Query Processing in Private Data Outsourcing Using Anonymization

Ahmet Nergiz, Chris Clifton

Query Processing in Private Data Outsourcing Using Anonymization

Ahmet Erhan Nergiz[1] and Chris Clifton[2]

[1] Purdue University, West Lafayette, IN 47907
anergiz@cs.purdue.edu
[2] Purdue University, West Lafayette, IN 47907
clifton@cs.purdue.edu

Abstract. We present a query processing scheme in a private data outsourcing model. We assume data is divided into identifying and sensitive data using an anatomy approach[20]; only the client is able to reconstruct the original identifiable data. The key contribution of this paper is a relational query processor that minimizes the client-side computation while ensuring the server learns nothing violating the privacy constraints.

Keywords: privacy, anonymization, data outsourcing, anatomy model

1 Introduction

Data outsourcing is a growing business. Cloud computing developments such as Amazon Relational Database Service promise further reduced cost. However, use of such a service can be constrained by privacy laws, requiring specialized service agreements and data protection that could reduce economies of scale and dramatically increase costs.

Most privacy laws apply to data "relating to an identified or identifiable natural person"[6], data that cannot be directly or indirectly linked to an individual is not restricted. Some laws are even more specific; the U.S. Healthcare laws apply only to identifiable *health information*[10]. We propose a private data outsourcing approach where the link between identifying information and sensitive (protected) information is encrypted, with the ability to decrypt this link residing only with the client. As the server no longer has access to *individually identifiable* protected information, it is not subject to privacy laws, and can offer a service that does not need to be customized to the needs of each country- or sector-specific requirements; any risk of violating privacy through releasing sensitive information tied to an individual remains with the client.

We admit that the legal and privacy issues of this model are open to debate (although some laws suggest the appropriateness of this model; U.S. laws applying to educational institutions specifically allow disclosure of "directory information" on an opt-out basis [7]); such debate is not in the scope of this paper. We propose a data model based on *anatomization*[20]. This divides data into *anatomy groups*, separates identifying and sensitive data into two tables, and provides a join key at the group level (see Figure 2.) We add an encrypted key that

does allow reconstructing the record, but the ability to decrypt and reconstruct resides only at the client. Note that this model can support a variety of privacy constraints, including k-anonymity[18, 19], discernibility/l-diversity[17, 14], and t-closeness[13]. While the original anatomization paper just considered a single table, extending this to a full relational database has been explored[16].

This paper presents a relational query processor operating within this model. The goal is to minimize communication and client-side computation, while ensuring that the privacy constraints captured in the anatomization are maintained. At first glance, this is straightforward: standard relational query processing at the server, except that any joins involving the encrypted key must be done at the client; an appropriate distributed query optimizer should do a reasonably good job of this. However, two issues arise that confound this simple approach:

1. By making use of the anatomy groups, and the knowledge that there is an one-to-one mapping (unknown to the server) between tuples in such groups, we can perform portions of the join between identifying and sensitive information at the server without violating privacy constraints, and
2. Performing joins at the client and sending results back to the server for further processing (as might be recommended by a distributed query optimizer) can violate privacy constraints.

We first give the threat model and related work in consequent subsections and then provide definitions and notations for an anatomized database in Section 2. In Section 3, we show how standard relational algebra operations can be performed to lower client-side cost using issue 1. We conclude our paper with Section 4.

1.1 Threat Model

In our private data outsourcing model, a data owner (i.e., client) first anonymizes the database such that individually identifiable links are encrypted besides the anonymization of such links. The data owner sends the modified database to a semi-honest third party (i.e., server) to delegate most of the query processing. The server is only allowed to try to infer additional information than that is allowed by the anonymization technique we use and it is assumed not to return incorrect or/and incomplete result, or alter the protocol in an attempt to gain information. Moreover, the server does not modify the database that the data owner sends at the beginning of the protocol.

1.2 Related Work

Private data outsourcing also known as database-as-a-service model was first introduced by Hacigumus et al. [9]. They used bucketization over encrypted database that allows the server to partially execute queries on the behalf of the client. There is a yet unmeasured trade-off between efficiency of the system and the privacy of individuals directly related to the size and the contents of

each bucket of encrypted values. Although, there has been an effort to address the optimization of this trade-off in [11], no privacy measurement showing the amount of information leakage is given. However, Damiani et al. [4] proposed another technique that uses hashing for bucketization and encrypted B+ trees for indexing. They give an aggregate metric showing the exposure of the database contents in various adversarial models. However, we note that an aggregate exposure metric fails to ensure the privacy of each individual's identity.

Instead of using encryption, Aggarwal et al. [1] proposed vertical fragmentation to hide functional dependencies from an adversary. They require two non-colluding servers to send each fragment. Another approach described in [3] is to fragment the tables into partitions and have the client store a small partition storing the sensitive values. The rest is stored in the server in plaintext. They prove that finding the optimal partitioning is NP-hard and give a heuristic solution instead.

As far as we know the closest idea to ours is in [12]. They give an l-diverse partitioning scheme based on anatomization[20] for a single table having multiple sensitive attributes. Our work is orthogonal to their work such that we give detailed query evaluation strategies given such an l-diverse partitioning scheme exists for multirelational databases.

2 Data Outsourcing using Anatomy

As stated before, we assume use of the anatomy model[20] to meet privacy constraints. Making this work for multiple tables does demand extra thought; a solution for this is given in [16]. This paper assumes an anatomized database meeting privacy constraints; we now present relevant definitions and notations (based on [16]) that we will use in describing query processing.

2.1 Definitions and Notations

Definition 2.1 (Equivalence class/QI-group). *An equivalence class, E_j, is a subset of tuples in table T such that $T = \bigcup_{j=1}^{m} E_j$ and for any pair, (E_{j_1}, E_{j_2}) where $1 \leq j_1 \neq j_2 \leq m$, $E_{j_1} \cap E_{j_2} = \emptyset$.*

Definition 2.2 (l-diversity). *A set of equivalence classes is said to be* **l-diverse***, if each equivalence class, E_j where $1 \leq j \leq m$, satisfies*

$$\forall v \in \pi_S E_j,\ f(v, E_j)/|E_j| \leq 1/l$$

where S is the sensitive attribute in T, $f(v, E_j)$ returns the frequency of v in E_j and $|E_j|$ is the number of tuples in E_j.

