Compatibility Checking Between Privacy and Utility Policies: A Query-Based Approach

Hira Asghar, Christophe Bobineau, Marie-Christine Rousset

▶ **To cite this version:**

Hira Asghar, Christophe Bobineau, Marie-Christine Rousset. Compatibility Checking Between Privacy and Utility Policies: A Query-Based Approach. [Research Report] Université Grenoble Alpes; CNRS; Grenoble INP; Laboratoire d'informatique de Grenoble. 2021. ⟨hal-03385977⟩

HAL Id: hal-03385977

https://hal.science/hal-03385977v1

Submitted on 19 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Compatibility Checking Between Privacy and Utility Policies: A Query-Based Approach

Hira Asghar
Christophe Bobineau
firstname.lastname@univ-grenoble-alpes.fr
Université Grenoble Alpes, CNRS, Grenoble INP, LIG
Grenoble, France

Marie-Christine Rousset
Marie-Christine.Rousset@univ-grenoble-alpes.fr
Université Grenoble Alpes, CNRS, Grenoble INP, IUF, LIG
Grenoble, France

Abstract

Data sharing over the internet through smart devices is susceptible to disclose sensitive information of data producers. To protect the privacy of data producers, we propose a query-based approach where data producers keep their data on decentralized personal data servers and only disclose data to data consumers over secure communication links according to their privacy policies. Data consumers specify the data needed to provide services as utility policies. In our approach, we express the privacy and utility policies as sets of temporal aggregated conjunctive queries. We make explicit several sufficient conditions of compatibility between privacy and utility policies based on their query expressions. On the basis of these results, sensitive data breaches can be prevented by checking whether one of these sufficient conditions is satisfied.

Keywords: temporal RDF graphs, temporal aggregated conjunctive queries, utility policy, privacy policy

1 Introduction

Personal data are increasingly disseminated over Internet through mobile devices and smart environments, and are exploited for developing more and more sophisticated services and applications. All these advances come with serious risks for privacy breaches that may reveal private information wanted by users to remain undisclosed. It is therefore of utmost importance to help data producers to keep the control on their data for their privacy protection while preserving the utility of disclosed data for service providers.

In this paper, we approach the problem of utility-aware privacy preservation in the setting of applications where service providers (e.g., power suppliers) perform data analytics on data concerning their customers (e.g., smart home occupants) for optimization or recommendation purposes. In such settings, (sensor) data are gathered, abstracted and transferred through internet protocols from data producers environment (e.g., smart home, smart personal devices) to a centralized data consumer in charge of aggregating data for conducting varied analytics tasks.

Sensitive data leakage can occur at different stages and places due to security vulnerabilities of *(1)* the network, *(2)* the centralized server used by the data consumer for collecting data outsourced by the different data producers, and *(3)* the local servers of each data producer.

Following the vision of [1], we propose, first, to rely on data encryption to secure data exchange through the network and, second, to avoid the privacy risks of data centralization by keeping the data produced by each data owner decentralized in secure personal data servers.

The approach that we promote to face the privacy versus utility dilemma in this setting can be summarized as follows:

1. Data producers keep the control on the data they accept to transmit to the data consumer according to their own *privacy policy*.
2. The data consumer makes explicit *his/her utility policy* to explain for which task or service s/he requests data from data producers.
3. In case of incompatibility of the utility policy with the privacy policy of a data producer, the data producer negotiates with the data consumer to find an acceptable privacy-utility trade-off.

In the remaining of this paper, we focus on the problem of checking compatibility between privacy and utility policies, that is at the core of our approach.

Our contribution is twofold. First, we extend the framework proposed in [5] to *formalize privacy and utility policies as temporal aggregate queries*. Second, we formally define and study the compatibility problem in this query-based framework. In particular, we *exhibit several conditions for compatibility or incompatibility* that can be automatically verified by comparing the query expressions defining privacy and utility policies.

The paper is organized as follows. In Section 2, we provide the formal background on which our approach relies. In Section 3, we describe the query-based formal framework that we propose to define privacy and utility policies and their compatibility. In Section 4, we summarize our contribution for checking compatibility between privacy and utility policies. Section 5 presents related work, and Section 6 concludes our paper and highlights the future work that we plan to conduct.

2 Formal background

This section provides the notations and definitions of temporal aggregate conjunctive queries and their evaluation over temporal RDF graphs, on which our approach relies. We illustrate these notions through examples.

2.1 Preliminaries

Let I, L, B, and Var be pairwise disjoint sets representing IRIs, literals, blank nodes, and variables. In the set L of literals we distinguish TS as a subset denoting timestamps. In the set of variables Var we distinguish one specific time variable denoted $?timeWindowEnd$.

We will call *constants* the elements of $I \cup L$.

Definition 1 (Temporal RDF graphs)**.** A temporal RDF triple is a pair $(s\ p\ o\ , t)$ where $s\ p\ o \in (I \cup B) \times I \times (I \cup L \cup B)$ and $t \in TS$.
A temporal RDF graph is a set of temporal RDF triples.

By convention and for homogeneity purpose we consider RDF graphs where all the triples are temporal by using *any* as a special timestamp when the corresponding triples holds at any time. We will call *static* the properties involved in triples with *any* as timestamps.

Example 1 illustrates a temporal RDF graph describing the data of a given house owner using a simple ontology with properties `familySize`, `address`, `city`, `street` (that are static) and `consumption`.

Example 1. The family size, address and street are expressed as triples with special timestamp *any* whereas energy consumption triples are timestamped.

Temporal RDF graph of data producer DP_1

$(dp_1\ familySize\ 4, any)$
$(dp_1\ address\ \_ : b_1, any)$
$(\_ : b_1\ street\ \text{“}AlsaceLorraine\text{”}, any)$
$(\_ : b_1\ city\ \text{“}Grenoble\text{”}, any)$
$(dp_1\ consumption\ 40, 2021-04-04T15:30:00)$
$(dp_1\ consumption\ 20, 2021-04-04T16:00:00)$
$(dp_1\ consumption\ 30, 2021-04-04T16:30:00)$
$(dp_1\ consumption\ 10, 2021-04-04T17:00:00)$

Definition 2 (Temporal graph pattern)**.** A Temporal graph pattern is a finite set of temporal triple patterns, where a temporal triple pattern is a pair $(s\ p\ o, ?ts)$ such that $s\ p\ o \in (I \cup B \cup Var) \times (I \cup Var) \times (I \cup L \cup B \cup Var)$. Variables $?ts$ are called timestamp variables.
For a temporal graph pattern GP, we will denote $Var(GP)$ the set of all variables occurring in GP, including $?timeWindowEnd$.

Definition 3 (Graph homomorphisms)**.** Let H and H' be temporal RDF graphs or temporal graph patterns. An homomorphism from H' to H is an application $h : (I \cup L \cup B \cup Var) \rightarrow (I \cup L \cup B \cup Var)$ such that h(c) = c for $c \in L \cup I$ and $h(H') \subseteq H$ where:

$$h((s\ p\ o, t)) = (h(s)\ h(p)\ h(o), h(t))$$

Notation: For a tuple of variables $\bar{x}$, $\mu(\bar{x})$ is the tuple obtained by replacing each variable x by its value $\mu(x)$.

Definition 4 (Unifiable graph patterns)**.** Let GP_1 and GP_2 two temporal graph patterns. GP_1 and GP_2 are *unifiable* if there exists a function s replacing variables from GP_1 and GP_2 by constants or by variables of GP_1, such that $s(GP_1) = s(GP_2)$.

Definition 5 (Overlapping graph patterns)**.** Let GP_1 and GP_2 two temporal graph patterns. GP_1 and GP_2 are overlapping if they contain subgraphs that are unifiable.

Example 2. The two following listings show two overlapping temporal graph patterns, where the unifiable subgraphs are indicated in bold.

