Chase Anonymisation: Privacy-Preserving Knowledge Graphs with Logical Reasoning

Luigi Bellomarini
Banca d'Italia
luigi.bellomarini@bancaditalia.it

Costanza Catalano
Banca d'Italia
costanza.catalano@bancaditalia.it

Andrea Coletta
Banca d'Italia
andrea.coletta@bancaditalia.it

Michela Iezzi
Banca d'Italia
michela.iezzi@bancaditalia.it

Pierangela Samarati
Università degli Studi di Milano
pierangela.samarati@unimi.it

***Abstract*—We propose a novel framework to enable Knowledge Graphs (KGs) sharing while ensuring that information that should remain private is not directly released nor indirectly exposed via derived knowledge, maintaining at the same time the embedded knowledge of the KGs to support business downstream tasks. Our approach produces a privacy-preserving KG as an augmentation of the input one via controlled addition of nodes and edges as well as re-labeling of nodes and perturbation of weights. We introduce a novel privacy measure for KGs, which considers derived knowledge, a new utility metric that captures the business semantics we want to preserve, and propose two novel anonymisation algorithms. Our extensive experimental evaluation, with both synthetic graphs and real-world datasets, confirms the effectiveness of our approach.**

***Index Terms*—Reasoning, knowledge graph, anonymisation, isomorphism.**

I. Introduction

Knowledge Graphs (KGs) are gaining increasing scientific interest [16, 27], as also witnessed by a broad adoption in many industrial domains, from healthcare to biotechnology, from logistics to finance [8, 36, 40, 41, 42]. While we are still lacking a consolidated and shared definition of KGs, a distinguishing characteristic is the presence of some form of intensional knowledge, an encoding of the business experience that can be leveraged to generate new—derived—nodes and edges, through a *reasoning* process [4]. Logic-based approaches to KGs see nodes and edges as facts of a database (i.e., the *extensional component*) and encode the business knowledge through a logic program (i.e., the *intensional component*). The semantic of such a logic program is usually specified through the CHASE procedure [37], which applies the rules to the database, as long as they produce new facts (i.e., the *derived extensional component*). Consider for example Figure 1. It shows a company ownership KG: the nodes represent companies, and an edge from x to y, with weight w, indicates that x owns a fraction w of the shares of y. The red edges are part of the derived extensional component,

The work of Pierangela Samarati was supported in part by the EC under project GLACIATION (101070141) and by project SERICS (PE00000014) under the MUR NRRP funded by the EU - NGEU. The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Banca d'Italia, EU or the Italian MUR.

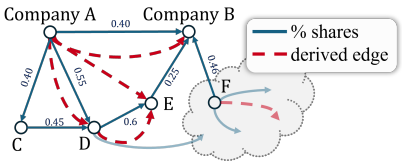


Fig. 1: An ownership Knowledge Graph.

obtained by putting into action the business notion of *company control*: *a company x controls y if it owns more than 50% of y's shares, or if it controls a set of companies that jointly, and possibly with x itself, own the majority of y.*

Sharing Knowledge Graphs. Financial organisations, central banks, economic research entities, and national supervision authorities need to share KGs to address crucial business tasks, such as banking supervision, support to policymaking, anti-money laundering, and economic research. At the same time, the identity of the involved subjects and their relationships, be they individuals or companies, may not be disclosed [34, 38] while preserving the utility of the released data for downstream tasks [2]. This represents a severe hurdle, since classical approaches such as *differential privacy* (DP) [11, 30] and *structural anonymisation* (e.g., k-anonymity [46]) are unsuitable to handle KGs, where the intensional knowledge plays a crucial role and opens new avenues to privacy disclosure. DP approaches are not applicable, as the injected noise undermines the effectiveness of downstream tasks, for example hampering the performance of machine learning models, especially those relying on multiple correlated graph statistics [25]. Structural anonymisation techniques are more promising, but still insufficient for our goals. The methods of this family usually modify the topology of the graph, so that it exhibits at least k "similar" (e.g., isomorphic) structures with respect to the adversary knowledge [21, 47]. Their typical setting is the so-called *neighbourhood Attack Graph* (NAG), also known as a *subgraph-based attack* [47]: the attacker tries to use some knowledge about the neighbourhood of a target node to re-map it in the original graph and re-identify it. The graph is modified to create, for each substructure of a given size, many undistinguishable isomorphic clones.

Unfortunately, these techniques fail when the attacker has information on some derived knowledge and, in the worst case, they have full visibility of the business rules. More in general, anonymising KGs presents various unique challenges, originating from the presence of derived knowledge, which we will discuss in detail: *(i)* if the attacker knows the logical derivation steps (chase graph), even isomorphic structures in the graph can be distinguished, since, intuitively, their elements have a different "lineage"; as a consequence, anonymising a KG does not trivially consist of pre-materializing the derived knowledge and adopting existing techniques on the obtained graph *(ii)* in preserving the "meaningfulness" of the KG, the anonymisation process should account for the accuracy of business tasks *(iii)* financial applications have high-levels privacy requirements, which cover not only the names of the entities, but also their "role" and relationships in the network.

In this work, we propose a *reasoning-aware anonymisation approach* for Knowledge Graphs, where the derived knowledge is a first-class citizen. In particular, we introduce the (k, x)*-chase anonymisation*, a novel structural anonymisation technique producing KGs that are NAR-resistant (resistant to NAG + Reasoning). Our technique modifies the KG so that each induced subgraph of size at most x has other k *chase-isomorphic* structures, i.e. structures that are also indistinguishable with respect to the knowledge produced by the reasoning process (challenge *(i)*). Our structurally indistinguishable subgraphs exhibit different node labels, in-degrees, and out-degrees, along with perturbed edge weights (challenge *(iii)*). In detail, this paper contributes the following.

- The definition of a (k, x)**-chase anonymisation** for a KG, resistant to attacks where there is information of the reasoning rules or derived knowledge.
- A novel **semantic utility metric**, which accounts for the usefulness of the anonymised KG for specific downstream tasks (challenge *(ii)*). Our anonymisation procedure maximises this metric and guarantees that the anonymised KG resembles the original one for the tasks of interest.
- Two new **anonymisation algorithms**, namely, KLONE and KGUARD, able to reach a (k, x)-chase anonymisation while maximising the semantic utility metrics. The former is our first technique, which clones and differentiates the graph structures k times, regardless of the size x of the attack. The latter is a second approach that minimises the number of modifications by exploiting already existing structures.
- An **extensive experimental evaluation** on well-known network models and various real-world datasets, including the ownership KG of the Italian Central Bank [36].

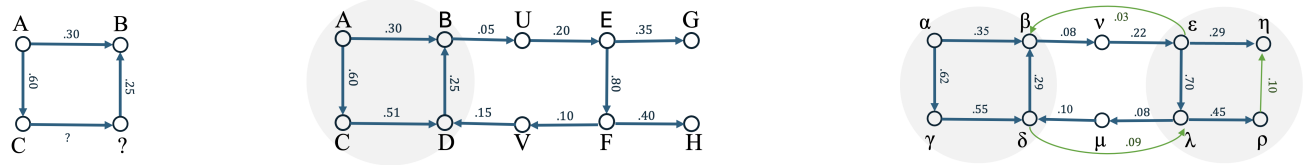
Overview. Section II: problem definition and sketch of our approach. Section III: background. Section IV: adversary attacks. Section V: definition of chase anonymisation. Sections VI and VII: anonymisation algorithms. Section VIII: experimental evaluation. Section IX: related work. Section X: conclusions.

II. Problem definition and sketch of the approach

We consider the problem of releasing a privacy-preserving version of a Knowledge Graph protecting the actual identities of the entities (nodes) in the graph and the relationships between them, even against attackers that have knowledge of portions of the original KG and reasoning rules on it or some derived knowledge. At the same time we aim at maintaining the utility of the privacy-preserving KG for downstream tasks for which it is released. As running example, we consider the KG in Figure 2b, where edges between nodes represent company ownership, labels on the edges represent the share of such ownership. We consider the set of rules formalised in Vadalog [3] in Table I. A first set of rules defines company control, namely: a company x controls itself (σ_1) and any other company z for which it collectively has more than 50% either directly or through any company it controls, summing up all the edge share (σ_2). A second set of rules defines reachability, meaning direct (σ_3) or indirect (σ_4) existence of a path from one node to the other; we will elaborate on queries when discussing utility and experimental results. Classical structural anonymity would protect the KG by aliasing the identity of the nodes, perturbing edge labels, and possibly adding synthetic edges to ensure, for any subgraphs of KG of up to a given size x (adversary's strength), the existence of at least k (where k is the privacy guarantee) isomorphic subgraphs. Structural anonymity protects against attackers knowing (from external knowledge) portions of the original graph and aiming at remapping it in the released KG to discover additional information. For instance, the privacy-preserving KG in Figure 2c protects against an attacker knowing Neighbourhood Attack Graphs (NAGs) comprising up to four nodes with k = 2. Indeed, the NAG in Figure 2a has at least 2 isomorphic subgraphs (greyed) in such KG.

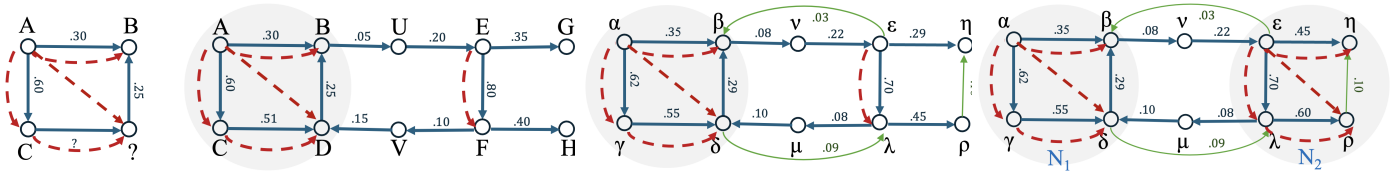
However, structural anonymity falls short when the attacker has information on the reasoning rules producing derived knowledge from KG. In fact, knowledge of the rules and hence of the derived edge allows the attacker to tell apart isomorphic subgraphs. This is illustrated in Figure 3a where the knowledge derived from the application of the rules (red dotted edges) enables distinguishing between the two subgraphs that were structural isomorphic in Figure 2c, thus enabling the exact remapping of the NAR in the sanitized KG.

Our chase anonymisation aims at guaranteeing privacy protection against such NAR attacks, in which attackers have knowledge not only of a subgraph of the ground extensional component of the KG (NAG), but also of the set of reasoning rules. Ensuring such protection requires operating on the KG so to ensure isomorphism not only with respect to the structure but also with respect to the additional edges (derived knowledge). Even more, it requires ensuring indistinguishability with respect to the sequences of facts (new knowledge) produced by the application of the rules to the graph, i.e., with respect to their *chase graphs*. Given a KG and a set of rules, the chase graph is a graph whose nodes represent the (intensional or derived) facts and edges connects facts to other facts which they contributed to derive; a label on the edge reports the rules enabling such derivation. For instance, Figure 4b reports the chase graphs of the left grey subgraph in Figure 3c. The anonymized graph in Figure 3d satisfies



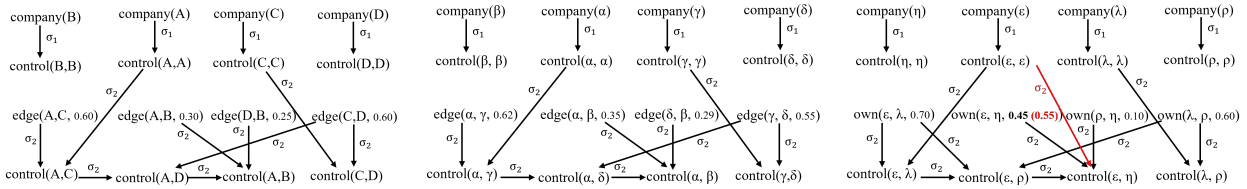
(a) NAG (subgraph). (b) Real Knowledge Graph. (c) Structural anonymity.

