Identifying Privacy Risks raised by Utility Queries *

Hira Asghar, Christophe Bobineau, and Marie-Christine Rousset

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

Abstract. Personal data are increasingly disseminated over the Web 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 to remain undisclosed by data producers. It is therefore of utmost importance to help them to identify privacy risks raised by requests of service providers for utility purposes. In this paper, we first formalize privacy risks by *privacy queries* expressed (and kept secret) by each data producer to specify the data they do not want to be disclosed. Then, we develop a formal approach for detecting incompatibility between privacy and utility queries expressed as *temporal aggregate conjunctive queries*. The distinguishing point of our approach is to be data-independent and to come with an *explanation* based on the query expressions only. This explanation is intended to help data producers understand the detected privacy breaches and guide their choice of the appropriate technique to correct it.

Keywords: Temporal aggregated conjunctive queries · Utility queries · Privacy queries

1 Introduction

Personal data are increasingly disseminated over the 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 request collecting data from data producers in order to perform aggregate data analytics for optimization or

* 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

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

- Data producers specify by a set of *privacy queries* (kept secret) the (possibly aggregated) data that they do not want to disclose.

- Data consumers make explicit by a set of *utility queries* the data that they request from each data producer for offering them services in return.

- The compatibility between privacy and utility queries is automatically verified, and in case of incompatibility data producers get an explanation that can be exploited later to help them find an acceptable privacy-utility trade-off.

Our contribution is twofold. First, we handle the temporal aspect in the definition of privacy and utility by expressing privacy and utility queries as *temporal aggregate queries*. Taking into account the temporal aspect for privacy protection is very important since many applications handle dynamic data (e.g., electrical consumption, time series, mobility data) for which temporal data are considered as sensitive and aggregates on time are important for data analytics. Second, we formally define and *automatically verify incompatibility between privacy and utility queries based on their query expressions only*, and thus independently of the data. Incompatibility proofs come with explanations which can be exploited subsequently for helping data producers choose an appropriate defence strategy while limiting the utility loss. This can be done by applying to their data existing anonymization techniques based on differential privacy [9] or k-anonymity [14].

The paper is organized as follows. Section 2 provides an illustrative scenario of our approach. Section 3 formally defines the queries that we consider. Section 4 is dedicated to checking incompatibility between privacy and utility queries. Section 5 presents related work, and Section 6 concludes our paper.

2 Illustrative Scenario

We consider a use-case related to smart power grids, in which the data producers are customers with smart meters in their home. A service provider has a catalog of energy efficiency products (including energy efficient insulation, windows, appliances) and requests collecting data from all the customers to adapt the proposed services to the profile of each of them based on some personal data.

We assume that the service provider and the customers understand each other through a common vocabulary using a a simple ontology such as the one [1] that we have extracted from the ISSDA dataset, a real world power grid dataset provided by the *Irish Social Science Data Archive (ISSDA) Commission for Energy Regulation (CER)*[2]. This shared vocabulary allows service providers to specify their data needs in a precise way through a set of *utility* queries, that can then be compared to a set of *privacy* queries used by each data producer to state the data that they not want to be disclosed directly or indirectly.

Let us suppose that the service provider has the following data needs:

[1] available at https://raw.githubusercontent.com/fr-anonymous/puck/main/issda_schema.ttl

[2] https://www.ucd.ie/issda/data/commissionforenergyregulationcer/

(1) for each identifier of customers that are owners of their home, their yearly income if it is more than 75000;

(2) for each identifier of customers, their smart meter number;

(3) for each smart meter number, the sum of consumptions computed every hour over the meter readings of the previous 3 hours.

These needs can be translated into the utility queries shown below by using SPARQL-like query language that will be explained in Section 3.

The utility queries into SPARQL-like query language

```
UQ1: SELECT ?o ?y
     WHERE { ?o issda:yearlyIncome ?y . ?o issda:own ?s.
     FILTER(?y > 75000)}
UQ2: SELECT ?sm ?o
     WHERE { ?sm issda:associatedOccupier ?o .
     ?o issda:nbOfPersons ?n }
UQ3: SELECT ?sm  ?timeWindowEnd SUM(?c)
     WHERE {(?sm issda:consumption ?c, ?ts)}
     GROUP BY ?sm ?timeWindowEnd
     TIMEWINDOW (3h, 1h)
```

Now, suppose that a given customer, possibly with the help of privacy officer or tool, states that, among the data they accept to transmit, they want to prevent:

- the association between their smart meter number and their yearly income;
- the disclosure of their energy consumption measurements aggregated over intervals of 6 hours.