We use a variation of the definition of Anatomy in [20].

Definition 2.3 (Anatomy). *Given a table T partitioned into m equivalence classes using l-diversity without generalization, anatomy produces a quasi-identifier table (QIT) and a sensitive table (SNT) as follows. QIT has schema*

$$(A_1, \ldots, A_d, GID, SEQ)$$

where $A_i \in Q_T$ *for* $1 \leq i \leq d = |Q_T|$, Q_T *is the set of identifying attributes in* T, *GID is the group id of the equivalence class and SEQ is the unique sequence number for a tuple. For each* $E_j \in T$ *and each tuple* $t \in E_j$, *QIT has a tuple of the form:*

$$(t[1], \ldots, t[d], j, s)$$

The SNT has schema

$$(HSEQ, GID, A_{d+1})$$

where A_{d+1} *is the sensitive attribute in* T, *GID is the group id of the equivalence class and HSEQ contains the outputs of* $\mathrm{H}_{\bar{k}}(s)$ *defined as in Definition 2.4 where* s *is the corresponding unique sequence number in QIT for a tuple. For each* $E_j \in T$ *and each tuple* $t \in E_j$, *SNT has a tuple of the form:*

$$(\mathrm{H}_{\bar{k}}(s), j, v)$$

For instance, the anatomy anonymization of person specific tables *Physician* and *Patient* in Figure 1 is shown Figure 2.

Doctor	*Gender*	*Patient*
Alice	Female	Ike
Carol	Female	Eric
Bob	Male	Olga
Dave	Male	Kelly
Carol	Female	Faye
Alice	Female	Mike
Dave	Male	Jason
Carol	Female	Max

(a) Physician

Patient	*Age*	*City*	*Disease*
Ike	41	Dayton	Cold
Eric	22	Richmond	Fever
Olga	30	Lafayette	Flu
Kelly	35	Lafayette	Cough
Faye	24	Richmond	Flu
Mike	47	Richmond	Fever
Jason	45	Lafayette	Cough
Max	31	Lafayette	Flu

(b) Patient

Fig. 1. Original Database

Note that we show a (keyed) hash as the "join key" between the two subtables. We use HMAC [2] for hiding the join links due to the efficiency of cryptographic hash functions; one could also encrypt the key using a standard mechanism (with nonces) or a Probabilistic Encryption method [8] to achieve semantic security. We formally describe this problem below.

Definition 2.4 (Hiding Join Link). *Given two tables* T_1 *and* T_2 *having the same cardinality and a joining attribute, SEQ in domain* D, *mapping* T_1 *1:1 to* T_2, *a function* $\mathrm{H} : \bar{k} \times D \rightarrow D'$ *is said to hide the join link, SEQ, once each value* v *in* $T_2.SEQ$ *is updated with* $\mathrm{H}_{\bar{k}}(v)$ *if*

- *Without knowing the secret* $\bar{k}$ *used in* H, *it is hard to join* T_1 *with* T_2 *on attribute SEQ.*
- *In case* H *can be applied to inputs with unbounded length, it is hard to encounter two values,* v_1 *and* v_2, *such that* $\mathrm{H}_{\bar{k}}(v_1) = \mathrm{H}_{\bar{k}}(v_2)$.

Remark 1. When HMAC used, one needs to apply HMAC to the attribute $T_1.SEQ$ to join T_1 and T_2 since HMAC is hard to invert even the used key $\bar{k}$ is known whereas when encryption is used, one needs to decrypt each $\mathrm{H}_{\bar{k}}(v)$ in T_2 and then T_1 and T_2 can be joined since the strategy used in HMAC cannot be used in randomized encryptions where encrypting the same value each time results in a different ciphertext based on the random used during the encryption process.

Doctor	*Gender*	*GID*	*SEQ*
Alice	Female	1	1
Carol	Female	1	2
Bob	Male	2	3
Dave	Male	2	4
Carol	Female	3	5
Alice	Female	3	6
Dave	Male	4	7
Carol	Female	4	8

(a) $\text{Physician}_{\text{QIT}}$

HSEQ	*GID*	*Patient*
$\mathrm{H}_{\bar{k}_1}(1)$	1	Ike
$\mathrm{H}_{\bar{k}_1}(2)$	1	Eric
$\mathrm{H}_{\bar{k}_1}(3)$	2	Olga
$\mathrm{H}_{\bar{k}_1}(4)$	2	Kelly
$\mathrm{H}_{\bar{k}_1}(5)$	3	Faye
$\mathrm{H}_{\bar{k}_1}(6)$	3	Mike
$\mathrm{H}_{\bar{k}_1}(7)$	4	Jason
$\mathrm{H}_{\bar{k}_1}(8)$	4	Max

(b) $\text{Physician}_{\text{SNT}}$

Patient	*Age*	*City*	*GID*	*SEQ*
Ike	41	Dayton	1	1
Eric	22	Richmond	1	2
Olga	30	Lafayette	2	3
Kelly	35	Lafayette	2	4
Faye	24	Richmond	3	5
Mike	47	Richmond	3	6
Jason	45	Lafayette	4	7
Max	31	Lafayette	4	8

(c) $\text{Patient}_{\text{QIT}}$

HSEQ	*GID*	*Disease*
$\mathrm{H}_{\bar{k}_2}(1)$	1	Cold
$\mathrm{H}_{\bar{k}_2}(2)$	1	Fever
$\mathrm{H}_{\bar{k}_2}(3)$	2	Flu
$\mathrm{H}_{\bar{k}_2}(4)$	2	Cough
$\mathrm{H}_{\bar{k}_2}(5)$	3	Flu
$\mathrm{H}_{\bar{k}_2}(6)$	3	Fever
$\mathrm{H}_{\bar{k}_2}(7)$	4	Cough
$\mathrm{H}_{\bar{k}_2}(8)$	4	Flu

(d) $\text{Patient}_{\text{SNT}}$

Fig. 2. Anatomized Database

In Theorem 2.1, we show that the probability of having a collision in the hash values of any equivalence group is negligible which in return proves our model is correct with overwhelming probability.

