View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Electronic Communications of the EASST (European Association of Software Science and...

Electronic Communications of the EASST
Volume 75 (2018)

43rd International Conference on Current Trends in Theory and Practice of Computer Science

-

Student Research Forum, 2017 (SOFSEM SRF 2017)

On Privacy and Utility while Improving Software Quality

Fayola Peters

14 pages

Guest Editors: Anila Mjeda
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On Privacy and Utility while Improving Software Quality

Fayola Peters[1]

[1] fayolap@acm.org, http://www.fayolapeters.com/
Lero - The Irish Software Research Centre
University of Limerick, Ireland

Abstract: Software development produces large amounts of data both from the process, as well as the usage of the software product. Software engineering data science turns this data into actionable insights for improving software quality. However, the processing of this data can raise privacy concerns for organizations, which are obligated by law, regulations and polices, to protect personal and business sensitive data. Early data privacy studies in sub-disciplines of software engineering found that applying privacy algorithms often degraded the usefulness of data. Hence, there is a recognized need for finding a *balance* between privacy and utility. A survey of data privacy solutions for software engineering data was conducted. Overall, researchers found that a combination of data minimization and obfuscation of data, produced results with high levels of privacy while allowing data to remain useful.

Keywords: Data privacy, Software Engineering Data Science, Cross project defect prediction, Effort estimation, Test coverage, Debugging.

1 Introduction

Software engineering data scientists turn the large amounts of data into actionable insights for improving software quality [MZ13, KZDB16]. This data can be personal and business sensitive. Therefore, data *controllers* and *processors* need to vet their *data supply chain* in order to ensure that they are in compliance with privacy laws and regulations, prior to *processing* and sharing their data (Italic terms are defined in Table 1).

Due to the 'push' for open science and reproducibility in scientific experiments, researchers in academia and industry (controllers and/or processors) are encouraged to share their data and models when their work is accepted for publication. As a result, their data supply chain will expand to include *recipients* (data scientists) who will in turn apply what was learned to their projects and share their work. However researchers, particularly of proprietary projects are more cautious about sharing their data. This is because extracting project data from an organization is often very difficult due to the personal and business sensitivity associated with the data. For example, in 2009, Menzies et al. [PM12, PMGZ13, PML15] reported that NASA divided its independent verification and validation (IV&V) work between multiple contractors who all worked on similar projects. Every three years, all the contracts were subject to competitive bidding. Hence, those contractors were very reluctant to share data about (say), the effort associated with finding mission-critical errors in manned spacecraft, as that data could be used against them during the bidding process [PM12, PMGZ13, PML15].

Table 1: Definitions of privacy related and other relevant terms.

Terms	Definitions
Personal Data (Personally Identifiable Information)	Any information relating to an identified or identifiable natural person (data subject); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person [Eur16].
Processing	Any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction [Eur16].
Controller	The natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law [Eur16].
Processor	A natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller [Eur16].
Recipient	A natural or legal person, public authority, agency or another body, to which the personal data are disclosed, whether a third party or not [Eur16]. A recipient can be legitimate, seeking solutions to improve software quality, or malicious, seeking sensitive data to cause harm.
Data Supply Chain	A life-cycle for data that propagates and procures data on behalf of the corporation. The supply chain looks at data, the inputs and outputs of data across the company, across systems, across platforms, across organizations and manages it as an asset. (Source: https://www.inetsoft.com/business/solutions/definition_of_data_supply_chain/).
Sensitive Data	Information that is protected against unwarranted disclosure. Protection of sensitive data may be required for legal or ethical reasons, for issues pertaining to personal privacy, or for proprietary considerations. (Source: https://its.unc.edu/security/sensitive-data/)
Data Science	The knowledge of deriving meaningful outcomes from data [MKP+13]. Examples of data science projects include software defect prediction and effort estimation.
Anonymization	All personal identifiers must be filtered from the datasets to make it unlikely that the data controller or any other third party would reasonably be able to re-identify the data subject on the basis of the data at stake [VM14].

