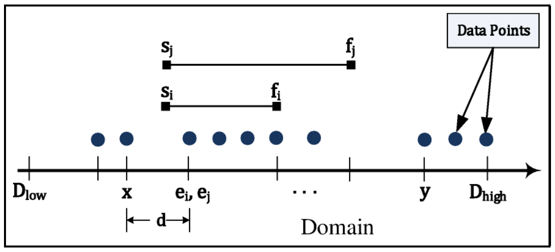


Figure 5: Case $e_j = e_i$ when generating constraints for $E_i \subset E_j$.

$$(s_j \leq (s_i + \tau)) \wedge (f_i \leq (f_j + \tau)).$$

Proof. Expressions $(s_j \leq (s_i + \tau))$ and $(f_i \leq (f_k + \tau))$ are connected by conjunction in this constraint. Therefore, it is sufficient to show that both these expressions hold independently of each other. At first, let us prove that the expression $(s_j \leq (s_i + \tau))$ holds.

Let $e_i \in E_i$ and $e_j \in E_j$ be the lowest elements in E_i and E_j respectively. Since $E_i \subset E_j$, e_i can not be strictly less than e_j. Therefore, one of the following two conditions must hold.

(a) $e_j < e_i$. This scenario is depicted in Fig. 4. In this case, since $e_i \in E_i$ and $e_j \notin E_i$, the inequality $e_j < s_i \leq e_i$ holds. Again, since $e_j \in E_j$, we have $s_j \leq e_j$. Combining these two inequalities, we can say that the inequality $s_j \leq e_j < s_i$ holds.

Again, since $\tau \geq 0$ and $s_j < s_i$, s_j is less than $s_i + \tau$. Therefore, the inequality $s_j \leq s_i + \tau$ holds.

(b) $e_j = e_i$. This scenario is depicted in Fig. 5. Since $e_i = e_j$, we can interchangeably use e_i and e_j to denote the lowest point of E_i and E_j without loss of generality. Let us assume $x < e_i, e_j$ be the highest data point in D such that $x \notin E_i, E_j$. According to the definition of x, there exists no data point y such that $x < y < e_i$. If such an y exists, then y becomes the maximum data point such that $y < e_i$ and $y \notin E_i, E_j$, contradicting the definition of x. Let d be the distance between x and e_i. That is, $d = e_i - x$. Since x and e_i are two consecutive data points in D, the inequality $d \leq \tau$ must hold according to the definition of τ.

On the other hand, since $x < e_i$ and x does not belong to E_i, E_j, the inequality $x < s_i, s_j \leq e_i, e_j$ holds. Since both s_i and s_j are located between the points x and e_i, the difference between s_i and s_j can not possibly be greater than the distance between x and e_j. Therefore, the constraint $|(s_j - s_i)| \leq d$ holds. That is, s_i can not be less than s_j by more than d. Therefore, the inequality $s_j \leq (s_i + d)$ holds.

Since $d \leq \tau$ and $s_j \leq (s_i + d)$ holds, we can conclude that the inequality $s_j \leq (s_i + \tau)$ holds.

In the same way, we can show that $f_i \leq (f_j + \tau)$. This concludes the proof. $\square$

Lemma 3. *If $[s_i, f_i]$ and $[s_j, f_j]$ be a given pair of queries and E_i and E_j be their corresponding set of* etuples *s.t. $(((E_i \cap E_j) \neq \phi) \wedge ((E_i - E_j) \neq \phi) \wedge ((E_j - E_i) \neq \phi))$, then the following constraint holds.*

$$(((s_i < s_j) \wedge (f_i < f_j)) \vee ((s_j < s_i) \wedge (f_j < f_i)))$$

Proof. Let us assume $[s_i, f_i]$ and $[s_j, f_j]$ be a given pair of queries and E_i and E_j be their corresponding set of *etuples*

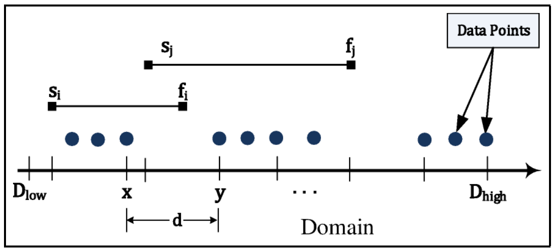


Figure 6: Generating constraints when $E_i \cap E_j = \phi$

s.t. $(((E_i \cap E_j) \neq \phi) \wedge ((E_i - E_j) \neq \phi) \wedge ((E_j - E_i) \neq \phi))$. It should be noted that for this given pair of queries, the equality $s_i = s_j$ can not hold. Because, if $s_i = s_j$ holds, then either E_i is a subset of E_j if $f_i \leq f_j$, or E_j is a subset of E_i if $f_j \leq f_i$. Therefore, either one of the following two conditions must hold.

(a) $s_i < s_j$. According to a similar argument presented in the preceding paragraph, the equality $f_i = f_j$ can not hold under the premises of this lemma. Again, since $s_i < s_j$, f_i can not possibly be greater than f_j. This is because, $((s_i < s_j) \wedge (f_i > f_j)) \Rightarrow (E_j \subset E_i)$ holds. But, this contradicts the premise $(E_j - E_i) \neq \phi$. Therefore, $(f_i < f_j)$ holds. Therefore, in this case, $(s_i < s_j) \wedge (f_i < f_j)$ holds.