Theorem 2.1 (Correctness). *Given QIT, SNT tables each having n tuples and structured as in Definition 2.3, and HMAC with l-bit outputs used for hiding the actual join link between QIT and SNT; one can construct the original table T by joining $QIT_{updated}$ and SNT with overwhelming probability if $2^l \gg n$ where $QIT_{updated}$ is computed by updating each value v in QIT.SEQ with $\mathrm{H}_{\bar{k}}(v)$ value.*

Proof. T can only be constructed if $\langle t_1.GID, t_1.SEQ\rangle$ pair matches with exactly one tuple t_2 of SNT for each tuple t_1 of $QIT_{updated}$. Hence the pair $\langle t_1.GID, t_1.SEQ\rangle$ needs to be unique across the tuples of $QIT_{updated}$. The same

is also true for $\langle t_2.HSEQ, t_2.GID\rangle$ in SNT. Since all sequence values in QIT.SEQ is unique, the only case that there are more than one same $\langle t_1.GID, t_1.SEQ\rangle$ value is when there is a collision in one of the equivalence class. Recall that $T = \bigcup_{j=1}^{m} E_j$ and let c be $max(|E_1|, \ldots, |E_m|)$. Then the probability, $\mathcal{P}$, of not having the same $\mathrm{H}_{\bar{k}}(v)$ value for any v in any equivalence class in $QIT_{updated}$ or SNT can be approximated by using the birthday problem analysis [5].

$$\mathcal{P} \approx \left(e^{-\binom{c}{2}/2^l}\right)^m \approx \left(e^{-cn/2^{l+1}}\right)$$

Considering the current world population and having a tuple for each person in the world, the largest database can hold at most 2^{33} tuples. When $l = 160$ and $n = 2^{33}$ and assuming c is a small constant, $\mathcal{P} \approx 1$.

2.2 Privacy Preservation

Given QIT and SNT, a semi-honest adversary can only associate each individual to a sensitive attribute with some probability based on the size of an equivalence class. Lemma 2.1 gives the formulation for this probability.

Lemma 2.1. *Given* $\mathrm{H}_{\bar{k}}(.)$ *is a cryptographic hash function, the probability that a tuple in QIT,* $(t[1], \ldots, t[d], j, s)$*, matches with a tuple in SNT* $(\mathrm{H}_{\bar{k}}(s'), j, v)$ *is*

$$\mathcal{P}((t[1], \ldots, t[d], v) \in T) = f(v, E_j)/|E_j|$$

where $f(v, E_j)$ returns the frequency of v in E_j, $|E_j|$ is the number of tuples in E_j and $\bar{k}$ is the unknown key for the cryptographic hash function, $\mathrm{H}_{\bar{k}}(.)$.

Proof. Each tuple belonging to some equivalence class E_j^{QIT} in QIT, joins with every tuple in the corresponding equivalence class, E_j^{SNT}, in SNT due to the same GID, j. Thus for a tuple $t \in E_j^{QIT}$, $\{t\} \times E_j^{SNT}$ is the set of all the tuples that t contributes to $QIT \bowtie SNT$. Therefore the sample space for t's possible matching sensitive value v is $|\{t\} \times E_j^{SNT}| = |E_j^{SNT}| = |E_j|$. However there exists only one tuple, t', such that $t' \in \{t\} \times E_j^{SNT}$ and $t' \in T$ by Definition 2.3. Due to the first property of function H in Definition 2.4, it is infeasible to guess t' correctly out of $\{t\} \times E_j^{SNT}$ tuples without knowing the key $\bar{k}$ in H to get HSEQ values. Thus, the probability that t matches with sensitive value v in E_j^{SNT} is the count of v in E_j^{SNT} divided by the sample space (i.e., $|E_j|$).

For instance, the probability of the individual represented by the first tuple in $\mathrm{Patient}_{\mathrm{QIT}}$ in Figure 2, $\langle Ike, 41, Dayton\rangle$, having $Cold$ is 1/2 since $|E_1| = 2$ and the frequency of $Cold$ in E_1 is 1 (i.e., $f(Cold, E_1) = 1$)

Theorem 2.2. *The client cannot safely send any information resulting from a join between identifying and sensitive information back to the server, unless such information would provide no benefit to further join processing.*

Proof. Let QIT and SNT be the anatomization of T such that $\forall t_1 \in QIT$; $\exists t_2 \in SNT$, $(t = t_1 \bowtie t_2) \in T$ and the probability, $\mathcal{P}'$, of finding each tuple t from QIT and SNT is $1/k$. Then each equivalence class has k items and there are n/k number of equivalence classes in both QIT and SNT where n is the number of tuples in T. Hence there are $(k!)^{n/k}$ possible tables that can be derived from QIT and SNT and at least one of these tables corresponds to the original table T. Let T_i^j denote each of these possible tables where $1 \leq i \leq (k!)^{n/k-1}$ and $1 \leq j \leq k!$; and $\bar{T}$ denote the set of all T_i^j's. Then T^j denotes all possible tables where an equivalence class, E, has a fixed permutation (i.e., j^{th} permutation of equivalence class, E) and T_i denotes all possible tables where all equivalence classes except E has a fixed permutation (i.e. i^{th} permutation of all equivalence classes except E). Then we get the probability formulas,

$$\mathcal{P}\left\{T_i^j = T \mid T \in T_i\right\} = \frac{1}{k!}$$

$$\mathcal{P}\left\{T \in T^j\right\} = \sum_{i=1}^{(k!)^{n/k-1}} \mathcal{P}\left\{T_i^j = T \mid T \in T_i\right\} \mathcal{P}\left\{T \in T_i\right\} = \frac{1}{k!}$$