The General Data Protection Regulation (GDPR) [Eur16], has fostered privacy research in software engineering. Some areas of this research include privacy requirements [BA08, SKM14], cross project defect prediction [PM12, PMGZ13, PML15], effort estimation [QJZ+16], testing [TGX11, LGP14], and debugging [CCM08, BLJL11, CO11]. Researchers have drawn from the fields of privacy awareness and privacy-preserving data publishing for solutions, which seek to find the balance between privacy and utility, i.e., reduces the likelihood of recipients learning specific sensitive values from disclosed data while allowing that data to remain useful. For example with cross-project defect prediction where models are built with data from other sources [ZNG+09], the goal of a researcher and a legitimate recipient, is to find defects in a software system before deployment. While a malicious recipient's goal is to seek out sensitive information, which could allow them to learn about the complexity of source code as well as the level of maintenance required for a software product. Privacy solutions exploit the difference between the goals of a malicious recipient and the goals of defect prediction. Therefore, data *processors and/or controllers* (Table 1) could defend against malicious recipients while maintaining the usefulness of the data for legitimate recipients.

This paper surveys the privacy solutions for various sub-disciplines of software engineering. We explore how software engineering researchers find the balance between privacy and utility. In the case of cross-project defect prediction, this question was answered in the book chapter, "Becoming Goldilocks: Privacy and data sharing in just right conditions" [Pet16]. This paper expands on the book chapter to include the other sub-disciplines mentioned above.

This paper is organized as follows. Section 2 looks at four principles of data protection from the GDPR [Eur16]. Section 3 explores the identification and extraction of privacy requirements for the design and implementation of software products. Section 4 describes the background of privacy solutions from the field of Privacy-Preserving Data Privacy (PPDP) and briefly shows how they are applied in the sub-disciplines of software engineering surveyed here. Finally, the paper concludes in Section 5.

2 Legal Principles of Data Protection

In the article, "Becoming Goldilocks" [Pet16], three lessons were described about how to apply privacy algorithms to software defect data in single- and multi-controller contexts. These lessons were as follows:

1. **Do not share all your data**: It was found that limiting the amount of data shared, did not significantly reduce the utility of the remaining data.

2. **A little change is good**: Small changes in the values of the data, did not significantly reduce the utility of the data.

3. **Do not share what others have already shared**: If data is being combined among a group of controllers, and there are similarities in the data, it should only be shared by one controller. Avoiding duplicates in the resulting data did not significantly reduce the utility.

These lessons are aligned with the legal privacy principle of *data minimization*, which is included in as part of the GDPR [Eur16] and requires that data can only be processed, if the processing is adequate, relevant, and not excessive in relation to the specific purpose for which the data was collected [VM14]. Other legal principles include, *purpose limitation*, *transparency*, and *data security* and are reflected in the area of privacy requirements (Section 3). Purpose limitation constrains the secondary use of personal data for purposes incompatible with the specified purpose of the initial processing [VM14]. Transparency means that users must be informed about the reasons behind the processing, the categories of data being processed about them, and the "destiny" of their data [VM14]. While data security implies that confidentiality, integrity, and availability should be safeguarded as appropriate to the sensitivity of the information [VM14]. This is done by including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures [Eur16].

Sharing data with recipients in some cases can be incompatible with the initial purpose of data processing. To avoid this concern, techniques for data anonymization, and/or, data obfuscation [VM14] is applied to data. Any data which is anonymized is excluded from having to comply with GDPR and so, many of the solutions surveyed use data obfuscation methods in combination with data minimization in order to prevent the disclosure of sensitive data. In the

remainder of this paper, we explore how each of the software engineering sub-disciplines incorporated the data protection principles to balance privacy and utility. Note, that the use of the term *privacy* in this paper covers both personal and business sensitive data.

3 Identifying and Extracting Privacy Requirements

Dealing with privacy concerns in software engineering data science begins with privacy requirements [OCS+13]. It is difficult to specify what a privacy requirement is when designing and building a system [SKM14]. According to a survey done by Sheth et al. [SKM14], this may be due to a mismatch between how users and developers perceive privacy and the mitigation of privacy concerns. It is therefore important to ensure that the elicitation of privacy requirements, considers multiple sources, i.e., users, laws and regulations, and software developers, in an effort to reduce privacy concerns. In addition, the survey also showed that users tended to map privacy with security, while data sharing and data breaches were their biggest concerns. In contrast, developers preferred technical measures such as data anonymization and thought that privacy laws and policies were less effective. As shown in this paper, the sub-disciplines in software engineering research cover both areas separately.

