2011

Query Processing in Private Data Outsourcing Using Anonymization

Ahmet Erhan Nergiz
Purdue University, anergiz@cs.purdue.edu

Chris Clifton
Purdue University, clifton@cs.purdue.edu

Report Number:
11-004

Query Processing in Private Data Outsourcing Using Anonymization

Ahmet Erhan Nergiz and Chris Clifton

Purdue University, West Lafayette, IN 47907
`anergiz, clifton @cs.purdue.edu`

Abstract. We present a scheme for query processing in a private data outsourcing model. We assume that data is divided into identifying and sensitive data using an anatomy approach[24]; 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 that 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"[8], 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*[13]. 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 [10]); such debate is not in the scope of this paper. We propose a data model based on *anatomization*[24]. 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 3.) 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[21, 23], discernibility/l-diversity[20, 18], and t-closeness[17]. While the original anatomization paper just considered a single table, extending this to a full relational database has been explored[19].

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. We show how standard relational algebra operations can be performed to lower client-side cost using issue 1 (Section 3) and analysis the cost of each of these standard operations (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.

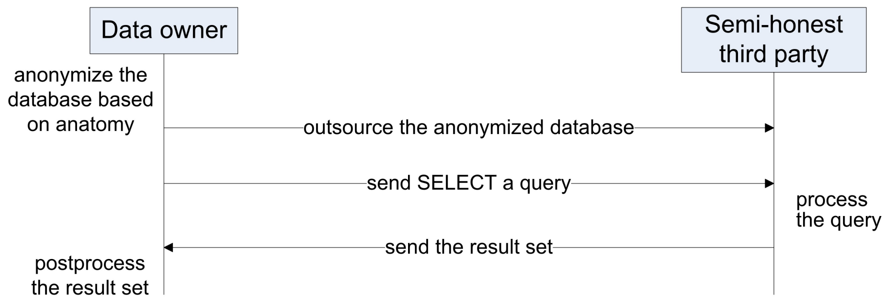
Data owner
Semi-honest third party
anonymize the database based on anatomy
outsource the anonymized database
send SELECT a query
process the query
send the result set
postprocess the result set

Fig. 1. Protocol Model

Figure 1 shows the outline of our private outsourcing technique. Although the outline of our technique is similar to the ones in the secure outsourcing literature, the privacy of our technique reduces to already existent anonymization techniques thus achieving accepted privacy metrics throughout the anonymization literature.

1.2 Related Work

Private data outsourcing also known as database-as-a-service model was first introduced by Hacigumus et al. [12]. 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 and the privacy of individuals in the database while choosing 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 [14], no privacy measurement showing the amount of information leakage On the other hand, Damiani et al. [6] purposed another technique that uses hashing for bucketization and encrypted B+ trees for indexing. They also 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.

Another approach is using searchable encryption for range queries [22, 2]. Since the client needs to decrypt each query result and also group-by queries are not supported, encryption is far from being practical for large databases. Kantarcioglu et al.[16] shows that an efficient private data outsourcing scheme based on encryption cannot be proven to meet cryptographic-style definitions. Instead they give the outline of an efficient private data outsourcing scheme using encryption with provable security, though they require tamper-resistance hardware on the server side.

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. However we believe that assuming two unaware servers from each other is too strong in the real world. Ciriani et al. [5] proposed to vertically fragment a table and store the rest of the attribute values encrypted into each partition. This enables the client to have queries involving only the unencrypted values in a fragment to be executed without decryption. However in this method the size of the database is m times the original size where m is the number of fragments and in the case of more complicated queries involving both encrypted and plaintext attributes, the client needs to decrypt the query result to refine it. Another approach described in [4] 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 [15]. They give an l-diverse partitioning scheme based on anatomization[24] for a single table having multiple sensitive attributes. Our work is orthogonal to their work in the sense 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[24] to meet privacy constraints. Making this work for multiple tables does demand extra thought; a solution for this is given in [19]. This paper assumes an anatomized database meeting privacy constraints; we now present relevant definitions and notations (based on [19]) 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 [24].

Definition 2.3 (Anatomy). *Given an l-diverse table, T, with m equivalence classes, 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 $1 \leq i \leq d$, $|Q_T| = d$, 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 $\{A_{d+1}\} = S_T$, GID is the group id of the equivalence class and HSEQ contains the outputs of $\mathrm{H}_k(s)$ 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:

$$(\mathrm{H}_k(s), j, v)$$

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*	*Address*	*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. 2. Original Database

For instance, the anatomy anonymization of person specific tables *Physician* and *Patient* in Figure 2 is shown Figure 3.

Note that we show a (keyed) hash as the "join key" between the two subtables. We use HMAC [3] 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 [11] 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 $\mathrm{H} : k \times D \rightarrow D'$ is said to hide the join link, SEQ, once each value v in T_2.SEQ is updated with $\mathrm{H}_k(v)$ if*

- *Without knowing the secret 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 v_1 and v_2 such that $\mathrm{H}_k(v_1) = \mathrm{H}_k(v_1)$*

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 k is known whereas when encryption is used, one needs to decrypt each $\mathrm{H}_k(v)$ 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) $\mathrm{Physician}_{\mathrm{QIT}}$

HSEQ	*GID*	*Patient*
$\mathrm{H}_{k_1}(1)$	1	Ike
$\mathrm{H}_{k_1}(2)$	1	Eric
$\mathrm{H}_{k_1}(3)$	2	Olga
$\mathrm{H}_{k_1}(4)$	2	Kelly
$\mathrm{H}_{k_1}(5)$	3	Faye
$\mathrm{H}_{k_1}(6)$	3	Mike
$\mathrm{H}_{k_1}(7)$	4	Jason
$\mathrm{H}_{k_1}(8)$	4	Max

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

Patient	*Age*	*Address*	*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	Dayton	4	8

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

HSEQ	*GID*	*Disease*
$\mathrm{H}_{k_2}(1)$	1	Cold
$\mathrm{H}_{k_2}(2)$	1	Fever
$\mathrm{H}_{k_2}(3)$	2	Flu
$\mathrm{H}_{k_2}(4)$	2	Cough
$\mathrm{H}_{k_2}(5)$	3	Flu
$\mathrm{H}_{k_2}(6)$	3	Fever
$\mathrm{H}_{k_2}(7)$	4	Cough
$\mathrm{H}_{k_2}(8)$	4	Flu

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

Fig. 3. 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 $\mathrm{H}_k(v)$ 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, P, of not having the same $\mathrm{H}_k(v)$ value for any v in any equivalence class in $QIT_{updated}$ or SNT can be approximated by using the birthday problem analysis [7].

