PrivTru: A Privacy-by-Design Data Trustee Minimizing Information Leakage

Lukas Gehring[1][0009−0006−7876−861X] and
Florian Tschorsch[1][0000−0001−6716−7225]

Technische Universität Dresden, 01062 Dresden, Germany
{lukas.gehring,florian.tschorsch}@tu-dresden.de

Abstract. Data trustees serve as intermediaries that facilitate secure data sharing between independent parties. This paper offers a technical perspective on data trustees, guided by privacy-by-design principles. We introduce PrivTru, an instantiation of a data trustee that provably achieves optimal privacy properties. Therefore, PrivTru calculates the minimal amount of information the data trustee needs to request from data sources to respond to a given query. Our analysis shows that PrivTru minimizes information leakage to the data trustee, regardless of the trustee's prior knowledge, while preserving the utility of the data.

Keywords: Data Trustee · Privacy by design · Privacy Engineering

1 Introduction

The concept of a data trustee (or sometimes data trust) facilitates a data economy that aligns with privacy goals and data protection regulations, particularly in the EU [12,3,2]. It addresses challenges in sharing data between independent data sources and data receivers. This is particularly relevant in fields such as medicine, public administration, mobility, and other domains where data can drive innovation and create value. Politically, data trustees are presented as an alternative to the dominance of data platforms, which risk creating a "data oligopoly" that centralizes power and limits competition [12].

Most discussions have focused on the organizational aspects [3], situating data trustees as a third party between data sources (entities who *have* data) and data receivers (entities who *want* data). A core idea of the concept is that sources and receivers are independent systems, operated by any entity, enabling data sharing (and potentially linking) across institutional boundaries. To fulfill this role, a data trustee needs to be constituted as an independent organization, which has no interest in the data itself. As such it can reasonably balance the interests of the involved parties. The primary task of a data trustee is, thus, to manage data "on behalf of and in the interest of" [12] the data sources.

The literature [2,12] categorizes trustees into two types: *data stewards* and *data exchange*. In the former, the trustee stores the data, whereas in the latter, the trustee facilitates matching between sources and receivers and relays the data without storing it.

In this paper, we present a technical perspective on data trustees, focusing specifically on data stewards and data exchanges. This work addresses a gap in the discussion on data trustees, which has largely been explored from organizational, political, and legal perspectives. By applying Hoepman's privacy design strategies [9], we identify the challenges of implementing privacy-respecting data trustees and demonstrate that the *data exchange* model is preferable from a privacy-centric standpoint.

We then introduce *PrivTru*, a data exchange designed to be private by design. To this end, we extend relational algebra, enabling the formulation of individual subqueries for generalized data from the data sources. By leveraging this approach, PrivTru outsources pre-processing tasks to the data sources, ensuring that only the necessary information is retrieved to answer a given query. We further prove that our instantiation is optimal in minimizing the information the data trustee learns about the provided data. The proposed solution achieves this without any loss of utility and remains optimal regardless of the prior knowledge the central exchange may have about the input data. Accordingly, our contributions can be summarized as follows:

- We formulate a technical perspective on data stewards and data exchanges, the two primary types of data trustees (Section 2).
- We use Hoepman's privacy design strategies [9] to analyze the privacy properties of the proposed models (Section 3).
- We introduce *PrivTru*, an instantiation of a data exchange aligned with privacy engineering principles (Section 4).
- We analyze the information gain of the central data exchange in *PrivTru* and prove that it is minimal compared to all other relational data exchange implementations (Section 5).

2 Technical Perspective on Data Trustees

In this section, we provide a technical perspective on the notion of a *data trustee* by introducing two models capturing its main variations: *data steward* and *data exchange*. To encompass most concepts covered by this notion, the models focus on the core functionality of a data trustee, which is providing access to data from *data sources* to *data receivers*. Apparently, given the broad understanding of a data trustee, there may be some instantiations of data trustees, stewards, and exchanges that are not compatible with our model. We additionally consider the question of bootstrapping the connection between sources, receivers, and the trustees, notably the implementation of access control, to be out of scope. Nevertheless, the models are designed to be extensible, allowing for adaptation to specific organizational and legal requirements of data trustees or to provide better guidance for practical implementation.

In the data trustee model, the third party sitting between the sources and the receivers is a *steward* or an *exchange*. These are differentiated by their mandate to store the processed data, which leads to distinct data flows when in operation. A *steward* stores data on behalf of the sources, while an *exchange*

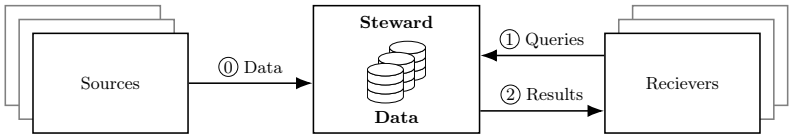

Fig. 1. Illustration of a data steward.

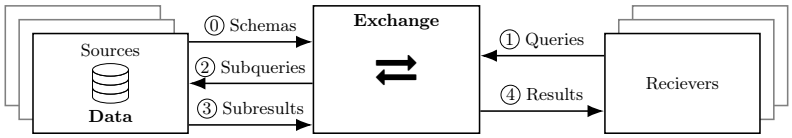

Fig. 2. Illustration of a data exchange.

merely processes the information from the sources, without storing any data. Figure 1 illustrates the architecture and functionality of a data steward. In the setup phase, all sources send *all* of their data to the steward (Step ⓪), which then stores it centrally. Subsequently, the steward provides one or more API endpoints for the receivers to query the data (Step ①). It processes these queries by sending results back to the corresponding receiver (Step ②).