This can be translated into the following privacy queries for which no answer should be transmitted or inferred by any external data consumer.

The privacy queries of a given customer

```
PQ1: SELECT ?sm ?y
     WHERE {?sm issda:associatedOccupier ?o .
     ?o issda:yearlyIncome ?y}
PQ2: SELECT  ?timeWindowEnd SUM(?c)
     WHERE {(?sm issda:consumption ?c , ?ts)}
     GROUP BY  ?timeWindowEnd
     TIMEWINDOW (6h, 6h)
```

With our approach, as it will be explained in Section 4, we can automatically detect that the above utility and privacy queries are incompatible, and provide the following privacy diagnosis:

1) The first privacy risk is due to the possible violation of the privacy query PQ1 by the combination of answers to the utility queries UQ1 and UQ2.

2) The second privacy risk is due to the possible violation of the privacy query PQ2 by answers to the utility query UQ3 because:

a) PQ2 and UQ3 compute the same aggregate under the same conditions;

b) groups of UQ3 are partitions of groups of PQ2;

c) and finally, all time windows of PQ2 can be obtained as disjoint unions of some time windows of UQ3.

Based on the above explanations, the data producer could, for example, inform the service provider that:

- they will refuse to answer at least one the utility queries UQ1 or UQ2;
- they could accept to answer UQ3 if the time window is changed , for instance by modifying the step between each consumption computation.

3 Formal Background

We first define *temporal aggregate conjunctive queries* (TACQ) with a SPARQL-like syntax extended with time windows for capturing aggregate on time. Then, we define the compatibility problem between utility and privacy TACQs.
To simplify the exposition, we will only consider aggregate queries with 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 [2]. In particular, AVG can be computed by the union of two queries, one for computing SUM and the other one for computing $COUNT$.
Definition 1 relies on the notion of *temporal graph pattern* that is an extension of the standard notion of graph pattern in SPARQL in which we allow to associate *timestamp variables* to triple patterns involving *dynamic* properties. Through homomorphisms from temporal graph patterns to temporal data graphs, timestamp variables can only be assigned to timestamps appearing in timestamped RDF triples instantiating the dynamic properties.

Definition 1 (Temporal aggregated conjunctive query). *A $TACQ$ is:*
$SELECT\ \bar{x},\ agg(y)$
$WHERE\ \{GP\ .\ FILTER\}$
$GROUP\ BY\ \bar{x}$
$TIMEWINDOW\ (Size,\ Step)$
where:

- *GP is a temporal graph pattern;*
- *$FILTER$ is a conjunction of atomic comparisons of the form $t\ \theta\ t'$ where t and t' are variables of GP or literals (numbers, strings or dates) and $\theta \in \{<>, <, <=, =, >=, >\}$;*
- *$\bar{x}$ 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 $Size = \infty$ ($Step = 0$), 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 $?ts$:

$$?ts \leq ?timeWindowEnd \wedge ?ts > ?timeWindowsEnd - Size.$$

where $?timeWindowEnd$ is a specific timestamp variable that will be mapped successively to the upper bound of each time window computed from the timestamp at which the query is executed.
Given a TACQ defined in Definition 1, its evaluation over a temporal data graph G is defined in terms of filtered homomorphisms (Definition 2) and groups (Definition 3) for obtaining its answer set.

Definition 2 (Filtered homomorphisms). *Let M be the set of homomorphisms from GP to G. The filtered set of homomorphisms is the subset of M of homomorphisms μ such that $\mu(FILTER) =$ `TRUE`.*

Definition 3 states that there are as many groups as filtered 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 3 (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 $\mu(?ts) \in]now - k \times Step - Size, now - k \times Step])$ and $\mu(?timeWindowEnd) = now - k \times Step\}$.*

It is important to note that if there is no aggregate term, there is only one time window (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.
The answer set of a query Q_{window}:

SELECT $\bar{x}$ $agg(y)$ 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 $\bar{x}$ $agg(y)$ WHERE $\{GP.\ FILTER\}$ GROUP BY $\bar{x}$

By interpreting GP as the logical conjunction of atomic formulas, Definition 4 defines the logical signature of an answer to a query as the logical formula characterizing all the (unknown) temporal data graphs leading to this answer for this query.

