Privacy-Preserving Linked Data Integration

Rémy Delanaux

Université Claude Bernard Lyon 1

Numéro d'ordre NNT : 2019LYSE1303

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON
Opérée au sein de **l'Université Claude Bernard Lyon 1**

Ecole Doctorale 512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 13 décembre 2019, par :

Rémy DELANAUX

Intégration de données liées respectueuse de la confidentialité

Privacy-Preserving Linked Data Integration

Devant le jury composé de :

M. Hamamache KHEDDOUCI Professeur, Université Claude Benard Lyon 1	**Président de Jury - Examinateur**
M. Nicolas ANCIAUX Directeur de Recherche, INRIA Saclay Île-de-France	**Examinateur**
M. Benjamin NGUYEN Professeur, INSA Centre-Val de Loire	**Rapporteur**
Mme. Nathalie PERNELLE Maîtresse de conférence HDR, Université Paris-Sud	**Examinatrice**
Mme. Hala SKAF-MOLLI Maîtresse de conférence HDR, Université de Nantes	**Rapporteure**
Mme. Angela BONIFATI Professeure, Université Claude Bernard Lyon 1	**Directrice de thèse**
Mme. Marie-Christine ROUSSET Professeure, Université Grenoble-Alpes	**Co-Directrice de thèse**
M. Romuald THION Maître de conférences, Université Claude Bernard Lyon 1	**Co-Directeur de thèse**

Université Claude Bernard – LYON 1

Président de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Jean-François MORNEX
Directeur Général des Services	M. Damien VERHAEGHE

COMPOSANTES SANTE

Faculté de Médecine Lyon-Est – Claude Bernard	Doyen : M. Gilles RODE
Faculté de Médecine et Maïeutique Lyon Sud Charles. Mérieux	Doyenne : Mme Carole BURILLON
UFR d'Odontologie	Doyenne : Mme Dominique SEUX
Institut des Sciences Pharmaceutiques et Biologiques	Directrice : Mme Christine VINCIGUERRA
Institut des Sciences et Techniques de la Réadaptation	Directeur : M. Xavier PERROT
Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

UFR Biosciences	Directrice : Mme Kathrin GIESELER
Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
UFR - Faculté des Sciences	Administrateur provisoire : M. Bruno ANDRIOLETTI
UFR (STAPS)	Directeur : M. Yannick VANPOULLE
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Ecole Polytechnique Universitaire Lyon 1	Directeur : Emmanuel PERRIN
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Institut de Science Financière et d'Assurances	Directeur : M. Nicolas LEBOISNE
ESPE	Administrateur Provisoire : M. Pierre CHAREYRON

Foreword

Acknowledgements

First and foremost, I would like to deeply thank Hala Skaf-Molli and Benjamin Nguyen for accepting to review my thesis. And I would also like to thank Nathalie Pernelle, Nicolas Anciaux, and Hamamache Kheddouci for being part of my Ph.D. committee.

Then, I would obviously like to thank my three advisors for this Ph.D. thesis: Angela Bonifati and Romuald Thion at the LIRIS in Lyon, and Marie-Christine Rousset at the LIG in Grenoble. Their mentorship and their insight allowed me to present this manuscript today, and encouraged me until the end. Their patience, their trust, and their collaboration in my work have always been a huge helping hand, and I cannot be grateful enough.

I would also like to thank every member of the BD team at the LIRIS, whose daily support was very much appreciated. I also want to thank many other members and staff from the lab, notably Catherine Lombardi for her help in any administrative inquiry, and Pierre-Antoine Champin for his support regarding my teaching activities. More pragmatically, I would also like to thank the Région Rhône-Alpes for providing the funding for this Ph.D. in 2016.

A Ph.D. would not be a good experience without other Ph.D candidates. As such, I would like to warmly thank the many doctoral students with whom I shared those 3 years, sometimes even sharing offices, and who were always welcoming, supportive, and kind. In particular, I would like to thank those colleagues-turned-friends: Arthur, Maxime, Valentin, Alexis, Marc, Charles, Antoine W., Antoine D., Thibault, Julia, Matthieu, Jocelyn, Agathe, Alice, Samuel, Thomas, and all the others.

It is unusual, but I would also like to thank my closest online friends. While they were not often physically there to support me, their daily support and relief was so beneficial that I have to mention them. Louis, Kevin, Maxime, Gaétan, Arqa, Victor, Ammar, and everybody I forgot, thank you so much for enduring my rants and remarks on various social networks and for being available even in the middle of night to discuss just about anything.

Finally, I am obviously indebted with gratitude towards my parents and my brother for their support through thick and thin.

Résumé étendu en français

Sujet et motivations

Cette thèse porte sur la rencontre entre deux secteurs clés de la gestion de données sur Internet : la confidentialité et protection de données sensibles, et le *Linked Open Data* (ou « LOD », en français « web des données ouvertes » ou encore « données liées ouvertes »). Le but est alors de concevoir, théoriser et développer des solutions d'anonymisation de données dans ce contexte précis du LOD avec son contexte et ses standards techniques.

Ces données liées sont structurées sous forme de graphes RDF (pour *Resource Description Framework*) structurant l'information en triplets (sujet / propriété / objet), à la différence de données relationnelles par exemple. S'il existe déjà de nombreuses techniques d'assainissement de jeux de données contenant des informations sensibles, elles ne s'appliquent pas ou difficilement aux graphes RDF du LOD et tiennent pas compte de politiques de confidentialités associés à cette structure. Il convient ainsi d'avancer l'état de l'art à cette rencontre de domaines, pour le moment assez maigre [Grau16, Heitmann17]. Ceci doit être fait proposant de nouvelles méthodes pour assurer l'anonymité et la sécurité d'ensembles de données, en garantissant le maximum d'utilité pour les données anonymisées, le tout dans un format structuré, standardisé et beaucoup utilisé notamment au niveau institutionnel. Elle s'inscrit dans un contexte où la confidentialité des données est devenu un aspect crucial dans le monde des donées (publiques comme privées), comme démontré par les réglements internationaux mis en place à ce sujet tel le Règlement Général sur la Protection des Données (RGPD) européen.

Cette thèse est soutenue par le financement ARC 6 (« T.I.C. et Usages Informatiques Innovants ») de la région Auvergne-Rhône-Alpes.

Formalisation de concepts pour la confidentialité de graphes du LOD

En premier lieu, il convient de définir la *confidentialité de données* dans un tel contexte, à la fois sur la formalisation de garanties de confidentialité et d'utilité, mais aussi d'opérateurs applicables pour pouvoir les garantir. Traditionnellement, on considère des champs de données appelés *identifiants* qui identifient directement une personne ou une information, sans information externe et des *quasi-identifiants*, pouvant agir comme identifiants quand ils sont recoupés avec des informations tierces. Ces deux types d'attributs sont les éléments visés au sein de processus d'anonymisation comme la *k-anonimity* et ses dérivés [Sweeney02b, Machanavajjhala07, Li07] dont le but est de rendre indistinguable une entrée de $k-1$ autres, possédant les mêmes valeurs pour leurs quasi-identifiants. Avec une structuration en triplets RDF, cette décomposition ne se fait pas naturellement et implique beaucoup de pertes : de nombreuses requêtes ne

retourneront plus les résultats escomptés sur les graphes anonymisés. Ces méthodes ne permettent donc pas de bénéficier pas d'une bonne *utilité* des données. Une autre famille de méthodes concerne la confidentialité différentielle (*differential privacy*), dont le but est de préserver l'intégrité statistique globale d'un ensemble de données, en introduisant du bruit modifiant des entrées individuelles. Toutefois, pour des requêtes non-statistiques, ces méthodes ne fournissent qu'une utilité moindre. En effet, si les résultats de fonctions mathématques peuvent être facilement préservés (somme, moyennes, espérance...), ce n'est pas le cas pour des requêtes ayant pour but de récupérer des informations précises (ce qui est souvent le cas dans le Web Sémantique qui porte avant tout un aspect institutionnel et culturel plutôt que statistique).

Partant de ce constat, notre choix s'est porté sur une formalisation basé entièrement sur des requêtes, permettant à la fois une déclaration statique de contraintes et d'opérations et l'utilisation des technologies spécifiques à RDF (notamment le langage SPARQL). Cette proposition simple dite *query-based* permet une accessibilité pour les fournisseurs de données afin de s'assurer de garanties pertinentes de confidentialité et d'utilité.

Le concept clé parmi ces outils est la notion de **politique** : il s'agit d'un ensemble de requêtes SPARQL servant à modéliser les contraintes sur les données qui devront être respectées post-anonymisation. C'est donc l'outil principal du fournisseur de données (ou du *Data Protection Officer* (DPO), pour reprendre la terminologie du RGPD) pour exprimer ses besoins. Selon le framework, ces politiques peuvent avoir des sémantiques très variées. Dans les deux contributions presentées dans ce rapport, il s'agit soit de politiques servant à exprimer les données devant être cachées ou anonymisées, ou à représenter celles qui doivent être conservées post-anonymisation.

L'exemple utilisé en fil rouge pendant la thèse présente le cas d'un réseau de transport public voulant publier ses données (réseau et lignes, itinéraires empruntés, statistiques...) sans divulguer d'informations sensibles sur ses utilisateurs : les politiques données en exemple sont donc des requêtes SPARQL simples utilisant un schéma plausible qui modélise un tel ensemble de données, schéma qui sera utilisé pour créer un graphe associé via un générateur Python[1]. Le détail de ce schéma est donné en annexe de cette thèse.

Anonymisation « locale » pour un équilibre confidentialité/utilité

Le premier outil d'anonymisation fonctione via *une politique de confidentialité et une politique d'utilité* : les contraintes du producteur de données sont donc fourniers des requêtes de confidentialité qui ne doivent pas révéler de données sensibles et des requêtes d'utilité dont les résultats doivent conserver leur pertinence. Cette contribution a été le cadre d'une soumission acceptée à la conférence internationale ISWC 2018. Les sémantiques données à ces politiques sont les suivantes :

[1] https://github.com/RdNetwork/DataLyon2RDF/

- Chaque **requête de confidentialité** ne doit plus renvoyer de résultat une fois lancée sur le graphe anonymisé, ou des tuples composés d'au moins un nœud blanc.
- Chaque **requête d'utilité** doit renvoyer les mêmes résultats sur le graphe original et sur le graphe anonymisé.

Ces sémantiques sont contraignantes mais simples, et permettent la conception directe d'algorithmes générant des séquences d'anonymisation. En Figure 1 se trouve un exemple de ces 2 types de politiques dans notre exemple fil rouge.

Figure 1: Exemple de politiques pour une base de données d'un réseau de transport public

Privacy policy

```
SELECT ?n
WHERE {
  ?u  rdf:type          tcl:User.
  ?u  foaf:familyName   ?n.
}
SELECT ?u ?lat ?long
WHERE {
  ?c  rdf:type        tcl:Journey.
  ?c  tcl:user        ?u.
  ?c  geo:latitude    ?lat.
  ?c  geo:longitude   ?long.
}
```

Utility policy

```
SELECT ?u ?age
WHERE {
  ?u  rdf:type  tcl:User.
  ?u  foaf:age  ?age.
}
SELECT ?c ?lat ?long
WHERE {
  ?c  rdf:type        tcl:Journey.
  ?c  geo:latitude    ?lat.
  ?c  geo:longitude   ?long.
}
```

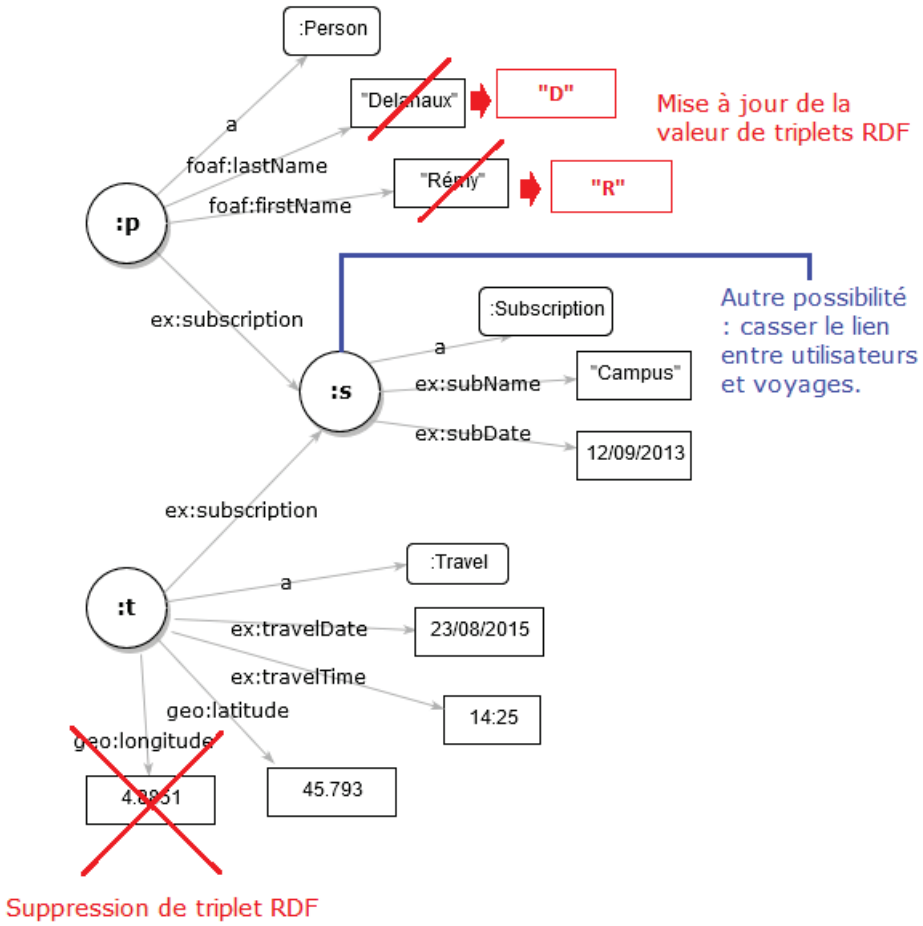
Avant de s'attaquer à la concept d'algorithmes d'anonymisation, nous étudions la *compatibilité* de ces politiques : il existe en effet de nombreux cas où l'anonymisation est impossible, car les politiques se chevauchent trop, causant des situations où l'on veut « cacher ce que l'on veut garder » ou vice-versa. Cette compatibilité est en fait notamment liée au corps de ces requêtes, et à quel point il est possible d'unifier des morceaux du corps de requêtes d'utilité dans le corps d'une requête de confidentialité. Vérifier cela étant trop coûteux, nous montrons une autre propriété : les deux politiques sont compatibles si pour toute requête de confidentialité, il existe au moins un *triple pattern* (motif de triplet RDF) dans le corps de cette requête qui n'est pas unifiable avec quelconque triplet d'une requête d'utilité.

Il faut désormais modéliser les opérations d'anonymisation applicables sur les données, et comment trouver ces opérations de façon pertinente en fonction des contraintes. Pour continuer dans cette démarche *query-based*, les opérations sont définies via des requêtes de mise à jour de graphes RDF, des requêtes du type `DELETE` (suppression de triplets) et `DELETE/INSERT` (mise à jour de triplets).

Pour trouver quelles opérations appliquer, un algorithme a été défini en 2 étapes : un premier cas pour une politique de confidentialité dite « unitaire » contenant une seule requête, puis un algorithme pour le cas général.

Le principe de cet algorithme est de parcourir l'ensemble des triplets contenus dans chaque requête de confidentialité (chaque requête content un sous-graphe propre, sous forme de *graph pattern*), et de vérifier s'il est possible d'unifier chacun des triplets avec ceux contenus dans les requêtes de la politique d'utilité. Si oui, alors il n'est pas possible de le supprimer car cela rentrerait en conflit avec cette politique d'utilité. Si après parcours de toutes les requêtes d'utilité, aucun conflit n'est trouvé, alors ce triplet est candidat à une suppression potentielle. On cherche de cette manière tous les triplets supprimables satisfaisant les sémantiques des 2 politiques.

Figure 2: Exemple d'opérations possibles sur un *graph pattern*



Les opérateurs utilisés par cet algorithme sont la suppression de triplets ou le remplacement d'un sujet ou objet d'un triplet par un nœud blanc. Ces opérations ont

l'avantage d'être implémentables en langage SPARQL natif (donc dans tous les *triple stores* RDF), de s'intégrer facilement aux sémantiques de confidentialité et d'utilité voulues, et d'être facilement extensibles (suppressions sous conditions, ajout de propriétés dans les nœuds blancs insérés...). D'autres opérations sont également envisagées, telles que le remplacement de littéraux par d'autres valeurs littérales qu'on dira *neutralisées* (moins sensibles ou plus générales, par exemple remplacer un âge par une tranche d'âge). La Figure 2 présente de façon visuelle de telles opérations appliquées sur un *graph pattern*, structures utilisées dans les requêtes.

Une étude expérimentale a montré que la composante statique de ce framework rend l'exécution de l'algorithme très rapide dans des cas d'utilisation plausibles utilisant des requêtes de politiques et schémas de données simulés. Le code du prototype ainsi que le pipeline logiciel de l'étude expérimentale sont disponible sur un dépôt GitHub en libre accès[2].

L'analyse expérimentale des séquences d'opérations d'anonymisation produites par cet outil évaluait à la fois quelle chance avaient des politiques synthétiques (composées de requêtes générées de façon plausibles et complètes) d'être compatibles ou non, mais analysait également la longueur de ces séquences. Il en résulte que l'élément crucial est la composition des politiques de confidentialité, dont l'expressivité va permettre de composer plus d'opérations d'anonymisation, comparativement à la politique d'utilité dont le rôle est plus limité. Par ailleurs, lors de ces tests où environ 50% des politiques créées étaient compatibles entre elles, il est apparu que le nombre de séquences d'opération d'anonymisation pouvait potentiellement être gigantesque, en particulier si les politiques n'ont que peu de triplets communs.

Une ouverture importante dans ce contexte serait de trouver comment décider de la *meilleure* séquence parmi toutes celles générées, et comment choisir ces critères d'optimalité.

Anonymisation « globale » sûre face aux attaques par lien

Le second outil développé ne tient plus compte de contraintes d'utilité, mais uniquement de confidentialité. Il s'agit cette fois de se prémunir fac aux attaques par lien (*linkage attacks*) qui sont possibles de façon native avec le fonctionnement par IRI/URI inhérent au Linked Open Data et à RDF. La sémantique de confidentialité détaillée dans la première contribution est renforcée et adaptée à ce nouveau contexte. Une anonymisation va être considérée comme *sûre* si, en faisant l'union du graphe anonymisé avec n'importe quel graphe externe, le résultat des requêtes de la politique de confidentialité n'ajoute aucun tuple de constantes comparativement au résultat de ces mêmes requêtes sur le graphe externe seul. En somme, il faut la garantie qu'aucun tuple constante déclaré comme sensible ne pouvait pas déjà être trouvé sur le graphe externe considéré tout seul. Ce framework génère donc des opérations plus destructrices que le précé-

[2] https://github.com/RdNetwork/Declarative-LOD-Anonymizer

dent. Il a lui aussi été implémenté via un prototype en Python, également disponible de façon ouverte sur un dépot GitHub[3]. Il est le cadre d'un article accepté à la conférence internationale WISE 2019.

L'algorithme générant des opérations permettant de satisfaire ces conditions fonctionne en traitant chaque composante connexe de chaque corps des requêtes de confidentialité. Il va y chercher des éléments que l'on nomme des *termes critiques* d'une requête : ce sont les variables et constantes apparaissant plus d'une fois dans le corps de la requête. Ce sont eux que l'algorithme cible, car une fois ces éléments généralisés par des nœuds blancs, les entités remplacées perdent leur sensibilité. Pour chaque composante connexe trouvée, l'algorithme analyse deux cas :

- On décèle les **termes critiques** de la composante connexe considérée, et on les remplace par des nœuds blancs (chaque occurrence de la même variable étant remplacée par le même nœud blanc);
- Si la composante connexe considérée ne contient aucune des variables résultat de la requête, il s'agit alors d'une composante booléenne et on en supprime un triplet au hasard pour que cette composante n'ait plus d'image dans le graphe.

Chacun de ces cas va générer des requêtes SPARQL `UPDATE` associées. La subtilité étant qu'il faut répéter cette étape de généralisation pour *tous les sous-graphes possibles de la composante connexe analysée*, car sans cette étape il est possible de manquer des occurrences des triplets sensibles, et donc de ne pas tout remplacer.

Cet algorithme ayant une complexité exponentielle dans le pire cas (due au calcul des sous-graphes), une alternative polynomiale l'approximant a été conçue également: au lieu de considérer tous les sous-ensembles possibles du graphe de chaque requête, on remplace chaque occurrence de chaque terme critique par un nœud blanc unique, même si le terme est répété plusieurs fois dans le corps de la requête. Cet algorithme est plus efficace, mais créé des anonymisations plus générales que le premier algorithme, et donc cause plus de perte d'information (on perd les liens entre entités identiques).

Le second cas pris en compte est celui où des failles peuvent être exploitées via la présence de liens `sameAs` dans les données, c'est-à-dire des triplés indiquant qu'une ressource est identique à une autre. Ces liens peuvent être explicites (présence de triplets `sameAs`) ou implicites (triplets `sameAs` que l'on peut peut déduire d'autre triplets). Pour remédier aux problèmes inhérents à ces liens d'égalité, on définit une **sûreté modulo `sameAs`**, qui fonctionnement avec la même propriété, mais où les réponses aux requêtes de la politique de confidentialité sont évaluées également *modulo sameAs*, c'est-à-dire en générant tous les liens `sameAs` qu'il est possible d'inférer dans le graphe considéré pour l'application de la requête SPARQL. Dans le cas de liens `sameAs` explicites, la séquence d'opération fournie par les deux algorithmes évoqués dans le cas standard garantit déjà la sûreté modulo `sameAs` du graphe anonymisé.

[3] https://github.com/RdNetwork/safe-lod-anonymizer

Toutefois, des **liens `sameAs` implicites** sont déductibles lorsque certaines propriétés sont établies comme étant *fonctionnelles* (ou *inversement fonctionnelles* dans le sens inverse), c'est-à-dire que pour une entité s ne peut avoir qu'une unique valeur v associée à une propriété p donnée. Si deux triplets (s, p, v_1) et (s, p, v_1) existent dans le graphe et que p est explicitement décrite comme fonctionnelle, on peut en déduire le triplet $(v_1, \texttt{sameAs}, v_2)$. Même chose pour une p inversement fonctionnelle, où avec deux triplets (s_1, p, v) et (s_2, p, v) on peut déduire $(s_1, \texttt{sameAs}, s_2)$.

Les liens `sameAs` peuvent être également déduits lorsqu'on connaît la *fermeture* d'une propriété, c'est-à-dire lorsqu'on possède dans un graphe externe la connaissance de toutes les valeurs possibles pour cette propriété. Si ce cas arrive, il faut alors ajouter une requête à la politique de confidentialité pour cacher toutes les valeurs possibles aux deux extrémités de tous les triplets utilisant cette propriété, ce qui est très destructeur.

Evaluation expérimentale

Une évaluation expérimentale poussée a été réalisée pour cette seconde contribution, en particulier sur son application dans un contexte réel, en appliquant véritablement les opérations sur des ensembles de données réels ou synthétiques et en étudiant la perte d'information que ces opérations causaient en pratique. Le processus d'évaluation et les résultats liés sont le cadre d'une contribution acceptée à l'atelier APVP 2019.

En s'intéressant à divers datasets synthétiques (données de transport TCL, données d'université du benchmark LUBM) ou réel (base de données de médicaments Drugbank, données de patrimoine suédois Europeana), nous avons pu constater que comme pour le premier framework, la partie statique de l'exécution (la génération d'opération d'anonymisation) était toujours très performante.

En ce qui concerne l'anonymisation elle même et l'étude des graphes anonymisés, les mesures ayant été effectuées calculent le nombre de nœuds blancs insérés par le processus d'anonymisation (dans l'absolu, ou relativement au nombre d'IRI existant dans le graphe au départ), la distance entre les distributions des degrés du graphe avant et après anonymisation, la similarité triplet par triplet entre le graphe original et sa version anonymisée, ainsi que le nombre de composantes connexes entre graphe original et anonymisé. La première est une mesure de précision simple de l'impact des opérations générées tandis que les suivantes mesurent de différentes façons la perte d'utilité dite *structurelle* sur le graphe. Ces mesures sont effectuées en créant des *mutations* de chaque politique considérée, c'est-à-dire des politiques où l'on applique des modifications incrémentales qui la rendent plus spécifique (on remplace une variable par une constante prise dans les valeurs existantes dans le graphe pour ce triplet, la requête ainsi modifiée sélectionnant ainsi moins de triplets dans le graphe originel). On peut donc ainsi mesurer la perte d'utilité en fonction de la spécificité de la politique.

Sur chaque graphe réel considéré, on observe que la précision baisse de façon assez linéaire quand la politique baisse en spécificité : quand elle est plus générale (avec peu

de constantes), on va généraliser plus de triplets et donc ajouter plus de nœuds blancs. Toutefois, comme nos algorithmes essaient de modifier que « ce qui est nécessaire » (donc ce qui est indiqué dans les politiques de confidentialité) et pas plus, si les motifs concernés ne sont qu'une petite portion de ce que contient l'entièreté du graphe, la précision et donc l'utilisabilité des données reste bonne.

En ce qui concerne la distance entre distributions de degrés, les résultats sont plus flous. Sur certains datasets, cette distance semble augmenter quand la policy est moins spécifique. Si le graphe a une structure plutôt simple, la distance entre distribution de degré peut varier grandement si les triplets modifiés sont centraux au sein du graphe.

Afin de trouver des mesures structurelles plus pertinentes, nous mesurons également la similarité et l'évolution du nombre de composantes connexes entre le graphe original et le graphe anonymisé. On peut observer que la similarité baisse quand la spécificité de la politique monte, mais reste haute dans tous les cas (plus de 75% en permanence sur les 4 graphes étudiés). Notre approche est donc peu destructrice à l'échelle d'un graphe entier.

Le nombre de composantes connexes augmente lui de façon significative quand la spécificité de la politique monte, indiquant des modifications structurelles potentiellement conséquentes. Ceci peut indiquer une perte d'utilité certaine, mais il est à noter que cette valeur dépend énormément de la structure originale du graphe et à quel point certaines propriétés sont réutilisées par beaucoup d'entités au sein de ce graphe.

Conclusion et perspectives

De nombreux aspects de ce projet peuvent être des axes d'amélioration ou d'innovation : au niveau des politiques, leur utilisation peut être étendue en prenant en compte d'autres types de requêtes, plus complexes ou plus expressives que des simples requêtes conjonctives (telles des *path queries*). De nouvelles opérations d'anonymisation peuvent être considérées, en partant des bases permises par le standard SPARQL Update (suppression et mise à jour de triplets). Il s'agit notamment de considérer des motifs plus complexes que des suppressions/insertions uniques, et par exemple des mises à jour des valeurs littérales utilisant la généralisation de valeurs.

Une autre perspective est l'amélioration des algorithmes et de la génération des séquences, en termes de complexité et de performance mais également en ce qui concerne d'éventuelles redondances dans les séquences d'opérations trouvées : certaines opérations peuvent en effet essayer de supprimer ou mettre à jour des triplets supprimés, ce qui peut rallonger inutilement l'exécution.

Enfin, une prise en compte plus poussée des risques liés au raisonnement et à l'inférence semble importante vu leur aspect central en RDF et dans le monde du Web sémantique.

Du reste, ce projet montre qu'il est possible de concevoir des cadres clairs et simples d'utilisation pour l'anoymization de graphes RDF avant publication dans le Web des données ouvertes et utilisation comme portion du Web sémantique. Les deux sémantiques étudiées ont montré que trouver un compromis confidentialité/utilisé est viable dans un cadre simple si les politiques définies sont plausibles, et qu'il est possible d'avoir une confidentialité satisfaisante dans un cadre réaliste de vulenérabilité aux attaques par liens. La simplicité de ce framework en fait donc un travail précurseur réutilisable et potentiellement important pour tout un champ de recherche.

Abstract

INDIVIDUAL privacy is a major and largely unexplored concern when publishing new datasets in the context of Linked Open Data (LOD). The LOD cloud forms a network of interconnected and publicly accessible datasets in the form of graph databases modeled using the RDF format and queried using the SPARQL language. This heavily standardized context is nowadays extensively used by academics, public institutions and some private organizations to make their data available. Yet, some industrial and private actors may be discouraged by potential privacy issues.

To this end, we introduce and develop a declarative framework for privacy-preserving Linked Data publishing in which privacy and utility constraints are specified as policies, that is sets of SPARQL queries. Our approach is data-independent and only inspects the privacy and utility policies in order to determine the sequence of anonymization operations applicable to any graph instance for satisfying the policies. We prove the soundness of our algorithms and gauge their performance through experimental analysis.

Another aspect to take into account is that a new dataset published to the LOD cloud is indeed exposed to privacy breaches due to the possible linkage to objects already existing in the other LOD datasets. In the second part of this thesis, we thus focus on the problem of building safe anonymizations of an RDF graph to guarantee that linking the anonymized graph with any external RDF graph will not cause privacy breaches. Given a set of privacy queries as input, we study the data-independent safety problem and the sequence of anonymization operations necessary to enforce it. We provide sufficient conditions under which an anonymization instance is safe given a set of privacy queries. Additionally, we show that our algorithms are robust in the presence of sameAs links that can be explicit or inferred by additional knowledge.

To conclude, we evaluate the impact of this safety-preserving solution on given input graphs through experiments. We focus on the performance and the utility loss of this anonymization framework on both real-world and artificial data. We first discuss and select utility measures to compare the original graph to its anonymized counterpart, then define a method to generate new privacy policies from a reference one by inserting incremental modifications. We study the behavior of the framework on four carefully selected RDF graphs. We show that our anonymization technique is effective with reasonable runtime on quite large graphs (several million triples) and is gradual: the more specific the privacy policy is, the lesser its impact is. Finally, using structural graph-based metrics, we show that our algorithms are not very destructive even when privacy policies cover a large part of the graph.

By designing a simple and efficient way to ensure privacy and utility in plausible usages of RDF graphs, this new approach suggests many extensions and in the long run more work on privacy-preserving data publishing in the context of Linked Open Data.

Contents

List of Figures

List of Tables

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

Tim Berners-Lee – *Weaving the Web*, ch. 12 (1999)

1 Introduction

▷ *Please note that each chapter will feature a summary of its contributions and contents, as well as a table of contents of the chapter in the following page.*

As a starting point, we introduce the general context of RDF, the Semantic Web and the Linked Open Data cloud, as well as why confidentiality matters in this situation too. We continue by describing more in detail what are the motivations and objectives of this Ph.D thesis, in which theoretical and technical scopes it is relevant and finally, the main contributions it induced. We also design some running examples to be used later. We end by detailing the global structure of this manuscript. ◁

Chapter summary

THE Web has become the biggest place where information is exchanged between individuals, corporate companies, researchers, or public institutions since its conception and its opening to the masses. It is maintained both by automated processes and user-generated content. The core component of the context for this thesis is an extension of the common notion of World Wide Web named **Semantic Web**, and more precisely the **Linked Open Data cloud**. While this initiative is very important for the Web as a whole, dealing with massive amounts of data, it also raises some issues when dealing with personal data at a smaller, individual scale.

1.1 A Web of open and structured sets of data

The semantic Web and Linked Open Data are concepts and implementations designed by the World Wide Web Consortium, notably starting the 1990s at the instigation of Tim Berners-Lee, inventor of the World Wide Web [Berners-Lee01, Berners-Lee06]. Their goal is to *extend* the classical Web to make it machine-readable and ease data reuse between institutions, companies, and individuals. Notably, instead of only using *URLs* (Uniform Resource Locator) to identify websites, this notion is generalized to create *URIs* (Uniform Resource Identifier) identifying any type of resource, whether it is a website, a person, a real-life location, and so on. Acting both as a set of integration standards and as a data network, the Semantic Web has been largely used by academics and public institutions in the latest decades. In itself, **Linked Data** represents the whole subset of data available on the Web, and interlinked using semantic Web standards; in a way, Linked Data shapes the semantic Web, and the semantic Web is comprised of Linked Data[1].

The motivation is technical as well as ethical. While the goal is mostly to turn the Internet into a giant "database of knowledge and information", in order to make it queryable by any machine, there is an usual emphasis on Linked *Open* Data. The general aim is to create freely accessible sets of data, making it possible to create applications fetching this data and rendering public knowledge.

From a technical standpoint, this is performed by using a standardized data modeling framework: **RDF data** [Beckett04] (for Resource Description Framework), which describes any information in a triple format (Subject - Property - Value) about entities referenced by means of URIs. The subject can be any entity, and the value can be another entity or a literal(string, numerical or data constants, for example). Multiple RDF triples form a labeled directed graph, creating a simpler model closer to usual knowledge engineering models rather than typical relational databases. Each graph can also be identified with its own URI, each triple being therefore identifiable as a *quadruple* rather than a *triple*, and a collection of RDF graphs is called an RDF *dataset*,

[1] Berners-Lee himself repeatedly said that Linked Data was "*semantic Web done right*", for example in [Berners-Lee08].

as per the standard guidelines [Wood14]. There are many different syntactic dialects that can be used to write RDF data, each with their share of simplified syntax, yet each can represent the same RDF data: RDF is not a syntax, rather a format and a standard way to describe data. To query this RDF data, a specific query language is used: **SPARQL** [Seaborne08], loosely inspired from its equivalent for relational databases, SQL.

To evaluate what is "good" Linked Open Data, Berners-Lee defined five incremental criteria using stars [Srivastava13], where each level includes the previous ones:

- ★ The data is freely available on the Web, whatever the format, *with an open license*.
- ★★ It is available in a machine-readable format, as opposed to a binary one (i.e not a scanned image or an executable file).
- ★★★ It is available in a non-proprietary structured format, (i.e CSV and not Microsoft Excel's XLS or DOC).
- ★★★★ It is published using open semantic web standards from the W3C (RDF for data modeling and storage, and SPARQL for querying).
- ★★★★★ It links to other Linked Open Data resources.

The Linked Open Data *cloud* is therefore a finite set of RDF graphs describing identified entities. It is still rapidly growing as nowadays, and contains 1,239 RDF datasets connected by 16,147 links as of March 2019[2] (as represented on Figure 1.1, where each colored circle represents an RDF dataset. Since 2007, the number of RDF graphs published to the LOD cloud has grown by two orders of magnitude. The latest statistics (as of 2019) tell us that it is currently dominated by four main fields: institutional data (government or cities' public data), life science data (biological databases), linguistics data (dictionaries, corpora, or typography lists), and of course cross-domain datasets (general knowledge data such as encyclopedia), notably Wikidata and DBpedia. Some smaller, usually thematic graphs also exist (such as MusicBrainz or the BBC programs).

1.2 When confidentiality and openness clash

Whereas many organizations, institutions and governments participate to the LOD movement by making their data accessible and reusable to citizens, the risks of identity disclosure in this process are not completely understood. As an example, it is easy to picture a context in smart city applications where information about users' journeys in public transportation could help re-identify the individuals if they are joined

[2] All statistics from `https://lod-cloud.net/`.

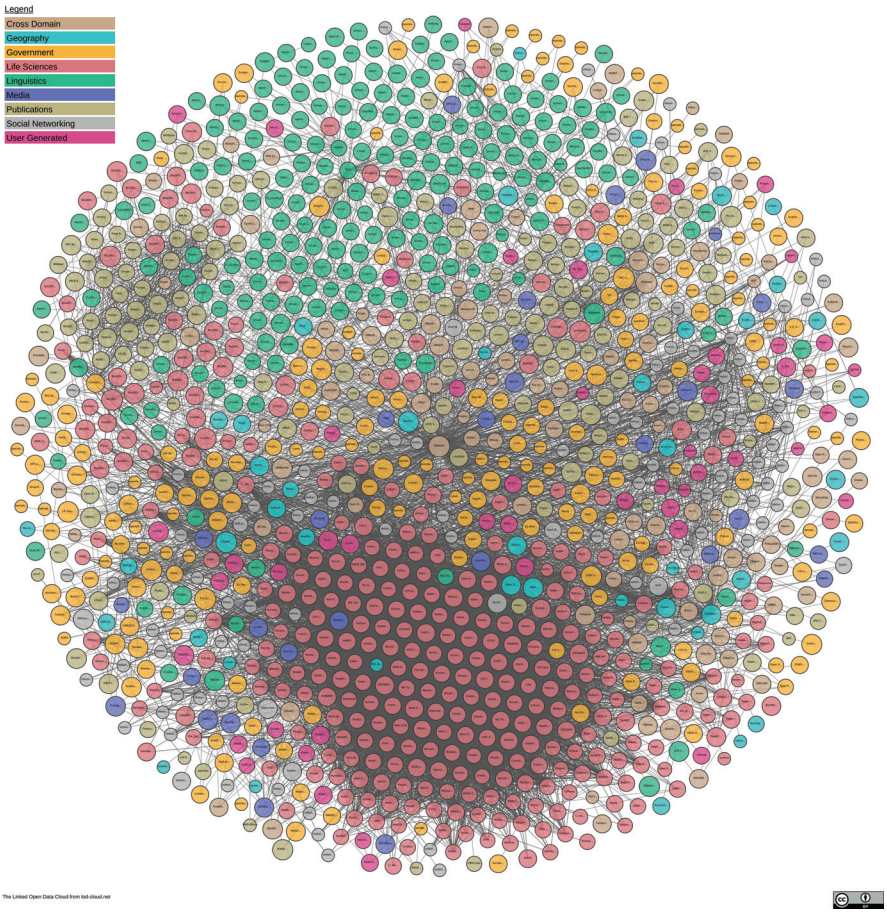


Figure 1.1: The state of Linked Open Data cloud as of March 2019. From lod-cloud.net

with other public data sources by leveraging "quasi-identifiers", that is elements that are not unique per individual, but could form personally identifying information once combined with other accessible quasi-identifiers [Dalenius86, Sweeney02b]. By definition, using any set of data containing information regarding individuals whether it is directly (through *identifiers* such as names or social security numbers) or indirectly (through *quasi-identifiers* such as date of birth, gender, etc.) can be sensitive to some extent. Besides, past research showed that many traditional databases are likely to be subject to re-identification of individuals [Sweeney02b, Narayanan08]. To avoid this, the publishing organizations simply refrain from publishing, or publish only aggregated datasets such as statistical databases, which in itself prevents privacy leakages but loses all individual information. They also still tend to delete whole chunks of data to prevent such privacy breaches or leakages, or sometimes even refuse to publish their data to the LOD cloud, thus holding up the growth of the Semantic Web.

For data providers willing to publish useful data from collected datasets that

contain personal data about individuals, the main problem is to determine which anonymization operations must be applied to the original dataset in order to preserve both individuals' privacy and data utility. For all these reasons, data providers should have at their disposal the means to readily anonymize their data prior to publication into the LOD cloud.

Handling personal information has become crucial in Europe given the context of the official EU General Data Protection Regulation (GDPR)[3], to which every organization must comply: safeguards must be put in place, and anonymization (or pseudo-anonymization) must be guaranteed for every user of any given computerized system. For each company or entity whose core occupation imply the collection of personal data, a *Data Protection Officer* (DPO) must be employed to ensure compliance with the GDPR regulations[4]; such anonymization tools will there be used by so called DPOs in conjunction with the data provider themselves.

But data utility usually pays the price of this necessary privacy, particularly if it happens in a technical context where less anonymization solutions have been designed, or if they are not accessible or understandable by data providers. The whole point of data anonymization, from the standpoint of the provider, is to reach the best data utility/-data privacy compromise, i.e. keeping as much as possible of useful and relevant data. This may sound counter-intuitive, as we tend to figure out data utility and data privacy as two extremities of the same axis. Those goals are indeed contradictory, and impossible to reach if an attacker has sufficient prior knowledge on the data [Rastogi07]. But the point is to keep data utility *maximal* while data privacy is *guaranteed* given some specific constraints set beforehand and which are (at least at the time when they are written) specifically stating which parts of the data are sensitive and should not be displayed in an anonymized rendition of the dataset. Therefore, this is more of an optimization problem rather than a contradiction: we can think of it as maximizing the value of one variable (utility) while strictly keeping another variable (privacy) below a precise threshold. This is, of course, only possible if the stated and desired privacy and utility constraints are not contradictory themselves, in which case any anonymization is impossible from the beginning.

1.3 Motivation and goals

In general, the whole notion of data privacy and finding the best or a good compromise between privacy and utility has been extensively studies in the case of relational databases, a field itself active for several decades now. It is a major field of study in data management and databases, notably because it has both a theoretical interest (in terms of anonymization algorithms, mathematical and logical privacy guarantees) and

[3]Full text available here: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679 - Summarized and arranged here: https://gdpr-info.eu/

[4]See Articles 37, 38 and 39 of the GDPR regulations.

a practical interest for any application dealing with sensitive data. Despite being an intensively studied field, some entities may still struggle in finding ways to model privacy or utility constraints to be guaranteed on an anonymized version of their database.

In addition to this and as mentioned before, it is crucial for data privacy and data management research in general to progress with regard to the Semantic Web and the Linked Open Data cloud. While numerous standards have been developed since the early 2000s, it is clear that many institutions, both private and public, are still reluctant to publish data to the LOD cloud, for various reasons such as:

- Lack of awareness about the field in general: Semantic Web remains unknown or misunderstood for a significant number of people in I.T., which may result in a lack of interest or motivation to join the movement;
- Lack of confidence regarding privacy: no real, accessible and comprehensive Linked Data-based software solution have been proposed, standardized or universally used to ensure privacy on RDF graphs.

The global goal of this Ph.D. is therefore to think an anonymization framework, and design, implement and evaluate algorithms and techniques to ensure the anonymization of Linked Open Data in the shape of RDF graphs, while providing guarantees on the usability of such anonymized data and how these guarantees could be expressed and proved. This entails a detailed analysis of multiple aspects. First, how data privacy and data utility can be precisely defined in the context of RDF graph databases. Second, what type of anonymization operations can be conceived and applied on RDF graphs, and how they should be written or encoded. Also, how to design anonymization algorithms so that these operations are applied in a correct way, and prove that this process actually anonymizes data as intended, and keeps data utility as desired as well. Then, we must think on how such a system works with the specificities and caveats inherent to Linked Data and RDF; notably, how our approaches handle privacy leakages caused by linking attacks.

In short, the end goal is to provide with a sound, usable (in terms of quality of the results), efficient (in terms of performance) software solution to anonymize any given RDF graph before it is published to the LOD cloud, by building and proving all the logical components of such a framework and by designing every technical component so that it can be implemented efficiently.

1.4 Contributions

The main contributions of this thesis are separated in two parts, both aiming at developing generic RDF anonymization solutions in two different situations modeling different privacy semantics. From a practical standpoint, both are designed by leveraging at the

same time the expressive power of SPARQL queries and the maturity and effectiveness of SPARQL query engines.

1.4.1 Declarative anonymization framework in a local context

The first main contribution [Delanaux18], which sets up most of the tools that will be used through the following chapters, is a novel declarative framework that:

- allows the data providers to specify both the privacy and utility requirements they want to enforce as standard SPARQL queries, named *policies*;
- checks whether the specified policies are compatible with each other;
- automatically builds candidate sets of anonymization operations that are guaranteed to transform any input dataset into a dataset satisfying the required privacy and utility policies, based on a set of basic update queries. These operation are native, standard SPARQL queries as well.

This is completed by an algorithm implementing this *data-independent method* starting from privacy and utility policies only, and we prove its soundness. The anonymization processes resulting from the application of the algorithm are then compared in terms of information loss and provided to the data producers who may eventually choose which anonymization procedure they prefer to apply. The data providers can thus decide which part of their dataset should be kept secret, and explicitly state which part should rather be published in the LOD cloud, all of this simply by queries on the original dataset to be anonymized.

The result is a static software solution to anonymize any RDF graph on a local scale, given some explicit privacy and utility constraints on the data.

1.4.2 Declarative anonymization framework preventing external linkage

The second main contribution [Delanaux20] extends this static anonymization framework for RDF data, but switches its approach to one focusing on a core aspect of RDF and Linked Data: actual *linking* between datasets, and the potential problems of sharing known URIs across several sources in the LOD cloud. We name this problem *safety*, as opposed to simply *privacy*. This goal is therefore to focus on the problem of building *safe* anonymizations of an RDF graph, that is anonymizing with guarantees that linking the anonymized graph with any external RDF graph will not cause privacy breaches. While the technical basis remains the same, the goal drastically changes.

This new solution is thus built on the former declarative and query-based approach, still using SPARQL queries to specify both privacy policies and anonymization operations. We exhibit a sufficient condition for the result of an anonymization to be safe and

we provide an algorithm building a set of safe anonymization operations to be applied on any local RDF graph prior to publication to the LOD cloud.

This second contribution focuses on the problem of building safe anonymizations of RDF graphs to guarantee that linking the anonymized RDF graph with any external RDF graph will not lead to sensitive information disclosure. This is split in two separate cases: first, the problem of preventing such privacy leakages caused by the union of the anonymized data with any external RDF graph; then, this study is extended to deal with sameAs links that can be explicit or inferred by some additional knowledge.

Similarly to the first solution, the anonymization algorithms provided here produce anonymization operations (in the form of SPARQL update queries) with the guarantee that their application to any RDF graph will satisfy the privacy policy.

1.5 Running example: transportation data

For the sake of simplicity, we design a simple running example that will be used throughout the following chapters as a concrete application of our contributions.

The general context of this example is that we have in our possession sets of data related to public transportation in a given city, here the city of Lyon and its TCL network. This data includes transportation lines and stops (bus, tramway, subway). To account for possible sensitive data, we also include public data of all places of worship existing in the limits of Lyon. Similarly, since personal user data is obviously not available publicly, we would like to add fictitious personal data of users of the TCL network. That way, we have all the ingredients to create a sensitive context: in theory and without any anonymization process, it would be possible to link users' travels with a place of worship, and possibly gain information regarding their religious habits.