The architecture of a data exchange is depicted in Figure 2. During the bootstrapping process (Step ⓪), the central entity only receives the schema of the data provided by the sources. Exchanges provide the same interface as stewards, enabling receivers to query information and obtain results (Steps ① and ④). However, unlike stewards, exchanges do not store any data. Instead, upon receiving a query (Step ①), they generate subqueries for each data source and assembles the subresults to answer the full query (Steps ② and ③).

3 Privacy Analysis of Data Trustees

A system that prioritizes privacy should be designed with respect to *privacy-by-design* strategies. In this section, we apply these design strategies to our two data trustees models and argue that *data exchanges* are preferable to *data stewards* under these principles.

Privacy Design Strategies To bridge the gap between privacy requirements for IT systems driven by ethical and legal requirements and software development, Hoepman proposed eight privacy design strategies [9]. According to this perspective, a system's design plays a significant role in shaping its level of privacy. Therefore, privacy considerations should be taken into account from the very beginning, when the foundational concepts and ideas for a new system are first being developed, which includes our models of data trustees. His contributions are categorized into data-oriented strategies and process-oriented strategies. Since we focus on the technical side of data trustees, our analysis is

limited to the former. Of course, process-oriented strategies must also be considered when deploying a data trustee. The data-oriented privacy strategies as formulated by Hoepman [9] are as follows:

- Minimize: The amount of personal data that is processed should be limited to the minimal amount possible.
- Hide: Any personal data, and their relationships, should be concealed from plain view.
- Separate: Personal data should be processed in a distributed manner, using separate compartments whenever possible.
- Aggregate: Personal data should be processed at the highest level of aggregation and with the least detail necessary for it to remain useful.

Comparing Data Stewards and Data Exchanges In the following analysis, we compare the data trustee models (as outlined in Section 2) based on the data-oriented privacy strategies. Hereby we focus on the differences between the two models and always assume the same premises when possible.

Minimize: Data trustees generally allocate information from multiple data sources to be jointly provided to one or more data receivers. This is a problem for data minimization because when two or more personal data points concerning one individual are joined, the generated information is in most cases more privacy-sensitive than the two data points alone since it increases the risk of *re-identification attacks* [13]. Still, one can argue that the data is sufficiently minimized if two requirements are fulfilled. Firstly, the data sources need to make sure, that the data they provide is already minimized. Secondly, the purpose of the whole system is formulated broadly. So that it encompasses all use cases where any data might be requested. If the sources give their consent to the possible wide range of data usage, such usage is not a privacy violation. If a data trustee satisfies these conditions, we can consider the system to be in line with Minimize. In the data exchange model, data sources can minimize queries locally, thereby enhancing privacy when compared to data stewards.

Hide: For data trustees, it is essential to ensure that data remains confidential. Encrypting connections secures the content during transfer. However, when data requests involve only a subset of sources, it may also be necessary to obscure traffic patterns—especially when a single data source corresponds to an individual or a small group, a challenge that solely affects data exchanges. In such cases, obfuscation techniques can mitigate risks. This leaves data exchanges with a slight but manageable disadvantage compared to data stewards.

Separate: Organizational data trustees are formulated as centralized systems, which opposes the Separate design strategy. As discussed in Section 2, a centralized organizational concept does not necessarily demand a centralized technical model. The data exchange model is fundamentally embracing the Separate strategy, while the data steward model is breaking it. This is especially true if a data exchange is instantiated by a distributed protocol between the data source and receivers, and not through a centralized service. Regarding Separate, data exchanges are to be favored over data stewards.

AGGREGATE: Data aggregation can occur at the data sources or by the data steward/exchange. For certain techniques, such as k-anonymity [13], access to all data points is critical for a reasonable privacy-utility tradeoff. Thus, aggregation strategies must be carefully tailored to each use case. Data exchanges are generally preferred due to their flexibility in determining aggregation levels per query, whereas the data steward model requires defining these levels during the bootstrapping phase.

Summary of Comparison Summarizing the analysis, we observe that data exchanges are mostly preferable from a privacy perspective. This is largely due to the SEPARATE strategy, which data exchanges fully implement and data stewards compromise by default. For HIDE, both models are comparable, although data exchanges are more prone to information leakage via side channels. MINIMIZE & AGGREGATE can be achieved by both models; however, the data exchange model conceptually offers more flexibility, as the data sources can minimize or aggregate the data on a per-query basis.

4 PrivTru: Minimizing Information Leakage

In this section, we introduce *PrivTru*, a data trustee designed with *privacy-by-design* principles. Our analysis revealed that designing PrivTru as a data exchange is preferable for aligning with privacy principles. Yet, implementing MINIMIZE and AGGREGATE poses challenges. In our work, we focus on MINIMIZE and argue that AGGREGATE can be addressed similarly, as discussed in Section 7. We specifically examine how queries to PrivTru are structured and how subqueries are derived.

Data and Query Format For PrivTru, we adopt the commonly used relational database model, where data is stored in tables T, also referred to as *relations*. Each table consists of multiple columns, identified by a set of *attributes* $\mathbb{A}(T) = \{A_1, \dots A_n\}$, with $n > 0$. Each attribute A_i is associated with a *domain* $\mathbb{D}(A_i)$, which defines the set of permitted values in this column. Accordingly, the datapoints t in T are the rows of the table. Formally, they are tuples of length n with $t_i \in \mathbb{D}(A_i)$ for $i = 1, \dots n$. In the system model of data exchanges, there are multiple data sources $S_1, \dots S_s$, each providing a portion of the data. For readability, we assume that each source S_i contributes exactly one table T_i to the system. In addition, the attributes across all tables are assumed to be pairwise distinct, meaning that every attribute in the system is present in exactly one table. Note, that data trustees in general and data exchanges specifically are not tied to a specific data model.