Fig. 2: (a) Attacker's knowledge: an employee of C is aware that company A owns 60% of their company's shares and 30% of the shares of a third company B; they also know that C is acquiring the majority of the shares of an unknown company (?), which in turn holds 25% of shares of B. (b) The NAG matches a uniquely identifiable structure within the real KG (grey shaded circle), revealing sensitive information. (c) Traditional structural anonymity prevents re-identification by introducing isomorphic structures, using additional edges (green edges).



(a) NAR. (b) Real KG with derived edges. (c) Failure of structural anonymity. (d) Chase anonymisation.

Fig. 3: Anonymisation of KGs with reasoning. (a) The employee of C has full knowledge of the control rules, maybe grasped from a thoughtful reading of regulations, discovering all the control relationships and underlying facts that generate them. (b) The NAR matches a uniquely identifiable structure within the real KG. (c) Traditional structural anonymity fails, as the use of reasoning reveals the unique structure of the NAR within the graph (grey shaded circle). (d) Our *chase anonymisation* addresses this limitation by explicitly considering the reasoning into the anonymisation procedure.



(a) $V = \{A, B, C, D\}$. (b) $V = \{\alpha, \beta, \gamma, \delta\}$. (c) $V = \{\varepsilon, \eta, \lambda, \rho\}$.

Fig. 4: (a) Chase graph of the NAR of Figure 3a; question marks are substituted by a placeholder "D" for the node and by random number > 0.5 for the edge weight. (b) Chase graph of the subgraph N_1 of Figure 3d (c) Chase graph of the subgraph N_2 of Figure 3d (black edges). The red edge indicates how the chase graph changes if the weight of (ε, η) is set to 0.55. Subgraphs are induced by the vertex set V.

the chase anonymisation as the two grey subgraphs N_1 and N_2 in it, not only have the same derived knowledge, but also the same (apart from renaming) chase graphs (Figures 4b and 4c, only black edges). However, had the weight between ε and η in N_2 been set to 0.55 instead of 0.45, while the two subgraphs remain isomorphic, their chase graphs would have differed, the one of N_2 having one additional edge (red edge in Figure 4c). In our approach we develop two algorithms that, for each induced subgraphs of size x, guarantee the presence of other k chase-isomorphic subgraphs within the anonymised KG. Moreover, labels and in- and out-degrees of the vertices are different between isomorphic copies, ensuring the privacy of entities and their relations. Also edge weights are perturbed with respect to the original KG preventing their use for re-identification. Lastly, we optimise the anonymisation process (in the weight perturbation and weight assignment to the newly introduced synthetic edges) by defining a set of business tasks on which we want the anonymised KG to be as close as possible to the original one. The business tasks are modeled as conjunctive queries (as the ones in Table I, for its semantics see Section III), and a Jaccard-based similarity index is set up to measure the difference between the answers returned by the anonymised and original KG, which is dynamically optimised in the proposed algorithms. For its formal definition and results we refer the reader, respectively, to Section V and Section VIII. We also consider a lighter attack where the attacker has information on some derived edges but not the set of reasoning rules. In this case, we show that the anonymisation algorithms can be simplified to make the anonymised graph resistant to these types of attacks (but vulnerable to NAR).

III. Basic concepts

Notation and graph concepts. We set $[n] := \{1, 2, \ldots, n\}$. The restriction of a function $f : B \to C$ to a subset $A \subseteq B$ is denoted by $f_{|A}$; the identity function on a set A is indicated by id_A. A (labelled weighted directed) graph is a tuple $G = (V, E, L, \omega, l)$ where V is the set of vertices, $E \subseteq V \times V$ is the set of edges, L is the set of labels, $\omega : E \to [0, 1]$ is the edge weight function and $l : V \to L$ is the vertex label function. Selfloops, i.e. edges of type (v, v), are allowed. The cardinality $|V|$ of the vertex set is indicated by n. Given $v \in V$, we set $N_G^-(v) := \{u \in V : (u, v) \in E\}$ and $N_G^+(v) := \{u \in V : (v, u) \in E\}$. The in-/out-degree of a vertex v are defined, respectively, as $d_G^-(v) := |N_G^-(v)|$ and

$d^+_G(v) := |N^+_G(v)|$. When clear from the context, we may omit the subscript G. A directed graph is *weakly connected* if, by replacing all of its directed edges with undirected ones, there exists a path connecting any pair of vertices [1]. We write $V(G)$, $E(G)$, $L(G)$, ω_G and l_G to denote, respectively, the vertex set, the edge set, the label set, the weight function, and the label function of a graph G. Given $X \subseteq V$, the subgraph of G *induced* by X is given by $G[X] := (X, E_X, L, \omega_{|_{E_X}}, l_{|_X})$, where $E_X = \{(u,v) \in E : u,v \in X\}$. Two graphs G and G' are *isomorphic* through the *isomorphism* $\psi : V(G) \to V(G')$ if ψ is bijective and $(u,v) \in E(G)$ iff $(\psi(u), \psi(v)) \in E(G')$.

Relational foundations and Knowledge Graphs. Let $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ be disjoint countably infinite sets of *constants*, *(labelled) nulls* and *variables*, respectively. They are also known as *terms*. A *(relational) schema* $\mathbf{S}$ is a finite set of predicates with associated arities. An *atom* is an expression $R(\bar{v})$, where $R \in \mathbf{S}$ is of arity $n \geq 0$ and $\bar{v}$ is an n-tuple of terms. A *database* D over $\mathbf{S}$ associates to each relation symbol in $\mathbf{S}$ a relation of the respective arity over the domain of constants and nulls. We denote by $dom(D)$ the set of constants in D. Relation members are called *tuples* or *facts*. Datalog$^\pm$ rules [7] are first-order implications $\forall \mathbf{x}\ \boldsymbol{\phi}(\mathbf{x}) \to \exists \mathbf{z}\ \boldsymbol{\psi}(\mathbf{y}, \mathbf{z})$, where $\boldsymbol{\phi}(\mathbf{x})$ (the *body*) and $\boldsymbol{\psi}(\mathbf{y}, \mathbf{z})$ (the *head*) are conjunctions of atoms over $\mathbf{S}$ and boldface variables denote vectors of variables, with $\mathbf{y} \subseteq \mathbf{x}$. We write these existential rules as $\boldsymbol{\phi}(\mathbf{x}) \to \exists \mathbf{z}\ \boldsymbol{\psi}(\mathbf{y}, \mathbf{z})$, using commas to denote conjunction of atoms in $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\psi}(\mathbf{y}, \mathbf{z})$. The semantics of a set of Datalog$^\pm$ rules Σ applied to a database D, can be defined operationally through the chase procedure [37]. The chase iteratively expands the database D with new facts, possibly containing labeled nulls, into a database $chase(D, \Sigma)$ by iteratively applying the rules Σ to it until a fixed point is reached. We adopt the Vadalog [3] syntax, a restriction of Datalog$^\pm$ that incorporates all the mentioned features while assuring convergence of the process. The *chase graph* $\mathcal{G}(D, \Sigma)$ is a directed graph having nodes labelled after facts from $chase(D, \Sigma)$ and having an edge from a node a to a node b labeled by σ if the fact b is obtained from a and possibly from other facts by the application of rule $\sigma \in \Sigma$ (a chase step). We consider databases that can be represented in terms of a weighted labelled directed graph G. A Knowledge Graph (KG) is a pair (G, Σ) with G a weighted labelled directed graph (*ground extensional component*) and Σ is a finite set of Vadalog rules (*intensional component*).[1] The new facts produced by the application of Σ on G are mapped into new edges for G (*derived extensional component*), also called *derived* edges. The set of derived edges is denoted by $\mathcal{D}_{G,\Sigma}$ and no weight is assigned to them. In Figure 1, $\mathcal{D}_{G,\Sigma}$ is the set of red edges, derived from the application of the control rules $\Sigma = \{\sigma_1, \sigma_2\}$ of Table I. We require Σ to be such that the facts produced on a smaller instance of a database D to be facts also produced when performed on the entire D. This is the case, e.g., of the control and reachability rules in Table I but does not necessarily hold true. Formally, given a KG (G, Σ), Σ is said to be *monotonic* on G if for all $X \subseteq Y \subseteq V$, we have that $\mathcal{G}(G[X], \Sigma) \subseteq \mathcal{G}(G[Y], \Sigma)$. Without loss of generality, in this paper we consider KGs with a weakly connected ground extensional component; general cases can be dealt with by applying our method to each weakly connected component of the KG.

[1] In this paper we focus on KGs having edge weights in [0,1] as our industrial application is of this type. We note however that our approach can be extended to a more general case of labelled edges, assuming the specification of an ontology capturing labels and distance between them to enable label perturbation while maintaining utility. Specifically, Algorithms 1 (Section VI) and 3 (Section VII) can be adapted to this general case by substituting the function WeightNoising with any other function that perturbs the edge's labels of the KG while maintaining high utility.

TABLE I: Reasoning rules and queries. The term $\text{edge}(x, y, w)$ represents an edge from entity x to y of weight w.

Reasoning rules: company control
σ_1 : $\text{company}(x) \to \text{control}(x, x)$
σ_2 : $\text{control}(x, y), \text{edge}(y, z, w), v = \text{sum}(w), v > 0.5 \to \text{control}(x, z)$
Reasoning rules: reachability
σ_3 : $\text{edge}(x, y, w), x \neq y, w > 0 \to \text{reach}(x, y)$
σ_4 : $\text{reach}(x, z), \text{edge}(z, y, w), x \neq y \neq z, w > 0 \to \text{reach}(x, y)$
Queries
Q_1 : $\text{reach-03}(x, y) \leftarrow \text{edge}(x, y, w), x \neq y, w \geq 0.3$
Q_2 : $\text{reach}_{\rightleftarrows}(x, y) \leftarrow \text{edge}(x, y, w), \text{edge}(y, x, o), x \neq y, w > o$
Q_3 : $\text{2-edge}(x) \leftarrow \text{edge}(x, y, w), x \neq y, k = \text{sum}(1, \langle y \rangle), k \geq 2$
Q_4 : $\text{2}q\text{-edge}(x) \leftarrow \text{edge}(x, y, w), x \neq y, k = \text{sum}(1, \langle y \rangle), w > q, k \geq 2$
Q_5 : $\text{direct-control}(x, y) \leftarrow \text{edge}(x, y, w), x \neq y, w > 0.5$
Q_6 : $\text{chain-control}(x, y) \leftarrow \text{control}(x, y), \text{edge}(x, t, \_), \text{edge}(t, z, \_), \text{edge}(z, y, \_), x \neq y \neq t \neq z.$
Q_7 : $\text{ultimate-control}(x, y) \leftarrow \text{control}(x, y), \neg\text{control}(\_, x)$

Queries. We express business tasks on KGs as queries, like the one in Table I. A *conjunctive query* Q over a schema $\mathbf{S}$ of a database D is an implication $q(\mathbf{x}) \leftarrow \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$, where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{S}$, q is an n-ary predicate, and $\mathbf{x}$ and $\mathbf{y}$ are vectors of terms. In the presence of Σ, it is evaluated as a set of tuples as follows: $Q(D) = \{t \in dom(D)^n : q(t) \in chase(D, \Sigma)\}$.

