iCAT: An Interactive Customizable Anonymization Tool

Momen Oqaily[1],Yosr Jarraya[2], Mengyuan Zhang[2], Lingyu Wang[1], Makan Pourzandi[2] and Mourad Debbabi[1]

[1] Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada
[2] Ericsson Security Research, Ericsson Canada, Montreal, QC, Canada

Abstract. Today's data owners usually resort to data anonymization tools to ease their privacy and confidentiality concerns. However, those tools are typically ready-made and inflexible, leaving a gap both between the data owner and data users' requirements, and between those requirements and a tool's anonymization capabilities. In this paper, we propose an interactive customizable anonymization tool, namely *iCAT*, to bridge the aforementioned gaps. To this end, we first define the novel concept of *anonymization space* to model all combinations of per-attribute anonymization primitives based on their levels of privacy and utility. Second, we leverage NLP and ontology modeling to provide an automated way to translate data owners and data users' textual requirements into appropriate anonymization primitives. Finally, we implement *iCAT* and evaluate its efficiency and effectiveness with both real and synthetic network data, and we assess the usability through a user-based study involving participants from industry and research laboratories. Our experiments show an effectiveness of about 96.5% for data owners and 92.6% for data users.

1 Introduction

Nowadays, network data has become a highly valuable resource for different stakeholders as its analysis can serve many use-cases. However, data owners are generally reluctant to share their data due to the risk of information disclosure and potentially staggering financial fines imposed by privacy regulations such as the European General Data Protection Regulation (GDPR) [20]. This reluctance is worsened with the increase in the number of the publicly announced data breach and misuse incidents[3] [4]. To this end, data anonymization is a well-known solution for easing data owners' concerns. However, the effectiveness of sharing anonymized data critically depends on data owners to make the right choice of anonymization approach, and to apply the approach properly to achieve the right trade-off between utility and privacy. However, this can be a difficult task since

[3] https://www.wsj.com/articles/google-exposed-user-data-feared-repercussions-of-disclosing-to-public-1539017194
[4] https://www.techworld.com/security/uks-most-infamous-data-breaches-3604586/

most data owners likely lack a systematic understanding of the search space (i.e., all possible anonymization approaches). To make things worse, most existing anonymization tools only provide very limited choices, and manually translating privacy/utility requirements into the tools' anonymization capabilities is usually tedious and error-prone, as demonstrated in the following.

Motivating Example. Figure 1 depicts how three data users translate their different analysis needs into utility requirements (left), while the data owner translates his/her levels of trust for those users into different privacy requirements (right). Four existing anonymization tools (top-center) are applied to the four data attributes (middle-center) to show the limitations (bottom-center).

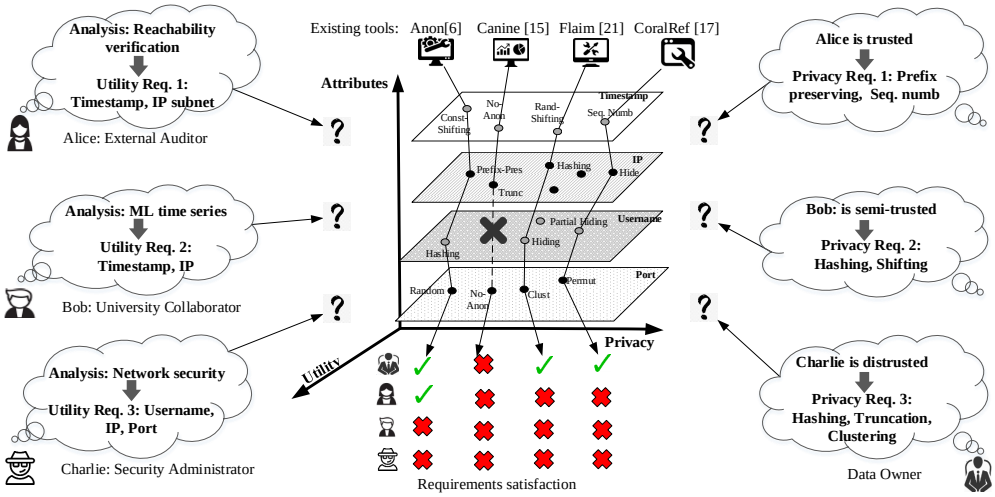


Fig. 1: The motivating example

- The existing anonymization approach assumes that each data user (e.g. Alice) can easily understand what is needed for his/her analysis (e.g., verifying network reachability) and translate that need into concrete utility requirements (e.g., the timestamps and the subnet relationship of IPs need to be preserved). This might not be the case in practice (as confirmed by our user-based experiments in Section 5), which could lead to many iterations between the data owner and the data user before finding the right answer.
- It is also expected that the data owner can easily understand his/her level of trust for each data user and translate it into concrete privacy requirements (e.g., Alice can only be given prefix-preserving and sequentially numbered data), and he/she is willing to understand each user's utility requirement (since the user is not involved in selecting the tool), and reconcile them with his/her privacy requirements. However, real-world data owners are usually not so considerate and might simply go with whatever provided by some handy anonymization tools.
- As shown in the middle of Figure 1, existing tools generally only implement a small set of anonymization primitives suitable for a subset of the data

attributes. Theoretically, the data owner can resort to a collection of such tools to cover all attributes. However, practically, this could be a difficult task since most tools do not offer the needed customization, e.g., which attributes to anonymize and to what privacy/utility levels, and the selected tools may not be compatible with each other and such incompatibility can potentially result in erroneous or inconsistent results.

In this paper, we propose an *i*nteractive *c*ustomizable *a*nonymization *t*ool, namely *iCAT*, to address the aforementioned challenges. Intuitively, *iCAT* is designed to cover the entire "space" shown in the middle of Figure 1 (instead of a few points covered by each existing tool), and to help both the data owner and data users by automating their requirements translation. Specifically, we first propose the novel concept of *anonymization space*, which models all possible combinations of existing anonymization primitives (which are applicable to the given data attributes) as a lattice based on their relationships in terms of privacy and utility. Second, as an application of the anonymization space concept, the privacy and utility requirements are jointly enforced through a simple access control mechanism. Third, we develop an ontology-driven Natural Language Processing (NLP) approach to automatically translate the textual requirements from both parties into combinations of anonymization primitives inside the anonymization space. Therefore, our main contributions are threefold:

1. To the best of our knowledge, our notion of anonymization space is the first systematic model of existing anonymization primitives that characterizes their capabilities in terms of privacy and utility, as well as their relationships. This model provides data owners with clearer understanding of possible anonymization choices, and it also, for the first time, allows the data users to be actively involved in the decision process.
2. To realize the potential of anonymization space, we design and implement an automated tool, *iCAT*, by leveraging existing anonymization primitives and a popular NLP engine [17]. In contrast to most of the existing anonymization tools, *iCAT* provides more flexibility (by allowing access to the entire anonymization space) and better usability (by automating requirements translation). The interactive nature of *iCAT* also implies the potential of a new paradigm for providing data anonymization as a service.
3. We evaluate the effectiveness and efficiency of *iCAT* using both synthetic and real data, and assess its usability through a user-based study involving participants from both industry and academia. Our results demonstrate the effectiveness and efficiency of our solution.