Using relational databases as our data model naturally leads to relational algebra as the foundation for our query model. Specifically, we adopt the relational algebra as introduced by Codd [7].

Definition 1 (Relational Algebra [1]). *The* relational algebra *consists of three primitive operators, working on tables (or relations) T.*

(a) Selection*: $\sigma_F(T)$ selects tuples in T that fulfill a given propositional formula F. The formula F consists of the logical operators $\wedge, \vee, \neg$, connecting literals of the form* `true`*,* `false`*, $A\theta B$, and $A\theta c$ with $A, B \in \mathbb{A}(T), c \in \mathbb{D}(A)$, and $\theta \in \{=, <\}$.*

(b) Projection*: $\pi_{(A_1,\dots A_n)}(T)$ outputs a new relation containing the columns identified by $A_1, \dots A_n$ in T.*

(c) Join*: $(T \bowtie S)$ outputs a relation O with $\mathbb{A}(O) = \mathbb{A}(T) \cup \mathbb{A}(S)$. Where every o over $\mathbb{A}(O)$ is in O if and only if its restrictions are in the original table, that is, $o[\mathbb{A}(T)] \in T$ and $o[\mathbb{A}(S)] \in S$. In case that $\mathbb{A}(T) \cap \mathbb{A}(S) = \varnothing$, the join operator is equal to calculating the cross-product.*

We now present an extension of the relational algebra, which we will use in the next subsection to achieve Minimize in PrivTru. This extension enables data sources to evaluate basic propositions on their local entries and to communicate the corresponding truth-values without revealing the underlying data. To enhance readability, we write $(\neg)$, to represent optional negations. As a result, the following $(\neg)$ are also evaluated as negation in the given context.

Definition 2 (Relational Algebra with Propositional Projections). *The* relational algebra with propositional projections (RAPP) *consists of the three primitive operators of the relational algebra, where the* projection *operator is replaced by the following: $\pi_{(A_1,\dots A_n)}(T)$ outputs a new relation containing the columns identified by $A_1, \dots A_n$ in T. Additionally, any A_i can have the form $p : \bigvee_i (\neg)\gamma_i$, where p is any string and all γ_i are literals of the form* `true`*,* `false`*, $A\theta B$ or $A\theta c$, with $A, B \in \mathbb{A}(T), c \in \mathbb{D}(A)$ and $\theta \in \{=, <\}$. In this case, the resulting relation $R = \pi_{(A_1,\dots A_n)}(T)$ consists of a new column identified by p, where each tuple in R holds* `true` *or* `false` *depending on the evaluation of $\bigvee_i (\neg)\gamma_i$ on that tuple in T.*

For example, Table 1 shows a table of persons with their names, ages, and income. Using the propositional projections of RAPP, we can query this data to produce a table that indicates only whether each person is older than 30 (see Table 2). This projection is also supported in modern SQL dialects.

Table 1. T

Name	Age	Income
Alice	30	44k
Bob	33	40k
Carol	50	52k
Eve	42	66k

Table 2. $\pi_{\text{Name, over\_30: age>30}}T$

Name	over_30
Alice	`false`
Bob	`true`
Carol	`true`
Eve	`true`

Calculating Subqueries for Relational Data Exchanges Every data exchange compatible with our model must have a strategy to derive subqueries from the main query. We formally define this problem as follows.

Problem 1 (CalculateSubqueries (CS-Problem))
Input*: Query q in the language of relational algebra over relations $T_1, \dots T_s$.*
Output*: For $i = 1, \dots s$, subqueries $\widehat{q}_i$ in the language of RAPP over T_i and a collecting query $\widehat{q}$ over the relations $R_1, \dots R_s$. So that for every set of tables $T_1, \dots T_s$, the evaluation of q on $T_1 \dots T_s$ is equal to the evaluation of $\widehat{q}$ on $R_1, \dots R_s$, where R_i is the result of the evaluation of $\widehat{q}_i$ on T_i.*

Every algorithm $\mathcal{A}$ that solves Problem 1 can be canonically transferred into a data exchange, given we have algorithms for evaluating queries in relational algebra and RAPP. For this, the exchange calculates $\widehat{q_1}, \dots \widehat{q_s}$ from the input query q (using $\mathcal{A}$) and sends these subqueries to the respective data sources $S_1, \dots S_s$. Each source evaluates its query on its local data and returns the results $R_1, \dots R_s$ to the exchange. The exchange then evaluates $\widehat{q}$ over $R_1, \dots R_s$ and sends the result back to the data receiver.

A trivial algorithm to solve Problem 1 is to set $\widehat{q}_i := T_i$ and set $\widehat{q}$ to q. In this case, the exchange requests all data from all sources, enabling it to calculate q locally. However, this solution directly contradicts the privacy analysis in Section 3. It violates Minimize and undermines the spirit of Separate, since the exchange gains complete knowledge of the data on any query.

To develop a more privacy-friendly solution, we first need to introduce some theoretical preliminaries. Definition 3 defines a form for expressing queries in relational algebras, based on the commonly used normal form [1]. Lemma 1 then presents rewrite rules for these queries.