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

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, $P \approx 1$.

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 that* $\mathrm{H}(.)$ *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* $(\mathrm{H}_k(s'), j, v)$ *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 k is the unknown key for the cryptographic hash function, H(.).

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 k used with the function 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 3, $\langle Ike, 41, Dayton \rangle$, 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 given that the server knows the transcript of the join query, 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,

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

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 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.

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:*

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

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

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

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'':

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

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*

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

Proof. Rewriting the selection query on the right hand side above into multiple separate selection queries reflects the intuition behind the steps of Definition 3.2.

$$\begin{aligned}
\sigma_P(QIT \bowtie SNT) &= \sigma_{P_{QS}}\left(\sigma_{P_{QIT}}\left(\sigma_{P_{SNT}}(QIT \bowtie SNT)\right)\right) && (3.1)\\
&= \sigma_{P_{QS}}\left(\sigma_{P_{QIT}} QIT \bowtie \sigma_{P_{SNT}} SNT\right) && (3.2)\\
&= \sigma_{P_{QS}}(QIT' \bowtie SNT') && (3.3)\\
&= \sigma_{P_{\beta+1}}\left(\sigma_{P_{\beta+2}}\left(\ldots\left(\sigma_{P_n}(QIT' \bowtie SNT')\right)\right)\right) && (3.4)\\
&= \bigcap_{i=\beta+1}^{n} \left(\sigma_{P_{\beta+1}}(QIT' \bowtie SNT')\right) && (3.5)\\
&= \sigma_{P_{QS}}(QIT' \bowtie SNT') \ltimes \bigcap_{i=\beta+1}^{n} \left(\sigma_{P_{\beta+1}}(QIT' \bowtie SNT')\right) && (3.6)\\
&= \sigma_{P_{QS}}(QIT' \bowtie SNT') \bowtie \bigcap_{i=\beta+1}^{n} (S_{P_{\beta+1}}) && (3.7)\\
&= \sigma_{P_{QS}}(QIT' \bowtie SNT') \bowtie S_{GID}\\
&= \sigma_{P_{QS}}((QIT' \bowtie S_{GID}) \bowtie (SNT' \bowtie S_{GID})) && (3.8)\\
&= \sigma_{P_{QS}}(QIT'' \bowtie SNT'') && (3.9)
\end{aligned}$$

Equation 3.1 is due to the cascading of selections. Equation 3.2 is derived by commuting selections with the join operation since these selections can be only applied to QIT or SNT. As in Equation 3.3, QIT' and SNT' can be substituted for the inner two selections since join of these two tables is the same as the join of these selections. Equation 3.4 shows the cascading of P_{QS} predicates and intersecting individual selection operations yields the same result as in shown in equation 3.5. Combining equation 3.3 with equation 3.5 by a semi-join does not change the validity of the equality. The selection inside the union in the second term of the right hand side of equation 3.6 is substituted in equation 3.7 by a new term which is proven to be equal to S_{GID} below. S_{GID} can be distributed into both tables QIT' and SNT' with a join as in equation 3.8. By Definition 3.2, equation 3.9 is concluded.

$$S_{P_i} = \pi_{GID}(\sigma_{P_i}(QIT' \bowtie SNT')) \qquad (3.10)$$

$$= \pi_{GID}(\sigma_{P_i^{QIT} \vee P_i^{SNT}}(QIT' \bowtie SNT')) \tag{3.11}$$
$$= \pi_{GID}((\sigma_{P_i^{QIT}}(QIT' \bowtie SNT')) \cup (\sigma_{P_i^{SNT}}(QIT' \bowtie SNT'))) \tag{3.12}$$
$$= \pi_{GID}((\sigma_{P_i^{QIT}}(QIT') \bowtie SNT') \cup (QIT' \bowtie \sigma_{P_i^{SNT}}(SNT'))) \tag{3.13}$$
$$= (\pi_{GID}(\sigma_{P_i^{QIT}}(QIT')) \bowtie \pi_{GID}SNT') \cup (\pi_{GID}QIT' \bowtie \pi_{GID}(\sigma_{P_i^{SNT}}(SNT'))) \tag{3.14}$$
$$= \pi_{GID}(\sigma_{P_i^{QIT}}(QIT')) \cup \pi_{GID}(\sigma_{P_i^{SNT}}(SNT')) \tag{3.15}$$

Above equations show that S_{P_i} defined as in equation 3.10 corresponds to a term in the intersection of S_{GID} in Definition 3.2. Equation 3.11 is due to the terms P_i in P_{QS} in Definition 3.1. Predicates having OR can be separated by applying union to the result of the two selection operations as in Equation 3.12. Equation 3.13 commutes each selection with their join operation. Since GID is the only join attribute, one can commute the projection further with the join operations as in equation 3.14. $\pi_{GID}SNT'$ and $\pi_{GID}QIT'$ can be omitted as in equation 3.15 since they contain superset of group-ids with respect to the selection operations due to the fact that QIT' and SNT' have the same set of GIDs.

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. $s \to \mathrm{H}_{k_d}(s)$). 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.*

Proof. Deriving $\sigma_{P_{QS}}(QIT''_{updated} \bowtie SNT'')$ from $\sigma_P(T)$ assuming the unique separation of T into tables QIT and SNT in Definition 2.3 proves $\sigma_P(T) = \sigma_{P_{QS}}(QIT''_{updated} \bowtie SNT'')$

$$\sigma_P(T) = \sigma_P(QIT_{updated} \bowtie SNT) \tag{3.16}$$
$$= \sigma_{P_{QS}}(QIT''_{updated} \bowtie SNT'') \tag{3.17}$$

(3.16) can be obtained due to the Definition 2.3. The join operation can be applied to the unique 1-1 mapping attributes SEQ and HSEQ once SEQ is hashed to uncover the link in HSEQ. Derivation of (3.17) from (3.16) is proved in Lemma 3.1.