To create a concrete RDF graph based on this data, a custom RDF generator was designed and implemented in Python, based on the actual data publicly available on the Grand Lyon Open Data platform[5] and using synthetic personal data (using fake names, fake addresses from the area, and fake TCL subscriptions). This generator program is publicly available on GitHub.[6]

This dataset describes:

- For users: their name, surname, postal address, date of birth, and potential subscription;
- For subscriptions: the type of subscription, when it began, and optionally when it ended;

[5] https://data.grandlyon.com/
[6] https://github.com/RdNetwork/DataLyon2RDF/

- For each transportation line: the type of transportation and its name, number, and identifier, as well as a list of stops with their coordinates;
- For each place of worship: its name, latitude, and longitude, an identifier and its establishment data;
- For each travel of an user: the ticket validation machine used, the spatial and temporal coordinates of the ticket validation, and the (possibly registered) user who traveled.

A technical documentation of the complete schema of the graph is available on Appendix B, and a few select extracts from the graph are shown on Example 1.1. Prefixes indicated on this example will not be repeated in the following chapters for the sake of conciseness, but they hold the same meaning as in this example.

1.6 Manuscript structure

After this introductory chapter, this thesis is organized as follows.

First, a study of the state of the art is presented on Chapter 2 regarding the multiple aspects combined and inter-connected by this subject. A detailed explanation of the technical and formal backgrounds is provided in the following Chapter 3, stating the logical components used to model RDF objects and SPARQL queries (and answers to those queries), as well as their usage in our framework. Each main contribution is then described and detailed in their own respective chapter, from their theoretical design to their technical implementation: the general framework and its local privacy/utility implementation in Chapter 4, and the global anti-sensitive linkage contribution in Chapter 5. Finally, each solution is then evaluated in an experimental chapter (Chapter 6) where each experiment, its relevance and its goal are described. The manuscript ends with a synthesis of the thesis and some concluding remarks on Chapter 7.

Appendices can be found after the final chapter.

Example 1.1

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix datex:  <http://vocab.datex.org/terms#> .
@prefix tcl:    <http://example.org/> .
@prefix gld:    <http://data.grandlyon.com/> .
@prefix vcard:  <http://www.w3.org/2006/vcard/ns#> .
@prefix gtfs:   <http://vocab.gtfs.org/terms#> .

gld:w15     a   lgdo:placeOfWorship;
    rdfs:label          "Eglise␣Jean␣XXIII";
    geo:latitude        45.7771074615;
    geo:longitude       4.99118788908;
    gld:id              "S2848";
    gld:creationDate    "2001-02-07"^^xsd:date.
[...]
tcl:b77     a   gtfs:Bus;
    tcl:lineNumber      "296";
    tcl:indexNumber     "";
    rdfs:label  "Mions␣Croix-Rouge␣-␣Gare␣Part-Dieu␣Villette";
    tcl:orientation     "Retour";
    tcl:stops       (
        [...]
        [a      gtfs:Stop;
        geo:latitude        "45.6513252454";
        geo:longitude       "4.96193016954"]
[..]
tcl:v26830  a   tcl:Validation;
    tcl:validator       "2932";
    tcl:validationDatetime  "2017-04-27T19:46:51"^^xsd:dateTime;
    tcl:user            tcl:u9590;
    geo:latitude        "45.7599724233";
    geo:longitude       "4.82594480511".
[...]
tcl:u4      a   tcl:User;
    datex:subscription [
        a       datex:Subscription;
        datex:subscriptionStartTime "2008-12-05"^^xsd:date;
        datex:subscriptionStopTime  "2013-03-02"^^xsd:date;
        datex:subscriptionReference "Pro";
    ];
    foaf:givenName      "Mills";
    foaf:familyName     "Cecil";
    vcard:hasAddress    "204␣Boulevard␣A␣69XXX␣Fleurieu-sur-Saone";
    tcl:birthday        "1926-08-20"^^xsd:date.
```

The scientists of today think deeply instead of clearly. One must be sane to think clearly, but one can think deeply and be quite insane.

Nikola Tesla – *Modern Mechanics and Inventions*, "Radio Power Will Revolutionize the World" (1934)

2 Related work

▷ *In this chapter, we delve into the existing literature and find out about classical techniques for anonymizing data, achieving a balance between data privacy and utility, preventing data leakages via linkage attacks, or any safety-related Linked Data literature.* ◁

Chapter summary

DATA privacy in general has been a well-explored field for at least 50 years, as exemplified by the decade-spanning work of researchers like Rein Turn [Turn72, Turn82, Turn90]. With databases came the concern for the data inside them, whether it is for security concerns or for personal safety. It gained traction again in the latest years with the advent and the growing popularity of *Big Data* systems, whose emphasis on large volumes of data sped up the development of sub-fields such as *privacy-preserving data mining* (PPDM), where the goal is to prevent the access or the display to specific, individual data while preserving aggregated or structural properties on the dataset [Agrawal00, Dinur03]. It also became pressing for some to devise simple and clear privacy models to comply with coming regulations such as the European GDPR.

As we will see in this chapter, the majority of the solutions proposed so far tend to be mainly devoted to relational legacy systems and are thus not applicable to the context of RDF data, or they are too abstract and therefore vulnerable to attackers using the particularities of LOD. Such solutions tend to rely on variants or adaptations of differential privacy [Dwork06a] or k-anonymity [Sweeney02b].

While it would be a tedious process to scan the whole, extensive literature regarding database anonymization, we select relevant sub-fields and anonymization principles and techniques, and we thoroughly discuss these methods and preliminary work on Linked Data publishing or *privacy-preserving data publishing* (PPDP) in general in Section 2.1. We then detail our position in this spectrum in Section 2.2.

2.1 Survey of existing data anonymization principles

Whatever the anonymization method used, and whatever the guarantees provided by the anonymization process, some abstract principles are common and agnostic to the data format or the techniques used. The main focus is to achieve a good (or *the best*) privacy/utility trade-off when publishing data. This was studied in various contexts, and with various types of concrete data (such as smart grid measurements [Rajagopalan11] or genomics [Humbert14]) and remains the main objective of data anonymization, provided that the data shall be used by anybody after it has been anonymized. PPDP has been a long-standing goal for several research communities, as witnessed by a flurry of work on the topic [Fung10]. A rich variety of privacy models have been proposed, as will be demonstrated in the following sections. For each of these methods, one or more attack models (such as record linkage, attribute linkage, table linkage and probabilistic attacks) are considered, amounting to make two fundamental assumptions: (i) what an attacker is assumed to know about the victim; (ii) under which conditions a privacy threat occurs.

Before reaching to specific methods (notably the ones surveyed in [Fung10] in addition to more recent developments), we focus on the agnostic principles of data anonymization and PPDP, i.e. notions and rules that are valid whatever the type of

database and the techniques or languages used for data modeling, data management, and anonymization. We then split a group of methods following [Fung10], grouping methods in two broad categories: techniques preventing record or table linkage, versus methods preventing probabilistic attacks. The first part will see the study of *k*-anonymity and its main derivatives, showing what additional improvements and guarantees they provided; in the second, we will tackle more advanced models and differential privacy along with its derivatives and other recent probability-based statistical anonymization methods. Finally, we study more specific methods dealing with graph anonymization and RDF anonymization.

2.1.1 Generic database anonymization

First, we shall explore some general and language- or framework-agnostic principles of data anonymization, with its main objective in mind: looking for a good *privacy/utility balance*. Major studies from the 2000s deal with the case of *data-mining utility* and *data publishing*, that is how well you can still find relevant information in a published datase. They then try to design ad-hoc logical boundaries between data privacy and data utility in this context. In [Rastogi07], Rastogi et al. define a boundary between membership disclosure of a tuple in a set and approximations of counting queries, showing that if the external knowledge is too high, then no anonymization is possible. Brickell and Shmatikov study in [Brickell08] how *k*-anonymity-like systems fare (as opposed to trivial removal of quasi-identifiers) using data mining algorithms as utility measurements. In another theoretical direction, Kifer and Machanavajjhala argued that it is not possible to guarantee both privacy and utility without any guarantee regarding how the database's contents are generated [Kifer11].

Rather than principles and guarantees, there have been many tries to define generic privacy frameworks as well, such as the Pufferfish framework [Kifer14] which is a customizable, generic Bayesian probability-based privacy framework relying on the input of a domain expert defining "secrets". Using this framework, it is therefore possible to encompass and describe many existing statistical privacy frameworks such as variations of differential privacy (see Section 2.1.3). This was further extended to other frameworks, notably *blowfish privacy* [He14]. The latter is based on the differential privacy principles and the Pufferfish framework, allowing PPDP through the use of a "policy" specifying both *secrets* to hide from the database, and *specific constraints* known about the data, used to prevent attacks based on knowledge of correlations in the data.

Another class of techniques embraces the point of view of, rather than anonymizing the data itself, blocking the access to specific portions of data when it is queried, depending on various factors such as permissions and user control. These techniques are called **access control** methods and have been extensively studied for many decades, and in many various subfields as well [Squicciarini15]. As opposed to other methods studied in this section, these privacy mechanisms are interactive, as privacy is "decided"

when specific tuples from the database are requested. Other classes of anonymization mechanisms that we will not explore place anonymization mechanisms at other steps of data publishing; for example, *input perturbation* methods make each publishing entity anonymize independently sanitize its data, as pioneered in [Mishra06]. Finally, some other mechanisms do not rely on data anonymization, but rather on secure query answering [Roth10] which may cause its own types of vulnerabilities.

2.1.2 Protecting individual records: preventing linkage attacks

The first modern approach to database anonymization came from preventing the re-identification of personal records or sensitive information. Indeed, traditional practices for "anonymizing" datasets usually consisted in roughly deleting what were though to be sensitive information or *identifers*, i.e. fields that are deemed unique (or non-anonymous) for an individual or an entity: full name, social security number, ID card number, and so on. This is therefore a problem targeting published data on an individual scale (among big, aggregate sets of data) usually named *microdata*[1].

In the late 1990s and early 2000s, P. Samarati and L. Sweeney showed how the use of so-called *quasi-identifiers* could be used to de-anonymize datasets that were previously stripped of their identifying columns only by joining them using their common data fields, as shown on Figure 2.1.

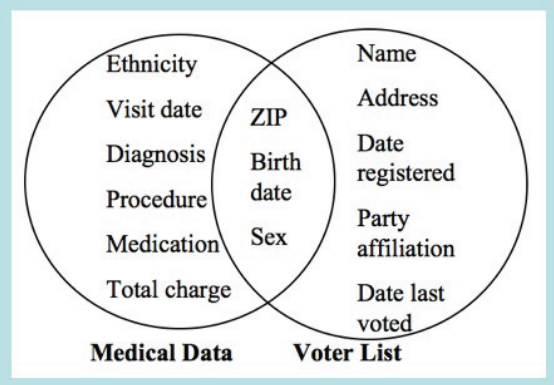


Figure 2.1: Using quasi-identifiers to de-anonymize medical records using a list of registered voters. From L. Sweeney [Sweeney02b]

They therefore designed a privacy framework to prevent this type of attack: k-anonymity [Samarati01, Sweeney02b, Sweeney02a]. In itself, these original k-anonymity propositions encompass two things: a definition of privacy, and how to achieve it. A dataset release is k-anonymous if each record (or individual, in the case of personal data) in a dataset is *indistinguishable* from at least k other records. This is

[1]OECD definition: https://stats.oecd.org/glossary/detail.asp?ID=1656

done by editing values in the database, using two distinct operations: *generalizations* (where a value is replaced by a more general value, e.g. an age is replaced by an age group, or a disease by a type a disease) or *suppressions*, where a value is simply deleted. Operations can be applied to all the values of a column, or only a portion of them.

While *k*-anonymity is one of the most primitive data privacy model in data management or data publishing, it is also one of the simplest, and one of the easiest to understand. This means that despite more modern techniques being developed for decades, and despite many improvements or modifications to this "indistinguishability" class of privacy frameworks, it is still widely used by a general audience and taken as example of reference when dealing with data anonymity. In 2018 for example, the *k*-anonymity model was used by many password-related websites and password managers to check if a given password had been leaked on the Internet[2]. Recent research is still sometimes based on *k*-anonymity as well [Wang19].

Given the pioneering and fresh aspect of *k*-anonymity, it spawned numerous derivative models in the following years and still continues to do so. Some are contextual adaptations or complements of the basic model (sometimes for specific contexts), such as a partial *k*-anonymity model for social network data [Liu17], or an extension to better handle hierarchical data [Ozalp16]. But many also designed improvements in order to address flaws in this very simple framework. We explore the most notable ones of the latter category in the following.

An interesting aspect to survey is the one regarding extension or augmentation of *k*-anonymity, rather than directly seeking for derivative methods. While there have been many simple (i.e. based on the basic generalization and suppression principles) implementations of algorithms or methods designed to reach *k*-anonymity in a data release [Sweeney02a, Iyengar02, Jr.05, Wang04, Fung05], some tried to define more complex or efficient solutions. For example, Lefevre et al. [LeFevre06a] designed the Mondrian algorithm, a greedy algorithm based on a multidimensional partitioning of data in order to reach *k*-anonymity that is both more efficient in terms of performance, and in terms of utility loss (with regard to the evaluation of aggregate queries). Recent developments from Y. Tong and J. Wang also focus on *identity* as a level of granularity rather than *record*: this covers the case when an individual may have several records in the same database by defining identity-reserved anonymity models and associated algorithms [Wang19].

These simple PPDP algorithms proved vulnerable to so-called *minimality attacks*, where knowledge of the generalization algorithm used can create loopholes and leakages, as detailed by Wong et al. [Wong07] and Zhang et al. [Zhang07]. Several paths were taken afterwards: some designed mechanisms to thwart attacks based on prior knowledge of the anonymization algorithms used [Jin10, Cormode10, Xiao10], some found out other vulnerabilities in this basic model, such as using learning algorithms

[2]See recent news regarding 1Password, Okta, Have I Been Pwned?, or even Google's Password Checkup add-on.

over sanitized data [Kifer09], and some defined some new, stronger models, that we will study in the rest of this section.

One of the main follow-ups of k-anonymity is l-diversity [Machanavajjhala07], which aims at protecting against weaknesses of k-anonymity. Notably, how two things affect anonymization: data homogeneity in generalization (what if all the values generalized are the same, or close, to begin with?), and prior knowledge of quasi-identifiers or sensitive values (what if an attacker knows that a given individual has a high probability of having a certain value for a given predicate?). To prevent these attacks, l-diversity is achieved when an equivalence class has at least l "well-represented" values (e.g. at least l different values appear; but other constraints can be used). This, in short, means that any entity can be linked with an equal probability $\frac{1}{l}$ to any of the l sensitive values.

An example of alternative technique allowing k-anonymity and l-diversity, and therefore a complement to these generalization and suppression techniques is the *anatomy* privacy model by Xiao and Tao [Xiao06]. This is a technique where the data is grouped into multiple buckets, and then split in two: on one side the quasi-identifiers, and on the other the rest of the data, those two parts being linked by a new column, indicating which record belongs to which bucket. This approach is easier than the original operations of the Sweeney model, but its efficiency with regard to data utility is unclear compared to the original framework.

The next major development following in line the l-diversity model is the t-closeness model [Li07], which focuses on the data distribution: in short, the goal is to make sure that even when values have been deleted or generalized, the new value distribution for a given predicate is similar (to an extent provided by the threshold t) to the original distribution, since some attackers might infer sensitive values if they have prior knowledge on their distribution and if the anonymization is too deterministic. Again, this induces a loss of utility (in terms of query answering or data management) in order to give more privacy guarantees in particular cases of attack. A running theme of all these incremental improvements is that they all focus on attacks induced by the prior knowledge of an attacker, as opposed to the early, simplest models where this knowledge is supposed to be very limited. In that regard, t-closeness starts a line of models designed to prevent probabilistic attacks, based on what is known as the *uninformative principle* [Machanavajjhala07]: an attacker should gain as few knowledge as possible from the anonymized release of a dataset. While k-anonymity-based techniques are not the most efficient to respect this principle, it will be the guiding light for the next family of techniques that we will explore (aimed at guaranteeing statistical integrity against probabilistic attacks, see Section 2.1.3) and for a few more advanced techniques.

Another similar idea, following t-closeness, was the δ-disclosure model [Brickell08]. In this case, we not only consider knowledge about the data distribution as a weakness, but the goal is to quantify the information possibly usable by an attacker knowing about

it, based on the equivalence classes of sensitive values. Suppose that each sensitive value in the database has a frequency f_0 in the dataset, and a frequency f_{eq} in a given equivalence class. Here, this equivalence class is δ-disclosure-private if and only if for each sensitive value, $log(f_{eq}/f_0) < \delta$. This restriction also enshrines an information theory result about the restriction of knowledge gained by an attacker when this δ-disclosure is respected. But this method has numerous drawbacks, notably the fact that sensitive values may not always be in each equivalence class, and the fact the utility has to be greatly reduced in some cases for this property the hold.

This list can easily go on and on, as methods have been continuously developed (especially in the mid to late 2000s) to address each other's drawbacks, such as β-likeness related to erratic privacy behavior even with a fixed closeness value [Cao12] or enhanced models for databases where quasi-identifiers and sensitive attributes overlap multiple times [Sei19], both tackling limitations of t-closeness. In other cases, some methods are designed to address other, more specific contexts of background attacker knowledge: among numerous others, we mention *skyline privacy* where external knowledge is modeled in a multidimensional fashion, and an adequate privacy criterion [Chen07]; privacy models where the background knowledge is assumed to be the worst possible and unknown [Martin07]; or ϵ-privacy, a probability-based model against "realistic" (i.e. not particularly weak or strong) adversaries [Machanavajjhala09].

Finally, we explore one last type of k-anonymity variation that will help transitioning to differential privacy: δ-presence [Nergiz07a], whose goal is to protect against guessing the presence of an individual in the database. In that regard, its goals are the same as differential privacy, but with a classical, non-statistically-based outlook. It suggests to bound the probability of deducing the presence of any individual record within a specified range δ, using generalization operations on the data. This can prevent linkage attacks as well, because if the attacker has at most δ% of confidence that a target's record exists in the data release, this means that the probability they can retrieve the target's sensitive attributes is at most δ% as well. The main drawback of this method, in addition to the usual utility loss induced by these restrictions, is the requirement that the data provider has in theory access to the same external knowledge than any potential attacker.

2.1.3 Protecting statistical integrity against probabilistic attacks

With time, considerations shifted to statistical and aggregate databases, as they were considered to be more realistic approach in terms of current data usage in the time of Big Data. Even when dealing with microdata, more statistical attacks and approaches were considered, such the de-anonymization attacks on the Netflix Prize dataset by Narayanan and Shmatikov [Narayanan08]. These frameworks focus on the knowledge of an attacker, and model it as a probability of discovering a targeted sensitive value.

Formally, they are based on how the attacker would change their probabilistic belief on the sensitive values related to a target when looking at a data release. Note that these methods tend to be very specific to numerical or statistical databases, for example by using distance metrics. So in addition to their goals being different, their applicability is also changing compared to the previously studied techniques.

Privacy with guarantees against probabilistic attacks is often associated with *differential privacy*, but this type of framework was considered even before that. Indeed, in parallel with the development of differential privacy, many methods were based on the fact that an attacker should not gain too much knowledge in accordance to the uninformative principle mentioned on the previous section. One of the first was (c, t)-isolation [Chawla05], aiming at defining a privacy model for statistical databases, one of the foundations of differential privacy and probabilistic methods. Preventing (c, t)-isolation in a statistical database means that for a target value of an attacker P and the data inferred by an attacker Q for a given point in the dataset, distant of a value d, a ball of radius $c \times d$ centered on Q contains less than t points in the dataset.

Another work of note regarding probabilistic privacy is the (d, λ)-privacy and its associated algorithm designed to achieve it, the $\alpha\beta$ algorithm [Rastogi07]. In this mode, the knowledge gain of an attacker (its probability of finding a sensitive attribute) when looking at an anonymized release of the data is bounded. When the prior probability is small, an optimal privacy/utility balance can be found, but this can be rarely achieved in practice. Besides, it only protects again attackers who either know that a specific record exists in the dataset (prior probability is 100%, so no protection needed), or that they don't know much about its existence (prior probability lower than d, which can be a low threshold), which is also not always plausible [Machanavajjhala08].

One of the most influential work on privacy from the recent years if the design of the *ϵ-differential privacy* criterion by C. Dwork [Dwork06a]. The intuition is simple, and takes another path from usual methods: there should not be an increased risk of privacy leakage whether or not an individual (or entity) is included in a statistical database. This models therefore studies the knowledge not before and after anonymization, but with or without a record in the dataset. A *differentially private* data release therefore guarantees that removing or adding a single record would not change the statistical integrity or analysis of the data. While this is the notable advantage of protecting against dataset linkage and against attackers whatever their prior knowledge, it also amounts to the possibility of having fake data going unnoticed in a data release; one could argue that this is not an important requirement for aggregate studies, rather than microdata concerns. Dwork also provided a mechanism based in the introduction of Laplace noise in the dataset in order to achieve privacy [Dwork06c]. Other mechanisms subsequently developed include randomized responses, an approach inspired by an old survey anonymization technique [Warner65, Wang16], a "staircase" mechanism based on geometrically stacked uniform random variables [Geng15], and an exponential mechanism, where noise is distributed exponentially [Dwork06c, McSherry07].

Dwork's efforts quickly spawned numerous applications and derivatives, as exemplified by Dwork herself in a survey only two years later [Dwork08]. But rather than inducing incremental changes like k-anonymity did, differential privacy was quickly adopted as a *de facto* standard to uphold, and was instead analyzed and discussed on many different sub-fields which in turned tried their luck at designing algorithms to achieve a differentially private data release in their scope of study. This also means that surveys on differential privacy as a whole are rare (the main survey being now 10 years old [Dwork08] and the other main survey from the recent years being only available in Chinese [Xiong14]) or tend to be informal, or focused on application on a given sub-field (such as machine learning [Ji14]). Still, some tried to tighten or relax the differential privacy criterion to build other privacy models; we now explore some of these works.

Dwork herself et al. suggested a relaxation of ϵ-differential privacy, providing a probability "error margin" bounded by a value δ: this is the (ϵ, δ)-differential privacy generalization [Dwork06b]. This provides a relief for "unlikely" events that could easily break the initial, very strict definition of differential privacy. Another recent generalization and relaxation of the original ϵ-differential privacy based on the Rényi divergence (a generalization of the Kullback-Leibler divergence, measuring a distance between probability distributions) is the (α, ϵ)-Rényi differential privacy [Mironov17] where α is the dimension used to compute the distance. (∞, ϵ)-Rényi differential privacy is equivalent to ϵ-differential privacy, while $(1, \epsilon)$-Rényi differential privacy would be validated using the the Kullback-Leibler divergence. This approach allows for "bad outcomes" to happen without a margin as big as the original δ factor, while having the same composition power as other differential privacy methods. Another modern relaxation concurrently developed is *concentrated differential privacy*, designed by C. Dwork and G. Rothblum in [Dwork16] and studied in [Bun16], where privacy is achieved if the random variable modeling privacy loss has a small mean and follows a sub-Gaussian distribution.

While there is still some effort to develop new approaches or privacy models enhancing the original differential privacy or its derivatives, the current trend tends to focus on data management or data release of already differentially private datasets, as surveyed in [Zhu17]. Another trend is the design of *ad hoc* frameworks, based on differential privacy, but adapted to specific contexts, e.g. governmental data [Piao19], location data for connected devices [Zhou19], or Internet of Things data [Ghayyur18].

We highlight a few algorithms ensuring ϵ-differential privacy in specific contexts. One of the most relevant is the general PPDP algorithm from [Mohammed11], providing an algorithm based on generalization operations and uncertainty in a non-interactive context, guaranteeing enough data utility to use classifying data mining algorithms on the sanitized data. Other famous applications include the context of recommendation systems [McSherry09] building on the infamous Netflix data leakage stated in [Narayanan08] and mentioned earlier, as well as potential solutions to bring

privacy in this database; Rastogi and Nath provided a differentially private aggregation algorithm for time-series data, using a perturbation mechanism based on Fourier noise [Rastogi10]; Chen et al. proposed a differentially private anonymization algorithm for sequential transit data [Chen12]; finally, Xu et al. suggested a framework for publishing data histograms satisfying differential privacy, with two possible noise mechanisms depending on the desired focus [Xu13].

Another recent line of work is trying to reconcile k-anonymity-based approaches along with differential privacy. For example, Soria-Comas et al. have shown that using a k-anonymity approach on a dataset, based on micro-aggregation, could then reduce the amount of noise necessary to reach ϵ-differential privacy [Soria-Comas14]. This noise is indeed added to each aggregated equivalence class. This approach then evolved into strict t-closeness guarantees compatible with the differential privacy criterion [Soria-Comas15].

2.1.4 Graph database anonymization

Most of the methods mentioned so far are based either on abstract designs of databases, or on traditional relational databases, where a dataset is a set of records, those records being tuples where each values corresponds to a predicate taken as a column. To cater to the particularities of LOD, it might be more relevant to study privacy mechanisms based on graph databases, since RDF is heavily shaped on this model. Graph anonymization is closely related to social networks anonymization, their influence getting bigger and bigger since the beginning of the decade, and their modeling as graphs being intuitive (each node is a person or an organization, and they are interlinked by various relationships: friendship, family links, retweets, media sharing, etc.). Privacy in social graph networks has been heavily studied by now, and surveyed multiple times ([Zhou08b, Zheleva11, Wu10, Ding10], and more recently in [Kayes17]). Recently, some more general surveys on graph privacy also emerged [Ji17, Casas-Roma17]. We therefore, once again, only select the most influential and relevant portion of these works.

There are two visions of graph database anonymization. The first is to adapt existing techniques, such as the ones from the previous sections, and study the behavior of such algorithms or principles induced by this structural changes. The second is to develop new techniques, sometimes inspired by previous work, to natively handle nodes and edges instead of records in a table. In a slightly different fashion, we will focus on two aspects: we begin by structural anonymization techniques (such as adaptations of k-anonymity and derivatives, working directly on nodes and edges) and conclude with various adaptations of differential privacy for graph databases. For other existing approaches, to ease readability we adopt the six-folded classification from [Ji17], since it already includes some categories we studied in previous sections. This decomposition is as follows:

- *Naive removal of identifiers*: this is roughly equivalent to the pseudonymization techniques used in relational databases where only columns containing sensitive values are deleted. Intuitively, even on graph databases, these are vulnerable to many attacks such as equivalents of the linkage attacks or learning-based approaches mentioned earlier [Backstrom11, Narayanan09]. These *structural-based attacks* are among the most common in social network study.

- *Edge editing techniques*: Ying and Wu suggested models based on edge editing, notably one where a fixed number of existing edges are deleted and the same number of edges are created between random nodes [Ying08]. While this approach has the advantage of preserving the spectrum of the graph, it introduces wrong information and can represented a heavy loss of utility in terms of microdata release or data mining. A specific type of edge editing name are the *random walk-based approaches* first developed by Mittal et al. [Mittal13], where an edge between two entities e_1 and e_2 is replaced by an edge between entities e_1 and e_R, where e_R is the destination of a random walk on the considered graph, starting from e_1.

- *Clustering- or aggregation-based techniques*: in a bit of a similar fashion to k-anonymity (since it induces indistinguishability between entities), many techniques are based on clustering users or entity in groups or classes. For example, it is possible using partitions to create a "generalized" graph consisting of edges of various density between groups of nodes [Hay08] or to create classes of entities based on the label of the edges between them, while preventing inference leakages [Cormode09].

- *k-anonymity-based techniques*: see Section 2.1.4.1.

- *Differential privacy-based techniques*: see Section 2.1.4.2.

It should be noted that many other research tracks regarding privacy in graphs exist, some of them not being focused on anonymizing graphs themselves, but for example on the reverse process (*de-anonymization attacks*), or on the criteria required to be anonymous or vulnerable (*de-anonymizability analysis*). Attacks tend to be focused on structure-based attacks given the specificity of graph databases and following the initial work from Narayanan et al. [Narayanan09], and try to find unique characteristics of each entity found in the graph to de-anonymize it. These attacks can be based on a set of pre-identified entities called *seeds* (such as in [Narayanan09, Korula14]), or not (like in [Pedarsani13, Ji16a]). Regarding de-anonymizability, which in itself is also slightly out of the scope of this thesis, some work was also performed using seeds [Ji15, Korula14], but since attacks have been possible without seed data, de-anonymizability should also be considered without seeds. Some conditions based on the structure of the anonymized graph and the external graph considered were established [Pedarsani11] while Ji et al. studied how to quantify this de-anonmyzability, based on each entity's structural importance in the graph [Ji16b].

2.1.4.1 *k*-anonymity-based graph anonymization approaches

Various basic implementations or adaptations of *k*-anonymity to graph databases were suggested, starting at the end of the 2000s. The first works were designed to identify the most intuitive attacks, and to prevent them through mechanisms similar to *k*-anonymity techniques on relational data. The basic attacks found were *neighborhood attacks* [Zhou08a] where a node would be recognized based on its neighbor nodes, and *degree attacks* [Liu08] where a node could be identified using its outgoing and incoming edges; these issues can be respectively solved simply by grouping nodes with similar neighbors and by creating an anonymized graph where at least k nodes have the same degree. Some more advanced attacks were then considered, such as attacks using subgraph enumeration and patterns of "hubs" (nodes that are central to the graph due to their links with other nodes) [Hay08], or linkage attacks [Nobari14] where some tighter techniques were necessary, notably using strong structural similarity between parts of the graph, such as the *k*-automorphism [Zou09] and *k*-isomorphism [Cheng10]. Both criteria protect against all these structural attacks by ensuring respectively that each entity in the graph has $k-1$ symmetric entities with respect to $k-1$ automorphic functions, or that a graph is basically disjointed into k equivalent automorphisms or isomorphisms.

It quickly appeared that *k*-anonymity on its own was not suitable for graph databases, and was easily vulnerable to many structure-based attacks [Pedarsani13, Narayanan08] because, even if entities in the graph were protected from specific structural attacks, it is still possible to compute other graph analytics to identify nodes (e.g., centrality, path length statistics, combined metrics, betweenness, etc.). Therefore, the only way to preserve a lot of privacy with this type of approach is to separate sensitive data from the rest, in terms of connectivity; which destroys data utility.

2.1.4.2 Differential privacy-based graph anonymization approaches

Similarly to *k*-anonymity methods, adaptations of differential privacy for graph databases were quickly set up at the end of the 2000s. These approaches are mostly focused on PPDM, most notably on releasing specific statistics while being "private" (without leaking data in the process): usually a value or a vector in recent works [Raskhodnikova16], or things like the degree distribution of a graph [Hay09]. This is as opposed to releasing the whole graph, and works mostly for social network analysis, since as we have seen earlier this is a big part of current studies on graph databases. Most of these techniques work by computing sensitivity studies, notably smooth sensitivity [Nissim07], that is computing the variability of the output of a function providing a release of the data of some sort.

Differential privacy variants for graph databases are usually classified in two: *edge* differential privacy, where the neighboring graphs considered should differ with only

one edge, and *node* differential privacy where one of the considered graph can be obtain using the other by *deleting a node and its edges*. The first works, at the start of the decade, were based on reaching edge differential privacy when providing various statistics such as the degree distribution of the graph [Hay10, Lin13], the number of copies of a small subgraph in the original graph [Karwa11], or the cut of a graph [Gupta12]. Some analog works at the time also decided on slightly changing the original privacy model, sometimes creating models that were *weaker* (such as considering only Bayesian adversaries [Rastogi09]) or *stronger* (such as a "zero knowledge gained" model still able to provide degree computation and neighborhood for some graphs [Gehrke11]) than the original differential privacy.

Newer techniques are now focused on achieving *node differential privacy* for any input graph, and can provide better algorithms for the edge counterpart as well while still providing usual utility metrics regarding subgraphs or degree distribution. These recent methods are based on using finer notions of sensitivity [Blocki13, Kasiviswanathan13] or new mechanisms such as a recursive algorithm to answer linear queries [Chen13].

In an other direction, some tried to define *differentially private graph models*, such as the Pygmalion graph model [Sala11] where a graph is modeled by the degree distributions grouped by connected components of the same size in a target graph; this statistical output is then perturbed to achieve classical ϵ-differential privacy. Another model proposed by Proserpio et al. focuses on graph topology metrics [Proserpio12].

2.1.5 RDF and Linked Data anonymization

The sub-field of data anonymization and PPDP related to this thesis is the one regarding *Linked Data and RDF anonymization*; this means that we will be able to compare more directly the work presented in this manuscript with existing works. But the literature touching the subject is rare, and sometimes lacking. In short, logical frameworks for privacy-preserving Linked Data publishing were briefly studied, but were not investigated further than logical foundations and theoretical problems. We analyze how the main aspects of our contributions were previously explored, if possible in the context of Linked Data, but otherwise in other relevant contexts.

2.1.5.1 Data privacy and utility concerns for the LOD

Most of the models we explored in the previous sections, first conceived for relational databases, have been recently extended to the setting of the Semantic Web [Kirrane18]. Among them, the privacy model that is definitely the closest to our work is k-anonymity and its derivatives, for which there has been recently some partial work to adapt it for RDF graphs [Radulovic15, Heitmann17]. These works focus on defining operations of generalization, suppression or perturbation to apply to values in the range of properties

known to be quasi-identifiers for personal identification, along with metrics to measure the resulting loss of information.

An alternative approach to anonymization for protecting against privacy breaches consists in applying access control methods to Linked Data [Oulmakhzoune12, Villata11, Kirrane17, Endris18]. In the Semantic Web setting, when data is described by description logics ontologies, preliminary results on role-based access control have been obtained in [Baader17] 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. Compared to access control techniques that perform verification at runtime, we focus on a static analysis approach executed only once and guaranteeing that the published datasets do not contain sensitive information.

2.1.5.2 Linkage concerns

In line with existing works on the subject [Deutsch05, Miklau07, Grau08] on safety models defined in terms of secret or privacy queries for relational data, or logic-based information systems, a query-based safety model for RDF data has been introduced in [Grau16, Grau19] where linking RDF graphs is reduced to their union. Several results are provided on the computational complexity of the decision problem consisting in checking whether an anonymization of an RDF graph is safe with regard to a given privacy *policy*, i.e. a given query whose results must not leak information.

Privacy-preserving record linkage has been recently considered in [Vatsalan17] as the problem of identifying and linking records that correspond to the same real-world entity without revealing any sensitive information about these entities. For preserving privacy while allowing the linkage, masking functions are proposed to transform original data in such a way that there exists a specific functional relationship between the original data and the masked data. The problem of privacy-preserving record linkage is a difficult problem that is significantly different from the privacy-preserving data publishing problem considered in this thesis, in which `sameAs` links are input of the anonymization process.

2.2 Position of this thesis

As Chapter 1 and Section 2.1 show, it turns out that concrete and usable frameworks for fine-tuned anonymization of RDF data and for the LOD are rare. Previous efforts are focus rather on theoretical guarantees and logical modeling, or on rough adaptations of existing anonymization techniques for relational or graph databases, which do not fully address the semantic particularities of Linked Data. To remedy this, we decide on building on the logical framework introduced by [Grau19] where privacy is expressed in a declarative manner as conjunctive queries composing *policies*. We then

extend this declarative framework by modeling utility through queries too, as well as the anonymization operations as update queries.

We now develop the main distinguishing points that our approach requires and that will be developed in our contributions.

Generic and data-independent approach of anonymization

A typical caveat or sacrifice that anonymization frameworks have to suffer is their heavy data-dependence. Indeed, an intuitive way to anonymize a given database would be to analyze the entirety of its records and sequentially decide how to process and sanitize it based on the observed values. But this implies a complete knowledge and availability of the data, and is computing-heavy. The same goes for frameworks where a specific anonymization process is applied to specific unsafe values, or where anonymization operations are conditioned by the number of records possessing a specific value for a given predicate.

Besides, RDF graphs in general can get really big, even when a shortened syntax like Turtle is used. For example, considering only the interlinked graphs from the LOD cloud in 2017 amounted to a dataset whose size was over 500GB, and with more than 28 million triples [Fernández17]. For such huge sets of data, a data-dependent solution would therefore amount to a very long running time, and would also lead to more complex configuration and pre-processing to be done by the end user. Indeed, there would be additional customization and optimization for each graph to be anonymized, which would imply additional time spent on researching optimal settings and analyzing the contents of the graph to look for specific adjustments.

Therefore, we decide on designing a **static, data-independent solution**: the framework we intend on creating must allow for the anonymization of *any valid RDF graph*, whatever its contents or its schema.

Query-based modeling of privacy/utility policies and anonymization operations

In order to be precise and fine-tuned, anonymization requires *rules*: these principles can be materialized in many ways and formats. They can be the result of a spoken discussion between business-related and IT-related services, they can be automated or automatically generated (in particular when the anonymization is done in a data-dependent fashion), based on the data or on its structure. These rules must be simple, to be both human- and machine-readable in the best possible way, but still need a subsequent amount of expressiveness to designate subtle or specific parts of the data that must be anonymized. Besides, the DPO (in the case of an RGPD-compliant publisher) is supposedly the one person knowing the most about privacy requirements regarding the data the provider wants to publish; simple declarative constraints would therefore give them the possibility to express these requirements.

To achieve this, we focus on providing a simple way to express constraints on the final data (to be published to the LOD cloud), preferably in a way requiring as few con-

versions and processing as possible to be implemented simply in an RDF anonymization framework. This means **using something as clear as possible in terms of technology and language** for the data provider.

The same thought process is valid for the anonymization operators computed by the framework as well: while they do not need as much readability as the data disclosure constraints, they have to be understandable as well (e.g. for debugging or quality control).

Native operators and tools

While using declarative constraints might be useful for the data provider, it might not be more easily machine-readable for the anonymization software used; that is why it should also be required to as many native technologies as possible, and as many official standards (here, W3C recommendations) as possible. This is again valid as well for anonymization operators. Indeed, to ease the design of the software pipeline and to ensure functionality and efficiency, they also need to be as native as possible in terms of format and processing. This is the other side of the spectrum mentioned in the previous paragraph, meaning that we need to **use formats and software which are the closest we can to native RDF and SPARQL**.

To succeed in implementing each of these requirements, we intend on leveraging the logical tools designed in [Grau19] and on building on this basis to design concrete privacy guarantees as well as concrete algorithms and implementations.

Compared to general LOD anonymization methods presented in Section 2.1.5.1, our approach is more generic and also fully declarative since it allows the definition of fine-grained privacy policies specified by queries, and to obtain candidate sets of anonymization operations allowing to practically enforce the requested privacy without loosing the desired utility.

In contrast with existing approaches based on k-anonymity, we will focus on generalizations that replace constants by blank nodes. We also focus on guaranteeing *safe* anonymizations, a subject not explored in these simple adaptations. Techniques focused on linkage issues, such as the ones mentioned in Section 2.1.5.2, are therefore the closest to our approach, with the notable exception that in our contributions, we slightly extend the considered safety model and we address the data-independent construction problem underpinning safety, i.e. how to produce a sequence of update operations that are safe *for any RDF graph*, given a privacy policy expressed as queries. We also include the notion of utility policies and sameAs linkage in more pragmatic terms. To the best of our knowledge, ours is the first approach providing generation of anonymization operations for ensuring safety under a set of privacy queries.

3

Context and formal background

▷ *After exploring our positioning among the existing literature, we go more in depth in the current context of the Semantic Web and Linked Open Data, detailing both its W3C technical standards and how we formalize it in terms of logical and theoretical objects.* ◁

Chapter summary

BASED on the goal of this thesis and the previously developed state of the art, we now lay the detailed technical context, as well as the logical and theoretical foundations of all tools used in future contributions. This includes definitions of classical RDF, SPARQL, and query-related concepts, as well as their equivalencies with current W3C standards. Section 3.1 provides a concrete overview of usual standards and tools, before detailing their logical representation in Section 3.2.

3.1 Linked Open Data standards

The W3C writes and maintains various standards and documents for Linked Open Data and Semantic Web notions[1], whether they are languages, usage recommendations, lists of best practices, glossaries, vocabularies, or primers. In this context, this includes both formal definitions and grammars of RDF data and SPARQL queries, as well as various RDF syntaxes.

3.1.1 RDF

RDF (Resource Description Framework) [Wood14] is a framework intended to model information or knowledge regarding *resources*, which can be anything: moral concepts, persons, institutions, abstract ideas, websites, physical documents, and so on. Its goal is therefore to model as much as possible of one's knowledge regarding anything, in a way that is processable by software applications, servers, or any automated system. RDF uses a *graph-based data model*, where data is modeled as *triples* consisting of a *subject*, a *predicate* (or *property*) and an *object*. Sets of triples are therefore akin to *graph databases*. Each node from the triples of an RDF graph is either:

- an **IRI (Internationalized Resource Identifier)**[2], a string identifying a resource globally, formatted following [Duerst05];
- a **literal**, i.e. a raw string, a numeric value or a date value (possibly explicitly typed using specific *datatype IRIs* taken from XSD, the XML schema specification language [Pan06]);
- a **blank node**, i.e. a *local* and *anonymous* resource identifier.

For instance, and following our running example presented in Section 1.5, we define a simple graph consisting in the triples contained in Table 3.1, schematically represented on Figure 3.1.

[1] See `https://www.w3.org/TR/?tag=data` for an exhaustive list.

[2] Not to be confused with **URIs (Uniform Resource Identifiers)** or **URLs (Uniform Resource Locators)**: IRIs are an extension of URIs, since URIs only use ASCII characters as opposed to the whole Unicode set of characters for IRIs. On the other hand, URLs are a subset of IRIs/URIs, denoting the address of a web resource on a given network.

Table 3.1: Example of a simple RDF graph.

Subject	Predicate	Object
http://example.org/v26830	http://www.w3.org/1999/02/22-rdf-syntax-ns	http://example.org/Validation
http://example.org/v26830	http://example.org/validationDatetime	"2017-04-27T19:46:51"
http://example.org/v26830	http://www.w3.org/2003/01/geo/wgs84_pos#latitude	"45.7599724233"
http://example.org/v26830	http://www.w3.org/2003/01/geo/wgs84_pos#longitude	"4.82594480511"
http://example.org/v26830	http://example.org/user	http://example.org/u9590
http://example.org/u9590	http://www.w3.org/1999/02/22-rdf-syntax-ns	http://example.org/User
http://example.org/u9590	http://xmlns.com/foaf/0.1/givenName	"Germon"
http://example.org/u9590	http://xmlns.com/foaf/0.1/familyName	"Anthony"
http://example.org/u9590	http://www.w3.org/2006/vcard/ns#hasAddress	"180 Rue A 69XXX Curis-au-Mont-d'Or"
http://example.org/u9590	http://example.org/birthday	"1946-03-26"

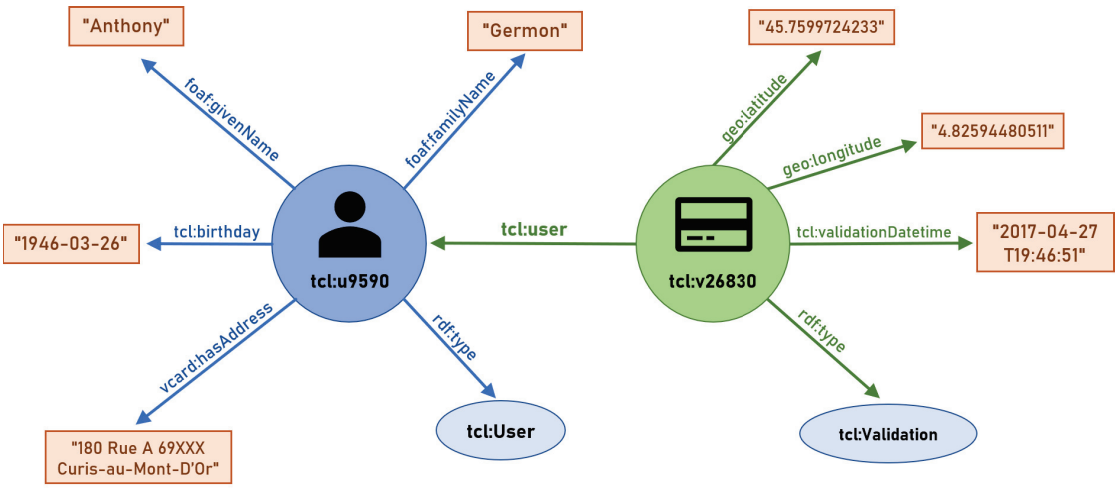


Figure 3.1: Graph visualization of the RDF graph from Table 3.1.

The set of all IRIs used as predicates in a graph is called the *schema* of an RDF graph. Generally speaking, collections of predefined IRIs named *vocabular-*

ies are used to write RDF graphs, and their usage and readability can be simplified using *prefixes* indicating namespaces. On the example RDF graph represented on Figure 3.1, IRIs from Table 3.1 are shortened, e.g. `rdf:` is a shorthand for `http://www.w3.org/1999/02/22-rdf-syntax-ns#` and `foaf:` replaces `http://xmlns.com/foaf/0.1/`. This also helps identifying the original vocabulary of a given IRI in a more readable way. Prefixes can be customized are not set in stone, for example one could use `customprefix:` as a shorthand for `http://www.w3.org/1999/02/22-rdf-syntax-ns#` instead of `rdf:` if they desire it, but it is not optimal. Some namespaces are nonetheless quasi-standardized by usage, and online directories are available to find the most common ones[3].

RDF triple stores can describe multiple graphs, and standards provide the notion of *RDF datasets*, which are sets of optional named graphs (the name being an IRI as well), with a mandatory unnamed default graph. In this case, RDF triples can effectively be considered as *RDF quads*, where a graph identifier IRI is added to the original subject, predicate and object.

RDF can be written in many different syntaxes, and serialized in various formats as well. The W3C defines for example *RDF/XML*, which is a way to represent RDF data in XML format [Schreiber14]. The W3C, for example, also defines concise plain text syntaxes such as Turtle [Carothers14] or N-Triples [Seaborne14], and other serialization formats such as a JSON syntax for RDF [Kellogg14]. Non-standard syntaxes include for example RDF encoding for standard HTML forms[4], a plain text syntax for encoding microblogging data[5], or dataset/quad-oriented syntaxes such as TriX [Carroll04]. Throughout this manuscript, we will use the Turtle syntax, where prefixes can be used, and repeated subjects or properties can be omitted when writing triples.

One of the main aspects of RDF graphs and an advantage of its simple structure is the fact that RDF triple stores can infer new information from existing triples, using automated *deductive reasoning*. These reasoners can work in RDF using the two main types of inference: *forward chaining* (deduct new triples by applying inference rules on existing facts and iterate this process until no new triples can be created) and *backward chaining* (proving a fact or answering a query using existing facts).

3.1.2 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [Harris13] is the query language defined by the W3C mainly used for the interrogation and edition of RDF graphs. It is inspired by relational query languages such as SQL and is used to query RDF data sources, on one or multiple graphs, whether they are local or external. The main language is the query language, but the W3C extended SPARQL to include an update

[3] For example, the Prefix.CC search engine.
[4] `http://www.lsrn.org/semweb/rdfpost.html`
[5] `http://buzzword.org.uk/2009/microturtle/spec`

language as well [Polleres13] commonly named **SPARQL Update**[6], therefore making SPARQL usable to complete all CRUD operations (creating, reading, updating and deleting data from RDF graphs, or entire graphs themselves). The current version is SPARQL 1.1, which is the version used as a reference here. Note that RDF prefixes can also be used in SPARQL queries as a shorthand to write IRIs.

SPARQL generally works by finding *graph patterns* or *triple patterns* in RDF graphs, which are triples where variables can occur in addition to standard RDF constants (which may be filtered), using the `WHERE` clause of a query, and then performing operations on such patterns. In the case of the SPARQL query language, these operations can be to `SELECT` variables (extract specific values from triples), `CONSTRUCT` graphs based on found patterns, `ASK` if a given pattern exists (boolean queries), and `DESCRIBE` a given graph. Example 3.2 is a `SELECT` query which could be run on the graph from Figure 3.1 and Table 3.1, yielding the results presented in Table 3.2. Query results are either RDF graphs (in the case of `CONSTRUCT` and `DESCRIBE` queries) or a set of bindings found in the graph for each variable (for typical `SELECT` queries).

Example 3.2 SPARQL query selecting the travel dates of user #9590.