Definition 4 (Logical signature of answers). *For an answer $(\bar{a}, r)$ to a 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 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 signatures of the respective answers `(sm1031, 75000)` and `(sm1031, 2020-07-14T03:30:00, 88)` to the (simple conjunctive) privacy query PQ1 and the TACQ utility query UQ3 used in the scenario of Section 2, are given below:

Logical signature of the answer (sm1031, 75000) of PQ1

$\exists ?o,$ `sm1031 issda:associatedOccupier` $?o$
$\wedge$ $?o$ `issda:yearlyIncome 75000`

Logical signature of the answer ($sm1031$, 2020-07-14T03:30:00, 88) of UQ3

$\exists ?c\ \exists ?ts$ `(sm1031 issda:consumption` $?c$,$?ts$`)`
$\wedge$ $?ts \leq$ `2020-07-14T03:30:00` $\wedge$ $?ts >$ `2020-07-14T00:30:00`
$\wedge$ `SUM {`$?c$ | $\exists ?ts$ `(sm1031 issda:consumption` $?c$, $?ts$`)`
$\wedge$ $?ts \leq$ `2020-07-14T03:30:00` $\wedge$ $?ts >$ `2020-07-14T00:30:00 } = 88`

Definition 5 formalizes incompatibility as the possibility of inferring answers of a privacy query from answers of utility queries on the same data graph *without necessarily knowing it.*

Definition 5 (Incompatibility between privacy and utility queries). *A privacy query is incompatible with a set of utility queries if the logical signature of an answer to the privacy query is entailed by a conjunction of logical signatures of answers to some utility queries.*

4 Incompatibility Detection

The privacy queries are specific to, and kept secret by, each data producer. Then, the detection of incompatility is done by algorithms launched by each data producer, given the set of utility queries they receive from a data consumer.
In this section, we provide a characterization of incompatibility by distinguishing the cases where privacy queries are without aggregate (Theorem 1) or with aggregate (Theorem 2 , Theorem 3 and Theorem 4).
Without loss of generality, by renaming variables within each query, we consider that queries have no variable in common. We will use the following notations:

Privacy query Q_p*:*
SELECT $\bar{x}_p$ $agg_p(y_p)$ WHERE $\{GP_p.\ FILTER_p\}$
GROUP BY $\bar{x}_p$ TIMEWINDOW ($Size_p$, $Step_p$)
Utility query Q_{u_i} *:*
SELECT $\bar{x}_{u_i}$ $agg_{u_i}(y_{u_i})$ WHERE $\{GP_{u_i}\ .\ FILTER_{u_i}\}$
GROUP BY $\bar{x}_{u_i}$ TIMEWINDOW ($Size_{u_i}$, $Step_{u_i}$)

For a TACQ in its general form, we will often rely on its *conjunctive part*, and also on its *plain variant* without aggregate.

Definition 6 (Conjunctive part and plain variant of a query). *Let Q be 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 simple conjunctive query:

$Conj(Q)$: SELECT $\bar{x}$ WHERE $\{GP\}$

The plain variant of Q, noted $Plain(Q)$, is the query without aggregate that is evaluated before computing the aggregate function:

$Plain(Q)$: SELECT $\bar{x}$ y WHERE $\{GP\ .\ FILTER\}$

Theorem 1 provides a full characterization of incompatibility of a privacy query Q_p *without aggregate* and a set of n utility queries Q_{u_1}, ..., Q_{u_n}. It relies on evaluating the privacy query on all the (small) data graphs that are representative of the different ways of joining answers of utility queries. Each of these data graph is obtained by *freezing* (Definition 7) the variables in the union of graph patterns in the utility queries, in a way that allows to replace distinct output variables with a same constant (in order to mimic possible joins between output variables coming from different utility queries).