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

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

$P_{QIT} = (Age > 40)$, $P_{SNT} = (Disease = Flu \vee Disease = Cough)$, and $P_{QS} = (Disease = Fever \vee Age < 3)$. QIT' has the 6th and 7th tuples of table $\mathrm{Patient_{QIT}}$ based on P_{QIT}. 1th tuple is not included since the corresponding group of $\mathrm{Patient_{SNT}}$ doesn't satisfy the P_{SNT} (which is ensured with semi-join). SNT' has 5th, 7th and 8th tuples of table $\mathrm{Patient_{SNT}}$. And $S_{GID} = \{4\}$ since none of the tuples in group 3 satisfies the predicate P_{QS}. Then QIT'' has 7th tuple of $\mathrm{Patient_{QIT}}$ and SNT'' has 7th and 8th tuples of $\mathrm{Patient_{SNT}}$. After server sends these intermediate results to the client, client updates the SEQ field of QIT'' and computes $R = \langle Jason, 45, Lafayette, 4, \mathrm{H}_{k_2}(7), \mathrm{H}_{k_2}(7), 4, Cough\rangle$. Intermediate tables are shown in Figure 4.

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.

Patient	*Age*	*Address*	*GID*	*SEQ*
Mike	47	Richmond	3	6
Jason	45	Lafayette	4	7

(a) Patient$_{\text{QIT'}}$

HSEQ	*GID*	*Disease*
H$_{k_2}$ (5)	3	Flu
H$_{k_2}$ (7)	4	Cough
H$_{k_2}$ (8)	4	Flu

(b) Patient$_{\text{SNT'}}$

Patient	*Age*	*Address*	*GID*	*SEQ*
Jason	45	Lafayette	4	7

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

HSEQ	*GID*	*Disease*
H$_{k_2}$ (7)	4	Cough
H$_{k_2}$ (8)	4	Flu

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

Fig. 4. Intermediate tables in Example 1

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:

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

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

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

and R'_{QIT}, R'_{SNT}, and S; defined as

$$\begin{aligned}
R'_{QIT} &= \pi_{A'_{QIT},GID}\left(QIT\right) \\
R'_{SNT} &= \pi_{A'_{SNT},GID}\left(SNT\right) \\
S &= \left\{i : \left|\sigma_{GID=i}\left(R'_{QIT}\right)\right| > 1 \wedge \left|\sigma_{GID=i}\left(R'_{SNT}\right)\right| > 1\right\}
\end{aligned}$$

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

$$R' = \pi_{A'}\left(\sigma_{GID\notin S}\left(R'_{QIT} \bowtie R'_{SNT}\right)\right) = \pi_{A'}\left(\sigma_{GID\notin S}\left(QIT_{updated} \bowtie SNT\right)\right) \tag{3.18}$$

Proof. Let i be any group id such that $i \notin S$ and $T^i = \sigma_{GID=i}T$. We can rewrite the above equation so that each selection is pushed to the lowest level of the query evaluation tree. Handling each single group separately then applying union over each single operation yields to

$$\begin{aligned}
R' &= \pi_{A'}\left(\bigcup_{i\notin S}\pi_{A'_{QIT},GID}\left(QIT^i\right) \bowtie \pi_{A'_{SNT},GID}\left(SNT^i\right)\right) \\
&= \pi_{A'}\left(\bigcup_{i\notin S} QIT^i_{updated} \bowtie SNT^i\right)
\end{aligned}$$

which can be proven by showing that for each $i \notin S$

$$\pi_{A'}\left(\pi_{A'_{QIT},GID}\left(QIT^i\right) \bowtie \pi_{A'_{SNT},GID}\left(SNT^i\right)\right) = \pi_{A'}\left(QIT^i_{updated} \bowtie SNT^i\right)$$

Based on the anonymization by Anatomy model, for any group i there is a $1/k$ possibility that a tuple in QIT^i matches with a tuple in SNT^i and vice versa where $k = |QIT^i| = |SNT^i|$. Similarly, for any group i there is a $\frac{1}{\left|\pi_{A'_{SNT}}SNT^i\right|}$ possibility that a tuple in $\pi_{A'_{QIT}}QIT^i$ matches with a tuple in $\pi_{A'_{SNT}}SNT^i$ and vice versa. Therefore; if the cardinalities of at least one side is 1, the matching probability becomes 1. Since any group id $i \notin S$ represents the groups that have a matching probability of 1, the join of the projected QIT^i and SNT^i is the same as joining and projecting QIT^i and SNT^i after revealing the encrypted link between them. Therefore the above equation holds which implies the Equation 3.18.

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. $s \rightarrow \mathrm{H}_{k_d}(s)$). 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.*

Proof. Simply substituting variables in the formulation of R'' according to their definitions shows the equality in Theorem 3.2,

$$\begin{aligned}
R'' &= \pi_{A'}\left(R' \cup \pi_{A'}\left(T'_{QIT_{updated}} \bowtie T'_{SNT}\right)\right) \\
&= \pi_{A'}\left(\pi_{A'}\left(\sigma_{GID\notin S}\left(R'_{QIT} \bowtie R'_{SNT}\right)\right) \cup \pi_{A'}\left(T'_{QIT_{updated}} \bowtie T'_{SNT}\right)\right) \qquad (3.19) \\
&= \pi_{A'}\left(\pi_{A'}\left(\sigma_{GID\notin S}\left(QIT_{updated} \bowtie SNT\right)\right) \cup \pi_{A'}\left(T'_{QIT_{updated}} \bowtie T'_{SNT}\right)\right) \qquad (3.20) \\
&= \pi_{A'}\left(\sigma_{GID\notin S}\left(QIT_{updated} \bowtie SNT\right) \cup \sigma_{GID\in S}\left(QIT_{updated} \bowtie SNT\right)\right) = \pi_{A'}\left(T\right) \qquad (3.21)
\end{aligned}$$

We get Equation 3.19 when we substitute R' with its formulation. Equation 3.20 is derived from Equation 3.18 in Lemma 3.2. Equation 3.21 gives emphasis on selection in T'_{QIT} and T'_{SNT}. Since the two selections are disjoint and compliments of each other, the union of the two gives original table T.

Example 2. Given query $\pi_{Address,Disease}(\text{Patient}_{\text{QIT}}, \text{Patient}_{\text{SNT}})$; $S = \{1\}$ then all other groups (i.e. $\{2, 3, 4\}$) can be projected without the knowledge of actual link between $\text{Patient}_{\text{QIT}}$, $\text{Patient}_{\text{SNT}}$. Intermediate tables are shown in Figure 5.