```
SELECT ?trv ?date
WHERE {
    ?trv    rdf:type    tcl:Validation.
    ?trv    tcl:user    tcl:u9590.
    ?trv    tcl:validationDatetime  ?date.
}
```

Table 3.2: Results obtained when evaluating the query from Example 3.2 on the graph from Figure 3.1 and Table 3.1.

trv	*date*
tcl:v26830	"2017-04-27T19:46:51"

The SPARQL Update language also works using graph and triple patterns. SPARQL Update queries performing actions on the graph are usually called *operations*. These operations serve two main purposes, quite similar to those from usual query languages such as SQL:

- Data management (CRUD operations over triples from one or more graphs): `DELETE` triples and optionally `INSERT` new ones, `LOAD` a graph into another one, or `CLEAR` all triples;

[6]The SPARQL Update language is also sometimes nicknamed "SPARUL" to follow the name on its original W3C Submission.

- Graph management (CRUD operations over graphs themselves): `CREATE`, `DROP` (delete), `MOVE` (rename) and `COPY` a graph, as well as `ADD` the content of a graph into another.

Example 3.4 is a `DELETE/INSERT` query which could once again be used on the graph from Figure 3.1 and Table 3.1. The results of this evaluation is the updated graph presented in Table 3.3 (prefixes are used for clarity and ease of reading).

Example 3.4 SPARQL Update query editing user #9590's birth date.

```
1 DELETE { tcl:u9590   tcl:birthDate  ?date . }
2 INSERT { tcl:u9590   tcl:birthDate  "1994-05-13"^^xsd:date . }
3 WHERE  { tcl:u9590   tcl:birthDate  ?date . }
```

Table 3.3: RDF graph from Example 3.1 edited by the update query from Example 3.4.

Subject	Predicate	Object
tcl:v26830	rdf:type	tcl:Validation
tcl:v26830	tcl:validationDatetime	"2017-04-27T19:46:51"
tcl:v26830	geo:latitude	"45.7599724233"
tcl:v26830	geo:longitude	"4.82594480511"
tcl:v26830	tcl:user	tcl:u9590
tcl:u9590	rdf:type	tcl:User
tcl:u9590	foaf:givenName	"Germon"
tcl:u9590	foaf:familyName	"Anthony"
tcl:u9590	vcard:hasAddress	"180 Rue A 69XXX Curis-au-Mont-d'Or"
tcl:u9590	tcl:birthday	"1994-05-13"

Note that SPARQL Update queries do not return results, according to the standards; they only perform the structural modifications described in their header and body, possibly returning failure or success flags depending on the operation. Nevertheless, some triple stores and query engines implement specific return values for SPARQL Update queries, such as the number of triples altered by the query.

Both types of SPARQL queries can be used in triple stores and query engines locally, but also by the general public through the use of **SPARQL endpoints**, which are public or private interfaces allowing anyone to query RDF triple stores using legal SPARQL queries (or other syntaxes if they are supported by the query engine). These endpoints can be used by software usually as a REST API, or manually by an user through web interfaces.

3.1.3 Ontologies using RDFS and OWL

The classical and generic definition of **ontologies** was written by Tom Gruber in 1993, defining them as *"explicit specifications of a conceptualization"* [Gruber93]. More concretely, in the case of the Semantic Web, ontologies are complex vocabularies encompassing specific terms and relationships on a given theme or subject, to be then reused on any RDF graph in need of such terminology. They help the design of an RDF graph by offering a variety of pre-existing concepts and values, helping the uniformity of RDF graphs in general by making it easier to reuse an unique vocabulary to describe the same notions. Ontologies are therefore one of the building blocks of Linked Data itself. The W3C designed standards to help conceiving ontologies, rather than directly providing ontologies themselves. The two main ones are **RDF Schema (RDFS)** and the **Web Ontology Language (OWL)**. In a way, both are *vocabularies to help building vocabularies*.

RDFS [Brickley14] is an extension of the original, native RDF vocabulary allowing the description of simple meta-data over an RDF graph, such as classes and properties along with various concepts related to them (sub-classes, sub-properties). Based on this, it is also possible to define the *range* and the *domain* of a property, using the previously defined classes. All these terms are IRIs prefixed with the string `rdfs:`, as a replacement for `http://www.w3.org/2000/01/rdf-schema#`.

OWL2 [McGuinness12] is the most recent version of OWL and is once again a language designed to write ontologies. OWL can be seen a more complex and more advanced alternative to RDFS: its vocabulary is much bigger, and includes most of RDFS in itself. Also, while RDFS only acts as additional metadata, OWL encompasses more logical formalism as it also prohibits some relationships (e.g. a class instance cannot be a class itself). OWL2 includes three sub-languages (or "profiles"), since reasoning on the entirety of the OWL logic might be too expensive in terms of computation complexity: **EL** (allowing a lot of expressivity and decent reasoning time with many classes and properties), **RL** (focusing on reasoning time for complex systems and forward chaining, at the expense of expressivity), and **QL** (allowing decent query answering time, for databases using a lot of class instances) [Wu12]. OWL can also express restrictions on data, such as a minimal/maximal cardinality for a given property, or a list of items in which a class instance must have its value.

Many ontologies exist online, published by experts and usually specialized in a given subfield, and they are mostly based on RDFS and OWL. It is also possible to find online repositories to help one finding a satisfying ontology or vocabulary namespace, such as the specialized BioPortal website from the National Center for Biomedical Ontology [Jonquet11]. Note that the same vocabulary can be imported in different ways when modeling an RDF graph, since this is all a matter of syntax and there is no compilation or validation of the IRIs written in a graph, which may cause interoperability issues in itself [Atemezing13].

The usual theoretical formalization for RDF graphs and SPARQL queries will be detailed in the following section, inspired by the one detailed in [Gutiérrez04].

3.2 Logical formalization

3.2.1 RDF and SPARQL basics

Let $\mathbf{I}$, $\mathbf{L}$ and $\mathbf{B}$ be countably infinite, pairwise disjoint sets representing respectively *IRIs*, *literals* and *blank nodes*.

We denote by $\mathbf{T} = \mathbf{I} \cup \mathbf{L} \cup \mathbf{B}$ the set of *terms*, in which we distinguish *constants* (IRIs and literals) from *blank nodes*, which are used to model unknown IRIs or literals like in [Goasdoué13, Buron19] and correspond to *labeled nulls* in traditional database formalizations [Abiteboul95].

We also assume an infinite set $\mathbf{V}$ of variables disjoint from the previously defined sets. Throughout the following chapters, we adhere to the SPARQL conventions: variables in $\mathbf{V}$ are prefixed with a question mark (`?`), IRIs in $\mathbf{I}$ are prefixed with a colon (`:`), and blank nodes in $\mathbf{B}$ are prefixed with an underscore and a colon (`_:`).

Definition 1 (RDF graph)**.** *An* RDF graph *is a finite set of* RDF triples (s, p, o)*, where* $(s, p, o) \in (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times \mathbf{T}$.

IRIs appearing in second position in triples (as p) denote predicates, and provides the *schema*, i.e. the graph structure of the data.

SPARQL queries are structured in two main parts: a *head* and a *body*. The body of a query describes the patterns to be found in the target graph with possible filtering conditions or advanced processing (union of patterns, optional patterns, grouping...), while the head designates the variables whose value must be fetched, with or without additional computations. For example, the query can return the requested values themselves, or compute aggregate functions such as a sum or a counting function.

The body of a query is an *RDF graph pattern*, which is a set of triples that may contain variables and that is defined as follows:

Definition 2 (Graph pattern)**.** *A* triple pattern *is a triple* $(s, p, o) \in (\mathbf{I} \cup \mathbf{B} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{T} \cup \mathbf{V})$*. A* graph pattern *is a finite set of triple patterns.*

We now define logical entailment between graphs or between graph patterns. First, we define homomorphisms between these objects.

Definition 3 (Graph and graph pattern homomorphisms)**.** *Let H and H' two RDF graphs or two graph patterns. An homomorphism from H' to H is an application $h : (\mathbf{T} \cup \mathbf{V}) \rightarrow$*

$(\mathbf{T} \cup \mathbf{V})$ *such that* $h(H') \subseteq H$ *and where:*

$$\begin{cases} h((s,p,o)) = (h(s), h(p), h(o)) \\ h(c) = c, \text{ for } c \in \mathbf{L} \cup \mathbf{I} \end{cases}$$

We define ***mappings*** *as a particular case of homomorphisms where* H *is an RDF graph.*

We define $\phi(H)$ as the logical modeling of a graph (resp. graph pattern) H, that is the conjunction of its triples where each is interpreted as an atomic formula and where each blank node (resp. each variable) is interpreted as an existentially quantified variable whose scope is H.

Example 3.5 Graph and graph pattern entailment

Let the following graphs:

```
G1 = { ex:a ex:p ex:b.  ex:b ex:q ex:c. }
G2 = { ex:a ex:p _:b.   _:b  ex:q ex:c. }
G3 = { _:b  ex:p  ex:o. ex:o ex:p _:b. }
```

We have the following modelizations, where T is the logical evaluation of an RDF triple:

$$\phi(G_1) = T(\texttt{ex:a}, \texttt{ex:p}, \texttt{ex:b}) \wedge T(\texttt{ex:b}, \texttt{ex:q}, \texttt{ex:c})$$

$$\phi(G_2) = \exists ?b \mid T(\texttt{ex:a}, \texttt{ex:p}, ?b) \wedge T(?b, \texttt{ex:q}, \texttt{ex:c})$$

$$\phi(G_3) = \exists ?b \mid T(?b, \texttt{ex:p}, \texttt{ex:o}) \wedge T(\texttt{ex:o}, \texttt{ex:p}, ?b)$$

Let the following graph pattern:

```
GP = { ?x ex:p ?y.  ?y ex:q ?z. }
```

We have:

$$\phi(GP) = \exists ?x, ?y \mid T(?x, \texttt{ex:p}, ?y) \wedge T(?y, \texttt{ex:q}, ?z)$$

Definition 4 (RDF graph entailment (from [de Bruijn10])). *For any two graphs or graph patterns* H *and* H'*,* H *entails* H' *if and only if every model of* $\phi(H)$ *is also a model of* $\phi(H')$*. We write* $H \models H'$*.*

We now link graph (and graph pattern) entailment to homomorphisms in Theorem 1.

Theorem 1. *Let* H *and* H' *two RDF graphs or two graph patterns.* $H \models H'$ *if and only if there exists an homomorphism from* H' *to* H*.*

Proof sketch. Using Definition 4, we can write that $H \models H' \Leftrightarrow \mathsf{Ans}(\mathsf{bool}(H), G) \subseteq \mathsf{Ans}(\mathsf{bool}(H'), G)$ for any graph G, where $\mathsf{bool}(Q)$ is the boolean query built from a conjunctive query Q: we have $Q = \{\emptyset, \mathsf{body}(Q)\}$.

The equivalence we are looking for amounts in fact to the *homomorphism theorem* from [Chandra77] and [Abiteboul95] (Theorem 6.2.3), both long-established statements.

It is a variant of the *interpolation lemma* from [Hayes14, Gutiérrez11] for the first-order logic semantics of Definition 4. □

We also define the notion of *unification between two graph patterns*. This definition works when the considered graph patterns do not possess variables with the same name. Without loss of generality, variables can be renamed to make this definition applicable.

Definition 5 (Unifiable graph patterns)**.** *Let GP_1 and GP_2 two 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 $s(GP_1) = s(GP_2)$. A most general unifier replaces variables by constants appearing in GP_1 or GP_2 only if no unifier exists that replaces the corresponding variables with variables.*

Example 3.6

Let the following graph patterns GP_1 and GP_2. Note than in this case, neither of those graph patterns entails the other.

```
1 GP1 = { ?x1 p ?x1 .  ?x1 q ?y1 . }
2 GP2 = { ?x2 p ?y2 .  ?z2 q ?z2 . }
```

GP_1 and GP_2 are unifiable, using this unifier s:

$$s(\texttt{?x1}) = \texttt{?x1} \quad s(\texttt{?y1}) = \texttt{?x1}$$
$$s(\texttt{?x2}) = \texttt{?x1} \quad s(\texttt{?y2}) = \texttt{?x1}$$
$$s(\texttt{?z2}) = \texttt{?x1}$$

We also define the *canonical RDF graph* of a graph pattern which is in fact the most general RDF graph satisfying this graph pattern as stated in Proposition 1.

Definition 6 (Canonical RDF graph)**.** *The canonical (RDF) graph of a graph pattern GP is the RDF graph G_C obtained by replacing each variable from GP by a distinct fresh constant, without using blank nodes.*

Proposition 1. *Let GP a graph pattern, and G_C its canonical RDF graph. The following statements hold:*

- $G_C \models GP$;
- *For any graph or graph pattern H, if $G_C \models H$ then $GP \models H$.*

Proof sketch. We prove both affirmations separately.

- $G_C \models GP$

By definition of G_C, we have a bijection between the variables from GP and the new constants from G_C. This implies there is a μ such that $\mu(GP) = G_C$. By Theorem 1 1, this implies that $G_C \models GP$.

- For any graph or graph pattern H, if $G_C \models H$ then $GP \models H$

We know that $G_C \models GP$, therefore there is a μ such that $\mu(GP) \subseteq G_C$. Besides, since $G_C \models H$, there is a μ' such that $\mu'(H) \subseteq G_C$.

Let $v \in \mathbf{V}$ a variable in H. We have $\mu'(v) = c$ with c a constant, but by definition of G_C, there is a variable $v' \in \mathbf{V}$ in GP such that $\mu(v') = c$.

We can progressively build the mapping μ'' such that $\mu''(v) = v'$. μ'' is a function, since for each v

By construction, we have $\mu''(H) \subseteq Gp$, and by Theorem 1, $GP \models H$.

□

3.2.2 Types of queries

We can now define the three types of queries that we consider in our approach, along with the answers to these queries.

The first type of queries introduced in Definition 7 is a formalization of *conjunctive queries*. It is in fact a slight restriction of the standard definition of graph pattern queries, and it will be the basis for formalizing the sensitive information that must not be disclosed. The second type, defined in Definition 11, corresponds to *counting queries*. This type of queries will be used in our approach to model a form of utility that may be useful to preserve for analytical tasks. Both of these types of queries are based on the semantics of SPARQL `SELECT` queries.

Finally, Definition 12 defines the third type of queries: *update queries*, which model the anonymization operations handled in our framework, based on the SPARQL Update semantics. They are logical modelings of SPARQL Update queries to be used as anonymization operators.

Definition 7 ((Restricted) Conjunctive query)**.** *A **conjunctive query** Q is defined by an expression* `SELECT` $\bar{x}$ `WHERE` $GP(\bar{x}, \bar{y})$ *where $GP(\bar{x}, \bar{y})$ (also denoted* $\mathsf{body}(Q)$*) is a graph pattern without blank nodes, and $\bar{x} \cup \bar{y}$ is the set of its variables among which $\bar{x}$ are the **result variables** (also called the* distinguished variables*), and where the subset of variables in predicate position is disjoint from the subset of variables in subject or object position. A conjunctive query Q can be alternatively written as a pair $Q = \langle \bar{x}, GP \rangle$.*

*A **boolean query** is a conjunctive query of the form $Q = \langle \emptyset, GP \rangle$.*

Note that considering graph patterns without blank nodes to define conjunctive queries is not a restriction, since the normative SPARQL query evaluation of a query treats its blank nodes as variables.

It should also be noted that conjunctive queries with variables in predicate position are allowed, if such variables *do not appear in a subject or object position*, as it is the case in Example 3.7. Example 3.8 shows an example of a query that does not conform to Definition 7. This restriction is necessary in order to ensure that within a conjunctive query, all occurrences of a given variable are in the same connected component (see Definition 8).

Example 3.7

The following statement is a conjunctive query, in conformity with Definition 7.

```
SELECT ?s ?v
WHERE {
    ?s ?p ?o.
    ?o ?p ?v.
    ?o rdf:type tcl:User.
}
```

Example 3.8

Consider the following query:

```
SELECT ?s ?v
WHERE {
    ?s ?p ?v.
    ?p rdf:type owl:ObjectProperty.
}
```

This does not conform to Definition 7, because the variable `?p` appears in a predicate position in the first triple pattern, and in a subject position in the second triple pattern. According to Definition 8, this query has two connected components, each of them containing one occurrence of `?p`.

Definition 8 (Connected components of a query)**.** *Given a conjunctive query $Q = \langle \bar{x}, GP \rangle$, let $G_Q = \langle N_Q, E_Q \rangle$ be the undirected graph defined as follows: its nodes N_Q are the distinct variables and constants appearing in subject or object position in GP, and its edges E_Q are the pairs of nodes (n_i, n_j) such that there exists a triple (n_i, p, n_j) or (n_j, p, n_i) in GP.*

Each subgraph SG_Q of G_Q corresponds to the subgraph of $\mathsf{body}(Q)$ *made of the set of triples (s, p, o) such that either (s, o) or (o, s) is an edge of SG_Q. These subgraphs are the* ***connected components*** *of G_Q. By slight abuse of notation, we will call the* ***connected components of the query*** *Q the (disjoint) subsets of $GP = \mathsf{body}(Q)$ corresponding to the connected components of G_Q.*

A connected component GP_C of the query Q is called boolean *when it contains no result variable.*

Example 3.10 Connected components in a query.

Let Q be the following SPARQL query (remind that in SPARQL syntax, `a` is a shorthand for `rdf:type`):

```
SELECT ?u ?n
WHERE {
    ?u  a   tcl:User .
    ?u  foaf:givenName  ?n.
    ?v  a   tcl:Validation.
    ?v  tcl:validationDatetime  ?d.
}
```

Q has two connected components, namely GP_1 and GP_2:

```
GP1 = {
    ?u  a   tcl:User .
    ?u  foaf:givenName  ?n.
}

GP2 = {
    ?v  a   tcl:Validation.
    ?v  tcl:validationDatetime  ?d.
}
```

GP_2 is a boolean connected component as it contains neither the variable `?u` nor the variable `?n` that are the only result variables of the query Q. It expresses a boolean condition for satisfying the query: there must exist (in the graph) a validation for which there is a validation date and time provided. If the boolean condition holds, GP_1 expresses the constraints on the pairs $(user, name)$ of constants respectively instantiating the result variables `?u` and `?n` to be returned as answers to the query: *user* must be an user with *name* as their given name.

We now define the notion of *critical terms* in a query, in Definition 9. They will play an important role in the algorithms of both contributions presented in Chapters 4 and 5. Example 3.12 uses the query from Example 3.10 to illustrate this notion.

Definition 9 (Critical terms)**.** *A variable (resp. constant) in subject or object position having several occurrences within the body of a query is called a* join variable *(resp.* join constant*). We name the union of the join variables, join constants and result variables of a query as its* ***critical terms****.*

We now define how a query is evaluated, and the answers this evaluation might return. The evaluation of a query $Q = \langle \bar{x}, GP \rangle$ over an RDF graph G consists in finding mappings μ assigning the variables in GP to terms such that the set of triples, denoted

Example 3.12 Critical terms in a query.

We use the same query as in Example 3.10.

```
SELECT ?u ?n
WHERE {
    ?u  a   tcl:User .
    ?u  foaf:givenName  ?n.
    ?v  a   tcl:Validation.
    ?v  tcl:validationDatetime  ?d.
}
```

This query has three critical terms, namely u, v (since both have multiple occurrences in the body of the query) and n (since it is a result variable of the query).

$\mu(GP)$ and obtained by replacing with $\mu(z)$ each variable z appearing in GP, is included in G. The corresponding answer is defined as the tuple of terms $\mu(\bar{x})$ assigned by μ to the result variables. We write $G[b_0 \leftarrow b'_0, \ldots, b_k \leftarrow b'_k]$ for the graph in which each occurrence of b_i is replaced by b'_i for every $i \in [1..k]$.

Definition 10 (Evaluation of a conjunctive query ([Pérez09])). *Let $Q = \langle \bar{x}, GP \rangle$ be a conjunctive query and let G be an RDF graph. The* answer set of Q over G *is defined by :* $\mathsf{Ans}(Q, G) = \{\mu(\bar{x}) \mid \mu(GP) \subseteq G\}$.

According to Definition 10, a boolean query Q has only two possible answers, indeed for all G, $\mathsf{Ans}(Q, G)$ is either $\emptyset$ or $\{\emptyset\}$. Definition 10 is then extended to counting queries as well, in Definition 11.

Definition 11 (Counting query). *Let Q be a conjunctive query. The query* $\mathsf{Count}(Q)$ *is a* counting query, *whose answer over a graph G is defined by:* $\mathsf{Ans}(\mathsf{Count}(Q), G) = |\mathsf{Ans}(Q, G)|$

We now define an additional ingredient: *update queries*, based on the eponymous SPARQL mechanism. As opposed to the previous conjunctive queries, update queries are not designed to return the contents of a graph, but rather modify its contents, by deleting or replacing parts of the graph matching a given pattern. Intuitively, an update query `DELETE` $D(\bar{x})$ `INSERT` $I(\bar{y})$ `WHERE` $W(\bar{z})$ `isNotBlank`$(\bar{b})$ executed on a graph G searches for the instances of the graph pattern $W(\bar{z})$ in G, then deletes the instances of $D(\bar{x})$ and finally inserts the $I(\bar{y})$ part.

The `isNotBlank`$(\bar{b})$ operator, which translates to `FILTER(!isBlank(b1) && ... && !isBlank(bn))` in SPARQL (where $b_1, ..., b_n \in \bar{b}$), rules out instances where a variable in $\bar{b}$ is already mapped to a blank node. It will be used in Algorithm 3 to avoid replacing the images of critical terms that are already blank nodes, because treating these cases would be unnecessary.

Note that it is indeed allowed to insert blank nodes in $I(\bar{y})$. For example, `_:b1` is a blank node designating the same resource in each mention of `_:b1` in the same

graph pattern, but reusing `_:b1` in another query or graph pattern would indicate a new blank node. These semantics of blank nodes model the SPARQL specification.[7]

Definition 12 (Update query)**.** *An* update query *(or* update operation*)* Q_u *is defined by* `DELETE` $D(\bar{x})$ `INSERT` $I(\bar{y})$ `WHERE` $W(\bar{z})$ `isNotBlank`$(\bar{b})$ *where* D *(resp.* W*) is a graph pattern whose set of variables is* $\bar{x}$ *(resp.* $\bar{z}$*) such that* $\bar{x} \subseteq \bar{z}$*; and* I *is a graph pattern where blank nodes are allowed, whose set of variables is* $\bar{y}$ *such that* $\bar{y} \subseteq \bar{z}$*.* `isNotBlank`$(\bar{b})$ *is a parameter where* $\bar{b}$ *is a set of variables such that* $\bar{b} \subseteq \bar{z}$*. The evaluation of* Q_u *over an RDF graph* G *is defined by:*

$$\begin{aligned}\mathsf{Result}(Q_u, G) = G \setminus \{\mu(D(\bar{x})) | \mu(W(\bar{z})) \subseteq G \wedge \forall x \in \bar{b}, \mu(x) \notin \mathbf{B}\} \\ \cup \{\mu'(I(\bar{y})) | \mu(W(\bar{z})) \subseteq G \wedge \forall x \in \bar{b}, \mu(x) \notin \mathbf{B}\}\end{aligned}$$

where μ' *is an extension of* μ *renaming blank nodes from* $I(\bar{y})$ *to fresh blank nodes, i.e. a mapping such that* $\mu'(x) = \mu(x)$ *when* $x \in \bar{z}$ *and* $\mu'(x) = b_{new} \in \mathbf{B}$ *otherwise.*

When $\bar{b}$ *is empty, we avoid writing* `isNotBlank`$(\emptyset)$*. Similarly, when* `INSERT` $I(\bar{y})$ *is empty, the query is written* `DELETE` $D(\bar{x})$ `WHERE` $W(\bar{z})$ *and is called a* ***deletion query****.*

The application of an update query Q_u *on a graph* G *is alternatively written* $Q_u(G) = \mathsf{Result}(Q_u, G)$*. This notation is naturally extended to a sequence of operations* $O = \langle Q_u^1, \ldots Q_u^n \rangle$ *by* $O(G) = Q_u^n(\ldots(Q_u^1(G))\ldots)$*.*

For the sake of conciseness, we write $\mathsf{Update}(H, I, W)$ for the function to the update query `DELETE` H `INSERT` I `WHERE` W and $\mathsf{Delete}(H, W)$ for the function of the deletion query `DELETE` H `WHERE` W, that is:

$$\mathsf{Update}(H, I, W) = \lambda G.\, \mathsf{Result}(\texttt{DELETE}\ H\ \texttt{INSERT}\ I\ \texttt{WHERE}\ W, G)$$

$$\mathsf{Delete}(H, W) = \lambda G.\, \mathsf{Result}(\texttt{DELETE}\ H\ \texttt{WHERE}\ W, G)$$

Example 3.13

The following query taken from Example 3.4 conforms to Definition 12 and is a valid update query.

```
1 DELETE { tcl:u9590   tcl:birthDate  ?date . }
2 INSERT { tcl:u9590   tcl:birthDate  "1994-05-13"^^xsd:date . }
3 WHERE  { tcl:u9590   tcl:birthDate  ?date . }
```

As we will need to compare query and query results, we will also use the notion of *query containment*. The classical definition of query containment is recalled and adapted to our notations in Definition 13.

[7] See [Harris13], Section "SPARQL Grammar, Blank Nodes and Blank Node Labels": https://www.w3.org/TR/sparql11-query/#grammarBNodes

Example 3.14

The following query conforms to Definition 12 as well, as blank nodes are allowed in the insertion graph pattern of the query.

```
1 DELETE { tcl:u9590   tcl:birthDate  ?date . }
2 INSERT { tcl:u9590   tcl:birthDate  _:blank . }
3 WHERE  { tcl:u9590   tcl:birthDate  ?date . }
```

Definition 13 (Query containment (from [Chirkova09])). *Let Q_1 and Q_2 two conjunctive queries. Q_1* ***is contained in*** *Q_2 if and only if $\mathsf{Ans}(Q_1, G) \subseteq \mathsf{Ans}(Q_2, G)$ for any graph G.*

Query containment as a sub-field has been extensively studied in database theory [Doan12], and many algorithms have been designed to check it along with associated complexity results. Checking query containment is NP-complete for conjunctive queries [Millstein03]. The classic algorithm to check if a query Q_1 is contained in a query Q_2 works by evaluating Q_2 on the canonical graph of Q_1's body. If there is an answer, Q_1 is contained in Q_2; otherwise it is not.

Humankind cannot gain anything without first giving something in return. To obtain, something of equal value must be lost.

Hiromu Arakawa – *Fullmetal Alchemist*, vol. 1 (2001)

4

Query-based privacy and utility-preserving anonymization

▷ *Previous chapters detailed the context of this work. We now present the first main contribution, from its motivation and the statement of the considered problem to a concrete implementation and its characteristics. This chapter introduces our static, data-independent framework and deals with the local anonymization of RDF graphs, provided some privacy and utility policies expressed as queries. We design an anonymization algorithm for this framework and prove its soundness.* ◁

Chapter summary

AFTER presenting the main motivations of this project, the technical and formal context it is part of, and the state of the art regarding related and previous work, all the ingredients are available to develop a full-fledged solution to the problems mentioned in Chapters 1 and 2. Notably, the lack of simple *data-independent, native and declarative RDF anonymization solution* handling the duality between privacy and utility restrictions. The main aspects of this contribution are also developed in [Delanaux18].

Note that in this framework, we assume that **the RDF graphs (their schema and their data parts) are not evolving during the anonymization process**. This is a reasonable assumption, since anonymization is usually performed at a fixed point in time before publishing the graph rather than in a real-time fashion which would be closer to access control approaches.

4.1 Motivation and approach

While providing a generic RDF anonymization framework is itself a good enough motivation, we need to explain the main principles that have guided our approach and to position them with respect to Section 2.2.

- **Privacy should be balanced with utility.**

Anonymization is all about finding the right, or the best balance between data privacy and data utility. An extreme, simple solution would be to delete the whole data: while this preserves privacy, this removes the entire utility and relevance of the data. We therefore want to be able to build anonymized RDF graphs satisfying both privacy and utility guarantees. This required an appropriate formalization of privacy and utility policies to be able to customize the desired balance between them.

- **Privacy/utility policies should be:**
  - **Expressed by data providers**, or at least be written in an accessible and understandable way to be handled by them;
  - **Adequately formally defined** to enable their processing to check some properties (such as their compatibility).

This leverages two challenges. First, find a way to encore policies in a simple (preferably native) format while keeping them easily machine-readable and useful. And secondly, define a way to check the compatibility of these privacy and utility constraints, since they may be contradictory or erroneous: humans make mistakes, and they can program software which can in turn make mistakes. This means that we have to validate or provide theoretical criteria for a set of privacy and utility policies to be not incompatible.

- **Anonymization operations should also be readable and understandable by data providers to allow a selection among multiple candidate anonymizations**.

This represents the core of our framework: define an automated anonymization algorithm, agnostic to the graphs contents, using declarative operators to satisfy both privacy and utility demands, and doing so *every time it is possible to do it*. Since there may be multiple ways to satisfy both policies, these operators must not be opaque and should be both human-readable and machine-readable to allow for a relevant decision.

Following this, we can express the goal of this thesis as such: we want to design **full-fledged, static algorithms anonymizing RDF graphs** prior to their publication to the LOD cloud, using **declarative constraints on data and anonymization operators** and **theoretical privacy guarantees** following a **fixed privacy/utility trade-off** applicable in the context of the Semantic Web.

Sketch of our approach

Based on these requirements, we detail the following contributions in the following sections:

- We formalize the problem of finding sequences of anonymization operations that satisfy policy and utility constraints provided as sets of conjunctive queries named *policies* (Section 4.2), with operators generated as SPARQL update queries;
- We study the requirements for a privacy policy and a utility policy to be *compatible* between one another (Section 4.3);
- We provide a static analysis method that does not need to inspect the dataset to be anonymized in order to solve the aforementioned problem by building candidate sequences of basic anonymization operations reduced to atomic delete or update queries. As such, our method is independent of the size of the datasets and depends solely on the size of the queries. We also prove the soundness of our method (Section 4.4);
- Finally, we discuss how our framework places itself in other techniques and contexts, such as reasoning with ontologies (Section 4.5);

To the best of our knowledge and accordingly to the survey from Chapter 2, such a framework is the first to provide practical algorithms for building candidate sequences of atomic operations, which has still been described as an open research challenge in the latest years [Grau19]. We study the characteristics of the candidate sequences outputted by our algorithm in an experimental analysis in Section 4.6.

In the end, such a framework is tailored for data publishing in the LOD since, as expressed before, it is important to strike a balance between non-disclosure of information (in particular, values likely to serve as quasi-identifiers when interconnected

with external LOD datasets) and utility preservation for end-users querying the graph or the LOD cloud in general. Our framework is carefully designed to meet all these requirements by leveraging at the same time the expressive power of SPARQL queries and the maturity and effectiveness of SPARQL query engines. It builds on the common wisdom that queries from data providers are more and more available through online SPARQL endpoints [Bonifati17]. Such queries are a valuable resource to understand the real utility needs of users on publicly available data and to guide the data providers in safe RDF data publishing.

For all these reasons, this approach paves the way to a democratization of privacy-preserving mechanisms for LOD data and in the Semantic Web community in general.

4.2 A query-based framework

We develop the two main logical objects used in our framework: (i) the declarative modling of privacy and utility by unions of conjunctive queries called **policies**, and (ii) modeling **anonymzation operations** as update queries to be performed on the graph to anonymize it.

4.2.1 Modeling policies through queries

The notion of **policy** encapsulates data privacy or utility demands regarding the anonymized graph. In itself, a policy is simply a *set of queries*, as expressed in Definition 14.

Definition 14 (Policy)**.** *A **policy** is a set of conjunctive queries. A policy consisting of a set of only one query is named an **unitary policy**.*

The role of a policy in the framework, i.e. if it is a *privacy policy* (to indicate data that should be hidden in the anonymized graph) or a *utility policy* (data that should be preserved and disclosed), is the determining factor of its semantics and formal meaning, as we will see now. Following [Grau19], a privacy policy satisfies the anonymization process if none of the sensitive answers holds in the resulting dataset. This is achieved by letting the privacy queries return no answer or, alternatively, answers with blank nodes, as shown in the remainder. We also model utility policies by sets of queries, that can be either conjunctive queries or counting queries useful for data analytics. On the other hand, to satisfy a utility policy the anonymization process must preserve the answers of all the specified utility queries. We now formally define both types of policies.

Definition 15 (Privacy and utility policies: query-based semantics)**.** *Let $\mathcal{P}$ (resp. $\mathcal{U}$) be a **privacy (resp. utility) policy** expressed as a set of conjunctive queries (resp. conjunctive or counting queries).*

Let $\mathsf{Anonym}(G)$ *be the result of an anonymization process of an RDF graph G by a sequence of anonymization operators.*

The privacy policy $\mathcal{P}$ *is* satisfied *on* $\mathsf{Anonym}(G)$ *if for every* $P \in \mathcal{P}$ *and for any tuple of constants* $\bar{c}$*, it holds that:* $\bar{c} \notin \mathsf{Ans}(P, \mathsf{Anonym}(G))$.

The utility policy $\mathcal{U}$ *is* satisfied *on* $\mathsf{Anonym}(G)$ *if for every* $U \in \mathcal{U}$ *it holds that for any tuple* $\bar{c}$ *of constants,* $\bar{c} \in \mathsf{Ans}(U, G) \Leftrightarrow \bar{c} \in \mathsf{Ans}(U, \mathsf{Anonym}(G))$.

We use the conventional notation $|\mathcal{P}|$ (resp. $|\mathcal{U}|$) to denote the *cardinality* i.e. the number of queries in each type of policy.

For a policy $\mathcal{P}$ (resp. $\mathcal{U}$) consisting of n queries $P_i = \langle \bar{x}_i^P, G_i^P \rangle$ (resp. m queries $U_i = \langle \bar{x}_i^U, G_i^U \rangle$), we call the sum of the cardinalities of their bodies (i.e. the number of triples in each query body) the **size of the policy**, defined by $\sum_{i=1}^{n} |G_i^P|$ (resp. $\sum_{i=1}^{m} |G_i^U|$).

Example 4.1, designed following our running example, shows that privacy and utility policies might impose constraints on overlapping portions of a dataset. Note that in this example, the size of the privacy policy is 6 (2+4 triples) and the size of the utility policy is 5 (2+3 triples).

4.2.2 Modeling anonymization operations as update queries

Regarding anonymization operations, we extend the notion of "suppressor functions" considered in [Grau19] that replace IRIs with blank nodes by allowing also triple deletions. The anonymization operations that we consider correspond to the previously defined **update queries** (Definition 12).

In the following section, we will focus on two kinds of atomic anonymization operations that correspond respectively to **triple deletions** (i.e. the particular case of Definition 12 where $D(\bar{x})$ is reduced to a triple pattern and $I(\bar{y})$ is empty) and **replacement of IRIs by blank nodes** (i.e. the particular case of Definition 12 where $D(\bar{x})$ and $I(\bar{y})$ are triple patterns that differ only by the fact that one bound variable of $D(\bar{x})$ is replaced with an existential variable in $I(\bar{y})$).

These two atomic anonymization operations are illustrated in Examples 4.2 and 4.3 respectively. From now on, by slight abuse of notation with regard to Definition 12 we will use the SPARQL standard notation for blank nodes `[ ]` to denote single existential variables.

Example 4.1

Consider a privacy policy $\mathcal{P} = \{P_1, P_2\}$ written over our running data example, defined by the two following conjunctive queries written in concrete SPARQL syntax. The first privacy query expresses that travelers' postal addresses are sensitive and shall be protected, and the second privacy query specifies that the disclosure of users identifiers associated with geolocation information (like latitude and longitude as given by the user ticket validation) may also pose a risk (for re-identification by data linkage with other LOD datasets).

```
# Privacy query P1
SELECT ?ad
WHERE {
  ?u  a     tcl:User.
  ?u  vcard:hasAddress  ?ad.
}
```

```
# Privacy query P2
SELECT ?u ?lat ?long
WHERE {
  ?c  a     tcl:Validation.
  ?c  tcl:user       ?u.
  ?c  geo:latitude   ?lat.
  ?c  geo:longitude  ?long.
}
```

As a consequence, any query displaying either users' addresses or users' identifiers together with their geolocation information would infringe this privacy policy, violating the anonymization of the underlying dataset to be published as open data. The counterpart utility policy is the set of queries $\mathcal{U} = \{U_1, U_2\}$. This set states that users' ages and location related to journeys are to be preserved.

```
# Utility query U1
SELECT ?u ?age
WHERE {
  ?u  a  tcl:User.
  ?u  foaf:age  ?age.
}
```

```
# Utility query U2
SELECT ?c ?lat ?long
WHERE {
  ?c  a     tcl:Validation.
  ?c  geo:latitude   ?lat.
  ?c  geo:longitude  ?long.
}
```

Example 4.2

In the same setting as Example 4.1, the following query specifies an operation deleting users' addresses.

```
DELETE  {
    ?u  vcard:hasAddress   ?ad.
}
WHERE   {
    ?u  a   tcl:User.
    ?u  vcard:hasAddress   ?ad.
}
```

Example 4.3

In the same context, this query replaces users' identifiers related to a ticket validation by a blank node.