IV. Adversary attacks

Our adversary model extends the classical assumption of adversaries knowing a portion of the knowledge graph, called Neighbourhood Attack Graph (NAG), by providing also protection against adversaries knowing the reasoning rules or some derived knowledge. We consider two kinds of adversary attacks. The first attack, denoted as NAR (for NAG + Reasoning) models the worst case scenario in which the attacker knows a NAG as well as the Reasoning rules. The second attack, denoted as NAD (for NAG + Derived), models the case where the attacker knows a NAG and some derived edges.

NAR attack. The first scenario we consider is where the adversary knows:

1) the topology of a weakly connected subgraph of the KG;
2) the label of at least one vertex of the subgraph;
3) the weight of at least one edge of the subgraph;
4) the set Σ of reasoning rules.

Given X a set of x vertices, the maximal knowledge that an attacker may have in terms of items (1)–(4) is to have information on the full topology of the induced subgraph $G[X]$, together with all its vertex labels and edge weights, of the set Σ of reasoning rules, which implies also the knowledge

of all the derived edges of the subgraph. We are going to formalise the attack in this worst-case scenario.

Definition 4.1 (NAR): Given (G, Σ) a KG, the information possessed by the adversary is a pair $M = (G[X], \Sigma)$, where X is a subset of vertices of G and $G[X]$ is weakly connected. We call M the *neighbourhood Attack graph with Reasoning (NAR)* of size $x = |X|$.

The knowledge of the NAR $(G[X], \Sigma)$ implies the knowledge of the chase graph $\mathcal{G}(G[X], \Sigma)$.

NAD attack. We consider a second weaker scenario in which the adversary knows items (1)–(3) as in the previous NAR attack model, but instead of having information on the full set of reasoning rules Σ, they have knowledge of

(4') some derived edges between the vertices of the subgraph.

In this scenario, the adversary has knowledge only of the (possibly partial) results of the application of the rules of Σ to the subgraph, i.e. the derived edges. Given X a set of x vertices, the maximal knowledge that an attacker may have in terms of items (1)–(4') is to have information on the full topology of the induced subgraph $G[X]$, together with all its vertex labels, edge weights and derived edges $D_{G[X],\Sigma}$. We formalise the attack in this worst-case scenario.

Definition 4.2 (NAD): Given (G, Σ) a KG, the information possessed by the adversary is a pair $N = (G[X], \mathcal{D}_{G[X],\Sigma})$, where X is a subset of vertices of G and $G[X]$ is weakly connected. We call N the *neighbourhood Attack graph with Derived edges (NAD)* of size $x = |X|$.

We protect the KG from a NAR (and hence NAD) attack of size x, by guaranteeing that in the anonymised KG each subgraph of size at most x has at least other $k-1$ subgraphs that are chase-isomorphic, i.e. that are indistinguishable not only in their topology (including derived edges), but also in the sequence of facts generated by the reasoning. In this way, the adversary cannot re-identify the structure, as it has no information on how to discern between them. Moreover, we guarantee that the corresponding vertices in the isomorphism have all different labels, in-degree and our-degree, thus ensuring that such information cannot be deduced by the adversary. Like in structural anonymity, our protection degree k models the privacy guarantee (uncertainty of re-mapping) against the adversary.

V. (k, x)-CHASE ANONYMISATION

In this section, we illustrate our (k, x)-chase anonymisation to protect against the attacks previously described. We first model protection against the most powerful NAR attack and then show how it can be simplified if only NAD attacks are to be considered.

Protection from NAR attacks. Our approach to protect against NAR attacks relies on the following new concept of chase-isomorphism.

Definition 5.1 (Chase-isomorphism): Let (G, Σ) and (G', Σ) be two KGs with the same set of reasoning rules and vertex sets V and V' respectively. We say that they are *chase-isomorphic* ($G \cong_\Sigma G'$) if:

1) their chase graphs $\mathcal{G}(G, \Sigma)$ and $\mathcal{G}(G', \Sigma)$ are isomorphic through the isomorphism Ψ;
2) $\Psi_{|_V}$ is a bijection between V and V';
3) for each edge (a, b) of $\mathcal{G}(G, \Sigma)$, σ is the label of (a, b) if and only if σ is the label of $(\Psi(a), \Psi(b))$ in $\mathcal{G}(G', \Sigma)$.

I.e. , two KGs are chase-isomorphic if at each chase step they produce the same facts, up to a re-labeling of the vertices.

Clearly, KG anonymisation requires anonymising the nodes in the KG. In this respect we note that what makes nodes identifiable is not only their label (which can be aliased) but also their topological characteristics in the graph (i.e., in- and out-degree) that can make a node recognisable. We denote the triple comprising label and in- and out-degree as the sensitive attributes of a node, as captured by the following definition.

Definition 5.2 (Sensitive Attributes): Given a KG (G, Σ) and a node $v \in V$, the *sensitive attributes* of v is the triple: $\xi_G(v) = \{l_G(v), d^-_G(v), d^+_G(v)\}$, made of, respectively, its label, in-degree and out-degree.

We write $\xi_G(u) \neq \xi_G(v)$ to indicate that $l_G(u) \neq l_G(v)$, $d^-_G(u) \neq d^-_G(v)$ and $d^+_G(u) \neq d^+_G(v)$. To protect the KG from a NAR attack, the idea is to require each induced subgraph of x vertices to have at least other $k-1$ chase-isomorphic subgraphs with different sensitive attributes. This guarantees the NAR to match at least with k different subgraphs, each exhibiting different sensitive attributes: this implies that the attacker, if they were to choose a structure at random to match their NAR, would retrieve the correct structure with probability less than or equal to $1/k$. We call it a (k, x)-*chase anonymisation*.

More specifically, we want to anonymise the released graph by creating a new KG (A, Σ), such that 1) A is obtained by adding synthetic edges and vertices to G; 2) the vertex labels of A do not coincide with the ones of G, so that the attacker cannot retrieve the real names of the entities; 3) the edge weights of A are different from those of G, so that the attacker cannot use them for matching; 4) each induced subgraph of G of size x is (ii) chase-isomorphic to other $k-1$ induced subgraphs in A, which must be (i) pairwise vertex-disjoint and (iii) with different sensitive attributes. In particular, each node of a subgraph has different label, in-degree and out-degree with respect to the corresponding nodes in each of the k isomorphic subgraphs. This is formalised as follows.

Definition 5.3 ((k,x)-chase anonymisation): Let (G, Σ) be a KG with $G = (V, E, L, \omega, l)$ weakly connected and let $x \in [n]$ and $k \in \mathbb{N}$, $k \leq \binom{n}{x}$. A (k, x)-*chase anonymisation* of (G, Σ) is a KG (A, Σ) with $A = (V_A, E_A, L_A, \omega_A, l_A)$ such that:

1) $V \subseteq V_A$, $E \subseteq E_A$; *[augmentation]*
2) $L \cap L_A = \varnothing$; *[anonymisation of labels]*
3) $\omega_A(e) \neq \omega(e)$ for all $e \in E$; *[perturbation of weights]*
4) for all $X_1 \subset V$ such that $|X_1| = x$ and $G[X_1]$ is weakly connected, there exist $X_2, X_3, \ldots, X_k \subseteq V_A$ such that:
   (i) $X_i \cap X_j = \varnothing$ for all $i, j \in [k]$, $i \neq j$; *[vertex-disjointness]*
   (ii) $A[X_1] \cong_\Sigma A[X_i]$ through the chase-isomorphism Ψ_i, for all $i \in [k]$ with $\Psi_1 = id_{X_1}$; *[anonymity]*
   (iii) $\xi_A(\Psi_i(v)) \neq \xi_A(\Psi_j(v))$ for all $v \in X_1$, $i, j \in [k]$ with $i \neq j$; *[diversity]*

Here, parameter x refers to the maximum size of the substructures we can resist the attack, while k is related to the probability of correct re-identification (higher values guarantee stronger protection). At the same time, high values of k and x might substantially increase the computational cost of the anonymisation (see Section VIII). Therefore, the choice of k and x is to be evaluated case by case, depending on the application and on the desired level of privacy. The following result guarantees that a (k,x)-chase anonymisation is also resistant to NAR attacks of smaller size; the proof can be found in the supplementary material [5].

Proposition 5.1: Let (A,Σ) be a (k,x)-chase anonymisation of a KG (G,Σ) with Σ monotonic. Then (A,Σ) is a (k,x')-chase anonymisation of (G,Σ) for any $x' \leq x$.

Proposition 5.1 can be violated if G is not weakly connected. In this case, a (k,x)-chase anonymisation A does not guarantee that a weakly connected component C of size $x' < x$ is chase-isomorphic to $k-1$ other subgraphs, as it is not contained in any weakly connected subgraphs of size x. This problem can be overcome by performing a (k,x')-chase anonymisation on every weakly connected component C of G, with $x' = \min\{x, |C|\}$. Conversely, the hypothesis of the NAR to be weakly connected could be removed. Indeed, suppose to have a (k,x)-chase anonymisation A of G and a NAR of size x not weakly connected. Let $C_1,\dots,C_m$ be its weakly connected components with $|C_j| = x_j$ for $j \in [m]$: then each C_j is a weakly connected NAR with $x_j < x$. By Proposition 5.1, A is anonymised for any of such components, hence it is resistant to the original attack. Hence, a NAR that is not weakly connected can be equivalently seen as a sequence $C_1,\dots,C_m$ of weakly connected NARs of smaller size.

Protection from NAD attacks. The (k,x)-chase anonymisation presented in the previous paragraph makes the KG resistant also to attacks of type NAD, as they involve less knowledge than NAR attacks. We here introduce a lighter anonymisation framework that is resistant only to a NAD attack, but which may result in fewer added structures. Thus, despite the worst case computational complexity remaining the same as in the NAR case, this anonymisation framework may lead to smaller computational costs in real-world scenarios (see Section VIII). It relies on the concept of *KG-isomorphism*, extending the classical concept of graph-isomorphism.

Definition 5.4 (KG-isomorphism): Let (G,Σ) and (G',Σ) be two KGs with vertex sets V and V' and edge sets E and E' respectively. We say that they are *KG-isomorphic* if there exists a bijective function $\Phi : V \to V'$, called *KG-isomorphism*, such that for all $u,v \in V$:

$$(u,v) \in E \iff (\Phi(u),\Phi(v)) \in E' \tag{1}$$

$$(u,v) \in \mathcal{D}_{G,\Sigma} \iff (\Phi(u),\Phi(v)) \in \mathcal{D}_{G',\Sigma}\,. \tag{2}$$

I.e. two KGs are KG-isomorphic if they share both the topology of the ground extensional component (eq.(1) refers to classic isomorphism between graphs) and on the derived component (eq.(2) enforces the correspondence also on the derived edges). If two KGs are chase-isomorphic, then they are KG-isomorphic; the converse is not true, as shown in Fig.4.

If we are interested in being resistant only to a NAD attack, we can modify Definition 5.3 by substituting the chase-isomorphism of item (4ii) with the KG-isomorphism, namely:

(ii) $(A[X_1],\Sigma)$ and $(A[X_i],\Sigma)$ are KG-isomorphic through the *KG-isomorphism* Φ_i, for all $i \in [k]$ with $\Phi_1 = id_{X_1}$.

Semantic utility. We introduce two *semantic utility* metrics to quantify the coincidence of information in the original KG and in the anonymised one. This is done on a use-case basis, by focusing the metrics on a set of queries $\mathcal{Q}$ that can represent the most frequent downstream tasks asked by the users.

Notice that, given (A,Σ) a chase-anonymisation of a KG (G,Σ), the augmentation property (Definition 5.3, item (1)) implies that there is a bijective correspondence between the constants of the database G_D and (a subset of) the constants of A_D. When evaluating a query Q over G and A, in any set operation between $Q(G)$ and $Q(A)$ we assume the equivalence of their constants under this bijective correspondence.