The remainder of this paper is organized as follows. Section 2 defines our anonymization space model and describes the privacy/utility access control. In Section 3, we describe the requirements translation process using NLP and ontology modeling. Section 4 details the implementation. Section 5 gives experimental results. Section 6 provides more discussions. Section 7 reviews related works and Section 8 concludes this paper.

2 Anonymization Space

In this section, we first define our threat model, then we review existing anonymization primitives and finally we define the *anonymization space* model and privacy/utility access control mechanism.

2.1 Threat Model

We define the parties involved in the data anonymization process and their trust relationships as the following :

- The data owner, who has useful datasets that can be used for different purposes, is interested in protecting the privacy of his/her data to avoid any data misuse. The data owner has different trust levels of the data users, which will determine the amount of information that he/she is willing to outsource.
- The data users, who have different use-cases of the data (e.g., auditing, research purposes, etc.), are interested in having the maximum data utility, in order to achieve valid results. The data users trust the data owners and are willing to share their use-cases with them.

In scope threats: We assume that both data owner and user will follow the procedure to express their requirements, while the latter is interested in obtaining output with higher utility if the tool provides him/her such an opportunity.

Out of scope threats: Our tool is not designed to mitigate any weakness or vulnerability of the underlying anonymization primitives (e.g., frequency analysis, data injection attacks, or data linkage attacks). Whereas, those primitives are used as a black box in our data anonymization module and can be replaced by other, better primitives when available. Moreover, we consider the failure in requirement translation by the NLP engine out of the scope.

2.2 Anonymization Primitives

There exist many data anonymization primitives in the literature even though most existing tools only support a limited number of those primitives (a detailed review of related work is provided in Section 7). To facilitate further discussions, Table 1 provides a list of common anonymization primitives, examples of plain data, and the corresponding anonymized data obtained using the primitives [5].

2.3 Lattices-based Anonymization Space

Following our motivating example shown in Figure 1, suppose the data owner is dissatisfied with those existing anonymization tools. Instead, he/she would like to apply the anonymization primitives given in Table 1. Obviously, he/she would find himself/herself facing a plethora of choices as follows:

[5] This list is not meant to be exhaustive, and our model and methodology can be extended to include other anonymization primitives

Primitive	Plain Data Example	Corresponding Anonymized Data
Prefix-preserving	IP1:**12.8.3.**4 ; IP2:**12.8.3.**5	IP1:**51.22.7.**33 ; IP1:**51.22.7.**19
Truncation	IP1:12.8.**3.4** ; IP2:12.8.**3.5**	IP1:12.8.**X.X** ; IP2:12.8.**X.X**
Const. Substitution	Version:2.0.1	Version: VERSION
Const. Shifting	Time1: **2019**-03-31; Time2: **2019**-03-30	Time1: **2022**-03-31; Time2: **2022**-03-30
Random Shifting	Time1: **2019**-03-31; Time2: **2019**-03-30	Time1: **2003**-03-31; Time2: **2015**-03-30
Sequ. Numbering	Time1: 2019-03-31; Time2: 2019-03-30	Time1: T1; Time2: T2
Partial Hiding	Time1: 2019-**03-31**; Time2: 2019-**03-30**	Time1: 2019-**X-X**; Time2: 2019-**X-X**
Hashing	ID:40018833	ID: H3%s2*D9
Clustering	Port1:2**25**; Port2: 2**77**	Port1:**200**; Port2: 2**77**
Permutation	Port1:2**25**; Port2: 2**77**	Port1:2**77**; Port2: 2**25**
Randomization	Port1: **225**; Port2: **277**	Port1:**423**; Port2: **29**

Table 1: Anonymization Primitives

- First, each data attribute may be anonymized using a different collection of the anonymization primitives (e.g., IPs may work with prefix preserving, truncation, hashing, etc., while IDs with clustering, hashing, etc., and both can be either completely hidden or plainly given with no anonymization).
- Second, different anonymization primitives applied to an attribute may yield different levels of, and sometimes incomparable, privacy and utility (e.g., for IPs, hashing provides more privacy/less utility than prefix preserving, whereas they are both incomparable to truncation or randomization).
- Finally, the data owner and data users' requirements typically involve multiple attributes, as demonstrated in Figure 1, and sometime in a complex fashion, e.g., the data owner might say "I can only give you the data with the IPs hashed, or with the IDs clustered, but not both", while a data user asks "I know I may not get the data with the IPs truncated and the IDs hashed, but what would be my next best option?"

The above discussions clearly demonstrate a need for a systematic way to represent and organize all the possible choices of anonymization primitives that can be applied to a given dataset. For this purpose, we propose a novel concept, namely *anonymization space*, by considering each data attribute as a *dimension*, and each combination of anonymization primitives that can cover all the attributes as a *point* inside the anonymization space. Considering the fact that anonymization primitives are not always comparable in terms of privacy/utility, and inspired by the Denning's Axioms [4, 21], we consider the collection of anonymization primitives applicable to each attribute to form a lattice based on their relationships in terms of privacy and utility, and the product of all those lattices (which is also a lattice by lattice theory [5]) represents the anonymization space. The following more formally defines those concepts.

Definition 1. ***(Anonymization Space)*** *Given $\mathbb{A} = \langle a_1, a_2, \ldots, a_n \rangle$ as a sequence of attributes to be anonymized, and given $F_i = \{f_1, f_2, \ldots, f_m\} (1 \leq i \leq n)$ as the set of anonymization primitives that are applicable to a_i, we define*

- *the attribute lattice $\mathcal{L}_i (1 \leq i \leq n)$ as a lattice $\langle F_i, \prec \rangle$ where for any $f_1, f_2 \in F_i$, we have $f_1 \prec f_2$ iff f_1 provides better utility and more stringent privacy than f_2 when both are applied to a_i, and*
- *the anonymization space corresponding to $\mathbb{A}$ as $\prod_{i=1}^{n} \mathcal{L}_i$.*

Example 1. Fig. 2.A (top) shows some examples of anonymization primitives and Fig. 2.B shows their applicability (using their indices) to six attributes. Fig. 2.C shows the six attribute lattices. Due to space limitations, we omit the anonymization space (which would have a size of $20,736$).