Temporal graph pattern GP_1

(?**dp address** ?**a**, ?**ts**)
(?**a city** ?**city**, ?**ts**)
$(?a\ street\ ?street, ?ts)$

Temporal graph pattern GP_2

$(?dp\ consumption\ ?consumption, ?ts)$
(?**dp address** ?**a**, ?**ts**)
(?**a city** "**Grenoble**", ?**ts**)

Definition 6 (Disjoint graph patterns)**.** Let GP_1 and GP_2 two temporal graph patterns. GP_1 and GP_2 are disjoint if they are not overlapping.

Definition 7 (Isomorphic graph patterns)**.** Let GP_1 and GP_2 two temporal graph patterns. GP_1 and GP_2 are isomorphic if there is a homomorphism h from GP_1 to GP_2 that is bijective and its inverse is also a homomorphism.

2.2 Temporal aggregated conjunctive queries

We first formally define the general form of temporal aggregate conjunctive queries (TACQs) with a SPARQL-like syntax extended with time windows for capturing aggregate on time.

To simplify the exposition, we will only consider aggregate queries which have a single aggregation term. In most cases, queries with several aggregate terms are equivalent to the unions of queries with same body and a single aggregate term [4].

Definition 8 (Temporal aggregated conjunctive query)**.** A $TACQ$ is defined as
SELECT $\bar{x}$, $agg(y)$
WHERE {GP . $FILTER$}
GROUP BY $\bar{x}$
TIMEWINDOW ($Size$, $Step$)
where

- GP is a temporal graph pattern,
- $FILTER$ is a boolean combination of atomic comparisons of the form $t > t'$ or $t \geq t'$ where t and t' are variables of $Var(GP)$ or literals (numbers, strings or dates),
- $\bar{x}$ is a tuple of variables called the output (or grouping) variables,
- when the aggregate term $agg(y)$ is present, y (called the aggregate variable) is not in $\bar{x}$ and agg is an aggregate function that produces a single value when applied to a set of values assigned to y.
- $Size$ and $Step$ are time durations (i.e. differences between timestamps),

The general syntax can be simplified as follows for capturing particular cases:

- When either $\bar{x}$ is empty or there is no aggregate term, we can omit the GROUP BY clause.
- When $Size = \infty$ (and thus $Step = 0$), the TIMEWINDOW clause can be omitted.
- The $FILTER$ clause can be omitted when the corresponding boolean expression is TRUE (called empty $FILTER$). Note however, that when TIMEWINDOW is specified, FILTER always contains the implicit following constraints for each timestamp variable $?ts$: $?ts \leq ?timeWindowEnd \wedge ?ts > ?timeWindowsEnd - Size$.
- For static properties, the timestamp variables can be omitted as they can only be assigned to the special timestamp *any*, and the corresponding temporal triple patterns can be simplified into standard triple patterns.

In Example 3, we illustrate several of these cases.

Example 3.
$TACQ_1$ is an example of a standard conjunctive query which is a particular case of a TACQ in which there is no (temporal) aggregation. It asks the street and city of each data producer.

$TACQ_1$: a particular case of a simple conjunctive query

```
SELECT  ?dp ?street ?city
WHERE {?dp address ?a . ?a street ?street .
       ?a city ?city}
```

$TACQ_2$ is a temporal aggregate conjunctive query where the aggregate is on the temporal variable $?timeWindowEnd$ for a sliding time window of 6 hours and asks the sum of consumption of all data producers computed every hour over the previous 6 hours.

$TACQ_2$: a simple temporal aggregate conjunctive query

```
SELECT ?timeWindowEnd SUM(?consumption)
WHERE {(?dp consumption ?consumption, ?timestamp)}
GROUP BY ?timeWindowEnd
TIMEWINDOW (6h, 1h)
```

$TACQ_3$ is a general TACQ that asks the average consumption, computed every hour over the previous 3 hours, and grouped by consumer, street and the time window end, of data producers living in Grenoble whose size family is greater than 2.

$TACQ_3$: a general temporal aggregate conjunctive query

```
SELECT ?dp ?street  ?timeWindowEnd
                    AVG(?consumption)
WHERE {
  (?dp consumption ?consumption, ?ts) .
  ?dp familySize ?fSize . ?dp address ?a .
  ?a street ?street . ?a city "Grenoble" .
  FILTER (?fSize > 2)}
GROUP BY  ?dp ?street  ?timeWindowEnd
TIMEWINDOW (1h, 1h)
```

Given a TACQ :
SELECT $\bar{x}$, $agg(y)$
WHERE $\{GP\ .\ FILTER\}$
GROUP BY $\bar{x}$
TIMEWINDOW ($Size$, $Step$)
its evaluation over a given temporal data graph G is defined in terms of filtered homomorphism and groups (respectively defined in Definition 9 and Definition 10) for obtaining its answer set (Definition 11).

Definition 9 (Filtered homomorphisms)**.** Let M the set of homomorphisms from GP to G. The filtered set of homomorphisms is the subset of M of homomorphisms μ such that μ(FILTER) = TRUE.

When FILTER is empty then the filtered set of homomorphisms is equal to M.

Definition 10 states that there are as many groups as homomorphisms allowing to match the tuple $\bar{x}$ with tuples of values $\bar{v}$ multiplied by the number of time intervals defined by values of k as: $]now - k \times Step - Size, now - k \times Step]$ where now denotes the timestamp at which the query is executed.

Definition 10 (Groups)**.** Let FM be the set of filtered homomorphisms from GP to G. Groups are defined for each tuple $\bar{v}$ and each time interval k as follows:
$Group_k(\bar{v}) = \{\mu(y) \mid \mu \in FM, \mu(\bar{x}) = \bar{v},$ and for each timestamp variable $?ts$ $\mu(?ts) = any$ or $\mu(?ts) \in]now - k \times Step - Size, now - k \times Step])$ and $\mu(?timeWindowEnd) = now - k \times Step\}$.

It is important to note that if there is no aggregate term, there is only one time interval (i.e., $]-\infty, now]$) and there are as many groups as distinct tuples $\bar{v}$.

For each group $Group_k(\bar{v})$, an answer is: either the tuple $\bar{v}$ if there is no aggregate term, or the tuple $(\bar{v}, r)$ obtained by concatenating the tuple $\bar{v}$ with the result r of the aggregation function applied to the values in the group. This is formalized in Definition 11.

Definition 11 (Answer set)**.** The answer set of $TACQ$ evaluated over G is defined as follows:

- if there is no aggregate term:
  $Ans(TACQ, G) = \{\bar{v} \mid Group_k(\bar{v})$ is a group of $TACQ$ for $G\}$.
- else:
  $Ans(TACQ, G) = \{(\bar{v}, agg(Group_k(\bar{v}))) \mid Group_k(\bar{v})$ is a group of $TACQ$ for $G\}$.

Definition 12 makes explicit the homomorphism support of an answer.

Definition 12 (Homomorphism support of an answer)**.** Let $\bar{a}$ an answer obtained from a given group $Group_k(\bar{v})$. Its homomorphism support, denoted $Hsupport(\bar{a})$, is the subset of filtered homomorphisms μ such that $\mu(\bar{x}) = \bar{v}$.

Example 4 shows the answers sets of queries in Example 3 over the temporal RDF graph DP_1 of Example 1.

Example 4.
The answer set of $TACQ_1$ is:

Answer set of $TACQ_1$ over DP_1

$Ans(TACQ_1, DP_1) =$
{(dp_1, "Alsace Lorraine", "Grenoble")}

The homomorphism support of its single answer is restricted to the single homomorphism h shown below.

Homomorphism support of $Ans(TACQ_1, DP_1)$

$Hsupport = \{h\}$
$h : \{?dp/dp_1, ?a/\_ : b_1, ?street/"AlsaceLorraine", ?city/"Grenoble"\}\}$

When $TACQ_3$ is evaluated at 17:30 the same day as the temporal triples in DP_1 (i.e., now = 2021-04-04T17:30:00), its answer set is:

Answer set of $TACQ_3$ over DP_1

$Ans(TACQ_3, DP_1) =$
{(dp_1, "Alsace Lorraine", 2021-04-04T17:30:00, 10)
(dp_1, "Alsace Lorraine", 2021-04-04T16:30:00, 25)
(dp_1, "Alsace Lorraine", 2021-04-04T15:30:00, 40)}

The homomorphism support of the second answer is made of two homomorphisms h_1 and h_2 which differ in the assignment of the aggregate variable ?consumption.