Definition 5.5 (Semantic utility metrics): Let (G,Σ) be a KG, (A,Σ) be a (k,x)-chase anonymisation of it and $\mathcal{Q} = \{Q_1,\dots,Q_N\}$ be a set of queries. We define the following utility metrics:

$$\mathcal{U}(G,\Sigma,A,\mathcal{Q}) := \frac{1}{N}\sum_{Q\in\mathcal{Q}} \frac{|Q(G)\cap Q(A)|}{|Q(G)|}\,, \tag{3}$$

$$\mathcal{U}_\triangle(G,\Sigma,A,\mathcal{Q}) := \frac{1}{N}\sum_{Q\in\mathcal{Q}} \frac{|Q(G)\cap Q(A)|}{|Q(G)\cup Q(A)|}\,, \tag{4}$$

with the convention that if the denominator of a term is equal to zero, then the whole fraction is set to zero.

It holds that $0 \leq \mathcal{U}_\triangle(G,\Sigma,A,\mathcal{Q}) \leq \mathcal{U}(G,\Sigma,A,\mathcal{Q}) \leq 1$. Eq.(3) measures the amount of retained information of the anonymised KG with respect to the original one, i.e. the ratio of correct answers provided by A with respect to the ones given by G. Eq.(4) also considers possible incorrect answers provided by A and not existent in G due to the augmentation, measuring the ratio of correct answers returned by A (the ones coinciding with the ones provided by G) with respect to the total answers returned. In both cases, 1 is the most desirable value, where we have complete retention of information, i.e. no loss ($\mathcal{U}$) nor incorrect answers ($\mathcal{U}_\triangle$). Example of queries on which the semantic utility metrics can be defined are in Table I; for their meaning, we refer the reader to Section VIII.

The utility metrics will be used by the algorithms of the next sections for targeting the chase-anonymised KG, keeping the most information possible in terms of downstream tasks. This would translate into bringing $\mathcal{U}_\triangle$ the closest to 1.

VI. KLONE

Algorithm 1 introduces KLONE, our first method to obtain a (k,x)-chase anonymisation of a KG. In particular, the anonymised graph A in output is a (k,x)-chase anonymisation of the input KG for every $x \in [n]$; in other words, A is robust to a NAR attack (and hence also to a NAD one) regardless of its size. The anonymised graph is constructed by: (a) noising the edge weights of the original KG while maximising the utility metric $\mathcal{U}_\triangle$; (b) copying the KG k times to guarantee the

Algorithm 1: KLONE

Input: A knowledge graph (G, Σ) with $G = (V, E, L, \omega, l)$, $k \in \mathbb{N}$, set of queries $\mathcal{Q}$, in-degree distribution p^-, out-degree distribution p^+, L_A set such that $L_A \cap L = \emptyset$, weight distribution p_ω, $M \in \mathbb{N}$.
Output: A (k, x)-chase anonymisation (A, Σ) of (G, Σ) for any x.

1 $G \leftarrow$ WeightNoising$(G, G, \Sigma, E, \mathcal{Q}, p_\omega, M)$;
2 **for** $j = 1, \ldots, k$ **do** $G^j \leftarrow$ copy of G, $V(G^j) \leftarrow \{v_1^j, \ldots, v_n^j\}$;
3 $A \leftarrow \bigcup_{j=1}^{k} G^j$; $\mathcal{A} \leftarrow E(A)$,$\mathcal{V} \leftarrow \{\}$;
4 add a random edge between G^j and G^{j+1} for $j \in [k-1]$;
5 **for** $i = 1, \ldots, n$ **do**
6 $\delta_1^- \leftarrow d_A^-(v_i^1)$; $\delta_1^+ \leftarrow d_A^+(v_i^1)$; $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_i^1\}$;
7 **for** $j = 2, \ldots, k$ **do**
8 **for** $\varphi \in \{-, +\}$ **do**
9 $\delta_j^\varphi \leftarrow d_A^\varphi(v_i^j)$;
10 **while** $\exists \delta \in \{\delta_1^\varphi, \ldots, \delta_{j-1}^\varphi\}$ *s.t.* $\delta_j^\varphi = \delta$ **do**
11 $\delta_j^\varphi \leftarrow \max\{\delta_j^\varphi + 1, z \sim p^\varphi\}$
12 $\Delta \leftarrow \delta_j^\varphi - d_A^\varphi(v_i^j)$;
13 $\mathcal{C} \leftarrow \{v \in V(A) : v \notin (N_A^\varphi(v_i^j) \cup \mathcal{V} \cup V(G_j))\}$;
14 **if** $|\mathcal{C}| < \Delta$ **then**
15 $\mathcal{X} \leftarrow$ set of $\Delta - |\mathcal{C}|$ new vertices;
16 $V(A) \leftarrow V(A) \cup \mathcal{X}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{X}$;
17 select Δ elements $\{c_1, \ldots, c_\Delta\}$ from $\mathcal{C}$;
18 **if** $\varphi = -$ **then** $E(A) \leftarrow \bigcup_{t=1}^{\Delta}(c_t, v_i^j) \cup E(A)$;
19 **if** $\varphi = +$ **then** $E(A) \leftarrow \bigcup_{t=1}^{\Delta}(v_i^j, c_t) \cup E(A)$;
20 $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_i^j\}$;
21 $S \leftarrow E(A) \setminus \mathcal{A}$; assign to each $v \in V(A)$ a different label of L_A;
22 $A \leftarrow$ WeightNoising$(A, G, \Sigma, S, \mathcal{Q}, p_\omega, M)$;

Algorithm 2: WeightNoising

Input: A, G weighted labeled directed graphs, inference rules Σ, $S \subseteq E(A)$, set of queries $\mathcal{Q}$, weight distribution p_ω, $M \in \mathbb{N}$.
Output: A weighted labeled directed graph A.

1 **for** $i = 1, \ldots, M$ **do**
2 $A_i \leftarrow$ copy of A; **for** $e \in S$ **do** $\omega_{A_i}(e) \leftarrow z \sim p_\omega$;
3 $u(i) \leftarrow \mathcal{U}_\Delta(G, \Sigma, A_i, \mathcal{Q})$
4 $I \leftarrow \arg\max_i u(i)$; $A \leftarrow A_I$;

k chase-isomorphisms per each induced subgraph; (c) adding synthetic edges to reach diversity of in- and out-degrees; (d) anonymising the vertex labels (e) assign the weights to the synthetic edges while again maximising $\mathcal{U}_\Delta$.

More in details, (a) is performed in **line 1**, where the weights are chosen in such a way that the utility $\mathcal{U}_\Delta$ is maximised. Ideally, we would like the change of weights to completely preserve the output of the queries on the KG. This is done by the *WeightNoising* function in Algorithm 2, which randomly samples new weights for M times and chooses the ones that reach the highest value of $\mathcal{U}_\Delta$. Phase (b) is in **lines 2-3**. At this point, the graph A is the disjoint union of k copies of G, which guarantees that each induced subgraph is chase-isomorphic to other $k-1$ different subgraphs according to Definition 5.1. These copies are linked together in **line 4** to guarantee that A is weakly connected. Phase (c) is addressed in **lines 5–20**. The **for** loop in **line 5** iterates on the vertices of a copy of G while the **for** loop in **line 7** iterates on the copies of G, such that v_i^j refers to the i-th vertex in the j-th copy G^j of G; the last **for** loop in **line 8** repeats the procedure both for in-degree and out-degree. The goal is to add synthetic edges such that all the copies $\{v_i^2, \ldots, v_i^k\}$ of the vertex v_i^1 have different sensitive attributes (Definition 5.2). To do so, we randomly assign a new in- (out-) degree to each of them with the use of the input degree distributions p^- and p^+, making sure that this value is bigger than the original one (since we do not want to remove any of the original edges) and that they are all different from one another (**lines 9–11**). We then select a set of candidate vertices $\mathcal{C}$ from (to) which adds Δ new edges to (from) each v_i^j in order to meet the chosen in- (out-) degree. This set $\mathcal{C}$ consists of all the vertices of the graph that (i) do not belong to the same copy G^j of v_i^j, (ii) for which there is not already an existing edge and (iii) that do not belong to $\mathcal{V}$, i.e., the set of vertices for which we have already added the synthetic edges to meet the chosen in- (out-) degree (**line 13**). Condition (i) is necessary to not break the KG-isomorphisms between the subgraphs on the different copies of G. If the cardinality of $\mathcal{C}$ is less than the required number of edges that have to be added, we augment the graph with the needed number of new vertices (**lines 14–16**). We then randomly select Δ vertices from $\mathcal{C}$ and add the corresponding edges (**lines 17–19**). Finally, phase (d) is addressed in **line 21**, where to each vertex is assigned a new label; in phase (e) we assign a weight to each added synthetic edge (**line 22**) to maximise the utility $\mathcal{U}_\Delta$ again by the use of the function *WeightNoising*.

Correctness and complexity. KLONE correctly returns a (k, x)-chase anonymisation of the input KG and has polynomial time complexity. To argue for these properties, we assume that distributions p^- and p^+ have support in $[n]$; this is reasonable as n is the largest in-/out-degree that a vertex can have in G. Proofs are in the supplementary material [5].

Lemma 6.1: If the degree distributions p^- and p^+ have support in $[n]$, then the anonymised graph A output of Algorithm 1 is such that $kn \leq |V(A)| \leq 2kn + 1$.

Proposition 6.1: Algorithm 1 returns a (k, x)-chase anonymisation of the input knowledge graph for every $x \in [n]$. Moreover, it runs in $O(Mk^2n^2)$ time, under the hypothesis that the computation of the utility metric $\mathcal{U}_\Delta$ is $O(1)$ and the distributions p^- and p^+ have support in $[n]$.

VII. KGUARD

Algorithm 3 introduces KGUARD, our second proposed algorithm to obtain a (k, x)-chase anonymisation of a KG. Our aim is to reduce the number of synthetic vertices and edges added by KLONE by leveraging the subgraph chase-isomorphisms that already exist in the original KG. The anonymised graph is constructed by: (a) noising the edge weights of the original KG while maximising the utility metric $\mathcal{U}_\Delta$ as not to lose information; (b) finding all the induced subgraphs of size x and bucketing them into chase-isomorphic classes; (c) selecting k subgraphs in each bucket to guarantee the k chase-isomorphisms and/or duplicating some of such subgraphs if the bucket has less than k elements; (d) assigning different in- and out-degrees to the vertices mapped into each other by the chase-isomorphisms to reach diversity; (e) adding the synthetic edges to meet the requested in- and out-degrees;

Algorithm 3: KGUARD

Input: A knowledge graph (G, Σ) with $G = (V, E, L, \omega, l)$, $k, x \in \mathbb{N}$, set of queries Q, in-degree distribution p^-, out-degree distribution p^+, L_A set such that $L_A \cap L = \varnothing$, weight distribution p_ω, $M \in \mathbb{N}$.
Output: A (k, x)-chase anonymisation (A, Σ) of (G, Σ).