Address	*Disease*
Lafayette	Cough
Lafayette	Flu
Richmond	Fever
Richmond	Flu

(a) R'

Address	*SEQ*
Dayton	1
Richmond	2

(b) T'_{QIT}

Address	*SEQ*
H_{k_2} (1)	Cold
H_{k_2} (2)	Fever

(c) T'_{SNT}

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

(d) R''

Fig. 5. 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 based on which tables have the join criteria*

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

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. $s_j \rightarrow \text{H}_{k_{d_j}}(s_j)$) 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.*

Proof. When each possible formulation of $\langle R_1, R_2, R_3 \rangle$ in Definition 3.6 joined together as in Definition 3.7 is the same as joining all sub-tables altogether (i.e. $(QIT_1 \bowtie SNT_1) \bowtie (QIT_2 \bowtie SNT_2)$) after *SEQ* fields are updated. The only difference between the possible formulations of $\langle R_1, R_2, R_3 \rangle$ is the order of the join operation.

Example 3. According to Definition 3.6 and 3.7, given query

$$(\text{Physician}_{\text{QIT}}, \text{Physician}_{\text{SNT}}) \bowtie (\text{Patient}_{\text{QIT}}, \text{Patient}_{\text{SNT}})$$

results in $R_1 = \text{Physician}_{\text{SNT}} \bowtie \text{Physician}_{\text{SNT}}$, $R_2 = \text{Physician}_{\text{QIT}}$, and $R_3 = \text{Patient}_{\text{SNT}}$. After updating the *SEQ* attributes, $R_1 \bowtie R_2 \bowtie R_3$ is the same as

$$\text{Physician}_{\text{QIT}} \bowtie (\text{Physician}_{\text{SNT}} \bowtie \text{Patient}_{\text{QIT}}) \bowtie \text{Patient}_{\text{SNT}}$$

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.*

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

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.*

Algorithm 1 gives a pseudo-code for evaluating $\dot{\bowtie}$ operator based on a single equivalence class.

Algorithm 1: Calculation of $\dot{\bowtie}$ Operator for an equivalence class

```
input : Two sets of tuples, $T_1 = \pi_{A'_{QIT}}(\sigma_{GID=j} QIT)$ and $T_2 = \pi_{A'_{SNT}}(\sigma_{GID=j} SNT)$ for some $j$, $A'_{QIT}$, and
        $A'_{SNT}$
output: $T = T_1 \dot{\bowtie} T_2$
for $i = 1$ to $|T_1|$ do
    $t_1 \leftarrow$ the next tuple in $T_1$;
    $t_2 \leftarrow$ the next tuple in $T_2$;
    write $t = t_1 | t_2$ into $T$ ;        /* Bar | between $t_1$ and $t_2$ represents concatenation */
end
```

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',

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

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

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

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

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

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

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

Proof. There are three types of groups (i.e. equivalence class) that the set S can have according to its formulation in Definition 3.11

1. The number of distinct tuples after all grouping attributes, A'_{QIT} and A'_{SNT}, are projected is 1 for both $T_1 = \sigma_{GID=i}QIT$ and $T_2 = \sigma_{GID=i}SNT$ tables respectively where i is the identifier for the group. In this case any group-by query, $Q = {}_{A'}\gamma_{F(X)}\left(T_1 \bowtie T_2\right)$, has only one tuple in its result. This implies that there is no need to know the actual link between T_1 and T_2 to calculate Q. The tuples of T_1 and T_2 can be randomly 1:1 mapped.
2. The number of distinct tuples after the grouping attributes, A'_{QIT}, and the attributes, X_{QIT}, are projected is 1 for $T_1 = \sigma_{GID=i}QIT$ table where i is the identifier for the group. In this case, each tuple in $T_2 = \sigma_{GID=i}SNT$ has to match with the single tuple resulted from the projection of T_1. Therefore there is again no need to know the actual link between T_1 and T_2 to calculate Q.
3. Same as 2, only this time $\left|\pi_{A'_{SNT},X_{SNT}}\left(\sigma_{GID=i}SNT\right)\right| = 1$.

Since there is no need to know the actual link between QIT and SNT to calculate the group-by query Q for these three types of groups explained above, the server can use $\dot{\bowtie}$ operator to calculate Q.

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 2, we present an algorithm that calculates the $\dot{\cup}$ operator.

Algorithm 2: Calculation of $\dot{\cup}$ operator

```
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
                $\exists \text{COUNT}(x_i) \in F(X) \cup F'(X)$;
                $AVG_{x_i} \leftarrow \frac{\sum_{j=1}^{2} t_j.\text{AVG}(x_i) \times t_j.\text{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\text{COUNT}(x_i)}$;
                if $f_i(\cdot) = AVG$ then $t.f_i(x_i) \leftarrow AVG_{x_i}$;
                else if $f_i(\cdot) = STDEV$ then
                    $\exists \text{AVG}(x_i) \in F(X) \cup F'(X)$;
                    $t.f_i(x_i) \leftarrow \sqrt{\frac{\sum_{j=1}^{2} (t_j.\text{AVG}(x_i)^2 + t_j.\text{STDEV}(x_i)^2) \times t_j.\text{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\text{COUNT}(x_i)} - AVG_{x_i}^2}$;
                else if $f_i(\cdot) = VAR$ then
                    $\exists \text{AVG}(x_i) \in F(X) \cup F'(X)$;
                    $t.f_i(x_i) \leftarrow \frac{\sum_{j=1}^{2} \big(t_j.\text{AVG}(x_i)^2 + t_j.\text{VAR}(x_i)\big) \times t_j.\text{COUNT}(x_i)}{\sum_{j=1}^{2} t_j.\text{COUNT}(x_i)} - AVG_{x_i}^2$;
                end
            end
        end
        write $t$ into table $T$;
    end
end
write all remaining tuples of $T'_1$ and $T'_2$ into $T$;
```

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. $s \to \mathrm{H}_k(s)$). Then the final result of group-by query is computed by using special $\dot\cup$ operator in Definition 3.12,*