(b) $s_j < s_i$. If we interchange the indices i and j and follow a similar argument as presented in the previous paragraph, we can conclude that the constraint $(s_j < s_i) \wedge (f_j < f_i)$ holds.

Since one of the two conditions depicted above always holds, we can conclude that the constraint holds. This concludes the proof. □

LEMMA 4. *If $[s_i, f_i]$ and $[s_j, f_j]$ be a given pair of queries and E_i and E_j be their corresponding set of* etuples *s.t. $(E_i \cap E_j) = \phi$, then the following constraint holds.*

$$(f_i \leq (s_j + \tau)) \vee (f_j \leq (s_i + \tau)).$$

PROOF. Let us assume $[s_i, f_i]$ and $[s_j, f_j]$ be a given of queries and E_i and E_j be their corresponding set of etuples s.t. $(E_i \cap E_j) = \phi$. Now, at least one of the following conditions must hold.

(a) $s_i \leq s_j$. This scenario is depicted in Fig. 6. It should be noted that although the condition $(E_i \cap E_j) = \phi$ holds in Fig. 6, interestingly enough, the ranges $[s_i, f_i]$ and $[s_j, f_j]$ are overlapping with each other. Now, let $y \in E_j$ be the lowest data point. Quite naturally, the condition $s_j \leq y \leq f_j$ holds. Again, let $x \in D$ be the maximum data point such that $y < s_j$. Since $y < s_j$ holds, y can not be a member of E_j.

On the other hand, according to the definition of x and y, there can be no more data point $z \in D$ s.t. $x \leq z \leq y$ holds. That means, x and y are two consecutive data points in D. Therefore, if d be the distance between x and y, the inequality $d \leq \tau$ would always hold according to the definition of τ. Now, there can be one of the two possibilities.

1. $x \leq f_i$. Since $E_i \cap E_j = \phi$, the inequality $y \notin E_i$ holds. Therefore, both s_j and f_i fall belongs to the range $[x, y]$. Therefore, the value of $f_i - s_j$ can not be larger than the distance between x and y. That is, $(f_i - s_j) \leq d$ holds. Since $d \leq \tau$ holds, the inequality $f_i - s_j \leq \tau$ also holds. Therefore, we can conclude that $f_i \leq s_j + \tau$ holds.
2. $x > f_i$. In this case, since $x < s_j$ holds, the inequality $f_i < s_j$ trivially holds. Therefore, the inequality $f_i \leq s_j + \tau$ also holds.

Therefore, we can conclude that if $s_i \leq s_j$ holds, the inequality $f_i \leq s_j + \tau$ would also hold.

(b) $s_j \leq s_i$. By interchanging the indices i and j and following the similar arguments as presented above, we can conclude that if $s_j \leq s_j$ holds, the inequality $f_j \leq s_i + \tau$ would also hold.

Since at least one of the conditions always holds, the constraint $(f_i \leq (s_j + \tau)) \vee (f_j \leq (s_i + \tau))$ holds. □

Using *Lemma* $1-4$, the set of constraints T can be generated using Alg. 1.

Algorithm 1 Building Constraint Set: T

```
$T \leftarrow \emptyset$
for all $i \in [1, m]$ do
  $T \leftarrow T \cup \{s_i \leq f_i\}$
end for
for all $i \in [1, k]$ do
  $T \leftarrow T \cup \{(s_{c_i} = a_{c_i}), (f_{c_i} = b_{c_i})\}$
end for
for all $\langle [s_i, f_i], [s_j, f_j] \rangle$ do
  if $(E_i = E_j)$ then
    $T \leftarrow T \cup \{(|s_1 - s_2| \leq \tau) \wedge (|f_1 - f_2| \leq \tau)\}$
  end if
  if $(E_i \subset E_j)$ then
    $T \leftarrow T \cup \{(s_j \leq (s_i + \tau)) \wedge (f_i \leq (f_j + \tau))\}$
  end if
  if $(E_j \subset E_i)$ then
    $T \leftarrow T \cup \{(s_i \leq (s_j + \tau)) \wedge (f_j \leq (f_i + \tau))\}$
  end if
  if $((E_i \cap E_j) = \emptyset)$ then
    $T \leftarrow T \cup \{(f_i \leq (s_j + \tau)) \vee (f_j \leq (s_i + \tau))\}$
  end if
  if $(((E_i \cap E_j) \neq \emptyset) \wedge ((E_i - E_j) \neq \emptyset) \wedge ((E_j - E_i) \neq \emptyset))$
  then
    $T \leftarrow T \cup$
      $\{((s_i < s_j) \wedge (f_i < f_j)) \vee ((s_j < s_i) \wedge (f_j < f_i))\}$
  end if
end for
return $T$
```

4. ATTACK AGAINST BUCKETIZATION

In this section, we carry out an inference attack against the "bucketization" approach. Intuitively speaking, the bucketization approach is more resistant to access pattern based attacks. Partitioning data values into buckets can be viewed as a way of adding noise to the query access patterns. Therefore, attack techniques like the one described in §3.2 are ill-suited to attack the bucketization approach.