Definition 3 (Extended Normal Form). *A query q in the language relational algebra is in* extended normal form *if:*

(a) q is in normal form, which means q can be written as $q = \pi_\beta \sigma_F (T_1 \bowtie \dots \bowtie T_k)$.
(b) F is in conjunctive normal form, meaning it can be written as $F := \bigwedge_i x_i := \bigwedge_i \bigvee_j (\neg)\gamma_{i,j}$, where x_i are clauses that are disjunctions of the literals $\gamma_{i,j}$.
(c) If F is not a trivial tautology ($F =$ `true`) or trivially not satisfiable ($F =$ `false`), then F is satisfiable and not a tautology and every clause x in F is satisfiable and not a tautology and every literal γ in F is satisfiable and not a tautology.

Database theory established that every query q can be rewritten in normal form, and every propositional formula F can be rewritten in conjunctive normal form [1]. Furthermore, Definition 3(c) can be achieved using rewrite rules and an algorithm that checks for satisfiability. Consequently, every query q can be expressed in extended normal form. Definition 3 makes the queries manageable

in our system. Definition 3(c) is particularly useful, as it enables mapping literals to the tables that determine their values. More precisely, for any set of literals Γ used in a RAPP query, let $\mathbb{T}(\Gamma)$ be the set of tables T containing $A \in \mathbb{A}(T)$, which is used in a literal $\gamma \in \Gamma$. We sometimes omit parentheses when using $\mathbb{T}$. Since we assume that attributes are unique across all tables and because of Definitions 2 and 3(c), $1 \leqslant \mathbb{T}(\gamma) \leqslant 2$ holds for all γ in our scenario.

The following lemma provides useful rewrite rules for propositions and queries, which we will use to build an algorithm that solves Problem 1.

Lemma 1. *For every set of relations $R_1, \dots R_s$, proposition formula F, literals $\gamma, \gamma_1, \dots \gamma_m$, list of attributes β and $i \in \{1, \dots s\}$:*

(a) If $F := F' \wedge (x)$, where $x = \bigvee_{i=1}^{m} (\neg_i)\gamma_i$ and $\mathbb{T}(\gamma_1, \dots \gamma_m) = \{R_i\}$, then $\sigma_F(R_1 \bowtie \cdots \bowtie R_n) \equiv \sigma_{F'}(R_1 \bowtie \cdots \sigma_x(R_i) \cdots \bowtie R_n)$.

(b) If $F := F' \wedge (\neg_1)\gamma_1 \vee \cdots \vee (\neg_m)\gamma_m$ and for a subset $\Gamma \subset \{\gamma_1, \dots \gamma_m\}, \mathbb{T}(\Gamma) = \{R_i\}$, then $\sigma_F(R_1 \bowtie \cdots \bowtie R_n) \equiv \sigma_{F' \wedge (\bigvee_{\gamma_i \in \{\gamma_1, \dots \gamma_m\} \setminus \Gamma} (\neg_i)(\gamma_i) \vee p = \texttt{true})}(R_1 \bowtie \cdots \pi_{p: \bigvee_{\gamma_j \in \Gamma} (\neg_j)(\gamma_j)} R_i \cdots \bowtie R_n)$.

(c) Let $\mathbb{A}(R_i, F, \beta)$ denote the set of attributes A in β and F, with $A \in \mathbb{A}(R_i)$, then $\pi_\beta \sigma_F(R_1 \bowtie \cdots \bowtie R_n) \equiv \pi_\beta \sigma_F(R_1 \bowtie \cdots \pi_{\mathbb{A}(R_i, F, \beta)}(R_i) \cdots \bowtie R_n)$.

We can now introduce Algorithm 1, which is a solution to Problem 1 as we will prove with Theorem 1. In the algorithm, we treat attribute lists and propositional formulas as sets. More precisely, for a projection π_β in RAPP, β is a set of attributes and constructs of the form $p : \gamma$. Propositional formulas F (in conjunctive normal form) are viewed as a set of sets containing literals γ, where each inner set corresponds to a clause x. The conjunction F is formed by combining these clauses, while each clause represents a disjunction of its literals.

Theorem 1. *Algorithm 1 solves the CS-Problem, for all queries in extended normal form.*

Proof. We can easily see, that Algorithm 1 fulfills the syntactical requirements of Problem 1. It remains to show that q evaluated on $T_1, \dots T_s$ is equal to $\widehat{q}$ evaluated on $R_1, \dots R_s$, where R_i is the evaluation of $\widehat{q}_i$ on T_i. The proof is structured into substatements about the algorithm, utilizing Lemma 1. In the following, we will examine the query Q at different intermediate states of the algorithm. We assume that Q is constructed from the intermediate values of $\beta, \widehat{F}, \beta_i, F_i$ as defined in Statement 1. At the end of the algorithm, $Q = \widehat{q}$.

Statement 1. Q is from the form $\pi_\beta \sigma_{\widehat{F}}(\bowtie_{i=1}^{s} \pi_{\beta_i} \sigma_{F_i}(T_i))$, making Lemma 1 generally applicable to it.

This follows directly from Lines 36 and 39, the construction of Q, and the fact that R_i is the evaluation of $\pi_{\beta_i} \sigma_{F_i} \widehat{q}_i$ on T_i.

Statement 2. The foreach-loop in Line 7 preserves equivalence for Q.