Assume a query q' that is $q'(q(T) \bowtie C)$ where C is another table and the client sends the intermediate result $q(T)$ to the server for improved evaluation of q'. If $\forall T_i^j \in \bar{T}\ q(T_i^j) = q(T)$, sending the result of $q(T)$ does not give any benefit to the server since it can compute $q(T)$ by itself. If $q(T_i^j) \neq q(T)$ for some $T_i^j \in \bar{T}$, sending the result of $q(T)$ violates the privacy since $\mathcal{P}\{T \in T^j\} < 1/k!$ due to the fact that $\mathcal{P}\{T_i^j = T\} = 0$. If $\mathcal{P}\{T \in T^j\} < 1/k!$, there is at least one j' such that $\mathcal{P}\{T \in T^{j'}\} > 1/k!$ and therefore $\mathcal{P}'$ is not $1/k$ for all the tuples in E.

$$\mathcal{P}\left\{T \in T^j\right\} = ((k!)^{n/k-1} - 1) \times \frac{1}{(k!)^{n/k}} < \frac{1}{k!}$$

3 Query Operators

Query processing that operates on only the QIT or SNT sub-tables can be performed at the server without raising privacy issues; it is when these must be combined that we must take care. A simple solution is to operate on each independently, then send the results to the client to decrypt and combine. However, we can often do better. We now detail how relational query operations can be performed in ways that minimize the computation performed on the client. Interested reader may refer to our technical report [15] in which we include all proofs and algorithms that we omit in this paper due to the page limit.

3.1 Selection

Selection on a single table T anonymized into QIT and SNT can be broken into selection on QIT, selection on SNT, and selection criteria requiring the join of

the two. The single sub-table selections are performed first. The resulting tables are then queried to determine where an anatomization group contains values that could satisfy the cross-subtable criterion. If so, all possible matching tuples from each group are passed to the client, which can decrypt, join, and complete the selection.

Definition 3.1 (CNF Predicates). *A set of predicate, P, being in CNF form with respect to a table T anonymized as two tables QIT and SNT, has the following form:*

$$P = \bigwedge_{1 \le i \le n} \left(\bigvee_{1 \le j \le m_i} P_i^j \right)$$

where P_i^j is a single-literal clause having a form att op value *or* att op att. *Without losing generality, each set of P_i's are defined further as*

$$P_{QIT} = P_1 \wedge \ldots \wedge P_\alpha, \quad P_{SNT} = P_{\alpha+1} \wedge \ldots \wedge P_\beta, \quad P_{QS} = P_{\beta+1} \wedge \ldots \wedge P_n$$

where P_{QIT} and P_{SNT} are only applicable to attributes of QIT and SNT respectively. P_{QS} contains predicates applicable to both QIT and SNT in each its disjunctions. Let P_i^T be the i^{th} disjunction of single-literal clauses in P_{QS} that is only applicable to the attributes of table T. Then P_{QS} is defined as

$$P_{QS} = \bigwedge_{\beta < i \le n} \left(P_i^{QIT} \vee P_i^{SNT} \right)$$

Definition 3.2 (Server-side Selection Query). *Given QIT and SNT tables derived from a table T using Anatomy model anonymization and a set of predicates P in conjunctive normal form defined as in Definition 3.1, a selection query written as $\sigma_P(QIT, SNT)$ returns two tables, QIT'' and SNT'':*

$$\begin{aligned}
QIT' &= (\sigma_{P_{QIT}}(QIT)) \ltimes (\sigma_{P_{SNT}}(SNT)) \\
SNT' &= (\sigma_{P_{SNT}}(SNT)) \ltimes (\sigma_{P_{QIT}}(QIT)) \\
S_{GID} &= \bigcap_{i=\beta+1}^{n} \left(\pi_{GID}(\sigma_{P_i^{QIT}} QIT') \cup \pi_{GID}(\sigma_{P_i^{SNT}} SNT') \right) \\
QIT'' &= QIT' \bowtie S_{GID} \\
SNT'' &= SNT' \bowtie S_{GID}
\end{aligned}$$

Lemma 3.1. *Given table QIT, and SNT along with a predicate P defined as in Definition 3.1, and tables QIT'' and SNT'' calculated with the steps defined in Definition 3.2; the following property holds*

$$\sigma_{P_{QS}}(QIT'' \bowtie SNT'') = \sigma_P(QIT \bowtie SNT)$$

Definition 3.3 (Client-side Selection Query). *Given QIT'' and SNT'' tables computed by the server and a predicate P_{QS} in conjunctive normal form*

defined as in Definition 3.1 where each P_i in P_{QS} checks at least one attribute from each QIT and SNT table, the client updates each tuple of QIT'' by replacing the values of SEQ attribute with their corresponding keyed hash value (i.e., $s \to \mathrm{H}_{\bar{k}}(s)$). Then the final selection query is written as

$$R = \sigma_{P_{QS}}(QIT''_{updated} \bowtie SNT'')$$

Theorem 3.1. *Given P as in Definition 3.1, QIT, and SNT; R derived according to Definition 3.2 and 3.3 is equal to $\sigma_P(T)$ if the pair $\langle QIT, SNT\rangle$ is an anatomization of table T according to Definition 2.3.*

Example 1. According to Definition 3.2 and 3.3, given query

$$\sigma_{(Age>40)\wedge(Disease=Flu \text{ or } Cough)\wedge(Disease=Cough\vee Age<3)}(\mathrm{Patient}_{\mathrm{QIT}}, \mathrm{Patient}_{\mathrm{SNT}})$$