```
DELETE  {
    ?c  tcl:user   ?u.
}
INSERT  {
    ?c  tcl:user   [].
}
WHERE   {
    ?c  a   tcl:Validation.
    ?c  tcl:user       ?u.
    ?c  geo:latitude   ?lat.
    ?c  geo:longitude  ?long.
}
```

4.3 Compatibility between privacy and utility policies

Given a privacy policy and a utility policy, the first concern we want to address is to make sure that these policies are *compatible* before attempting to generate anonymization operations. We address this as the COMPATIBILITY problem, which is a decision problem.

Problem 1. *The* COMPATIBILITY *problem.*

Input : $\mathcal{P} = \{P_i\}$ *a privacy policy and* $\mathcal{U} = \{U_j\}$ *a utility policy*

Output: `True` *if for any graph* G, there exists *a sequence of operations* O *such that* $O(G)$ *satisfies both* $\mathcal{P}$ *and* $\mathcal{U}$ *and* `False` *otherwise.*

We first provide a decision procedure which solves the compatibility problem when the anonymization operations are reduced to triple deletions or replacement by blank nodes. Theorem 2 shows that for overlapping privacy and utility policies, i.e., such that their bodies have *unifiable subgraphs* (see Definition 5), checking their compatibility can be reduced to the evaluation of their satisfiability over (the finite number of) all the anonymization operations of the corresponding *canonical RDF graphs* (see Definition 1).

Theorem 2. *Let a privacy policy* $\mathcal{P}$ *and a utility policy* $\mathcal{U}$. $\mathcal{P}$ *and* $\mathcal{U}$ *are* ***compatible*** *if and only if, for every* $P \in \mathcal{P}$ *and for every* $U \in \mathcal{U}$ *such that* $\mathsf{body}(U)$ *has an subgraph* GP_U *that is unifiable with* $\mathsf{body}(P)$*, there exists a transformation* $O(G_c)$ *of the canonical graph* G_C *of* $s(\mathsf{body}(P) \cup \mathsf{body}(U))$ *which satisfies* P *and* U*, where* s *is the most general unifier extended to variables in* $\mathsf{body}(U)$ *that are not in* GP_U*, and* O *is a transformation that deletes triples or replaces constants by blank nodes.*

Proof. Two separate cases are possible: either there is no subgraph from $\mathsf{body}(U)$ unifiable with $\mathsf{body}(P)$, or there is such a subgraph.

Case 1: for every $P \in \mathcal{P}$ and for every $U \in \mathcal{U}$, there is no subgraph from $\mathsf{body}(U)$ unifiable with $\mathsf{body}(P)$.

In this case, there is a triple $t \in \mathsf{body}(P)$ that is not unifiable with any triple from $\mathsf{body}(U)$.

Let μ each mapping from $\mathsf{body}(P)$ to G matching the answers of $\mathsf{Ans}(P, G)$: For each μ, let $O(G)$ the operation deleting $\mu(t)$: by construction, the answer matching the considered μ is not an answer of P over $O(G)$ anymore and $\mathcal{P}$ would be satisfied.

For each U, for each μ' from $\mathsf{body}(U)$ to G, and for each triple $t' \in \mathsf{body}(U)$, we have $\mu'(t') \neq \mu(t)$ since $\mu' \circ \mu$ would be an unifier between t and t', which is wrong by hypothesis (t cannot be unified with a triple from $\mathsf{body}(U)$).

Thus, the image of $\mathsf{body}(U)$ by μ' is the same in $O(G)$ and in G. Therefore, $Ans(U, O(G)) = Ans(U, G)$ for every U, and $\mathcal{U}$ is satisfied.

Both policies are satisfied, and therefore compatible.

Case 2: there is $P \in \mathcal{P}$ and $U \in \mathcal{U}$ such that there is a subgraph (pattern) $GP_U \subseteq \mathsf{body}(U)$ unifiable with $\mathsf{body}(P)$.

We rename the variables from P and U so that there are no common variables.

Let G_C the canonical graph of $s(\mathsf{body}(P) \cup \mathsf{body}(U))$, where s is the most general unifier extended to the variables from $\mathsf{body}(U)$ that are not in GP_U by the identity (i.e. variables x such that $s(x) = x$). We have $\mathsf{Ans}(P, G_C) \neq \emptyset$ and $\mathsf{Ans}(U, G_C) \neq \emptyset$.

Let $\mathcal{O}$ the set of possible operations over G_C (triple deletions or replacement by blank nodes): there are a finite number of operations, but it is potentially big. The number of operations replacing constants by blank nodes is computed based on the size of the queries. Let $\{c_1, c_2, ..., c_k\}$ the set of constants from G_C: $k < 2 \times |G_C|$ and $|G_C| < \mathsf{max}(\mathsf{size}(p), \mathsf{size}(u))$.

Then c_1 can be left untouched or replaced by a blank node b_1, c_2 can similarly be left intact or replaced either b_1 or another blank node b_2, and so on (c_k stays identical or is replaced by b_1 or b_2 ... or b_k).Therefore the number of possible transformations is $N = 1 \times 2 \times 3 \; times ... \times k \times (k+1) = (k+1)!$.

Two subcases occur in this situation.

Subcase 2a: For each operation $O \in \mathcal{O}$ preventing P from having answers containing blank nodes over $O(G_C)$, we have $\mathsf{Ans}(U, G_C) \neq \mathsf{Ans}(U, O(G_C))$.

If we cannot find an operation O such that $\mathsf{Ans}(U, G_C) = \mathsf{Ans}(U, O(G_C))$, then there is no way to satisfy $\mathcal{U}$. $\mathcal{P}$ and $\mathcal{U}$ are therefore incompatibles.

Subcase 2b: Otherwise (i.e. there is an operation $O \in \mathcal{O}$ such that $\mathsf{Ans}(P, O(G_C))$ is empty or contains only tuples containing blank nodes, *and* $\mathsf{Ans}(U, O(G_C)) = Ans(U, G_C)$).

Let G a graph such that there is an answer $\bar{c_P} \in \mathsf{Ans}(P, G)$ only containing constants, and there is an answer $\bar{c_U} \in \mathsf{Ans}(U, G)$. Let μ and μ' the respective mapping from $\mathsf{body}(P)$ and $\mathsf{body}(U)$ to G.

If $G_P = \mu(\mathsf{body}(P)) \subseteq \mu'(\mathsf{body}(U))$, then let $GP_U \subseteq \mathsf{body}(U)$ such that $\mu'(GP_U) = \mu(\mathsf{body}(P))$. We know that GP_U and $\mathsf{body}(P)$ are unifiable: $m \circ \mu'$ is an unifier, more specific than their most general unifier s. Let G_C the canonical graph of $s(GP_U \cup \mathsf{body}(P))$.

Then there exists ν an homomorphism from G_C to G_P such that $\mu \circ \mu' = \nu \circ \mathsf{frozen} \circ s$, where some of the distinct constants from G_C can be mapped using ν to identical constants in G_P.

Let ν' is an extension of ν to blank nodes so that $\nu'(c) = \nu(c)$ for $c \in \mathbf{L}$ and $\nu'(b) = b$ for $b \in \mathbf{B}$. Consider the operation O' over G_P built using the operation O over G_C, such that $O' = \nu' \circ O$.

By construction, the only possible mapping μ'' from $\mathsf{body}(P)$ to $O'(G_P)$ can only be obtained through a composition of ν' with a mapping from $\mathsf{body}(P)$ to $O(G_C)$. By construction again, this last mapping either does not exist, or it returns blank nodes among its answers, something that μ'' would do as well. Thus, $\mathsf{Ans}(P, O'(G))$ does not contain the answer corresponding to μ anymore: $\mathcal{P}$ would be satisfied.

However, the mapping μ'' from GP_U to $O'(G_P)$ can be obtained as a composition of ν' with the mapping from GP_U to $O(G_C)$ that preserves answers from U over G_C, thus from U over G_P by applying ν': $\mathcal{U}$ would be satisfied.

Thus, $\mathcal{P}$ and $\mathcal{U}$ are compatible.

□

Theorem 2 is a decidability result that is difficult to turn into complete algorithms that would be feasible in practice due to the underlying enumeration process. However, it entails several *sufficient* conditions for compatibility or incompatibility.

In particular, Theorem 3 provides a sufficient condition for incompatibility based on the notion query containment reminded in Definition 13, while Theorem 4 states a sufficient condition for compatibility that will be the basis of the algorithms presented in Section 4.4.

Theorem 3. *A privacy policy $\mathcal{P}$ and a utility policy $\mathcal{U}$ are incompatible if there exists a utility query $U \in \mathcal{U}$ contained in a privacy query $P \in \mathcal{P}$.*

Proof. Let a query $U \in \mathcal{U}$ contained in a privacy query $P \in \mathcal{P}$. By definition, we have $\mathsf{Ans}(U, G) \subseteq \mathsf{Ans}(P, G)$ for any graph G.

Let G_{bis} the graph obtained from the body of U by replacing each variable by distinct IRIs, and let $\bar{t}$ the tuple of the IRIs corresponding to the distinguished variables of U. Then by construction, $\bar{t} \in \mathsf{Ans}(U, G_{bis})$. And since U is contained in P, we necessarily have $\bar{t} \in \mathsf{Ans}(P, G_{bis})$. Therefore, by definition of the privacy policy, G_{bis} violates P (and therefore $\mathcal{P}$ as well).

Now, let O a sequence of operations such that $O(G_{bis})$ satisfies both $\mathcal{P}$ and $\mathcal{U}$. $\mathsf{Ans}(U, O(G_{bis})) = \mathsf{Ans}(U, G_{bis})$ holds, by definition of the utility policy. We therefore have $\bar{t} \in \mathsf{Ans}(U, O(G_{bis}))$, and, by query containment, $\bar{t} \in \mathsf{Ans}(P, O(G_{bis}))$. This means that P is not satisfied, by definition of a privacy policy: there is a contradiction. □

Example 4.4

Let $\mathcal{P}$ a utility policy consisting of this single query P:

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasAddress  ?ad .
}
```

Let $\mathcal{U}$ a utility policy consisting of this single query U:

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasAddress            ?ad.
    ?ad tcl:professionalAddress     true .
}
```

Here, U is contained in P (all the results from U are included in the ones from P). Therefore $\mathcal{P}$ and $\mathcal{U}$ are incompatible.

Theorem 4. *A privacy policy $\mathcal{P}$ and a utility policy $\mathcal{U}$ are compatible if for every privacy query p in $\mathcal{P}$ there exists a triple pattern in body(p) that is not unifiable with any triple pattern in the body of any utility query u in $\mathcal{U}$.*

Proof. This corresponds to **Case 1** in the proof of Theorem 2. □

We now focus on computing anonymization operations in order to satisfy both policies.

4.4 Finding candidate sets of anonymization operations

Given two *compatible* privacy and utility policies, the second problem of interest that we address in this chapter is the ENUMOPERATIONS problem of generating anonymization

operations which, when applied to any RDF graph violating the privacy policy, will produce an anonymized RDF graph satisfying the privacy policy while preserving the utility policy.

Problem 2. *The* EnumOperations *problem.*

Input : $\mathcal{P} = \{P_i\}$ *a privacy policy and* $\mathcal{U} = \{U_j\}$ *a utility policy*

Output: *A set* $\mathcal{O}$ *of sequences of operations* O *such that* $O(G)$ *satisfies both* $\mathcal{P}$ *and* $\mathcal{U}$ *for any graph* G*.*

Note that an algorithm that solves the EnumOperations problem could also solve the Compatibility problem as well, by checking whether or not its output is $\emptyset$.

This section is devoted to the design of Algorithm 2, which solves the EnumOperations problem (Problem 2) using update operations (Definition 12) when the policies $\mathcal{P}$ and $\mathcal{U}$ are defined by conjunctive queries (Definition 7). We also define an intermediate step dealing with unitary privacy policies, using Algorithm 1, which is then used by Algorithm 2 in the global anonymization process.

Note that Algorithm 2 produces a set of *sets* of anonymization operations, and not a set of *sequences*. Since we guarantee that the *sets* of operations hereby computed solve Problem 2, any sequence obtained by reordering these sets would solve it as well. Hence, the difference between sets and sequences of operations is fairly immaterial in this context, although different sequences may cause different graphs as outputs; every one of them will satisfy both policies nonetheless.

If the answer set of a query Q is preserved by an anonymization process, then so does its cardinality. This implies that any solution for a non-counting query Q is also a solution for its counting counterpart $\mathsf{Count}(Q)$. Similarly, if a utility query Q is satisfied, then its counting counterpart $\mathsf{Count}(Q)$ is also satisfied. Therefore, we focus on non-counting queries in Algorithm 1. However, the opposite implication does not hold, hence we may miss some operations that may guarantee a utility counting query $\mathsf{Count}(Q)$ without guaranteeing a non-counting utility query Q.

4.4.1 For unitary privacy policies

We start with the case where the privacy policy is unitary, i.e. when it is reduced to a singleton $\mathcal{P} = \{P\}$.

The intuition of Algorithm 1 is that it tries to find edges that are in the graph pattern G^P of $P \in \mathcal{P}$ but in none of the utility policy graph patterns G_j^U from queries $U_i \in \mathcal{U}$; more precisely, it tries to unify each triple from the utility queries with the considered triple from the privacy query. As detailed in Section 4.3, we know that a privacy policy and a utility policy are compatible if such an unification is impossible on a graph pattern level, but we also know that checking this is too costly. We therefore opt for a tighter version of this, hence making this verification triple by triple.

For each edge that exists in G^P but not in any G^U_j, a delete operation is constructed, and possible update operations are considered. Update operations can be introduced in two manners: either the subject of the triple is replaced with a blank node, or its object is replaced with a blank node if it is an IRI. In both cases, the algorithm looks for three alternatives:

- The triple is part of a path of length ≥ 2 in the privacy graph pattern G^P, and therefore the update operation breaks the path, thus satisfying the privacy policy $\mathcal{P}$;
- The replaced subject (resp. object) is also the subject (resp. object) of another triple in the privacy query graph G^P and the update operation breaks the link between these triples, hence satisfying the privacy policy $\mathcal{P}$;
- The replaced subject (resp. object) of the triple is also part of the distinguished variables $\bar{x}$ of the privacy policy query, leading to a blank value in the query results, and therefore satisfying the privacy policy $\mathcal{P}$.

The soundness of this algorithm is encapsulated in Theorem 5.

To ease the readability of the algorithm, we define the following helper functions checking if update operations are possible according to the three cases mentioned earlier. These conditions in fact model our notion of *critical terms* (Definition 9). The intuition is to look for triples (s, p, o) that have a certain *centrality in the graph*, i.e. if s or o are used in other triples. This means checking that another triple in the considered graph pattern has s as object or as subject (in a triple different from (s, p, o)), and that another triple has o as a subject or as object (in a triple different from (s, p, o)).

$$\begin{aligned}
\texttt{check-subject}((s,p,o),G) = {} & \exists(s',p',s) \in G \vee \\
& (\exists(s,p',o') \in G \wedge \nexists\sigma\ (\sigma(s,p',o') = \sigma(s,p,o))) \\
\texttt{check-object}((s,p,o),G) = {} & \exists(o,p',o') \in G \vee \\
& (\exists(s',p',o) \in G \wedge \nexists\sigma\ (\sigma(s',p',o) = \sigma(s,p,o)))
\end{aligned}$$

Input : a unitary privacy policy $\mathcal{P} = \{P\}$ with $P = \langle \bar{x}^P, GP^P \rangle$
Input : a utility policy $\mathcal{U}$ made of m queries $U_j = \langle \bar{x}_j^U, GP_j^U \rangle$
Output: a set of operations O satisfying both $\mathcal{P}$ and $\mathcal{U}$

function `find-ops-unit`$(P, \mathcal{U})$**:**
  Let H be the graph GP^P with all its variables replaced by fresh ones[a];
  Let $O := \emptyset$;
  forall $(s, p, o) \in H$ **do**
    Let $c :=$ true;
    forall G_j^U **do**
      forall $(s', p', o') \in GP_j^U$ **do**
        if $\exists \sigma \ (\sigma(s', p', o') = \sigma(s, p, o))$ **then**
          $c :=$ false;
    if c **then**
      $O := O \cup \{$`DELETE` $\{(s, p, o)\}$ `WHERE` $H\}$;
      if `check-subject`$((s, p, o), H) \vee s \in \bar{x}^P$ **then**
        $O := O \cup \{$`DELETE` $\{(s, p, o)\}$ `INSERT` $\{($`[ ]`$, p, o)\}$ `WHERE` $H\}$;
      if $o \in \mathbf{I} \wedge$ (`check-object`$((s, p, o), H) \vee o \in \bar{x}^P$) **then**
        $O := O \cup \{$`DELETE` $\{(s, p, o)\}$ `INSERT` $\{(s, p,$ `[ ]`$)\}$ `WHERE` $H\}$;
  return O;

Algorithm 1: Find update operations to satisfy a unitary privacy policy

[a] i.e., with variables that do not appear in any GP_j^U

To ease readability and structure the proof of the soundness of this algorithm, we need to define intermediary lemmas. Lemma 1 asserts the monotonicity of the queries used in our works. Lemma 2 shows why boolean queries are used in the privacy proof.

Lemma 1 (Graph pattern monotonicity)**.** *Let GP and GP' be graph patterns with $GP \subseteq GP'$. Moreover, let G and G' be RDF graphs with $G \subseteq G'$. The following inclusions hold for all sets $\bar{x}$ of variables of GP.*

$$\mathsf{Ans}(\langle \bar{x}, GP' \rangle, G) \subseteq \mathsf{Ans}(\langle \bar{x}, GP \rangle, G) \subseteq \mathsf{Ans}(\langle \bar{x}, GP \rangle, G')$$

Proof. Let $\bar{c} \in \mathsf{Ans}(\langle \bar{x}, GP' \rangle, G)$. By definition, there exists some μ such that $\mu(\bar{x}) = \bar{c}$ and $\mu(GP') \subseteq G$ and in particular $\mu(GP) \subseteq \mu(GP')$. By the transitivity of inclusion, both inclusions hold. □

Lemma 2 (Boolean satisfiability)**.** *Let a query $Q = \langle \bar{x}, GP \rangle$, let G an RDF graph and let GP' be a subset of GP together with a function η such that $\eta(GP) \subseteq \eta(GP')$. Then* $\mathsf{Ans}(\langle \bar{x}, GP \rangle, G) = \emptyset$ if and only if $\mathsf{Ans}(\langle \langle \rangle, GP' \rangle, G) = \emptyset$

Proof. Let us denote the inclusion $GP' \subseteq GP$ by a function ι such that $\iota(GP') \subseteq \iota(GP)$. We prove the *only if* direction by contraposition. Assume that there is an answer in $\mathsf{Ans}(\langle\langle\rangle, GP'\rangle, G)$. By the definition of Ans, there is at least one μ such that $\mu(GP') \subseteq \mu(G)$. By composing μ and η we obtain $\mu \circ \eta(GP) \subseteq \mu \circ \eta(G)$, thus $\mathsf{Ans}(\langle\bar{x}, GP\rangle, G)$ is not empty. We prove the *if* direction by contraposition similarly. Assume that there is an answer in $\mathsf{Ans}(\langle\bar{x}, GP\rangle, G)$ and call it ν such that $\nu(GP) \subseteq \nu(G)$. By composing ν and ι we obtain $\nu \circ \iota(GP') \subseteq \nu \circ \iota(G)$, thus $\mathsf{Ans}(\langle\langle\rangle, GP'\rangle, G)$ is not empty. □

Finally, Lemma 3 encompasses the soundness of our privacy model.

Lemma 3 (Soundness for privacy)**.** *Let $q = \langle\bar{x}, GP\rangle$ be a query, let H be GP renamed with fresh variables, and let a triple $(s, p, o) \in H$. For every RDF graph G, the following update queries satisfy privacy policy $\mathcal{P} = \{q\}$:*

$$\texttt{DELETE}\ \{(s,p,o)\}\ \texttt{WHERE}\ H, G)$$

$$\texttt{DELETE}\ \{(s,p,o)\}\ \texttt{INSERT}\ \{(x_u,p,o)\}\ \texttt{WHERE}\ H, G)$$

$$\texttt{DELETE}\ \{(s,p,o)\}\ \texttt{INSERT}\ \{(s,p,x_u)\}\ \texttt{WHERE}\ H, G)$$

where $x_u \in \mathbf{B}$ a fresh blank node (equivalent to the `[ ]` convention used so far).

Proof. Let's consider the three possible cases.

First query: Let $G_{anon} = \mathsf{Delete}(\{(s,p,o)\}, H)(G)$ the graph obtained after deletion. By Lemma 2 and by definition of query answers and privacy policy satisfiability, it is equivalent to prove that $\mathsf{Ans}(\langle\langle\rangle, H\rangle,') = \emptyset$ that is, to prove that there is no ν such that $\nu(H) \subseteq \nu(G_{anon})$. For the sake of contradiction, assume that such a ν exists.

Let's consider the triple $\nu(s,p,o) \in G_{anon}$. On the other hand, $G_{anon} = G \setminus \{\mu(s,p,o) \mid \mu(H) \subseteq \mu(G)\}$ by the definition of Delete, but picking $\mu = \nu$ shows that $\nu(s,p,o) \notin G_{anon}$, which is a contradiction.

Second query: Let $G_{anon} = \mathsf{Update}(\{(s,p,o)\}, \{x_u,p,o)\}, H)(G)$ the graph obtained after subject update. Three possibles cases can trigger this operation.

- **Case 1:** $\exists(s', p', s) \in GP$

Let $a \in Ans(\langle\langle\rangle, H\rangle, G_{anon})$ an answer on G_{anon}, so that $\exists\mu \mid \mu(H) \subseteq G_{anon}$. In particular, this applies to the subgraph $\bar{H} = \{(s,p,o),(s',p',s)\}$, and we have $\mu(\bar{H}) \subseteq G$, which is equivalent to $\{(\mu s, \mu p, \mu o), (\mu s', \mu p', \mu s)\} \subseteq G_{anon}$.

But by the definition of Update, we have $G_{anon} = G \setminus \{\nu(s,p,o) | \nu(H) \subseteq \nu(G)\} \cup \{\nu(x_u,p,o) | \nu(H) \subseteq \nu(G)\}$, as we replace every subject of matching triples (s,p,o) by a fresh blank node. Therefore with $\mu = \nu$, we have $\mu s = b \in \mathbf{B}$, a fresh blank node.

We would then have $\mu(\bar{H}) = \{(b, \mu p, \mu o), (\mu s', \mu p', b)\}$, which is not possible since b is by construction created as a fresh variable in each insertion and cannot be found in two different triples: there is a contradiction and a cannot exist. With this operation and this condition $Ans(\langle\langle\rangle, H\rangle, G_{anon}) = \emptyset$ and the privacy is fulfilled.

- **Case 2:** $\exists(s, p', o') \in GP$ **and** $\nexists\sigma\ (\sigma(s, p', o') = \sigma(s, p, o))$

We apply the same methodology as in Case 1: Let $a \in Ans(\langle\langle\rangle, GP\rangle, G_{anon})$ an answer, and let a subgraph $\bar{GP} = \{(s, p, o), (s, p', o')\}$, and we then have $\mu(\bar{GP}) = \{(\mu s, \mu p, \mu o), (\mu s, \mu p', \mu o')\} \subseteq G_{anon}$. By construction of Update, we have $\mu s = b \in \mathbf{B}$, a fresh blank node. We would then have $\mu(\bar{GP}) = \{(b, \mu p, \mu o), (b, \mu p', \mu o')$. Plus, by hypothesis, (s, p, o) and (s, p', o') are not unifiable. Therefore, such a case is not possible and $Ans(\langle\langle\rangle, GP\rangle, G_{anon})$ must be empty. The privacy condition is satisfied.

- **Case 3:** $s \in \bar{x}^P$

Let's consider an answer $\mathsf{Ans}(\langle\bar{x}, GP\rangle, G_{anon})$. By definition of Update, $G_{anon} = G \setminus \{\mu(s, p, o) | \mu(GP) \subseteq \mu(G)\} \cup \{\mu(x_u, p, o) | \mu(GP) \subseteq \mu(G)\}$, as we replace every subject of matching triples (s, p, o) by a fresh blank node x_u. By hypothesis, $s \in \bar{x}^P$, therefore $\forall a \in \mathsf{Ans}(\langle\bar{x}, GP\rangle, G_{anon}), mus \in a$, that is $\exists x_u \in \mathbf{B} \mid x_u \subseteq a$. Which entails that for any tuple full of constants $\bar{c}$, $\bar{c} \notin \mathsf{Ans}(\langle\bar{x}, GP\rangle, G_{anon})$, which means that the privacy is ensured.

Third query: Let $G_{anon} = \mathsf{Update}(\{(s, p, o)\}, \{s, p, x_u)\}, GP)(G)$ the graph obtained after value update. Let's consider the triple $\nu(s, p, o) \in G_{anon}$.

We consider the same 3 cases as the second query and show using the same rules that:

- In the first case ($\exists(o, p', o') \in GP$), an answer to the query would mean that $\mu(\bar{GP}) = \{(s, \mu p, b), (b, \mu p, o)\}$ with $b \in \mathbf{B}$ a fresh blank node, which is not possible.
- In the second case ($(\exists(s', p', o) \in GP$ **and** $\nexists\sigma\ (\sigma(s', p', o) = \sigma(s, p, o))$, an answer would imply $\mu(\bar{GP}) = \{(\mu s, \mu p, b), (\mu s', \mu p', b)$ with $b \in \mathbf{B}$ a fresh blank node, which is not possible.
- In the third case ($o \in \bar{x}^P$), we have $\forall a \in \mathsf{Ans}(\langle\bar{x}, GP\rangle, G_{anon}), \mu o \in a$, that is $\exists b \in \mathbf{B}$ such that $b \in a$.

□

Theorem 5 (Soundness of Algorithm 1)**.** *Let $\mathcal{P}$ be a privacy policy consisting of a single query and let $\mathcal{U}$ be a utility policy. Let $O = \texttt{find-ops-unit}(\mathcal{P},\mathcal{U})$ computed by Algorithm 1. For all $o \in O$, for any RDF graph G, $\mathcal{P}$ and $\mathcal{U}$ are satisfied by $o(G)$ obtained by applying the update operation o to G.*

Proof. The privacy query P from $\mathcal{P}$ is satisfied because each operation created at Lines 11, 13 and 15 of Algorithm 1 is of a form covered by Lemma 3 for all choice of $(s,p,o) \in H$ made in the main loop at Line 4.

Next, we check that all queries $U_j = \langle \bar{x}_j^U, GP_j^U \rangle$ from $\mathcal{U}$ are satisfied, i.e. that $\mathsf{Ans}(GP_j^U, o_k(G)) = \mathsf{Ans}(GP_j^U, G)$ for all $U_j \in \mathbf{U}$.

Let $j \in [1..m]$ and $a \in \mathsf{Ans}(GP_j^U, G)$ an answer of GP_j^U on G. By definition of Ans, $a = \mu(\bar{x}_j^U)$ for some μ such that $\mu(GP_j^U) \subseteq \mu(G)$. We show that $\mu(GP_j^U) \subseteq \mu(o_k(G))$ as well, so $a \in \mathsf{Ans}(GP_j^U, o_k(G))$ and the proof is complete.

We now have to show that $\mathsf{Ans}(GP_j^U, o_k(G)) \subseteq \mathsf{Ans}(GP_j^U, G)$ We explore the three possibilities given by Lines 11, 13 and 15 of the algorithm to be applied as o_k.

Line 11: Let consider $t' = (s', p', o') \in GP_j^U$, for the sake of contradiction, assume that $\mu(t') \notin o_k(G)$, that is $\mu(t') \in DB \setminus o_k(G)$. By construction in Algorithm 1 and by the definition of the Delete operation $G \setminus o_k(G) = G \setminus \mathsf{Delete}(\{(s,p,o)\}, H)(G) = G \setminus G \setminus (\bigcup\{\nu(s,p,o) \mid \nu(H) \subseteq \nu(G)\}) = (\bigcup\{\nu(s,p,o) \mid \nu(H) \subseteq \nu(G)\})$. Thus $\mu(t') \in G \setminus o_k(G)$ implies that $\mu(t') = \nu(t)$ for some $t = (s,p,o) \in H$ and $\nu(H) \subseteq \nu(G)$. As μ and ν have distinct domains thanks to the renaming of G^P, they can be combined into a function σ such that $\sigma(t') = \sigma(t)$ defined by $\sigma(v) = \mu(v)$ when $v \in \mathsf{dom}(\mu)$, $\sigma(v) = \nu(v)$ when $v \in \mathsf{dom}(\nu)$ and $\sigma(v) = v$ otherwise. But this is precisely the condition at Line 8 so $o_k \notin O$. We obtained the desired contradiction so $a \in \mathsf{Ans}(U_j^U, o_k(G))$ and the proof is complete.

Line 13:

Let $j \in [1..m]$ and $a \in \mathsf{Ans}(GP_j^U, o_k(G))$ an answer of GP_j^U on $o_k(G)$. By definition of Ans, $a = \mu(\bar{x}_j^U)$ for some $\mu(GP_j^U) \subseteq \mu(G)$ and by definition of Update, we have $\mu(GP_j^U) \subseteq G \setminus \mathsf{Update}(\{(s,p,o)\}, \{(x_u,p,o)\}, H)(G)$. We can write this as $\mu(GP_j^U) \subseteq G \setminus d(G) \cup i(G)$ where d is the $d(G)$ and $i(G)$ are two graphs, respectively consisting in triples deleted from G and added to G by the Update operation. $a \in \mathsf{Ans}(GP_j^U, G)$ would imply that $\mu(GP_j^U) \cap i(G) = \emptyset$.

Let t a triple (s,p,o) such that $t \in \mu(GP_j^U)$ and $t \in i(G)$. The first criteria gives: $\exists t^U \in GP^U \mid t = \mu(t^U)$. Additionally, $t \in i(G)$ implies $\exists t^P \in H, \exists \mu^P, \exists s \mid \mu^P(t^P) = t = (s,p,o)$. We write t^U as (s^U, p^U, o^U) and t^P as (s^P, p^P, o^P). Therefore, the following equalities must hold:

$$\mu(s^U) = x_u,\ \mu(p^U) = p,\ \mu(o^U) = o$$
$$\mu^P(s^P) = s,\ \mu^P(p^P) = p,\ \mu^P(o^P) = o$$

We have a contradiction, since $\mu(t^U) \neq \mu^P(t^P)$ while by hypothesis, $\mu(t^U) = \mu^P(t^P) = t$. Therefore this indeed proves that $\mu(GP_j^U) \cap i(G) = \emptyset$, and by deduction that $\mathsf{Ans}(G_j^U, o_k(G)) \subseteq \mathsf{Ans}(GP_j^U, G)$.

Line 15:

We apply the same reasoning as the Line 13 proof, showing that in this case we would obtain $\mu(t^U) = (s, p, x_u)$ by definition of the Update operation used at this line, while $\mu^P(t^P) = (s, p, o)$.

This proves again that $\mu(GP_j^U) \cap i(G) = \emptyset$, and therefore that $\mathsf{Ans}(GP_j^U, o_k(G)) \subseteq \mathsf{Ans}(GP_j^U, G)$.

□

The behavior of Algorithm 1 is illustrated in the following Example 4.5.

Example 4.5 Application of Algorithm 1

Consider the policies $\mathcal{P} = \{P_1, P_2\}$ and $\mathcal{U} = \{U_1, U_2\}$ given in Example 4.1 with bodies G_1^P, G_2^P, G_1^U and G_2^U, respectively. Let us consider two different runs of Algorithm 1. The call to `find-ops-unit(`P_1,$\mathcal{U}$`)` produces the following set O_1 of operations whereas the call to `find-ops-unit(`P_2,$\mathcal{U}$`)` produces O_2:

$O_1 = \{$`DELETE` $\{(?u, \texttt{vcard:hasAddress}, ?ad)\}$ `WHERE` G_1^P,

`DELETE` $\{(?u, \texttt{vcard:hasAddress}, ?ad)\}$ `INSERT` $\{(\texttt{[ ]}, \texttt{vcard:hasAddress}, ?ad)\}$ `WHERE` G_1^P,

`DELETE` $\{(?u, \texttt{vcard:hasAddress}, ?ad)\}$ `INSERT` $\{(?u, \texttt{vcard:hasAddress}, \texttt{[ ]})\}$ `WHERE` $G_1^P\}$

$O_2 = \{$`DELETE` $\{(?c, \texttt{tcl:user}, ?u)\}$ `WHERE` G_2^P,

`DELETE` $\{(?c, \texttt{tcl:user}, ?u)\}$ `INSERT` $\{(\texttt{[ ]}, \texttt{tcl:user}, ?u)\}$ `WHERE` G_2^P,

`DELETE` $\{(?c, \texttt{tcl:user}, ?u)\}$ `INSERT` $\{(?c, \texttt{tcl:user}, \texttt{[ ]})\}$ `WHERE` $G_2^P\}$

Indeed, there is only one way to satisfy P_1, U_1 and U_2: delete or update the address $?ad$ of each user $?u$ as shown in O_1. This goes by either deleting it, replacing the address value by a blank node in the `hasAddress` triple (possible since $?ad$ is also a distinguished variable), or replacing the user with a blank node (possible since there is another triple originating from the user variable $?u$ in the policy query body). Notice that the update or deletion of the triple `(?u,a,tcl:User)` is not authorized, because U_1 would not be satisfied.
The only acceptable operations for P_2, U_1 and U_2 as shown in O_2, are either to delete the link between users and their journeys, or replace each argument of this relation with a blank node. Replacing the subject of the considered triple (the journey variable $?c$) is possible since it is also featured as the subject of other triples in the query body, while replacing the object (the user variable $?u$) is possible since it is a distinguished variable of the privacy query.

4.4.2 General case

We now extend the previous algorithm to the general case where $\mathcal{P}$ is a set of n queries. The idea is to compute operations that satisfy each P_i using Algorithm 1 and then to distribute the results. The soundness of this algorithm is encapsulated in Theorem 6 and its associated Corollary 1.

Input : a privacy policy $\mathcal{P}$ made of n queries $P_i = \langle \bar{x}_i^P, G_i^P \rangle$
Input : a utility policy $\mathcal{U}$ made of m queries $U_j = \langle \bar{x}_j^U, G_j^U \rangle$
Output: a set of sets of operations Ops such that each sequence obtained from ordering any $O \in Ops$ satisfies both $\mathcal{P}$ and $\mathcal{U}$

1 **function** `find-ops(`$\mathcal{P}$,$\mathcal{U}$`)`**:**
2 Let $Ops = \{\emptyset\}$;
3 **for** $P_i \in \mathcal{P}$ **do**
4 Let $ops_i :=$ `find-ops-unit(`P_i,$\mathcal{U}$`)`;
5 **if** $ops_i \neq \emptyset$ **then** $Ops := \{O \cup \{o'\} \mid O \in Ops \wedge o' \in ops_i\}$;
6 **return** Ops;

Algorithm 2: Find update operations to satisfy policies

Theorem 6 (Soundness of Algorithm 2)**.** *Let $\mathcal{P}$ be a privacy policy and let $\mathcal{U}$ be a utility policy. Let $\mathcal{O} =$ `find-ops`$(\mathcal{P},\mathcal{U})$ and let G be an RDF graph. For any set of operations $O \in \mathcal{O}$, and for any ordering S of O, $\mathcal{P}$ and $\mathcal{U}$ are satisfied by $S(G)$ obtained by applying to G the sequence of operations in S.*

Proof. First, note that O_k is either $\emptyset$ when some ops_i is empty or it is of the form $O_k = \{o_1, \ldots, o_n\}$ with $n = |\mathcal{P}|$. Indeed, the loop at Line 3 is executed once for each P_i, so at line 5, either one ops_i is empty and thus $Ops = \emptyset$ because $\{O \cup \{o'\} \mid O \in Ops \wedge o' \in \emptyset\} = \emptyset$, or all $ops_i \neq \emptyset$ an each $O_k \in Ops$ contains exactly one operation for each P_i.

By construction of Algorithm 2 and by Theorem 5, each $o \in O_k$ satisfies at least one of the P_i and all U_j and each P_i is satisfied by at least one $o \in O_k$. Thus any choice of an ordering S_k of O_k is such that all P_i are satisfied.

□

Theorem 6 guarantees the soundness of *all* sequences of operations built from the output of Algorithm 2. Corollary 1 uses this result for the COMPATIBILITY problem.

Corollary 1. *Let $\mathcal{P}$ be a privacy policy and let $\mathcal{U}$ be a utility policy made of counting and non-counting queries. If* `find-ops`$(\mathcal{P},\mathcal{U}) \neq \emptyset$ *then the* COMPATIBILITY *problem has* `True` *as a solution.*

The sets of operations produced by Algorithm 2 are *not* equivalent in the sense that they may delete *different sets* of triples in the dataset. Moreover, even for a *given* set of

operations, the choice of a possible reordering of its operations may have different effects on the dataset. Indeed, deletions and modifications of triples are not commutative operations but due to the soundness of the algorithm, every obtained solution *satisfies* the privacy and utility policies.

Regarding the complexity of Algorithm 1, its result $O = \texttt{find-ops-unit}(P,\mathcal{U})$ grows linearly with the size of P. Indeed, each triple in the body G^P of P produces at most one delete operation and two update operations. However, regarding the overall complexity of Algorithm 2, if each set O of operations $O \in \mathcal{O} = \texttt{find-ops}(\mathcal{P},\mathcal{U})$ has cardinality $|\mathcal{P}|$ by construction, the distribution of the results obtained by `find-ops-unit` on line 4 induces an exponential blowup on the size of $\mathcal{O}$ due to the cartesian product on Line 5. In our experimental assessment (Section 4.6), we will show that in practice the utility and privacy queries in $\mathcal{P}$ and $\mathcal{U}$ oftentimes overlap, thus decreasing drastically the actual number of sequences output by Algorithm 2, possibly to none.

4.5 Supporting ontologies for the anonymization process

As stated in Section 3.1.3, one of the main characteristics and interesting aspects of RDF is its ability to shape *Linked* Data, by linking resources from various sources and by introducing reasoning based on (external or local) additional *meta* knowledge as **ontologies**. While the dangers and circumvention of IRI re-utilization and linking will be studied in the following Chapter 5 and its contribution, we now consider how the use of ontologies could affect the usage of our framework and our privacy guarantees.

In fact, our query-based approach can be combined with ontology-based query rewriting and thus can support reasoning for first-order rewritable ontological languages such as RDFS [Brickley14], DL-Lite [Calvanese07] or EL fragments [Hansen15]. More precisely, given a pair of privacy and utility policies made of conjunctive queries defined over an ontology, each set of anonymization operations returned by Algorithm 2 applied to the two sets of their corresponding conjunctive rewritings (obtained using existing query rewriting algorithms, such as the ones from [Bursztyn15, Calvanese07, Hansen15]) will produce datasets that are guaranteed to satisfy the policies.

We provide some examples of such rewritings in the following sections, modeling situations where additional knowledge causes privacy breachs.

4.5.1 Incompatibility modulo knowledge

An additional factor to take into account is the fact that one may have additional knowledge regarding existing data. We must therefore extend the basic definition of query containment to account for this, which is done in Definition 16: a query Q_1 is contained

in a query Q_2 modulo some knowledge K if for each answer of Q_1 over G, we can find it in the answers of Q_2 *or* infer an answer of Q_2 by using K.

Definition 16 (Query containment modulo knowledge)**.** *Let Q_1 and Q_2 two conjunctive queries, and let K a graph representing external knowledge. Q_1* ***is contained in*** *Q_2* ***modulo*** *K if and only if for any graph G, $\mathsf{Ans}(Q_1, G) \cup K \models \mathsf{Ans}(Q_2, G)$.*

Thus, the algorithms checking if a query Q_1 is contained in another query Q_2 (in our context, if a utility query U is contained in a privacy query P) must be adapted as well to incorporate K in their process. There are two alternatives to do so, which can be seen as contextual adaptations of the backward- and forward-chaining reasoning techniques:

- **Solution 1:** Rewrite Q_2/P using K and add the rewritten query to the privacy policy;
- **Solution 2:** Complete Q_1/U using K and check the containment of the completed query with the privacy queries.

We illustrate both strategies in the following Example 4.6.

In short, the use of ontologies is perfectly compatible with this privacy framework *provided that reasoning has been applied to the policy queries beforehand*.

4.5.2 Fixing incompatibility between policies

In this more prospective section, we explore possible solutions to solve incompatibility. Once an incompatibility factor has been found between two policies, clashing elements must be resolved by modifying the privacy and utility policies. This would usually be done by discussions between the data provider and the people in charge of the anonymization process. We describe on of the main strategies that could be use to do so.

This solution, depicted on Example 4.7, is to *specialize privacy queries*, by being more specific in the triples requiring to be hidden.

In this example, the new privacy policy $\{P'\}$ and the original utility policy $\{U\}$ would now be compatible, even modulo the knowledge K from Example 4.6.

This type of approach is not the core of this work, as we focus more concretely on data anonymization algorithms, but future research regarding incompatible policies is an important complimentary line of work. Notably, the definition or formalization of optimal way to rewrite policies is very relevant.

Example 4.6 Illustration of query containment modulo some knowledge

Let $\mathcal{P}$ a utility policy consisting of this single query P:

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasAddress  ?ad .
}
```

Let $\mathcal{U}$ a utility policy consisting of this single query U:

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasProfessionalAddress  ?ad.
}
```

Finally, let K the following knowledge:

```
vcard:hasProfessionalAddress  rdfs:subPropertyOf  vcard:hasAddress
```

As is, U is not contained in P, but using K we can infer that U is contained modulo K in P.
If we address this using solution 1, we would have $\mathcal{P} = \{P, P'\}$ with $P' = \text{Rewriting}(P, K)$, giving the following query P' which would contain U.

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasProfessionalAddress  ?ad.
}
```

Using solution 2, we would have a new $u = Complete(U, K)$ giving that U would be contained in P.

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasProfessionalAddress  ?ad.
    ?u  vcard:hasAddress  ?ad.
}
```

In both cases, we successfully find that these policies would be incompatible.

4.6 Experimental evaluation

We design an empirical study devoted to gauge the efficiency of the main algorithm of the privacy/utility framework (Algorithm 2 from Chapter 4) and measure the impact of the overlap and the size of the policy queries on its output.

For this study, we need to focus on various characteristics of this solution that will form the three parts of our experiments:

Example 4.7 Specializing a privacy query

Using the original privacy query P from Example 4.6, an example of a more specific privacy query based on P would be the following query P':