Let Q_0 be the intermediate query before the start of the foreach-loop and Q_k the intermediate query after the k-th round. We need to show that for $k = 1, \dots n$,

Algorithm 1: Query Distribution

Data: Query $q = \pi_\beta \sigma_F(T_1 \bowtie \ldots \bowtie T_s)$ in extended normal form with $F := \bigwedge_{k=0}^{n} x_k := \bigwedge_{k=0}^{n} \bigvee_{j=0}^{n_k} (\neg)\gamma_{k,j}$.

if $F =$ `false` **then**
  return $q^\varnothing, \ldots q^\varnothing$; // $s+1$ times the empty query $q^\varnothing$
$\widehat{q} \leftarrow \varnothing$;
$\widehat{F} \leftarrow F$;
foreach $i \in \{1, \ldots s\}$ **do**
  $\widehat{q}_i \leftarrow \varnothing$; $\beta_i \leftarrow \varnothing$; $F_i \leftarrow \varnothing$; $\Gamma_i \leftarrow \varnothing$;
foreach $x \in F$ **do** // x is a set representing a clause
  if $|\mathbb{T}(x)| = 1$ **then** // Use Lemma 1(a)
    foreach $T_i \in \{T_1, \ldots T_s\}$ **do**
      if $\mathbb{T}(x) = \{T_i\}$ **then**
        $F_i \leftarrow F_i \cup \{x\}$;
        $\widehat{F} \leftarrow \widehat{F} \backslash \{x\}$;
  else // Use Lemma 1(b)
    foreach $(\neg)\gamma \in x$ **do** // Consider only literals in x
      if $\mathbb{T}(\gamma) = \{T_i\}$ **then**
        $\Gamma_i \leftarrow \Gamma_i \cup \{(\neg)\gamma\}$;
    foreach $i \in \{1, \ldots s\}$ **do**
      if $\Gamma_i \neq \varnothing$ **then**
        $\beta_i \leftarrow \beta_i \cup \{p_{x,i} : \bigvee_{(\neg)\gamma \in \Gamma_i} ((\neg)\gamma)\}$;
        $\widehat{x} \leftarrow x \backslash \Gamma_i \cup \{p_{x,i} =$ `true`$\}$;
        $\widehat{F} \leftarrow \widehat{F} \cup \{\widehat{x}\}$;
        $\widehat{F} \leftarrow \widehat{F} \backslash \{x\}$;
foreach $x \in \widehat{F}$ **do**
  foreach $(\neg) A\theta B \in x$ **do** // Literals written as $A\theta B$ $(\theta \in \{=, >\})$
    $\{T_i, T_j\} \leftarrow \mathbb{T}(A\theta B)$;
    $A_i \leftarrow \mathbb{A}(T_i) \cap \{A, B\}$;
    $A_j \leftarrow \mathbb{A}(T_j) \cap \{A, B\}$;
    $\beta_i \leftarrow \beta_i \cup A_i$;
    $\beta_j \leftarrow \beta_j \cup A_j$;
foreach $A \in \beta$ **do**
  $i \leftarrow \{i | A \in \mathbb{A}(T_i)\}$;
  $\beta_i \leftarrow \beta_i \cup A$;
foreach $i \in \{1, \ldots s\}$ **do**
  if $F_i = \varnothing$ **then**
    $F_i \leftarrow \{\{$`true`$\}\}$;
  $\widehat{q}_i \leftarrow \pi_{\beta_i} \sigma_{F_i}(T_i)$;
  if $\beta_i = \varnothing$ **then**
    $\widehat{q}_i \leftarrow q^\varnothing$;
$\widehat{q} \leftarrow \pi_\beta \sigma_{\widehat{F}}(R_1 \bowtie \cdots \bowtie R_s)$;
return $\widehat{q}, \widehat{q}_1, \ldots \widehat{q}_s$

it holds that $Q_{k-1} \equiv Q_k$. Let $k \in \{1, \dots n\}$ and x_k be the clause of the k-th round. If there is a $T_i \in \{T_1, \dots T_s\}$ with $\mathbb{T}(x_k) = \{T_i\}$, Lines 8–12 rewrites Q_{k-1} to Q_k according to Lemma 1 (a). If x_k can not be fully evaluated in one table the code in Lines 13–22 repeatedly rewrites Q_{k-1} using Lemma 1 (b). In Line 17 every Γ_i only holds γ with $\mathbb{T}(\gamma) = \{T_i\}$. As such the Γ_i are subsets, as defined in Lemma 1 (b). The Lines 18–22 only execute the Lemma for each of the subsets Γ_i, making Q_k equivalent to Q_{k-1}.

Statement 3. The code from Line 23 to Line 32 is preserves equivalence for Q.

This property holds because at Line 23 $\widehat{F}$ contains only literals γ with $|\mathbb{T}(\gamma)| = 2$, as literals with $|\mathbb{T}(\gamma)| = 1$ are removed in Lines 12 and 22. Consequently, in Line 30, every β_i holds the attributes that $\widehat{F}$ uses from table T_i ($i = 1, \dots s$). The foreach-loop starting at Line 30 ensures that $\beta_i = \mathbb{A}(T_i, F, \beta) \cup P_i$ where $\mathbb{A}(T_i, F, \beta)$ is as defined in Lemma 1 (c) and P_i are values held β_i before Line 23. Since for projections $\pi_A \pi_B$ can be replaced by $\pi_{A \cup B}$ for every pair of sets A, B, Statement 3 follows by using Lemma 1 (c) for all $i \in \{1, \dots s\}$.

The theorem follows from Statements 1 to 3.

5 Analysis of Information Leakage

In this section, we quantify the information leakage incurred by a central exchange when solving an instance of Problem 1. We show that Algorithm 1 minimizes leakage, regardless of the exchange's prior knowledge.