$P_{QIT} = (Age > 40)$, $P_{SNT} = (Disease = Flu \vee Disease = Cough)$, and $P_{QS} = (Disease = Fever \vee Age < 3)$. QIT' has the 6^{th} and 7^{th} tuples of table $\mathrm{Patient}_{\mathrm{QIT}}$ based on P_{QIT}. 1^{st} tuple is not included since the corresponding group of $\mathrm{Patient}_{\mathrm{SNT}}$ doesn't satisfy the P_{SNT} (which is ensured with semi-join). SNT' has 5^{th}, 7^{th} and 8^{th} tuples of table $\mathrm{Patient}_{\mathrm{SNT}}$. And $S_{GID} = \{4\}$ since none of the tuples in group 3 satisfies the predicate P_{QS}. Then QIT'' has 7^{th} tuple of $\mathrm{Patient}_{\mathrm{QIT}}$ and SNT'' has 7^{th} and 8^{th} tuples of $\mathrm{Patient}_{\mathrm{SNT}}$. After server sends these intermediate results to the client, client updates the SEQ field of QIT'' and computes $R = \langle Jason, 45, Lafayette, 4, \mathrm{H}_{\bar{k}_2}(7), \mathrm{H}_{\bar{k}_2}(7), 4, Cough\rangle$

Cross sub-table correlation The reader may have noticed an apparent issue: This process potentially returns a single value from the QIT and SNT from the server to the client, implying to the server that these are linked. The key is to remember that it is quite possible that these values do not join; the query result could be empty. This only becomes a problem if 1) attributes in QIT are correlated with attributes in SNT, and 2) the server knows of this correlation.

If attributes are not correlated, then the chance that a single tuple selected from a group in QIT based on a query matches a single tuple from the same group in SNT is $1/k$, and the server cannot infer that they match. Even if the values are correlated, if the server does not know of that correlation it must assume the match probability is $1/k$. If the server knows of the correlation, then it can infer that the two values match based on the QIT and SNT values alone, without even processing a query.

An issue arises when the server does not know of the correlation, but repeated queries suggest such a correlation. However, these issues are with the decision on how to anatomize the table, not with the query processing mechanism itself – the proposed query processing mechanism reveals only linkages that the server could discover from only the data, queries, and knowledge of correlations.

3.2 Projection

Projection is at first glance straightforward, as removing attributes can be done independently on each sub-table. The difficulty comes in removing duplicates:

two tuples may be identical in all non-encrypted attributes in QIT (or SNT), but not be a duplicate in the join.

There is an exception when all values in an anatomy group become identical under projection; then only a single tuple representing the entire group needs to be returned. However, this only works if no selection is performed on "projected out" attributes prior to the projection.

We show how projection operator, denoted by π, is processed in case of eliminating duplicates. We also use π^d throughout the paper to denote that the projection operator does not eliminate duplicates. Since calculating π^d is straightforward, we show the processing of π instead.

Definition 3.4 (Server-side Projection Query). *Given QIT and SNT tables derived from a table T using Anatomy model anonymization and a set of attributes A', projection query without duplicates written as*

$$\pi_{A'}(QIT, SNT),\ A' = A'_{QIT} \cup A'_{SNT} \text{ and } SEQ, GID, HSEQ \notin A'$$

returns a set of tables, R:

$$R = \begin{cases} \{R', T'_{QIT}, T'_{SNT}\} & \text{if } A'_{QIT} \neq \emptyset \text{ and } A'_{SNT} \neq \emptyset \\ \pi_{A'_{QIT}}(QIT) & \text{if } A'_{QIT} \neq \emptyset \text{ and } A'_{SNT} = \emptyset \\ \pi_{A'_{SNT}}(SNT) & \text{if } A'_{QIT} = \emptyset \text{ and } A'_{SNT} \neq \emptyset \end{cases}$$

where $R' = \pi_{A'}\left(\sigma_{GID \notin S}\left(R'_{QIT} \bowtie R'_{SNT}\right)\right)$, $T'_{QIT} = \pi^d_{A'_{QIT}, SEQ}\left(\sigma_{GID \in S} QIT\right)$, *and* $T'_{SNT} = \pi^d_{A'_{SNT}, HSEQ}\left(\sigma_{GID \in S} SNT\right)$; *and where* $R'_{QIT} = \pi_{A'_{QIT}, GID}\left(QIT\right)$, $R'_{SNT} = \pi_{A'_{SNT}, GID}\left(SNT\right)$, *and S is defined as*

$$S = \left\{i : \left|\sigma_{GID=i}\left(R'_{QIT}\right)\right| > 1 \wedge \left|\sigma_{GID=i}\left(R'_{SNT}\right)\right| > 1\right\}$$

Lemma 3.2. *Given R' and S as in Definition 3.4,*

$$R' = \pi_{A'}\left(\sigma_{GID \notin S}\left(R'_{QIT} \bowtie R'_{SNT}\right)\right) = \pi_{A'}\left(\sigma_{GID \notin S}\left(QIT_{updated} \bowtie SNT\right)\right)$$

Definition 3.5 (Client-side Projection Query). *Given the set of tables, R, computed by the server; if $|R| = 1$ then the client outputs the only table in R without any processing. Otherwise, the client updates each tuple of T'_{QIT} by replacing the values of SEQ attribute with their corresponding keyed hash value (i.e., $s \rightarrow \mathrm{H}_k(s)$). Then the final result is computed by*

$$R'' = \pi_{A'}\left(R' \cup \pi_{A'}\left(T'_{QIT_{updated}} \bowtie T'_{SNT}\right)\right)$$

Theorem 3.2. *Given $A' = A'_{QIT} \cup A'_{SNT}$, QIT, and SNT; R'' derived according to Definition 3.4 and 3.5 is equal to $\pi_{A'}(T)$ if the pair $\langle QIT, SNT\rangle$ is an anatomization of table T according to Definition 2.3.*

Example 2. Given query $\pi_{City, Disease}(\mathrm{Patient_{QIT}}, \mathrm{Patient_{SNT}})$; $S = \{1\}$ since neither $\pi_{City}(\mathrm{Patient_{QIT}})$ nor $\pi_{Disease}(\mathrm{Patient_{SNT}})$ have only one element when $GID = 1$. However all other groups (i.e., $\{2, 3, 4\}$) can be projected without the knowledge of actual link between $\mathrm{Patient_{QIT}}$, $\mathrm{Patient_{SNT}}$. Intermediate tables are shown in Figure 3.