Studies range from systematically extracting privacy requirements from natural language regulations [BA08], to eliciting them from data subjects [TBP+12, TBPN14]. In the latter case the privacy technique used is *selective disclosure*. This tends to focus on giving the data subject more control over their data in terms of who, when and how they share their data in order to receive benefits such as discounts or directions to an unoccupied parking spot.

In the former case, data protection laws and regulations are put in place in order to prevent the unauthorized use and disclosure of personal and sensitive information. Since these laws and regulations tend to be expressed in complex and ambiguous language, they need to be refined and analysed before they are implementable. Breaux et al. [BA08], presented a methodology to help elicit security and privacy requirements from these laws and regulations. Their solution extracted the access rights and obligations from regulation text. In addition, the access rights and obligations were seen as actions which stakeholders are allowed to or required to perform respectively. Therefore to support these actions developers must ensure that these systems also satisfy the constraints on the actions [BA08].

Regulations require software engineers to specify, design and implement systems that are accountable to and in compliance with laws and regulations. The goal of Breaux et al. [BA08] was to provide the software engineering community with a systematic method for specifying system requirements, which are accountable and compliant with the law. The systematic approach reduces susceptibility to security breaches and gives organizations the ability to demonstrate the transparency of their system, in other words, how their software systems comply with policies and regulations.

4 Privacy Preservation for Software Engineering Data

As mentioned in Section 1, there are many areas of research in software engineering. In this paper we focus on the subset, which studies data privacy. These are cross-project defect prediction,

effort estimation, test coverage, and debugging. Each of these are considered to be the utility of a software engineering data science task. Cross project defect prediction allows a project to build prediction models with data from other sources. This is because the project may be new and therefore little or no data has been collected [ZNG+09]. Effort estimation using algorithmic models, seeks to estimate in advance, the effort/staff resources required to complete a project on time [SSK96]. The last two areas of research studied here, testing and debugging, are concerned with the correctness and maintenance of software respectively. The data required to provide these insights may be considered as sensitive by data collectors.

The following sections offer a short primer on privacy-preserving data publishing, which details methods used for data privacy research in software engineering.

4.1 Privacy-Preserving Data Publishing

Privacy-preserving data publishing provides methods and tools for sharing useful data while preserving privacy [FWCY10]. The different aspects involved in PPDP research are: 1) privacy threats, 2) privacy techniques and algorithms, 3) privacy metrics and 4) utility [FWCY10]. The first two are described in Section 4.1.1 and Section 4.1.2 respectively. Privacy metrics are covered in Section 4.4, while throughout Section 4.2 and Section 4.3 there are examples of utility. Before elaborating on threats and techniques, it is important to know how data is defined in PPDP research. Consider the data in Table 2. Each row contains a record for a single individual or data subject. Each attribute in the set of attributes could have one or more of the following descriptions:

- Direct-identifiers: Attributes that explicitly identify an individual or project such as a name, social security number or filename.
- Quasi-identifiers: Attributes that in combination can be used to re-identify an individual target in a data set, for example, occupation, gender, and age.
- Sensitive Attributes: Attributes that we do not want an adversary to associate with an individual, in a data set.
- Dependent Attributes: Attributes used when evaluating the utility of data via classification.

Table 2: Example of personal data collected from data subjects.

Direct Identifier	Quasi Identifiers			Dependent Attribute	Sensitive Attribute
Name	Occupation	Gender	Age	Salary	Disease
Dawn	Lawyer	Female	35-40	100,000	Hepatitis
Jenny	Teacher	Female	20-25	45,000	HIV
Mary	Teacher	Female	20-25	37,000	Hepatitis
Joan	Teacher	Female	20-25	20,000	None
John	Nurse	Male	41	60,000	HIV
Stan	Mayor	Male	56	150,000	Hepatitis

4.1.1 Privacy Threats

Privacy threats are based on the top three attributes described above and are categorized as re-identification, membership disclosure, and sensitive attribute disclosure [BS08, FWCY10, GLS14]. When protecting data from privacy threats, the goal is to block malicious adversaries from uncovering personal and business sensitive data.

Re-identification occurs when an adversary with external information such as a voters list, can identify a data subject from data, which has been stripped of personally identifiable information. For example, in Table 1, the name column would be removed before release. Popular examples of this are the re-identification of William Weld from released health-care data [Swe02b] and Thelma Arnold from the AOL search data [BZH06].