To measure leakage, we assess the probability of the data exchange correctly reconstructing the full tables $T_1, \dots T_s$ based on $R_1 = \widehat{q_1}[T_1], \dots R_s = \widehat{q_s}[T_s]$. We consider T_i as a set of tuples over the cross-product of its attribute domains, $\mathbb{D}(T_i) := \times_{A \in \mathbb{A}(T_i)} \mathbb{D}(A)$, disregarding row order. The exchange's assumptions about T_i are modeled by the probability measure p^i, sampling from all possible tables $\tilde{T} \in \mathbb{D}(T_i)^2$. We consider only discrete p^i, where $p^i(\tilde{T}) > 0$ holds for a finite number of $\tilde{T}$, reflecting real-world scenarios where attribute domains are naturally finite. A relational exchange updates its assumptions for T_i upon receiving a result R_i. The updated assumptions are modeled by $p^i_{R_i}$. To determine the optimal solution for Problem 1, we evaluate the deviation of $p^i_{R_i}$ for all $i \in \{1, \dots s\}$ from correctly guessing T_i, where R_i is the (implicit) output of a solution.

To quantify the difference between two probabilities p and q, we use the Kullback-Leibler Divergence [11].

Definition 4 (Discrete Kullback-Leibler Divergence [5]). *For two discrete probability measures p and q with sample set X, where $q(x) = 0$ implies $p(x) = 0$, the* Kullback-Leibler Divergence *of p and q (with $0 \log(0) := 0$) is defined as:*

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}.$$

For our case, we calculate the divergence between $p^i_{R_i}$ and $p^i_{T_i}$, where $p^i_{T_i}$ fully reveals T_i. Specifically, $p^i_{T_i}(T_i) = 1$ and $p^i_{T_i}(\tilde{T}) = 0$ for all $\tilde{T} \in \mathbb{D}(T_i)^2 \backslash \{T_i\}$. From

$$D(p^i_{T_i} || p^i_{R_i}) = \sum_{\tilde{T} \in \mathbb{D}(T_i)^2} p^i_{T_i}(\tilde{T}) \log \left(\frac{p^i_{T_i}(\tilde{T})}{p^i_{R_i}(\tilde{T})} \right) = \log \left(p^i_{R_i}(T_i)^{-1} \right) \tag{1}$$

the divergence depends only on the value of $p^i_{R_i}$ at T_i.

Calculating $p^i_{R_i}$ Intuitively, $p^i_{R_i}$ should correspond to the initial probability p^i under the condition that R_i is the result of the evaluation of $\widehat{q}_i$. However, using plain conditional probability $p^i(X|R_i)$ is not feasible. This is because R_i is not guaranteed to be an element of $\mathbb{D}(T_i)^2$, as $\widehat{q}_i$ might utilize projections. Nonetheless, the information gained from R_i can be used to minimize the set of possible candidates $\tilde{T} \in \mathbb{D}(T_i)$. Since R_i is queried from T_i, all entries in R_i must have exactly one compatible entry in $\tilde{T}$. For compatibility, two conditions must be satisfied: (1) For all attributes of R_i that do not result from propositional projections, compatible entries must have the same values projected on these attributes in $\tilde{T}$. (2) For all columns originating from propositional projections, the truth-value in that entry must match the evaluation of that entry in $\tilde{T}$. Formally, we define the possible candidates for T_i given R_i as:

$$\begin{aligned} C^i(R_i) = \{\tilde{T} \in \mathbb{D}(T_i)^2 | \exists \text{ injective } m : R_i \hookrightarrow \tilde{T}, \\ \forall x \in R_i : \pi_{\mathbb{A}(R_i) \cap \mathbb{A}(T_i)} x = \pi_{\mathbb{A}(R_i) \cap \mathbb{A}(T_i)} m(x) \wedge \\ \forall p : \bigvee_i ((\neg)\gamma_i) \in \mathbb{A}(R_i) : \pi_p(x) = \bigvee_i (\neg)\gamma_i[m(x)]\}. \end{aligned} \tag{2}$$

This allows us to define $p^i_{R_i}$ as the probability of p^i under the condition $C^i(R_i)$:

$$p^i_{R_i}(X) := p^i(X \mid C^i(R_i)) = \frac{p^i(X \cap C^i(R_i))}{p^i(C^i(R_i))} \tag{3}$$

Note that if R_i is part of a solution to Problem 1, where T_i is part of the input, then $T_i \in p^i(C^i(R_i))$. As such, $p^i(C^i(R_i)) \neq 0$ if $p^i(T_i) > 0$.

Solving the CS-Problem with Minimal Information Leakage We can now prove that Algorithm 1 is the optimal solution for Problem 1.

Theorem 2. *Let $q = \pi_\beta \sigma_F (\bowtie T_i)$ be a query in RAPP in extended normal form over $T_1, \dots T_s$. Let $\widehat{q}, \widehat{q_1}, \dots \widehat{q_s}$ with $R_1, \dots R_s$ be the solution of Problem 1 given q produced by Algorithm 1. Let $\tilde{q}, \tilde{q_1}, \dots \tilde{q_s}$ with $\tilde{R_1}, \dots \tilde{R_s}$ be any other solution of Problem 1 given q. For any set of discrete probability measures $p^1, \dots p^s$ over $\mathbb{D}(T_1)^2, \dots \mathbb{D}(T_s)^2$ respectively with $p^i(T_i) > 0$ for any $i \in \{1, \dots s\}$:*

$$D(p^i_{T_i} || p^i_{R_i}) \geqslant D(p^i_{T_i} || p^i_{\tilde{R}_i})$$

Proof. Let $i \in \{1, \dots s\}$. With $T_i \in C^i(\tilde{R}_i)$, Eqs. (1) and (3), and the additive property of probability measures, the following holds:
$D(p^i_{T_i} || p^i_{R_i}) \geqslant D(p^i_{T_i} || p^i_{\tilde{R}_i}) \Leftrightarrow p^i(C^i(R_i)) \geqslant p^i(C^i(\tilde{R}_i)) \Leftrightarrow \sum_{\tilde{T} \in C^i(R_i)} p^i(\tilde{T}) \geqslant \sum_{\tilde{T} \in C^i(\tilde{R}_i)} p^i(\tilde{T})$.