Interestingly, if we slightly modify our attack goal, we may still be able to discover some useful information from the bucketization approach. First of all, we notice that from an adversary's point of view, each data point inside a bucket is indistinguishable from the other points in the same bucket. Secondly, the *bucket−ids* in the queries are sent in plaintext to the server in the bucketization method. Therefore, an adversary can easily identify the set of etuples that belongs to a set of buckets that appear in a query. Therefore, instead of predicting range query values, we look to identify bucket boundaries in the bucketization approach. In essence, an attacker generates assignments of domain values to bucket boundaries such that the number of data points returned in response to a list of bucket identifiers matches with the number of elements in the buckets according to the assignment for each query in the observed query set.

In bucketization, each range query is converted into a list of buckets and the bucket identifiers are sent to the server in plaintext. Therefore, the attacker sees a set of bucket identifiers and the set of etuples returned for this bucket set. Let $\beta_1, \beta_2, \cdots, \beta_p$ be the set of buckets. Let l_{β_i} and h_{β_i} be the low and high boundaries of bucket β_i assigned by the attacker where $\forall_i$, l_{β_i},$h_{\beta_i} \in [\mathcal{D}_{low}, \mathcal{D}_{high}]$. Let $\mathcal{Q} = \{q_i\}$ be the set of observed queries and E_i be the corresponding set of etuples for each query q_i. Let $B_i = \{\beta_t\}$ be the set of bucket identifiers for each query q_i. Since, B_i is sent to the server in plaintext, an adversary (e.g., the untrusted server) can see B_i. In our model, an adversary tries to assign values to l_{β_i} and h_{β_i} for all i s.t. the observed set of response matches the actual data distribution D. More specifically, the adversary tries to optimize the following expression.

$$\underset{\langle [l_{\beta_1}, h_{\beta_1}], \cdots [l_{\beta_p}, h_{\beta_p}] \rangle}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{Q}|} \left(\left| \bigcup_{\beta \in B_i} [l_\beta, h_\beta] \right| - |E_i| \right)^2 \tag{2}$$

It should be noted that an attacker does not need to know the values of any query beforehand for this attack. Again, the solution to the problem above is an approximation of the bucket boundaries. Therefore, it does not guarantee the revelation of the range queries exactly. However, if the buckets are small enough, an attacker can still get a very good approximation of the actual query values.

5. QUERY EVALUATION METRIC

In this section, we propose a new metric to evaluate range query prediction. Of course, we can use an exact right/wrong approach as our evaluation metric. Under this approach, for a given range query $[s, f]$, we evaluate a prediction $[s', f']$ as right if and only if the predicted query is exactly similar to the original query (i.e., $s = s' \wedge f = f'$). Unfortunately, such a simple approach does not do justice to the predictions which are almost accurate but not exactly right. Therefore, we need another metric to evaluate query predictions more accurately.

As a first step towards designing a rigorous evaluation metric, we intuitively identify the following factors which affect the quality of a given prediction.

1. *Absolute Error.* We define *absolute error* to be the difference between a given query and its prediction. That is, given a query $[s, f]$ and its predicted value $[s', f']$, we define *absolute error* $= |s - s'| + |f - f'|$. Quite naturally, a lower value of the absolute error should indicate a better prediction under a reasonable evaluation metric.

2. *Domain Interval.* The second key factor is the *domain interval*, i.e., $(\mathcal{D}_{high} - \mathcal{D}_{low})$. A given query prediction seems more accurate under a larger domain than a smaller domain if *absolute error* is kept constant.

3. *Query Interval.* We define *query interval* to be the value $(f - s)$. Intuitively, higher values of *query interval* indicates more accurate prediction when all the other factors are kept constant. For example, let $[20, 20]$ and $[30, 80]$ be any two given queries in the domain $[0, 100]$. Let $[19, 21]$ and $[29, 81]$ be the predictions for these two queries. Intuitively the second prediction ($[29, 81]$) appears to be more accurate than the first one ($[19, 21]$) even though the other factors are same for both these queries. Therefore, a good evaluation metric should take into account the *query interval.*

Definition 2. For a given domain $\mathcal{D} = [\mathcal{D}_{low}, \mathcal{D}_{high}]$, if a given query $Q = [s, f]$ is predicted as $Q' = [s', f']$, we define the prediction error $Err_{Q,Q'}$ in the following way.

$$Err_{Q,Q'} = \frac{|s - s'| + |f - f'|}{\beta \times (\mathcal{D}_{high} - \mathcal{D}_{low}) + \gamma \times (f - s)}$$

Here, β and γ are two real valued constants such that $0 < \beta \leq 1$ and $0 \leq \gamma \leq 1$. we purposefully do not allow β to be equal to *zero* in order to prevent the denominator in *Definition 2* from going to *zero.* Since we implicitly assume a non-empty domain (i.e., $|\mathcal{D}| > 0$), the denominator will always be greater than *zero* as long as $\beta > 0$.