Definition 7 (Freezing of graph patterns). *Let GP a temporal graph pattern and X a subset of its variables. 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, such that every variable in X that is not an output variable is replaced by a distinct constant.*

Theorem 1 (Incompatibility of privacy queries without aggregate). *A privacy query Q_p without aggregate is* incompatible *with a set of utility queries Q_{u_1}, ..., Q_{u_n}* if and only if *there exists a $freeze$ of the variables in $\bigcup_{i\in[1..n]} GP_{u_i}$, and an answer $\bar{c} = h(\bar{x})$ of the conjunctive part of Q_p over $freeze(\bigcup_{i\in[1..n]} GP_{u_i})$ and* if Q_p has a FILTER condition*: $freeze(\bigwedge_{i\in[1..n]} FILTER_{u_i}) \models h(FILTER_p)$*

Proof. 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 1. Let us consider the following privacy and utility queries PQ_1, UQ_1 and UQ_2, corresponding respectively to the first privacy query and the first two utility queries (up to variable renaming) of the scenario illustrated in Section 2:
PQ_1: SELECT ?sm ?y WHERE {?sm issda:associatedOccupier ?o .
?o issda:yearlyIncome ?y}
UQ_1: SELECT ?x1 ?y1 WHERE { ?x1 issda:yearlyIncome ?y1 .
?x1 issda:owns ?z1 . FILTER (?y1 > 75000)}
UQ_2: SELECT ?x2 ?y2
WHERE { ?x2 issda:associatedOccupier ?y2. ?y2 issda:nbOfPersons ?n }
The following $Frozen$ and $Frozen'$ are different freezing of the variables in the union of the utility graph patterns, where the constants corresponding to the freezing of output variables are denoted by constants oc_i:
$Frozen$ = {oc_1 issda:yearlyIncome oc_2 . oc_1 issda:owns c_3.
oc_4 issda:associatedOccupier oc_5 . oc_5 issda:nbOfPersons c_6}
obtained by the freezing: { ?x1/oc_1, ?y1/oc_2, ?z1/c_3, ?x2/oc_4, ?y2/oc_5, ?n/c_6}
$Frozen'$ = {oc_1 issda:yearlyIncome oc_2 . oc_1 issda:owns c_3 .
oc_4 issda:associatedOccupier oc_1 . oc_1 issda:nbOfPersons c_6}
obtained by a freezing in which ?x1 and ?y2 that are output variables in each of the utility queries are frozen to the same constant oc_1.

$Ans(PQ_1, Frozen)$ is empty but $Ans(PQ_1, Frozen') = \{(oc_4, oc_2)\}$.
This proves that PQ_1 is incompatible with the utility policy composed by the two utility queries UQ_1 and UQ_2. The corresponding freezing allows to exhibit the encountered privacy risk: an answer (for instance (oc_4, oc_2)) to the privacy query PQ_1 is disclosed each time answering UQ_1 and UQ_2 return answers where the first element of an answer to UQ_1 is equal to the second element of an answer to UQ_2 (for instance (oc_1, oc_2) and (oc_4, oc_1)) , thus violating any privacy policy containing PQ_1. It is important to note that PQ_1 is not incompatible with each of the utility queries taken in isolation.

Worst-case complexity: In the worst case, checking compatibility using Theorem 1 requires to evaluate the conjunctive part of 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 conjunctive part of 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. Handling privacy queries with FILTER conditions requires to check in addition logical entailment of conjunction of comparison atoms, which can be done using a constraint solver.

In practice: In fact, a freezing can be obtained from the initial most general freezing, which assigns each output variable to a distinct fresh constant, by equating a subset of these constants. The choice of constants to equate is constrained by the join variables within the privacy query to obtain an answer. We have exploited those constraints in our implementation[3].

We consider now the case of checking incompatibility of a *privacy query with aggregate* with a set of utility queries. In this case, we have to consider separately the utility queries with aggregates from the utility queries without aggregate. First, we check the incompatibility of the privacy query with the set of the *utility queries without aggregate*. For doing so, since groups are computed from the results obtained by evaluating first the query without aggregate, we just have to check incompatibility of the plain variant of Q_p (see Definition 6) with the utility queries without aggregate. This can be done using Theorem 1. Then, we check the incompatibility of the privacy query with the *utility queries with aggregate*.

Theorem 2 establishes the results when aggregates in the privacy query and a given utility query are defined over the same time window.