```
SELECT ?ad
WHERE {
    ?u  a   tcl:User .
    ?u  vcard:hasPersonalAddress  ?ad.
}
```

1. An experimental analysis of the risk of incompatibility between privacy and utility policies;
2. The experimental evaluation of the impact of the privacy and utility policies on the number of anonymizations alternatives produced by Algorithm 2;
3. Evaluating Algorithm 2's runtime performance.

4.6.1 Setup and goals

As mentioned in Section 6.1.3, given that this solution already provides formal guarantees on the data privacy and utility of the input graph through the input policies, this experimental study will rather be focused on other factors impacting the anonymization process.

While some experiments can be performed with real, concrete RDF graphs assuming the position of a Data Protection Officer, i.e. building manual and plausible policies, another possibility is to automate the creation of policies by using a generator or a query workload using a predefined graph schema and study how the structure of queries impacts various factors in the anonymization process. This is also in line with the need for an explicit usage workload expressed in many works studied previously. To do so, we use gMark [Bagan17], a schema-based synthetic graph and query workload generator, as a benchmark for our experimental study when an extensive and non-deterministic query batch is required. Due to the static nature of our approaches, we only need to use such a schema to generate query workloads without the need of generating actual graph instances. These workloads will then be used to compose privacy or utility policies.

We setup gMark by using the graph schema of our running transportation example, by including types and properties described in Section 1.5. Precisely, we defined a schema with 13 data types and 12 properties capturing information regarding users (including personal data and subscription data for cardholders), ticket validations and user rides (such as geographic coordinates of ticket validations and optional subscription-related data), and information on the transportation network (such

as maps). Using gMark, we then built a sample of 500 randomly generated conjunctive queries upon the aforementioned schema, each one containing between 1 and 6 distinguished variables with a size ranging between 1 and 6 triples. As shown in a recent study [Bonifati17], queries of such size are the most frequent ones in a large corpus of real-world query logs extracted from SPARQL endpoints. This further corroborates our assumption that our query sample is representative of real-world queries formulated by end-users. To account for the structural variability of real-world queries, experiments were performed on workloads using different shapes of queries: chain queries, star queries, star-chain queries and a random mix of star-chain and star queries. To ease readability, we present the results for star-chain queries only. The full list of experiments is available in a notebook at the project's GitHub repository, and results for other types of query are mentioned in Appendix C.

To generate privacy and utility policies, we fix a number of conjunctive queries to be part of the privacy and utility policies. Then, we randomly pick as many queries as necessary in the query sample to build the policies based on this cardinality, while avoiding duplicates in the same policy and in between both kinds of policies.

In all our experiments, we have opted for a balanced cardinality between privacy and utility policies: we have set the policy cardinality equal to 3 for the experiments in Sections 4.6.2 and 4.6.3. Depending on the experiment, policy size (i.e. the sum of the sizes of the conjunctive queries defining it) may vary since the picked queries have a varying size from 1 to 6.

To study differently how parts of the privacy and utility queries may overlap each other, we define a metric computing a numerical value modeling this fact. We define the *overlap degree* as the ratio between the number of triples appearing in privacy queries that can be mapped to a triple appearing in a utility query and the total size of the privacy policy. More formally, let $\mathcal{P} = \{P_i\}$ and $\mathcal{U} = \{U_j\}$ be privacy and utility policies. The overlap degree between $\mathcal{P}$ and $\mathcal{U}$ is a real number in $[0 \dots 1]$ defined as:

$$\mathsf{overlap}(\mathcal{P},\mathcal{U}) = \frac{\sum_{i=1}^{n} |\{t \in \mathsf{body}(P_i) \mid \exists j\ \exists t' \in \mathsf{body}(U_j)\ \exists \mu\ \mu(t) = \mu(t')\}|}{\sum_{i=1}^{n} |\,\mathsf{body}(P_i)|}$$

When privacy queries *fully overlap* with utility policies, i.e., when their overlap degree is equal to 1, the risk of incompatibility is high and it corresponds to the case where Algorithm 2 returns $\emptyset$ as output.

In Section 4.6.2, we will measure the risk of incompatiblity between randomly generated privacy and utility policies by counting the number of cases where this full overlap occurs.

The algorithms have been implemented and evaluated using Python 2.7. The code makes use of various libraries, notably the `rdflib` package[1] to manipulate RDF data,

[1] https://github.com/RDFLib/rdflib

the `fyzz` package[2] to parse SPARQL queries, and the `unification` package[3] to check the compatibility between privacy and utility policies using the unification of variables in triple patterns. At the time, all these tests were performed under Windows 10 on a Intel® Core™ i5-6300HQ CPU machine running at 2.30GHz and 8GB of RAM. One of the benefits of dealing with a query-driven static method for anonymization is to avoid dealing with the size of an input graph, which could impact performance by increasing runtime. Our static approach only deals with policy size when looking for candidate anonymization sets, which is likely to make the algorithm simple and efficient in general. To confirm this, we ran the Algorithm 2 for a batch of 100 executions corresponding to input privacy and utility policies of 10 queries each, and we measured the average running time. We obtained an average runtime of **0.843 seconds** over all executions, which turns to be satisfactory in practice. We can thus conclude that this static approach provides a fast way to enumerate all the candidate sets of anonymization operations.

4.6.2 Measuring overlapping degree between privacy and utility policies

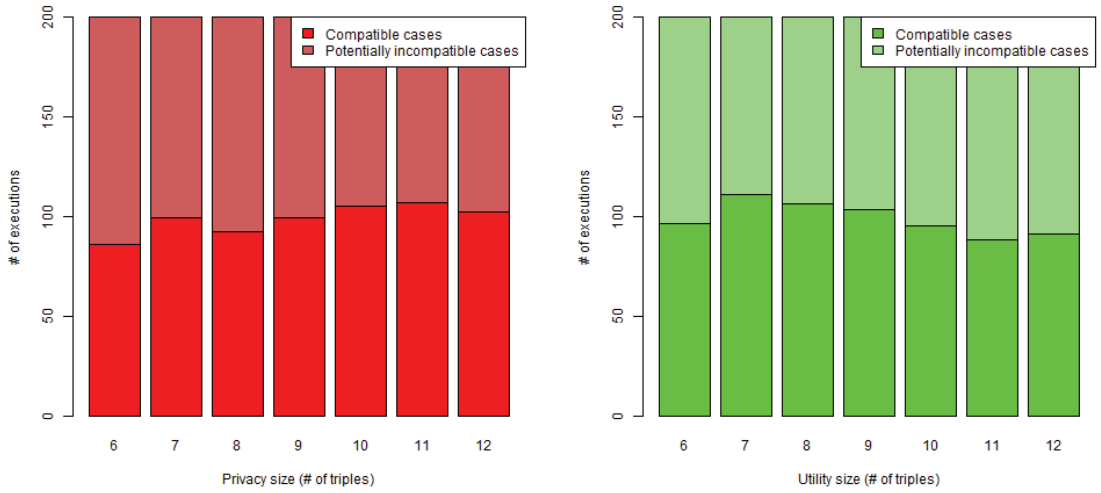
Our goal is to estimate the risk of incompatibility of privacy and utility policies randomly generated with a fixed cardinality of 3 and a varying size.

We have performed two experiments where we vary the size of the privacy (resp. utility) policy from 6 to 12, which corresponds to privacy (resp. utility) queries having between 2 and 4 triples, while keeping the size of the utility (resp. privacy) policy fixed to 9, which corresponds to utility (resp. privacy) queries with 3 triples. In the first (resp. second) experiment, for each of the 7 privacy (resp. utility) policy sizes, we launch 200 executions of Algorithm 2 and we count the number of executions returning $\emptyset$, which allows to compute the proportion of fully overlapping policies. For space reasons, we omit the corresponding histograms (available in our online notebook) and we describe the obtained results in the following.

Results are displayed on Figure 4.1. In both experiments, we observed that only **49%** of the 1400 (corresponding to 200 runs multiplied by 7 data points) executions exhibit compatible policies. This result clearly shows the necessity of designing an algorithm which automatically verifies policy incompatibility prior to the the anonymization process. It also reveals that even small policy cardinalities (equal to 3 for privacy and utility queries) can already substantially prevent possible anonymizations.

We also noted in the first experiment (Figure 4.1a) that the overlapping rate between privacy and utility policies tends to grow with the privacy policy size. This behavior is in clear contrast with the intuition that the more privacy policy is constrained, the less flexibility we have in satisfying them. The explanation however is that increasing the size of the privacy policy decreases the risk that all its triples are mapped with triples

[2] https://pypi.org/project/fyzz/
[3] https://pypi.org/project/unification/0.2.2/



(a) For privacy policies

(b) For utility policies

Figure 4.1: Compatiblity of privacy and utility policies in function of their size

in the (fixed size) utility policy, and thus augments the possibilities of satisfying the privacy and utility policies by triple deletions.

We observe the opposite trend in the second experiment (Figure 4.1b): the overlapping rate between privacy and utility policies decreases with the utility policy size. The reason is that requiring more utility for end-users restrains the possibilities of deleting data for anonymization purposes.

4.6.3 Measuring the number of anonymization alternatives

When applied to compatible privacy and utility policies, Algorithm 2 computes the set of all the candidate sets of update operations that satisfy the input policies. In the worst case, the number of candidate sets corresponds the product of the sizes of the privacy queries. In this experiment, we want to evaluate how this number evolves in practice depending on (1) the overlap between privacy and utility policies, and (2) the total size of the privacy and utility policies.

Algorithm 2 has been run on 7000 randomly generated combinations of privacy and utility policies, thus covering a wide spectrum of combinations exhibiting various overlap degrees with various types of queries. For each execution, we compute the overlap degree between the input privacy and utility policies and group results in clusters of 10% before plotting as a boxplot the number of candidate sets in executions featuring

the given overlap degree (Figure 4.2). This provides a representation of how many alternatives our algorithm provides for anonymizing a graph, depending on the policies overlap. The boxplot allows to visualize both extreme values and average trends, given that the randomization can easily create extreme cases and outlier values.

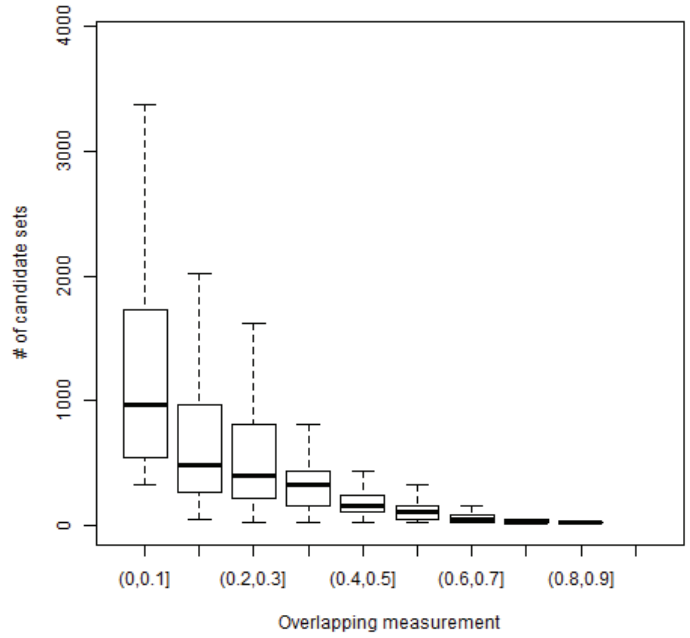
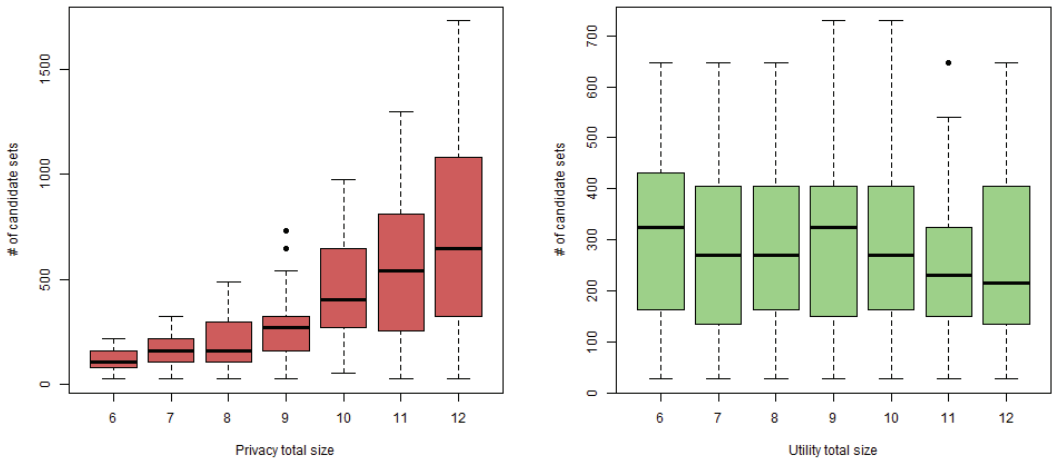


Figure 4.2: Candidate set length based on policy overlap

We can observe that the number of candidate sets quickly decreases when overlapping grows even slightly. This is easy to understand, given that increasing overlap degree induces that less deletion operations are permitted by the algorithm. As soon as the overlap degree reaches an high value, our algorithm provides very few anonymization alternatives since no possible operation exists to satisfy the given policies.

We use the same experimental settings as in Section 4.6.2 to evaluate how the number of candidate sets evolves as a function of policy size; results are displayed on Figure 4.3.

Figure 4.3a displays the results of this experiment when varying privacy size with a fixed utility size of 9 triples. We can observe a steady increase of the number of candidate sets with the privacy size. The explanation for this behavior is that increasing privacy size (with fixed utility size) provide more possible operations for the anonymization.



(a) Depending on privacy size

(b) Depending on utility size

Figure 4.3: Candidate set length based on the size of both policies

On the other hand, when varying utility size (with fixed privacy size), the number of candidate sets almost stagnates when increasing the utility size (Figure 4.3b). This means that increasing the size of utility queries without increasing the number of queries itself in the utility policies does not significantly reduce the anonymization opportunities as there are still as many candidates for anonymization.

In short, this experiment emphasizes the faint influence of utility policies on possible anonymizations sets, along with the crucial role of privacy policies in shaping possible anonymization operations.

5

Safety beyond privacy: Anonymization robust to data linkage

▷ *While the previous contribution focused on guaranteeing privacy by anonymizing a local graph using privacy and utility constraints, we now focus on guaranteeing safety, i.e. privacy in a global context featuring graph linking in addition to basic data disclosure constraints. We motivate this approach and emphasize the difference between the privacy and safety approaches, and design two variations of an anonymization algorithm for this new context, using our framework. Finally, we study how RDF linking mechanisms, implicit or explicit, can affect query answering, safety guarantees, and therefore our algorithms as well.* ◁

Chapter summary

In Chapter 4, we studied and focused on the notion of *privacy*, where privacy and utility policies were used in order to finely control the disclosure of data through an anonymization algorithm. We now show that, in the context of LOD, this privacy guarantee is not sufficient because data linkage with external RDF graph may cause privacy leakages. This contribution has been published in [Delanaux20].

5.1 Linkage creates privacy leaks

As expressed in Chapter 1 and Section 3.1, one of the core aspects of RDF graphs is that they can share entities, and form the so-called LOD cloud. Indeed, interoperability of knowledge bases remains one of the main objectives of the Semantic Web, and this principle resides mostly in the use of Linked Data-related best practices, such as common standards, common technologies, and common terms to describe and model data in RDF graphs [Atemezing14].

More concretely, linking attacks work by using common vocabularies to unify data modeling, usually sharing IRIs used as subjects, properties or objects. They can also work using ontologies, allowing the use of meta-triples expressing knowledge such as "*entities* `tcl:user365` *and* `ratp:user812` *are the same*", or "*predicates* `ratp:subscription` *and* `tcl:membershipType` *are equivalent*", among many other possibilities.

By nature, linking datasets of various origins is a well-known way of exposing privacy leakages in the database world, usually by leveraging (unknown beforehand) *quasi-identifiers*: those are data fields that are not by themselves identifiers of some entities, but can become identifiers when associated with other quasi-identifiers. The most famous example was demonstrated by Latanya Sweeney [Sweeney02b], by using two public datasets of "anonymized" medical data (hospital visits, where name, addresses, and social security numbers were removed, but still containing sex, zip codes and birth dates) and a list of registered voters (containing voters' names and dates of birth). Using common data fields which acted as quasi-identifiers, she successfully linked Massachusetts' at-the-time governor to its medical record (cf. Section 2.1)

Linkage attacks are therefore one of the most important sources of leakage against which any anonymization framework must shield itself. This phenomenon, while common to every subfield of data management and every type of database, is amplified in the case of RDF graphs from the LOD cloud because of this possibility to share IRIs between graphs.

We must update and supplement our list of requirements from Section 4.1 to account for these new risks and the challenges they occasion. Yet, as mentioned in positioning our thesis on Section 2.2, we obviously still want to provide a **simple, declarative way to model constraints and anonymizaiton operations** and a **full-fledged data-independent anonymization framework**; but the last requirement from Section 4.1 has

to be adapted for this new context using more constrained privacy semantics: **safety against linkage attacks**.

5.2 Safety model

We generalize the definition of a *safe anonymization* introduced in [Grau19] as follows: an RDF graph is safely anonymized if it does not disclose any new query result to a set of privacy queries when it is joined with any external RDF graph *even if the latter does not satisfy the privacy policy*. Additionally, we define a notion of *data-independent* safety for a sequence of anonymization operations independently of *any* RDF graph, which is also novel with respect to [Grau19].

5.2.1 Safety in the context of the LOD cloud

Let G be an RDF graph, let O be a sequence of update queries called anonymization operations and let $\mathcal{P}$ be a privacy policy, following definitions from Section 4.2.1, that is a set of queries (called privacy queries) that will be used to model data that should be hidden post-anonymization. We call $(G, O, \mathcal{P})$ an *anonymization instance*.

An anonymization instance $(G, O, \mathcal{P})$ is *safe* when the evaluation of privacy queries from $\mathcal{P}$ over the anonymized graph $O(G)$ cannot answer results other than blank nodes. Provided an additional graph G', any result without blank nodes of a query $P \in \mathcal{P}$ executed on $O(G) \cup G'$ can be obtained from G' only. Safety is formally defined in the following Definition 17.

Definition 17 (Safe anonymization instance)**.** *An anonymization instance* $(G, O, \mathcal{P})$ *is* safe if and only if *for every RDF graph* G'*, for every* $P \in \mathcal{P}$ *and for every tuple of constants* $\bar{c}$*, if* $\bar{c} \in \mathsf{Ans}(P, O(G) \cup G')$ *then* $\bar{c} \in \mathsf{Ans}(P, G')$.

This notion of *safety* property is **stronger than the *privacy* property used in Chapter 4**, which requires that for every privacy query P, $\mathsf{Ans}(P, O(G))$ does not contain any tuple made only of constants. Privacy can thus be seen as a specific case of safety where the external RDF graph G' is empty. This also means that we can no longer easily satisfy privacy *and* utility policies at the same time, as these new requirements would requirement too much destruction in the graph to be fulfilled. We therefore decide on not using utility policies for this new solution.

Just like in the previous contribution and in contrast with [Grau19], the safety problem that we consider is data-independent and is a construction problem. Given a set of privacy queries, the goal is to build a sequence of anonymization operations guaranteed to produce a safe anonymization when applied to any RDF graph, as defined below.

Definition 18 (Safe sequence of anonymization operations)**.** *Let O be a sequence of anonymization operations, let $\mathcal{P}$ be a set of privacy queries, O is safe for $\mathcal{P}$ if and only if $(G, O, \mathcal{P})$ is safe* for every RDF graph G.

Problem 3. *The data-independent* SAFETY *problem.*

Input : *$\mathcal{P}$ a set of privacy queries*

Output: *A sequence O of update operations such that O is safe for $\mathcal{P}$.*

Our approach to solve Problem 3 is to favor *whenever possible* update operations that replace IRIs and literals by blank nodes over update operations that delete triples. We exploit the standard semantics of blank nodes that interprets them as existential variables in the scope of local graphs. As a consequence, two blank nodes appearing in two distinct RDF graphs cannot be equated.

Before describing this new solution in detail in Section 5.3, we point out the issues and requirements to solve Problem 3 by means of examples in Section 5.2.2.

5.2.2 Illustrative examples

In this section, we consider the privacy query P stating that IRIs of journeys performed by users holding a subscription reserved for disabled people should not be disclosed.

```
SELECT ?x
WHERE {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
```

Let the RDF graph to anonymize be G that is made of the following triples describing the journey of an user:

```
tcl:u1  datex:subscription          tcl:s1 .
tcl:v1  tcl:user                    tcl:u1 .
tcl:v2  tcl:user                    tcl:u1 .
tcl:s1  datex:subscriptionReference "Disabled" .
```

We first show that deleting triples may guarantee privacy but not safety. Example 5.1 shows that the problem for safety comes from a possible join between an internal and an external constant (`tcl:s1` in our example).

This can be avoided by replacing some critical constants by blank nodes. We survey now different strategies to replace some constants by blank nodes in order to enforce safety. Examples 5.2 and 5.3 illustrate the strategy consisting of replacing only one constant per triple by a blank node so as to break chains in the RDF graph likely to enable mappings from join terms in the privacy query. As shown by the example, this strategy does not guarantee safety for all the anonymization instances. Then, Examples 5.4

Example 5.1

Let O_1 be the update query that deletes all the triples instances of the `tcl:user` property:

```
1   DELETE  { ?x   tcl:user   ?y }
2   WHERE   { ?x   tcl:user   ?y }
```

The resulting anonymized RDF graph $O_1(G)$ is made of the following triples:

```
tcl:u1  datex:subscription          tcl:s1 .
tcl:s1  datex:subscriptionReference "Disabled" .
```

O_1 preserves privacy (since the evaluation of the privacy query P against $O_1(G)$ returns no answer). However, O_1 is not safe since the union of $O_1(G)$ with an external RDF graph G' containing the triple $(\texttt{tcl:v1}, \texttt{tcl:user}, \texttt{tcl:u1})$ will provide `tcl:v1` as an answer (which is not an answer of P against G' alone).

and 5.5 show that in some cases, it is mandatory to replace all the constants in some critical triples by blank nodes in order to guarantee safety. While the latter might seem like a very destructive operation, it still preserves some utility, such as the possibility of evaluating joins and counting queries.

All these examples show the necessity of studying the different strategies to produce safe anonymization operations according to some desired utility, which we address in the remainder of the chapter.

5.3 Safe anonymization of an RDF graph

In this section, we provide an algorithm that computes a solution to the SAFETY problem. To this end, we first prove a *sufficient condition* (Theorem 7) guaranteeing that an anonymization instance is safe, then we define an algorithm based on this condition. We have to extend the definition of a mapping, which is now allowed *to map constants to blank nodes*.

Definition 19 (Extended mapping)**.** *An* extended mapping μ *is a function* $\mathbf{V} \cup \mathbf{I} \cup \mathbf{L} \to \mathbf{T}$. *For a triple* $\tau = (s, p, o)$ *we write* $\mu(\tau)$ *for* $(\mu(s), \mu(p), \mu(o))$.

Theorem 7 is progressively built on two conditions that must be satisfied by all the connected components of the privacy queries. The intuition of condition (i) is that if all *critical terms* are mapped to blank nodes in the anonymized graph, it is impossible to graft external pieces of information to the anonymized graph as they cannot have common blank nodes. Condition (ii) deals with boolean connected components with no result variable.

Theorem 7 (Safe anonymization)**.** *An anonymization instance* $(G, O, \mathcal{P})$ *is* ***safe*** *if the following conditions hold for every connected component* GP_c *of all privacy queries* $P \in \mathcal{P}$*:*

Example 5.2

Let O_2 be the following update query:

```
DELETE {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
INSERT {
    _:b1    tcl:user                    ?y .
    _:b2    datex:subscription          ?z .
    _:b3    datex:subscriptionReference "Disabled" .
}
WHERE  {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
```

Applying the anonymization operation O_2 to G results in the anonymized RDF graph $O_2(G)$ made of the following triples:

```
_:b1    tcl:user                    tcl:u1 .
_:b2    datex:subscription          tcl:s1 .
_:b3    datex:subscriptionReference "Disabled" .
```

$O_2(G)$ is a safe anonymization. The reason is that it is impossible to force the mapping from the query path {`?y datex:subscription ?z. ?z datex:subscriptionReference "Disabled"`} to $O_2(G)$ because of the distinct blank nodes. Then the only way to find a mapping is to have a corresponding path in G'. But, the union with $O_2(G)$ will just produce `_:b1` as answer to the privacy query, which is not a constant.

(i) *for every critical term x of GP_c, for every triple $\tau \in GP_c$ where x appears, for each extended mapping μ s.t. $\mu(\tau) \in O(G)$, $\mu(x) \in \mathbf{B}$ holds;*

(ii) *if GP_c does not contain any result variable, then there exists a triple pattern of GP_c without any image in $O(G)$ by an extended mapping.*

The proof of Theorem 7 is based on two separate lemmas.

The following lemma on connected components will be used later. We show by induction on the length of chains that subgraphs forming a partition of a connected component must be connected together through a join term.

Lemma 4. *Let GP_1 and GP_2 be a partition of a same connected component GP, then there exists a triple $t_1 \in GP_1$ and a triple $t_2 \in GP_2$ with a (join) term in common.*

Example 5.3

Now, let O_3 be the following update query:

```
DELETE {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
INSERT {
    _:b1    tcl:user                    ?y .
    ?y      datex:subscription          _:b2 .
    _:b2    datex:subscriptionReference "Disabled" .
}
WHERE  {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
```

Like O_2, O_3 replaces only one constant per triple by a blank node, as shown in the result $O_3(G)$:

```
_:b1    tcl:user                    tcl:u1 .
tcl:u1  datex:subscription          _:b2 .
_:b2    datex:subscriptionReference "Disabled" .
```

In contrast with $O_2(G)$, $O_3(G)$ is not safe (while still preserving privacy), as its union with an external RDF graph G' containing the triple $(\texttt{tcl:v1}, \texttt{tcl:user}, \texttt{tcl:u1})$ would return the constant `tcl:v1` as an answer to the query P.

Proof. Since GP_1 and GP_2 are not empty, there exists a triple $t_1 \in GP_1$ and a triple $t_2 \in GP_2$. Assume that t_1 an t_2 do not have a constant or variable in common in subject or object position. We proceed by contradiction.

Let c_1 be a constant or a variable in t_1 and let c_2 be a constant or variable of t_2. Since c_1 and c_2 appear in the same connected component GP, there exists a path of length n from c_1 to c_2, i.e., there exist chains $p^1, \ldots, p^n$ and $c^1, \ldots, c^{n+1}$ such that $c_1 = c^1$, $c_2 = c^{n+1}$ and for every $i \in [1, n]$ either $(c^i, p^i, c^{i+1}) \in GP$ or $(c^{i+1}, p^i, c^i) \in GP$.

Let k the greatest index such that *all* triples (c^i, p^i, c^{i+1}) or (c^{i+1}, p^i, c^i) with $i \leq k$ are in GP_1. If $k = n$, then $c^{k+1} = c_2$ and c_2 is common to two triples in GP_1 and GP_2, a contradiction. Otherwise, if $k < n$, then $(c^{k+1}, p^{k+1}, c^{k+2})$ or $(c^{k+2}, p^{k+1}, c^{k+1})$ is in GP_2 and c^{k+1} is common to two triples in GP_1 and GP_2, a contradiction again.

□

First we deal with the case where the privacy query is *not* boolean with only one connected component. We show that operators O replacing images of critical terms in

Example 5.4

Since Examples 5.2 and 5.3 show that replacing only one constant per triple may not guarantee safety, we have to consider update queries that replace all the constants in some triples by blank nodes, like the following update query O_4:

```
DELETE {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
INSERT {
    _:b1    tcl:user                    _:b2 .
    _:b2    datex:subscription          _:b3 .
    _:b3    datex:subscriptionReference "Disabled" .
}
WHERE  {
    ?x  tcl:user                    ?y .
    ?y  datex:subscription          ?z .
    ?z  datex:subscriptionReference "Disabled" .
}
```

The result RDF graph $O_4(G)$ is made of the following triples:

```
_:b1    tcl:user                    _:b2.
_:b2    datex:subscription          _:b3.
_:b3    datex:subscriptionReference "Disabled".
```

Similarly to $O_2(G)$ in Example 5.2, it can be shown that $O_4(G)$ is safe. In addition to this, since all the occurrences of the join variables are replaced by same blank nodes, the result of the counting query $\mathsf{Count}(P)$ is preserved, i.e., it returns the same value as when it is evaluated on the original RDF graph G.

G (i.e., result variables, join variables and join constants in subject or object position) with blank nodes guarantees that $(G, O, \{P\})$ is *safe* for any graph G.

Lemma 5. *Let $(G, O, \{P\})$ be an anonymization instance where $P = \langle \bar{x}, GP \rangle$ is a query made of a unique connected component and at least one distinguished variable. If for all critical term x of GP, for all triple $\tau \in GP$ where x appears, for each extended mapping μ s.t. $\mu(\tau) \in O(G)$, we have $\mu(x) \in \mathbf{B}$, then $(G, O, \{P\})$* is safe.

Proof. The main idea is to prevent linking $O(G)$ and G' together by using blank nodes. The key argument used here is that *blank nodes from different graphs are distinct*, in conformity with the logical interpretation of blank nodes as existential variables in the scope of a single RDF graph.

We proceed by contradiction. Suppose that $(G, O, \{P\})$ is *not* safe, thus there exists a graph G', a tuple of constants $\bar{c} \in \mathsf{Ans}(P, O(G) \cup G')$ with $\bar{c} \notin \mathsf{Ans}(P, G')$. Therefore by the definition of Ans, there exists a mapping μ such that $\mu(\bar{x}) = \bar{c}$ and $\mu(GP) \subseteq O(G) \cup$

Example 5.5

Following Example 5.4, we can show that safety does occur if we apply the following update query O_5 which breaks all join terms:

```
DELETE {
    ?x  tcl:user                    ?y .
    ?y2 datex:subscription          ?z .
    ?z2 datex:subscriptionReference "Disabled" .
}
INSERT {
    _:b1    tcl:user                    _:b2 .
    _:b3    datex:subscription          _:b4 .
    _:b5    datex:subscriptionReference "Disabled" .
}
WHERE  {
    ?x  tcl:user                    ?y .
    ?y2 datex:subscription          ?z .
    ?z2 datex:subscriptionReference "Disabled" .
}
```

The result RDF graph $O_5(G)$ is made of the following triples:

```
_:b1    tcl:user                    _:b2.
_:b3    datex:subscription          _:b4.
_:b5    datex:subscriptionReference "Disabled".
```

In fact, $O_5(G)$ is more general than $O_4(G)$, i.e. $O_4(G) \models O_5(G)$.

G' but $\mu(GP) \not\subseteq G'$. Let GP_1 be the largest subgraph of GP such that $\mu(GP_1) \subseteq O(G)$. Let GP_2 be its complement, $GP_2 = GP \setminus GP_1$.

If $GP_1 = \emptyset$, we are done, because this contradicts the assumption $\mu(GP) \not\subseteq G'$. Thus, there is some $\tau \in GP_1$. Now, if $GP_2 = \emptyset$, then all variables $x \in \bar{x}$ appear in GP_1, by hypothesis on $O(G)$, $\mu(x) \in \mathbf{B}$, but $\mu(x) \in \bar{c}$ forbids $\mu(x)$ to be a blank for all x, as $\bar{x}$ is not empty by hypothesis, we obtain a contradiction. Thus there is some $\tau' \in GP_2$. So both GP_1 and GP_2 are non empty and thus form a partition of GP.

By Lemma 4, there exist $\tau_1 \in GP_1$ and $\tau_2 \in GP_2$ with a join variable or a join constant in common, let x be this join variable or constant. If x is a constant, there exists a critical term x in a triple τ_1, and an extended mapping μ_{ext} such that $\mu_{ext}(x) \notin \mathbf{B}$, which is a contradiction. If x is a variable, then $\mu(x)$ is a blank node. Blank nodes are local, so the blank nodes of $O(G)$ are disjoint from those of G'. This contradicts that $\mu(\tau_2) \in G'$ with $\mu(x)$ being a blank node of G'.

□

Now we deal with the corner case where the privacy query has no result variable. Indeed, in that particular case, replacing all the images of critical variables and con-

stants by blank nodes does not even guarantee privacy because such a transformation preserves the existence of images of boolean queries, as shown in Exmaple 5.6.

Example 5.6

Let a privacy query Q, with $\mathsf{body}(Q) = \{(?x, p, o), (o, q, ?y)\}$ and no distinguished variable.
Assume the following graph G and its two following anonymized graphs $O(G)$ and $O'(G)$.

```
G        = { ex:x ex:p ex:o.  ex:o ex:q ex:y. }
O(G)     = { ex:x ex:p _:b.   _:b  ex:q ex:y. }
O'(G)    = { ex:x ex:p _:b. }
```

We have $\mathsf{Ans}(Q, O(G)) = \{\emptyset\}$, therefore privacy is not satisfied. But $O'(G)$ is safe with regard to Q.
Some triples must therefore be deleted, in addition to the replacement the images of join terms by blank nodes (i.e. the conditions of the lemmas must hold).

Lemma 6. *Let $(G, O, \{P\})$ be an anonymization instance where $P = \langle \emptyset, GP \rangle$ is a boolean query with only one connected component. Assume that $(G, O, \{P\})$ already satisfies the conditions of Lemma 5. If there exists some triple pattern $\tau \in GP$ with no image in $O(G)$ by any anonymization mapping, then $(G, O, \{P\})$* is safe.

Proof. When the query is boolean, the safety condition amounts to check that if there exists an image of GP in $O(G) \cup G'$ by a mapping μ, then $\mu(GP)$ must be included in G' alone. Basically, we have to extend the proof of Lemma 5 to cover the case where $\bar{x} = \emptyset$.

We again proceed by contradiction. Suppose that $(G, O, \{P\})$ is *not* safe, thus there exists a graph G', a tuple of constants $\bar{c} \in \mathsf{Ans}(P, O(G) \cup G')$ with $\bar{c} \notin \mathsf{Ans}(P, G')$. Therefore by the definition of Ans, there exists a mapping μ such that $\mu(\bar{x}) = \bar{c}$ and $\mu(GP) \subseteq O(G) \cup G'$ but $\mu(GP) \not\subseteq G'$. Let GP_1 be the largest subgraph of GP such that $\mu(GP_1) \subseteq O(G)$. Let GP_2 be its complement, $GP_2 = GP \setminus GP_1$.

If $GP_1 = \emptyset$, we are done, because this contradicts the assumption $\mu(GP) \not\subseteq G'$. Thus, there is some $\tau \in GP_1$.

With the new condition added in the current, there exists some triple pattern $\tau'' \in GP_1$ with no image in $O(G)$, thus $\mu(GP_1) \not\subseteq O(G)$ and GP_2 cannot be empty. The rest of the proof is similar.

□

The following lemma states that the safety a query can be ensured by the safety of its connected components.

Lemma 7. *Let $A_1 = (G, O, \{Q_1\})$ and $A_2 = (G', O, \{Q_2\})$ be two safe anonymization instances where $Q_1 = \langle \bar{x}_1, GP_1 \rangle$ and $Q_2 = \langle \bar{x}_2, GP_2 \rangle$ are two queries made of exactly one connected component without common terms or variables. The anonymization instance $(G, O, \{Q\})$ with $Q = \langle \bar{x}_1 \cup \bar{x}_2, GP_1 \cup GP_2 \rangle$ is safe as well.*

Proof. Assume that there is some $\bar{c} \in \mathsf{Ans}(Q, O(G) \cup G')$ for an arbitrary G' where $\bar{c} = \bar{c_1} \cup \bar{c_2}$ where $\bar{c_1}$ (resp. $\bar{c_2}$) is the restriction of $\bar{c}$ to the image of $\bar{x}_1$ (resp. $\bar{x}_2$).

As A_1 (resp. A_2) is safe, any tuple of constants $\bar{c}_1$ (resp. $\bar{c}_2$) found when evaluating Q_1 (resp. Q_2) on $O(G) \cup G'$ already exists in G'. Therefore there exists a mapping μ_1 such that $\mu_1(\bar{x}_1) = \bar{c}_1$ and $\mu_1(GP_1) \subseteq G'$ (resp. μ_2 s.t. $\mu_2(\bar{x}_2) = \bar{c}_2$ and $\mu_2(GP_2) \subseteq G'$). Since GP_1 and GP_2 are disjoint, they constitute two different connected components of $GP_1 \cup GP_2$. We can create a mapping μ' such that $\mu'(GP_1 \cup GP_2) \subseteq G'$ using μ_1 for the variables of Q_1 and μ_2 for the variables of Q_2. Finally, $\bar{c} = \bar{c}_1 \cup \bar{c}_2 = \mu_1(\bar{x}_1) \cup \mu_2(\bar{x}_2) = \mu'(\bar{x}_1) \cup \mu'(\bar{x}_2)$, therefore, $\bar{c}$ is an answer of Q over G which concludes the proof. □

We can now assemble all the pieces from previously defined lemmas to obtain the proof of Theorem 7.

Proof. To show that $(G, O, \mathcal{P})$ with $\mathcal{P} = \{P_1, \ldots, P_n\}$ is safe, we have to show that each $(G, O, \{P_i\})$ is safe. The conditions of Theorem 7 are exactly those of Lemma 5 and Lemma 6 for each connected component P_i^j of P_i, thus each $(G, O, \{P_i^j\})$ is safe by itself.

Consider $\bar{c} \in \mathsf{Ans}(P_i, O(G) \cup G')$. Let $GP_i = \mathsf{body}(P_i)$ and let $GP_i^j = \mathsf{body}(P_i^j)$. By the definition of Ans, there exists a mapping μ such that $\mu(\bar{x}) = \bar{c}$ and $\mu(GP_i) \subseteq O(G) \cup G'$. By definition $GP_i^j \subseteq GP^i$ holds, thus by the monotonicity of queries (Lemma 1), $\mu(GP_i^j) \subseteq O(G) \cup G'$ holds as well for each connected component GP_i^j. Since each P_i^j is safe, there exists μ^j such that $\mu^j(GP_i^j) \subseteq G'$ for each GP_i^j. As the subgraphs GP_i^j are connected components, they cannot share terms. By induction on the number of connected components, using repeatedly Lemma 7, we can construct an anonymization mapping μ' such that $\mu'(GP_i^j) \subseteq G'$ for all GP_i^j, but $GP_i = \bigcup GP_i^j$, so $\mu'(GP_i) \subseteq G'$.

We are now left to prove that $\mu'(\bar{x}) = \bar{c}$. For each variable $x \in \bar{x}$, there is a unique connected component $GP_i^{j_x}$ where x appears and $\bar{c}$ is nothing else but $\bigcup_{x \in \bar{x}} \mu^{j_x}(x)$.

□

We are now able to design an anonymization algorithm that solves the SAFETY problem. Algorithm 3 computes a sequence[1] of operations O for a privacy policy $\mathcal{P}$ such that O is safe for $\mathcal{P}$. Operations are computed for each connected component of each privacy query from $\mathcal{P}$. The crux is to turn conditions (i) and (ii) into update queries (Definition 12).

[1]The $+$ operator denotes the concatenation of sequences.

The starting point of Algorithm 3 is to compute join variables and constants (Lines 5 to 8) then to compute critical terms by adding the result variables (Lines 9 to 10). Note that critical terms *do not* include variables in predicate position, except when these variables are also result variables of the considered query. The update queries that replace the images of critical terms by blank nodes are built from Line 14 to Line 17. The subtle point is that, in order to guarantee that condition (i) of Theorem 7 is satisfied on any updated RDF graph, as many update queries as subsets of the connected component GP_c need to be constructed. Considering these subsets in decreasing order of cardinality (Line 11) and using the `isNotBlank`$(\bar{x}')$ construct introduced in Definition 12 (Line 17) guarantees that all the images of a critical term in a given RDF graph will be replaced only once and by the same blank node. Finally, if the connected component under scrutiny is boolean, one of its triple is deleted (Line 20), according to condition (ii). This triple is chosen in a non-deterministic way, i.e. any triple can work and any decision process could be used.

Continuing Examples 5.4 and 5.5 from Section 5.2.2, operation O_4 is the first one generated at Line 17. When applied to Example 3.10, Algorithm 3 will sequentially generate anonymization operations starting from those replacing the images of all the variables by blank nodes (since all its variables are critical), followed by those deleting all the triples (possibly modified by the preceding operations) corresponding to one of the triple patterns (`?v a tcl:Validation` or `?v tcl:validationDatetime ?d`) in the boolean connected component.

Note that some anonymizations may create an RDF graph where some properties have been replaced by blank nodes (in the case where a result variable was appearing in as a predicate in the body of a policy query). In this case, the output is not a standard RDF graph anymore, but a *generalized* RDF graph[2].

Theorem 8 states the soundness and computational complexity of Algorithm 3.

Theorem 8. *Let O = `find-safe-ops`$(\mathcal{P})$ be the sequence of anonymization operations returned by Algorithm 3 applied to the set $\mathcal{P}$ of privacy queries: O is safe for $\mathcal{P}$ and $G \models O(G)$ for any graph G. The worst-case computational complexity of Algorithm 3 is exponential in the size of $\mathcal{P}$.*

Proof. The main idea is to show that the conditions of Theorem 7 are satisfied when O is executed on G. Let $\mathcal{P}$ be a privacy policy with queries $P_i = \langle \bar{x}_i, GP_i \rangle$, let O = `find-safe-ops` $(\mathcal{P})$ and let G be an *arbitrary* RDF graph. For each connected component $GP_c \subseteq GP_i$ of each privacy query P_i, Algorithm 3 generates

- a sequence of operations at Line 17, mimicking condition (i) of Theorem 7;
- a non-deterministic delete operation at Line 20, mimicking condition (ii).

[2]As specified in Section 7 of the RDF 1.1 Specification: https://www.w3.org/TR/rdf11-concepts/#section-generalized-rdf.

Input : a privacy policy $\mathcal{P}$ of queries $P_i = \langle \bar{x}_i, GP_i \rangle$
Output: a sequence of operations O which is safe for $\mathcal{P}$

1 **function** `find-safe-ops(`$\mathcal{P}$`)`**:**
2 Let $O = \langle\,\rangle$;
3 **for** $P_i \in \mathcal{P}$ **do**
4 **forall** *connected components* $GP_c \subseteq GP_i$ **do**
5 Let $I := [\,]$;
6 **forall** $(s, p, o) \in GP_c$ **do**
7 **if** $s \in \mathbf{V} \vee s \in \mathbf{I}$ **then** $I[s] = I[s] + 1$;
8 **if** $o \in \mathbf{V} \vee o \in \mathbf{I} \vee o \in \mathbf{L}$ **then** $I[o] = I[o] + 1$;
9 Let $\bar{x}_c := \{v \mid v \in \bar{x}_i \wedge \exists \tau \in GP_c \text{ s.t. } v \in \tau\}$;
10 Let $T_{crit} := \{t \mid I[t] > 1\} \cup \bar{x}_c$;
11 Let $SGP_c = \{X \mid X \subseteq GP_c \wedge X \neq \emptyset \wedge X \text{ is connected}\}$ ordered by decreasing size;
12 **forall** $X \in SGP_c$ **do**
13 Let $X' := X$ and $\bar{x}' = \{t \mid t \in T_{crit} \wedge \exists \tau \in X \text{ s.t. } t \in \tau\}$;
14 **forall** $x \in \bar{x}'$ **do**
15 Let $b \in \mathbf{B}$ be a fresh blank node;
16 $X' := X'[x \leftarrow b]$;
17 $O := O + \langle$`DELETE` X `INSERT` X' `WHERE` X `isNotBlank(`$\bar{x}'$`)`$\rangle$
18 **if** $\bar{x}_c = \emptyset$ **then**
19 Let $\tau \in GP_c$ `// non-deterministic choice`
20 $O := O + \langle$`DELETE` τ `WHERE` $GP_c\rangle$
21 **return** O;

Algorithm 3: Find update operations to ensure safety

Conditions (i) and (ii) encapsulated in Theorem 7 correspond respectively to Lemmas 5 and 6. If we show that for each connected component GP_c Algorithm 3 generates a sequence of operation satisfying conditions (i) and (ii), the safety of O for $\mathcal{P}$ will follow.

First of all, in Algorithm 3, remark that by construction T_{crit} is the set of critical terms. Also, note that the order of operations *is* significant. Indeed, Lemmas 5 and 6 are cumulative: the second one relies on the first one. For GP_c a connected component of the privacy query $P_i = \langle \bar{x}_i, GP_i \rangle$, let $\bar{x}_c$ be the subset of variables from $\bar{x}_i$ that appear in GP_c. We consider both conditions separately.

Condition (i)

We have $\bar{x}_c \neq \emptyset$. The generated operations $O = \langle O_1, \ldots, O_p \rangle$ are only those from Line 17, the operations at Line 20 are not triggered. Let x be a join term (variable or constant) and let $GP_x \subseteq GP_c$ the (unique) subset of GP_c where x appears. Let GP'_x be

the largest subset of GP_x with an image in $O(G)$ and μ the mapping s.t. $\mu(GP'_x) \subseteq O(G)$. We have to show that $\mu(x) \in \mathbf{B}$. Let $m = |GP'_x|$ be the cardinality of GP'_x. There exists one operation $O_k \in O$ of the form $O_k =$ `DELETE` X_k `INSERT` X'_k `WHERE` X_k `isNotBlank`(x) which is obtained when $k = m$, that is when $X_k = GP'_x$ in the loop at Line 12. If there is no such O_k, the condition is vacuously satisfied as $GP'_x = \emptyset$. The operation O_k is exactly the replacement of every $\mu(x)$ by the fresh blank b of Line 15 thus $\mu(x) \in \mathbf{B}$ and the condition (i) is satisfied, hence the safety of O for P_i by Lemma 5.

Condition (ii)

GP_c is now a boolean query with no result variable. The operation `DELETE` τ `WHERE` GP_c at Line 20 deletes all occurrences of a triple τ (chosen non-deterministicly) such that $\mu(\tau) \in G$ thus, the condition (ii) required by Lemma 6 is satisfied and GP_c is safe.

We now show that the update operations (which are not mere deletions) guarantee $G \models O(G)$ as well. The sufficient condition to ensure safety is to replace critical IRIs and variables by blank nodes: no `DELETE` operation is performed. The key point to prove Theorem 9 is thus that `INSERT` operations produced by Algorithm 3 are in fact a one-to-many renaming of IRIs to blank nodes. Indeed in the update query produced ate Line 17, each critical term is replaced by a *fresh* blank node, but no two different IRIs are mapped to the same one since there is no reuse of previously generated blank nodes. Therefore, Algorithm 3 constructs a one-to-many μ renaming of join IRIs to blank nodes with inverse μ_1^{-1} with $\mu_1^{-1}(O(G)) = G$ which implies that $G \models O(G)$ by Theorem 1.

For the worst-case complexity, the size considered is defined here as $\text{size}(\mathcal{P}) = \Sigma_{P_i \in \mathcal{P}} |GP_i|$. The loop at Line 12 is executed exactly $2^n - 1$ times where n is the size of the largest connected component of G_i. Thus considering for instance $\mathcal{P} = \{P\}$ where P's body is made of a single component of cardinality n, the loop generates an exponential number of operations at Line 17.

□

Note that the `isNotBlank`(x) condition ensures that O_k is applied only once in O if it exists, indeed if $\mu(x) \in \mathbf{B}$ for some k, no other operation O_l with $l \geq k$ is applied. Hence, this condition is not needed to guarantee safety but ensures that the occurrences of the considered join term are replaced only once. Thus, we ensure that they use the same blank node and that some are not replaced further in the algorithm.

Theorem 9 establishes that the anonymization operations computed by Algorithm 3 preserve some information on $\mathsf{Count}(P)$ for any privacy query P with no boolean connected component.

Theorem 9. *Let $O =$ `find-safe-ops`$(\{P\})$ be the output of Algorithm 3 applied to a privacy query P with no boolean connected component. For every RDF graph G, $O(G)$ satisfies $\mathsf{Ans}(\mathsf{Count}(P), O(G)) \geq \mathsf{Ans}(\mathsf{Count}(P), G)$.*

Proof. The conditions here are the same as those of Lemma 5. We use the argument than the one used to show that $G \models O(G)$ in the proof of Theorem 8. The sufficient condition to ensure safety is thus to replace critical IRIs and variables by blank nodes: no `DELETE` operation is performed. The key point to prove Theorem 9 is that `INSERT` operations produced by Algorithm 3 are in fact a one-to-many renaming of IRIs to blank nodes. Indeed in the update query produced ate Line 17, each critical term is replaced by a *fresh* blank node, but no two different IRIs are mapped to the same one since there is no reuse of previously generated blank nodes.

□

Since Algorithm 3 is data-independent and therefore run once and for all without any performance penalty at runtime, its exponential worst-case complexity (due to the powerset SGP_c computed on Line 11) is not necessarily an important limitation in practice, as will be demonstrated in our experimental study in the next chapter.

Algorithm 4 is a polynomial approximation of Algorithm 3 obtained as follows: instead of considering all possible subsets of triple patterns of SG_c (Line 12), we simply construct update queries that replace, in each triple pattern $\tau \in GP_c$, every critical term with a fresh blank node. As a result, there does not exist anymore any equality between images of join variables, literals or IRIs (while in Algorithm 3 all occurrences of each critical term were replaced by the same blank node). For instance, Algorithm 4 generates a sequence of three update queries, one for each triple, equivalent to O_5 in Example 5.5 from Section 5.2.2.

Theorem 10 states that Algorithm 4 is sound and leads to anonymizations that are more general than those produced by Algorithm 3.

Theorem 10. *The worst-case computational complexity of Algorithm 4 is polynomial in the size of $\mathcal{P}$. Let O and O' be the result of applying respectively Algorithm 3 and Algorithm 4 (with the same non-deterministic choices) to a set $\mathcal{P}$ of privacy queries: for any RDF graph G,* $(G, O, \mathcal{P})$ is safe *and* $O(G) \models O'(G)$.

Proof. The proof of the safety of Algorithm 4 is similar to Theorem 8, the only difference lies at *Condition (i)*, the operations for *Condition (ii)* being the same. The actual difference between Algorithm 3 and Algorithm 4 is that the latter generates a fresh blank node *triple by triple* breaking multiple occurrences of join variables instead of preserving them. Indeed, for each join variable x of the connected component GP_c under scrutiny, the operation generated at Line 17, call it O_c, replaces it with a fresh blank node. There is exactly one such O_c for each GP_c, thus the output of Algorithm 4 is polynomial: the source of the exponential complexity of Algorithm 3 has been eliminated. Indeed, the loop at Line 12 do not browse the subsets $SGP_c \subseteq GP_c$ anymore but only its elements $\tau \in GP_c$.

Input : a privacy policy $\mathcal{P}$ of queries $P_i = \langle \bar{x}_i, GP_i \rangle$
Output: a sequence of operations O safe modulo sameAs for $\mathcal{P}$

1 **function** `find-safe-ops-sameas(`$\mathcal{P}$`)`**:**
2 Let $O = \langle\,\rangle$;
3 **for** $P_i \in \mathcal{P}$ **do**
4 **forall** *connected components* $GP_c \subseteq GP_i$ **do**
 `/* [...Lines 3 to 10 identical to find-safe-ops...] */`
10 Let $\bar{x}' := \{v \mid v \in \bar{x}_i \wedge \exists \tau \in GP_c \text{ s.t. } v \in \tau\}$;
11 Let $T_{crit} := \{t \mid I[t] > 1\} \cup \bar{x}'$;
12 **forall** $\tau \in GP_c$ **do**
13 **forall** $x \in T_{crit}$ **do**
 `/* Every occurrence of a critical x is replaced by a different blank node */`
14 $G' := \{\tau[x \leftarrow [\,]]\}$;
15 Let $v \in \mathbf{V}$ a fresh variable;
16 $G'' := \{\tau[x \leftarrow v]\}$;
17 $O := O + \langle$`DELETE` G'' `INSERT` G' `WHERE` G'' `isNotBlank(`$\bar{x}'$`)`$\rangle$;
 `/* [...End of algorithm identical to find-safe-ops...] */`
18 **return** O;

Algorithm 4: Find update operations to ensure safety modulo sameAs

Recall that, by Theorem 1, for two graphs with blank nodes $A \models B$ amounts to show that there exists a mapping μ of blank nodes to terms such that $\mu(B) \subseteq A$. The point is thus to exhibit such a mapping.

First of all, if $G' \subseteq G$ then clearly $G \models G'$ (pick μ as the identity), so the property holds for all delete operations in O which are common to both Algorithm 3 and Algorithm 4. These operations being the very same ones when the same non-deterministic choices are made by the two algorithms: for each component, we suppose that the triple τ picked at Line 19 is the same.

We are left to prove that $O(G) \models O'(G)$. For Algorithm 4, we observe that the renaming is not a function anymore because the same literal that occurs multiple times is mapped to *different blank* nodes. However, by construction as each occurrence of a critical τ is replaced by a fresh blank node, there exists some μ_2^{-1} from blank nodes to IRIs such that $\mu_2^{-1}(O'(G)) = G$. So now, construct μ_3^{-1} such that $\mathsf{dom}(\mu_3^{-1}) = \mathsf{dom}(\mu_2^{-1})$ defined by $\mu_3^{-1} = \mu_1 \circ \mu_2^{-1}$ which maps each blank node generated by O' to the one obtained by O. The mapping μ_3^{-1} is such that $\mu_3^{-1}(O'(G)) = \mu_1(\mu_2^{-1}(O'(G))) = \mu_1(G) = O(G)$ thus $O(G) \models O'(G)$.

□

Note that contrarily to Algorithm 3, we do not provide utility guarantees such as an extension of Theorem 9 for Algorithm 4.

5.4 Safe anonymization robust to sameAs links

One of the fundamental assets of LOD is the possibility to assert that two resources are the same by stating `owl:sameAs` triples (shortened to `:sameAs` later), also known as *entity linking*. We only consider sameAs links causing *entity alignment*, meaning that we do not consider `:sameAs` properties between properties themselves, and we interpret `:sameAs` triples (called `:sameAs` links) as equality between constants (including blank nodes) that are in subject or object position. With this interpretation, `:sameAs` links can also be inferred by a logical reasoning on additional knowledge known on some properties (e.g. that a property is functional). In this section, we study the impact of both explicit and inferred `:sameAs` links on safety.

We extend Definition 10 to the semantics of query answering in presence of a set sameAs of `:sameAs` links. We define $\mathsf{closure}(\mathsf{sameAs})$ as the *transitive, reflexive and symmetric closure* of the sameAs predicate, i.e, the set of all sameAs links that can be derived by transitivity and symmetry. This set can be computed in polynomial time.

Definition 20 (Answer of a query modulo sameAs)**.** *Let Q be a conjunctive query, G an RDF graph and* sameAs *a set of* `:sameAs` *links. A tuple $\bar{a}$ is an answer to Q over G modulo* sameAs *if and only if there exists $(b_0, \texttt{:sameAs}, b'_0), \ldots, (b_k, \texttt{:sameAs}, b'_k)$ in* $\mathsf{closure}(\mathsf{sameAs})$ *s.t. $\bar{a} \in \mathsf{Ans}(Q, G[b_0 \leftarrow b'_0, \ldots, b_k \leftarrow b'_k])$.*

We note $\mathsf{Ans}_{\mathsf{sameAs}}(Q, G)$ the answer set of Q over G modulo sameAs*.*

Hence, we extend Definition 17 to handle a set sameAs of `:sameAs` links.

Definition 21 (Safety modulo sameAs)**.** *An anonymization instance $(G, O, \mathcal{P})$ is* safe modulo sameAs *if and only if for every RDF graph G', for every $P \in \mathcal{P}$ and for any tuple of constants $\bar{c}$, if $\bar{c} \in \mathsf{Ans}_{\mathsf{sameAs}}(P, O(G) \cup G')$ then $\bar{c} \in \mathsf{Ans}_{\mathsf{sameAs}}(P, G')$.*

O is safe modulo sameAs *for $\mathcal{P}$ if $(G, O, \mathcal{P})$ is safe modulo* sameAs *for every RDF graph G and for every set* sameAs *of* `:sameAs` *links.*

We first study how to build anonymization operations that are robust to explicit `:sameAs` links. Then, we focus on handling the case of inferred `:sameAs` links through knowledge.

5.4.1 Explicit sameAs links

Explicit sameAs links exist when a graph contains triples using directly the `owl:sameAs` property between two entities, provided that these triples fulfilled our previous re-

quirements (the subject and object of the considered triple are not properties themselves).

Theorem 11 establishes that Algorithm 3 (and thus Algorithm 4 as well) computes safe anonymizations even in presence of a set sameAs of explicit `:sameAs` links.

Theorem 11. *Let O be the result of applying Algorithm 3 to a set $\mathcal{P}$ of privacy queries: for any set* sameAs *of explicit* `:sameAs` *links, O* ***is safe modulo*** sameAs *for $\mathcal{P}$.*

Proof. The proof is similar to the proof of Theorem 8, the only case of interest being that of condition (i), the rest is similar. Let G be the RDF graph to anonymize, G' be an external graph, and $P = \langle \bar{x}, GP \rangle$ be a privacy query. Let $\bar{c}$ be a tuple of constants such that $\bar{c} \in \mathsf{Ans}_{\mathsf{sameAs}}(P, O(G) \cup G')$.

By Definition 11 $\exists (b_0, :\ \mathsf{sameAs}, b'_0), ..., (b_n, :\ \mathsf{sameAs}, b'_n) \subseteq \mathsf{closure}(\mathsf{sameAs})$ statements, such that $\bar{c} \in \mathsf{Ans}(P, (O(G) \cup G')[b_i \leftarrow b'_i])$.

Consider the largest subset $GP_1 \subseteq GP$ such that $\mu(GP_1) \subseteq O(G)[b_i \leftarrow b'_i]$ and its complement $GP_2 = GP \setminus GP_1 \subseteq G'[b_i \leftarrow b'_i]$. With an argument similar to the proof of Lemma 5, GP_1 and GP_2 constitute a partition of GP so there must exist $\tau_1 \in GP_1$ and $\tau_2 \in GP_2$ with a join term variable or IRI call it x. Thus, there exist terms b_k and b'_k such that $\mu(x) = b'_k$ for some k with $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$. On the one hand, b_k is a term of $O(G)[b_i \leftarrow b'_i]$ because $\tau_1 \in G_1$, on the other hand b_k is a term of $G'[b_i \leftarrow b'_i]$ because $\tau_1 \in G_2$.

We have to show that $(b_k : \mathsf{sameAs}, b'_k)$ cannot be found in $\mathsf{closure}(\mathsf{sameAs})$. Recall that $\mathsf{closure}(\mathsf{sameAs})$ is the reflexive, symmetric and transitive closure of sameAs. The proof is by structural induction on the derivation of $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$, consider the last deduction rule used to conclude that $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$:

Case $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{sameAs}$**.** The operation at Line 17 of Algorithm 3 ensures that b_k is a fresh blank of $O(G)$ which cannot appear in G' or in sameAs since blank nodes are only local to the graph they appear in, a contradiction.

Case $(b_k : \mathsf{sameAs}, b_k) \in \mathsf{closure}(\mathsf{sameAs})$ **by reflexivity.** This is the nominal case of Lemma 5 with an argument similar to the previous one.

Case $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$ **by symmetry.** A contradiction again by the inductive hypothesis $(b'_k, : \mathsf{sameAs}, b_k) \in \mathsf{closure}(\mathsf{sameAs})$.

Case $(b_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$ **by transitivity.** By hypothesis there exists b''_k such that $(b_k : \mathsf{sameAs}, b''_k) \in \mathsf{closure}(\mathsf{sameAs})$ and $(b''_k : \mathsf{sameAs}, b'_k) \in \mathsf{closure}(\mathsf{sameAs})$. Here the contradiction is on $(b_k : \mathsf{sameAs}, b''_k) \in \mathsf{closure}(\mathsf{sameAs})$.

□

5.4.2 Inferred sameAs links

Now, we address two cases in which knowledge on properties may infer equalities: we call these cases *implicit* or *inferred* sameAs *links*, because they virtually amount to situations where equality between some entities is deductible rather than explicitly stated using a `:sameAs` triple.

The first case occurs in the ontology axiomatization of the OWL language[3] when some of the properties are functional or inverse functional, as in Definition 22, where we model equalities by `:sameAs` links.

Definition 22 (Functional and inverse functional RDF properties)**.** *A property p is* functional *if and only if for every $?x$, $?y_1$, $?y_2$:*

$(?x, p, ?y_1) \wedge (?x, p, ?y_2) \Rightarrow (?y_1, :sameAs, ?y_2)$.

A property p is inverse functional *if and only if for every $?x$, $?y_1$, $?y_2$:*

$(?y_1, p, ?x) \wedge (?y_2, p, ?x) \Rightarrow (?y_1, :sameAs, ?y_2)$.

For example, declaring that property `tcl:referrerOf` as inverse functional expresses the constraint that every user can be referred by at most one user. As shown in Example 5.7, exploiting this knowledge may lead to re-identifying blank nodes that have been produced by the previous anonymization algorithms.

One solution to prevent this type of attack is to add a query to the privacy policy for each functional property p and for each inverse functional property q, respectively the queries `SELECT ?x WHERE {?x p ?y.}` and `SELECT ?x WHERE {?y q ?x.}`.

By doing so, the update queries returned by our Algorithms 3 and 4 will replace each constant in subject position of a functional property by a fresh blank node, and each constant in an object position of an inverse functional property by a fresh blank node. In the previous Example 5.7, the constant `tcl:u3` in (`_:b1`, `tcl:referrerOf`, `tcl:u3`) would be replaced by a fresh blank node.

The second case that we consider may lead to inferred sameAs equalities when a property is completely known, i.e. when its closure is available in an external RDF graph. For instance, suppose that the closure of the property `tcl:registeredBy` is stored in an extension of the external RDF graph G' containing the following triples, describing users registered by staff members:

```
tcl:u1  tcl:registeredBy   tcl:s1 .
tcl:u6  tcl:registeredBy   tcl:s2 .
tcl:u7  tcl:registeredBy   tcl:s2 .
tcl:u9  tcl:registeredBy   tcl:s2 .
```

[3] See OWL 2 RDF-Based Semantics, notably section 5.13. https://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/#Semantic_Conditions

Example 5.7

Let $\mathcal{P}$ a privacy policy containing exactly one query P, defined as follows:

```
SELECT ?x
WHERE {
    ?x  tcl:registeredBy    ?y .
    ?x  tcl:referrerOf      ?z .
}
```

It describes the fact that we want to hide the IRI of users registered by a TCL staff member and referred by another user.
Let G, $O(G)$ and G' be the following RDF graphs where O is an update operation returned by Algorithm 3:

```
G  = {
    tcl:u1  tcl:registeredBy    tcl:s1 .
    tcl:u1  tcl:referrerOf      _:b1 .
    _:b1    tcl:referrerOf      tcl:u3 .
}

O(G) = {
    _:b0    tcl:registeredBy    tcl:s1 .
    _:b0    tcl:referrerOf      _:b1 .
    _:b1    tcl:referrerOf      tcl:u3 .
}

G' = {
    tcl:u1  tcl:referrerOf  tcl:u2 .
    tcl:u2  tcl:referrerOf  tcl:u3 .
}
```

From $O(G) \cup G'$ and the inverse functionality of `:referrerOf`, it can be inferred first (`tcl:u2`,`:sameAs`,`_:b1`) and second (`tcl:u1`, `:sameAs`,`_:b0`). Consequently, `_:b0` is re-identified as `tcl:u1`, which is returned as answer of P over $O(G) \cup G'$ modulo sameAs, and the anonymization operation O is not safe.

Knowing that G' is the complete extension of the `tcl:registeredBy` predicate allows to infer (`_:b0`, `:sameAs`,`tcl:u1`) and thus to re-identify the blank node `_:b0`, as only one user was registered by staff member `tcl:s1`.

A possible solution to this would be to add a privacy query `SELECT ?x ?y WHERE { ?x p ?y }` for each property p for which we suspect that a closure could occur in the LOD cloud. Then, the update queries returned by our algorithms will replace each constant in the subject or object position of such a property by a fresh blank node. For instance, in Example 5.7, the constant `tcl:s1` in (`_:b0`,`tcl:registeredBy`,`tcl:s1`) would be replaced by a fresh blank node.

Note that this process can in the end become very destructive in the graph, particularly if the considered properties are vital to the utility of the graph. Indeed, the more external knowledge there is on the data (whether it is possessed by an attacker or inferred through mechanisms such as the ones studied above), the harder it will be to guarantee utility without deleting or replacing large chunks of the graph, which in turn may often be among the "useful" part of the data.

6

Implementation and experimental evaluation

▷ *We now delve into a more concrete aspect of the framework, by detailing how the safety solution from Chapter 5 is implemented in practice, and how its efficiency and usability are tested with a carefully designed experimental study. We analyze both the performance of our algorithms and of the anonymization, and we discuss how to quantify the loss of information when altering a graph, and how to define what are the best anonymization sequences in our process.* ◁

Chapter summary

THE main motivation of this chapter is to test the efficiency and usability of the solution designed in Chapter 5 in plausible real-life use-cases. To do so, we explore the various types of metrics that could be used, as well as the ones we select and why (Section 6.1). We then explore the design of both experimental studies, first through the selection of data used in our experiments (Section 6.2) then with the detailed experiments ran on our safety solution (Section 6.3). Finally, we conclude with additional tests that could be designed as there are many possible directions to evaluate both privacy, utility, performance, or other aspects (Section 6.4).

The goal of our evaluation is to find ways to quantify the balance between privacy and utility. We must therefore take into account the characteristics and formal semantics of both solutions and how they change possible metrics. For example, the experiments designed on our first contribution focus more on compatibility issues, while the evaluation of the safety framework focuses more on actual usage and anonymization of real-world graphs.

The implementation developed for this evaluation is freely available online in a GitHub repository[1] and these experiments are easily reproducible on one's own system.

6.1 Motivation

In addition to theoretical guarantees, we need actual measurements of the usability and the efficiency of our solution, with regard to both privacy and utility. Our algorithms are guaranteed to satisfy the privacy requirements given as input, so we have to use other attributes to measure and check which factors impact those attributes.

The usual way to quantify its efficiency, besides privacy leakages and data disclosure, it to quantity the data utility left in the sanitized dataset. Since there is no universal way to do so, and since this heavily relies on the database's structure, many utility metrics exist in the literature. We analyze some of them that are relevant in our scope of study.

6.1.1 Measuring privacy

If one of the (proved) anonymization processes is applied, then absolutely no leakage of sensitive information is possible. Therefore, there is no specific privacy attribute that can be measured: if the framework is used for a given graph, privacy is automatically achieved to the extent of what the considered algorithm and privacy policy allow.

[1] https://github.com/RdNetwork/safe-lod-anonymizer

6.1.2 Measuring utility

We would now like to evaluate utility (i.e. the loss of useful information) in an anonymized graph: we know that the utility queries given a policy input will be satisfied, but we do not know how destructive our operations have been (since there are several possible sequences of operations in general), how usable is the remaining information in the graph and what affects this usability in the operations. To do so, there are several possibilities, depending on the focus and the chosen utility semantic. We browse a large panorama of existing utility measurements, summarized in Table 6.1 and described depending on their characteristics:

- **Static metrics:** are they static/data-independent metrics? (e.g. measurements based on the input to the anonymization framework)
- **Graph-based metrics:** are they based on and applicable to graph databases? (e.g. nodes/edges-related metrics, path computation)
- **RDF-based metrics:** are they based on and applicable to RDF triple stores and Linked Data, using specific elements from these contexts? (e.g. metrics based on blank nodes, IRIs)
- **Learning-based metrics:** are they based on data mining or learning techniques? (e.g. classification, regression, random forests, etc.)
- **Probability-based metrics:** are they based on probabilistic computations? (e.g. estimate that a certain value is preserved)
- **Query-based metrics:** are they dependent on a given workload of queries? (e.g. query estimation, data mining statistics)
- **Entropy-based metrics:** are they based on information theory metrics and entropy computations?
- **Distance-based metrics:** are they based on computing a form of distance between the original and anonymized graph (or a portion/generalization of them)?
- **Tuple-based:** are they based on computations computed on the tuples from the anonymized graph (number of modified values, equivalence classes, etc.)?

We first highlight some simple, intuitive **static metrics** applicable in our LOD context. Provided that the anonymization processes consists in (or can be abstracted to) native SPARQL update operations, various statistics could be used to create a hierarchy of the sequences outputted by our algorithms:

- The *number of* `DELETE` *and* `UPDATE` *queries by sequence* could be a simple but rough way of classifying anonymization sequences, as it basically emphasizes how many triple patterns will be replaced and and how many will just be deleted.

- The *number of operations performed on meta-triples*, such as `rdf:type` triples: this focuses on how much general information is lost on the graph, rather that focusing on its contents; in a way, this is a rough computation of how the abstract structure of the graph is affected by the anonymization sequence.
- The *number of triples per body of* `DELETE/UPDATE` *queries* is the simplest way to assume the data loss in a data-independent way. Rather than just counting `DELETE/UPDATE` operations, this focuses on the actual number of triples that could be deleted or replaced to guess how destructive an anonymization sequence would be.

These measurements are very intuitive and simple to understand, and also close to the best one can do in a data-independent fashion (i.e. on judging the anonymization process beforehand rather than the anonymized graph).

We now browse the literature for data utility measurements, as well as some state-of-the-art data management software such as ARX [Prasser15], to look for additional metrics that could be used.

One of the main inspirations of our work, the logical PPDP framework from Grau and Kostylev [Grau19], defines a theoretical *optimality criterion*, that is a preorder between "suppressor functions" (equivalent to our anonymization operators). For a given graph G, an operation o_1 would then be "more informative" than an operation o_2 if the anonymized graph $o_1(G)$ preserves more constants and labeled blank nodes from the original graph than $o_2(G)$. The optimal anonymization process is therefore one using the most informative operators with regard to this preorder.

We now focus on the metrics used in k-anonymity methods and its derivatives. Prior to Sweeney's work, P. Samarati already defined four basic utility metrics for its k-anonymity principle, to compare various k-anonymizations [Samarati01]. They were the absolute and relative distance between the original and anonymized database using various anonymization processes, a distribution metric (maximum distribution is achieved for the anonymized dataset containing the most distinct records) and a suppression metrics (count the number of deleted records). In the original k-anonymity framework [Sweeney02b, Sweeney02a], Sweeney based its utility measurement on the minimal distortion of the data table to be anonymized; this metric is heavily data-dependent, as it is then used to choose the "best" k-anonymous solution on a given table. A classic metric coming from k-anonymity and related methods is to use *minimal average group size*, i.e. the average size of the equivalence classes in a k-anonymous (or l-diverse, etc.) graph, as shown in multiple works [Samarati01, LeFevre05, Machanavajjhala07]. Again, this type of measurement is relevant to measure structural destruction in a relational database, as these equivalence classes are in fact the number of "bulks of records" in which each record is indistinguishable from the other. In addition to this, in l-diversity [Machanavajjhala07] and t-closeness [Li07], Machanavajjhala et al. and Li et al. use two measurements: *the generalization height*, that is the maximum height of a gen-

eralization tree or lattice used when anonymizing the dataset (in short, the maximum amount of generalization steps used when anonymizing) and a *discernability* metric (from [Jr.05]), i.e. the size of the equivalence classes/bulks of indistinguishable records. In their multidimensional implementation of *k*-anonymity, Lefevre et al. [LeFevre06a] use these metrics as well, adding a new one to the list: they compute the *mean and standard deviation of the absolute error* resulting from the evaluation of a query workload. This captures another form of information loss based on usage and value estimation, in addition to structural information loss. In the same fashion, V. Iyengar [Iyengar02] designed two usage-based metrics for *k*-anonymity methods. The first is a *loss metric*, based on the amount of different values between original and anonymized databases, and normalized for each column where a generalization or a suppression took place. The second is a *classification metric*, measuring the loss of "purity" of specific target variables that could be used for predictive modeling or machine learning.

These metrics were broadly criticized by Kifer and Gehrke [Kifer06]. Generalization height is considered to have too much variability, as there can be many different attributes with various possible generalization trees or lattices. Using equivalence classes, loss computation and discernability depends too much on the original data distribution, and are therefore seen as too *ad hoc* metrics. Iyengar's classification metric is very specific, and doesn't really work if classifiers must be built for multiple columns. In general, they criticize what they call "local heuristics", such as information/loss ratios (as in [Wang05]), since they make difficult comparisons between the utility of two different anonymized databases. They in the end suggest a new type of measurement, more formal, using the Kullback-Leibler divergence (previously mentioned in Section 2.1.3) to measure the preservation of marginal values in the data.

Others focused on query-based and workload-based metrics, or other forms of usage-based measurements. Brickell and Shmatikov emphasized this in [Brickell08], judging that any utility analysis of a dataset must be, at least in part, based on a workload context.

> *Without a workload context, it is meaningless to say whether a graph is "useful" or "not useful", let alone to quantify its utility. [...] It has been recognized that utility of sanitized databases must be measured empirically, in terms of specific workloads such as classification algorithms. This does not necessarily contradict the "unknown workload" premise of sanitization. It simply acknowledges that even when sanitization satisfies a syntactic damage minimization requirement, it may still destroy the utility of a graph for certain tasks; it is thus essential to measure the latter when evaluating effectiveness of various sanitization methods.*
>
> (J. Brickell, V. Shmatikov. In [Brickell08])

Some examples of such an utility analysis were actually designed prior to this statement, but are good example of workload-based utility evaluation. LeFevre et al. [LeFevre06b] designed a set of "workload-aware" anonymization metrics, based on

classification and regression techniques. What is evaluated as utility is there the accuracy over a test set, using decision trees, naive Bayes classification, random forests, SVM, linear regression and regression trees. Using a query workload, Rastogi et al. uses the estimation of counting queries as a boolean utility metric [Rastogi07]. The point is to define a guaranteed upper bound on the absolute error of the estimator, satisfied by all queries from the workload.

By nature, differential privacy offers more formal guarantees in terms of utility, as it is part of its definition to keep statistical integrity. Nevertheless, some more precise metrics have also been defined for specific differential privacy algorithms. For example, Hardt et al. defined estimation-based metrics to test their "Multiplicative Weights - Exponential Mechanism" differentially private algorithm [Hardt12]. This includes computing the *total squared error* for given range queries, the *relative entropy* in the case of binary contingency table release, and the *average absolute error* for three-dimensional contingency tables.

Graph anonymization frameworks usually have their own sets of utility definitions and metrics, usually focused on the structural integrity of the graph. Intuitive measurements can for example be focused on vertices and edges, connected components, links or joins between graphs, or blank nodes in the specific case of RDF. Many statistics can then be computed, related to equivalence classes, cardinality, clustering, and many more. We now provide multiple examples of previous works illustrating this.

A simple idea is to adapt classic methods from the relational world to the structure of graph databases. For example, for their adaptation of k-anonymity for RDF graphs (k-RDFanonymous graphs), Radulovic et al. adapt the minimum distortion metric from Sweeney, but on triples from RDF graphs instead of tuples from relational databases [Radulovic15]. Zheleva and Getoor [Zheleva07] also previously introduced the notion of "observations" removed by the anonymization process, in a way similar to various loss metrics mentioned earlier. Although, they note themselves that more sophisticated measurements should be defined to account for the potential loss of structural properties. Cormode et al. [Cormode08] worked on adapting workload-based metrics, their utility analysis being defined the accuracy of sample aggregate queries.

For more specialized metrics, we can find metrics used in network analysis to help us in the context of graph databases. Hay et al. [Hay08] notably reported five major metrics to test the effects of graph anonymization:

- *Degree*: the distribution of the degrees of all vertices in the graph;
- *Path length*: the distribution of the lengths of the shortest paths between 500 randomly sampled pairs of vertices in the network;
- *Transitivity* (a.k.a. *clustering coefficient*): the distribution of values where, for each vertex, we find the proportion of all possible neighbor pairs that are connected;

- *Network resilience*: plotting the number of vertices in the largest connected component of the graph as nodes are removed in decreasing order of degree (originally from [Albert00]);
- *Infectiousness*: plotting the proportion of vertices infected by a hypothetical disease, which is simulated by first infecting a randomly chosen node and then transmitting the disease to each neighbor with the specified infection rate (originally from [Watts98]).

All or some of these metrics are very often reused in fields such as social network analysis, as shown in [Yuan13] where the average path length is used, in addition to the *average change of length of a sensitive path* using designated labels, and the *remaining ratio of the most influential entities in the graph* after anonymizing the graph. Nobari et al. [Nobari14] also focus on clustering and path length, computing the earth mover's distance between the shortest path distribution in both original and anonymized graph. This distance is also computed between the degree distributions of both graph.

The ARX Anonymization Tool includes many measurements taken from existing literature. We list each metric with its potential source, most of them being studied previously in this section. Previously detailed metrics include average equivalence class size (from [LeFevre05]), discernability (from [Jr.05]), generalization height (from [Machanavajjhala07]), loss (from [Iyengar02]), precision (from [Sweeney02a]), KL-Divergence (from [Kifer06]), and classification accuracy (from [Iyengar02]). Two other metrics are implemented:

- *Non-uniform entropy* (from [Prasser16]), used to measure distance between distributions of given attributes; it can be also be normalized.
- *Ambiguity* (from [Goldberger10, Nergiz07b]), which is the average size of the Cartesian products of the generalized values in each record of the database. This in fact measures the theoretical total number of possible combinations of the original records that a generalized record could imply.

All the metrics we mentioned in this section are summarized in Table 6.1 below, to summarize the study. It is easy to observe that there are many different ways to measure utility, towards completely different aspects of the considered dataset.

Metric	Static	Graph-based	RDF-based	Learning methods	Prob.-based	Query-based	Entropy-based	Distance-based	Tuple-based
Average equivalence class size								✓	
Discernability								✓	
Classification metric				✓	✓			✓	✓
Loss / Granularity								✓	✓
(Normalized) Non-uniform entropy							✓		✓
Ambiguity									✓
KL-Divergence (relative entropy)							✓	✓	
Publisher payout (prosecutor/journalist)					✓				
Entropy-based information loss							✓		
Suppressor optimality						✓			✓
Minimal distortion of table					✓			✓	✓
RDFPrec			✓					✓	✓
Generalization tree height									✓
Upper bound of counting query error						✓		✓	
Total squared error								✓	
Average absolute error								✓	
Absolute distance								✓	✓
Relative distance								✓	✓
Maximum distribution	✓								✓
Minimum suppression	✓								✓
Classification and regression				✓					
Nb. of removed observations		✓							✓
Accuracy of aggregate query types						✓		✓	
Degree of all vertices		✓							✓
Average path length		✓							✓
Clustering coefficient		✓							
Network resilience		✓							
Infectiousness		✓							
Avg. change of sensitive path length		✓							✓
Ratio of influential entities		✓							
Wasserstein distance (on degree dist.)		✓							
DELETE/UPDATE quantity	✓		✓						

Table 6.1: Comparative analysis of various data utility measurements

6.1.3 Conclusion and selected metrics

Using this existing literature and the characteristics of our context of work and our frameworks, we now decide on which metrics will be evaluated in our measurements. While we can easily conclude that measuring privacy is not the priority, we must spend more time regarding utility metrics.

In the context of our safety-related contribution, utility guarantees do not exist in the anonymization process (as opposed to the first contribution) and utility must therefore be checked and measured post-anonymization. For this more complex case, we decide on multiple utility metrics, capturing various aspects of the anonymized graphs, focusing on information loss on a structural level and using metrics relevant to the context of LOD.

Precision loss for LOD: relative and absolute number of added blank nodes

Following the *RDFprec* metric from [Radulovic15], itself inspired by Sweeney's original *Prec* metric [Sweeney02a], we devise a measurement of the loss of precision caused by the anonymization process in an RDF graph. It uses a concept inherent to the Semantic Web: blank nodes. By counting the number of blank nodes added by the anonymization, we can easily analyzes how much information could be lost by breaking links between triples using common IRIs where one occurrence was replaced by a blank node. Our precision loss measurement is a $[0, 1]$ value computed by the following formula:

$$Prec = \frac{\text{number of blank nodes added by the anonymization}}{\text{total number of IRIs in the original graph}}$$

It provides a precision value relative to the contents of the original graph, and how destructive was our anonymization in replacing some of the graph's IRIs by blank nodes.

Structural destruction of graph structure: distance between degree distribution

Looking at the degree distribution of a graph is one of the many typical metrics used in graph analysis [Hay09, Lin13] and comparing two of these distributions to compare two graphs as well. We use one of these comparisons using the distance between the degree distributions of a graph before and after anonymization, from [Nobari14]. The Wasserstein distance (or Earth mover's distance) between two degree distributions of the same size and sum is the "minimum work" necessary to transform one distribution into the other. Here, this distance represents the amount of nodes moved times the distance by which it is moved. This measurement is useful in the sense that it allows us to see structural differences in the graph, for example if a lot of "core" nodes (with a lot of incoming and outgoing edges) have been modified. This type of changes may not perceptible when computing simple precision values, while computing such a distance captures more abstract changes like this one.

Structural information loss: number of connected components

To go further in this structural study, we use another classical graph analysis metric: the *number of connected components* in an anonymized graph. Connected components are an important tool in social network analysis, for example to look for central patterns or regions in a graph (applications exist for example in biological networks [Albert00] or computer vision [Dillencourt92]). Assessing the number of connected components can therefore help summarize the structure of a graph: transitioning from few big components to many small components could indicate a big information loss in our anonymization process.

Triple information loss: similarity

Finally, another classic way to analyze the structural modifications to a graph is to measure more directly the information loss on the graph. Rather than computing statistics such as the degree distribution, we compute a similarity metric based on the number of identical triples found in the graph between the original and anonymized versions.

6.2 Experimental data: The quest for RDF graphs and workloads

We now focus on the RDF graphs where our experiments will be carried out: how they are selected, why, and what are their characteristics.

6.2.1 Criteria

Testing our frameworks requires RDF graphs, whether their contents are artificial or real-world data. And since we will need to define privacy and utility policies on those datasets, they should be adequately selected based on semantics and contents which fit these requirements.

In order to design a proper experimental evaluation, multiple RDF graphs of various purposes, sizes and structures have been selected: smaller and bigger sets of data, real-world data as well as synthetic data, and so on. The criteria to select them were:

1. their *availability* as usable data dumps;
2. the fact that they presented *explicit, named entities* such as users or products, to mimic sensitive data;
3. the *simplicity and readability of their schema and vocabulary*, to put ourselves in the role of a real-world Data Protection Officer designing policies.

Availability is a crucial concern. Previous works showed that many RDF graphs and SPARQL endpoints lacked stability, interoperability, discoverability and availabil-

ity [Aranda13, Neto16, Debattista18] with downtime periods, dead links, overloading issues, and even sometimes complete disappearance from the Web. Data publishing also lacks uniformity, sometimes not respecting the 5-star principles detailed in Section 1.1 or simply by containing erroneous data, and are therefore not easily downloadable or processable [Beek16]. Some graphs are also sometimes available as SPARQL endpoints, but not as complete data dumps (or not in a direct way), hence this requirement. To circumvent these issues, we use the LOD Laundromat project [Beek14, Beek16] which processes, unifies and store every graph detected in the LOD cloud, making it available in both its "dirty" (original) form or its "clean" (processed and unified) form.

6.2.2 Selected sets of data

For these experiments, we select four graphs of various sizes and semantics and following the criteria defined in the previous section.

The **TCL**[2] graph models synthetic transportation data based on real data from the Grand Lyon Open Data portal. As mentioned in Section 1.5, we use an adequately designed RDF data generator to simulate what a sensitive transportation would look like by adding personal user data to this already public information. We also model external knowledge by adding another public accessible information: location data regarding all places of worship in the Lyon area. Typical entities include transportation lines and stops, users, subscriptions, places of worship.

LUBM[3] is a famous benchmark to evaluate the performance of triple stores and query engines using big synthetic graphs modeling university data. We use a data generator to model the data for numerous universities, providing with a considerably-sized graph to be used as a performance test for the anonymization operations. To still make computation and experimentats possible, we settled on a value of 100 universities, which still created an extensive yet processable graph. Typical entities include universities, students, teachers, courses.

The **Drugbank**[4] graph contains real-world data regarding approved drugs in the United States of America, and details regarding their contents, their usage, their branding, as well as possible chemical composition and studies. Drugs are the prominent entity in this graph.

The **Europeana Swedish Heritage**[5] graph contains heritage data regarding the whole country of Sweden, collected from the Europeana heritage portal. Typical entities are heritage locations.

[2]Original data portal: https://data.grandlyon.com/

[3]Main project: http://swat.cse.lehigh.edu/projects/lubm/ - Generator software: https://github.com/rvesse/lubm-uba

[4]http://wifo5-03.informatik.uni-mannheim.de/drugbank/

[5]General Europeana portal: https://pro.europeana.eu/page/linked-open-data

The details regarding each graph and their associated reference policies are reported in Table 6.2. The indicators are the following:

- The *number of triples* in the graph;
- The *number of unique IRIs* (the number of different labels that appear in subject, predicate or object position);
- The *number of unique (anonymous) blank nodes* already in the graph;
- The *number of queries* in each reference policy;
- The *size* (total number of triples) of each reference policy.

Table 6.2: Summary of the various graphs used in our experiments.

Graph	Number of triples	No. of unique IRIs	No. of unique blanks	No. of queries in privacy policy	Privacy policy size
TCL	6,443,256	1,020,580	705,030	7	19
LUBM	13,405,383	13,363,445	0	3	7
Drugbank	517,023	109,494	0	3	14
Swedish Heritage	4,970,464	1,687,452	0	4	13

6.3 Evaluating the safety framework

We now evaluate various aspects of the framework from the second contribution presented in Chapter 5. This solution puts aside utility policies, but provides a tighter privacy guarantee protecting against linkage attacks using shared IRIs. In comparison to the previous one, this solution adds therefore more blank nodes to the graph, and does not have a direct way to judge the utility of the dataset.

Thus, we decide to focus this time on using real-world graphs that we actually anonymize, using fixed custom policies where we introduce variability to study how fine-tuned modifications to the privacy policy queries (and therefore how much of the initial graph they cover) impact the algorithm and its generated anonymization.

And in addition to the classical measurement of the running time, we only focus on utility measurements by selecting multiple metrics aimed at measuring information loss in various ways (similarity, structural alterations, lost IRIs).

6.3.1 Experimental setup

Input

In order to later study the output of the framework (the operation sequence and the subsequently anonymized graphs), we need to fix the input and decide which parameters are relevant to our evaluation. Indeed, given the input provided to the algorithms, many parameters could be fixed and studied, notably:

- The number of queries and the size (total number of triples) of each privacy policy;
- The semantics of each policy query (what is supposed to be hidden in the end);
- The result variables of each policy query.

We define the notion of **policy specificity**: a policy is *more specific* (or *less general)* than another if the total number of results of its queries is *smaller*.

In the same fashion, since it would be impossible to model every possible privacy policy (whether it is manually or automatically), we also provide a fixed input for the framework by manually writing policies for each selected graph. Thus we define a *reference policy* for each RDF graph, manually written with plausible queries based on the semantics and vocabulary of each graph. These "plausible semantics" are defined by looking at the graph's schema and its general goal as a dataset published to the LOD cloud. Reference policies are sets of conjunctive queries providing a simple representation of what a data provider could seemingly want to hide from their data. In a real-life scenario, these policies should be defined by the DPO in charge of defining the public part of the database to be published. For example, the chosen queries for the TCL graph aim at hiding the given name, family name, postal address, birth date and subscription details of any user existing in the graph. Full policies (as lists of SPARQL queries) are reported in Appendix A.

Policy mutation

To assess the resulting utility after anonymization, we create ***mutations* of the reference policies**, i.e. incremental modifications to the graph patterns of the privacy policy's queries. The type of mutation considered here is the replacement of a variable in a policy query's body by a constant; the resulting mutated policy will therefore be *more specific* than the original. The idea is to measure the utility loss on the anonymized graph as a function of the specificity of the policy: a mutated policy should be *less destructive*, indeed as the mutated applies only to more specific cases, lesser blank nodes should be introduced. When the policy is extremely specific, for instance when it hides only a completely instantiated subgraph without variables, no blank nodes should be introduced.

For each graph, we generate a finite set of mutations of the original reference policy by creating a $t \times m$ matrix of mutations with two parameters: a number t of parallel copies of the reference policy and a number m of mutations to apply in each copy. This way, we create t different threads of incremental mutations in order to explore the impact of specificity. We report an example of such a "thread" of mutations over a reference policy based on our running transportation example in Example 6.1. In this example, we have $\mathcal{P}_3 \models \mathcal{P}_2 \models \mathcal{P}_1 \models \mathcal{P}$.

Example 6.1

Let $\mathcal{P}$ a policy consisting in this unique query:

```
SELECT ?name ?dest
WHERE {
    ?user  a  ex:User .
    ?user  ex:familyName    ?name
    ?user  ex:destination   ?dest .
    ?user  ex:nationality   ?nat .
}
```

Let $m = 3$. A possible thread of mutations for this policy would be as follows. Each new mutations is **highlighted in bold**. For the first mutation $\mathcal{P}_1$:

```
SELECT ?name ?dest
WHERE {
    ?user  a  ex:User .
    ?user  ex:familyName    ?name
    ?user  ex:destination   "Madrid" .
    ?user  ex:nationality   ?nat .
}
```

For the second mutation $\mathcal{P}_2$:

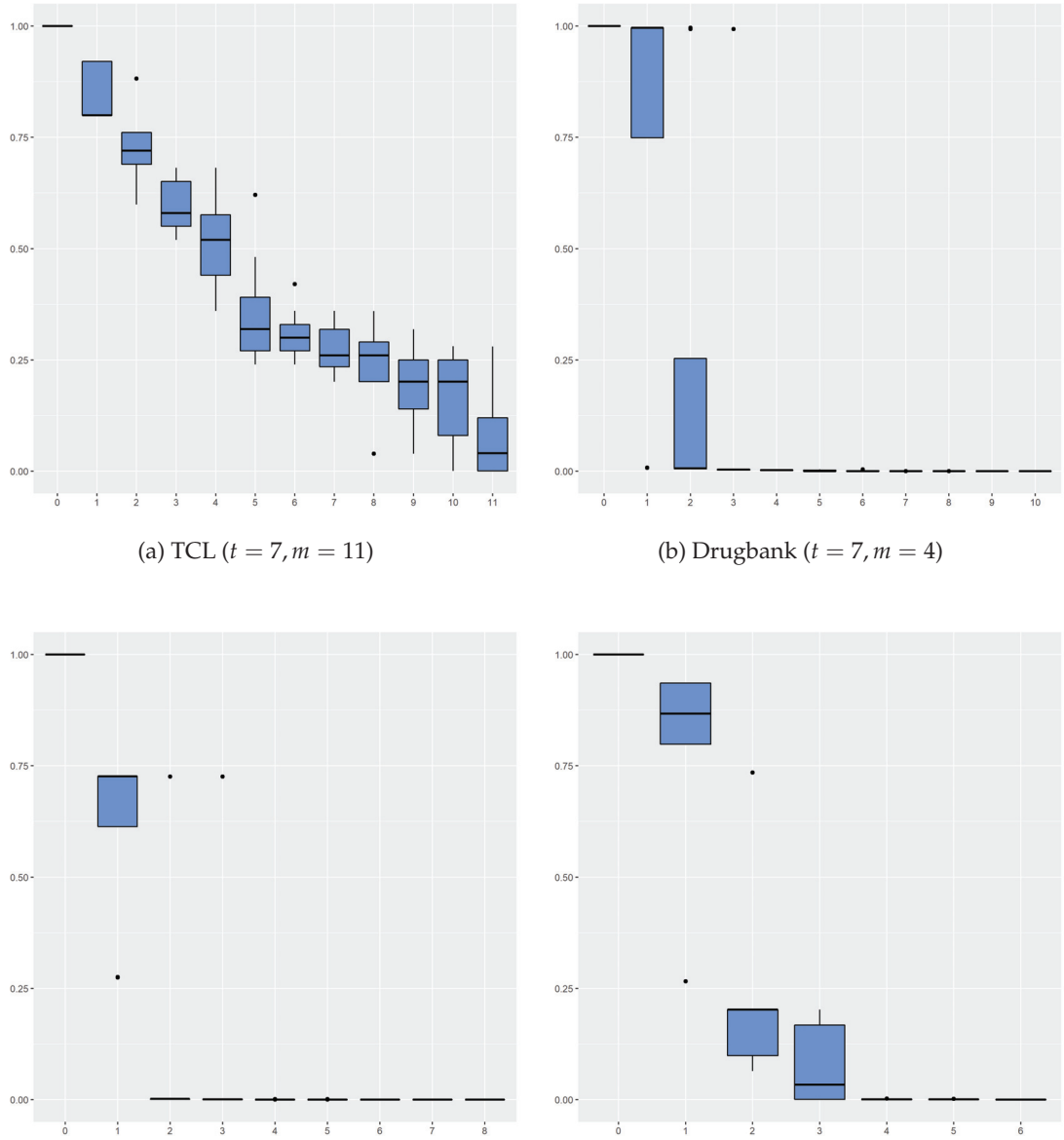
```
SELECT ?name ?dest
WHERE {
    ex:user28  a  ex:User .
    ex:user28  ex:familyName     ?name
    ex:user28  ex:destination    "Madrid" .
    ex:user28  ex:nationality    ?nat .
}
```

And for the third mutation $\mathcal{P}_3$:

```
SELECT ?name ?dest
WHERE {
    ex:user28  a  ex:User .
    ex:user28  ex:familyName    ?name
    ex:user28  ex:destination   "Madrid" .
    ex:user28  ex:nationality   "French" .
}
```

The first policy of each thread therefore matches the original policy. Formally, the

specificity of a mutated policy $\mathcal{P}'$ of a base reference policy $\mathcal{P}$ on a graph G is defined as $\mathsf{specificity}(\mathsf{P}') = |\mathsf{Ans}(\mathsf{P}', \mathsf{G})|/|\mathsf{Ans}(\mathsf{P}, \mathsf{G})|$. The boxplots on Figure 6.1 describes how policy specificity evolves with mutation depth; i.e., how many successive mutations we need to perform on a policy before its queries start having no results on the graph anymore.



(a) TCL ($t = 7, m = 11$)

(b) Drugbank ($t = 7, m = 4$)

(c) Heritage ($t = 7, m = 8$)

(d) LUBM ($t = 5, m = 8$)

Figure 6.1: Specificity depending on mutation depth

In the case of the TCL graph, 11 mutations is a fitting threshold, as shown on Figure 6.1. The figure shows that the mutation process covers the whole spectrum of selectivity, from 0 (a policy with very few or no variables that return almost no results on G) to 1 (the original reference policy). Choosing relevant values for t and m is indeed heavily dependent of the considered graph and its related policy; notably, the size of its policy.

Each mutated policy has its own specificity, and we compute a normalized value between 0 and 1 by dividing by the specificity of the original, non-muted policy. This will form the x-axis of our experiments.

The experiments were generated and performed using Python 2.7, using the same base code structure than the previous solution, and using the `SPARQLWrapper` package[6] to manage SPARQL endpoints and real SPARQL data through RDF graphs stored on a Virtuoso 7 triple store. The server machine used is a Linux Ubuntu 18.04 virtual machine with 128GB of RAM and 12 virtual CPUs, running using OpenStack.

In terms of performance, just like the previous contribution, the staticity of the approach ensures a good running time for the generation of anonymization sequences; but now, we have to consider the actual anonymization duration. The running time then depends on the size of graph and of the policy. Indeed, the bigger the policy and its queries are, the longer the operation sequence will be. Table 6.3 indicates for each graph the length of anonymization sequence, that is the number of updates to apply, and the maximum running time of the anonymization operations across all mutations.

Table 6.3: Running time of anonymization operations.

Graph	No. of operations	Algorithm duration (s)	Anonymization duration (s)	Size (MB)	Throughput (MB/s)
TCL	16	0.151	3.5	210	60
Drugbank	28	0.146	2.8	150	54
Swedish Heritage	22	0.149	18.7	910	48
LUBM	13	0.14	107.9	2,290	21

While the number of operations generated can become high, only a few will cost time: the ones dealing with the subgraphs patterns with many occurrences in the graph. In any case, we show that the running time is quite decent. Indeed, our approach uses only *standard SPARQL update queries* which are efficiently processed by heavily optimized RDF databases such as Virtuoso. We also report the performance results of the operation generation algorithm, with the same conclusions as in Section 4.6.1.

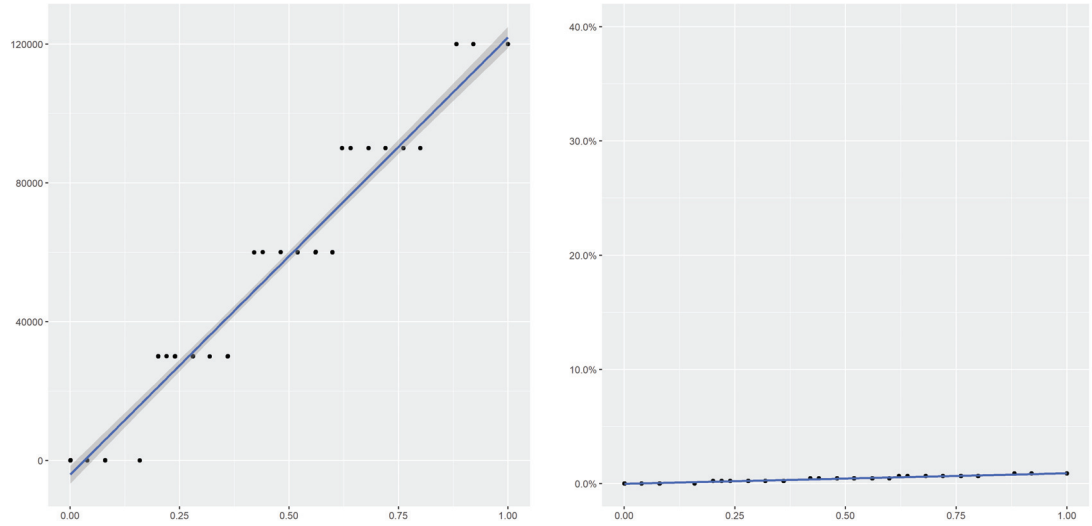
[6] https://rdflib.github.io/sparqlwrapper/

6.3.2 Results

We present the results for each of the four graphs selected in Section 6.2.

Precision loss analysis

To analyze the precision loss and the gradualness of our approach, we measure *the number of blank nodes* added by the anonymization process. Results displayed on Figures 6.2 to 6.5 show that this number grows linearly with policy specificity: the less precise the policy is in its selection of data, the more blank nodes will be inserted to the graph.



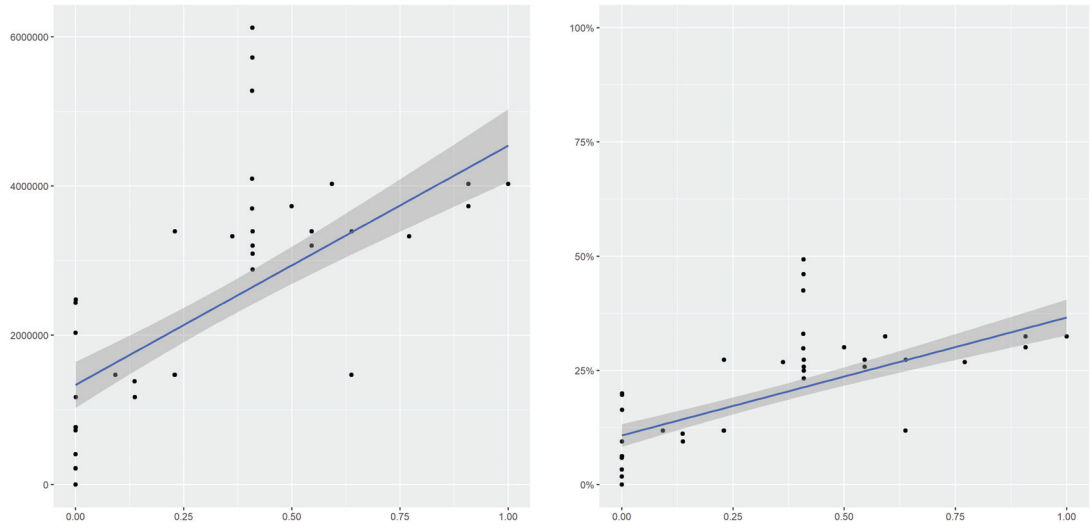
(a) TCL (Absolute precision)

(b) TCL (Relative precision)

Figure 6.2: Absolute and relative (*RDFprec*-based) number of blank nodes introduced depending on the policy specificity for the TCL graph.

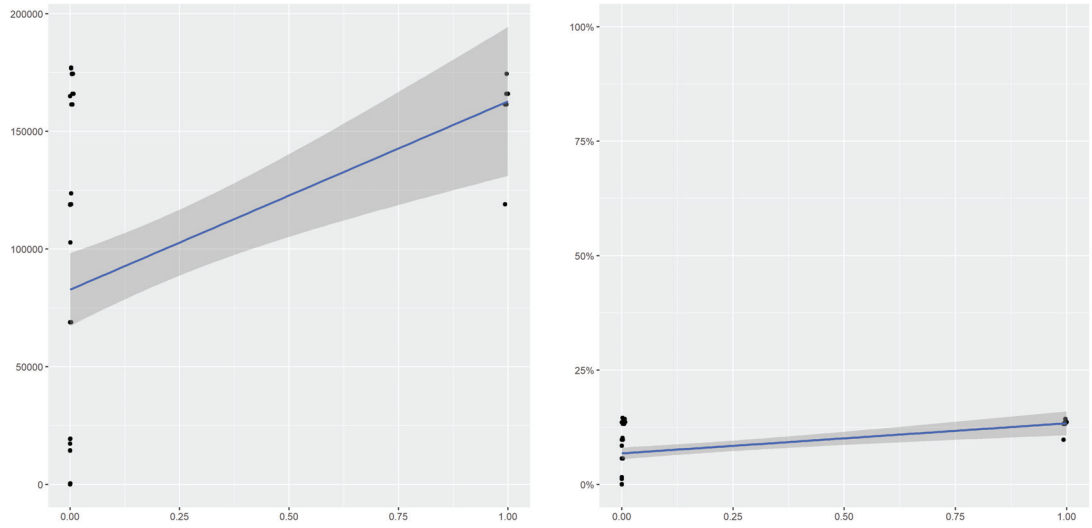
As a relative precision measurement, we use a variant of the *RDFprec* metric to measure how this addition of blank nodes impacts the graph as a whole. Here, *RDFprec* counts the ratio of non-blank nodes in the graph. To keep the same visual trend, the metric used to plot each figure is the *loss of precision*, i.e. the complement percentage.

We can observe on Figures 6.2b to 6.5b that precision is very dependent on the input: if the policy covers only a specific part of the whole data (e.g. only the subscription data in a graph containing data regarding subway lines, bus stops, etc.) its impact is quite small. This explains why in the case of the TCL graph (Figure 6.2a), this precision value only drops by a very low margin (99.9% to 99.4%). The trend is identical on other graphs: precision drops when the policy gets more general. It drops to around 75% in the case of the Swedish Heritage graph, 87% for the Drugbank graph, and 93% for the



(a) Swedish Heritage (Absolute precision) (b) Swedish Heritage (Relative precision)

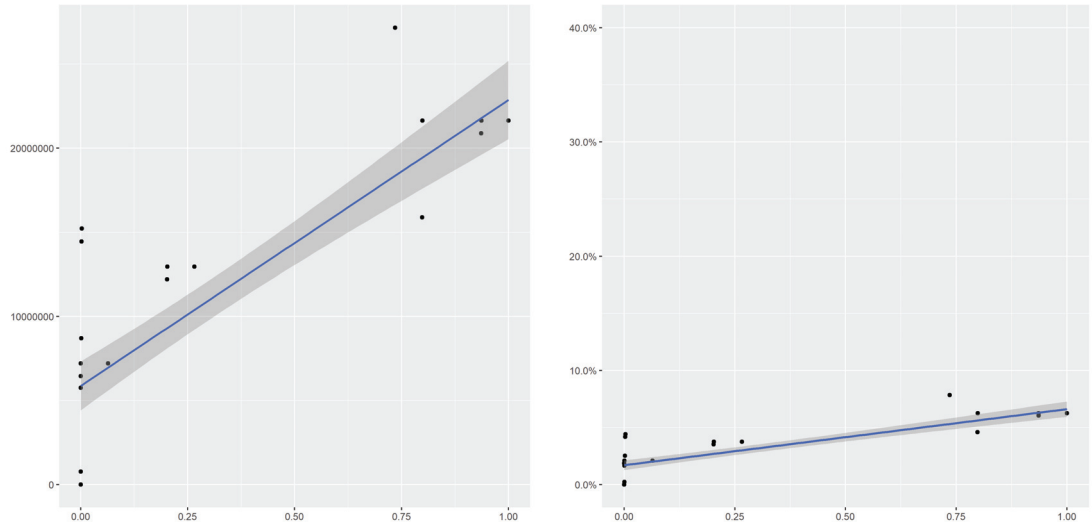
Figure 6.3: Absolute and relative number of blank nodes introduced depending on the policy specificity for the Swedish Heritage graph.



(a) Drugbank (Absolute precision) (b) Drugbank (Relatve precision)

Figure 6.4: Absolute and relative number of blank nodes introduced depending on the policy specificity for the Drugbank graph.

LUBM graph. This confirms that, using plausible policy semantics, the number of IRIs



(a) LUBM (Absolute precision) (b) LUBM (Relative precision)

Figure 6.5: Absolute and relative number of blank nodes introduced depending on the policy specificity for the LUBM graph.

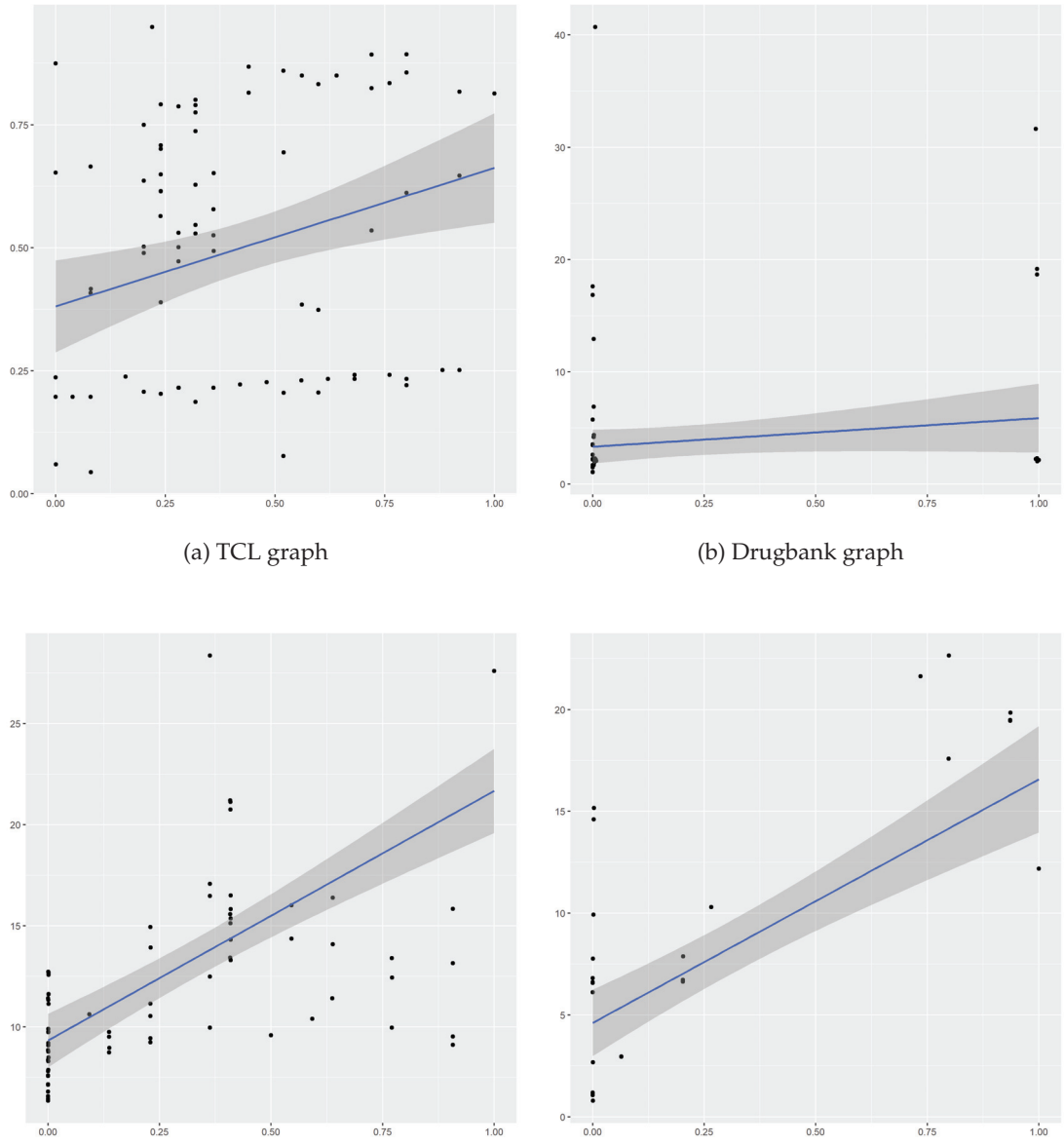
lost in the anonymization process is not huge.

However, we notice that for multiple graphs, most notably on Figures 6.4a and 6.4b for the Swedish Heritage graph, there is quite a large spread on the $x = 0$ line. Indeed, in some cases the mutation process can cause some very general pieces of information (e.g. typing triples, that are identical for many entities in the graph) to be impacted, which can cause a very early drop in precision, since in those situations the anonymization process thus leads to many replacements by blank nodes.

Structural destruction analysis

We then compute our Wasserstein distance measurement between degree distributions to assess the effect of the anonymization on the graphs' structures.

First, the results displayed on Figure 6.6 show that the distance value is low in general: less than 1 for the TCL graph, and between 10 and 30 for the other graphs. This number describes the "cost" of turning one degree distribution into the other, i.e. the number of degrees that would have to be moved. Considering graphs with several hundred thousand nodes, these values are quite low. We also note that for the TCL, Drugbank and Swedish Heritage graphs, the distance exhibits a clear raising tendency: the more stringent the policy is, the farther the anonymized graph is. But this increase can be very scattered: on the TCL graph, we note that the distance increased a lot on situation with a specificity between 0.2 and 0.4. This is again due to the very discrete nature of policies, than can create huge modifications in the graph and its structure,



(a) TCL graph

(b) Drugbank graph

(c) Swedish Heritage graph

(d) LUBM graph

Figure 6.6: Distance between degree distributions of the original graphs and the graphs anonymized using a given policy mutation

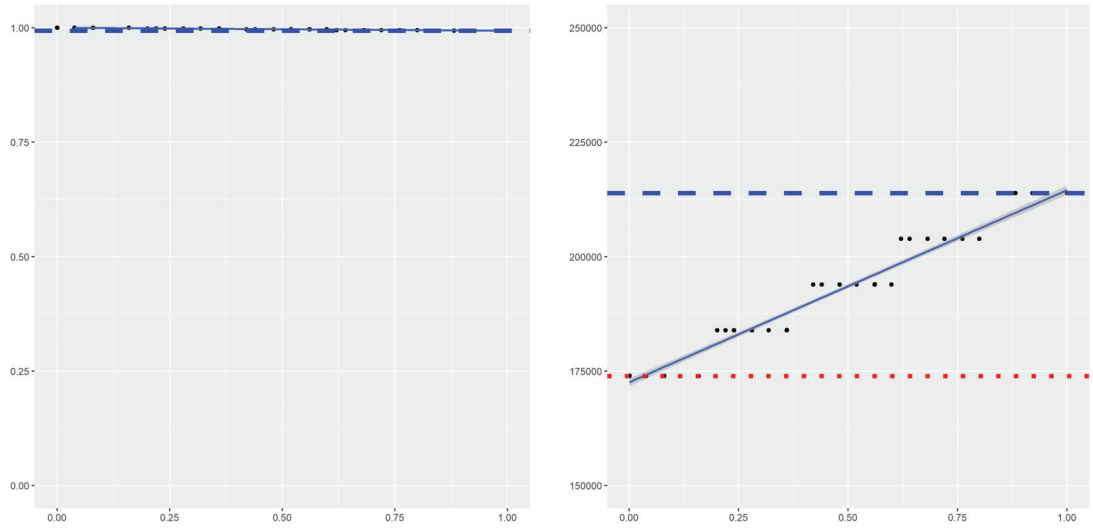
yet have only a few different specificity values, usually grouped in "chunks" that are clearly visible when the size of the privacy queries is low.

Even taking the into account, the fact that the Drugbank graph has a very straightforward structure and fairly simplistic policies creates very few different specificity

values (here, two "chunks" around 0 and 1), but where the distance can vary greatly depending on how prevalent are the edited nodes in the graph.

This problem and the fact that values tend to be too scattered for a definitive conclusion indicate that the distance between degree distributions may not be the most suitable metric to analyze the structural destruction of a graph.

We therefore select two other graph-based measurements for our study. We report the results for these metrics on Figures 6.7 to 6.10: the similarity metric and the number of connected components in the graph. In both figures, the blue dashed line represents the same values but for the graph anonymized using the original, non-mutated policy. Similarly, the red dotted line represents the number of connected components for the original graph, as a point of reference (this dotted red line is not displayed for the similarity graphs, as it would obviously be the line $y = 1$).



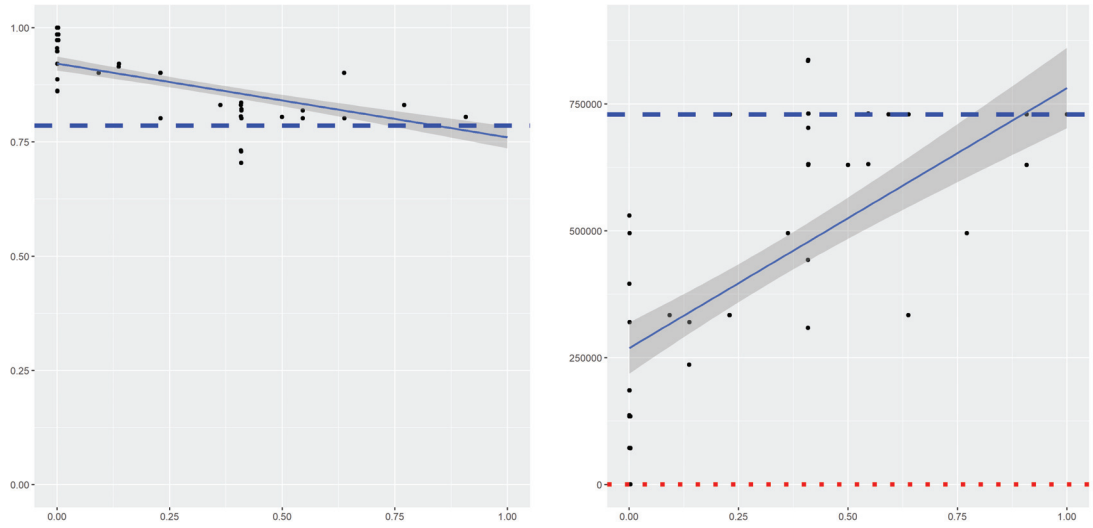
(a) TCL (Similarity)

(b) TCL (Connected comp.)

Figure 6.7: Comparison of the similarity and the number of connected components between original and anonymized versions of the TCL graph.

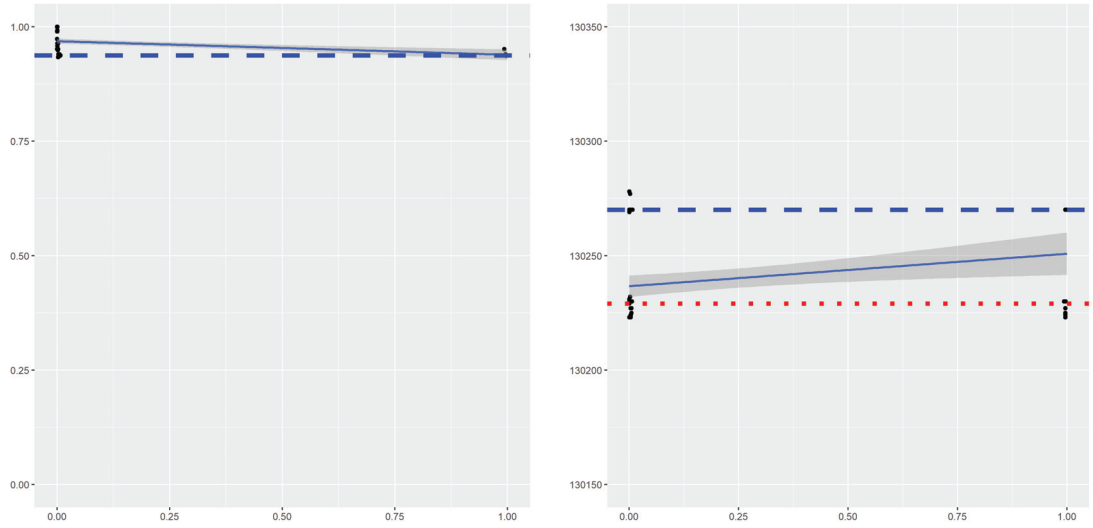
We observe that the similarity between original and anonymized graph is generally high (over 75% for three graphs, even reaching very high percentages for the TCL graph): once again, while this heavily depends on how policies cover the entire schema of each graph, this shows that our approach is not destructive on a triple-per-triple point of view. This explains why, in the case of the LUBM graph, much of this triple-to-triple similarity is lost when policies have high specificity, as they cover many different parts of the data.

The study of connected components produces results in line with our previous study. For the TCL, LUBM and Swedish Heritage graphs, the number of connected



(a) Swedish Heritage (Similarity) (b) Swedish Heritage (Connected comp.)

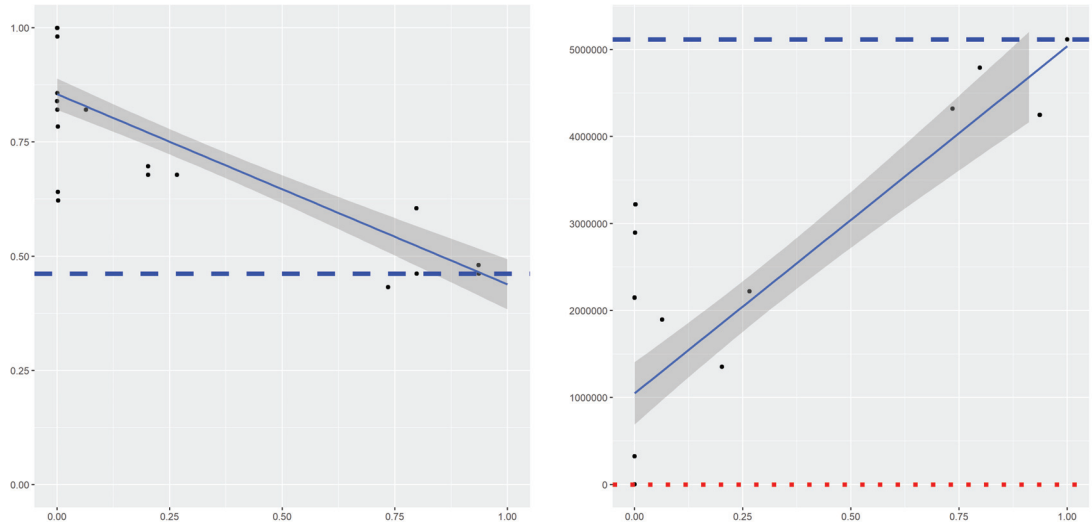
Figure 6.8: Comparison of the similarity and the number of connected components between original and anonymized versions of the Swedish Heritage graph.



(a) Drugbank (Similarity) (b) Drugbank (Connected comp.)

Figure 6.9: Comparison of the similarity and the number of connected components between original and anonymized versions of the Drugbank.

components grows with query specificity: this implies that policy queries covering the



(a) LUBM (Similarity)

(b) LUBM (Connected comp.)

Figure 6.10: Comparison of the similarity and the number of connected components between original and anonymized versions of the LUBM graph.

biggest portion of the graph cause the anonymization to "break" a lot of existing connected components, creating numerous, smaller components using blank nodes. We can note that in some cases as the Drugbank graph, the number of connected components may have only a very small variation. Indeed, the number of connected components heavily depends on the structure of the graph, notably if some triples such as typing exists in each entity: these entities will all link to the same IRI, creating huge components in the graph. When reaching heavily "influencing" nodes such as those, the number of connected components may suffer a big decrease. In our graphs, this notably happens in the TCL graphs: to solve this, we filter out triples using the `rdf:type` predicate. This could in fact be a potentially better way to study connected components: rather than looking at the whole graph (or specific portions by filtering some predicates), it would be interesting to study connected components *per predicate* and study the evolution of such clusters of nodes when anonymized.

In short, we can see that in practice, very strict privacy policies still preserve some structural integrity and does not incur too big of a precision loss. And besides, it is possible to create finer policies, less destructive, but providing better control of the precision and structure of the anonymized graph.

6.4 Experimental prospects

During our experiments, we were able to analyze in various ways how the contents of the input policies impact the anonymization process and the output graph. This further emphasizes their importance in the anonymization of any given database that has to be published to the LOD cloud (or any other public space, for that matter) and the centrality of the Data Protection Officer's role or any other person or organization in charge of precisely defining the boundaries of which data has to be kept intact or removed from a public iteration of a given dataset. This role is already gaining importance due to the scope of application of the GDPR regulations in Europe - and similar cases in the rest of the World - and our study is a simple example of how this position is crucial, not only in organizational matters, but also in concrete and technical contexts.

But still, many other factors could be studied as future research works, focusing notably on the potential public usage of databases anonymized using such solutions. Usage-based studies, such as the study of actual query usage logs of LOD graphs could indeed help shaping policies by identifying important data. In addition to this, our extensive study of possible metrics also indicates that many other numerical factors could be computed to assess the considered anonymization process, focusing on other structural aspects of graph databases, or even focusing quantifying privacy in the anonymized data. This could also lead to a more comprehensive and up-to-date survey on this matter in coming years.

When you do things right, people won't be sure you've done anything at all.
Ken Keeler – *Futurama*, "Godfellas" (2002)

7
Conclusion and perspectives

▷ *To end this manuscript, we recall the key points from each of our contributions and their experimental evaluation. We also report on possible caveats and extensions, as well as prospects in this subfield.* ◁

LOOKING at data privacy from the standpoint of Linked Open Data and the Semantic Web remains a widely open problem, and this exploratory thesis only marks the beginning of deep studies regarding privacy guarantees, frameworks and anonymization algorithms in this context. Our contributions and analysis open the door to many complementary approaches or extensions to our frameworks. We recall the main aspects of our contributions and discuss possible usages and caveats, concluding with potential prospects and obstacles towards reaching them.

We designed two full-fledged, data-independent solutions for anonymizing RDF graphs to be used before publishing them to the Linked Open Data cloud. Each one satisfies certain specific definitions and semantics of data privacy and utility, considering different contexts of usage and privacy concerns. This could be the beginning of a major research direction and change of usage in the grander scheme of the Semantic Web. Indeed, this kind of tool is currently almost non-existent, while privacy issues are still rampant and while the LOD movement has been lacking mainstream momentum in recent years.

Our first solution deals with RDF graphs taken individually, where a compromise must be found between specific *privacy* and *utility* requirements. This is the most intuitive type of anonymization framework, but it does not address specific privacy issues resulting from how Linked Data operates. To address specific privacy concerns related to linkage attacks, we also designed a more privacy-focused framework guaranteeing

a new notion of *safety*, considering these issues that are more inline with actual usage of RDF LOD datasets.

Both solutions are easy-to-use, as they only require marginal knowledge of Linked Data and SPARQL to formalize privacy or utility requirements, and the rest is knowledge regarding the contents and the semantics of the graphs to be anonymized. Any data provider should thus be able to handle these frameworks, especially the mandatory Data Protection Officers in a GDPR context. These two contributions were theoretically validated, as each provides proven privacy and utility guarantees and anonymization algorithms satisfying those guarantees. Yet they were validated in practice as well, though implementations in Python and experimental evaluations on both the execution of these algorithms and the anonymization of concrete RDF graphs.

Data privacy gets increasingly complex with each incremental constraint put on data release or on privacy semantics, such as the utility constraints studied in this thesis. Besides, new standards or software used in the field may pose new privacy challenges as well. Fortunately, given how our contributions are structured, possible extensions are basically infinite. Any other anonymization approach can be used in conjunction with our own anonymization framework. Moreover, our privacy semantics can be easily extended and the algorithms adapted in order to generate new correct anonymization operations in different contexts.

However, this is not without caveats. Notably, just like there was an escalation of tighter privacy guarantees for relational databases starting with k-anonymity and up to more modern derivatives, one can possibly design other types of attacks on RDF graphs. Those would use other Semantic Web technologies or specific loopholes in reasoning/query engines that would require new additions to our framework. Other potential problems could occur if some standards (such as versions of RDF and SPARQL) are deprecated and not handled well anymore by query engines. This would render some anonymization operations impossible to apply, or change their behavior in unpredictable or erratic ways.

As a whole, it is hard to predict the future of the Semantic Web and Linked Open Data. The whole field is mostly pushed by academics and public institutions in selected areas, and it lacks momentum to become a *de facto* standard or a huge technological leap. The notion of Semantic Web tends to be supplanted by usages of artificial intelligence and its derivatives technologies (such as deep learning) rather than declarative or static sets of data, and many think that Semantic Web and Linked Data have not picked up the pace and are almost obsolete[1]. It should be noted that even its W3C founders, while still actively maintaining committees and working groups on every existing standard, seems to focus more on Web applications especially regarding privacy, as shown by Tim Berners-Lee's new Solid project[2]. All of these contributions have therefore to

[1] https://twobithistory.org/2018/05/27/semantic-web.html
[2] https://solid.mit.edu/

be considered while taking account of the democratization of these usages, or on the contrary the potential loss of interest.

Nonetheless, and even if analyzing privacy issues within the scope of the Semantic Web and privacy-preserving data publishing of RDF graphs are both very new fields, this type of work is important. For example, some still think that LOD and Semantic Web are reaching a *"slope of enlightment"*, meaning that after a period of disillusionment, improvements should be arriving anytime soon[3]. Indeed, this line of research should provide with a different kind of research momentum for LOD and the Semantic, which could hopefully in turn encourage more research by various actors. The W3C could provide with new technical standards and tools handling anonymization or privacy modeling. Institutional and private actors could develop solutions dealing with concrete technical constraints for anonymizing data. And researchers could extend work such as this PhD project, or design new solutions or frameworks. Besides, the popular demand for more data transparency and data privacy is not likely to end soon, ensuring a future in any shape or form for the LOD cloud. The framework we designed , while linked to current Semantic Web standards, remains generic and declarative and can be thus adapted to any of these evolutions.

Short-term goals include defining more concrete degrees of privacy to provide with a whole range of LOD anonymization frameworks, and extending these frameworks to other Semantic Web mechanisms such as complex reasoning. Longer-term goals could be focused on studying the characteristics and caveats of a privacy/utility trade-off on the specific context of Linked Open Data, with both formal studies and usage studies performed by potential LOD providers, to evaluate more precisely the need for LOD anonymization solutions.

Immediate future work should thus also focus on how all these privacy-preserving LOD solutions could be integrated into existing data management pipelines and LOD usages, and how they could be automated or made more clearly available to actual data providers or Data Protection Officers. Given the current state of the Semantic Web, the main focus of future research (whether it is applied or theoretical) should be based on usage and usability, to solve concrete privacy problems in LOD contexts. Our approach using declarative constraints to ensure the GDPR-recommended *privacy by design* provides a solid basis to initiate usage-based discussions on how to formalize privacy.

[3] https://medium.com/@schivmeister/the-semantic-web-where-is-it-now-f4773f3097e3

A

Policies used for experiments

▷ *Complete SPARQL policies used in our experiments for each framework.* ◁

Chapter summary

We report each policy used when experimenting on both frameworks in Chapter 6. To ease readability, we use the SPARQL standard notation to write prefixed IRIs[1] instead of full IRIs.

A.1 TCL graph

Query 1: Hide users' given name(s)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?user ?name
WHERE {
    ?user foaf:givenName ?name .
}
```

Query 2: Hide users' family name

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?user ?surname
WHERE {
    ?user foaf:familyName ?surname .
}
```

Query 3: Hide users' address(es)

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
SELECT ?user ?address
WHERE {
    ?user vcard:hasAddress ?address .
}
```

Query 4: Hide users' birth date

```
PREFIX tcl: <http://localhost/>
SELECT ?user ?birth
WHERE {
    ?user tcl:birthday ?birth .
}
```

Query 5: Hide the starting date of each user's potential subscription

```
PREFIX datex: <http://vocab.datex.org/terms#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX tcl: <http://localhost/>
SELECT ?name ?surname ?startSubDate
WHERE {
    ?user a tcl:User .
    ?user foaf:givenName ?name .
    ?user foaf:familyName ?surname .
    ?user datex:subscription ?subDate .
```

[1] See the W3C SPARQL 1.1 Recommendation [Harris13], notably Section 4.1.1, "Syntax for IRIs".

```
10      ?tabDate datex:subscriptionStartTime ?startSubDate .
11  }
```

Query 6: Hide the stopping date of each user's potential subscription

```
1   PREFIX datex: <http://vocab.datex.org/terms#>
2   PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3   PREFIX tcl: <http://localhost/>
4   SELECT ?name ?surname  ?endSubDate
5   WHERE {
6       ?user a tcl:User .
7       ?user foaf:givenName ?name .
8       ?user foaf:familyName ?surname .
9       ?user datex:subscription ?subDate .
10      ?tabDate datex:subscriptionStopTime ?endSubDate .
11  }
```

Query 7: Hide the membership type of each user's potential subscription

```
1   PREFIX datex: <http://vocab.datex.org/terms#>
2   PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3   PREFIX tcl: <http://localhost/>
4   SELECT ?name ?surname ?subType
5   WHERE {
6       ?user a tcl:User .
7       ?user foaf:givenName ?name .
8       ?user foaf:familyName ?surname .
9       ?user datex:subscription ?subDate .
10      ?tabDate datex:subscriptionReference ?subType .
11  }
```

A.2 LUBM graph

Query 1: Hide students' name

```
1   PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2   PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
3   SELECT ?name
4   WHERE {
5       ?x  rdf:type    ub:GraduateStudent .
6       ?x  ub:name     ?name .
7   }
```

Query 2: Hide students' telephone number(s)

```
1   PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2   PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
3   SELECT ?name ?tel
4   WHERE {
5       ?x  rdf:type        ub:UndergraduateStudent .
6       ?x  ub:name         ?name .
7       ?x  ub:telephone    ?tel .
```

```
}
```

Query 3: Hide courses taken by each student

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?name ?cours
WHERE {
    ?student    ub:takesCourse  ?cours .
    ?student    ub:name         ?name .
}
```

A.3 Swedish Heritage graph

Queries 1 to 3: Hide who contributed for a precisely localized proxy at a given time

A proxy can be identified by its description, its spatial coordinates, or its title.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterm: <http://purl.org/dc/terms/>
SELECT ?c ?t ?m
WHERE {
    ?s dc:creator ?c .
    ?s dcterm:spatial ?t .
    ?s dcterm:temporal ?m .
}
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterm: <http://purl.org/dc/terms/>
SELECT ?c ?d ?m
WHERE {
    ?s dc:creator ?c .
    ?s dc:description ?d .
    ?s dcterm:temporal ?m .
}
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterm: <http://purl.org/dc/terms/>
SELECT ?c ?d ?m
WHERE {
    ?s dc:creator ?c .
    ?s dc:title ?d .
    ?s dcterm:temporal ?m .
}
```

Query 4: Hide the aggregations to which each user contributed

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
SELECT ?c ?u
WHERE {
```

```
    ?s edm:aggregatedCHO ?i .
    ?s edm:unstored ?u .
    ?p ore:proxyFor ?i.
    ?p dc:creator ?c .
}
```

A.4 Drugbank graph

Query 1: Hide the toxicity of branded drugs

```
PREFIX dbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/> .
SELECT ?b
WHERE {
    ?d  a   dbank:drugs.
    ?d  dbank:affectedOrganism  ?o .
    ?d  dbank:brandName ?b.
}
```

Query 2: Hide the brand name of drugs aimed at specific organismes

```
PREFIX dbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/> .
SELECT ?b
WHERE {
    ?d a dbank:drugs.
    ?d dbank:affectedOrganism ?o .
    ?d dbank:brandName ?b.
}
```

Query 3: Hide the details of interactions between branded drugs