It should be noted that β and γ are placed to restrict the influence of *Domain Interval* and *Query Interval* over the prediction error. For example, without these constants in place, a very large *Domain Interval* can utterly dominate a batch of queries with smaller intervals in the error measure. But, putting control parameters such as β and γ can prevent this domination under the current error definition.

Definition 3. A prediction Q' of a given query Q is *Correct* under the *ϵ-tolerant metric if and only if* the following condition holds.

$$Err_{Q,Q'} \leq \epsilon$$

Interestingly, according to Def. 3, a prediction $[s', f']$ of a given query $[s, f]$ is *correct* under the *0-tolerant metric if and only if* the condition $(s = s') \wedge (f = f')$ holds. Using different values of ϵ, we get a family of error measures which can be used to evaluate an attack mechanism such as ours.

6. EXPERIMENTS & RESULTS

6.1 Experimental Setup

We use two publicly available real datasets in our experiments. First of these datasets is the *Adult Dataset* [3] from the *UCI Machine Learning Repository.* We choose the *North Carolina Voter Dataset* [21] as our second real dataset. The Adult Dataset is a well used personnel dataset with 48842 data instances. The North Carolina Voter Dataset, on the other hand, is a publicly available dataset of real personal identifiers that contains 6190504 instances. Furthermore, we use age demographic statistics [2] of North Carolina to attack range queries on *North Carolina Voter Dataset* to demonstrate the efficacy of our attack model under a more real life settings.

In all these datasets, we choose *age* as our query attribute. Even though all these domains are around $[0, 100]$, our attack approach is equally applicable to attributes with larger domain such as *salary.* A larger domain (e.g., $[0, 100000]$) can easily be discretized into a smaller domain like that of *age.* For example, an attribute *salary* with Domain $\mathcal{D} = [0, 100000]$ can be converted to a shorter domain by dividing each domain value by 1000 and rounding up to the closest integer value. This way, an adversary can attack the converted domain to get a reasonable estimate of the range queries on the larger domain. Of course, this method inherently contains an error margin (in this case, ± 500), but for a larger domain size, this error margin seems quite reasonable. Therefore, we argue that **our attack approach is equally applicable to attributes with larger domain size**.

6.1.1 Query Generation.

In our experiments, we use two different query generation methods. For most of the experiments reported in this paper, we generate queries uniformly from the attribute domain. For a discrete domain of size n, there can be $\frac{1}{2}n(n+1)$ number of distinct queries. To generate a uniform query set, we uniformly choose two distinct values s and t from the domain and add a *range query* of the form $[min(s,t), max(s,t)]$ in our query set.

In §6.2.5, we investigate how a non-uniform query load affects the prediction accuracy of our proposed model. Furthermore, we devote §6.2.5 to explain how we generate non-uniform query sets. It should be noted that the query generation process is implicitly assumed to be uniform in all our experiments unless otherwise stated.

6.1.2 Simulated Annealing.

The attack model described in §3 is a hard one to solve. In fact, a simpler version of this problem has been shown to be $NP-Complete$ in [17]. The brute-force approach seems extremely inefficient in solving this optimization problem. Therefore, we alternatively seek to solve an approximation of the given problem using *Simulated Annealing* [19]. We justify our choice by empirically showing that *Simulated Annealing* is able to produce a reasonable solution very efficiently.

6.1.3 Batch Mode Execution.

As we have already mentioned in §3.3, a query set of size m has a constraint set size of $m+\frac{1}{2}m(m-1)+k$. *Simulated Annealing* generates a valid candidate solution at each round that respects each of these constraints. Therefore, generating a valid candidate solution may prove to be extremely inefficient for large values of m. On the other hand, a naive execution approach requires $10-20\%$ of the total queries to be known beforehand (*known query set*) to obtain a reasonable query prediction accuracy. Again, knowing this many queries beforehand may turn out to be quite impractical.

To circumvent both these problems, we solve the optimization problem in batches instead of a single execution. If we have a query set of size m with k known queries, we divide the $(m-k)$ unknown queries into l equi-size batches. Finally, we add the k known queries to each of these individual batches. We denote the size of each individual batch by B_{size}. Finally, we run each of these individual batches independently and agglomerate their results after their executions to get the prediction for the overall query load. Since both k and $Batch_{size}$ are user defined parameters and are intended to respect the inequality $k < B_{size} << m$, this batch mode execution paves the way to run unlimited number of queries with a very small constant number of known queries very efficiently. Furthermore, the individual batches can be run in parallel to improve the overall execution efficiency. It should be noted that the number of individual batches l is defined by (3).

$$l = \left\lceil \frac{(m-k)}{(B_{size}-k)} \right\rceil \tag{3}$$

Finally, to summarize the above discussion on batch execution, we list the following two important advantages that batch mode operation has to offer.

(a) As the size of each batch is kept to a smaller value, *Simulated Annealing* can run extremely fast for each individual batch.

(b) Since we are reusing the known queries, a very few number of known queries can be used to successfully predict a large set of unknown queries.

To further underscore the fact that batch mode execution reduces the number of known queries required, we like to mention that most of the experiments reported in this paper use **only 2 known queries for an overall query set size of 100. In fact, our model could ascertain a query set of size** 150 **with more than** 70% **accuracy using only 2 known queries** (please see Fig. 7). Therefore, it is quite apparent that because of batch mode execution, an extremely few number of known queries (e.g., 2) are required to ascertain even a very large set of queries.