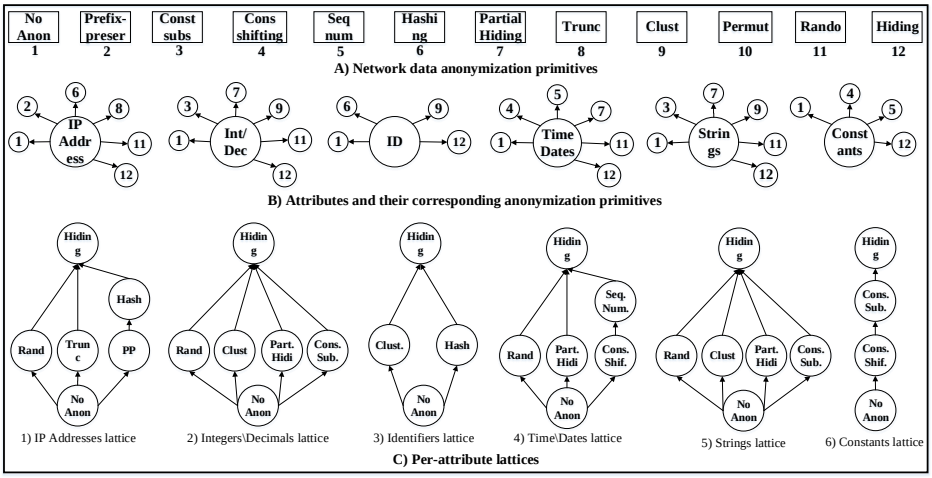


Fig. 2: *An example anonymization space* **A**) Examples of anonymization primitives with their indices, **B**) Examples of data attributes and their applicable anonymization primitives and **C**) The attribute lattices.

By providing a clearer picture of the anonymization primitives and their relationships, the anonymization space concept may have many use cases. For our purpose, we show how the concept can be used to jointly enforce the privacy and utility requirements through a simple access control mechanism (inspired by the Bell–LaPadula (BLP) model [3]), while allowing the data user to be actively involved in the anonymization process. Specifically, if we consider each point (which is a collection of anonymization primitives) in the anonymization space as a privacy/utility *level*, then the data owner's privacy requirement can be mapped to such a level (this mapping will be automated in Section 3), and everything above this level will also satisfy the privacy requirement since by definition it will yield more privacy, namely the *privacy-up* rule. Conversely, a data user's requirement can also be mapped to a level below which any level would also satisfy the utility requirement, namely the *utility-down* rule. This is more formally stated in Definition 2.

Definition 2. ***(Privacy/Utility Access Control)*** *Given the data attributes* $\mathbb{A}$*, the corresponding anonymization space* $\mathbb{AS} = \prod_{i=1}^{n} \mathcal{L}_i$*, and the privacy requirement* $L_p \in \mathbb{AS}$ *and utility requirement* $L_u \in \mathbb{AS}$ *(specified by the data owner and data user, respectively), any* $L \in \mathbb{AS}$ *will satisfy both requirements iff* $L_p \prec L$ *(privacy up) and* $L \prec L_u$ *(utility down) are both true.*

Example 2. Figure 3 shows an example of anonymization space corresponding to the IP and ID attributes. The data owner requires **Ha** (hashing) for IPs and

NA (no anonymization) for IDs. By the privacy-up rule, all levels inside the upper shaded area will also satisfy privacy requirements. The following discusses two data users' utility requirements.

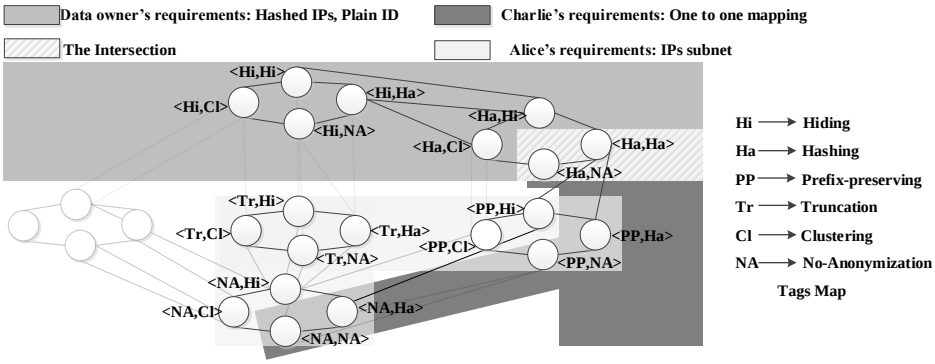


Fig. 3: An example of *anonymization space* for attributes IP and ID, and the privacy/utility access control for Alice and Charlie

1. Charlie requires to preserve the one-to-one mapping for both IPs and IDs. Following the utility-down rule, the dark gray area highlights all the levels that satisfy Charlie's utility requirements. Also, the area with crossing lines includes all levels that satisfy both the privacy and utility requirements, i.e., ⟨**Ha**, **Ha**⟩ and ⟨**Ha**, **Na**⟩.
2. Alice requires to preserve the IP subnets. The light gray area highlights all the levels that satisfy Alice's utility requirement. Since there is no intersection between the upper shaded area and the light gray area, no level can satisfy both the privacy-up and utility-down rules, which means no anonymization primitive can satisfy both the privacy and utility requirements for Alice. However, the anonymization space makes it easy to choose an alternative level that will satisfy the privacy requirement while providing the best possible utility to Alice, e.g., ⟨**Ha**,**Na**⟩.

3 Requirements Translation

To ease the burden on both data owners and data users, *iCAT* accepts requirements expressed in a natural language (English in our case) and translate them into anonymization primitives. In this section, we first discuss requirements translation using NLP and ontology modeling, and then explain ambiguity resolution.

Requirements Processing using NLP. The first step in translating the data owner and data user's requirements into combinations of anonymization primitives in the anonymization space is to understand them linguistically. For this purpose, *iCAT* leverages the Stanford Parser CoreNLP [17], which provides a set of natural language processing tools. Initially, the CoreNLP parser separates the English requirements into different sentences. Since CoreNLP can mark up

the structure of sentences in terms of phrases and syntactic dependencies and indicate which noun phrases refer to the same entities, we can obtain the sentence representing each requirement. After that, the Part-Of-Speech Tagger (POS Tagger) tool from CoreNLP is leveraged to filter and prepare the requirements for the ontology modeling step (c.f. Section 3). The POS tagger returns the sentences' words as a pair consisting of tokens and their part of speech tags (the linguistic type of the words, i.e., noun, verb, adjective, etc.). After that, unrelated words (i.e., pronouns, symbols, adverbs, etc.) are filtered out from each requirement, which will speed up the requirements translation.

Example 3. Figure 4 shows how a data owner's requirement "Data stored based on time occurrence" is processed to obtain the attribute data type *timestamp* and the associated anonymization primitive *shifting*.