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

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.*

Proof. Simply substituting variables in the formulation of R'' according to their definitions shows the equality in Theorem 3.4,

$$\begin{aligned}
R'' &= R' \dot\cup \left({}_{A'}\gamma_{F(X),F'(X)} \left(T'_{QIT_{updated}} \bowtie T'_{SNT} \right) \right) \\
&= {}_{A'}\gamma_{F(X),F'(X)} \left(\sigma_{GID \in S} \left(R'_{QIT} \dot\bowtie R'_{SNT} \right) \right) \\
&\quad \dot\cup \left({}_{A'}\gamma_{F(X),F'(X)} \left(T'_{QIT_{updated}} \bowtie T'_{SNT} \right) \right) & (3.22) \\
&= {}_{A'}\gamma_{F(X),F'(X)} \left(\sigma_{GID \in S} \left(R'_{QIT_{updated}} \bowtie R'_{SNT} \right) \right) \\
&\quad \dot\cup \left({}_{A'}\gamma_{F(X),F'(X)} \left(T'_{QIT_{update}} \bowtie T'_{SNT} \right) \right) & (3.23) \\
&= {}_{A'}\gamma_{F(X)} \left(\sigma_{GID \in S} \left(R'_{QIT_{updated}} \bowtie R'_{SNT} \right) \cup \left(T'_{QIT_{update}} \bowtie T'_{SNT} \right) \right) & (3.24) \\
&= {}_{A'}\gamma_{F(X)} \left(QIT_{updated} \bowtie SNT \right) = {}_{A'}\gamma_{F(X)}T & (3.25)
\end{aligned}$$

We get equation 3.22 when we substitute R' in the definition of R'' and by using Lemma 3.3 we substitute the first term (i.e. R''s substitute) with the updated R'_{QIT} table joining with R'_{SNT} in equation 3.23. Due to the property of $\dot\cup$ operator, we can rewrite equation 3.23 as in equation 3.24. Since the union operation in equation 3.24 is disjoint where the first term has all tuples with $GID \in S$ and the second term has all tuples with $GID \notin S$, we can combine them as the join of $QIT_{updated}$ and SNT which results in ${}_{A'}\gamma_{F(X)}T$.

Example 4. According to Definition 3.11 and 3.13, given query

$${}_{Gender,Address}\gamma_{AVG(AGE)}(\mathrm{Physician_{QIT}}, \mathrm{Physician_{SNT}} \bowtie \mathrm{Patient_{QIT}})$$

all groups in $S = \{1, 2, 3\}$ can be projected without knowing the link between $\mathrm{Physician_{QIT}}$ and $\mathrm{Physician_{SNT}}$. Intermediate tables are shown in Figure 6.

Gender	*Address*	*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*	*Address*
H_{k_1} (7)	45	Lafayette
H_{k_1} (8)	30	Lafayette

(c) T'_{SNT}

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

(d) R''

Fig. 6. Intermediate tables in Example 4

4 Query Analysis

We now give the cost analysis of each query operator given in Section 3. First we give algorithms for the query operators since relational algebra cannot express certain improvements on the evaluation of the query operators due to special properties of anatomization. Then we analyse the cost of each query operator based on its algorithm assuming reading or writing each tuple results in one I/O operation. Although this assumption is too strong especially for the write operations, cost analysis becomes simpler and easier to follow and since we follow the same assumption for all query operators, comparison of query operators stays unaffected. Table 1 gives the notations used in our cost analysis.

Table 1. Query Cost Notations

n	number of tuples in table T
m	number of groups (i.e. equivalence classes) in table QIT and SNT
c	average number of tuples in a single group in table QIT and SNT
rf_P	reduction factor for a predicate P in CNF form in table T
grf_P	reduction factor for a predicate P in CNF form among the groups (i.e. equivalence classes) of T
rf_{π_A}	reduction factor for a duplicate eliminating π_A operation
$\mathcal{P}_{A'}$	the probability that $\pi_{A'}(E_{QIT}, E_{SNT})$ can be calculated by the server where E is a single group.
$\mathcal{P}_{\gamma}$	the probability that ${}_{A'}\gamma_{F(X)}(E_{QIT}, E_{SNT})$ can be calculated by the server where E is a single group.
B	the number of disk pages the server/client can hold in memory.

Algorithm 3: Server-side selection in absence of index

```
input  : $QIT$, $SNT$ tables and predicate $P$ in CNF form
output: $QIT''$ $SNT''$ tables
sort $QIT$ and $SNT$ tables on attribute $GID$ if they are not sorted already;
for $g = 1$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $QIT$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $SNT$;
    $T_1' \leftarrow \sigma_{P_{QIT}} T_1$;
    $T_2' \leftarrow \sigma_{P_{SNT}} T_2$;
    if $T_1' \neq \emptyset$ and $T_2' \neq \emptyset$ then
        $select\_success \leftarrow 1$;
        for $i = \beta + 1$ to $n$ do
            if $\sigma_{P_i^{QIT}} T_1' = \emptyset$ and $\sigma_{P_i^{SNT}} T_2' = \emptyset$ then
                $select\_success \leftarrow 0$; break;
            end
        end
        if $select\_success$ then
            write tuples in $T_1'$ and $T_2'$ into $QIT''$ and $SNT''$ respectively;
        end
    end
end
```

4.1 Selection

We analyse selection without assuming any index on the attributes that the predicate P checks. The only assumption made is that QIT and SNT tables are sorted on attribute GID. This assumption is not far fetched since equivalence classes are sorted at the end of any anonymization algorithm and the server can store the equivalence classes in sorted order after receiving it for the first time from the client.

Server-side Considering the relational algebra operations for server-side selection in Definition 3.2, checking P_{QIT}, P_{SNT} and P_{QS} against the tuples of QIT and SNT is not necessarily done separately since tuples in the same group are checked for all three predicates. Server-side selection can be done as in Algorithm 3 in a single pass of retrieving tuples of both QIT and SNT. The number of I/O operations done is $2n$ assuming that the server has enough memory to hold T_1 and T_2 (i.e. tuples from a group of QIT and SNT respectively) and there is no need to write resulting tuples into disk since they can be sent to the client as produced. The number of tuples sent from the server to the client is approximately $N_s = 2n \times rf_{P_{QIT}} \times rf_{P_{SNT}} \times grf_{P_{QIT}} \times grf_{P_{SNT}} \times grf_{P_{QS}}$.