Homomorphism support of an answer of $TACQ_3$

$Hsupport = \{h_1, h_2\}$
h_1 :
{ ?dp/dp_1, ?street/"Alsace Lorraine", ?size/4, ?consumption/20, ?ts/2021-04-04T16:00:00, ?timeWindowEnd/2021-04-04T16:30:00}
h_2 :
{ ?dp/dp_1, ?street/"Alsace Lorraine", ?size/4, ?consumption/30, ?ts/2021-04-04T16:30:00, ?timeWindowEnd/2021-04-04T16:30:00}

Based on Definition 3, the evaluation over (partially instantiated) *graph patterns* of plain conjunctive queries is possible and the resulting set of answers is defined in Definition 13.

Definition 13 (Conjunctive query evaluated over graph patterns)**.**
Let Q: SELECT $\bar{x}$ WHERE GP be a plain conjunctive query, and let PIG a graph pattern. The answer set of Q over PIG is:

$Ans(Q, PIG) = \{\bar{a} | \bar{a}$ is a tuple of constants and h is an homomorphism from GP to PIG such that $h(\bar{x}) = \bar{a}\}$

3 Query-based specification of policies

We define utility and privacy policies in the form of TACQs that are built upon a common schema or ontology.

Definition 14 (Utility policy)**.** A utility policy is defined by a set of TACQ queries, called utility queries. A utility policy (issued by a service provider) is satisfied by a data producer if s/he accepts to provide the set of answers of all of the utility queries for any RDF graph storing her/his data.

Definition 15 (Privacy policy)**.** A privacy policy is defined by a set of TACQ queries, called privacy queries. A privacy policy, specific to each data producer, is satisfied when all the answers of all the privacy queries remain undisclosed for any temporal RDF graph storing her/his data.

Checking whether her/his privacy policy is incompatible with the utility policy of a service provider is an important property to be checked in the name of each data producer for guiding her/his decision of satisfying the utility policy.

Independently of any graph data, we have to prevent that answers to an utility query allow to infer answers to one of the privacy queries. Definition 17 formalizes incompatibility as inferring answers of a privacy query from sets of answers of utility queries on the same data graph G *without necessarily knowing it.*

It relies on Definition 16 that defines the logical signature of an answer (set) of a query as the logical formula characterizing all the (unknown) temporal data graphs leading to this answer (set) for this query.

Given a TACQ Q:
SELECT $\bar{x}$, $agg(y)$
WHERE $\{GP\ .\ FILTER\}$
GROUP BY $\bar{x}$
TIMEWINDOW ($Size$, $Step$)
we interpret GP as the logical conjunction of its triple pattern seen as atomic formulas.

Definition 16 (Logical signature of answers)**.** For an answer $(\bar{a}, r)$ to the query Q, let $\mu_{\bar{a}}$ the mapping assigning each grouping variable x in $\bar{x}$ to the corresponding constant a in $\bar{a}$. The logical signature of $(\bar{a}, r)$ and Q, denoted $\sigma((\bar{a}, r), Q)$, is the formula:

$(\exists y \exists \bar{z}\, \mu_{\bar{a}}(GP) \wedge \mu_{\bar{a}}(FILTER))$
$\wedge\, agg(\{y | \exists \bar{z},\, \mu_{\bar{a}}(GP) \wedge \mu_{\bar{a}}(FILTER)\}) = r$
where $\bar{z}$ is the (possibly empty) subset of variables in $Var(GP)$ non including the aggregate variable y.
When there is no aggregate variable, the logical signature is reduced to the formula: $(\exists y \exists \bar{z}\, \mu_{\bar{a}}(GP) \wedge \mu_{\bar{a}}(FILTER))$.

The logical signature of an answer set *AnswerSet* of a given query Q is the conjunction of the logical signatures of each answer:

$$\sigma(AnswerSet, Q) = \bigwedge_{ans \in AnswerSet} \sigma(ans, Q)$$

Example 5 shows logical signatures of answers of a simple conjunctive query and of a temporal aggregate query.

Example 5. The logical signature of the answer (dp$_1$, "Alsace Lorraine", "Grenoble") of $TACQ_1$ is provided in the following listing.

Logical signature of the answer (dp$_1$, "Alsace Lorraine", "Grenoble") of $TACQ_1$

```
σ((dp1, "Alsace Lorraine","Grenoble") , TACQ1) :
∃?a, dp1 address ?a ∧ ?a street "Alsace Lorraine"
     ∧ ?a city "Grenoble".
```

The logical signature σ((dp$_1$, "Alsace Lorraine",2021-04-04T17:30:00, 20), $TACQ3$) is provided in the following listing.

Logical signature of the answer (dp$_1$, "Alsace Lorraine", 2021-04-04T17:30:00, 20) of $TACQ_3$

```
σ((dp1, "Alsace Lorraine", 2021-04-04T17:30:00,
    20), TACQ3) :
∃?consumption ∃?ts ∃?fSize ∃?a,
(dp1 consumption ?consumption,?ts)
∧ dp1 familySize ?fSize ∧ dp1 address ?a
∧ ?a street "Alsace Lorraine" ∧ ?a city "Grenoble"
∧ ?fSize > 2 ∧ ?ts ≤ 2021-04-04T17:30:00
∧ ?ts > 2021-04-04T14:30:00
∧ AVG {?consumption | ∃?ts ∃?fSize ∃?a, (dp1
    consumption ?consumption, ?ts) ∧ dp1 familySize
    ?fSize ∧ dp1 address ?a ∧ ?a street "Alsace
    Lorraine" ∧ ?a city "Grenoble" ∧ ?fSize > 2 ∧
    ?ts ≤ 2021-04-04T17:30:00 ∧ ?ts > 2021-04-04T16
    :30:00 } = 20
```

Definition 17 (Incompatibility between privacy and utility)**.** A privacy policy is incompatible with a utility policy if the logical signature of an answer to a privacy query is entailed by the union of logical signatures of answers sets of utility queries.

Continuing Example 5, we have:
$\sigma((dp_1, \text{"}Alsace\ Lorraine\text{"}, 2021-04-04T17:30:00, 20), TACQ_3)$
$\models \sigma((dp_1, \text{"}Alsace\ Lorraine\text{"}, \text{"}Grenoble\text{"}), TACQ_1)$
Therefore a privacy policy made of the single privacy query $TACQ_1$ is incompatible with any utility policy containing $TACQ_3$ as utility query.

Definition 20 defines a weaker notion of incompatibility that prevents from the risk to get an answer to a privacy query among the answers inferred by variants of utility queries. A variant of a query only differs from the original query with the FILTER condition while preserving satisfiability.

Definition 18 (Satisfiable Boolean expression)**.** A Boolean expression *Exp* is satisfiable if there exists at least an assignment of variables in *Exp* that makes it TRUE.

Definition 19. A query Q' is a variant of Q if the two queries differ only on their FILTER part and FILTER$_Q \wedge$ FILTER$_{Q'}$ is satisfiable.

Example 6. Q'_u is a variant of Q_u where Q'_u is:
SELECT ?x WHERE {?x p ?y . FILTER (?y > 2)}
and Q_u is:
SELECT ?x WHERE {?x p ?y . FILTER (?y < 4)}

Definition 20 (Weak incompatibility between privacy and utility)**.** A privacy query Q_p is weakly incompatible with a set of utility queries if Q_p is incompatible with some variants of the utility queries.