6.1.4 Known Query Generation.

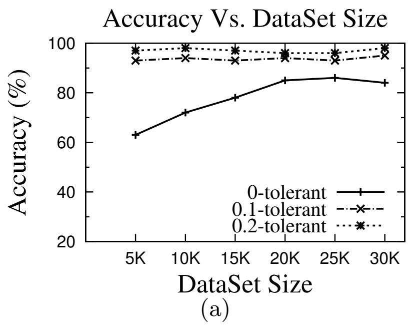
As mentioned earlier, our model requires a few queries to be known beforehand. These known queries work as pivot elements in assigning correct values to other queries. Therefore, the choice of known query set can largely affect the accuracy of our model. Unfortunately, we did not find any publicly available real query sets to use in our experiments. However, we argue that an adversary might not have any control to the set of known queries. That is, it is most likely that an adversary does not get to choose the set of known queries himself. With this argument in mind, we generate the known query set uniformly from the query set in all our experiments.

6.1.5 Execution Time.

We do not use any parallel processing in any of the experiments noted in this paper. All the experiments mentioned in this paper were run in a AMD Phenom II $X6$ 1045T Windows 7 machine clocked at 2.70 GHz with 8 GB of system memory. Each of the individual executions took no more than 1 hour to finish. Therefore, we argue that even a reasonable serial implementation of our attack model is quite efficient. However, the *Batch Mode Execution* leaves us adequate room to improve the efficiency even further by running the batches in parallel.

6.1.6 Evaluation Metric

In this paper, we evaluate our proposed attack model using three such evaluation metrics, namely 0-tolerant metric, 0.1-tolerant metric and finally, 0.2-tolerant metric as defined in §5. Furthermore, Since we use age as the query attribute, the domain interval does not dominate the query interval in our experiments. Therefore, we use a value of 1 for both β and γ in all our experiments. However, we strongly advise a bigger value of γ and a relatively smaller value of β for large domains.



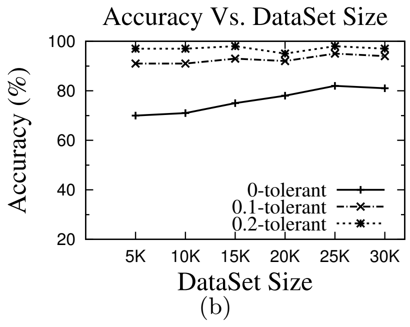


Figure 7: Accuracy for various dataset size. a) Adult dataset. b) North Carolina Voter dataset.

6.2 Experimental Results

In this section, we present a thorough empirical evaluation of our attack approach. Default parameters for all these experiments are as follows. Query set size: 100, known query set size: 2. Our empirical evaluation suggests that the batch size value of 5 is an ideal choice considering the overall accuracy and efficiency (see §6.2.3). These default values mentioned above are implicitly assumed to be used in all the experiments unless otherwise mentioned.

6.2.1 Accuracy on Different Datasets

In this section, we describe the accuracy of our attack model on different datasets. For these experiments, we choose datasets of different size uniformly from the *Adult Dataset* and *North Carolina Dataset*. We present the prediction accuracy of our model for various dataset size in Fig. 7. As is evident from Fig. 7, initially the prediction accuracy rises as the dataset size is increased. However, the accuracy does not rise significantly when the dataset size reaches to 10000 mark. Beyond that, any subsequent increase to the dataset size hardly improves the prediction accuracy. It is quite apparent from the figure that our model is able to predict around 80% of the queries correctly even under the *0-tolerant metric* with a reasonable dataset size. The performance is around high 90 percentile when we consider $0.1 - tolerant$ and $0.2 - tolerant$ metrics. Since most of the real-world database consists of thousands of tuples, our proposed model is likely to succeed with very high probability.

6.2.2 Changing Known Query Set Size

In these set of experiments, we choose known query set of different size and investigate its effect on query prediction accuracy. As Fig. 8 (a) & (b) suggests, prediction accuracy quickly rises as we increase the known query set size from 0 to 2. After that point, the model has enough information it needs and the accuracy does not improve significantly. This is why, we use 2 as our default known query set size parameter. Again, it should be noted that since our query load consists of 100 queries, we only need 2% of them to be known beforehand. Furthermore, if we increase the query set size, we need even a less percent of queries to be known beforehand.

Table 2: Discrepancies in Age Distribution between NC Voter Dataset and NC Census Data

Age Range	NC Voter Data (thousands)	NC Census Data (thousands)
20-24	570	661
25-29	626	627
30-34	637	619
35-39	620	660
40-44	684	667
45-49	669	699
50-54	678	670
55-59	631	601
60-64	567	538
65-69	478	403
70-74	345	295
74-79	264	224
80-85	214	165

6.2.3 Effect of Batch Size

In this section, we report the effect of various batch size on the adult dataset and the North Carolina voter dataset. We run our attack on both these datasets with batch size varying from 5 to 10. It can be readily seen from Fig. 8 (c) & (d) that changing batch size only affects the accuracy very insignificantly. However, as the batch size increases, the execution time of the attack also increases. For this reason, we have chosen 5 as the default batch size for all the other experiments in this paper to favor better execution time.