Client-side Client-side selection is given in Algorithm 4. Upon getting the results of the selection, the client joins the tuples from the same group, checks the resulting tuples against predicate P_{QS} and prints the tuples if the check is successful. Since T_1 can only join with the tuples from the same group in SNT, there is no need to materialize the results coming from the server. Instead the tuples can be processed on the fly as they come assuming once the client gets the first T_1 and T_2, it has enough memory to hold $m \times (t_1 - t_2) \times N_s/(m \times t_2) = N_s \times (t_1/t_2 - 1)$ tuples where t_1 is the average time to process any T_1 and T_2 pair, t_2 is the average time to receive any T_1 and T_2 pair from the server and $N_s/(m \times t_2)$ denote the number of tuples received per second. Therefore there is no I/O cost in the client side and the number of tuples at the end of the selection is $n \times rf_P$

Algorithm 4: Client-side selection in absence of index

```
input  : $QIT''$, $SNT''$ tables and predicate $P_{QS}$ in CNF form
output: table $R$
for $g = min(GID)$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $QIT''$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $SNT''$;
    foreach $t_1 \in T_1$ do
        $t_1.SEQ \leftarrow \mathrm{H}_k(t_1.SEQ)$;
        foreach $t_2 \in T_2$ do
            if $t_1.SEQ = t_2.HSEQ$ and $t \leftarrow t_1|t_2$ satisfies $P_{QS}$ then
                write $t$ into $R$;
            end
        end
    end
end
```

4.2 Projection

We give two different projection algorithm for both server and client. The first one has an improved communication complexity with respect to the second algorithm however the client has less computation and I/O overhead in the second algorithm than it has in the first algorithm. First algorithm is the naive approach directly based on relational algebra operations in Definition 3.4 and 3.5 whereas second algorithm, while obeying the given relational algebra operations most of the time, outsources most of the client's workload to the server.

Server-side version 1 Algorithm 5 shows how to evaluate the tables given in Definition 3.4. Since T_1 and T_2 is in memory, duplicate elimination in calculating T_1' and T_2' is done in memory. Therefore the number of tuples read is $2n$ which requires $2n$ I/O's and in memory sorting takes $O(mclogc) = O(nlogc)$ time which is dominated by I/O operations. If the tuples are sent to the client while they are produced, there is no need of write operation hence we do not include the I/O cost for write operations. However we need to write R' into disk since the duplicates need to be removed. The cost of writing R' is $mc\mathcal{P}_{A'}$ and the cost of external sort is $2mc\mathcal{P}_{A'}(\lceil\log_{B-1} mc\mathcal{P}_{A'}\rceil + 1)$. The number of tuples sent to client is approximately $rf_{\pi_{A'}} mc\mathcal{P}_{A'} + 2mc(1 - \mathcal{P}_{A'})$ assuming most of the time R' has c many tuples for a single group instead of having less than c.

Algorithm 5: Server-side projection version 1

```
input  : $QIT$, $SNT$ tables and projection attributes $A' = A'_{QIT} \cup A'_{SNT}$
output: $R'$, $T'_{QIT}$, and $T'_{SNT}$ tables
sort $QIT$ and $SNT$ tables on attribute $GID$ if they are not sorted already;
for $g = 1$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $QIT$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $SNT$;
    $T'_1 \leftarrow \pi_{A'_{QIT}} T_1$;
    $T'_2 \leftarrow \pi_{A'_{SNT}} T_2$;
    if $|T'_1| = 1$ or $|T'_2| = 1$ then
        write $T'_1 \bowtie T'_2$ into $R'$;
    else
        write $\pi_{A'_{QIT},GID,SEQ} T_1$ and $\pi_{A'_{SNT},GID,HSEQ} T_2$ into $T'_{QIT}$, and $T'_{SNT}$ respectively;
    end
end
eliminate duplicates in $R'$;
```

Algorithm 6: Client-side projection version 1

```
input  : $R'$, $T'_{QIT}$, and $T'_{SNT}$ tables and projection attributes $A'$
output: table $R$
for $g = min(GID)$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $T'_{QIT}$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $T'_{SNT}$;
    foreach $t_1 \in T_1$ do
        $t_1.SEQ \leftarrow \mathrm{H}_k(t_1.SEQ)$;
        foreach $t_2 \in T_2$ do
            if $t_1.SEQ = t_2.HSEQ$ then write $\pi_{A'} t$ into $T'$ ;
        end
    end
end
sort $T'$ on $A'$;
$b_1 \leftarrow 1$; $b_2 \leftarrow 1$;
while $T'$ and $R'$ has more tuples do
    if $b_1 = 1$ then $t_1 \leftarrow$ the next tuple in $R'$;
    if $b_2 = 1$ then $t_2 \leftarrow$ the next tuple different from $t_2$ in $T'$;
    if $t_1 < t_2$ then
        write $t_1$ into table $R$; $b_1 \leftarrow 1$; $b_2 \leftarrow 0$;
    else if $t_2 < t_1$ then
        write $t_2$ into table $R$; $b_2 \leftarrow 1$; $b_1 \leftarrow 0$;
    else
        write $t_2$ into table $R$; $b_1 \leftarrow 1$; $b_2 \leftarrow 1$;
    end
end
write all remaining tuples of $R'$ and distinct tuples of $T'$ into $R$;
```

Client-side version 1 Algorithm 6 gives the client-side steps of the first projection version. The for loop joins T'_{QIT} and T'_{SNT} tables into table T'. This operation does not require any I/O operation however table T' needs to be materialized so as R' since T' needs to be sorted on attributes A' after the join. The write operation requires $N_{R'+T'} = rf_{\pi_{A'}} mc\mathcal{P}_{A'} + mc(1-\mathcal{P}_{A'})$ number of I/O's. The number of tuples in each T'_{QIT} and T'_{SNT} is $N_{T'} = mc(1 - \mathcal{P}_{A'})$ as given in previous subsection. Assuming merge sort is used for external sorting of T', $2N_{T'}\lceil(\log_{B-1}(N_{T'})\rceil + 1)$ number of I/O's need to be done. In the while block, R' and T' is read once to get the combined result of the projection which is done in $N_{R'+T'}$ I/O's. The total number of I/O's required for the whole client-side projection is $2N_{R'+T'} + 2N_{T'}\lceil(\log_{B-1}(N_{T'})\rceil + 1)$.