Continuing Example 6, let us consider the privacy query Q_p:
SELECT ?x WHERE {?x p ?y. FILTER (?y > 2)}

Q_p is weakly incompatible with a utility policy made of the single utility query Q_u:
SELECT ?x WHERE {?x p ?y. FILTER (?y < 4)}

Note that Q_p is compatible (i.e., not incompatible) with Q_u since the logical signature of each answer a of Q_p is: $\exists$?y (a p ?y) $\wedge$ (?y > 2)
which is *not* logically entailed by the logical signature of an answer a of Q_u which is the formula: $\exists$?y (a p ?y) $\wedge$ (?y < 4).
However it is incompatible with the variant Q'_u of Q_u given in Example 6.

4 Compatibility checking

In this section, we exhibit a set of sufficient and necessary conditions for compatibility or incompatibility.

In Section 4.1, we start by focusing on the case where utility and privacy queries are plain conjunctive queries without aggregate terms and without FILTER conditions. Then, in Section 4.2, we handle the case where utility and privacy queries are conjunctive queries with FILTER conditions. Finally, Sections 4.3 and 4.4 are dedicated to the more general case of queries with aggregate terms possibly with time windows.

Without loss of generality, by renaming variables within each query, we consider that queries have no variable in common.

For a TACQ in its general form, we will often rely on its conjunctive part defined in Definition 21.

Definition 21 (Conjunctive part of a query)**.** Let Q a TACQ of the form:
SELECT $\bar{x}$ $agg(y)$
WHERE $\{GP\ .\ FILTER\}$
GROUP BY $\bar{x}$
TIMEWINDOW $(Size, Step)$
The conjunctive part of Q, noted $Conj(Q)$ is the plain conjunctive query defined as follows:
$Conj(Q)$: SELECT $\bar{x}$ WHERE $\{GP\}$

4.1 Case of plain conjunctive queries

Theorem 4.1 is a characterization of incompatibility of a privacy query Q_p and a set of n utility queries Q_{u_1}, ..., Q_{u_n} when all the queries are plain conjunctive queries.

We will use the following notations for the different query expressions:
Privacy query Q_p*:*
SELECT $\bar{x}_p$ WHERE $\{GP_p\}$
Utility query Q_{u_i}*:*
SELECT $\bar{x}_{u_i}$ WHERE $\{GP_{u_i}\}$

Theorem 4.1 relies on the evaluation of the privacy query over the union of partially instantiated graph patterns obtained by *freezing* the output variables in the graph patterns of the utility queries.

Definition 22 (Frozen graph patterns)**.** Let GP a temporal graph pattern and X a subset of variables occurring in it. A freezing of X in GP, denoted Frozen(GP,X), is the graph pattern obtained from GP by replacing each occurrence of $x \in X$ by a constant.

Theorem 4.1 (Incompatibility of conjunctive queries)**.** *The privacy query Q_p is incompatible with the set of utility queries if and only if $Ans(Q_p, Frozen) \neq \emptyset$ where $Frozen$ is a freezing in $\bigcup_{i\in[1..n]} GP_{u_i}$ of the output variables of the utility queries.*

Proof. In the case of conjunctive queries and according to Definition 16, the logical signature of an answer $\bar{a}$ to a query Q is a formula of the form: $\exists \bar{z}\ \mu_{\bar{a}}(GP)$ where GP is the conjunction of triples patterns in the graph pattern of Q interpreted as logical atoms and μ is an assignment of the tuple $\bar{x}$ of the output variables of Q to the tuple of constants $\bar{a}$, and $\bar{z}$ are the other variables occurring in Q.

If Q_p is incompatible with the utility queries, it means by definition that there exists tuples of constants $\bar{a}$, $\bar{a}_1$, ..., $\bar{a}_n$ such that $\exists \bar{z}_1 \ldots \exists \bar{z}_n\ \mu_{\bar{a}_1}(GP_{u_1}) \wedge \ldots \wedge \mu_{\bar{a}_n}(GP_{u_n}) \models \exists \bar{z}\ \mu_{\bar{a}}(GP_p)$.

Since the sets of variables in each query are pairwise disjoint, the entailment is only possible if there exists an homomorphism h from the variables in $\bar{z}$ to the variables or constants in the left hand side so that all the atoms in $h(\mu_{\bar{a}}(GP_p))$ appear in the union of the atoms in $\mu_{\bar{a}_1}(GP_{u_1}) \wedge \ldots \wedge \mu_{\bar{a}_n}(GP_{u_n})$.

Let $Frozen$ be the freezing of $\bigcup_{i\in[1..n]} GP_{u_i}$ obtained by replacing each output variable x_{u_i} by $\mu_{\bar{a}_i}(x_{u_i})$. The homomorphism $h \cup \mu_{\bar{a}}$ from the graph pattern GP_p to $Frozen$ allows to show that $\bar{a}$ is an answer of Q_p when evaluated over $Frozen$.

For the converse way of the proof, Let us consider $Frozen$ a freezing of the output variables of $\bigcup_{i\in[1..n]} GP_{u_i}$ such that there exists an answer $\bar{c}$ of Q_p when evaluated over $Frozen$. There exists an homomorphism h from GP_p to $Frozen$ such that $h(\bar{x}) = \bar{c}$. The homomorphism h allows to show the entailment between the formulas $\exists \bar{z}_u\ Frozen$ and $\exists \bar{z}\ h_{\bar{c}}(GP_p)$ where GP_p and $Frozen$ are interpreted as the conjunction of their respective triple patterns seen as logical atoms, and $h_{\bar{c}}$ is the restriction of h to the output variables of Q_p.

In fact, the formula $\exists \bar{z}\ h_{\bar{c}}(GP_p)$ is the logical signature of the answer $\bar{c}$ of Q_p, while the formula $\exists \bar{z}_u\ Frozen$ is the conjunction of logical signatures of the answers $freeze(\bar{x}_{u_i})$ of each Q_{u_i} where $freeze$ is the freezing function applied to the output variables of the different Q_{u_i} to obtain $Frozen$.

Therefore, the privacy query Q_p is incompatible with the utility queries. □

Example 7. Let us consider the following privacy and utility queries:
Q_p: SELECT ?x WHERE ?x p ?y . ?y q ?z
Q_{u_1}: SELECT ?x1 ?y1 WHERE ?x1 p ?y1? . ?y1 r ?z1
Q_{u_2}: SELECT ?x2 WHERE ?x2 q ?y2
The following $Frozen$ and $Frozen'$ are different freezing of the output variables in the union of the utility graph patterns:
$Frozen$ = {c1 p c2 . c2 r ?z1 . c3 q ?y2 }
$Frozen'$ = {c1 p c2 . c2 r ?z1 . c2 q ?y2 }
$Ans(Q_p, Frozen)$ is empty but $Ans(Q_p, Frozen') = \{c1\}$.
This is enough to prove that Q_p is incompatible with the utility policy composed by the two utility queries Q_{u_1} and Q_{u_2}.

Let us replace the utility query Q_{u_1} by the utility query with same graph pattern but with one output variable less:
Q'_{u_1}: SELECT ?x1 WHERE ?x1 p ?y1 . ?y1 r ?z1
No freezing of the output variable ?x1 of Q'_{u_1} combined with a freezing the output variable of Q_{u_2} can lead to an answer of Q_p when evaluated over the resulting graph pattern which is of the form: {c1 p ?y1. ?y1 r ?z1. c2 q ?y2 }
Therefore Q_p is compatible with Q'_{u_1} and Q_{u_2}.

Complexity: In the worst case, checking compatibility of plain conjunctive queries using Theorem 4.1 requires to evaluate the privacy query over the frozen graph patterns resulting from all the possible freezing of the output variables of the utility queries. The evaluation of the privacy query over a frozen graph pattern is polynomial in the size of the utility queries but the number of possible freezing is 2^{OV_u} where OV_u is the number of output variables of the utility queries. As a matter of fact, a freezing can be obtained from the initial freezing, which assigns each output variable

to a distinct fresh constant, by equating a subset of these constants. In practice, the choice of constants to equate is strongly constrained by the joins between variables of the privacy query that are required to make possible the existence of an answer to Q_p.