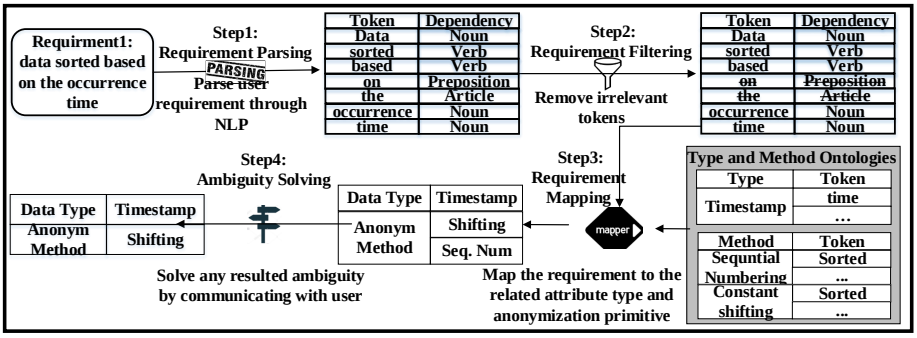
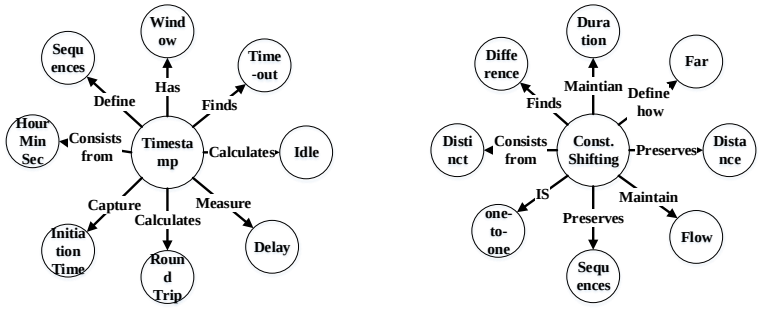


Fig. 4: Example showing the requirement translation process

Ontology Modeling. We use ontology modeling to define the relationship between requirements and data attributes/anonymization primitives as follows.

Ontology Learning We first define the concepts for data owner and user as: **i)** anony-methods; **ii)** method-func; **iii)** attribute-types; **iv)** attribute-synon. Based on our definitions, the instances of the anony-methods are the existing anonymization primitives and the method-func instances are manually created based on the functionality and unique properties that each anonymization primitive can achieve. Moreover, the instances of the attribute-type concept are the given attributes types and the attribute-synon instances are manually created based on the use/synonymous of each attribute type. After that, we find the relationships between those concepts' instances by defining relations between the anony-methods and the method-func concepts. Also, by defining relations between the attribute-types and the attribute-synon instances. For example, Figure 5 shows the type-ontologies related to the time-stamp attribute type and the method-ontology related to the constant shifting anonymization primitive. After that, we store the resulted ontologies into two separate tables, namely the type-ontology and the method-ontology.

Requirements Mapping We apply the learned ontologies to the processed and filtered requirements from the NLP in order to find the data attributes and the anonymization primitives corresponding to the user's requirements. This is done



A) Excerpt of the Timestamp type-ontology **B) Excerpt of the Constant-Shifting method-ontology**

Fig. 5: Ontologies of timestamp and constant shifting.

by matching every tokenized word in the processed requirement with the type and the method ontologies tables shown in Figure 4 as follows.

1. For each tokenized word in each annotated requirement, the tokenized word is matched first with the type ontology and then with the method ontology.
2. If the tokenized words are mapped to only one record from the type ontology table and one record from the method ontology table, then the requirement is translated properly, and the mapper will pass to the second requirement.
3. If none of the tokenized words match any record in both type and method ontologies tables, the word is dropped from the sentence annotations table.
4. If the user tokenized words fail to map to any record from the type and/or method ontologies or if the tokenized words have multiple matching, then the mapper will return an error message to the user reporting this issue and forward this conflict to the ambiguity solving process as discussed next.

Ambiguity Resolution. Ambiguity can occur for several reasons. We discuss how *iCAT* handles it as follows.

- The sentences entered by the user are mistakenly parsed at requirement parsing step (because of typos or NLP failures), which is not due to *iCAT*.
- The same requirements can be translated into different anonymization methods. For example, consider the following requirement; *Req-1: each IP address must be mapped to one IP address.* Both IP hashing and prefix-preserving can satisfy this requirement. In this case, the ambiguity solver of *iCAT* will display a small multi-choice menu to the user, such that this ambiguity can be resolved interactively.
- The same requirement can be expressed in different ways; For example, *the sequence of events is mandatory* versus *the order of logged records must be preserved.* This issue is discussed in Section 6.
- The data user's requirement is mapped to anonymization primitives that do not satisfy the data owner's requirement. In this case, *iCAT* suggests alternative anonymization primitives that offer the closest utility level to what is specified by the data owner.

4 Implementation

Figure 6 illustrates the flowchart (left) and main architecture (right) of *iCAT*, as detailed below.

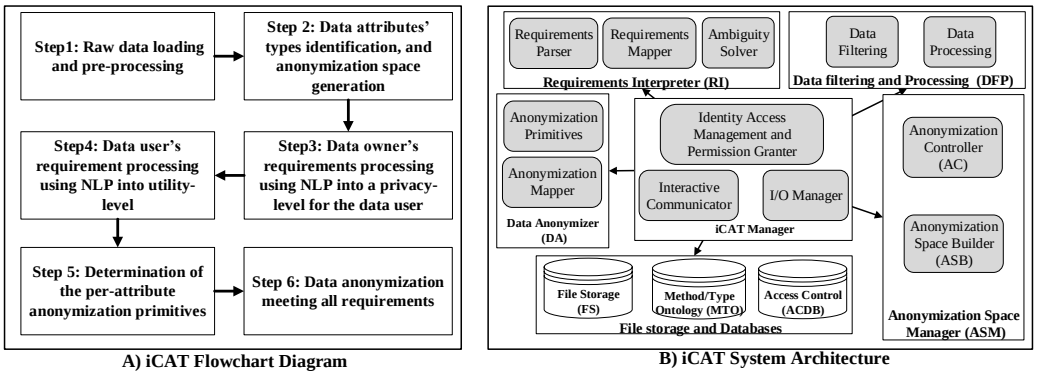


Fig. 6: Flowchart of *iCAT*.

***iCAT* Flowchart.** As shown in Figure 6.A, raw data is loaded into *iCAT* and pre-processed by the data owner where he can filter the attributes and clean the records if necessary (**step 1**). Then, the data attribute types are identified and used to build the anonymization space lattice and generate the privacy/utility access control model (**step 2**). Third, the data owner interacts with *iCAT* to input his privacy requirements (in natural languages) for a data user (**step 3**). These requirements are parsed and mapped to a privacy-level in the anonymization space. After that, the data user inputs the utility requirements which are also parsed and mapped to a utility-level in the anonymization space (**step 4**). Based on the privacy and utility levels, *iCAT* identifies the right combination of anonymization primitives (**step 5**). Finally, the data is anonymized and returned to the data user (**step 6**).