Theorem 2 (Incompatibility with aggregates). *Let us consider a privacy query Q_p and an utility query Q_u with the same aggregate function on a same time window. Q_p is incompatible with 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, they are incompatible if and only if the above condition is satisfied.*

Proof. Based on Definition 4, 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 [2], 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 2. Let us consider the following privacy query Q_p and the following utility query Q_u:

[3] Our code is available at https://github.com/fr-anonymous/puck

Q_p: SELECT ?building AVG(?nb)
WHERE {?sm issda:associatedBuilding ?b . ?b rdf:type ?building .
?sm issda:associatedOccupier ?o . ?o issda:nbOfPersons ?nb .
?o issda:yearlyIncome ?y}
GROUP BY ?building
Q_u: SELECT ?y' AVG(?nb')
WHERE {?sm' issda:associatedBuilding ?b' . ?b' rdf:type issda:Apartment.
?sm' issda:associatedOccupier ?o' . ?o' issda:nbOfPersons ?nb' .
?o' issda:yearlyIncome ?y'}
GROUP BY ?y'

Freezing the ?building variable of Q_p with the constant issda:Apartment appearing in Q_u results in:

$Frozen$ = {?sm issda:associatedBuilding ?b . ?b rdf:type issda:Apartment .
?sm issda:associatedOccupier ?o . ?o issda:nbOfPersons ?nb .
?o issda:yearlyIncome ?y}

which is isomorphic with the graph pattern of Q_u. Since, Q_p and Q_u have the same aggregate function, Q_p is incompatible with Q_u.

Let us consider now the case where Q_p and Q_u have *different time windows*: we have to check if time windows of Q_p can be built from time windows of Q_u. Computing an aggregate on a time window I_1 from the results of the same aggregate on different time windows I_2 and I_3 is only possible if I_1 can be built from I_2 and I_3 by *union* or *difference*. As shown in Figure 1, I_1 can be obtained by union of I_2 and I_3, but I_3 can also be obtained by difference between I_1 and I_2, and I_2 can be built by difference between I_1 and I_3. Therefore, testing if a time window can be built by difference amounts to testing if an union is possible. From now on, we thus focus, without loss of generality, on checking incompatibility by testing whether time windows of a privacy query can be built as unions of time windows of utility queries.

Theorem 3 provides a characterization of the incompatibiity between a privacy query Q_p and a single utility query Q_u based on the values of $Size_p$, $Step_p$, $Size_u$ and $Step_u$ defining the time windows considered in Q_p and Q_u respectively. It relies on identifying how to build a time window of Q_p as union of some time windows of Q_u, as illustrated in Figure 2.

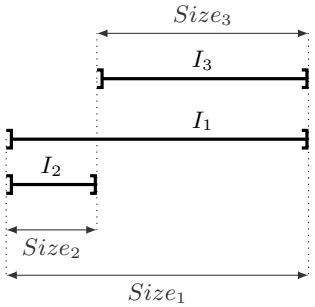


Fig. 1. Building a time window from other ones

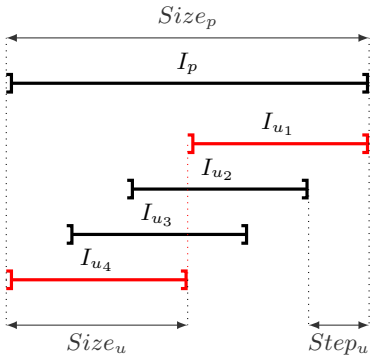


Fig. 2. Union of time windows from a single utility query

Theorem 3 (Incompatibility between a privacy query and a single utility query with aggregates on different time windows). *Let us consider a privacy query Q_p and a utility query Q_u computing the same aggregate Agg on different time windows, and such that Q'_p and Q'_u obtained from Q_p and Q_u by removing their TIMEWINDOW clauses are incompatible. Q_p is incompatible with Q_u using union if and only if the following two conditions are satisfied:*

(1) $\exists m \in \mathbb{N}$ such as $Size_p = Size_u + m \times Step_u$

(2) If Agg is Sum or Count, $\exists n \in \mathbb{N}^+$ and $\exists \alpha \in \mathbb{N}$ such as $Size_u = n \times Step_u$ and $m = \alpha \times n$.