6.2.4 Imperfect Background Information

All the experiments reported so far have been conducted under the assumption that the attacker has access to an exact plaintext distribution. In this section, we relax such an assumption by restricting an attacker's possession of perfect background information. In this section, we present two experiments where an attacker possesses two different kinds of imperfect background information. In the first experiment, we assume that an attacker is attacking range queries on North Carolina voter dataset. However, we assume that the attacker is completely unaware of the plaintext data distribution of this dataset. Rather, he uses the North Carolina demographic age statistics [2] to formulate his attack. In the second experiment, we assume that the attacker has a noisy version of the plaintext distribution at his disposal.

Using a different age distribution. In this experiment, we present a very practical attack using real world datasets to showcase the efficacy of our attack model. Here, we assume that an attacker is attacking range queries performed on the *age* attribute of the North Carolina voter dataset [21]. Again, we assume that he is completely unaware of the age distribution of this voter dataset. However, we show that an attacker can still launch a reasonably successful attack

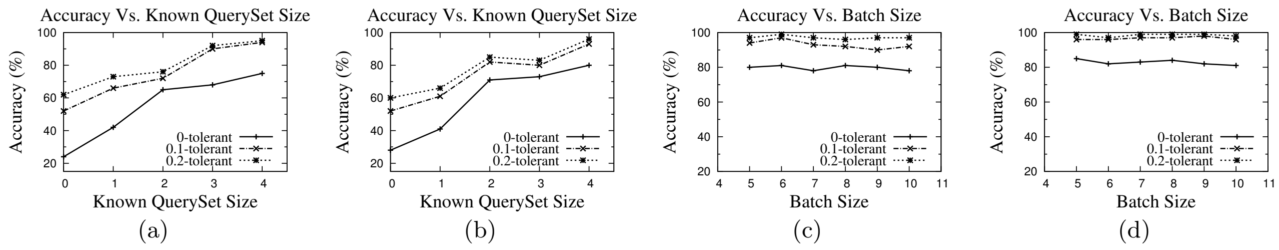


Figure 8: a) Effect of known query size on NC Voter dataset. b) Effect of known query size on adult dataset. c) Effect of batch size on NC voter dataset. d) Effect of batch size on adult dataset.

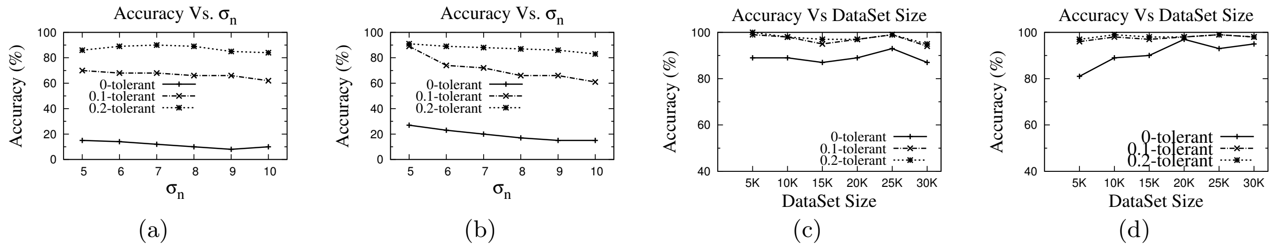


Figure 9: a) Noisy background knowledge on NC voter dataset. b) Noisy background knowledge on adult dataset. c) Zipfian query distribution on NC voter data. d) Zipfian query distribution on adult data.

Table 3: Accuracy for Imperfect Background Information

Query Set	0-Tolerant (%)	.1-Tolerant (%)	.2-Tolerant (%)
1	33	81	93
2	30	77	89
3	34	78	91

by using the age distribution of the people of North Carolina from the U.S. Census Bureau [2]. Before we present the empirical results, we like to point out that although both these datasets are focused on the inhabitants of North Carolina, there is a significant amount of discrepancy between the age statistics presented in these two datasets which is presented in $Table$ 2.

However, our empirical results show that an attacker can still launch a successful attack regardless of this discrepancy. We present the accuracy of such attacks on three different uniform query loads of size 100 in Table 3. It should be noted that the entire NC voter dataset has been used for this attack. Because of the discrepancies between the datasets, the $0 - tolerant$ accuracy of the attack model lies around 30%. However, considering the fact that an attacker uses a completely different dataset to approximate the real age distribution, the attack seems reasonably effective. Again, high accuracy values for $0.1 - tolerant$ and $0.2 - tolerant$ measure further indicates that a very high percantage of queries are reasonably approximated by the attack.