***iCAT* Architecture.** We provide a brief description of the main architecture of *iCAT*, as shown in Figure 6.B, while leaving more details to the Appendix due to space limitations. All the modules of *iCAT* are implemented in Java. The Data Loading and Processing (DLP) module is used to load the data, and enables filtering and cleansing operations. These operations allow performing statistical disclosure control for balancing privacy risks and data utility. The Requirements Interpreter (RI) module translates the data owner's and data user's requirements into data attributes types and anonymization primitives. The *iCAT* Manager module associates the data user identity with the privacy-level specified by the data owner and interacts with the data owner or data user. The I/O Manager module is responsible for configuring the data source and the loading the actual data. The Anonymization Space Manager module is for generating the anonymization space and implementing the access control mechanism. Finally, the Data Anonymizer module is for anonymizing the data and it is designed in a modular way to easily accommodate new anonymization primitives.

5 Experiments

In this section, we evaluate the effectiveness and usability of *iCAT* through a user-based study with participants from both industry and academia working on data analysis. Also, we evaluate the efficiency of *iCAT* using real data.

5.1 Experimental Settings

Datasets selection. We used four datasets in our experiments as shown in Table 2. The first is the Google cluster dataset [10], i.e., traces from requests processed by Google cluster management system. The second is cloud logs collected from different OpenStack Neutron services. The third dataset is a database dump of the OpenStack Nova service. The fourth dataset is the BHP-OBS machine learning dataset [24]. We select the aforementioned datasets for the following reasons: **i)** The privacy constraints and requirements are already known for datasets from the industrial collaborator; **ii)** The public datasets are widely used in research labs and the structure and usability of the data (as implementations exist to validate) could be easily identified by the researcher participants.

Datasets	Format	# of Records	# of Attributes	# of requirements
DS1: Google cluster	CSV	2,000	9	56
DS2: OpenStack Neutron	log	2,000	18	62
DS3: OpenStack Nova	DB	2,000	22	44
DS4: BHPOBS ML	text	1,027	22	43

Table 2: Different datasets used in evaluating *iCAT* and their statistics

Participants. We have two types of participants, i.e., data owner participants and data user participants. To solicit participants, we have placed an advertisement on the university campus and also sent it to our industrial collaborators. The on-campus flyer requires that: **i)** participants should be able to pose clear requirements (e.g., how to use the data and what properties need to be preserved). **ii)** participants should be able to evaluate the usefulness and usability of the data after the experiments. The request sent to research collaborators indicates that: **i)** participants should be able to write their institutional privacy constraints and requirements that govern data sharing; **ii)** participants should be able to verify whether the final anonymized output of the data meets those requirements/constraints. As a result, we have recruited nine researchers from different research labs, and 14 participants from four industrial organizations. Table 3 summarizes the participants' experience level in percentage, where we categorize them based on their educational level and industrial experience.

Category	Research		Industry	
Expertise Level	M.Sc.	Ph.D.	Junior	Senior
Participants percentage	30.4%	8.6%	43.4%	17.6%
Overall percentage	39%		71%	

Table 3: Distribution of participants over the user experience levels

Procedures. We divided our experiments into four main data anonymization operations based on the used datasets and asked the participants to select one of them corresponding to their domain. After that, the participants had to input their requirements and interact with *iCAT* until the anonymization operation finishes. Finally, we asked the participants to fill a post-experiment questionnaire to report the correctness of data usefulness and the privacy constraints. Note that, we recorded the requirements entered by the participants to evaluate the effectiveness of *iCAT* as it will be explained next.

5.2 Effectiveness

The main goal of this experiment is to evaluate the quality of the requirements translation. Since this is a multi-class problem, we evaluate the effectiveness of our system as the percentage of the requirements that were correctly translated by *iCAT*. To this end, we manually investigated the recorded user's requirements and categorized the failures as follows: **i)** the privacy leakage/utility loss caused by both data owners/users through mistakenly choosing anonymized methods. **ii)** the failures caused by *iCAT* misrecognizing either the data owners or the data user's requirements. Figure 7.A and Figure 7.B demonstrate the effectiveness of the translation process from both data owner and user sides. Figure 7.C shows a detailed analysis of the failed requirements.

Results. The overall effectiveness of translating data owners' requirements is relatively high as shown in Figure 7.A; the lowest percentage of correctly translated results is 87.5%. This is justified as the ambiguity solver implemented by *iCAT* reduces the error rate through interactive communication with the users, where they can directly intervene in the case of uncertain requirements. On the other hand, the two main reasons of translation's failures are: **i)** the correctness of the ontology modeling; **ii)** NLP fails to translate when the user's input contains typos. The percentage of failures for both Ontology and NLP are presented in Figure 7.A in white and gray patterns. By comparing through the dataset, we also observe that the number of attributes affects the success rate of the requirements translation in the opposite manner. Hence, users need to express their requirements more precisely to differentiate between different attributes.

Similarly to the previous experiment, the ambiguity solver contributes to the high accuracy in the translation of data users' requirements as shown in Figure 7.B. Besides the aforementioned two main reasons, we observed that data user participants often fail to understand the mapping between anonymization primitives suggested by *iCAT*'s ambiguity solver and their utility requirements. This lesson has led us to add a pop-up message showing an example of each primitive in order to guide the user and avoid selecting the wrong suggestion.

Figure 7.C shows our analysis results about the failed requirements. We can only observe privacy loss from data owners' side due to a miss in the ontology modeling, which has been fixed afterward. Utility loss could be caused at both data owners and data users' sides due to an incorrect translation of data owners' requirements and the misinterpreting of the anonymization methods by data users. Some no-translation requirements are due to typos in the input requirements. We will discuss those issues and how to address them in the following section.

5.3 Usability

The usability of *iCAT* is evaluated based on two questionnaires. The first follows the standardized usability questionnaires [2] and consists of 19 questions. It provides the evaluation of the users' satisfaction towards the services provided by the tool (e.g., whether this tool converges the views and bridges the gaps between

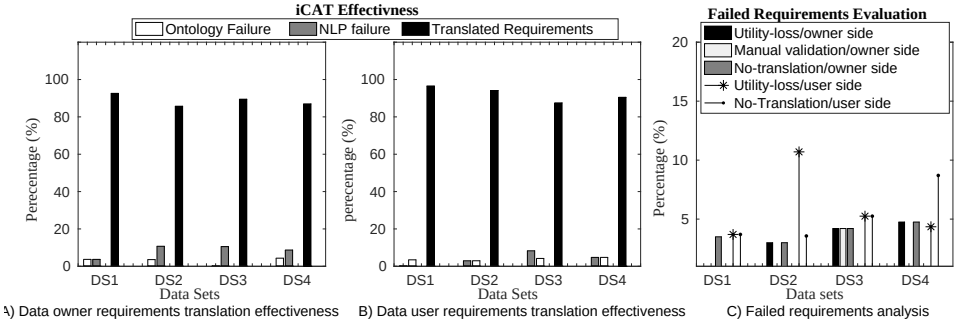


Fig. 7: The effectiveness of requirements translation

data owners and users). The second surveys the sensitivity of the attributes and the trust-level in different actors used to propose privacy/utility access control mechanism for different attributes anonymization.