City	*Disease*
Lafayette	Cough
Lafayette	Flu
Richmond	Fever
Richmond	Flu

(a) R'

City	*SEQ*
Dayton	1
Richmond	2

(b) T'_{QIT}

City	*SEQ*
$\mathrm{H}_{\bar{k}_2}(1)$	Cold
$\mathrm{H}_{\bar{k}_2}(2)$	Fever

(c) T'_{SNT}

City	*Disease*
Dayton	Cold
Lafayette	Cough
Lafayette	Flu
Richmond	Fever
Richmond	Flu

(d) R''

Fig. 3. Intermediate tables in Example 2

3.3 Join

Join is problematic, as it can be an expensive operation. We detail below a natural join. The key is to push join as late as possible, as it only results in reduction on the sub-tables containing the join criterion (e.g., the QIT sub-tables); the other sub-tables can only be reduced to the extent that the join eliminates complete anatomization groups.

Definition 3.6 (Server-side Join Query). *Given $Z_1 = QIT_1$, $Z_2 = QIT_2$ and their corresponding sensitive attribute tables, $Z_3 = SNT_1$ and $Z_4 = SNT_2$, derived from table T_1 and T_2 respectively using anatomization, join query written as $(QIT_1, SNT_1) \bowtie (QIT_2, SNT_2)$ returns three tables*

$$\langle R_1, R_2, R_3 \rangle = \langle Z_i \bowtie Z_j, Z_k, Z_l \rangle$$

where $\exists a : a \in A_{Z_i} \cap A_{Z_j}$ and $1 \leq i \neq j \neq k \neq l \leq 4$.

Definition 3.7 (Client-side Join Query). *Given $\langle R_1, R_2, R_3 \rangle$ tables computed by the server; for every R_i having attribute SEQ_j the client updates each tuple of R_i by replacing the value of SEQ_j attribute with its corresponding keyed hash value (i.e., $s_j \rightarrow \mathrm{H}_{\bar{k}_j}(s_j)$) where $1 \leq i \leq 3$ and $1 \leq j \leq 2$. Then the final join query would be computed as*

$$R = R_1 \bowtie R_2 \bowtie R_3$$

Theorem 3.3. *Given QIT_1, QIT_2, SNT_1 and SNT_2; R derived according to Definition 3.6 and 3.7 is equal to $T_1 \bowtie T_2$ if the pairs, $\langle QIT_1, SNT_1 \rangle$ and $\langle QIT_2, SNT_2 \rangle$, are anatomizations of table T_1 and T_2 respectively.*

3.4 Group-By

Group-by is challenging, as it is also an expensive operation, but can in some cases be done largely at the server. This is dependent on the type of aggregate being computed. In some cases, an anatomization group may be contained entirely in a group-by group; if so, an aggregate such as MAX need only return a single value for that anatomization group. However, if the values in an anatomization group are split across multiple group-by groups, all tuples must be returned, as the server has no way of knowing which tuple goes in which group.

We now show how to apply this optimization (when all tuples in an anatomization group are in the same group-by group) for several classes of aggregates.

Definition 3.8 (Aggregate Function Set). *Given a three sets of attributes, X_{QIT}, X_{SNT} and X^*, defined as a subset of A_{QIT}, A_{SNT}, and $\{*\}$ respectively; a group-by aggregate function set, F, consists of individual functions (e.g., COUNT, AVG) each defined on one of the attributes of X.*

$$F(X) = \{f_1(x_1), f_2(x_2), \ldots, f_k(x_k)\} \quad \text{where } X = X_{QIT} \cup X_{SNT} \cup X^*$$

Definition 3.9 (Auxiliary Function Set). *Given an aggregate function set F defined as in Definition 3.8 along with its input set X, an auxiliary function set F' is defined such that*

- *if $AVG(x_i) \in F(X)$ then also $COUNT(x_i) \in F(X) \cup F'(X)$*
- *if $S(x_i) \in F(X)$ then both $COUNT(x_i)$ and $AVG(x_i)$ are also in $F(X) \cup F'(X)$ where S could be STDEV, VAR, STDEVP, or VARP.*

Definition 3.10 ($\dot{\bowtie}$ operator). *Given two tables QIT and SNT derived from T by anonymization based on Anatomy model, $QIT \dot{\bowtie} SNT$ merges the two tables vertically such that each tuple of QIT in each group is joined with only one of the tuples in the same group of SNT without taking SEQ and $HSEQ$ into account.*

Definition 3.11 (Server-side Group-By Query). *Given QIT and SNT tables derived from a table T using Anatomy model anonymization, a set of attributes for the grouping, A', and a set of aggregate functions defined as in Definition 3.8; a group-by query is written as*

$$_{A'}\gamma_{F(X)}(QIT, SNT)$$

where $A' = A'_{QIT} \cup A'_{SNT}$. The above group-by query returns a set of tables R based on the grouping attributes A',

$$R = \begin{cases} \{R', T'_{QIT}, T'_{SNT}\} & \text{if } A'_{QIT} \neq \emptyset \text{ and } A'_{SNT} \neq \emptyset \\ {}_{A'}\gamma_{F(X)}\,(QIT) & \text{if } A'_{QIT} \neq \emptyset \text{ and } A'_{SNT} = \emptyset \text{ and } X_{SNT} = \emptyset \\ {}_{A'}\gamma_{F(X)}\,(SNT) & \text{if } A'_{QIT} = \emptyset \text{ and } A'_{SNT} \neq \emptyset \text{ and } X_{QIT} = \emptyset \end{cases}$$

where tables R', T'_{QIT}, and T'_{SNT}; defined as

$$\begin{aligned} R' &= {}_{A'}\gamma_{F(X),F'(X)}\left(\sigma_{GID \in S}\left(R'_{QIT} \dot{\bowtie} R'_{SNT}\right)\right) \\ T'_{QIT} &= \pi_{A'_{QIT}, SEQ, X_{QIT}}\left(\sigma_{GID \notin S} QIT\right) \\ T'_{SNT} &= \pi_{A'_{SNT}, HSEQ, X_{SNT}}\left(\sigma_{GID \notin S} SNT\right) \end{aligned}$$

where F' is as in Definition 3.9 and R'_{QIT}, R'_{SNT}, and S; are defined as

$$R'_{QIT} = \pi^d_{A'_{QIT}, GID, X_{QIT}}\left(QIT\right)$$
$$R'_{SNT} = \pi^d_{A'_{SNT}, GID, X_{SNT}}\left(SNT\right)$$
$$S = \left\{i : |\sigma_{GID=i}(\pi_{A'_{QIT}, GID} R'_{QIT})| = 1 \wedge |\sigma_{GID=i}(\pi_{A'_{SNT}, GID} R'_{SNT})| = 1\right\}$$
$$\cup \left\{i : \left|\sigma_{GID=i}\left(distinct(R'_{QIT})\right)\right| = 1 \vee |\sigma_{GID=i}\left(distinct(R'_{SNT})\right)| = 1\right\}$$