Proof. Computing an aggregate of Q_p from aggregates coming from Q_u is only possible if at least one time window I_p of Q_p can be built by union of m time windows I_{u_x} of Q_u as shown in Figure 2. The following conditions have thus to be satisfied: (a) the union of m time windows of Q_u have the same size than a single time window of Q_p; (b) a time window of Q_u and a time window of Q_p ends at the same time; (c) the union must be disjoint for *Sum* and *Count* (e.g. union of I_{u_1} and I_{u_4} in red in Figure 2). This is captured by the equations:

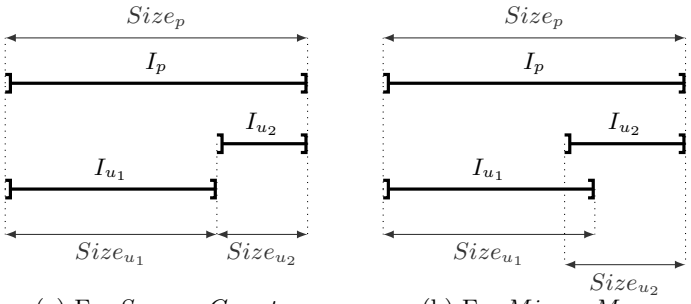
$$\begin{cases} Size_p = Size_u + (m-1) \times Step_u & (1) \\ k_p \times Step_p = k_u \times Step_u & (2) \\ Size_u = n \times Step_u \text{ for } Sum \text{ and } Count & (3.1) \\ Size_p = \alpha \times Size_u \text{ for } Sum \text{ and } Count & (3.2) \end{cases}$$

where k_p and k_u are unknown integers, $Size_p$, $Step_p$, $Size_u$ and $Step_u$ are constant integers, and m, n and α are strictly positive constant integers.

Equation (2) has obviously always solutions (e.g. $k_p = k_u = 0$) and can be discarded. Combining equation (1) and equation (3.2), we obtain:

$$\begin{cases} Size_p = Size_u + (m-1) \times Step_u & (1) \\ Size_u = n \times Step_u \text{ for } Sum \text{ and } Count & (3.1) \\ (\alpha - 1) \times Size_u = (m-1) \times Step_u \text{ for } Sum \text{ and } Count & (3.2) \end{cases}$$

The necessary and sufficient conditions of incompatibility are obtained by combining equation (3.1) and equation (3.2). □



(a) For *Sum* or *Count* (b) For *Min* or *Max*

Fig. 3. Union of time windows from two utility queries

This theorem allows to prove the incompatibility between the privacy query PQ2 and the utility UQ3 of the scenario in Section 2. Indeed, the size of the time window of PQ2 (equal to 6) is a multiple of the size of the time window of UQ3 (equal to 3), which is also a multiple of its step (equal to 1).
This latter condition would not be satisfied if the step of the time window clause of the utility query is replaced by 2 for instance. In this case, the same theorem would allow to infer that PQ2 is compatible with the modified utility query.

When a privacy query is compatible with all utility queries, it remains to check whether it can be incompatible with *combinations* of utility queries. Based on Theorem 2, this can only occur by combining utility queries whose graph patterns are isomorphic with the graph pattern of the privacy query.
Theorem 4 checks incompatibility of Q_p with two utility queries Q_{u_1} and Q_{u_2} by testing if it is possible to build a time window of Q_p by the union of time windows of Q_{u_1} and Q_{u_2} as illustrated in Figure 3.

Theorem 4 (Incompatibility between a privacy query and two utility queries with same aggregates and different time windows). *Let us consider a privacy query Q_p and two utility queries Q_{u_1} and Q_{u_2} computing the same aggregate on different time windows, and such that Q_p without its TIMEWINDOW clause is incompatible with both Q'_{u_1} and Q'_{u_2} obtained from Q_{u_1} and Q_{u_2} by removing their TIMEWINDOW clauses.*
Q_p is incompatible with the pair (Q_{u_1}, Q_{u_2}) using union if and only if the following two conditions are satisfied:

(a) $Size_p = Size_{u_1} + Size_{u_2}$ for SUM or $COUNT$ aggregates
or $Size_p \leq Size_{u_1} + Size_{u_2}$ for MIN or MAX aggregates
(b) $Size_p - Size_{u_1}$ is a multiple of $\gcd(Step_{u_1}, \sigma_p \times Step_{u_2})$
where $\sigma_p = \frac{Step_p}{\gcd(Step_p, Step_{u_2})}$