Server-side version 2 The server-side of projection version 2 is given in Algorithm 7. In the for loop, QIT and SNT tables are read once and approximately $mc\mathcal{P}_{A'}$ number of tuples is written into $\bar{R}$ if the projection can be done in the server. Otherwise, all possible matchings are written into $\bar{R}$. For each group, such a join operation produces c^2 tuples and since there are $m(1 - \mathcal{P}_{A'})$ such groups, the total number of tuples written into $\bar{R}$ is $N_{\bar{R}} = mc\mathcal{P}_{A'} + mc^2(1 - \mathcal{P}_{A'}) = nc - n(c-1)\mathcal{P}_{A'}$. External sorting of $\bar{R}$ requires $2N_{\bar{R}}(\lceil\log_{B-1}(N_{\bar{R}})\rceil + 1)$ I/O's. Then $\bar{R}$ is read once more to eliminate all tuples identical to the projected tuples in the server-side. Let $\mathcal{P}_{\bar{R}}$ denote the probability of eliminating a tuple in the last while block of Algorithm 7 then the number of tuples in $\bar{R}$ is reduced to $(1 - \mathcal{P}_{\bar{R}})N_{R'}$. Losslessly decomposing $\bar{R}$ into R' and T' results in $rf_{\pi_A}(1 - \mathcal{P}_{\bar{R}})N_{R'}$ number of tuples in R' and $(1 - \mathcal{P}_{\bar{R}})N_{R'}$ number of tuples in T'. All the tuples in R' and T' are sent to the client.

Algorithm 7: Server-side projection version 2

```
input  : $QIT$, $SNT$ tables and projection attributes $A' = A'_{QIT} \cup A'_{SNT}$
output: $R'$, $T'$ tables
sort $QIT$ and $SNT$ tables on attribute $GID$ if they are not sorted already;
for $g = 1$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $QIT$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $SNT$;
    $T'_1 \leftarrow \pi_{A'_{QIT}} T_1$;
    $T'_2 \leftarrow \pi_{A'_{SNT}} T_2$;
    if $|T'_1| = 1$ or $|T'_2| = 1$ then
        write $(T'_1 \bowtie T'_2) \times \langle -1, -1 \rangle$ into $\bar{R}$;
    else
        write $\pi^d_{A',SEQ,HSEQ}(T_1 \bowtie T_2)$ into $\bar{R}$;
    end
end
sort $\bar{R}$ on $A', SEQ$; $t' \leftarrow$ undefined;
while $\bar{R}$ has more tuples do
    $t \leftarrow$ the next tuple in $\bar{R}$;
    if $t.SEQ = -1$ and $t.HSEQ = -1$ then $t' \leftarrow \pi_{A'} t$;
    else if $t'$ is defined and $t' = \pi_{A'} t$ then
        delete $t$ from $\bar{R}$;
    else
        $t' \leftarrow$ undefined;
    end
end
Losslessly decompose $\bar{R}$ into $R'$ with schema $(A'_1, \ldots, A'_k, id)$ and $T'$ with schema $(id, SEQ, HSEQ)$;
```

Client-side version 2 Algorithm 8 outlines the client-side of the second version projection. If the size of a tuple in R' is much bigger than the size of a tuple in T', then the client can process T' as it receives the tuples and simultaneously checks whether tuples not deleted from T' semi-joins with R'. If not, only R'

needs to be written into disk and the while loop can be done on the fly as the tuples of T' received. Then the client reads R' once for semi-join assuming the id's of the qualified tuples of T' can be stored in memory.

Algorithm 8: Client-side projection version 2

```
input  : $R'$ and $T'$ tables
output: table $R$
while $T'$ has more tuples do
    $t \leftarrow$ the next tuple in $R'$;
    if $t.SEQ \neq -1$ and $t.HSEQ \neq -1$ and $\mathrm{H}_k(t.SEQ) \neq t.HSEQ$ then
        delete $t$ from $T'$;
    end
end
$R = \pi_{A'}(R' \ltimes T')$
```

4.3 Group-by

We give only one algorithm for group-by queries. Since projection eliminating duplicates is similar to the group-by, the reader can easily see how to introduce a second version of the group-by algorithm just like in the projection operator case. However it should be noted that communication would be worse than the projection since all possible inputs for aggregate functions should be sent to the client. Now we give the direct approach based on the Definition 3.11 and 3.13.

Server-side Group-by Algorithm 9 presents the server-side of the group-by operator. QIT and SNT tables are read once which leads to $2n$ I/O's. Intermediate table R' needs to be written to disk since at the end of the algorithm, group-by query is executed based on it. However tuples of T'_{QIT} and T'_{SNT} can be sent to the client as they produced. The cost of writing R' to the disk is $n\mathcal{P}_\gamma$ number of I/O's in the worst case. The result of group-by query at the end requires external sorting of R' on attributes A' then the approximate cost of group-by is $2n\mathcal{P}_\gamma(\lceil \log_{B-1}(n\mathcal{P}_\gamma)\rceil + 1)$ number of I/O's and the results can be sent to the client as they are produced.

Algorithm 9: Server-side group-by

```
input  : $QIT$, $SNT$ tables, grouping attributes $A' = A'_{QIT} \cup A'_{SNT}$ and aggregate function set $F(X)$
output: $R'$, $T'_{QIT}$, and $T'_{SNT}$ tables
sort $QIT$ and $SNT$ tables on attribute $GID$ if they are not sorted already;
for $g = 1$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $QIT$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $SNT$;
    $T'_1 \leftarrow \pi_{A'_{QIT}, X_{QIT}} T_1$;
    $T'_2 \leftarrow \pi_{A'_{SNT}, X_{SNT}} T_2$;
    if $(|\pi_{A'_{QIT}} T_1| = 1$ and $|\pi_{A'_{SNT}} T_2| = 1)$ or $(|T'_1| = 1$ or $|T'_2| = 1)$ then
        write $\left(\pi^d_{A'_{QIT}, X_{QIT}} T_1 \dot{\bowtie} \pi^d_{A'_{SNT}, X_{SNT}} T_2\right)$ into $R'$;
    else
        write $\pi_{A'_{QIT}, X_{QIT}, GID, SEQ} T_1$ and $\pi_{A'_{SNT}, X_{SNT}, GID, HSEQ} T_2$ into $T'_{QIT}$, and $T'_{SNT}$ respectively;
    end
end
$R' \leftarrow {}_{A'}\gamma_{F(X), F'(X)} R'$;
```

Client-side Group-by Algorithm 10 outlines the client-side of the group-by operator. T'_{QIT} and T'_{SNT} tables are read once in the for loop which has a cost of $2n(1 - \mathcal{P}_\gamma)$ I/O's. T', the join result of T'_{QIT} and

T'_{SNT} , needs to be written to disk since group-by query is executed on it right after the for loop. Writing T' into disk results in $n(1-\mathcal{P}_\gamma)$ number of I/O's. Executing the group-by query on T' requires external sorting in which $2n(1-\mathcal{P}_\gamma)(\lceil \log_{B-1}(n(1-\mathcal{P}_\gamma))\rceil + 1)$ number of I/O's are made. For the evaluation of $\dot{\cup}$ operator, each table is read once so the number of I/O's made for evaluating $\dot{\cup}$ operator is $|R'| + |T'|$.