Lemma 3.3. *Given R' and S defined in Definition 3.11,*

$$R' = {}_{A'}\gamma_{F(X),F'(X)}\left(\sigma_{GID \in S}\left(QIT_{updated} \bowtie SNT\right)\right)$$

Definition 3.12 ($\dot\cup$ operator). *Given two disjoint tables T_1 and T_2 having identical schemas, $\dot\cup$ operator merges two group-by query results:*

$$g'(T_1) \dot\cup g'(T_2) = g(T_1 \cup T_2)$$

where $g' : T \mapsto {}_{A'}\gamma_{F(X),F'(X)} T$ and $g : T \mapsto {}_{A'}\gamma_{F(X)} T$ for some set of attributes, A', and a set of aggregate functions defined as in Definition 3.8 and 3.9.

Remark 2. There are three types of aggregate functions:

1. Functions having the property $f(f(X), f(Y)) = f(X, Y)$ where X and Y are single valued datasets. Hence the results of such functions can be combined to get a single result for multiple datasets (e.g., MAX, MIN, SUM, COUNT).
2. Functions not having the above property since they require every single value in the dataset to evaluate the result (e.g., CHECKSUM, MEDIAN). The whole dataset should be given as an input to this type of functions.
3. Functions not having the above property unless there is some auxiliary information given about the dataset. For instance, the results of average function of multiple dataset cannot be combined unless the count of values in each dataset is also given. Similarly standard deviation or variation results can be combined when both average and count of values in each dataset is given.

The $\dot\cup$ operator in Definition 3.12 is general such that it covers both the first and third type of functions. If the aggregate functions are only of the first type, there is no need to include an auxiliary function set, $F'(X)$, in the formulation.

In Algorithm 1, we present an algorithm that calculates the $\dot\cup$ operator.

Definition 3.13 (Client-side Group-By Query). *Given the set of tables, R, computed by the server; if $|R| = 1$ then the client outputs the only table in R without any processing. Otherwise, the client updates each tuple of T'_{QIT} by replacing the values of SEQ attribute with their corresponding keyed hash value (i.e., $s \rightarrow \mathrm{H}_{\bar{k}}(s)$). Then the final result of group-by query is computed by using special $\dot\cup$ operator in Definition 3.12,*

$$R'' = R' \dot\cup \left({}_{A'}\gamma_{F(X),F'(X)}\left(T'_{QIT_{updated}} \bowtie T'_{SNT}\right)\right)$$

Algorithm 1: The algorithm for $\dot{\cup}$ operation

input : Tables, $T_1' = {}_{A'}\gamma_{F(X),F'(X)}T_1$ and $T_2' = {}_{A'}\gamma_{F(X),F'(X)}T_2$
output: Table $T = T_1' \dot{\cup} T_2'$
sort T_1' and T_2' on attribute list A' if they are not sorted already;
$b_1 \leftarrow 1$; $b_2 \leftarrow 1$;
while T_1' *and* T_2' *has more tuples* **do**
  if $b_1 = 1$ **then** $t_1 \leftarrow$ the next tuple in T_1';
  if $b_2 = 1$ **then** $t_2 \leftarrow$ the next tuple in T_2';
  if $t_1.A' < t_2.A'$ **then** write t_1 into table T, $b_1 \leftarrow 1$, $b_2 \leftarrow 0$;
  else if $t_2.A' < t_1.A'$ **then** write t_2 into table T, $b_2 \leftarrow 1$, $b_1 \leftarrow 0$;
  else
    $t.A' \leftarrow t_1.A'$;
    foreach $f_i(x_i) \in F(X)$ **do**
      if $f_i(\cdot)$ *is type 1* **then** $t.f_i(x_i) \leftarrow f_i(t_1.f_i(x_i), t_2.f_i(x_i))$;
      else if $f_i(\cdot)$ *is type 2* **then** $t.f_i(x_i) \leftarrow$ undefined;
      else if $f_i(\cdot)$ *is type 3* **then**
        $\exists \mathrm{COUNT}(x_i) \in F(X) \cup F'(X)$;
        $AVG_{x_i} \leftarrow \frac{\sum_{j=1}^{2} t_j.\mathrm{AVG}(x_i) \times t_j.\mathrm{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\mathrm{COUNT}(x_i)}$;
        if $f_i(\cdot) = AVG$ **then** $t.f_i(x_i) \leftarrow AVG_{x_i}$;
        else if $f_i(\cdot) = STDEV$ **then**
          $\exists \mathrm{AVG}(x_i) \in F(X) \cup F'(X)$;
          $t.f_i(x_i) \leftarrow \sqrt{\frac{\sum_{j=1}^{2}(t_j.\mathrm{AVG}(x_i)^2 + t_j.\mathrm{STDEV}(x_i)^2) \times t_j.\mathrm{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\mathrm{COUNT}(x_i)} - AVG_{x_i}^2}$;
        else if $f_i(\cdot) = VAR$ **then**
          $\exists \mathrm{AVG}(x_i) \in F(X) \cup F'(X)$;
          $t.f_i(x_i) \leftarrow \frac{\sum_{j=1}^{2}\left(t_j.\mathrm{AVG}(x_i)^2 + t_j.\mathrm{VAR}(x_i)\right) \times t_j.\mathrm{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\mathrm{COUNT}(x_i)} - AVG_{x_i}^2$;
    write t into table T; $b_1 \leftarrow 1$; $b_2 \leftarrow 1$;
write all remaining tuples of T_1' and T_2' into T;