Proof. Computing an aggregate of Q_p from aggregates coming from a two different utility queries Q_{u_1} and Q_{u_2} is possible if at least one time window I_p of Q_p can be built by union of a time window $I_{u_1}$of Q_{u_1} and a time window I_{u_2} coming from Q_{u_2} as shown in Figure 3. *Sum* and *Count* necessitate that I_{u_1}

and I_{u_2} are disjoint to avoid double counting of an overlap (a), Min and Max support overlapping time windows (b).
The following conditions have to be satisfied (Q_{u_1} and Q_{u_2} can be inverted): (a) the union of a time window of Q_{u_1} and a time window of Q_{u_2} have the same size than a single time window of Q_p; (b) a time window of Q_{u_1} starts when a time window of Q_p starts; (c) a time window of Q_{u_2} ends when the same time window of Q_p ends. This is captured by the following equations:

$$\begin{cases} Size_p = Size_{u_1} + Size_{u_2} \text{ (a) or } Size_p \leq Size_{u_1} + Size_{u_2} \text{ (b)} & (1) \\ k_1 \times Step_{u_1} + Size_{u_1} = k_p \times Step_p + Size_p & (2) \\ k_p \times Step_p = k_2 \times Step_{u_2} & (3) \end{cases}$$

where k_1, k_2 and k_p are unknown integers and $Step_p$, $Size_p$, $Step_{u_1}$, $Size_{u_1}$, $Step_{u_2}$ and $Size_{u_2}$ are constant integers.

The positive integer solutions of equation (3) are of the form:

$$\begin{cases} k_p = \kappa \times \sigma_{u_2} & (3.1) \\ k_2 = \kappa \times \sigma_p & (3.2) \end{cases} \text{ with } \kappa \in \mathbb{N}^+$$

where $\sigma_p = \frac{Step_p}{\gcd(Step_p, Step_{u_2})}$ and $\sigma_{u_2} = \frac{Step_{u_2}}{\gcd(Step_p, Step_{u_2})}$.

Injecting solutions of equation (3) into equation (2), we obtain:

$$\begin{cases} Size_p = Size_{u_1} + Size_{u_2} \text{ (a) or } Size_p \leq Size_{u_1} + Size_{u_2} \text{ (b)} & (1) \\ k_1 \times Step_{u_1} - \kappa \times \sigma_p \times Step_{u_2} = Size_p - Size_{u_1} & (2) \\ k_p = \kappa \times \sigma_{u_2} & (3.1) \\ k_2 = \kappa \times \sigma_p & (3.2) \end{cases}$$

Equations (3.1) and (3.2) are always satisfied and can be discarded. According to Bachet-Bézout theorem, the Diophantine equation (2) has a solution if and only if $Size_p - Size_{u_1}$ is a multiple of $\gcd(Step_{u_1}, \sigma_p \times Step_{u_2})$. □

Example 3. Let us consider the privacy query PQ2 of the scenario in Section 2 and the following utility queries:
Q_{u_1}: SELECT ?timeWindowEnd SUM(?cons)
WHERE {(?sm issda:consumption ?cons, ?ts)}
GROUP BY ?timeWindowEnd
TIMEWINDOW (4h, 2h)
Q_{u_2}: SELECT ?timeWindowEnd SUM(?cons')
WHERE {(?sm' issda:consumption ?cons', ?ts')}
GROUP BY ?timeWindowEnd
TIMEWINDOW (2h, 1h)
As $Size_p = 6$ and $Size_{u_1} + Size_{u_2} = 6$, the condition (a) of the theorem is satisfied.
As $Step_p = 6$, $Step_{u_1} = 2$ and $Step_{u_2} = 1$, we get: $\sigma_p = 6$ $\gcd(Step_{u_1}, \sigma_p \times Step_{u_2}) = 2$, and $Size_p - Size_{u_1} = 2$, thus making the condition (b) of the theorem also satisfied. So PQ2 is incompatible with Q_{u_1} and Q_{u_2}.
If we replace the $TIMEWINDOW$ of Q_{u_1} by (3h, 2h), the condition (a) of the theorem is not satisfied and PQ_2 is compatible with the new Q_{u_1} and Q_{u_2}.