Results. Table 4 shows a summary of our main evaluation criteria and the average rating out of seven. The results show that the data users are extremely positive by the fact that they are part of the anonymization process through expressing their requirements. On the other hand, the data owner participants from industry clearly show interests in this tool because they can have different anonymization levels of the same input data instead of the encrypt/hide policy which they currently use. Data users also report that the tool requires some privacy expertise, especially when it comes to deal with the ambiguity solver. As mentioned before, to this issue, we have revised our design by adding concrete examples for the anonymization primitives to make them easier to understand.

Category	Question	Score/7
Ease of use, interactivity and user friendly	It was simple to use *iCAT*	6.3
	I can effectively complete my work using *iCAT*	5.2
	I am able to complete my work quickly using *iCAT*	4.8
	I am able to efficiently complete my work using *iCAT*	5.45
	I feel comfortable using *iCAT*	5.7
	It was easy to learn to use *iCAT*	4.2
	I believe I became productive quickly using this system	6.4
	The interface of this system is pleasant	6.5
	like using the interface of this system	6.6
Errors detecting, reporting and recovery	*iCAT* gives error messages to fix problems	5.7
	I recover easily/quickly when I make a mistake	5.8
iCAT does not need support/ background to use	It is easy to find the information I needed	4.4
	The information provided for *iCAT* is easy to understand	3.5
	The information is effective in completing the tasks	3.6
	The information organization on *iCAT* screens is clear	5.7
This system has all the functions and capabilities I expect it to have Comment		6.1
The information provided with this system is clear (e.g., online help and other documentation)		NA
The overall satisfaction	I am satisfied with how easy it is to use *iCAT*	5.3
	I am satisfied with this system	6.2

Table 4: Usability results based on questionnaire designed following [2]

The second questionnaire is an online form and the results of this questionnaire are shown in the table of Figure 8. We applied the marginal distribution and drew the trend of each attribute and actor as shown in Figures 8.A and

8.B. In general, we can observe that the attributes and actors are associated with different sensitivity levels. The attribute *Time*, *ID*, *Constant* and *Numbers* have similar data sharing strategy; internal actors could have low privacy and high utility results, while competitors would be only provided with high privacy and low utility data. The main reason is those attributes are not as sensitive as personally identifiable information, but still can leak information that can be used to stage security attacks. Attribute *IP* and *Numbers* (salary in our survey) are considered to be sensitive attributes for all level actors who prefer to apply at least level 2 anonymization on them. This can be due to sharing policies or cultural background which makes them less willing to share the information carried by those attributes. Figure 8.B confirms the trust levels of the actors through the levels of anonymization methods they are mostly assigned. Internal auditors are mostly granted with level 1 anonymization only, while competitors could only get level 6 anonymization results. External auditors and researchers (generally under NDA) share similar trusted levels. This shows the participants share similar visions related to the internal auditor and competitors and consider the external auditors and researchers harmless.

Attribute	Actor	Level1	Level2	Level3	Level4	Level5	Level6
Time	I	95%				5%	
	E	45%	38%	6%	6%		5%
	R	25%	50%	10%	5%		10%
	C	5%			5%	20%	70%
ID	I	80%	5%	5%		10%	
	E	5%		70%		20%	5%
	R	50%	5%	5%	10%	20%	10%
	C	10%				25%	65%
String	I	55%	5%	40%			
	E			70%	15%	15%	
	R	25%		60%	5%	10%	
	C			20%	25%	55%	
IP	I	75%	20%		5%		
	E		35%	15%	20%	20%	5%
	R		40%		40%	10%	10%
	C					25%	75%
Constant	I	40%	40%	20%			
	E		55%	20%	10%	5%	10%
	R	45%	10%	30%	5%	10%	
	C	25%	5%			15%	55%
Number	I	60%	30%			5%	5%
	E	25%	50%	5%			20%
	R	5%	50%	5%	20%		20%
	C		5%			45%	55%

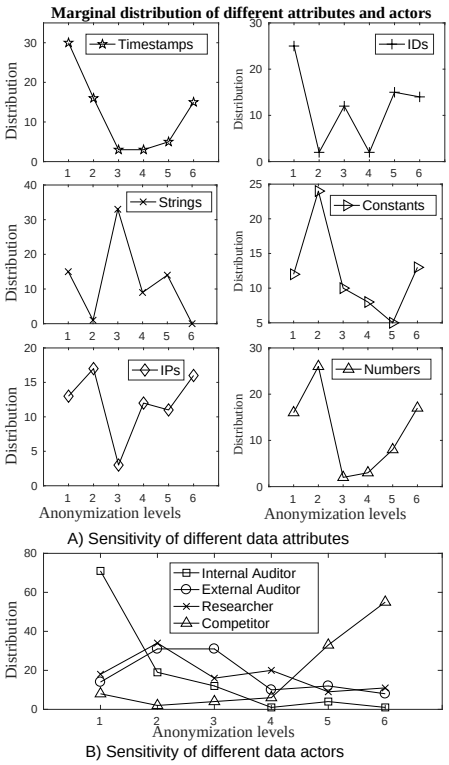


Fig. 8: Sensitivity questionnaire and results analysis

5.4 Efficiency

In order to evaluate the overhead from different modules of *iCAT*, we measure the time, memory and CPU consumption.

Results. Figure 9 shows the time, memory and CPU consumption of the data anonymization process according to the four datasets. We measure the aforementioned resource consumption according to four different events: **i)** E1: Data loading and pre-processing; **ii)** E2: Anonymization space and access control matrix generation; **iii)** E3: Ontology mapping; **iv)** E4: NLP translation. We also evaluate the resource consumption of anonymization. The first three results in Figure 9 are the overhead at the data owner side and the last two results are for the data user. From data owner side, beside the onetime effort to load the data, other operations have negligible consumptions. The overhead resulted from the last two events at the data user side is related to the use of NLP server and the anonymization primitives' implementation, which are both out of our control.