Because of this, it suffices to show that $C^i(R_i) \supseteq C^i(\tilde{R}_i)$. We will show this by proving four substatements.

Statement 1. $\mathbb{A}(R_i) \cap \mathbb{A}(T_i) \subseteq \mathbb{A}(\tilde{R}_i) \cap \mathbb{A}(T_i)$

Let $A \in \mathbb{A}(R_i) \cap \mathbb{A}(T_i)$. By definition of $R_i = \pi_{\beta_i} \sigma_{F_i}(T_i)$, it follows that $A \in \beta_i$. From the construction of β_i in Lines 30–32 of Algorithm 1, it holds that $A \in \beta$. Since $q(\bowtie T_i) = \tilde{q}(\bowtie R_i) = \tilde{q}(\bowtie \tilde{q}_i(T_i))$, it follows that $A \in \mathbb{A}(\tilde{R}_i) \cap \mathbb{A}(T_i)$.

Statement 2. For every $x \in F$ with $\{T_i, T_j\} \subseteq \mathbb{T}(x), (j \neq i)$ and every $e \in \tilde{R}_i$ there must exist a projection π_p such that $\pi_p e = \texttt{true}$ iff $\bigvee_{(\neg)\gamma \in \Gamma_i} (\neg)\gamma(e) = \texttt{true}$, where $\Gamma_i = \{(\neg)\gamma \in x \,|\, \mathbb{T}(x) = \{T_i\}\}$. Furthermore, $\pi_{p \cup \mathbb{A}(\tilde{R}_i)} \tilde{R}_i$ is also a solution to Problem 1.

This statement directly follows from the fact that $\tilde{R}_i$ is part of a solution to Problem 1 for q. Since x depends on values from at least T_i and T_j, $\tilde{R}_i$ must indicate, for each of its entries e, whether that entry sets x to `true`. By definition, this is the case if $\bigvee_{(\neg)\gamma \in \Gamma_i} (\neg)\gamma(e)$ is `true`.

Statement 3. For all $\tilde{T} \in C^i(\tilde{R}_i)$, there exists β and F such that $\pi_\beta \sigma_F(\tilde{T}) = \tilde{R}_i$.

Let $\beta = (\mathbb{A}(\tilde{R}_i) \cap \mathbb{A}(\tilde{T})) \cup \{p : (\neg)\gamma | p : (\neg)\gamma \in \mathbb{A}(\tilde{R}_i)\}$ and F be defined as $F(e) = \texttt{true}$ iff $e \in Img(\tilde{m})$. Here, $Img(\tilde{m})$ represents the image of the function $m : \tilde{R}_i \hookrightarrow \tilde{T}$, as defined in eq. (2).

Statement 4. Let $M_i : R_i \hookrightarrow T_i$ and $\tilde{M}_i : \tilde{R}_i \hookrightarrow T_i$ be the functions for $T_i \in C^i(R_i) \cap C^i(\tilde{R}_i)$ in eq. (2). Then, $Img(M_i) \subseteq Img(\tilde{M}_i)$.

We prove this statement by contradiction. Let $e \in Img(M_i) \backslash Img(\tilde{M}_i)$. Let β', F', $\tilde{\beta}_i$, and $\tilde{F}$ be such that $\pi_{\beta'} \sigma_{F'}(T_i) = R_i$ and $\pi_{\tilde{\beta}_i} \sigma_{\tilde{F}}(T_i) = \tilde{R}_i$ (as shown in Statement 3). By construction, $F'(e) = \texttt{true}$ and $\tilde{F}(e) = \texttt{false}$. Since R_i and $\tilde{R}_i$ are part of solutions of Problem 1, it holds that $q(\bowtie T_i) = \widehat{q}(\bowtie R_i) = \tilde{q}(\bowtie \tilde{R}_i)$. Therefore, e cannot be a part of the evaluation of $q(\bowtie T_i)$. More precisely, for every $t \in q(\bowtie T_i)$ with $\pi_{\mathbb{A}(e)} t = e$: $F(t) = \texttt{false}$. Since this needs to hold independently of the other $T_j (j \neq i)$, there exists a clause $x \in F$ with $\mathbb{T}(x) = \{T_i\}$ and $x(e) = \texttt{false}$. Let x be such a clause. We now look at the construction of R_i in Algorithm 1. By Line 11, $x \in F_i$. Hence, $e \notin R_i$. This is a contradiction with $F'(e) = \texttt{true}$.

Given the four statements, we can show that $C^i(R_i) \subseteq C^i(\tilde{R}_i)$. Let $\tilde{T} \in C^i(\tilde{R}_i)$ with $\tilde{m} : \tilde{R}_i \hookrightarrow \tilde{T}$ and $M_i, \tilde{M}_i$ as described in Statement 4. We need to show that there exists an $m : R_i \hookrightarrow \tilde{T}$ satisfying the properties in Eq. (2). We argue that $m : x \mapsto \tilde{m}(\tilde{M}_i^{-1}(M_i(x)))$ is such a function. The function is well-defined due to Statement 4. Its domain is $\tilde{T}$, and it is injective as a concatenation of injective functions. The expression $\pi_{\mathbb{A}(R_i) \cap \mathbb{A}(T_i)} x = \pi_{\mathbb{A}(R_i) \cap \mathbb{A}(T_i)} m(x)$ is only a problem when mapping $M_i(x)$ to $\tilde{M}_i^{-1}(M_i(x))$. However, since $\tilde{M}_i^{-1}$ satisfies the required property, it holds that $\pi_{\mathbb{A}(\tilde{R}_i) \cap \mathbb{A}(T_i)} \tilde{M}_i^{-1}(M_i(x)) = \pi_{\mathbb{A}(\tilde{R}_i) \cap \mathbb{A}(T_i)} M_i(x)$,