1 $A \leftarrow \text{WeightNoising}(G, G, \Sigma, E, Q, p_\omega, M)$;
2 $\mathcal{H} \leftarrow \textit{ConnectedInducedSubgraphs}(A, x)$; $\mathbb{G} \leftarrow \{\mathcal{G}(H, \Sigma) \mid H \in \mathcal{H}\}$;
3 $(\mathbb{B}, \mathbb{I}) \leftarrow \textit{IsomorphismBucketing}(\mathcal{H}, \mathbb{G})$; $\mathcal{V} \leftarrow \{\}$;
4 **for** $\mathcal{B} \in \mathbb{B}$ **do**
5 choose $\hat{\mathcal{B}} \subseteq \mathcal{B}$ s.t. $\forall H_1 \neq H_2 \in \hat{\mathcal{B}}$, $V(H_1) \cap V(H_2) = \varnothing$;
6 **while** $|\hat{\mathcal{B}}| < k$ **do**
7 choose $H \in \hat{\mathcal{B}}$; $H' \leftarrow$ copy of H;
8 $A \leftarrow A \cup H'$; $\hat{\mathcal{B}} \leftarrow \hat{\mathcal{B}} \cup H'$;
9 $\mathcal{B} \leftarrow$ choose k elements from $\hat{\mathcal{B}}$; $\mathcal{V} \leftarrow \bigcup_{H \in \mathcal{B}} V(H) \cup \mathcal{V}$;
10 **if** $\neg IsWeaklyConnected(A)$ **then** add randomly an edge from $A[V]$ to each other weakly connected component;
11 $\{V_1, \dots, V_s\} \leftarrow \textit{IsomorphismPartitioning}(\mathcal{V}, \mathbb{I})$;
12 **for** $i = 1, \dots, s$ **do**
13 $\{v_i^1, \dots, v_i^{n_i}\} \leftarrow$ vertices of V_i;
14 **for** $\varphi \in \{-, +\}$ **do**
15 $\{D^\varphi(v_i^1), \dots, D^\varphi(v_i^{n_i})\} \leftarrow$ *ChooseDeg*$(A, \varphi, p^\varphi, \{v_i^1, \dots, v_i^{n_i}\})$
16 **for** $\varphi \in \{-, +\}$ **do**
17 **for** $v \in \mathcal{V}$ **do**
18 $\Delta \leftarrow D^\varphi(v) - d_A^\varphi(v)$; $\mathcal{X}_1 \leftarrow \{\}$; $\mathcal{X}_2 \leftarrow \{\}$;
19 $\mathcal{C} \leftarrow \{u \in V(A) : D^{\mp\varphi}(u) - d_A^{\mp\varphi}(u) > 0\}$;
20 $\mathcal{C} \leftarrow \mathcal{C} \setminus (\bigcup_{\mathcal{B} \in \mathbb{B}} \bigcup_{H \in \mathcal{B} : v \in V(H)} V(H) \cup N_A^\varphi(v))$;
21 **if** $|\mathcal{C}| < \Delta$ **then**
22 $m \leftarrow \min\{\Delta - |\mathcal{C}|, |V(A) \setminus \mathcal{V}|\}$;
23 $\mathcal{X}_1 \leftarrow$ select m elements from $V(A) \setminus \mathcal{V}$;
24 **if** $|\mathcal{C}| + m < \Delta$ **then**
25 $\mathcal{X}_2 \leftarrow$ set of $\Delta - |\mathcal{C}| - m$ new vertices;
26 $V(A) \leftarrow V(A) \cup \mathcal{X}_2$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{X}_2$;
27 $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{X}_1$;
28 select Δ elements $\{c_1, \dots, c_\Delta\}$ from $\mathcal{C}$;
29 **if** $\varphi = -$ **then** $E(A) \leftarrow \bigcup_{t=1}^{\Delta} (c_t, v) \cup E(A)$;
30 **if** $\varphi = +$ **then** $E(A) \leftarrow \bigcup_{t=1}^{\Delta} (v, c_t) \cup E(A)$;
31 assign to each $v \in V(A)$ a different label of L_A;
32 $A \leftarrow \text{WeightNoising}(A, G, \Sigma, S, Q, p_\omega, M)$;

Algorithm 4: *ChooseDeg*

Input: A graph A, $\varphi \in \{-, +\}$, degree distribution p, a set of vertices $\{v^1, \dots, v^m\} \subseteq V(A)$.
Output: A sequence $\{D^\varphi(v^1), \dots, D^\varphi(v^m)\}$ of all different degrees.

1 $D^\varphi(v^1) \leftarrow d_A^\varphi(v^1)$;
2 **for** $j = 2, \dots, m$ **do**
3 $D^\varphi(v^j) \leftarrow d_A^\varphi(v^j)$;
4 **while** $\exists D \in \{D^\varphi(v^1), \dots, D^\varphi(v^{j-1})\}$ *s.t.* $D^\varphi(v^j) = D$ **do**
5 $D^\varphi(v^j) \leftarrow \max\{D^\varphi(v^j) + 1, z \sim p\}$

(f) anonymising the vertex labels; (g) assigning the weights to the synthetic edges while optimising $\mathcal{U}_\Delta$ as to maximise the usefulness of the anonymisation in terms of retained information. Items (a), (f) and (g) are the same as in KLONE, so we refer the reader to the previous section; they are in KGUARD respectively in **line 1**, **line 31** and **line 32**. We now provide details of all the other items. We use the convention that if $\varphi \in \{-, +\}$, then $\mp\varphi$ denotes the opposite sign of φ.

(b) Subgraphs identification and isomorphism bucketing. We first find all the weakly connected subgraphs of the KG induced by a set of x vertices (**line 2**, function *ConnectedInducedSubgraphs*, where we adapt the algorithm by S. Karakashian et al. [31] to the case of directed graphs) and gather them in $\mathcal{H}$. Then, for each subgraph $H \in \mathcal{H}$, we apply the reasoning rules Σ to obtain their chase graph; together, they form the set $\mathbb{G}$ (**line 2**). Then, we group together the subgraphs of $\mathcal{H}$ into chase-isomorphic clusters, referred to as *buckets*: each bucket only contains subgraphs that are chase-isomorphic to each other according to Definition 5.4. This happens in **line 3**, where the function *IsomorphismBucketing* returns $\mathbb{B}$, the set of buckets, and $\mathbb{I}$ the set of chase-isomorphism functions Ψ between elements of $\mathbb{G}$; to do so we use a variant of the algorithm in [12]. The process of isomorphism bucketing is quite straightforward, and it is done iteratively on the elements of $\mathcal{H}$. Indeed, let $\mathbb{B}$ be the set of chase-isomorphic buckets of the first i elements of $\mathcal{H}$ and let $H_{i+1} \in \mathcal{H}$ be the next considered subgraph. Since the chase-isomorphism induces an equivalence relation, it suffices to check whether H_{i+1} is isomorphic to a representative of each equivalence class (bucket): if so, then H_{i+1} is added to such bucket, otherwise a new bucket is created with H_{i+1} as representative.

(c) k-isomorphism. From each bucket $\mathcal{B} \in \mathbb{B}$, we want to choose k chase-isomorphic subgraphs with a pairwise disjoint set of vertices (**lines 4–9**). If the bucket $\mathcal{B}$ has less than k vertex-disjoint subgraphs, we copy (and add to A) some of them until reaching k vertex-disjoint subgraphs (**lines 6–8**). After this process, we leave in each bucket only the k chosen subgraphs (**line 9**): they will be the ones fulfilling item (4) of Definition 5.3 of (k, x)-chase anonymisation. We call $\mathcal{V}$ the set of all the vertices appearing in at least a subgraph of a bucket: these are the vertices for which we want to guarantee the diversity of the sensitive attributes ξ. In **line 10**, we ensure that the new graph A is weakly connected by randomly adding a synthetic edge between the original graph and each other weakly connected component. At the end of this part, each subgraph of the input KG G of cardinality x has other $k-1$ subgraphs in A that are chase-isomorphic to it.

(d) Diversity: degree assignment. **Lines 11–15** regard the assignment of a new in- and out-degree to the vertices of the chase-isomorphic subgraphs in order to guarantee diversity. To do so we first partition the set $\mathcal{V}$ in equivalence classes $\{V_1, \dots, V_s\}$ where $u, v \in \mathcal{V}$ belong to the same class if and only if there exists a chase-isomorphism $\Psi \in \mathbb{I}$ such that $\Phi(u) = v$. In other words, all the vertices in a class must have different sensitive attributes because they belong to subgraphs that are chase-isomorphic. This is done in **line 11** by the function *IsomorphismPartitioning*. Then **lines 12–15** assign a different in- and out-degree to each vertex belonging to the same class, with the use of the function *ChooseDeg*. This function, described by Algorithm 4, is similar to what is done in KLONE: we draw them at random according to the input distributions p^- and p^+, while making sure that the chosen values are always bigger or equal than the original ones and that they are all different between each others.

(e) Diversity: addition of synthetic edges. **Lines 16–30** add the necessary synthetic edges to meet the in- and out- de-

grees assigned in the previous step, thus making A satisfying condition (4iii) of Definition 5.3. For each $v \in \mathcal{V}$, we aim at selecting a set $\mathcal{C}$ of candidate vertices from (to) which add Δ new edges to (from) v in order to achieve the chosen in-(out-) degree. Such set $\mathcal{C}$ is made of all the vertices u of the graph such that (i) its current out-(in-) degree $d_A^{\mp\varphi}(u)$ is less than the assigned one $D^{\mp\varphi}(u)$, so that they have 'space' for new synthetic out- (in-)coming edges (**line 19**), (ii) it does not belong to any of the subgraphs H appearing in the buckets for which $v \in V(H)$ (**line 20**) and (iii) there does not already exist an edge between v and u (**line 20**). Item (i) is important for not exceeding $D^{\mp\varphi}(u)$, while item (ii) is crucial for not 'breaking' the isomorphism between subgraphs since if $u, v \in V(H)$, then adding an edge between v and u would modify the topology of H and hence its chase graph. If the cardinality of $\mathcal{C}$ is less than the required number Δ of edges to be added (**line 21**), we add to it $m = \Delta - |\mathcal{C}|$ vertices of A that do not belong to $\mathcal{V}$, as for them no prescribed in-/out-degree is required (**lines 22-23**). If this is not possible because $V(A) \setminus \mathcal{V}$ is too small, we create and augment A with new vertices (**lines 24–26**) to be added to $\mathcal{C}$ so that $|\mathcal{C}| = \Delta$. We then randomly select Δ vertices from $\mathcal{C}$ (**line 28**) and add the corresponding edges from (to) them to (from) v (**lines 29-30**).

Correctness and complexity. KGUARD is correct and has exponential worst-case complexity. The proof of the following proposition can be found in the supplementary material [5].

Proposition 7.1: Algorithm 3 returns a (k, x)-chase anonymisation (A, Σ) of the input knowledge graph (G, Σ), for chosen $x \in [n]$ and $k \in \mathbb{N}$, $k \leq \binom{n}{x}$.

The number of connected subgraphs with x vertices can be exponentially large in x, specifically $\binom{n}{x}$, potentially leading to exponential time and space complexities for the algorithm in the worst case. This issue is further compounded by the isomorphism [12], as the problem of determining whether two graphs are isomorphic is still not known to be polynomial or not.[2] Thus, in the general case, the algorithm for graph isomorphism might require exponential time. Nonetheless, it is important to note that anonymisation is a one-time process used for data release. In the subsequent experimental section, we empirically evaluate the computational time of KGUARD and show its applicability in a wide range of scenarios: high values of k and x increase in general the computational cost of the anonymisation, but also implies stronger protection in terms of probability of correct re-identification ($\leq 1/k$) and maximum size of the attack we can resist ($\leq x$). The choice of k and x hence strongly depends on the considered KG and the needed level of privacy and it is to be evaluated case by case, based on the application and on the desired level of privacy.

NAD attack. In the case where we want to protect the KG only from NAD attacks and thus achieve a weaker KG-anonymisation (see Section V), it suffices to modify Algorithm 3 in the following way. **Line 2**: $\mathbb{G} \leftarrow \{\Sigma(H) \mid H \in \mathcal{H}\}$, we just consider the derived edges. **Line 3**: the function $(\mathbb{B}, \mathbb{I}) \leftarrow$ *IsomorphismBucketing*$(\mathcal{H}, \mathbb{G})$ returns as $\mathbb{I}$ the set of KG-isomorphisms found between the elements of $\mathbb{G}$.