We end this section by two theorems that provide sufficient conditions of compatibility. Theorem 4.2 states a sufficient condition for compatibility of plain conjunctive queries, while Theorem 4.3 states a sufficient condition of compatibility for general TACQs based on checking their conjunctive parts.

Theorem 4.2 (Sufficient condition of compatibility for conjunctive queries)**.** *When privacy and utility queries are plain conjunctive queries, a privacy query is compatible with utility queries if its graph pattern is disjoint with every graph pattern of the utility queries.*

Proof. If the graph pattern GP_p of Q_p is disjoint with every graph pattern GP_{u_i} of the utility queries, based on Definition 6, there exists a triple pattern in GP_p that cannot be projected by any homomorphism to the union of GP_{u_i}. Based on Definitions 17 and 16, this means that no answer of Q_p can be inferred from answers to Q_{u_i}. □

Theorem 4.3 (Sufficient condition of compatibility for general TACQs)**.** *If the conjunctive part of a privacy query Q_p is compatible with the conjunctive parts of the utility queries then Q_p is compatible with the utility queries.*

Proof. If $Conj(Q_p)$ is compatible with $Conj(Q_{u_1}), ..., Conj(Q_{u_n})$, this means that no answer $\bar{a}$ to $Conj(Q_p)$ can be inferred from answers to $Conj(Q_{u_i})$. Any answer $(\bar{a}, r)$ to Q_p is such that $\bar{a}$ is an answer to $Conj(Q_p)$ and r is the result of an aggregate function. If there is no way to infer the logical signature of $\bar{a}$ from the logical signatures of answers to $Conj(Q_{u_i})$, a fortiori there will be no way to infer a more constrained logical signature of the form: $(\exists y\, \exists \bar{z},\ \mu_{\bar{a}}(GP_p) \wedge \mu_{\bar{a}}(FILTER_p)) \wedge agg\{y | \exists \bar{z},\ \mu_{\bar{a}}(GP_) \wedge \mu_{\bar{a}}(FILTER_p)\} = r$ from logical signatures of answers to Q_{u_i}. □

4.2 Case of conjunctive queries with FILTER conditions

We will use the following notations:

Privacy query Q_p:
SELECT $\bar{x}_p$ WHERE $\{GP_p.\ FILTER_p\}$
Utility query Q_{u_i}:
SELECT $\bar{x}_{u_i}$ WHERE $\{GP_{u_i}\ .\ FILTER_{u_i}\}$

Theorem 4.4 is based on extending the proof of Theorem 4.1 by the verification of the entailment of the FILTER conditions of Q_p by the FILTER conditions of utility queries. When their conjunction is just satisfiable, we obtain a characterization of weak compatibility in Theorem 4.5.

Note that, by definition, weak incompatibility and compatibility does not differ for the cases of queries without FILTER conditions.

Theorem 4.4 (Incompatibility of conjunctive queries with FILTER)**.** *The privacy query Q_p is incompatible with the set of utility queries if and only if there exists a $freeze$ of the output variables in $\bigcup_{i\in[1..n]} GP_{u_i}$, and an answer $\bar{c}$ of the conjunctive part of Q_p over $Frozen = freeze(\bigcup_{i\in[1..n]} GP_{u_i})$ with an homomorphism support h such that:*

$$freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \models h(FILTER_p)$$

Proof. In the case of conjunctive queries with FILTER, the logical signature of an answer $\bar{a}$ to a query Q is a formula of the form: $(\exists \bar{z},\ \mu_{\bar{a}}(GP) \wedge \mu_{\bar{a}}(FILTER))$ where GP is the conjunction of triples patterns in the graph pattern of Q interpreted as logical atoms and μ is an assignment of the tuple $\bar{x}$ of the output variables of Q to the tuple of constants $\bar{a}$, and $\bar{z}$ are the other variables occurring in Q.

If Q_p is incompatible with the utility queries, it means by definition that there exists tuples of constants $\bar{a}, \bar{a}_1, ..., \bar{a}_n$ such that $\exists \bar{z}_1 ... \exists \bar{z}_n\, \mu_{\bar{a}_1}(GP_{u_1}) \wedge \mu_{\bar{a}_1}(FILTER_{u_1}) \wedge ... \wedge \mu_{\bar{a}_n}(GP_{u_n}) \wedge \mu_{\bar{a}_n}(FILTER_{u_n}) \models \exists \bar{z}\ \mu_{\bar{a}}(GP_p) \wedge \mu_{\bar{a}}(FILTER_p)$.

Since the sets of variables in each query are pairwise disjoint, the entailment is only possible if there exists an homomorphism h from the variables in $\bar{z}$ to the variables or constants in the left hand side so that all the atoms in $h(\mu_{\bar{a}}(GP_p))$ appear in the union of the atoms in $\mu_{\bar{a}_1}(GP_{u_1}) \wedge ... \wedge \mu_{\bar{a}_n}(GP_{u_n})$, and $h(\mu_{\bar{a}}(FILTER_p))$ is entailed by $\mu_{\bar{a}_1}(FILTER_{u_1}) \wedge ... \wedge \mu_{\bar{a}_n}(FILTER_{u_n})$.

Let $Frozen$ be the result on $\bigcup_{i\in[1..n]} GP_{u_i}$ of the freezing $freeze$ that replaces each output variable x_{u_i} by $\mu_{\bar{a}_i}(x_{u_i})$. The homomorphism $h \cup \mu_{\bar{a}}$ from the graph pattern GP_p to $Frozen$ allows to show that $\bar{a}$ is an answer of the conjunctive part of Q_p when evaluated over $Frozen$, and: $freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \models h \cup \mu_{\bar{a}}(FILTER_p)$

For the converse way of the proof, Let us consider $Frozen$ the result on $\bigcup_{i\in[1..n]} GP_{u_i}$ of a freezing $freeze$ of the output variables such that there exists an answer $\bar{c}$ of Q_p when evaluated over $Frozen$ with a support homomorphism h such that $h(\bar{x}) = \bar{c}$ and $freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \models h(FILTER_p)$.

The homomorphism h allows to show the entailment between the formulas ϕ_1: $\exists \bar{z}_u\, Frozen \wedge freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i})$ and ϕ_2: $\exists \bar{z}\, h_{\bar{c}}(GP_p) \wedge h_{\bar{c}}(FILTER_p)$ where GP_p and $Frozen$ are interpreted as the conjunction of their respective triple patterns seen as logical atoms, and $h_{\bar{c}}$ is the restriction of h to the output variables of Q_p. .

In fact, ϕ_1 and ϕ_2 are respectively the conjunction of logical signatures of the answers $freeze(\bar{x}_{u_i})$ of each Q_{u_i}, and the logical signature of the answer $\bar{c}$ of Q_p.

Therefore, the privacy query Q_p is incompatible with the utility queries. □

Example 8. Let us consider the following privacy and utility queries, which are obtained by adding FILTER conditions to the queries in Example 7.

QF_p: SELECT ?x WHERE ?x p ?y . ?y q ?z .
FILTER (?y < 6 ∧ ?z > 3)

QF_{u_1}: SELECT ?x1 ?y1 WHERE ?x1 p ?y1? . ?y1 r ?z1.
FILTER (?y1 < 5)

QF_{u_2}: SELECT ?x2 WHERE ?x2 q ?y2. FILTER (?y2 > 10)

In Example 7, we have seen that the conjunctive part of QF_p (the query Q_p in Example 7) has the answer c1 when evaluated over $Frozen'$: {c1 p c2 . c2 r ?z1 . c2 q ?y2 }, obtained by applying the freezing function of the output variables of the utility queries: $freeze$(?x1) = c1 , $freeze$(?y1) = c2 and $freeze$(?x2) = c2.