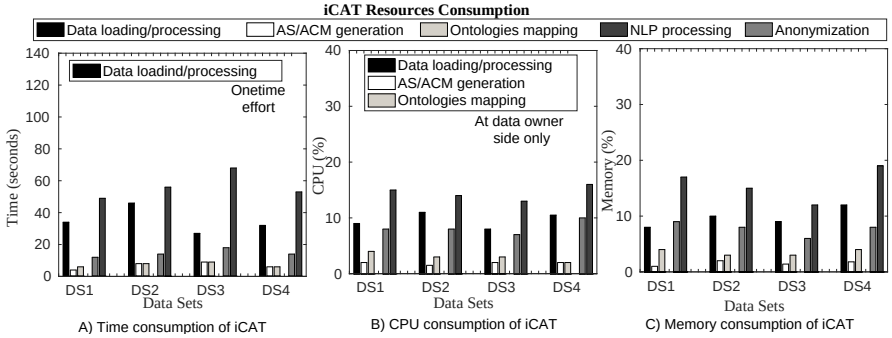


Fig. 9: The resources consumption of *iCAT*

6 Discussions

Compositional analysis. A well-known issue in anonymization is that releasing multiple views of the same data may breach privacy since an adversary can combine them. However, by the definition of our anonymization space lattice, whatever levels inside its 'privacy-up' region can be safely released, because all those views contain strictly less information than the specified privacy level (in fact, those views may be derived from the latter) so combining them lends the adversary no advantage. If, however, the data user is mistakenly assigned different privacy levels at different time, then he/she can potentially combine those views to gain more information. However, the anonymization space lattice makes it easy for the data owner to see exactly what he/she will gain (i.e., the GLB of those levels) and take appropriate actions.

Business-case. Nowadays data is becoming the most valuable asset and the determiner of success in many aspects. We believe *iCAT* can be used to provide "data anonymization as a service" in which the data owner sets the desired privacy level for each (type of) data user, without worrying about their utility

requirements. Afterward, the data users can query the tool in an interactive manner without any intervention from the data owner. The data owner can be sure that the privacy is preserved, whereas the data users can obtain as many anonymized views of the data as needed for different analyses.

Privacy analysis . *iCAT* does not propose any new anonymization primitive, but relies on the correctness of existing solutions. The privacy/utility level of *iCAT* output will be exactly the same as that of the anonymization primitives being used. However, it is possible that *iCAT* may mistakenly translate the data owner requirements and map them to unsafe levels. Therefore, in our design, the data owner-side requirement translation is only intended as a suggestion, which requires further validation by the data owner.

Tricking iCAT. As the data user and owner requirements are enforced independently by iCAT, the data user cannot influence iCAT to use a primitive that breaches the data owner's requirements. This is enforced as follows: i) during requirements translation, the ontologies for the data owner and user, respectively, are stored and used separately; *ii)* iCAT, by design, does not allow the data owner to publish the dataset until a privacy level is assigned to each data attribute (either by processing requirements through NLP or manually).

Data linkage. We emphasize that such a limitation, de-anonymizing a given dataset using publicly available data, is not due to iCAT as we mentioned earlier in our threat model (Section 2.1). Nonetheless, using iCAT, the data owner will have the flexibility to assign a privacy level for each data attribute and for each data user. As a result, the data owner can always specify a higher privacy level that is more resistant to linkage attacks (e.g., randomization) for less trusted users or more sensitive attributes.

Ontologies learning. As we mentioned in our experiments, we have reported requirements translation failures due to missing ontologies matching. We believe a major opportunity here is to add a feedback module that learns the new ontologies from both data owners and data users' responses. We consider this feedback module as future work.

7 Related Work

This section reviews existing works and their limitations.

Cryptography-based Anonymization Tools Most of the existing tools in this category use cryptography-based anonymization primitives, such as prefix-preserving, hashing and permutation. Existing tools in this category are used to anonymize the network traces and mainly anonymize the TCP header. Some of those tools support live interfaces anonymization. Table 5 compares those tools according to the anonymized fields (e.g., IP, header, port, etc.) and the anonymization primitives they use. As shown in the table, unlike *iCAT*, none of those tools can support all the attributes or anonymization primitives (let alone the flexibility for customization).

Replacement-based Anonymization Tools The existing tools in this category deal mainly with log files and anonymize data by replacing the sensitive

Tool	Anonymized Fields					Anonymization Primitive					
Name	NF fields	IP	Port	Header	Payload	Pref-Pres	Hiding	Permutation	Truncation	Hashing	Shifitng
AnonToo [8]	✓	✓				✓	✓	✓		✓	
CANINE [16]	✓	✓	✓			✓	✓	✓	✓		✓
CoralReef [18]	✓	✓	✓			✓	✓		✓		
Flaim [22]	✓	✓	✓			✓	✓	✓		✓	✓
IPsumdump [6]		✓		✓		✓					
NFDump [12]		✓				✓					
SCRUB [25]		✓		✓	✓		✓	✓			✓
TCPanon [9]					✓		✓				
tcpdpriv [11]		✓		✓	✓	✓	✓	✓	✓		
TCPmkpub [19]		✓		✓	✓	✓	✓		✓		
TCPurify [7]		✓			✓		✓	✓	✓		

Table 5: Comparing existing network data anonymization tools. The symbol ✓ indicates that the proposal offers the corresponding feature.

attributes (e.g., passwords, IPs, paths) in the log with some values predefined by the user in the so-called rule-file or generated using deterministic cryptography algorithms. The rule file contains patterns used by the tool to perform pattern matching and the conversion state of the anonymization can be stored in a look-up table. Table 6 compares between these tools in terms of anonymized fields, anonymization primitives used and how the mapping is achieved. This category of anonymization provides a higher utility output, compared to the first category, because it preserves some property of the original data (e.g., equality, format, order, etc.). However, this also leaves the door open for de-anonymization attacks, known as semantic attacks (e.g., frequency analysis, injection and shared text matching attacks). Moreover, those tools are generally not user-friendly and require knowledge about conducting tool-based search patterns and managing the conversion state of the anonymized data.

Tool	Anonymized Fields						Anonymization Primitive					Mapping	
Name	Number	Path	ID	string	IP	Timestamp	Hiding	Substitution	Randomization	Hashing	Shifitng	Look-up table	algorthim
Camouflage [13]	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Loganon [23]	✓		✓	✓	✓	✓		✓				✓	
Log-anon [23]			✓		✓			✓				✓	
Flaim [22]			✓	✓	✓	✓		✓	✓	✓	✓		✓
NLM [14]		✓	✓	✓	✓		✓		✓				✓
bsmpseu [1]	✓	✓	✓	✓	✓	✓		✓	✓		✓		✓

Table 6: Comparing different features of existing replacement anonymization tools. The symbol ✓ indicates that the proposal offers the corresponding feature.