The computational time in the general case remains the same.

[2]Contrarily to the subgraph isomorphism problem, known to be NP-hard. This is not our case, as we need only to assess whether two graphs are isomorphic.

VIII. Experiments

We evaluate KLONE and KGUARD for different KGs and reasoning tasks. We also compare our work against existing privacy preserving techniques for structural and neighbourhood attacks, stemming from the classical k-anonymity [46].

Datasets. Our first set of experiments considers two well-known random graph models: the Erdős-Rényi and the Scale-Free network. For the Erdős-Rényi graph, we consider the directed model $D(n, M)$, where n is the number of vertices and M is the number of directed edges that are sampled uniformly at random among the n^2 possible ones. We set $M = n\ln(n)/2$, which corresponds to the threshold for weak connectivity in the limit $n \to +\infty$ (it follows from Theorems 1 and 4 in [18]). In this model, the in- and out-degree distribution is binomial (Poisson in the limit), thus making vertices with very large out-degree, called *hubs*, unlikely to appear. The edge weights are set to 0 with probability 0.5, while with probability 0.5 are sampled uniformly in $(0, 1]$. For the Scale-Free model, we consider a random graph where the out-degree profile of each vertex is randomly sampled from a truncated power-law distribution, i.e., for each vertex v:

$$\mathbb{P}(d_G^+(v) = d) = \frac{d^{-\alpha}}{\sum_{k=0}^{n-1} k^{-\alpha}} \quad \text{if } 0 \leq d < n, \ = 0 \text{ otherwise,} \quad (5)$$

where $\alpha > 0$ is a parameter. Low values of α correspond to a high probability of having nodes with large out-degree, while for high values of α, almost all the vertices have small out-degree with high probability. A feature of graphs with such power law distributions (particularly when α is small) is the presence of *hubs*. These networks are called *scale free* [22, 6] and have been shown to model economic networks [17]. The edge weights are assigned by sampling in $[0, 1]$ uniformly at random, and then rescaled such that for each $v \in V$, it holds that $\sum_{u \in N^-(v)} \omega(u, v) \leq 1$: this is because we want to model a company network with ownership relationships between companies, where the total sum of shares owned of a node cannot be more than 100%. For both models, we enforce the generated graph to be weakly connected by randomly adding edges between its components. Alternatively, our algorithms can be applied separately to each component.

Lastly, we consider five different real-world graphs from the literature, namely MovieLens small [20], Econ-Mahindas [44], Power-1138-Bus [44], Bitcoin Alpha [33] and an anonymised fragment of the Company Ownership graph [36]. This latter graph and its importance among central banks and financial authorities have already been discussed in the introduction and Figure 1. From each of the above-mentioned networks, we extract and anonymise a weakly connected component, whose characteristics can be found in Table III.

Reasoning Tasks. Our experiments use two distinct sets of reasoning rules. For the networks modelling company ownerships (namely the *Scale-Free* and the *Company Ownership*

graphs), we use the *control* rules between two companies ($\Sigma = \{\sigma_1, \sigma_2\}$ in Table I). For all the others, we use rules on *reachability*: a derived edge from vertex u to vertex v is added if there exists a path from u to v whose product of the edges' weight is greater than 0. In Vadalog [3], this can be written as the set of rules $\Sigma = \{\sigma_3, \sigma_4\}$ in Table I.

Evaluation Metrics. We evaluate our work according to three key aspects of synthetic graph data, namely *fidelity*, *utility*, and *privacy* [42]. The fidelity measures how much the anonymised graph is statistically *close* to the original one. We evaluate it in terms of the number of vertices added to the original graph and of the Wasserstein-1 distance [14] of the degree and weight distributions between the original and the anonymised graph. We then evaluate the utility of the anonymised graph with the metrics $\mathcal{U}$ and $\mathcal{U}_\triangle$ of Definition 5.5, measuring the correctness of potential down-stream tasks in the form of reasoning queries. We consider the queries of Table I. Query Q_1 returns all the couple of nodes connected by an edge of weight > 0.3, Q_2 returns each couple of nodes that are connected in both directions with different weights, Q_3 selects all the vertices of the KG with at least 2 out-going edges, while Q_4 selects all the vertices of the KG with at least 2 out-going edges with weight greater than q, Q_5 returns each couple for which there is a direct control (shares above 50%), Q_6 selects all the couple for which one controls the other with a (ownership) path of length three, and Q_7 returns all the couple (x, y) such that x controls y and x is not controlled by any entity, thus being the ultimate controller of y. Queries $\{Q_1, Q_2, Q_3, Q_4\}$ can be applied to any generated network described in the previous section, while queries $\{Q_5, Q_6, Q_7\}$ are specifically for the company ownership networks. For Q_4 we set $q = 0$ for all the considered networks but the company ownership ones, for which we set $q = 0.5$, as it is the threshold for establishing control.

The set of queries used to evaluate the utility metrics will be a subset $\mathcal{Q}$ of $\{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7\}$, chosen to best represent the potential analysis tasks based on the network type, i.e., company ownership networks or general graphs.

Finally, we evaluate the improvement in terms of privacy reached by our approach (the weaker one based on KG-isomorphism) with respect to classical structural anonymisation. For the comparison, we consider an algorithm that provides anonymisation by forming k isomorphic subgraphs in the graph (*k-isomorphism*) and we measure the percentage of subgraphs that are correctly anonymised by this approach, meaning that we count how many isomorphic copies are also KG-isomorphic. Our experiments confirm that neglecting derived knowledge leads to potential leaks of information as classical approaches are not able to provide privacy for each substructure: the presence of derived edges might break the isomorphism, making a subgraph uniquely identifiable.

Implementation. We implemented our analyses and algorithms in Python, using NetworkX [19]. In the experiments we model the in-degree and out-degree distributions, namely p^- and p^+, as a binomial distribution fit on the degree profile of the input graphs. The weight distribution p_ω is chosen by partitioning the weights of the input graph into 200 equal-width bins, from which we estimate an empirical distribution. We recall that different approaches, like a non-parametric KDE [13], can be easily integrated. Finally, the results report the mean and the standard error over five different random trials; while the ↓ (↑) symbol alongside each metric indicates that a lower (higher) value is better.

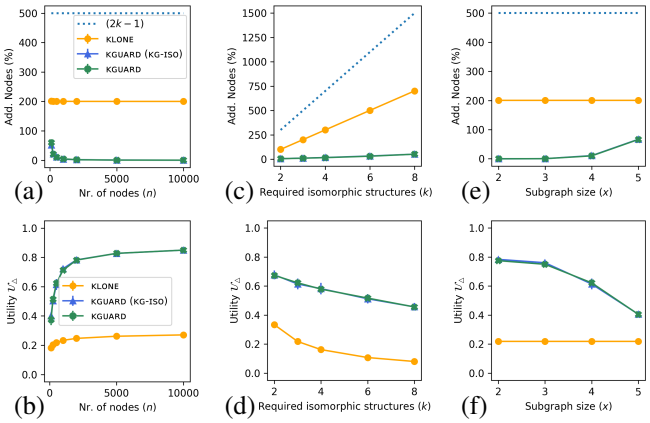


Fig. 5: Erdős-Rényi model with $\Sigma = \{\sigma_3, \sigma_4\}$ and $\mathcal{Q} = \{Q_5, Q_6\}$, varying n, k and x.

A. Simulated and real-world networks

In this section, we extensively evaluate our approaches on different graph models, varying the number of vertices, edges, and privacy requirements, such as the number of chase-isomorphic subgraphs k and the size of considered subgraphs x. We refer to KGUARD and KGUARD (KG-ISO) for the chase-isomorphism and the weaker KG-isomorphism anonymisation (see Section V), respectively. Where not otherwise stated, we consider $n = 500$, $k = 3$ and $x = 4$.

Erdős-Rényi. In Figure 5, we investigate the fidelity and utility of the anonymisation for the Erdős-Rényi model. In Figure (5a) and (5b) we vary the number of vertices n of the input graph from 100 to 10,000. Figure (5a) shows the percentage of vertices added during the anonymisation by the algorithms (*Add. Nodes (%)*): KLONE consistently adds around 200% more vertices as expected, influenced by the privacy requirement $k = 3$, whereas KGUARD introduces fewer additional vertices by leveraging existing isomorphic subgraphs within the input graph. The dotted horizontal line represents the theoretical upper bound of $(2k - 1)\%$ added vertices for KLONE (Lemma 6.1). The efficiency of KGUARD with respect to KLONE, in terms of added structures, is also reflected in Figure (5b), where KGUARD exhibits significantly higher utility compared to KLONE. We also notice that KGUARD (KG-ISO) has slightly better performance than KGUARD, as fewer redundant structures are required to satisfy the weaker anonymisation requirement. In terms of computational cost, both algorithms require approximately the same amount of time to anonymize a graph (e.g., around 3 hours for a graph with $5,000$ nodes). However, for larger graphs, KLONE proves to be more efficient, requiring only 5 hours for a graph with

n	α	Algo	$\mathcal{U}\uparrow$	$\mathcal{U}_\triangle\uparrow$	Weights ↓	Degree ↓	A.Nodes(%)↓
500	3	KLONE	**0.96 ± 0.01**	0.62 ± 0.00	0.10 ± 0.00	3.43 ± 0.09	200.2 ± 0.22
		KGUARD	**0.96 ± 0.01**	**0.75 ± 0.02**	**0.05 ± 0.00**	**2.14 ± 0.11**	**32.7 ± 16.33**
500	4	KLONE	**0.97 ± 0.01**	0.61 ± 0.00	0.10 ± 0.01	3.20 ± 0.06	200.3 ± 0.26
		KGUARD	**0.97 ± 0.01**	**0.77 ± 0.03**	**0.05 ± 0.01**	**2.10 ± 0.05**	**15.65 ± 13.5**
500	5	KLONE	**0.96 ± 0.01**	0.61 ± 0.00	0.10 ± 0.01	3.05 ± 0.05	200.0 ± 0.32
		KGUARD	**0.96 ± 0.01**	**0.76 ± 0.02**	**0.05 ± 0.00**	**1.89 ± 0.13**	**15.49 ± 5.55**
2k	5	KLONE	**0.97 ± 0.01**	0.61 ± 0.00	0.10 ± 0.00	3.10 ± 0.05	200.0 ± 0.04
		KGUARD	0.96 ± 0.00	**0.82 ± 0.01**	**0.04 ± 0.00**	**0.75 ± 0.05**	**1.77 ± 0.67**
5k	5	KLONE	**0.93 ± 0.05**	0.61 ± 0.00	0.13 ± 0.03	3.15 ± 0.02	200.0 ± 0.02
		KGUARD	0.92 ± 0.05	**0.81 ± 0.03**	**0.03 ± 0.02**	**0.35 ± 0.03**	**0.65 ± 0.54**
10k	5	KLONE	**0.95 ± 0.04**	0.61 ± 0.00	0.11 ± 0.03	3.17 ± 0.04	200.0 ± 0.01
		KGUARD	0.93 ± 0.05	**0.82 ± 0.03**	**0.04 ± 0.03**	**0.18 ± 0.01**	**0.25 ± 0.18**

TABLE II: Anonymisation for scale-free model with $k = 3$, $x = 4$, $\Sigma = \{\sigma_1, \sigma_2\}$, $\mathcal{Q} = \{Q_5\}$, and varying n and α.