We get:

$freeze$ ((?y1 < 5) ∧ (?y2 > 10)) = (c2 < 5) ∧ (?y2 > 10)

The homomorphism support h of the answer c1 of QF_p over $Frozen'$ is: h(?x) = c1 , h(?y) = c2 , h(?z) = ?y2.

and h(?y < 6 ∧ ?z > 3) = (c2 < 6 ∧ ?y2 > 3)

Checking that (c2 < 6 ∧ ?y2 > 3) is entailed by (c2 < 5) ∧ (?y2 > 10) is enough to prove that QF_p is incompatible with the utility policy composed by the two utility queries QF_{u_1} and QF_{u_2}.

Let us replace the utility query QF_{u_2} by QF'_{u_2}:

QF'_{u_2}: SELECT ?x2 WHERE ?x2 q ?y2. FILTER (?y2 > 1)

We have to check whether (c2 < 6 ∧ ?y2 > 3) is entailed by (c2 < 5) ∧ (?y2 > 1).

This is not the case and so QF_p is compatible with the utility queries QF_{u_1} and QF'_{u_2}.

However, QF_p is weakly incompatible with the utility queries QF_{u_1} and QF'_{u_2}.

Theorem 4.5 (Weak incompatibility of conjunctive queries with FILTER)**.** *The privacy query Q_p is weakly incompatible with the set of utility queries if and only if there exists a freezing $freeze$ of the output variables in $\bigcup_{i\in[1..n]} GP_{u_i}$, and an answer $\bar{c}$ of the conjunctive part of Q_p over $Frozen = freeze(\bigcup_{i\in[1..n]} GP_{u_i})$ with an homomorphism support h such that:*

$freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \wedge h(FILTER_p)$ *is satisfiable.*

Proof. If Q_p is weakly incompatible with the set of utility queries, Q_p is incompatible with a set of variants Q'_{u_i} of the utility queries Q_{u_i}. By applying Theorem 4.4, there exists a freezing $freeze$ of the output variables in $\bigcup_{i\in[1..n]} GP'_{u_i}$, and an answer $\bar{c}$ of the conjunctive part of Q_p over $Frozen = freeze(\bigcup_{i\in[1..n]} GP'_{u_i})$ with an homomorphism support h such that:

$freeze(\bigwedge_{i\in[1..n]} FILTER'_{u_i}) \models h(FILTER_p)$

This means that every variable assignment satisfying

$freeze(\bigwedge_{i\in[1..n]} FILTER'_{u_i})$

satisfies $h(FILTER_p)$ too.

By definition of the variants, there exists a variable assignment satisfying both

$freeze(\bigwedge_{i\in[1..n]} FILTER'_{u_i})$ and $freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i})$. This assignment satisfies $h(FILTER_p)$ too.

Thus $freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \wedge h(FILTER_p)$ is satisfiable.

For the converse way, let us suppose that there exists a freezing $freeze$ of the output variables in $\bigcup_{i\in[1..n]} GP_{u_i}$, and an answer $\bar{c}$ of the conjunctive part of Q_p over $Frozen = freeze(\bigcup_{i\in[1..n]} GP_{u_i})$ with an homomorphism support h such that:

$freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \wedge h(FILTER_p)$ is satisfiable.

The goal is to build variants Q'_{u_i} of utility queries by adding to the FILTER constraints $FILTER_{u_i}$ some constraints making $h(FILTER_p)$ true.

For doing so, first we remark that each freezing satisfying the conditions of the theorem can be constrained by equating freezing constants for getting a *connected* freezing satisfying also the conditions of the theorem. A freezing is connected if each single GP_{u_i} has a fresh constant in common with the freezing of atleast another GP_{u_j}).

Then:

- for each atomic comparison t comp t' in $h(FILTER_p)$ such that t and t' are either numbers or terms in the freezing of a single GP_{u_i}, we add to $FILTER_{u_i}$ the atomic constraint obtained by defreezing the constants possibly involved in t *comp* t'.

- for each atomic comparison t *comp* t' in $h(FILTER_p)$ such that t and t' are not in the freezing of single GP_{u_i}, we can build a chain of comparisons t_0 *comp* t_1, ...,t_{k-1} *comp* t_k where $t_0 = t$ and $t_k = t'$ where each pair t_j, t_{j+1} are terms appearing in the freezing of single GP_{u_j}. We just have to add to each $FILTER_{u_j}$ the atomic constraint obtained by defreezing the constants possibly involved in t_j *comp* t_{j+1}. □

Theorem 4.6 is the counterpart of Theorem 4.2 for conjunctive queries with FILTER conditions.

Theorem 4.6 (Sufficient condition of compatibility for conjunctive queries with FILTER)**.** *A privacy query Q_p is compatible with utility queries if for each utility query Q_u with a graph pattern GP_u overlapping with GP_p through an unifier overlap:*

$overlap(FILTER_p) \wedge overlap(FILTER_u)$ *is unsatisfiable.*

Proof. If Q_p is incompatible with the utility queries, it means by definition that there exists tuples of constants $\bar{a}, \bar{a}_{i_1}, ..., \bar{a}_{i_k}$ such that :

$\exists \bar{z}_1 ... \exists \bar{z}_k\ \mu_{\bar{a}_{i_1}}(GP_{u_{i_1}}) \wedge \mu_{\bar{a}_{i_1}}(FILTER_{u_{i_1}}) ... \mu_{\bar{a}_{i_k}}(GP_{u_{i_k}})$

$\wedge\ \mu_{\bar{a}_{i_k}}(FILTER_{u_{i_k}}) \models \exists \bar{z}\ \mu_{\bar{a}}(GP_p) \wedge \mu_{\bar{a}}(FILTER_p)$

where $GP_{u_{i_j}}$ are the graph patterns of utility queries overlapping with the graph pattern GP_p of Q_p.

Thus, there exists an homomorphism h from $\mu_{\bar{a}}(GP_p)$ to $\mu_{\bar{a}_{i_1}}(GP_{u_{i_1}}) \wedge ... \wedge \mu_{\bar{a}_{i_k}}(GP_{u_{i_k}})$ such that every model satisfying $h(\mu_{\bar{a}_{i_1}}(FILTER_{ui_1})) \wedge ... h(\mu_{\bar{a}_{i_k}}(FILTER_{u_{i_k}}))$, and in particular each $overlap(h(\mu_{\bar{a}_{i_1}}(FILTER_{u_{i_j}})))$, also satisfies

$\mu_{\bar{a}}(FILTER_p)$, and in particular $overlap(\mu_{\bar{a}}(FILTER_p))$, which contradicts the if condition of the theorem. □

Example 9. Let us consider the following privacy and utility queries, which are obtained by modifying the FILTER conditions to the queries in Example 9.

QF'_p: SELECT ?x WHERE ?x p ?y . ?y q ?z .
FILTER (?y < 6 ∧ ?z > 3)

QF'_{u_1}: SELECT ?x1 ?y1 WHERE ?x1 p ?y1? . ?y1 r ?z1.
FILTER (?y1 > 8)

QF'_{u_2}: SELECT ?x2 WHERE ?x2 q ?y2. FILTER (?y2 < 2)

An overlapping of QF'_p with the graph pattern of QF'_{u_1} is based on unifying the triple patterns ?x p ?y and ?x1 p ?y1? with the unifier $overlap$(?x1) = ?x and $overlap$(?y1) = ?y
$overlap$(?y < 6 ∧ ?z > 3) ∧ $overlap$(?y1 >8)
= ?y < 6 ∧ ?z > 3 ∧ ?y > 8
which is unsatisfiable.

Similarly, the overlap of QF'_p with the graph pattern of QF'_{u_2} based on the triple patterns ?y q ?z and ?x2 q ?y2 leads to the unsatisfiability of: ?y < 6 ∧ ?z > 3 ∧ ?z < 2.

This makes Q_p compatible with the utility queries QF'_{u_1} and QF'_{u_2}.

4.3 Case of aggregates with same time windows

From now, based on Theorem 4.3, we aim at conditions for a privacy query with aggregate to be compatible or incompatible with utility queries with aggregate in the case where their conjunctive parts are incompatible.