Theorem 3.4. *Given QIT, SNT, a set of attributes, A', for grouping and aggregate functions f_1 through f_k along with their inputs x_1 through x_k; R'' derived according to Definition 3.11, 3.12 and 3.13 is equal to ${}_{A'}\gamma_{F(X)}T$ if the pair $\langle QIT, SNT \rangle$ is an anonymization of table T based on Anatomy model.*

Example 3. According to Definition 3.11 and 3.13, given query

$$_{Gender,City}\gamma_{AVG(AGE)}(\mathrm{Physician}_{\mathrm{QIT}}, \mathrm{Physician}_{\mathrm{SNT}} \bowtie \mathrm{Patient}_{\mathrm{QIT}})$$

all groups in $S = \{1, 2, 3\}$ can be projected without knowing the link between $\mathrm{Physician}_{\mathrm{QIT}}$ and $\mathrm{Physician}_{\mathrm{SNT}}$. Intermediate tables are shown in Figure 4.

Gender	*City*	*AVG(AGE)*	*COUNT(*)*
Female	Dayton	41	1
Female	Richmond	31	3
Male	Lafayette	32.5	2

(a) R'

Gender	*SEQ*
Male	7
Female	8

(b) T'_{QIT}

HSEQ	*Age*	*City*
$H_{\bar{k}_1}(7)$	45	Lafayette
$H_{\bar{k}_1}(8)$	30	Lafayette

(c) T'_{SNT}

Gender	*City*	*AVG(AGE)*
Female	Dayton	41
Female	Lafayette	30
Female	Richmond	31
Male	Lafayette	32

(d) R''

Fig. 4. Intermediate tables in Example 3

4 Conclusions and Further Work

We have shown how given an anatomization of a database that meets privacy constraints, we can store that database at an untrusted (semi-honest) server and perform queries that minimize the load on the client. This frees the server from constraints imposed by privacy law, allowing it to provide a service while avoiding concerns over privacy.

There has been extensive work on storing and processing *encrypted* data. Our approach is to minimize the encryption, while still satisfying privacy constraints. This provides not only significant performance advantages, but also allows the server to provide "value-added" services. Such services could include address correction and normalization (cleaning individual data) as well as data analysis. Such services provide a more compelling business case for private data outsourcing than an "encrypt everything" approach, while still ensuring that outsourcing does not pose a privacy risk.

As a future work, we will implement the presented query operators and evaluate the performance of our system in a real anatomized database. Moreover, this paper looks only at a fixed database and read-only queries. Insert, update, and delete pose additional challenges, and are also left as future work. Another challenge that arises is data modeling: given a database and privacy constraints, what is the appropriate normalization for an anatomized database?

Acknowledgments. This publication was made possible by the support of an NPRP grant from the QNRF. The statements made herein are solely the responsibility of the author. Partial support for this work was provided by MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

References

[1] Aggarwal, G., Bawa, M., Ganesan, P., Garcia-molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed

architecture for secure database services. In: Proc. CIDR (2005)
[2] Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. pp. 1–15. CRYPTO '96, Springer-Verlag, London, UK (1996)
[3] Ciriani, V., De Capitani Di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: outsourcing data while maintaining confidentiality. pp. 440–455. ESORICS'09, Springer-Verlag, Berlin, Heidelberg (2009)
[4] Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational dbmss. pp. 93–102. ACM CCS '03, Washington D.C., USA (2003)
[5] DasGupta, A.: The birthday and matching problems. In: Fundamentals of Probability: A First Course, pp. 23–28. Springer Texts in Statistics, Springer NY (2010)
[6] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal of the European Communities No I.(281), 31–50 (Oct 24 1995)
[7] Family educational rights and privacy act of 1974. Congressional Record 120, 39862–39866 (1974), http://www2.ed.gov/policy/gen/guid/fpco/ferpa/
[8] Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proc.s of the 14th ACM symposium on Theory of computing. pp. 365–377. STOC '82, ACM, New York, NY, USA (1982)
[9] Hacıgümüş, H., Iyer, B.R., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data. pp. 216–227. Madison, Wisconsin (2002)
[10] Standard for privacy of individually identifiable health information. Federal Register 67(157), 53181–53273 (Aug 14 2002), http://www.hhs.gov/ocr/privacy/hipaa/administrative/privacyrule/index.html
[11] Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proc. of the 30th Int. Conf. on Very large data bases - Volume 30. pp. 720–731. VLDB '04, VLDB Endowment (2004)
[12] Jiang, X., Gao, J., Wang, T., Yang, D.: Multiple sensitive association protection in the outsourced database. In: Database Systems for Advanced Applications, Lecture Notes in Computer Science, vol. 5982, pp. 123–137. Springer (2010)
[13] Li, N., Li, T.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: Proc. of the 23rd Int. Conf. on Data Engineering (ICDE '07). Istanbul, Turkey (2007)
[14] Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: *l*-diversity: Privacy beyond *k*-anonymity. ACM Trans. on Know. Discovery from Data (TKDD) (1) (2007)
[15] Nergiz, A.E., Clifton, C.: Query processing in private data outsourcing using anonymization. Tech. Rep. 11-004, CS, Purdue University, http://www.cs.purdue.edu/research/technical_reports/2011/TR%2011-004.pdf
[16] Nergiz, M.E., Clifton, C., Nergiz, A.E.: Multirelational k-anonymity. IEEE Transactions on Knowledge and Data Engineering 21(8), 1104–1117 (Aug 2009)
[17] Øhrn, A., Ohno-Machado, L.: Using boolean reasoning to anonymize databases. Artificial Intelligence in Medicine 15(3), 235–254 (Mar 1999)
[18] Samarati, P.: Protecting respondent's privacy in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (Nov/Dec 2001)
[19] Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (5), 557–570 (2002)
[20] Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proc. of 32nd Int. Conf. on Very Large Data Bases (VLDB). Seoul, Korea (2006)