10,000 nodes while KGUARD and KGUARD (KG-ISO) take nearly 10 hours. In Figures (5c) and (5d), we vary the privacy requirement k from 2 to 8. As expected, in Figure (5c) KLONE shows a linear increase in additional vertices with respect to k, while KGUARD exhibits just a slight increase. This trend is also reflected in the utility (Figure 5d), where KLONE has a much lower utility compared to KGUARD. In Figures (5e) and (5f), we evaluate the anonymisation quality by varying the size x of the considered subgraphs. The performance of KLONE remains unaffected as expected, since the algorithm provides a (k, x)-chase anonymisation for any $x \in [n]$ (Proposition 6.1), instead KGUARD overhead increases with the size of subgraphs x. In general, KGUARD shows a better performance, while KLONE might be the better choice when x is sufficiently large or if we are uncertain about the attacker knowledge. In fact, while KGUARD overhead and computational time increase with x, KLONE maintains a consistent cost as a single anonymisation A of G is effective for any choice of x. Notice that the utility $\mathcal{U}$ is close to 1 (the highest value) for all settings and algorithms.

Scale-free. Table II reports the performance of our algorithms on a scale-free network with varying numbers of vertices up to $10,000$, and $\alpha \in \{3, 4, 5\}$. The results show that KGUARD generally outperforms KLONE as it adds significantly fewer nodes to the anonymised graph (% Add. Nodes ↓), and it has distributions that are closer to the original graph both for weights (*Weights* ↓), and degrees (*Degree* ↓), along with better utility metrics. Similarly to the Erdős-Rényi model, as the number of nodes increases, KGUARD has a significantly lower overhead than KLONE: e.g., with $10,000$ nodes KGUARD adds only 0.25% of new nodes. Finally, the utility metrics, namely $\mathcal{U}$ and $\mathcal{U}_\triangle$, remain consistently high, close to the optimal value of 1. Especially for KGUARD, this indicates that the anonymised graph retains nearly all the information of the original graph, when evaluated on the given set of queries. Notice that, as the performance difference between KGUARD (KG-ISO) and KGUARD is consistent and minimal, we omit the former in the remaining results.

Real-world graphs. We investigate the performance of our approaches for five real-world graphs from various domains, including energy and economics. Table III reports for each graph the number of vertices and edges, along with the anonymisation performance for KLONE and KGUARD, with $k = 3$ and $x = 4$. The results show that KGUARD consistently outperforms KLONE across almost all the evaluated metrics. As KGUARD introduces significantly less structural overhead (*Add. Nodes (%)*), it better preserves the degree (*Degree* ↓) and weight (*Weights* ↓) distributions in the anonymised graph. Both KGUARD and KLONE consistently achieve almost a perfect utility $\mathcal{U}$, close to 1, indicating that the anonymised graph A has no loss of information compared to graph G in terms of utility queries. However, for the utility $\mathcal{U}_\triangle$ KLONE shows a more significant presence of incorrect answers due to the k copies it makes of the original graph, as clearly reflected in the higher percentage of additional nodes. In contrast, KGUARD maintains a relatively high utility, thanks to a low percentage of additional nodes, meaning that only a few vertex additions are needed to ensure the k chase-isomorphisms.

B. Utility and state-of-the-art comparison

Utility. Table IV shows the performance of our algorithms when different sets of queries are considered in the semantic utility metrics, both on a scale-free network and an Erdős-Rényi graph of $2,000$ nodes. For example, the first results in Table IV regard a scale-free network with reasoning rules $\Sigma = \{\sigma_1, \sigma_2\}$, where the metrics $\mathcal{U}$ and $\mathcal{U}_\Delta$ are computed on the query Q_3 asking *"How many companies have direct control over a company by holding more than 50% of its shares?"*. For both algorithms, the analysts obtain an approximately correct lower bound on the real number (since $\mathcal{U}$ is close to 1), meaning that the companies that would have been identified in response to this query in the original graph G are also present in the anonymised graph A. However, to preserve company identities, both approaches have created some redundant structures that might also positively answer this question. The utility $\mathcal{U}_\triangle$ also takes into account the number of additional companies retrieved from this query that exist in A but not in G. In general KGUARD has higher utility $\mathcal{U}_\triangle$, as it depends on the number of additional structures created in the anonymised graph A. For KLONE, the additional structures only partially depend on the input graph and are mainly influenced by the number of anonymisation copies k required (as shown also in Figure 5). In contrast, KGUARD works specifically on each subgraph, and additional structures are only needed if the original graph contains very unique subgraphs that require duplication. This difference is also evident across the different sets of queries: while KLONE maintains relatively low utility $\mathcal{U}_\triangle$, KGUARD achieves values close to 1, representing the optimal solution. The most challenging case for both algorithms occurs with the query set $\mathcal{Q} = \{Q_5, Q_6\}$, as only a small subset of nodes from the original graph match such queries. Therefore, we notice that even a limited number of incorrect new matches in the anonymised graph substantially impact the metric $\mathcal{U}_\triangle$. To summarise, Table IV shows that KGUARD generally performs better than KLONE due to the fewer added structures, though different query sets may lead to varying results, depending on their complexity.

State-of-art comparison. We now investigate the privacy of our approaches compared to classical structural anonymisation (*k-Iso*). This method anonymises graphs by forming k pairwise isomorphic subgraphs, making them indistinguishable to an attacker. However, as we qualitatively showed in Figure 3c,

TABLE III: Anonymisation results for real-world graphs with $k = 3$ and $x = 4$.

Dataset	Metric	KLONE	KGUARD
Company Ownership [36]	utility $\mathcal{U}$ ↑	**0.99 ± 0.00**	0.99 ± 0.01
$n = 3,000$, $\lvert E\rvert = 3,900$	utility $\mathcal{U}_\triangle$ ↑	0.66 ± 0.01	**0.82 ± 0.01**
$\Sigma = \{\sigma_1, \sigma_2\}$, $\mathcal{Q} = \{Q_5\}$	weights ↓	0.47 ± 0.00	**0.40 ± 0.00**
	degree ↓	3.78 ± 0.01	**2.55 ± 0.04**
	add. nodes (%) ↓	200.0 ± 0.1	**40.8 ± 2.77**
Econ-Mahindas [44]	utility $\mathcal{U}$ ↑	1.00 ± 0.00	**1.00 ± 0.00**
$n = 1,200$, $\lvert E\rvert = 3,300$	utility $\mathcal{U}_\triangle$ ↑	0.24 ± 0.00	**0.68 ± 0.00**
$\Sigma = \{\sigma_3, \sigma_4\}$, $\mathcal{Q} = \{Q_3, Q_4\}$	weights ↓	0.38 ± 0.00	**0.34 ± 0.00**
	degree ↓	4.54 ± 0.05	**2.33 ± 0.07**
	add. nodes (%) ↓	200.1 ± 0.1	**8.22 ± 0.26**
MovieLens small [20]	utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	**1.00 ± 0.00**
$n = 2,100$, $\lvert E\rvert = 3,200$	utility $\mathcal{U}_\triangle$ ↑	0.09 ± 0.00	**0.83 ± 0.01**
$\Sigma = \{\sigma_3, \sigma_4\}$, $\mathcal{Q} = \{Q_3, Q_4\}$	weights ↓	0.42 ± 0.00	**0.21 ± 0.00**
	degree ↓	4.19 ± 0.03	**0.35 ± 0.01**
	add. nodes (%) ↓	200.1 ± 0.1	**2.12 ± 0.08**
Power-1138-Bus [44]	utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	**1.00 ± 0.00**
$n = 1,100$, $\lvert E\rvert = 2,600$	utility $\mathcal{U}_\triangle$ ↑	0.25 ± 0.00	**0.90 ± 0.01**
$\Sigma = \{\sigma_3, \sigma_4\}$, $\mathcal{Q} = \{Q_3, Q_4\}$	weights ↓	0.20 ± 0.00	**0.15 ± 0.00**
	degree ↓	3.78 ± 0.07	**0.64 ± 0.05**
	add. nodes (%) ↓	200.2 ± 0.1	**0.35 ± 0.00**
Bitcoin Alpha [33]	utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	**1.00 ± 0.00**
$n = 1,700$, $\lvert E\rvert = 3,100$	utility $\mathcal{U}_\triangle$ ↑	0.13 ± 0.00	**0.64 ± 0.01**
$\Sigma = \{\sigma_3, \sigma_4\}$, $\mathcal{Q} = \{Q_3, Q_4\}$	weights ↓	0.36 ± 0.00	**0.30 ± 0.00**
	degree ↓	4.12 ± 0.06	**0.85 ± 0.04**
	add. nodes (%) ↓	200.1 ± 0.1	**6.21 ± 0.08**

TABLE IV: Anonymisation results for different query sets and graph models ($k = 3$, $x = 4$, $n = 2,000$, $\alpha = 5$).

Model	Query set	Metric	KLONE	KGUARD
Scale-free	$\mathcal{Q} = \{Q_5\}$	add. nodes (%) ↓	200.0 ± 0.0	**1.77 ± 0.67**
$\Sigma = \{\sigma_1, \sigma_2\}$		utility $\mathcal{U}_\triangle$ ↑	0.61 ± 0.00	**0.82 ± 0.01**
		utility $\mathcal{U}$ ↑	**0.97 ± 0.01**	0.96 ± 0.00
Scale-free	$\mathcal{Q} = \{Q_6\}$	add. nodes (%) ↓	200.1 ± 0.0	**2.17 ± 0.25**
$\Sigma = \{\sigma_1, \sigma_2\}$		utility $\mathcal{U}_\triangle$ ↑	0.50 ± 0.00	**0.83 ± 0.29**
		utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	**1.00 ± 0.00**
Scale-free	$\mathcal{Q} = \{Q_3, Q_4\}$	add. nodes (%) ↓	200.1 ± 0.0	**1.86 ± 0.72**
$\Sigma = \{\sigma_1, \sigma_2\}$		utility $\mathcal{U}_\triangle$ ↑	0.02 ± 0.00	**0.10 ± 0.02**
		utility $\mathcal{U}$ ↑	**0.95 ± 0.06**	0.94 ± 0.06
Scale-free	$\mathcal{Q} = \{Q_7\}$	add. nodes (%) ↓	200.1 ± 0.0	**1.87 ± 0.62**
$\Sigma = \{\sigma_1, \sigma_2\}$		utility $\mathcal{U}_\triangle$ ↑	0.56 ± 0.00	**0.65 ± 0.01**
		utility $\mathcal{U}$ ↑	0.79 ± 0.01	**0.80 ± 0.01**
Erdős-Rényi	$\mathcal{Q} = \{Q_1\}$	add. nodes (%) ↓	200.1 ± 0.1	**1.84 ± 0.58**
$\Sigma = \{\sigma_3, \sigma_4\}$		utility $\mathcal{U}_\triangle$ ↑	0.63 ± 0.00	**0.93 ± 0.00**
		utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	0.99 ± 0.00
Erdős-Rényi	$\mathcal{Q} = \{Q_3, Q_4\}$	add. nodes (%) ↓	200.1 ± 0.1	**1.84 ± 0.58**
$\Sigma = \{\sigma_3, \sigma_4\}$		utility $\mathcal{U}_\triangle$ ↑	0.25 ± 0.00	**0.78 ± 0.01**
		utility $\mathcal{U}$ ↑	**1.00 ± 0.00**	**1.00 ± 0.00**
Erdős-Rényi	$\mathcal{Q} = \{Q_2\}$	add. nodes (%) ↓	200.1 ± 0.1	**1.86 ± 0.58**
$\Sigma = \{\sigma_3, \sigma_4\}$		utility $\mathcal{U}_\triangle$ ↑	0.59 ± 0.02	**0.68 ± 0.07**
		utility $\mathcal{U}$ ↑	**0.96 ± 0.06**	**0.96 ± 0.06**

neglecting derived links in KGs results in severe privacy issues and information leaks. In Table V, we quantitatively evaluate such leaks by measuring the percentage of subgraph structures that are correctly anonymised, i.e. for which there exists other $k - 1$ KG-isomorphic subgraphs within the KG; we call this index δ-*anonymity*. Results are shown for $x = 4$ and $k = 3$. The table confirms our theoretical analysis: our approaches anonymise each individual subgraph and consistently achieve δ-*anonymity* = 1. Contrarily, the state-of-the-art approach *k-Iso* does not protect the privacy of all entities: in the worst case (Bitcoin-Alpha), only 60% of subgraphs are effectively not uniquely identifiable, leaving 40% of potentially identifiable entities by an attacker. The best case for the state-of-the-art algorithm is Econ-Mahindas, still showing around 8% of subgraph structures vulnerable to attacks.