Using noisy background information. In this experiment, we assume that the attacker has a noisy version of the plaintext distribution of the query attribute. More specifically, we change the actual $|S_{i,j}|$ values by adding a noise term $\mathcal{N}(0, \sigma_n^2)$. More specifically, for a given value of $|S_{i,j}|$, we generate a noisy version $|S'_{i,j}|$ in the following way.

$$|S'_{i,j}| = |S_{i,j}| \left(1 + \frac{\mathcal{N}(0, \sigma_n^2)}{100}\right)$$

Here, σ_n is the *Standard Deviation* that determines the amount of noise. Fig. 9 (a) & (b) shows the prediction accuracy under noisy background information for three different datasets. Although the accuracy suffers under the *0-tolerant metric*, our model still does quite well under the *0.1-metric* and *0.2-metric*. Therefore, we argue that our approach is capable of inferring a reasonable amount of sensitive information for a noisy version of the background information.

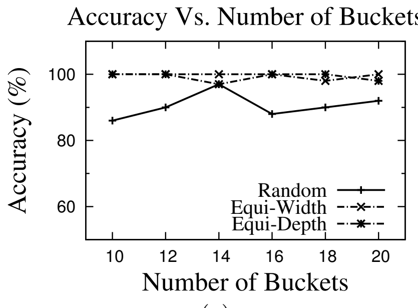
6.2.5 Non-Uniform Query Load

All the experiments discussed so far assume an uniform query generation process. But, this assumption of uniform query load may not hold for all practical applications. Therefore, the question arise, how a non-uniform query load affects the prediction accuracy of our proposed model? In this section, we answer this question by generating different query loads using *Zipfian Distribution* and conducting several experiments using such query loads on different datasets.

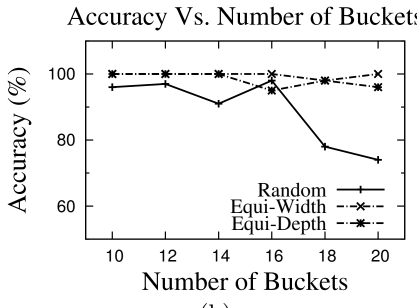
To use the *Zipfian Distribution*, we choose a total ordering among all the possible queries uniformly. That is, we uniformly assign a rank to each of the $\frac{n(n+1)}{2}$ possible queries. Finally, we use *Zipfian Distribution* to choose queries in our query load. That is, if a particular query has a rank j in the ordering, we choose this query in our query set with probability $P_j = \frac{\frac{1}{j}}{N}$, where $N = \sum_{i=1}^{t} \frac{1}{i}$ is the *Normalization Factor* and $t = \frac{n(n+1)}{2}$ is the total number of all possible queries.

Fig. 9 depicts the prediction accuracy of our model for *Zipfian Query Distribution* on the real datasets, namely the *Adult* dataset and the *North Carolina Voter* dataset. The prediction accuracy of *non-uniform query load* on these two datasets is very high even under the rigorous $0 - tolerant$

metric. Therefore, Fig. 9 (c) & (d) indicates that our approach is capable of exhibiting reasonable accuracy under any given queryload.



(a)



(b)

Figure 10: a) NC voter data. b) Adult Data.

6.3 Attack Against Bucketization

Figure 10 reports the accuracy of attacks against bucketization on different datasets for various bucket size. The metric used to evaluate these results is very similar to the one we defined for the range queries. However, we replace the *query interval* parameter with the *bucket interval* in this evaluation metric. Furthermore, the empirical results presented in this section have been evaluated under the $0-tolerant$ metric only. In these experiments, we consider three bucketization approaches, namely the *random bucketization*, the *equi-width bucketization* and the *equi-depth bucketization* [20]. Our empirical results show that our approach can successfully ascertain the bucket boundaries for all these three bucketization methods with very high accuracy even under the $0-tolerant$ metric. Although protocols like [15,16] favors equi-width or equi-depth bucketization, our empirical results show that an adversary has higher probability of success for these two approaches than that of the *Random* bucketization. Therefore, we conclude that *random* bucketization approach is more resistant to access pattern based inference attacks than the *equi-width* and *equi-depth* ones.

7. RELATED WORK

Search Over Encrypted Text. Goldreich et al. proposed their oblivious RAM model in [14]. Unfortunately, even the most efficient variant of the oblivious RAM model is impractical for real world applications. Vimercati et al. proposed a secure B+-tree based shuffle index technique in [12]. A similar technique has been also proposed in [28]. Song et al. were the first to propose an efficient encryption scheme in [24] that allows keyword search over a remote encrypted document set. Following their footsteps, many new schemes were proposed with different levels of security guarantees (e.g., [9,13]).

DAS Model. Hacigümüş et al. are the first to propose a notion of encrypted database in [15], which supports a limited execution of SQL queries in the remote server. Damiani et al. proposed a metric called *inference exposure* to quantitatively measure the extent of information an adversary can gain probabilistically about the sensitive data [8,10]. Again, Hore et al. proposed a privacy-preserving index for range query in [16]. Even though all these techniques shares the same basic structure, they significantly differ in the bucket construction.

There are quite a few schemes that use encryption techniques to support range query execution(e.g., [4–6,23]). The most notable among these techniques is the one proposed by E. Shi et al. in [23]. Their protocol can handle multi dimensional range queries. Again, there are several techniques in the literature [4–6] that use *Order Preserving Encryption* to facilitate range queries on *encrypted data*. None of these techniques allow any false positives, and therefore are quite efficient. In [22], Popa et al. presented CryptDB, a comprehensive DBMS that supports SQL query execution in encrypted database incurring a very modest overhead. The *order preserving encryption* presented in [5] has been used in this work to facilitate the range query execution. Also, Stephen Tu et al. [26] proposed a complete encrypted database for executing analytical workloads. Since both Monomi and CryptDB uses precise query protocols to facilitate range query, they are vulnerable to our attack.