Since the aggregate values are computed based on groups that are specific to each query, if the conjunctive part $Conj(Q_p)$ of a privacy query Q_p is compatible with the conjunctive part $Conj(Q_{u_i})$ of *each utility query* Q_{u_i}, there is no way to infer the aggregate value r of an answer $(\bar{a}, r)$ of Q_p from the union of answer sets of utility queries with aggregates.

Therefore we focus now on the compatibility between one privacy query Q_p and one utility query Q_u such that $Conj(Q_p)$ is incompatible with $Conj(Q_u)$.

We will use the following notations:

Privacy query Q_p*:*
SELECT $\bar{x}_p$ WHERE $\{GP_p.\ FILTER_p\}$
GROUP BY $\bar{x}_p$ TIMEWINDOW $(Size_p, Step_p)$

Utility query Q_u*:*
SELECT $\bar{x}_u$ WHERE $\{GP_u\ .\ FILTER_u\}$
GROUP BY $\bar{x}_u$ TIMEWINDOW $(Size_u, Step_u)$

By definition, the answer set of a query Q_{window}:
SELECT $\bar{x}$ WHERE $\{GP.\ FILTER\}$
GROUP BY $\bar{x}$ TIMEWINDOW $(Size, Step)$
is the union of the answer sets resulting of the evaluation over each time window of Q_{window} of the query Q:
SELECT $\bar{x}$ WHERE $\{GP.\ FILTER\}$ GROUP BY $\bar{x}$

Thus, in this section, we focus on studying the compatibility between Q_p and Q_u evaluated on a common time window, e.g., the first time windows of Q_p and Q_u ($k_u = k_p = 0$) when $Size_u = Size_p$.

Theorem 4.7 provides conditions for incompatibility of utility and privacy queries with aggregates.

Theorem 4.7 (Incompatibility conditions for aggregate queries)**.** *A privacy query Q_p is incompatible with the utility query Q_u if there exists a (possibly empty) freezing f_p of output variables in GP_p with constants of GP_u, or a (possibly empty) freezing f_u of output variables in GP_u with constants in GP_p such that $f_p(GP_p)$ and $f_u(GP_u)$ are isomorphic.*

When Q_p and Q_u have no FILTER conditions and the same aggregate functions, they are incompatible if and only if the above condition is satisfied.

Proof. Based on Definition 16, an answer $(\bar{a}, r)$ of an aggregate query Q_p can be inferred from a set of answers $\{(\bar{a}_u, r_u)\}$ only if the group $Group_p(\bar{a}) = \{y_p | \exists \bar{z},\ \mu_{\bar{a}}(GP_p) \wedge \mu_{\bar{a}}(FILTER_p)\}$ can be obtained as a group, or the union of groups, of Q_u, i.e., unions of $\{y_u | \exists \bar{z}_u,\ \mu_{\bar{a}_u}(GP_u) \wedge \mu_{\bar{a}_u}(FILTER_u)\}$.

Based on [4], this situation is true only if (if and only if, when there is no FILTER conditions) either $\mu_{\bar{a}}(GP_p)$ and (GP_u) are isomorphic, or if there exists an answer $\bar{a}_u$ of GP_u such that $\mu_{\bar{a}}(GP_p)$ and $\mu_{\bar{a}_u}(GP_u)$ are isomorphic, i.e, GP_p (or one of its freezing of output variables by constants in GP_u) is isomorphic to GP_u (or to one of its freezing of output variables by constants in GP_p). □

Example 10. Let us consider the following privacy and utility queries:

Q_p:
SELECT ?city AVG(?s)
WHERE {?dp city ?city . ?dp street ?street . ?dp familySize ?s}
GROUP BY ?city

Q_u:
SELECT ?street', AVG(?s')
WHERE {?dp' city "Grenoble" . ?dp' street ?street' .
?dp' familySize ?s'}
GROUP BY ?street'

Freezing the ?city variable of Q_p with the constant "Grenoble" in Q_u results in:
$Frozen$ = {?dp city "Grenoble" . ?dp street ?street . ?dp familySize ?s}
which is isomorphic with the graph pattern of Q_u.

In this case, the group $Group$("$Grenoble$") of Q_p can be obtained as the union of the groups of Q_u on all the values of ?$street'$ resulting from the evaluation of Q_u.

Since, in addition, the aggregate function is the same in Q_p and Q_u, Q_p is thus incompatible with Q_u .

Let us consider now Q'_p:
SELECT ?city AVG(?s)
WHERE {?dp city ?city . ?dp street "Alsace Lorraine" . ?dp familySize ?s}

GROUP BY ?city

Q'_p is also incompatible with Q_u since freezing the variable ?street' of Q_u with "Alsace Lorraine" and the variable ?city of Q_p with "Grenoble" makes the resulting graph patterns isomorphic, and the group $Group(\text{"}Grenoble\text{"})$ of Q'_p can be obtained as the group $Group(\text{"}Alsace\ Lorraine\text{"})$ of Q_u.

4.4 Case of aggregate with different time windows

In the general case of TACQ with explicit TIMEWINDOW clause, a privacy query Q_p and a utility query Q_u are compatible if no time window of Q_p can be built by unions and/or differences of time windows of one or more Q_u. A sufficient condition is that for any Q_u no boundary of a time window of Q_u should correspond to a boundary of a time window of Q_p, except of course the first one (i.e. $k_u = k_p = 0$) where the ends of time windows is *now*.

Theorem 4.8 (Sufficient condition of compatibility)**.** *A privacy query Q_p is compatible with a set of utility queries if there is no utility query Q_u such that:*

$\exists k_u \in \mathbb{N}\ \exists k_p \in \mathbb{N}, k_u \times Step_u - Size_u = k_p \times Step_p - Size_p$
$\vee\ \exists k_u \in \mathbb{N}^+\ \exists k_p \in \mathbb{N}^+, k_u \times Step_u = k_p \times Step_p$
$\vee\ \exists k_u \in \mathbb{N}\ \exists k_p \in \mathbb{N}^+, k_u \times Step_u - Size_u = k_p \times Step_p$
$\vee\ \exists k_u \in \mathbb{N}^+\ \exists k_p \in \mathbb{N}, k_u \times Step_u = k_p \times Step_p - Size_p$

Proof. Consider a privacy query Q_p and a set of utility queries Q_{u_1}, ... Q_{u_n}. As the answer of a TACQ Q with an explicit TIMEWINDOW clause is the union of the answers of its subquery Q' without the TIMEWINDOW clause evaluated over each time window of Q, aggregates on a group a Q_p can be computed from aggregates of groups of one or more Q_{u_i} if and only if a time window of Q_p can be built from the time windows of considered Q_{u_i}.

Let consider first only two time windows of one or two Q_{u_i}. More complex combinations can easily be composed from this simple case.

Building a time window T_p from two time windows T_{u_1} and T_{u_2} can be done only in two ways:

- either $T_p = T_{u_1} \cup T_{u_2}$
- either $T_p = T_{u_1} - T_{u_2}$

In the first case, assuming that T_{u_1} starts before T_{u_2} then T_{u_1} has to start at the same time than T_p and T_{u_2} has to end at the same time than T_p.

If there is no Q_{u_i} such that $\exists k_u \in \mathbb{N}\ \exists k_p \in \mathbb{N}$, $k_u \times Step_u - Size_u = k_p \times Step_p - Size_p \vee \exists k_u \in \mathbb{N}^+\ \exists k_p \in \mathbb{N}^+$, $k_u \times Step_u = k_p \times Step_p$, meaning that no time window of a Q_{u_i} starts at the same time than a time window of Q_p and that no time window of a Q_{u_i} ends at the same time than a time window of Q_p, the building of a time window of Q_p by union is impossible.

In the second case, either T_{u_1} and T_{u_2} start as the same time and T_{u_1} ends at the same time than T_p and T_{u_2} ends when T_p starts, either T_{u_1} and T_{u_2} end at the same time and T_{u_1} starts at the same time than T_p and T_{u_2} starts when T_p ends.