TABLE V: Anonymisation rate with $k = 3$ and $x = 4$.

Reasoning	Graph	$\delta - anonymity$ ↑		
		K-Iso	KLONE	KGUARD
Reach	Scale-free ($n = 100, \alpha = 5$)	0.795	**1.000**	**1.000**
$\Sigma = \{\sigma_3, \sigma_4\}$	Scale-free ($n = 500, \alpha = 5$)	0.873	**1.000**	**1.000**
	MovieLens small [20]	0.818	**1.000**	**1.000**
	Power-1138-Bus [44]	0.723	**1.000**	**1.000**
	Bitcoin Alpha [33]	0.605	**1.000**	**1.000**
	Econ-Mahindas [44]	0.923	**1.000**	**1.000**
Control	Scale-free ($n = 100, \alpha = 5$)	0.720	**1.000**	**1.000**
$\Sigma = \{\sigma_1, \sigma_2\}$	Scale-free ($n = 500, \alpha = 5$)	0.728	**1.000**	**1.000**
	Company Own [36]	0.680	**1.000**	**1.000**

IX. Related Work

Several definitions of privacy have been proposed over the years, ranging from traditional syntactic privacy definitions [46] to more recent semantic ones like differential privacy [29, 39, 30, 15]. Differential privacy (DP) frameworks are designed to protect individual entities during data analysis, i.e. they prevent an attacker from determining whether a specific individual was included in the input data. However, DP faces significant challenges when applied to highly correlated and network data [30]. Adding noise to nodes or edges often fails to conceal the overall structure and relationships within the data. In addition, the anonymization algorithm must be carefully designed to ensure that the DP mechanism preserves the statistics of interest. These limitations make DP not the best candidate in a KG setting, where structural anonymisation is often preferred to provide privacy while preserving data utility [25]. In the last years, structural anonymisation concepts originally developed for relational databases [9, 28] have been extended to graph data, including models such as k-degree and k-neighbourhood anonymity [38, 43]. These approaches usually focus on modifying the graph so that it exhibits at least k "similar" (e.g., isomorphic) structures with respect to the adversary knowledge [21, 47]. However, existing methods target only specific graph types — like directed [32, 10] or weighted graphs [35] — and when applied to KGs they may expose sensitive information, since adversaries can exploit the rich KG attributes [23]. Specifically for KGs, only few anonymisation solutions exist [25, 26, 24, 23, 45]. They focus mainly on the privacy of single entities [23, 45], providing also personalized anonymisation [25]; as well as the privacy protection of sequential published data [26, 24]. Differently from existing work, we focus on a more complex attack model considering subgraph structures, where attackers can exploit logical reasoning and derived knowledge, commonly present in business settings, to identify target entities.

X. Conclusion

We discussed the application of privacy protecting schemes in the realm of KGs, showing that existing structural approaches fail to maintain privacy in presence of derived knowledge. We proposed new structural anonymisation techniques that guarantee privacy also under attacks with information on the reasoning rules, and described and evaluated two algorithms achieving it by generating synthetic variations of the input graph while preserving its utility for downstream tasks.

XI. AI-Generated Content Acknowledgement

Authors acknowledge that no GenAI tools were used in any stage of the research, nor in the writing.

References

[1] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. 2nd. Springer, 2008. ISBN: 1848009976.

[2] Sergio Barezzani et al. "TADA: Target-aware data anonymization". In: *IEEE Transactions on Privacy* 2 (2025), pp. 15–26.

[3] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. "The Vadalog system: Datalog-based reasoning for knowledge graphs". In: *Proc. of the VLDB Endowment* 11.9 (2018).

[4] Luigi Bellomarini et al. "Knowledge graphs and enterprise AI: the promise of an enabling technology". In: *ICDE'19*. IEEE. 2019, pp. 26–37.

[5] Luigi Bellomarini et al. *Supplementary material*. https://drive.google.com/drive/folders/1jp4BhiyGSNnfMn63sudaQ3gmqfffeiRu?usp=drive_link. [Online; June-2025]. 2025.

[6] Béla Bollobás et al. "Directed scale-free graphs". In: *Proc. of SODA'03*. 2003, pp. 132–139.

[7] Andrea Cali, Georg Gottlob, and Thomas Lukasiewicz. "A general datalog-based framework for tractable query answering over ontologies". In: *PODS'09*. 2009.

[8] Longbing Cao. "AI in finance: challenges, techniques, and opportunities". In: *ACM Computing Surveys (CSUR)* 55.3 (2022), pp. 1–38.

[9] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. "A survey of graph-modification techniques for privacy-preserving on networks". In: *Artificial Intelligence Review* 47 (2017), pp. 341–366.

[10] Jordi Casas-Roma et al. "k-Degree anonymity on directed networks". In: *Knowledge and Information Systems* 61 (2019), pp. 1743–1768.

[11] Xihui Chen, Sjouke Mauw, and Yunior Ramírez-Cruz. "Publishing community-preserving attributed social graphs with a differential privacy guarantee". In: *Proc. on Privacy Enhancing Technologies* (2020).

[12] Luigi Pietro Cordella et al. "An improved algorithm for matching large graphs". In: *IAPR-TC15 workshop*. Citeseer. 2001, pp. 149–159.

[13] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. "Remarks on some nonparametric estimates of a density function". In: *Selected Works of Murray Rosenblatt* (2011), pp. 95–100.

[14] Roland L. Dobrushin. "Prescribing a system of random variables by conditional distributions". In: *Theory Prob. Applications* 15 (1970), pp. 458–486.

[15] Cynthia Dwork. "Differential privacy". In: *International colloquium on automata, languages, and programming*. Springer. 2006, pp. 1–12.

[16] Lisa Ehrlinger and Wolfram Wöß. "Towards a definition of knowledge graphs". In: *SEMANTiCS (Posters, Demos, SuCCESS)*. Vol. 1695. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[17] Diego Garlaschelli et al. "The scale-free topology of market investments". In: *Physica A: Statistical Mechanics and its Applications* 350.2 (2005), pp. 491–499.

[18] Alasdair J. Graham and David A. Pike. "A note on thresholds and connectivity in random directed graphs". In: *Atlantic Electr. J. of Mathematics* 3.1 (2008), pp. 1–5.

[19] Aric Hagberg, Pieter J Swart, and Daniel A Schult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[20] F Maxwell Harper and Joseph A Konstan. "The movielens datasets: history and context". In: *ACM trans. on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[21] Michael Hay et al. "Resisting structural re-identification in anonymized social networks". In: *Proc. VLDB Endowment* 1.1 (2008), pp. 102–114.

[22] Cesar Hidalgo and Albert-Laszlo Barabasi. *Scale-free networks*. Scholarpedia, 2008.

[23] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Cluster-based anonymization of knowledge graphs". In: *ACNS'20*. Springer. 2020, pp. 104–123.

[24] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Privacy-preserving sequential publishing of knowledge graphs". In: *ICDE'21*. IEEE. 2021.

[25] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Protecting privacy in knowledge graphs with personalized anonymization". In: *IEEE Trans. on Dependable and Secure Computing* 21.4 (2024), pp. 2181–2193.

[26] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Time-aware anonymization of knowledge graphs". In: *ACM Trans. on Privacy and Security* (2023).

[27] Aidan Hogan et al. "Knowledge graphs". In: *ACM Computing Surveys (Csur)* 54.4 (2021), pp. 1–37.

[28] Shouling Ji, Prateek Mittal, and Raheem Beyah. "Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey". In: *IEEE Comm. Surveys & Tutorials* 19.2 (2016), pp. 1305–1326.

[29] Xun Jian, Yue Wang, and Lei Chen. "Publishing graphs under node differential privacy". In: *IEEE Transactions on Knowledge and Data Engineering* 35.4 (2023), pp. 4164–4177.

[30] Honglu Jiang et al. "Applications of differential privacy in social network analysis: A survey". In: *IEEE Trans. on Knowledge and Data Engineering* 35.1 (2021), pp. 108–127.

[31] Shant Karakashian, Berthe Y Choueiry, and Stephen G Hartke. "An algorithm for generating all connected subgraphs with k vertices of a graph". In: *Lincoln, NE* 10 (2013).

[32] Maryam Kiabod, Mohammad Naderi Dehkordi, and Behrang Barekatain. “A fast graph modification method for social network anonymization”. In: *Expert Systems with Applications* 180 (2021), p. 115148.

[33] Srijan Kumar et al. “Rev2: Fraudulent user prediction in rating platforms”. In: *Proc. of WSDM’18*. ACM. 2018, pp. 333–341.

[34] Yang Li et al. “Private graph data release: A survey”. In: *ACM Computing Surveys* 55.11 (2023), pp. 1–39.

[35] Yidong Li et al. “Practical anonymity models on protecting private weighted graphs”. In: *Neurocomputing* 218 (2016), pp. 359–370.

[36] Davide Magnanimi et al. “Reactive company control in company knowledge graphs”. In: *ICDE’23*. IEEE. 2023, pp. 3336–3348.

[37] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. “Testing implications of data dependencies”. In: *ACM TODS* 4.4 (1979), pp. 455–469.

[38] Abdul Majeed and Sungchang Lee. “Anonymization techniques for privacy preserving data publishing: A comprehensive survey”. In: *IEEE access* 9 (2020), pp. 8512–8545.

[39] Tamara T Mueller et al. “SoK: Differential privacy on graph-structured data”. In: *arXiv preprint arXiv:2203.09205* (2022).

[40] Natalya Fridman Noy et al. “Industry-scale knowledge graphs: lessons and challenges”. In: *Commun. ACM* 62.8 (2019), pp. 36–43.

[41] Ciyuan Peng et al. “Knowledge graphs: opportunities and challenges”. In: *Artificial Intelligence Review* (2023), pp. 1–32.

[42] Vamsi K Potluru et al. “Synthetic Data Applications in Finance”. In: *arXiv preprint arXiv:2401.00081* (2023).

[43] Weilong Ren, Kambiz Ghazinour, and Xiang Lian. “kt-Safety: graph release via k-anonymity and t-closeness”. In: *IEEE Trans. on Knowledge and Data Engineering* (2022).

[44] Ryan A. Rossi and Nesreen K. Ahmed. “The network data repository with interactive graph analytics and visualization”. In: *AAAI*. 2015.

[45] Maxime Thouvenot, Olivier Curé, and Philippe Calvez. “Knowledge graph anonymization using semantic anatomization”. In: *Big Data’20*. IEEE. 2020, pp. 4065–4074.

[46] Sabrina De Capitani di Vimercati et al. “k-Anonymity: from theory to applications”. In: *Trans. Data Priv.* 16 (2023), pp. 25–49.

[47] Bin Zhou and Jian Pei. “Preserving privacy in social networks against neighborhood Attacks”. In: *Proc. of ICDE’08*. 2008, pp. 506–515.