5 Related Work

A rich variety of privacy models have been proposed, ranging from k-anonymity [14] and l-diversity [12] to t-closeness [11] and ϵ-differential privacy [6]. 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. Our data-independent approach is complementary and should be used beforehand for detecting privacy breaches. It comes with explanations that can help choose the privacy model to apply to the data concerned by the detected privacy breaches.

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 [1].

An alternative approach for protecting against privacy breaches consists in applying access control methods to RDF data [13,15,10]. However, all these works do not handle utility queries.

A query-based logical framework for RDF data has been introduced in [7,8], where sensitive information is expressed as SPARQL queries whose results must not disclose sensitive information of individual. It has been extended to handling utility queries in [4,5]. These approaches however are restricted to simple conjunctive queries. They do not consider aggregates and they cannot apply to temporal data.

6 Conclusion and Future Work

In this paper we have proposed a data-independent framework for a formal specification and verification of compatibility between privacy and utility policies expressed as temporal aggregate conjunctive queries.

This framework is well suited for helping data producers to keep the control on the protection of their data in many real-world situations where sensitive data are collected by mobile personal devices or smart environments. When a privacy query turns out to be incompatible with utility queries, several solutions can be applied, such as ciphering the exposed data made explicit by the explanation built from the incompatibility proof, anonymize them [3], or use differential privacy [9].

Based on the implementation of this framework, we plan to design and implement a negotiation mechanism that will be triggered in this case. 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 acceptable trade-off 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.

References

1. Ciucanu, R., Lafourcade, P.: Goose: A secure framework for graph outsourcing and sparql evaluation. In: IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec). pp. 347–366 (2020)
2. Cohen, S.: Containment of aggregate queries. ACM SIGMOD Record **34**(1), 77–85 (2005)
3. Delanaux, R., Bonifati, A., Rousset, M.C., Thion, R.: Query-Based Linked Data Anonymization. In: International Semantic Web Conference (ISWC). pp. 530–546 (2018)
4. Delanaux, R., Bonifati, A., Rousset, M.C., Thion, R.: Query-based linked data anonymization. In: The Semantic Web-ISWC 2018. pp. 530–546. Springer, Cham (October 8-12, 2018 2018). https://doi.org/10.1007/978-3-030-00671-6_31
5. Delanaux, R., Bonifati, A., Rousset, M.C., Thion, R.: Rdf graph anonymization robust to data linkage. In: Proceedings of WISE 2019 (20th International Conference on Web Information Systems Engineering) (2019)
6. Dwork, C.: Differential privacy. In: ICALP (2). LNCS, vol. 4052, pp. 1–12. Springer (2006)
7. Grau, B.C., Kostylev, E.V.: Logical foundations of privacy-preserving publishing of linked data. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 943–949. The AAAI Press, Palo Alto, California (February 12–17, 2016 2016). https://doi.org/10.5555/3015812.3015953
8. Grau, B.C., Kostylev, E.V.: Logical foundations of linked data anonymisation. Journal of Artificial Intelligence Research **64**, 253–314 (2019)
9. Hassan, M.U., Rehmani, M.H., Chen, J.: Differential privacy techniques for cyber physical systems: A survey. IEEE Communications Surveys Tutorials **22**(1), 746–789 (2020). https://doi.org/10.1109/COMST.2019.2944748
10. Kirrane, S., Mileo, A., Decker, S.: Access control and the resource description framework: A survey. Semantic Web **8**(2), 311–352 (2017)
11. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE. pp. 106–115. IEEE Computer Society (2007)
12. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: *L*-diversity: Privacy beyond *k*-anonymity. TKDD **1**(1), 3 (2007)
13. Oulmakhzoune, S., Cuppens-Boulahia, N., Cuppens, F., Morucci, S.: Privacy policy preferences enforced by SPARQL query rewriting. In: ARES. pp. 335–342. IEEE Computer Society (2012)
14. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(5), 557–570 (2002)
15. Villata, S., Delaforge, N., Gandon, F., Gyrard, A.: An access control model for linked data. In: OTM Workshops. LNCS, vol. 7046, pp. 454–463. Springer (2011)