If there is no Q_{u_i} such that $\exists k_u \in \mathbb{N}\ \exists k_p \in \mathbb{N}^+$, $k_u \times Step_u - Size_u = k_p \times Step_p \vee \exists k_u \in \mathbb{N}^+\ \exists k_p \in \mathbb{N}$, $k_u \times Step_u = k_p \times Step_p - Size_p$, meaning than no time window of a Q_{u_i} ends when a time window of Q_p starts and that no time window of a Q_{u_i} starts when a time window of Q_p ends, the building of a time window of Q_p by difference is impossible.

Therefore no time window of Q_p can be built from time windows of one or more Q_{u_i} and no aggregate of Q_p can be computed from aggregates of one or more Q_{u_i}, and thus Q_p is compatible with the utility queries. □

If it is possible to build a time window of a privacy query from time windows of one or more utility queries, Theorems 4.7 can be applied to check compatibility.

Example 11. Let us consider $TACQ_2$ of Example 3 as privacy query and the following utility queries:

Q_{u_1}: SELECT ?timeWindowEnd, SUM(?cons)
WHERE {((?dp consumption ?cons), ?ts)}
GROUP BY ?timeWindowEnd
TIMEWINDOW (7h, 2h)

Q_{u_2}: SELECT ?timeWindowEnd, SUM(?cons')
WHERE {((?dp' consumption ?cons'), ?ts')}
GROUP BY ?timeWindowEnd
TIMEWINDOW (1h, 8h)

We have $0 \times Step_{u_1} - Size_{u_1} = 1 \times Step_p - Size_p$ meaning that the first interval $I_{u_1}(k_{u_1} = 0)$ of Q_{u_1} starts at the same time than the second interval $I_p(k_p = 1)$ of $TACQ_2$ at $\mathsf{now} - 7h$.

We have also $0 \times Step_{u_2} - Size_{u_2} = 1 \times Step_p$ meaning that the first interval $I_{u_2}(k_{u_2} = 0)$ of Q_{u_2} starts at the end of the second interval $I_p(k_p = 1)$ of $TACQ_2$ at $\mathsf{now} - 1h$.

This example does not satisfy the sufficient condition of Theorem 4.8 for both Q_{u_1} and Q_{u_2}. In fact, as $I_p(K_p = 1) = I_{u_1}(k_{u_1} = 0) - I_{u_2}(k_{u_2} = 0)$, the sum of consumption of $TACQ_2$ for its second interval can be computed in the following way:

- $Sum_{u_1} = SUM(?cons)$ in $I_{u_1}(k_{u_1} = 0)$
- $Sum_{u_2} = SUM(?cons')$ in $I_{u_2}(k_{u_2} = 0)$
- $Sum_p = SUM(?consuption)$ in $I_p(k_p = 1) = Sum_{u_1} - Sum_{u_2}$

5 Related work

Privacy preserving data publishing has been a long-standing research goal for several research communities, as witnessed by a flurry of work on the topic [8]. A rich variety of privacy models have been proposed, ranging from k-anonymity [15] and l-diversity [13] to t-closeness [12] and ϵ-differential privacy [7]. Compared to our work, all these approaches are based on changing the exposed data either by adding noise in the data or by applying generalization operations on sensitive data.

Data security is also an important topic for which secure protocols based on encryption has been proposed that enable to do some computations on encrypted outsourced data. In contrast with our work, each protocol may be specific to the target computations to be feasible in practice like in [3].

An alternative approach for protecting against privacy breaches consists in applying access control methods to RDF data ([11, 14, 16]). In the Semantic Web setting, when data are described by description logics ontologies, preliminary results on role-based access control have been obtained in [2] for the problem of checking whether a sequence of role changes and queries can infer that an anonymous individual is equal to a known individual. However, all these works do not handle utility queries.

A query-based logical framework for RDF data has been introduced in [9, 10], where sensitive information is expressed as a privacy policy in the form of SPARQL query whose results must not disclose sensitive information of individual. It has been extended to handling utility queries in [5, 6]. These approaches however are restricted to privacy and utility policies that are simple conjunctive queries.

6 Conclusion and future work

In this paper we have proposed a query-based declarative framework for a formal specification and verification of privacy and utility policies expressed as temporal aggregate conjunctive queries.

We do think that this framework is well suited for guaranteeing data producers to keep the control and protect their data in many real-world situations where sensitive data are collected by mobile personal devices or smart environments.

Based on the implementation of this framework, we plan to design and implement a negotiation mechanism that will be triggered when a utility policy turns out to be incompatible with a privacy policy. New relaxed utility queries will be automatically computed to restore compatibility with the privacy policy of a given data producer. They will be the formal basis of a dialogue between each data producer and the service provider in order to find a trade-off acceptable in terms of utility while guaranteeing privacy preservation for each data producer.

We also plan to extend our framework to take into account ontological knowledge in the possible inference of answers of privacy queries by answers of utility queries. This will bring stronger constraints on compatibility between privacy and utility policies.

7 Acknowledgments

This work has been partially supported by MIAI@Grenoble Alpes (ANR-19-P3IA-0003), PERSYVAL-Lab (ANR-11-LABX-0025-01) and TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

[1] Tristan Allard, Nicolas Anciaux, Luc Bouganim, Yanli Guo, Lionel Le Folgoc, Benjamin Nguyen, Philippe Pucheral, Indrajit Ray, Indrakshi Ray, and Shaoyi Yin. 2010. Secure personal data servers: a vision paper. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 25–35.

[2] Franz Baader, Daniel Borchmann, and Adrian Nuradiansyah. 2017. Preliminary Results on the Identity Problem in Description Logic Ontologies. In *Description Logics (CEUR Workshop Proceedings, Vol. 1879)*. CEUR-WS.org.

[3] Radu Ciucanu and Pascal Lafourcade. 2020. GOOSE: A Secure Framework for Graph Outsourcing and SPARQL Evaluation.. In *IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec)*. 347–366.

[4] Sara Cohen. 2005. Containment of aggregate queries. *ACM SIGMOD Record* 34, 1 (2005), 77–85.

[5] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset, and Romuald Thion. 2018. Query-Based Linked Data Anonymization. In *The Semantic Web-ISWC 2018* (Monterey, California, United States). Springer, Cham, 530–546. https://doi.org/10.1007/978-3-030-00671-6_31

[6] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset, and Romuald Thion. 2019. RDF graph anonymization robust to data linkage. In *Proceedings of WISE 2019 (20th International Conference on Web Information Systems Engineering)*.

[7] Cynthia Dwork. 2006. Differential Privacy. In *ICALP (2) (LNCS, Vol. 4052)*. Springer, 1–12.

[8] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.* 42, 4 (2010), 14:1–14:53.

[9] Bernardo C. Grau and Egor V. Kostylev. 2016. Logical Foundations of Privacy-Preserving Publishing of Linked Data. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona USA). The AAAI Press, Palo Alto, California, 943–949. https://doi.org/10.5555/3015812.3015953

[10] Bernardo C. Grau and Egor V. Kostylev. 2019. Logical Foundations of Linked Data Anonymisation. *Journal of Artificial Intelligence Research* 64 (2019), 253–314.

[11] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. 2017. Access control and the Resource Description Framework: A survey. *Semantic Web* 8, 2 (2017), 311–352.

[12] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *ICDE*. IEEE Computer Society, 106–115.

[13] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. *L*-diversity: Privacy beyond *k*-anonymity. *TKDD* 1, 1 (2007), 3.

[14] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens, and Stephane Morucci. 2012. Privacy Policy Preferences Enforced by SPARQL Query Rewriting. In *ARES*. IEEE Computer Society, 335–342.

[15] Latanya Sweeney. 2002. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 5 (2002), 557–570.

[16] Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard. 2011. An Access Control Model for Linked Data. In *OTM Workshops (LNCS, Vol. 7046)*. Springer, 454–463.