and the property follows from Statement 1. From the construction of R_i in Algorithm 1, we know that R_i only has propositional projections if there exists a clause $x \in F$ with $\{T_i, T_j\} \subseteq \mathbb{T}(x), (j \neq i)$, and all have the form $p_{x,i} : \bigvee_{(\neg)\gamma \in \Gamma_i} (\neg)\gamma$. To fulfill the second property of Eq. (2), we assume WLOG that $\tilde{R}_i$ has attributes $\tilde{p}_{x,i}$ for all $p_{x,i}$ which are true iff the disjunction of $p_{x,i}$ above is `true` (as described in Statement 2). With this, m satisfies the second property of Eq. (2), because $\tilde{m}$ satisfies it.

6 Related Work

The concept of data trustees aligns with models used in distributed database systems, where data is managed across multiple sources. Unlike distributed databases, which often replicate data across nodes, data trustees typically manage unique data from different sources. An overview of distributed database principles can be found in [14].

To minimize information leakage, we push query execution as close to the data sources as possible. This approach is different to the late materialization technique used in database management systems to enhance performance [6]. Despite these advantages, PrivTru currently does not support non-relational data formats, which limits its applicability in certain cases. For example, sharing medical data such as MRI images from multiple sources poses challenges, as discussed by Kolain and Malavi [10]. Extracting only the minimum required information from such detailed data remains a significant challenge.

Beyond the privacy strategies implemented in our work, systems can be evaluated using frameworks such as the LINDUNN [4] or the privacy protection goals proposed in [8]. We believe PrivTru fulfills their requirements as well.

7 Conclusion

In this paper, we introduced PrivTru, a privacy-by-design data trustee, which facilitates data exchange from data sources to data receivers. We contributed to the ongoing discussion on whether data trustees should act as data stewards (storing data) or as data exchanges (relaying information). We argue that data exchanges are preferable, as they comply with the Separate design strategy while also adhering to the principles of Hide, Minimize, and Aggregate. PrivTru exemplifies an instantiation of data exchanges, achieving Minimize by ensuring that each data source provides only the minimum data necessary to answer a query. We proved that this property holds regardless of the exchange's prior knowledge. While we did not explore Aggregate in detail, it can be achieved by allowing queries with aggregation functions to be evaluated at the data sources, as done with other operations in PrivTru. In summary, our work establishes a foundation for designing data trustees that balance data utility with robust privacy protection, contributing to a more trustworthy data economy.

Acknowledgements

This research has been funded by the Federal Ministry of Education and Research of Germany (BMBF, Project 16KISA038).
The final publication is available at Springer via
https://doi.org/10.1007/978-3-031-92882-6_8

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases (1995)
2. Aline Blankertz: Designing data trusts. Why we need to test consumer data trusts now (2020), https://www.interface-eu.org/storage/archive/files/designing_data_trusts_e.pdf, accessed: 2024-12-11
3. Blankertz, A., Specht, L.: What regulation for data trusts should look like (2021), https://www.interface-eu.org/storage/archive/files/regulation_for_data_trusts_0.pdf, accessed: 2024-12-11
4. Cavoukian, A.: Privacy by Design [Leading Edge]. IEEE Technology and Society Magazine **31**, 18–19 (2012). https://doi.org/10.1109/MTS.2012.2225459
5. Chambert-Loir, A.: Information Theory: Three Theorems by Claude Shannon, UNITEXT, vol. 144. Cham (2022). https://doi.org/10.1007/978-3-031-21561-2
6. Chernishev, G., Galaktionov, V., Grigorev, V., Klyuchikov, E., Smirnov, K.: A Comprehensive Study of Late Materialization Strategies for a Disk-Based Column-Store. In: DOLAP. pp. 21–30 (2022)
7. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**, 377–387 (1970). https://doi.org/10.1145/362384.362685
8. Hansen, M., Jensen, M., Rost, M.: Protection Goals for Privacy Engineering. IEEE Security and Privacy Workshops (2015). https://doi.org/10.1109/SPW.2015.13
9. Hoepman, J.H.: Privacy Design Strategies. In: ICT Systems Security and Privacy Protection. p. 446 (2014). https://doi.org/10.1007/978-3-642-55415-5_38
10. Kolain, M., Molavi, R.: Zukunft Gesundheitsdaten (2019), https://www.bundesdruckerei-gmbh.de/files/dokumente/pdf/studie_zukunft-gesundheitsdaten.pdf, accessed: 2024-12-11
11. Kullback, S., Leibler, R.A.: On Information and Sufficiency. The Annals of Mathematical Statistics **22**, 79–86 (1951). https://doi.org/10.1214/aoms/1177729694
12. Reiberg, A., Appelt, D., Kraemer, P.: Data Trusts, Data Intermediation Services and Gaia-X (2023), https://gaia-x-hub.de/wp-content/uploads/2023/11/GX-White-Paper-Data-Trusts.pdf, accessed: 2024-05-29
13. Samarati, P., Sweeney, L.: Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Proceedings of the IEEE Symposium on Research in Security and Privacy (1998)
14. Xu, Q., Yang, C., Zhou, A.: Native Distributed Databases: Problems, Challenges and Opportunities. Proceedings of the VLDB Endowment pp. 4217–4220 (2024)