```
PREFIX dbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/> .
SELECT ?x ?y ?t
WHERE {
    ?a a dbank:drugs .
    ?a ?d dbank:brandName ?x.
    ?b a dbank:drugs .
    ?b ?d dbank:brandName ?y.
    ?i a dbank:drug_interactions .
    ?i dbank:interactionDrug1 ?a .
    ?i dbank:interactionDrug2 ?b .
    ?i dbank:text ?t.
}
```

B

Schema and characteristics of the synthetic TCL graph

▷ We detail the schema and the modeling of the TCL rDF graph, which was specifically designed for this thesis, using public data available on the Grand Lyon Open Data platform and artificial data to model sensitive information. ◁

Chapter summary

We detail each predicate used in the graph, its semantic meaning, and the type of values it can take. We first analyze how previously available data was adapted to RDF, then which type of data is generated to model sensitive personal information.

Every entity local to the transportation network is prefixed with `gld:`, for Grand Lyon Data.

B.1 Public data

B.1.1 Lines and stops

Using the SYTRAL (institution managing the Lyon transportation network) data and the information they provide on the complete network, we can create entity for each stop and each line in the network. Each line is a numbered entity, prefixed by `b` for bus lines, `t` for tramway lines, and `s` for subway lines, and has the following data:

- `rdf:type`: Typing varies on the transportation vehicle. Subway lines are typed `gtfs:Subway`, tramway are `gtfs:LightRail`, and buses are `gtfs:Bus`;
- `tcl:lineNumber`: the name and number of the line on the network (a character string);
- `tcl:indexNumber`: an internal identifier (a character string);
- `rdfs:label`: the full name of the line, with its origin and destination (a character string);
- `tcl:orientation`: the direction of line, either "Aller" (original orientation) or "Retour" (reverse orientation);
- `tcl:stops`: a collection of `gtfs:Stop` objects. Each stop is itself described by two coordinates (`geo:latitude` and `geo:longitude`);
- `tcl:titanCode`: another internal software identifier (a character string);
- `tcl:garageCode`: a code for the type of vehicle used, e.g. the train type for a subway line (a character string).

B.1.2 Places of worship

Places of worship are numbered entities starting with the letter `w`, of the type `lgdo:placeOfWorship`. Using the actual list of the 197 places of worship in the Grand Lyon area and the available pieces of information for each place, we provide the following data on each place of worship:

- `rdfs:label`: the name of this place of worship (a character string);
- `geo:latitude`: the latitude coordinate of this place of worship (a float, supposedly between 45 and 46);
- `geo:longitude`: the longitude coordinate of this place of worship (a float, supposedly between 4 and 5);
- `gld:id`: an internal identifier for this place of worship (a character string);
- `gld:creationDate`: the establishment date of this place of worship (an `xsd:date` formatted string);

B.2 Artificial data

B.2.1 Users

10,000 fictional users are modeled in this graph. They are all given fake names (using a software generator) and fake addresses (included in our generator code, using fictional street names, and cities and ZIP codes from the area).

Each user is a `tcl:User` described by:

- `foaf:givenName`: their first name (a character string);
- `foaf:familyName`: their last name (a character string);
- `vcard:hasAddress`: their postal address (a character string);
- `foaf:givenName`: their birth date (an `xsd:date` formatted string);
- `datex:subscription`: their (optional) subscription to the TCL network (see following section).

B.2.2 Subscriptions

Users may hold a subscription to the network. To account for this, we model them as `datex:Subscription` objects, described by:

- `datex:subscriptionReference`: the name of subscription (a character string);
- `datex:subscriptionStartTime`: the starting date of the subscription (an `xsd:date` formatted string);
- `datex:subscriptionStopTime`: the ending data of the subscription (an `xsd:date` formatted string).

B.2.3 Journey validations

Journey validations model the fact that users are traveling on the TCL network. This graph include 1 million validations, which are `tcl:Validation` objects described by:

- `tcl:validator`: an identifier of the terminal used to the validate the journey (a character string);
- `tcl:validationDateTime`: the exact date and time of the validation (an `xsd:date` formatted string);
- `tcl:user`: the (optional) user who validated this journey (a `tcl:User`);
- `geo:latitude`: the latitude coordinate of this validation (a float, supposedly between 45 and 46);
- `geo:longitude`: the longitude coordinate of this validation (a float, supposedly between 4 and 5).

C

Additional experimental results for the privacy/utility solutions

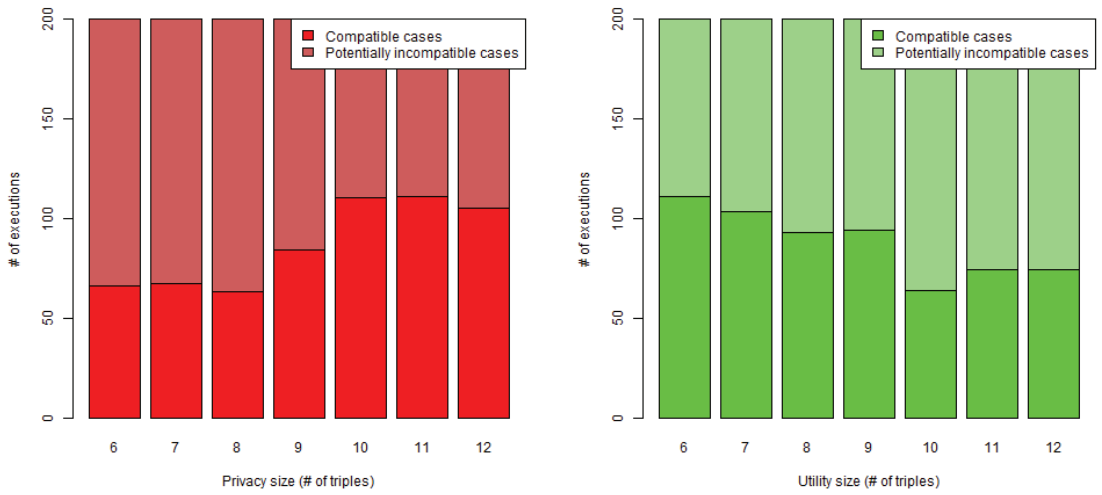
▷ *We provide additional experimental results for the solution designed in Chapter 4, using various types of queries in the genrerted policies. The conclusions and observations remain the same.* ◁

Chapter summary

C.1 Chain queries

C.1.1 Compatibility metrics

Compatibility rates for each type of policy are reported on Figure C.1.



(a) For privacy policies

(b) For utility policies

Figure C.1: Policy compatibility based on privacy and utility policy size [Chain queries]

C.1.2 Properties of the candidate anonymization sets

We report properties of the outputted candidate sets: first on the overlap ratio on Figure C.2, then on the length of those candidate sets on Figure C.3.

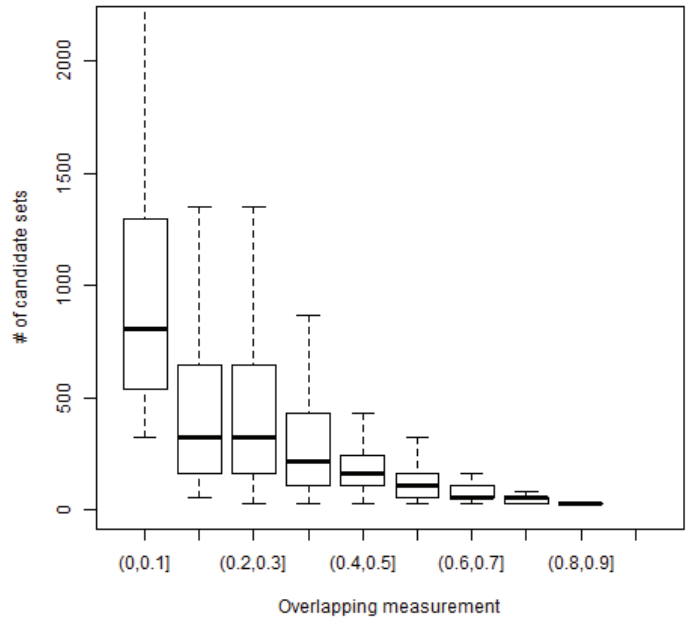
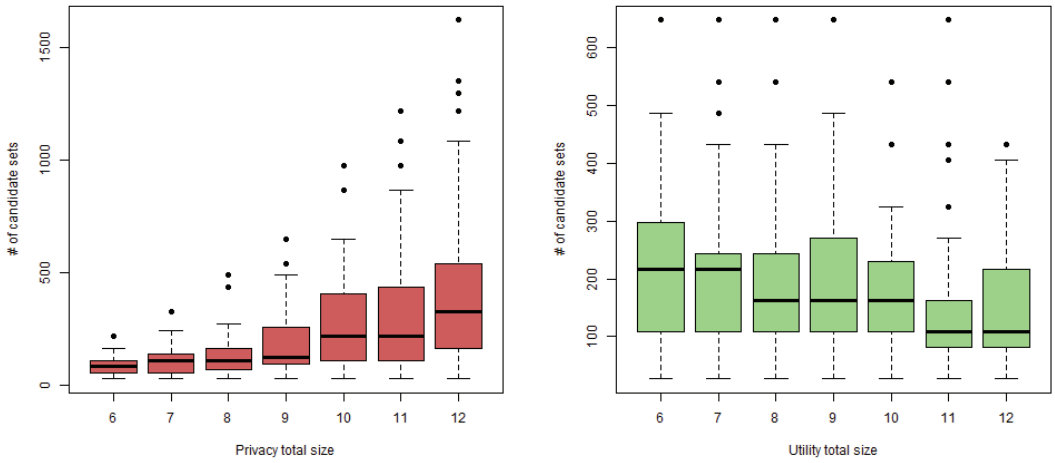


Figure C.2: Candidate set length based on policy overlap [Chain queries]



(a) Depending on privacy size

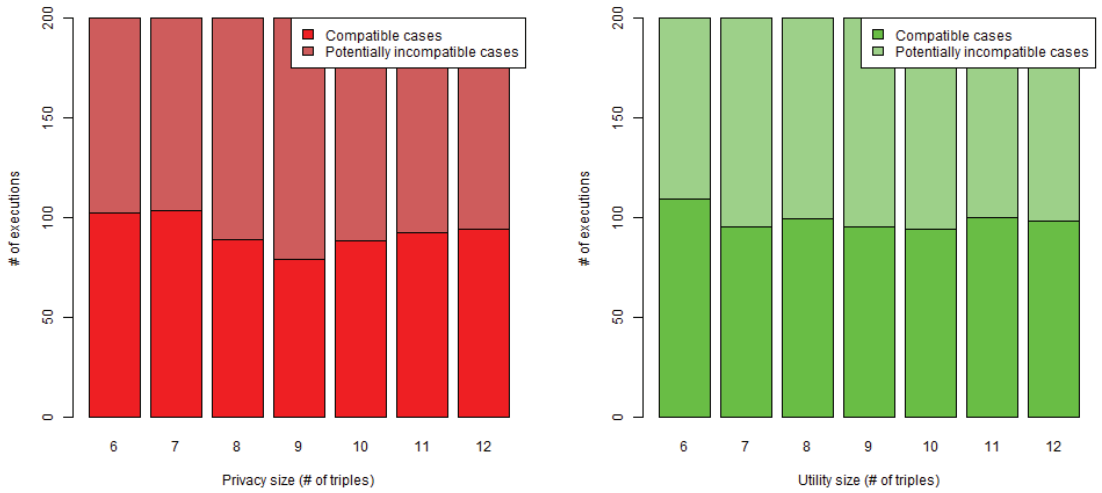
(b) Depending on utility size

Figure C.3: Candidate set length based on the size of both policies [Chain queries]

C.2 Star queries

C.2.1 Compatibility metrics

Compatibility rates for each type of policy are reported on Figure C.4.



(a) For privacy policies

(b) For utility policies

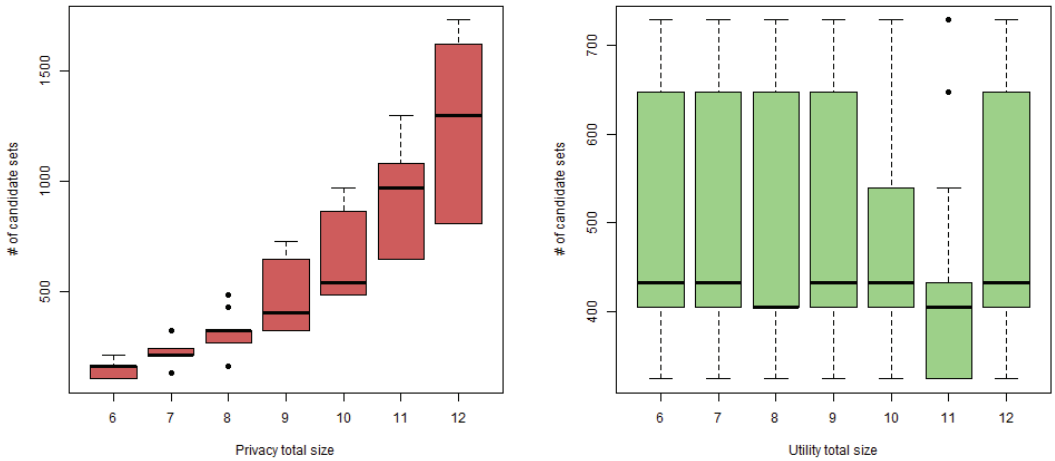
Figure C.4: Policy compatibility based on privacy and utility policy size [Chain queries]

C.2.2 Properties of the candidate anonymization sets

We report properties of the outputted candidate sets. In the case of a workload consisting exclusively in star queries, the first overlap graph is not relevant since every execution of the algorithm featuring policies with an overlapping measure over 0% results in an incompatible case. Compatible cases are then exclusively without any overlapping, which makes such a graph (candidate sets length over overlapping measure) irrelevant. We also note that the trend of incompatibility measurements is quite different than in the previous cases.

This happens because of the particular structure of star queries. We think that our notion of overlapping tends to be quite strict for this class of queries, since we work with edge-based overlapping. Such a kind of overlapping is problematic when it happens with star queries, since the center of star is always involved and blocks possible deletions. This suggests another alternative for defining overlapping, a node-centered definition.

We therefore only show results the length of those candidate sets on Figure C.5.



(a) Depending on privacy size

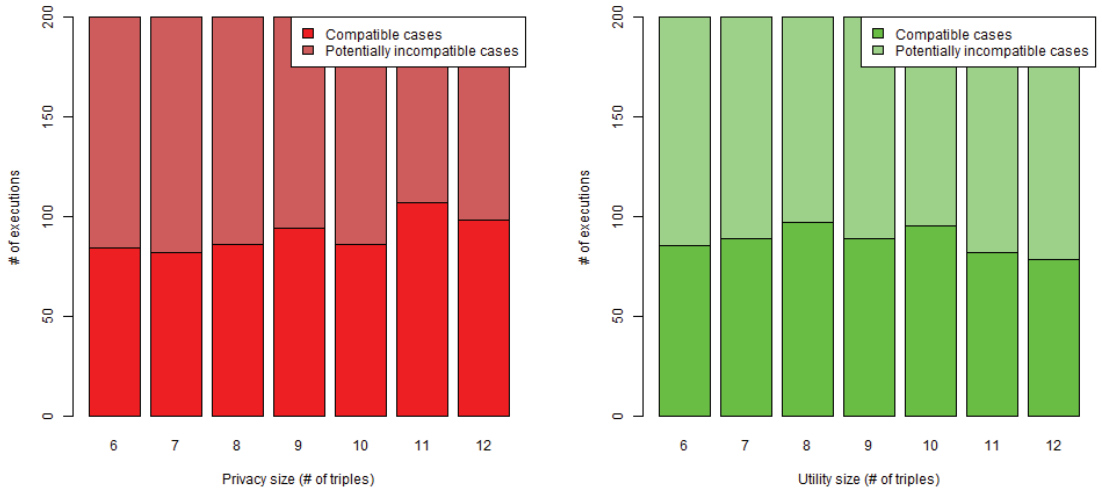
(b) Depending on utility size

Figure C.5: Candidate set length based on the size of both policies [Chain queries]

C.3 Star and starchain queries

C.3.1 Compatibility metrics

Compatibility rates for each type of policy are reported on Figure C.6.



(a) For privacy policies

(b) For utility policies

Figure C.6: Policy compatibility based on privacy and utility policy size [Star and starchain queries]

C.3.2 Properties of the candidate anonymization sets

We report properties of the outputted candidate sets: first on the overlap ratio on Figure C.7, then on the length of those candidate sets on Figure C.8.

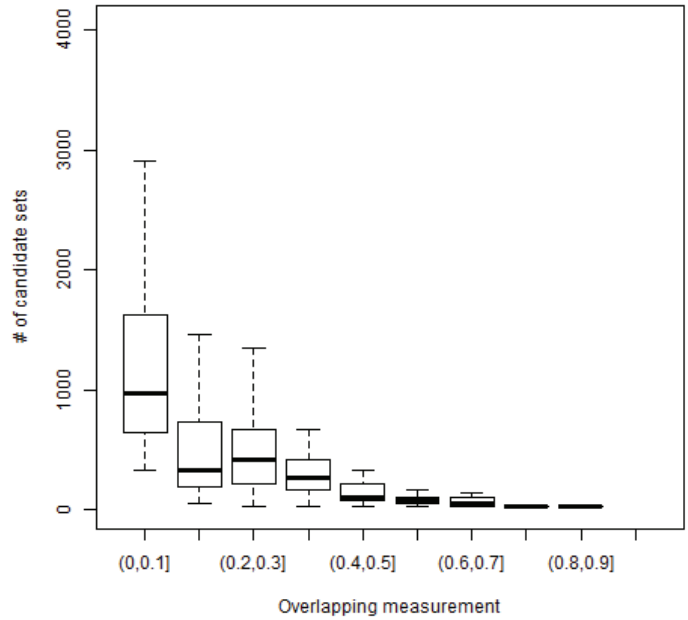
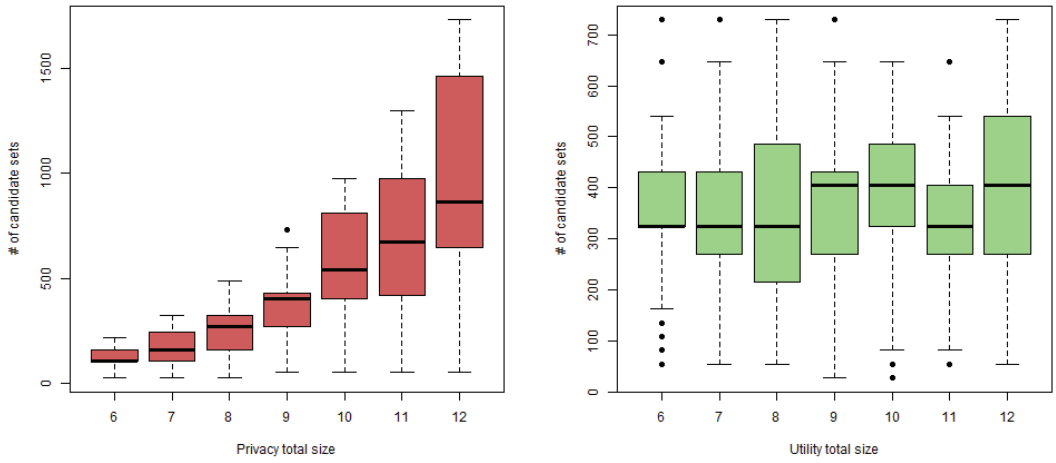


Figure C.7: Candidate set length based on policy overlap [Star and star-chain queries]



(a) Depending on privacy size

(b) Depending on utility size

Figure C.8: Candidate set length based on the size of both policies [Star and star-chain queries]

Bibliography

[Abiteboul95] Serge Abiteboul, Richard Hull & Victor Vianu. Foundations of databases. Addison-Wesley, 1995. 34, 36

[Agrawal00] Rakesh Agrawal & Ramakrishnan Srikant. *Privacy-Preserving Data Mining*. In SIGMOD Conference, pages 439–450. ACM, 2000. 12

[Albert00] Réka Albert, Hawoong Jeong & Albert-László Barabási. *Error and attack tolerance of complex networks*. Nature, vol. 406, no. 6794, pages 378–382, Nature Publishing Group, 2000. 99, 102

[Aranda13] Carlos Buil Aranda, Aidan Hogan, Jürgen Umbrich & Pierre-Yves Vandenbussche. *SPARQL Web-Querying Infrastructure: Ready for Action?* In International Semantic Web Conference (2), volume 8219 of *Lecture Notes in Computer Science*, pages 277–293. Springer, 2013. 103

[Atemezing13] Ghislain Auguste Atemezing, Bernard Vatant, Raphaël Troncy & Pierre-Yves Vandenbussche. *Harmonizing Services for LOD Vocabularies: A Case Study*. In WaSABi@ISWC, volume 1106 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013. 33

[Atemezing14] Ghislain Auguste Atemezing, Boris Villazón-Terrazas & Bernadette Hyland. *Best Practices for Publishing Linked Data*. W3C note, W3C, January 2014. http://www.w3.org/TR/2014/NOTE-ld-bp-20140109/. 72

[Baader17] Franz Baader, Daniel Borchmann & Adrian Nuradiansyah. *Preliminary Results on the Identity Problem in Description Logic Ontologies*. In Description Logics, volume 1879 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. 24

[Backstrom11] Lars Backstrom, Cynthia Dwork & Jon M. Kleinberg. *Wherefore art thou R3579X?: anonymized social networks, hidden patterns, and structural steganography*. Commun. ACM, vol. 54, no. 12, pages 133–141, 2011. 21

[Bagan17] Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George H. L. Fletcher, Aurélien Lemay & Nicky Advokaat. *gMark: Schema-Driven Generation of Graphs and Queries*. IEEE Trans. Knowl. Data Eng., vol. 29, no. 4, pages 856–869, 2017. 65

[Beckett04] Dave Beckett. *RDF/XML Syntax Specification (Revised)*. W3C recommendation, W3C, February 2004. 2

[Beek14] Wouter Beek, Laurens Rietveld, Hamid R. Bazoobandi, Jan Wielemaker & Stefan Schlobach. *LOD Laundromat: A Uniform Way of Publishing Other People's Dirty Data*. In International Semantic Web Conference (1), volume 8796 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2014. 103

[Beek16] Wouter Beek, Laurens Rietveld, Stefan Schlobach & Frank van Harmelen. *LOD Laundromat: Why the Semantic Web Needs Centralization (Even If We Don't Like It)*. IEEE Internet Computing, vol. 20, no. 2, pages 78–81, 2016. 103

[Berners-Lee01] Tim Berners-Lee, James Hendler, Ora Lassila*et al.* *The semantic web*. Scientific american, vol. 284, no. 5, pages 28–37, New York, NY, USA:, 2001. 2

[Berners-Lee06] Tim Berners-Lee. *Linked Data*, 2006. W3C Design Issues. 2

[Berners-Lee08] Tim Berners-Lee. *Linked Open Data*, 2008. Linked Data Planet. 2

[Blocki13] Jeremiah Blocki, Avrim Blum, Anupam Datta & Or Sheffet. *Differentially private data analysis of social networks via restricted sensitivity*. In ITCS, pages 87–96. ACM, 2013. 23

[Bonifati17] Angela Bonifati, Wim Martens & Thomas Timm. *An Analytical Study of Large SPARQL Query Logs*. PVLDB, vol. 11, no. 2, pages 149–161, 2017. 47, 66

[Brickell08] Justin Brickell & Vitaly Shmatikov. *The cost of privacy: destruction of data-mining utility in anonymized data publishing*. In KDD, pages 70–78. ACM, 2008. 13, 16, 97

[Brickley14] Dan Brickley & Ramanathan Guha. *RDF Schema 1.1*. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/. 33, 62

[Bun16] Mark Bun & Thomas Steinke. *Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds*. In TCC (B1), volume 9985 of *Lecture Notes in Computer Science*, pages 635–658, 2016. 19

[Buron19] Maxime Buron, François Goasdoué, Ioana Manolescu & Marie-Laure Mugnier. *Reformulation-based query answering for RDF graphs with RDF ontologies*. In ESWC, to appear, 2019. 34

[Bursztyn15] Damian Bursztyn, François Goasdoué & Ioana Manolescu. *Reformulation-based query answering in RDF: alternatives and performance*. PVLDB, vol. 8, no. 12, pages 1888–1891, 2015. 62

[Calvanese07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini & Riccardo Rosati. *Tractable Reasoning and Efficient Query Answering in Description Logics: The* DL-Lite *Family*. J. Autom. Reasoning, vol. 39, no. 3, pages 385–429, 2007. 62

[Cao12] Jianneng Cao & Panagiotis Karras. *Publishing Microdata with a Robust Privacy Guarantee*. PVLDB, vol. 5, no. 11, pages 1388–1399, 2012. 17

[Carothers14] Gavin Carothers & Eric Prud'hommeaux. *RDF 1.1 Turtle*. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-turtle-20140225/. 30

[Carroll04] Jeremy J Carroll & Patrick Stickler. *TriX: RDF triples in XML*. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, pages 412–413. ACM, 2004. 30

[Casas-Roma17] Jordi Casas-Roma, Jordi Herrera-Joancomartí & Vicenç Torra. *A survey of graph-modification techniques for privacy-preserving on networks*. Artif. Intell. Rev., vol. 47, no. 3, pages 341–366, 2017. 20

[Chandra77] Ashok K. Chandra & Philip M. Merlin. *Optimal Implementation of Conjunctive Queries in Relational Data Bases*. In STOC, pages 77–90. ACM, 1977. 36

[Chawla05] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam D. Smith & Hoeteck Wee. *Toward Privacy in Public Databases*. In TCC, volume 3378 of *Lecture Notes in Computer Science*, pages 363–385. Springer, 2005. 18

[Chen07] Bee-Chung Chen, Raghu Ramakrishnan & Kristen LeFevre. *Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge*. In VLDB, pages 770–781. ACM, 2007. 17

[Chen12] Rui Chen, Benjamin C. M. Fung, Bipin C. Desai & Nériah M. Sossou. *Differentially private transit data publication: a case study on the montreal transportation system*. In KDD, pages 213–221. ACM, 2012. 20

[Chen13] Shixi Chen & Shuigeng Zhou. *Recursive mechanism: towards node differential privacy and unrestricted joins*. In SIGMOD Conference, pages 653–664. ACM, 2013. 23

[Cheng10] James Cheng, Ada Wai-Chee Fu & Jia Liu. *K-isomorphism: privacy preserving network publication against structural attacks*. In SIGMOD, pages 459–470. ACM, 2010. 22

[Chirkova09] Rada Chirkova. Query containment, pages 2249–2253. Springer US, Boston, MA, 2009. 42

[Cormode08] Graham Cormode, Divesh Srivastava, Ting Yu & Qing Zhang. *Anonymizing bipartite graph data using safe groupings*. PVLDB, vol. 1, no. 1, pages 833–844, 2008. 98

[Cormode09] Graham Cormode, Divesh Srivastava, Smriti Bhagat & Balachander Krishnamurthy. *Class-based graph anonymization for social network data*. PVLDB, vol. 2, no. 1, pages 766–777, 2009. 21

[Cormode10] Graham Cormode, Ninghui Li, Tiancheng Li & Divesh Srivastava. *Minimizing Minimality and Maximizing Utility: Analyzing Method-based attacks on Anonymized Data*. PVLDB, vol. 3, no. 1, pages 1045–1056, 2010. 15

[Dalenius86] Tore Dalenius. *Finding a needle in a haystack or identifying anonymous census records*. Journal of official statistics, vol. 2, no. 3, page 329, Statistics Sweden (SCB), 1986. 4

[de Bruijn10] Jos de Bruijn & Stijn Heymans. *Logical Foundations of RDF(S) with Datatypes*. J. Artif. Intell. Res., vol. 38, pages 535–568, 2010. 35

[Debattista18] Jeremy Debattista, Christoph Lange, Sören Auer & Dominic Cortis. *Evaluating the quality of the LOD cloud: An empirical investigation*. Semantic Web, vol. 9, no. 6, pages 859–901, 2018. 103

[Delanaux18] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset & Romuald Thion. *Query-Based Linked Data Anonymization*. In ISWC (1), volume 11136 of *LNCS*, pages 530–546. Springer, 2018. 7, 45

[Delanaux20] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset & Romuald Thion. *RDF graph anonymization robust to data linkage*. In WISE (2), volume TBD of *Lecture Notes in Computer Science*. Springer, 2020. 7, 72

[Deutsch05] Alin Deutsch & Yannis Papakonstantinou. *Privacy in Database Publishing*. In ICDT, pages 230–245, 2005. 24

[Dillencourt92] Michael B. Dillencourt, Hanan Samet & Markku Tamminen. *A General Approach to Connected-Component Labelling for Arbitrary Image Representations*. J. ACM, vol. 39, no. 2, pages 253–280, 1992. 102

[Ding10] Xuan Ding, Lan Zhang, Zhiguo Wan & Ming Gu. *A Brief Survey on De-anonymization Attacks in Online Social Networks*. In CASoN, pages 611–615. IEEE Computer Society, 2010. 20

[Dinur03] Irit Dinur & Kobbi Nissim. *Revealing information while preserving privacy*. In PODS, pages 202–210. ACM, 2003. 12

[Doan12] AnHai Doan, Alon Y. Halevy & Zachary G. Ives. Principles of data integration. Morgan Kaufmann, 2012. 42

[Duerst05] M. Duerst & M. Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987, RFC Editor, January 2005. `http://www.rfc-editor.org/rfc/rfc3987.txt`. 28

[Dwork06a] Cynthia Dwork. *Differential Privacy*. In ICALP (2), volume 4052 of *LNCS*, pages 1–12. Springer, 2006. 12, 18

[Dwork06b] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov & Moni Naor. *Our Data, Ourselves: Privacy Via Distributed Noise Generation*. In EUROCRYPT, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006. 19

[Dwork06c] Cynthia Dwork, Frank McSherry, Kobbi Nissim & Adam D. Smith. *Calibrating Noise to Sensitivity in Private Data Analysis*. In TCC, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. 18

[Dwork08] Cynthia Dwork. *Differential Privacy: A Survey of Results*. In TAMC, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008. 19

[Dwork16] Cynthia Dwork & Guy N. Rothblum. *Concentrated Differential Privacy*. CoRR, vol. abs/1603.01887, 2016. 19

[Endris18] Kemele M. Endris, Zuhair Almhithawi, Ioanna Lytra, Maria-Esther Vidal & Sören Auer. *BOUNCER: Privacy-Aware Query Processing over Federations of RDF Datasets*. In DEXA (1), volume 11029 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2018. 24

[Fernández17] Javier D. Fernández, Wouter Beek, Miguel A. Martínez-Prieto & Mario Arias. *LOD-a-lot - A Queryable Dump of the LOD Cloud*.

In International Semantic Web Conference (2), volume 10588 of *Lecture Notes in Computer Science*, pages 75–83. Springer, 2017. 25

[Fung05] Benjamin C. M. Fung, Ke Wang & Philip S. Yu. *Top-Down Specialization for Information and Privacy Preservation*. In ICDE, pages 205–216. IEEE Computer Society, 2005. 15

[Fung10] Benjamin C. M. Fung, Ke Wang, Rui Chen & Philip S. Yu. *Privacy-preserving data publishing: A survey of recent developments*. ACM Comput. Surv., vol. 42, no. 4, pages 14:1–14:53, 2010. 12, 13

[Gehrke11] Johannes Gehrke, Edward Lui & Rafael Pass. *Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy*. In TCC, volume 6597 of *Lecture Notes in Computer Science*, pages 432–449. Springer, 2011. 23

[Geng15] Quan Geng, Peter Kairouz, Sewoong Oh & Pramod Viswanath. *The Staircase Mechanism in Differential Privacy*. J. Sel. Topics Signal Processing, vol. 9, no. 7, pages 1176–1184, 2015. 18

[Ghayyur18] Sameera Ghayyur, Yan Chen, Roberto Yus, Ashwin Machanavajjhala, Michael Hay, Gerome Miklau & Sharad Mehrotra. *IoT-Detective: Analyzing IoT Data Under Differential Privacy*. In SIGMOD Conference, pages 1725–1728. ACM, 2018. 19

[Goasdoué13] François Goasdoué, Ioana Manolescu & Alexandra Roatis. *Efficient Query Answering against Dynamic RDF Databases*. In EDBT, pages 299–310, 2013. 34

[Goldberger10] Jacob Goldberger & Tamir Tassa. *Efficient Anonymizations with Enhanced Utility*. Trans. Data Privacy, vol. 3, no. 2, pages 149–175, 2010. 99

[Grau08] Bernardo Cuenca Grau & Ian Horrocks. *Privacy-Preserving Query Answering in Logic-based Information Systems*. In ECAI, pages 40–44, 2008. 24

[Grau16] Bernardo Cuenca Grau & Egor V. Kostylev. *Logical Foundations of Privacy-Preserving Publishing of Linked Data*. In AAAI, pages 943–949. AAAI Press, 2016. ii, 24

[Grau19] Bernardo Cuenca Grau & Egor V. Kostylev. *Logical Foundations of Linked Data Anonymisation*. J. Artif. Intell. Res., vol. 64, pages 253–314, 2019. 24, 26, 46, 47, 48, 73, 96

[Gruber93] Thomas R Gruber. *A translation approach to portable ontology specifications*. Knowledge Acquisition, vol. 5, no. 2, pages 199–220, Elsevier, 1993. 33

[Gupta12] Anupam Gupta, Aaron Roth & Jonathan Ullman. *Iterative Constructions and Private Data Release*. In TCC, volume 7194 of *Lecture Notes in Computer Science*, pages 339–356. Springer, 2012. 23

[Gutiérrez04] Claudio Gutiérrez, Carlos A. Hurtado & Alberto O. Mendelzon. *Foundations of Semantic Web Databases*. In PODS, pages 95–106. ACM, 2004. 34

[Gutiérrez11] Claudio Gutiérrez, Carlos A. Hurtado, Alberto O. Mendelzon & Jorge Pérez. *Foundations of Semantic Web databases*. J. Comput. Syst. Sci., vol. 77, no. 3, pages 520–541, 2011. 36

[Hansen15] Peter Hansen, Carsten Lutz, Inanç Seylan & Frank Wolter. *Efficient Query Rewriting in the Description Logic EL and Beyond*. In IJCAI, pages 3034–3040. AAAI Press, 2015. 62

[Hardt12] Moritz Hardt, Katrina Ligett & Frank McSherry. *A Simple and Practical Algorithm for Differentially Private Data Release*. In NIPS, pages 2348–2356, 2012. 98

[Harris13] Steven Harris & Andy Seaborne. *SPARQL 1.1 Query Language*. W3C recommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/. 30, 41, 122

[Hay08] Michael Hay, Gerome Miklau, David D. Jensen, Donald F. Towsley & Philipp Weis. *Resisting structural re-identification in anonymized social networks*. PVLDB, vol. 1, no. 1, pages 102–114, 2008. 21, 22, 98

[Hay09] Michael Hay, Chao Li, Gerome Miklau & David D. Jensen. *Accurate Estimation of the Degree Distribution of Private Networks*. In ICDM, pages 169–178. IEEE Computer Society, 2009. 22, 101

[Hay10] Michael Hay, Vibhor Rastogi, Gerome Miklau & Dan Suciu. *Boosting the Accuracy of Differentially Private Histograms Through Consistency*. PVLDB, vol. 3, no. 1, pages 1021–1032, 2010. 23

[Hayes14] Patrick Hayes & Peter Patel-Schneider. *RDF 1.1 Semantics*. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/. 36

[He14] Xi He, Ashwin Machanavajjhala & Bolin Ding. *Blowfish privacy: tuning privacy-utility trade-offs using policies*. In SIGMOD Conference, pages 1447–1458. ACM, 2014. 13

[Heitmann17] Benjamin Heitmann, Felix Hermsen & Stefan Decker. *k–RDF-Neighbourhood Anonymity: Combining Structural and Attribute-based Anonymisation for Linked Data*. In PrivOn@ISWC, volume 1951. CEUR-WS.org, 2017. ii, 23

[Humbert14] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux & Amalio Telenti. *Reconciling Utility with Privacy in Genomics*. In WPES, pages 11–20. ACM, 2014. 12

[Iyengar02] Vijay S. Iyengar. *Transforming data to satisfy privacy constraints*. In KDD, pages 279–288. ACM, 2002. 15, 97, 99

[Ji14] Zhanglong Ji, Zachary Chase Lipton & Charles Elkan. *Differential Privacy and Machine Learning: a Survey and Review*. CoRR, vol. abs/1412.7584, 2014. 19

[Ji15] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal & Raheem A. Beyah. *On Your Social Network De-anonymizablity: Quantification and Large Scale Evaluation with Seed Knowledge*. In NDSS. The Internet Society, 2015. 21

[Ji16a] Shouling Ji, Weiqing Li, Mudhakar Srivatsa & Raheem A. Beyah. *Structural Data De-Anonymization: Theory and Practice*. IEEE/ACM Trans. Netw., vol. 24, no. 6, pages 3523–3536, 2016. 21

[Ji16b] Shouling Ji, Weiqing Li, Shukun Yang, Prateek Mittal & Raheem A. Beyah. *On the relative de-anonymizability of graph data: Quantification and evaluation*. In INFOCOM, pages 1–9. IEEE, 2016. 21

[Ji17] Shouling Ji, Prateek Mittal & Raheem A. Beyah. *Graph Data Anonymization, De-Anonymization Attacks, and De-Anonymizability Quantification: A Survey*. IEEE Communications Surveys and Tutorials, vol. 19, no. 2, pages 1305–1326, 2017. 20

[Jin10] Xin Jin, Nan Zhang & Gautam Das. *Algorithm-safe privacy-preserving data publishing*. In EDBT, volume 426 of *ACM International Conference Proceeding Series*, pages 633–644. ACM, 2010. 15

[Jonquet11] Clément Jonquet, Paea LePendu, Sean M. Falconer, Adrien Coulet, Natalya Fridman Noy, Mark A. Musen & Nigam H. Shah. *NCBO Resource Index: Ontology-based search and mining of biomedical resources*. J. Web Semant., vol. 9, no. 3, pages 316–324, 2011. 33

[Jr.05] Roberto J. Bayardo Jr. & Rakesh Agrawal. *Data Privacy through Optimal k-Anonymization*. In ICDE, pages 217–228. IEEE Computer Society, 2005. 15, 97, 99

[Karwa11] Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith & Grigory Yaroslavtsev. *Private Analysis of Graph Structure*. PVLDB, vol. 4, no. 11, pages 1146–1157, 2011. 23

[Kasiviswanathan13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova & Adam D. Smith. *Analyzing Graphs with Node Differential Privacy*. In TCC, volume 7785 of *Lecture Notes in Computer Science*, pages 457–476. Springer, 2013. 23

[Kayes17] Imrul Kayes & Adriana Iamnitchi. *Privacy and security in online social networks: A survey*. Online Social Networks and Media, vol. 3-4, pages 1–21, 2017. 20

[Kellogg14] Gregg Kellogg, Markus Lanthaler & Manu Sporny. *JSON-LD 1.0*. W3C recommendation, W3C, January 2014. http://www.w3.org/TR/2014/REC-json-ld-20140116/. 30

[Kifer06] Daniel Kifer & Johannes Gehrke. *Injecting utility into anonymized datasets*. In SIGMOD Conference, pages 217–228. ACM, 2006. 97, 99

[Kifer09] Daniel Kifer. *Attacks on privacy and deFinetti's theorem*. In SIGMOD Conference, pages 127–138. ACM, 2009. 16

[Kifer11] Daniel Kifer & Ashwin Machanavajjhala. *No free lunch in data privacy*. In SIGMOD Conference, pages 193–204. ACM, 2011. 13

[Kifer14] Daniel Kifer & Ashwin Machanavajjhala. *Pufferfish: A framework for mathematical privacy definitions*. ACM Trans. Database Syst., vol. 39, no. 1, pages 3:1–3:36, 2014. 13

[Kirrane17] Sabrina Kirrane, Alessandra Mileo & Stefan Decker. *Access control and the Resource Description Framework: A survey*. Semantic Web, vol. 8, no. 2, pages 311–352, 2017. 24

[Kirrane18] Sabrina Kirrane, Serena Villata & Mathieu d'Aquin. *Privacy, security and policies: A review of problems and solutions with semantic web technologies*. Semantic Web Journal, vol. 9, no. 2, pages 153–161, 2018. 23

[Korula14] Nitish Korula & Silvio Lattanzi. *An efficient reconciliation algorithm for social networks*. PVLDB, vol. 7, no. 5, pages 377–388, 2014. 21

[LeFevre05] Kristen LeFevre, David J. DeWitt & Raghu Ramakrishnan. *Incognito: Efficient Full-Domain K-Anonymity*. In SIGMOD Conference, pages 49–60. ACM, 2005. 96, 99

[LeFevre06a] Kristen LeFevre, David J. DeWitt & Raghu Ramakrishnan. *Mondrian Multidimensional K-Anonymity*. In ICDE, page 25. IEEE Computer Society, 2006. 15, 97

[LeFevre06b] Kristen LeFevre, David J. DeWitt & Raghu Ramakrishnan. *Workload-aware anonymization*. In KDD, pages 277–286. ACM, 2006. 97

[Li07] Ninghui Li, Tiancheng Li & Suresh Venkatasubramanian. *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity*. In ICDE, pages 106–115. IEEE Computer Society, 2007. ii, 16, 96

[Lin13] Bing-Rong Lin & Daniel Kifer. *Information preservation in statistical privacy and bayesian estimation of unattributed histograms*. In SIGMOD Conference, pages 677–688. ACM, 2013. 23, 101

[Liu08] Kun Liu & Evimaria Terzi. *Towards identity anonymization on graphs*. In SIGMOD Conference, pages 93–106. ACM, 2008. 22

[Liu17] Peng Liu, Yan Bai, Lie Wang & Xianxian Li. *Partial k-Anonymity for Privacy-Preserving Social Network Data Publishing*. International Journal of Software Engineering and Knowledge Engineering, vol. 27, no. 1, pages 71–90, 2017. 15

[Machanavajjhala07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke & Muthuramakrishnan Venkitasubramaniam. L-*diversity: Privacy beyond* k-*anonymity*. TKDD, vol. 1, no. 1, page 3, 2007. ii, 16, 96, 99

[Machanavajjhala08] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke & Lars Vilhuber. *Privacy: Theory meets Practice on the Map*. In ICDE, pages 277–286. IEEE Computer Society, 2008. 18

[Machanavajjhala09] Ashwin Machanavajjhala, Johannes Gehrke & Michaela Götz. *Data Publishing against Realistic Adversaries*. PVLDB, vol. 2, no. 1, pages 790–801, 2009. 17

[Martin07] David J. Martin, Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke & Joseph Y. Halpern. *Worst-Case Background Knowledge for Privacy-Preserving Data Publishing*. In ICDE, pages 126–135. IEEE Computer Society, 2007. 17

[McGuinness12] Deborah McGuinness, Jie Bao, Peter Patel-Schneider & Elisa Kendall. *OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)*. Rapport technique, W3C, December 2012. http://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/. 33

[McSherry07] Frank McSherry & Kunal Talwar. *Mechanism Design via Differential Privacy*. In FOCS, pages 94–103. IEEE Computer Society, 2007. 18

[McSherry09] Frank McSherry & Ilya Mironov. *Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders*. In KDD, pages 627–636. ACM, 2009. 19

[Miklau07] G. Miklau & D. Suciu. *A formal analysis of information disclosure in data exchange*. Journal of Computer and System Sciences, vol. 73, no. 3, pages 507–534, 2007. 24

[Millstein03] Todd D. Millstein, Alon Y. Halevy & Marc Friedman. *Query containment for data integration systems*. J. Comput. Syst. Sci., vol. 66, no. 1, pages 20–39, 2003. 42

[Mironov17] Ilya Mironov. *Rényi Differential Privacy*. In CSF, pages 263–275. IEEE Computer Society, 2017. 19

[Mishra06] Nina Mishra & Mark Sandler. *Privacy via pseudorandom sketches*. In PODS, pages 143–152. ACM, 2006. 14

[Mittal13] Prateek Mittal, Charalampos Papamanthou & Dawn Xiaodong Song. *Preserving Link Privacy in Social Network Based Systems*. In NDSS. The Internet Society, 2013. 21

[Mohammed11] Noman Mohammed, Rui Chen, Benjamin C. M. Fung & Philip S. Yu. *Differentially private data release for data mining*. In KDD, pages 493–501. ACM, 2011. 19

[Narayanan08] Arvind Narayanan & Vitaly Shmatikov. *Robust De-anonymization of Large Sparse Datasets*. In IEEE Symposium on Security and Privacy, pages 111–125. IEEE Computer Society, 2008. 4, 17, 19, 22

[Narayanan09] Arvind Narayanan & Vitaly Shmatikov. *De-anonymizing Social Networks*. In IEEE Symposium on Security and Privacy, pages 173–187. IEEE Computer Society, 2009. 21

[Nergiz07a] Mehmet Ercan Nergiz, Maurizio Atzori & Chris Clifton. *Hiding the presence of individuals from shared databases*. In SIGMOD Conference, pages 665–676. ACM, 2007. 17

[Nergiz07b] Mehmet Ercan Nergiz & Chris Clifton. *Thoughts on k-anonymization*. Data Knowl. Eng., vol. 63, no. 3, pages 622–645, 2007. 99

[Neto16] Ciro Baron Neto, Dimitris Kontokostas, Sebastian Hellmann, Kay Müller & Martin Brümmer. *Assessing Quantity and Quality of Links Between Link Data Datasets*. In LDOW@WWW, volume 1593 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. 103

[Nissim07] Kobbi Nissim, Sofya Raskhodnikova & Adam D. Smith. *Smooth sensitivity and sampling in private data analysis*. In STOC, pages 75–84. ACM, 2007. 22

[Nobari14] Sadegh Nobari, Panagiotis Karras, HweeHwa Pang & Stéphane Bressan. *L-opacity: Linkage-Aware Graph Anonymization*. In EDBT, pages 583–594. OpenProceedings.org, 2014. 22, 99, 101

[Oulmakhzoune12] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens & Stephane Morucci. *Privacy Policy Preferences Enforced by SPARQL Query Rewriting*. In ARES, pages 335–342. IEEE, 2012. 24

[Ozalp16] Ismet Ozalp, Mehmet Emre Gursoy, Mehmet Ercan Nergiz & Yücel Saygin. *Privacy-Preserving Publishing of Hierarchical Data*. ACM Trans. Priv. Secur., vol. 19, no. 3, pages 7:1–7:29, 2016. 15

[Pan06] Jeff Pan & Jeremy Carroll. *XML Schema Datatypes in RDF and OWL*. W3C note, W3C, March 2006. http://www.w3.org/TR/2006/NOTE-swbp-xsch-datatypes-20060314/. 28

[Pedarsani11] Pedram Pedarsani & Matthias Grossglauser. *On the privacy of anonymized networks*. In KDD, pages 1235–1243. ACM, 2011. 21

[Pedarsani13] Pedram Pedarsani, Daniel R. Figueiredo & Matthias Grossglauser. *A Bayesian method for matching two similar graphs without seeds*. In Allerton, pages 1598–1607. IEEE, 2013. 21, 22

[Pérez09] Jorge Pérez, Marcelo Arenas & Claudio Gutiérrez. *Semantics and complexity of SPARQL*. ACM Trans. Database Syst., vol. 34, no. 3, pages 16:1–16:45, 2009. 40

[Piao19] Chunhui Piao, Yajuan Shi, Jiaqi Yan, Changyou Zhang & Liping Liu. *Privacy-preserving governmental data publishing: A fog-computing-based differential privacy approach*. Future Generation Comp. Syst., vol. 90, pages 158–174, 2019. 19

[Polleres13] Axel Polleres, Alexandre Passant & Paula Gearon. *SPARQL 1.1 Update*. W3C recommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-update-20130321/. 31

[Prasser15] Fabian Prasser & Florian Kohlmayer. *Putting Statistical Disclosure Control into Practice: The ARX Data Anonymization Tool*. In Medical Data Privacy Handbook, pages 111–148. Springer, 2015. 96

[Prasser16] Fabian Prasser, Raffael Bild & Klaus A. Kuhn. *A Generic Method for Assessing the Quality of De-Identified Health Data*. In MIE, volume 228 of *Studies in Health Technology and Informatics*, pages 312–316. IOS Press, 2016. 99

[Proserpio12] Davide Proserpio, Sharon Goldberg & Frank McSherry. *A workflow for differentially-private graph synthesis*. In WOSN, pages 13–18. ACM, 2012. 23

[Radulovic15] Filip Radulovic, Raúl García-Castro & Asunción Gómez-Pérez. *Towards the Anonymisation of RDF Data*. In SEKE, pages 646–651. KSI Research Inc., 2015. 23, 98, 101

[Rajagopalan11] S. Raj Rajagopalan, Lalitha Sankar, Soheil Mohajer & H. Vincent Poor. *Smart Meter Privacy: A Utility-Privacy Framework*. CoRR, vol. abs/1108.2234, 2011. 12

[Raskhodnikova16] Sofya Raskhodnikova & Adam D. Smith. *Lipschitz Extensions for Node-Private Graph Statistics and the Generalized Exponential Mechanism*. In FOCS, pages 495–504. IEEE Computer Society, 2016. 22

[Rastogi07] Vibhor Rastogi, Sungho Hong & Dan Suciu. *The Boundary Between Privacy and Utility in Data Publishing*. In VLDB, pages 531–542. ACM, 2007. 5, 13, 18, 98

[Rastogi09] Vibhor Rastogi, Michael Hay, Gerome Miklau & Dan Suciu. *Relationship privacy: output perturbation for queries with joins*. In PODS, pages 107–116. ACM, 2009. 23

[Rastogi10] Vibhor Rastogi & Suman Nath. *Differentially private aggregation of distributed time-series with transformation and encryption*. In SIGMOD Conference, pages 735–746. ACM, 2010. 20

[Roth10] Aaron Roth & Tim Roughgarden. *Interactive privacy via the median mechanism*. In STOC, pages 765–774. ACM, 2010. 14

[Sala11] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng & Ben Y. Zhao. *Sharing graphs using differentially private graph models*.

In Internet Measurement Conference, pages 81–98. ACM, 2011. 23

[Samarati01] Pierangela Samarati. *Protecting Respondents' Identities in Microdata Release*. IEEE Trans. Knowl. Data Eng., vol. 13, no. 6, pages 1010–1027, 2001. 14, 96

[Schreiber14] Guus Schreiber & Fabien Gandon. *RDF 1.1 XML Syntax*. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/. 30

[Seaborne08] Andy Seaborne & Eric Prud'hommeaux. *SPARQL Query Language for RDF*. W3C recommendation, W3C, January 2008. 3

[Seaborne14] Andy Seaborne & Gavin Carothers. *RDF 1.1 N-Triples*. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-n-triples-20140225/. 30

[Sei19] Yuichi Sei, Hiroshi Okumura, Takao Takenouchi & Akihiko Ohsuga. *Anonymization of Sensitive Quasi-Identifiers for l-Diversity and t-Closeness*. IEEE Trans. Dependable Sec. Comput., vol. 16, no. 4, pages 580–593, 2019. 17

[Soria-Comas14] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez & Sergio Martínez. *Enhancing data utility in differential privacy via microaggregation-based $$k$$ k -anonymity*. VLDB J., vol. 23, no. 5, pages 771–794, 2014. 20

[Soria-Comas15] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez & Sergio Martínez. *t-Closeness through Microaggregation: Strict Privacy with Enhanced Utility Preservation*. IEEE Trans. Knowl. Data Eng., vol. 27, no. 11, pages 3098–3110, 2015. 20

[Squicciarini15] Anna Cinzia Squicciarini & Ting Yu. *Privacy and Access Control: How are These Two concepts Related?* In SACMAT, pages 197–198. ACM, 2015. 13

[Srivastava13] Biplav Srivastava, Ghislain Auguste Atemezing, Bernadette Hyland & Michael Pendleton. *Linked Data Glossary*. W3C note, W3C, June 2013. 3

[Sweeney02a] Latanya Sweeney. *Achieving k-Anonymity Privacy Protection Using Generalization and Suppression*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pages 571–588, 2002. 14, 15, 96, 99, 101

[Sweeney02b] Latanya Sweeney. *k-Anonymity: A Model for Protecting Privacy*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pages 557–570, 2002. ii, xvii, 4, 12, 14, 72, 96

[Turn72] Rein Turn & Norman Z. Shapiro. *Privacy and security in databank systems: measures of effectiveness, costs, and protector-intruder interactions*. In AFIPS Fall Joint Computing Conference (1), volume 41 of *AFIPS Conference Proceedings*, pages 435–444. AFIPS / ACM / Thomson Book Company, Washington D.C., 1972. 12

[Turn82] Rein Turn. *Privacy Protection in the 1980s*. In IEEE Symposium on Security and Privacy, pages 86–89. IEEE Computer Society, 1982. 12

[Turn90] Rein Turn. *Information Privacy Issues for the 1990s*. In IEEE Symposium on Security and Privacy, pages 394–400. IEEE Computer Society, 1990. 12

[Vatsalan17] Dinusha Vatsalan, Ziad Sehili, Peter Christen & Erhard Rahm. *Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges*. In Handbook of Big Data Technologies, pages 851–895. Springer, 2017. 24

[Villata11] Serena Villata, Nicolas Delaforge, Fabien Gandon & Amelie Gyrard. *An Access Control Model for Linked Data*. In OTM Workshops, volume 7046 of *LNCS*, pages 454–463. Springer, 2011. 24

[Wang04] Ke Wang, Philip S. Yu & Sourav Chakraborty. *Bottom-Up Generalization: A Data Mining Solution to Privacy Protection*. In ICDM, pages 249–256. IEEE Computer Society, 2004. 15

[Wang05] Ke Wang, Benjamin C. M. Fung & Philip S. Yu. *Template-Based Privacy Preservation in Classification Problems*. In ICDM, pages 466–473. IEEE Computer Society, 2005. 97

[Wang16] Yue Wang, Xintao Wu & Donghui Hu. *Using Randomized Response for Differential Privacy Preserving Data Collection*. In EDBT/ICDT Workshops, volume 1558 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. 18

[Wang19] Jinyan Wang, Kai Du, Xudong Luo & Xianxian Li. *Two privacy-preserving approaches for data publishing with identity reservation*. Knowl. Inf. Syst., vol. 60, no. 2, pages 1039–1080, 2019. 15

[Warner65] Stanley L Warner. *Randomized response: A survey technique for eliminating evasive answer bias*. Journal of the American Statistical Association, vol. 60, no. 309, pages 63–69, Taylor & Francis, 1965. 18

[Watts98] Duncan J Watts & Steven H Strogatz. *Collective dynamics of 'small-world' networks*. Nature, vol. 393, no. 6684, page 440, Nature Publishing Group, 1998. 99

[Wong07] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang & Jian Pei. *Minimality Attack in Privacy Preserving Data Publishing*. In VLDB, pages 543–554. ACM, 2007. 15

[Wood14] David Wood, Markus Lanthaler & Richard Cyganiak. *RDF 1.1 Concepts and Abstract Syntax*. W3C recommendation, W3C, February 2014. 3, 28

[Wu10] Xintao Wu, Xiaowei Ying, Kun Liu & Lei Chen. *A Survey of Privacy-Preservation of Graphs and Social Networks*. In Managing and Mining Graph Data, volume 40 of *Advances in Database Systems*, pages 421–453. Springer, 2010. 20

[Wu12] Zhe Wu, Ian Horrocks, Boris Motik, Bernardo Cuenca Grau & Achille Fokoue. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C recommendation, W3C, December 2012. http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/. 33

[Xiao06] Xiaokui Xiao & Yufei Tao. *Anatomy: Simple and Effective Privacy Preservation*. In VLDB, pages 139–150. ACM, 2006. 16

[Xiao10] Xiaokui Xiao, Yufei Tao & Nick Koudas. *Transparent anonymization: Thwarting adversaries who know the algorithm*. ACM Trans. Database Syst., vol. 35, no. 2, pages 8:1–8:48, 2010. 15

[Xiong14] Ping Xiong, TQ Zhu & XF Wang. *A survey on differential privacy and applications*. Jisuanji Xuebao/Chinese Journal of Computers, vol. 37, no. 1, pages 101–122, Kexue Chubanshe/Science Press, 2014. 19

[Xu13] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu & Marianne Winslett. *Differentially private histogram publication*. VLDB J., vol. 22, no. 6, pages 797–822, 2013. 20

[Ying08] Xiaowei Ying & Xintao Wu. *Randomizing Social Networks: a Spectrum Preserving Approach*. In SDM, pages 739–750. SIAM, 2008. 21

[Yuan13] Mingxuan Yuan, Lei Chen, Philip S. Yu & Ting Yu. *Protecting Sensitive Labels in Social Network Data Anonymization*. IEEE Trans. Knowl. Data Eng., vol. 25, no. 3, pages 633–647, 2013. 99

[Zhang07] Lei Zhang, Sushil Jajodia & Alexander Brodsky. *Information disclosure under realistic assumptions: privacy versus optimality*. In ACM Conference on Computer and Communications Security, pages 573–583. ACM, 2007. 15

[Zheleva07] Elena Zheleva & Lise Getoor. *Preserving the Privacy of Sensitive Relationships in Graph Data*. In PinKDD, volume 4890 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2007. 98

[Zheleva11] Elena Zheleva & Lise Getoor. *Privacy in Social Networks: A Survey*. In Social Network Data Analytics, pages 277–306. Springer, 2011. 20

[Zhou08a] Bin Zhou & Jian Pei. *Preserving Privacy in Social Networks Against Neighborhood Attacks*. In ICDE, pages 506–515. IEEE Computer Society, 2008. 22

[Zhou08b] Bin Zhou, Jian Pei & Wo-Shun Luk. *A brief survey on anonymization techniques for privacy preserving publishing of social network data*. SIGKDD Explorations, vol. 10, no. 2, pages 12–22, 2008. 20

[Zhou19] Lu Zhou, Le Yu, Suguo Du, Haojin Zhu & Cailian Chen. *Achieving Differentially Private Location Privacy in Edge-Assistant Connected Vehicles*. IEEE Internet of Things Journal, vol. 6, no. 3, pages 4472–4481, 2019. 19

[Zhu17] Tianqing Zhu, Gang Li, Wanlei Zhou & Philip S. Yu. *Differentially Private Data Publishing and Analysis: A Survey*. IEEE Trans. Knowl. Data Eng., vol. 29, no. 8, pages 1619–1638, 2017. 19

[Zou09] Lei Zou, Lei Chen & M. Tamer Özsu. *K-Automorphism: A General Framework For Privacy Preserving Network Publication*. PVLDB, vol. 2, no. 1, pages 946–957, 2009. 22