Algorithm 10: Client-side group-by

```
input  : $R'$, $T'_{QIT}$, and $T'_{SNT}$ tables, grouping attributes $A'$ and aggregate function set $F(X)$
output: table $R$
for $g = min(GID)$ to $max(GID)$ do
    $T_1 \leftarrow$ tuples with $GID = g$ in $T'_{QIT}$;
    $T_2 \leftarrow$ tuples with $GID = g$ in $T'_{SNT}$;
    foreach $t_1 \in T_1$ do
        $t_1.SEQ \leftarrow \mathrm{H}_k(t_1.SEQ)$;
        foreach $t_2 \in T_2$ do
            if $t_1.SEQ = t_2.HSEQ$ then write $\pi_{A',X} t$ into $T'$ ;
        end
    end
end
$T' \leftarrow {}_{A'}\gamma_{F(X),F'(X)} T'$;
$R \leftarrow R' \dot{\cup} T'$ ;                                /* calculated with Algorithm 2 */
```

5 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 (e.g., medical product safety monitoring such as the FDA's Sentinel initiative [9].) 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.

This paper looks only at a fixed database and read-only queries. Insert, update, and delete pose additional challenges, and are 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?

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: In Proc. CIDR (2005)

[2] Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Proceedings of the 27th annual international cryptology conference on Advances in cryptology. pp. 535–552. CRYPTO'07, Springer-Verlag, Berlin, Heidelberg (2007), `http://portal.acm.org/citation.cfm?id=1777777.1777820`

[3] Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. pp. 1–15. CRYPTO '96, Springer-Verlag, London, UK (1996), `http://portal.acm.org/citation.cfm?id=646761.706031`

[4] Ciriani, V., De Capitani Di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: outsourcing data while maintaining confidentiality. In: Proceedings of the 14th European conference on Research

in computer security. pp. 440–455. ESORICS'09, Springer-Verlag, Berlin, Heidelberg (2009), http://portal.acm.org/citation.cfm?id=1813084.1813120

[5] Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Biskup, J., Lpez, J. (eds.) Computer Security ESORICS 2007, Lecture Notes in Computer Science, vol. 4734, pp. 171–186. Springer Berlin / Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-74835-9_12

[6] Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational dbmss. In: Proceedings of the 10th ACM conference on Computer and communications security. pp. 93–102. ACM Press, Washington D.C., USA (2003)

[7] DasGupta, A.: The birthday and matching problems. In: Fundamentals of Probability: A First Course, pp. 23–28. Springer Texts in Statistics, Springer New York (2010), http://dx.doi.org/10.1007/978-1-4419-5780-1_2

[8] 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), http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm

[9] FDA's sentinel initiative (2011), http://www.fda.gov/Safety/FDAsSentinelInitiative/

[10] Family educational rights and privacy act of 1974. Congressional Record 120, 39862–39866 (Nov 19 1974), http://www2.ed.gov/policy/gen/guid/fpco/ferpa/

[11] Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp. 365–377. STOC '82, ACM, New York, NY, USA (1982), http://doi.acm.org/10.1145/800070.802212

[12] Hacıgümüş, H., Iyer, B.R., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. pp. 216–227. Madison, Wisconsin (Jun 4-6 2002), http://doi.acm.org/10.1145/564691.564717

[13] 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

[14] Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. pp. 720–731. VLDB '04, VLDB Endowment (2004), http://portal.acm.org/citation.cfm?id=1316689.1316752

[15] Jiang, X., Gao, J., Wang, T., Yang, D.: Multiple sensitive association protection in the outsourced database. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) Database Systems for Advanced Applications, Lecture Notes in Computer Science, vol. 5982, pp. 123–137. Springer Berlin / Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-12098-5_10

[16] Kantarcioglu, M., Clifton, C.: Security issues in querying encrypted data. In: Jajodia, S., Wijesekera, D. (eds.) Data and Applications Security XIX, Lecture Notes in Computer Science, vol. 3654, pp. 325–337. Springer Berlin / Heidelberg (2005), http://dx.doi.org/10.1007/11535706_24

[17] Li, N., Li, T.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE '07). Istanbul, Turkey (Apr 16-20 2007), http://dx.doi.org/10.1109/ICDE.2007.367856

[18] Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) (1) (Mar 2007), http://doi.acm.org/10.1145/1217299.1217302

[19] Nergiz, M.E., Clifton, C., Nergiz, A.E.: Multirelational k-anonymity. IEEE Transactions on Knowledge and Data Engineering 21(8), 1104–1117 (Aug 2009), http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.210

[20] Øhrn, A., Ohno-Machado, L.: Using boolean reasoning to anonymize databases. Artificial Intelligence in Medicine 15(3), 235–254 (Mar 1999), http://dx.doi.org/10.1016/S0933-3657(98)00056-6

[21] Samarati, P.: Protecting respondent's privacy in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (Nov/Dec 2001), http://dx.doi.org/10.1109/69.971193

[22] Shi, E., Bethencourt, J., Chan, T.H.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy. pp. 350–364. SP '07, IEEE Computer Society, Washington, DC, USA (2007), http://dx.doi.org/10.1109/SP.2007.29

[23] Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (5), 557–570 (2002), http://dx.doi.org/10.1142/S0218488502001648

[24] Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proceedings of 32nd International Conference on Very Large Data Bases (VLDB 2006). Seoul, Korea (Sep 12-15 2006), http://www.vldb.org/conf/2006/p139-xiao.pdf