8 Conclusion

We presented in this paper *iCAT*, a novel anonymization tool that brings customization and interactivity to the data anonymization process to bridge the existing gap between the data owners and the data users. Our tool leveraged existing anonymization primitives in a systematic fashion based on the novel concept of anonymization space. It also improved the usability by providing users with the means to express their requirements using natural languages and found for them the best-fit combination of anonymization primitives inside the anonymization space using our ontology-driven NLP approach. Finally, *iCAT* proposed a new privacy/utility access control model that allow involving the data user in the anonymization process without compromising the data owner's privacy requirements. The main limitations of our work and the corresponding future directions are as follows. First, we have mainly focused on the relational model in this paper and we believe our tool can be extended to handle other data models. Second, currently the ontology modeling is done offline, and implementing a feedback module that allow for ontologies learning from users'

requirements can further improve the performance and minimize the use of the ambiguity solver. Third, we have focused on network data, and extending our model to cover more applications is left as future work.

Acknowledgment: The authors thank the anonymous reviewers for their valuable comments. This work is partially supported by the Natural Sciences and Engineering Research Council of Canada and Ericsson Canada under CRD Grant N01823 and by PROMPT Quebec.

References

1. Konrad Rieck . Pseudonymizer for solaris audit trails, 2018. Available at: `http://www.mlsec.org/bsmpseu/bsmpseu.1`.
2. A. Assila, H. Ezzedine, et al. Standardized usability questionnaires: Features and quality focus. *Electronic Journal of Computer Science and Information Technology: eJCIST*, 6(1), 2016.
3. E. D. Bell and J. L. La Padula. Secure computer system: Unified exposition and multics interpretation, 1976.
4. D. E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, May 1976.
5. T. Donnellan. *Lattice Theory*. Pergamon press, Oxford, 1968.
6. Eddie Kohler.
7. Ethan Blanton.
8. M. Foukarakis, D. Antoniades, S. Antonatos, and E. P. Markatos. Flexible and high-performance anonymization of netflow records using anontool. In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, pages 33–38. IEEE, 2007.
9. Franceesco Gringoli.
10. Google. Traces from requests processed by Google cluster management system, 2019. Available at: `https://github.com/google/cluster-data`.
11. Greg Minshall of Ipsilon Networks. Tcpdpriv, 2005. Available at: `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`.
12. P. Haag. Nfdump. *Available from World Wide Web: http://nfdump. sourceforge. net*, 2010.
13. IMPREVA. Camouflage data masking, 2018. Available at: `https://www.imperva.com/products/data-security/data-masking/`.
14. Kayaalp, M., Sagan, P., Browne, A.C., McDonald. Nlm-scrubber, 2018. Available at: `https://scrubber.nlm.nih.gov/files/`.
15. C. Kreibich. Design and implementation of netdude, a framework for packet trace manipulation. In *Proc. USENIX/FREENIX*, 2004.
16. Y. Li, A. Slagell, K. Luo, and W. Yurcik. Canine: A combined conversion and anonymization tool for processing netflows for security. In *International conference on telecommunication systems modeling and analysis*, volume 21, 2005.
17. C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.
18. D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy. The coralreef software suite as a tool for system and network administrators. In *Proceedings of the 15th USENIX conference on System administration*, pages 133–144. USENIX Association, 2001.

19. R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, 2006.
20. Rules for the protection of personal data inside and outside the EU. Gdpr, 2018. Available at: `https://ec.europa.eu/info/law/law-topic/data-protection_en`.
21. R. S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, 1993.
22. A. J. Slagell, K. Lakkaraju, and K. Luo. Flaim: A multi-level anonymization framework for computer and network logs. In *LISA*, volume 6, pages 3–8, 2006.
23. Sys4 consults. a generic log anonymizer, 2018. Available at: `https://github.com/sys4/loganon`.
24. UCIMLR. Burst Header Packet flooding attack on Optical Burst Switching Network Data Set, 2019. Available at: `https://archive.ics.uci.edu/ml/datasets/`.
25. W. Yurcik, C. Woolam, G. Hellings, L. Khan, and B. Thuraisingham. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007*, pages 49–56. IEEE, 2007.

Appendix

The following details each module of *iCAT* as shown in Figure 6.B.

A) Data Loading and Processing (DLP). This module is used to load the data, and enables filtering and cleansing operations. This module consists of following sub-modules:

Data Processing: This sub-module enables performing data pre-processing and adjustment operations. It can also automatically detect all data attributes and their types, which are needed by the Anonymization Space Manager to build the anonymization space lattice.

Data Filtering: This sub-module deploys several algorithms that can be automatically and manually used to filter and remove records from data (e.g., column deletion, row deletion, searched deletion and frequency deletion).

B) Requirements Interpreter (RI). This module translates the data owner's and data user's requirements into data attributes types and anonymization primitives. It consists of the following three sub-modules:

Requirements Parser: It takes the English statement and transforms them into a set of requirements using the Stanford CoreNLP. Then, it processes and filters those requirements using the POS tool.

Requirements Mapper: This sub-module takes the parsed requirements and communicates with the Method-Ontology and the Type-Ontology databases in order to map each requirement into the related attribute type and then the corresponding anonymization primitives.

Ambiguity Solver: This sub-module is mainly responsible of communicating with the user (i.e. data owner or data user) through the Interactive Communicator (IC) sub-module in order to solve any ambiguity that occurs at the Requirement Mapper sub-module.

C) iCAT Manager.

Identity Access Management and Permission Granter (IPG): This module associates the data user identity with the privacy-level specified by the data owner, which is needed to determine the anonymization sub-space assigned to him based on privacy-up principle.

Interactive Communicator: This sub-module is mainly responsible for interacting with the data owner or data user and handles the communications between them and the RI module.

I/O Manager: This module is responsible for configuring the data source from where the data is fetched (e.g. from a file system or a database) and the loading of the actual data to be anonymized.

D) Anonymization Space Manager. This module is mainly responsible of generating the anonymization space and implementing the access control mechanism over the anonymization space for the data user. This module consists of the following sub-modules:

Anonymization Space Builder (ASB): This sub-module automatically builds the entire anonymization space, which consists of all available combination of anonymization primitives for each data attribute based on its type. Building the anonymization space lattice is detailed in Section 2.3. The resulting anonymization-space lattice will be stored in the Access Control database.

Anonymization Controller: This module implements the access control mechanism over the anonymization space for the data user. It receives the utility-level from the data user and perform an intersection/masking operation between the privacy level and utility level in order to determine the allowed combinations of anonymization primitives. It also ensures that the Data Anonymizer only accesses the allowed anonymization primitives for the user.

E) Data Anonymizer. This module is mainly responsible for anonymizing the data with the respect to the trust-level assigned to the users. It is designed in a building-blocks manner such that if there exist new or more efficient anonymization primitives they can be easily integrated into *iCAT*. This module holds the following sub-modules:

Anonymization Primitives: This sub-module holds the implementation of all existing anonymization algorithms corresponding to the 12 anonymization primitives discussed in Section 2.

Anonymization Mapper: This sub-module is responsible of creating a mapping file that maps the plain-text data into their anonymized values for later recognition purposes (e.g., if hashing is used to anonymize IP addresses, a file contains the original IP addresses and their hashes are created).