Attacks on outsourced database. Islam et al. proposed a new attack model in [17] which leverages access pattern leakage to disclose significant amounts of sensitive information. Although the basic attack approach of [17] is similar to this attack, the context as well as the methodologies used are quite different. Again, the model described in [17] is only applicable to keyword search. Our work, on the other hand, is tailored towards the more general concept of range queries. Again, Dautrich et al. presented an attack on precise query protocols to ascertain the order of the query attribute values in [11]. However, their attack can not be applied to imprecise protocols such as bucketization. Furthermore, the objective of [11] is to obtain the ordering of etuples based on the query attribute. The objective of our approach is to ascertain the encrypted range queries. Again, their attack does not guarantee a solution for any given range query load and may require a very large number of queries to ascertain a complete ordering. Our approach, on the other hand, is able to disclose encrypted range queries with reasonable accuracy for query load with size as low as 5.

8. CONCLUSIONS

In this paper, we propose an attack model that leverages query access pattern to disclose significant amount of sensitive information about the encrypted range queries on outsourced databases. We empirically show the efficacy of our model by applying it to several real world datasets. Our empirical results suggest that the proposed model can predict encrypted range queries with high accuracy under a rigorous evaluation metric. Furthermore, we also show that a variation of this attack can be used on the bucketization approach to disclose the bucket boundaries with high accuracy.

Therefore, we argue that in cases where data security is the primary concern, it is better to use secure, albeit inefficient protocols like that of *Oblivious RAM*. However, in controlled settings where an adversary is guaranteed not to possess significant amount of prior knowledge about the plaintext data distribution, a user may prefer to use sophisticated and efficient protocols.

9. ACKNOWLEDGMENTS

This work was partially supported by Air Force Office of Scientific Research FA9550-12-1-0082,National Institutes of Health Grants 1R0-1LM009989 and 1R01HG006844, National Science Foundation (NSF) Grants Career-CNS-0845803, CNS-0964350, CNS-1016343, CNS-1111529, CNS-1228198 and Army Research Office Grant W911NF-12-1-0558.

10. REFERENCES

[1] Google BigQuery. `http://code.google.com/apis/bigquery/`.

[2] Profile of general population and housing characteristics: 2010 demographic profile data. `http://factfinder2.census.gov/bkmk/table/1.0/en/DEC/10_DP/DPDP1/0400000US37`.

[3] Adult Data Set. Uci machine learning repository. `http://archive.ics.uci.edu/ml/datasets/Adult`.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD 2004*, pages 563–574. ACM, 2004.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT 2009*, volume 5479, pages 224–241, 2009.

[6] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO 2011*, volume 6841, pages 578–595. Springer, 2011.

[7] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, volume 4392, pages 535–554. Springer, 2007.

[8] Ceselli, Damiani, di Vimercati, Jajodia, Paraboschi, and Samarati. Modeling and assessing inference exposure in encrypted databases. *ACMTISS: ACM Transactions on Information and System Security*, 8, 2005.

[9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS*, pages 79–88, 2006.

[10] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *CCS 2003*, pages 93–102. ACM Press, 2003.

[11] J. L. Dautrich Jr and C. V. Ravishankar. Compromising privacy in precise query protocols. In *EDBT*, pages 155–166. ACM, 2013.

[12] S. D. C. di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati. Efficient and private access to outsourced data. In *ICDCS*, pages 710–719. IEEE Computer Society, 2011.

[13] E. Goh. Secure indexes. *Cryptology ePrint Archive*, (Report 2003/216), 2003.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *JACM: Journal of the ACM*, 43, 1996.

[15] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *SIGMOD*, pages 216–227, 2002.

[16] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB 2004*. Morgan Kaufmann Publishers, 2004.

[17] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*. ISOC, 2012.

[18] M. Kantarcioglu and C. Clifton. Security issues in querying encrypted data. In *DBSec*, 2005.

[19] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–679, 1983.

[20] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 89–103. Springer, 2006.

[21] North Carolina Voter Data. `ftp://www.app.sboe.state.nc.us/data`.

[22] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *SOSP*, 2011.

[23] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364. IEEE Computer Society, 2007.

[24] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, 2000.

[25] E. Stefanov and E. Shi. Oblivistore: High performance oblivious cloud storage. In *Proc. of IEEE Symposium on Security and Privacy*, 2013.

[26] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. In *PVLDB*, pages 289–300, 2013.

[27] P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In *CCS*, pages 139–148. ACM, 2008.

[28] K. Yang, J. Zhang, W. Zhang, and D. Qiao. A light-weight solution to preservation of access pattern privacy in un-trusted clouds. In *ESORICS*, volume 6879, pages 528–547. Springer, 2011.

[29] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving queries on encrypted data. In *ESORICS 2006*, pages 479–495. Springer, 2006.