Membership disclosure focuses on protecting a data subjects micro data. This privacy threat manifests itself if an adversary is able to confirm that the subjects data is contained in a particular dataset. For example, if an application only collects data, which contains information only on HIV patients, then it can be inferred that a target is HIV-positive [GLS14].

Finally, sensitive attribute disclosure occurs when a data subject is associated with information about their sensitive attributes, such as a disease. For example, from Table 2, we see that John is the only 41-year-old male nurse in the dataset, therefore an adversary with access to this data can infer with 100 percent certainty that John is HIV positive. While Jenny has a 33 percent chance of being disease free. Sensitive attribute disclosure is the main privacy threat addressed in the software engineering data science tasks explored in this paper.

4.1.2 Privacy Techniques

Techniques resulting in obfuscation are meant to "satisfy specified privacy requirements" [FWCY10]. There are six basic techniques for data obfuscation: 1) generalization, 2) suppression, 3) bucketization, 4) anatomization, 5) permutation, and 6) perturbation. Most PPDP methods and tools use one or more of these techniques and each one has its advantages and disadvantages. This section describes the three techniques (1, 2, and 6) used for data obfuscation in software engineering research to share data for data science tasks.

Generalization and Suppression: Generalization works by replacing exact numeric values with intervals that cover a sub-range of values; e.g., 17 might become 15...20 or by replacing symbols with more general terms; e.g., "date of birth" becomes "month of birth". While suppression, replaces exact values with symbols such as a star or a phrase like "dont know" [VBF+04]. According to Fung et al. [FWCY10], generalization and suppression tend to hide potentially important details in the quasi-identifiers, which can confuse classification.

Privacy models such as *k-anonymity*, *l-diversity*, and, *t-closeness* use generalization and suppression. With k-anonymity, each instance in the table is indistinguishable with k-1 other instances [SS98, Swe02a]. The limitations of k-anonymity are many-fold. It does not hide whether a given individual is in the database. In addition, in theory, k-anonymity hides uniqueness (and hence identity) in a data set, thus reducing the certainty that an adversary has uncovered sensitive information. However, in practice, k-anonymity does not ensure privacy if the adversary has background knowledge for a subject in a dataset [BS08]. The aim of l-diversity is to address the

limitations of k-anonymity by requiring that for each *QID group* [1], there are at least l distinct values for each sensitive attribute value. In this way, an adversary is less likely to "guess" the sensitive attribute value of any member of a QID group [MGKV06, MKGV07].

Work by Li et al. [LLV07], later showed that l-diversity was vulnerable to skewness and similarity attacks, making it insufficient to prevent attribute disclosure. Hence, they proposed t-closeness to address this problem. With t-closeness the distance between the distributions of a sensitive attribute in a QID group and that of the whole table are no more than a threshold t apart. The intuition is that even if an adversary could locate the QID group of a data subject, as long as the distribution of the sensitive attribute was similar to the distribution in the whole table, any knowledge gained by the adversary could not be considered as a privacy breach because the information would be considered as public. However, with t-closeness, information about the correlation between QIDs and sensitive attributes is limited and so causes degradation of data utility [LLV07]. Research in test coverage, and debugging use k-anonymity, extended with methods to mitigate its drawbacks [GCFX10, BLJL11].

Perturbation: The perturbation technique replaces original data values with synthetic data values, so that the statistical information computed from the perturbed data does not differ significantly from the statistical information computed from the original data. It is important to note that the perturbed data do not correspond to real world data subject. Also, methods used for perturbation include, additive noise, data swapping and synthetic data generation [FWCY10]. The drawback of these transforms is that they may not guarantee privacy. For example, suppose an adversary has access to multiple independent samples from the same distribution from which the original data was drawn. In that case, a principal component analysis could reconstruct the transform from the original to the perturbed data [GLK09]. Data swapping and synthetic data generation are the perturbation techniques used in defect prediction, effort estimation and test coverage [PM12, PMGZ13, TGX11, LGP14].

4.2 Data Obfuscation in Defect Prediction and Effort Estimation

Obfuscation offers privacy by simply changing data enough so that it is different from its original. Section 4.1.2 describes some basic techniques for obfuscation such as generalization and perturbation. Some studies chose to combine data minimization and obfuscation methods as part of a framework to preserve the privacy of personal and business sensitive data. Three of these are described in this section.

Data minimization reduces the dimensions of a data set via feature selection and instance selection. The result is a representation which best describes the entire data set. The representation is a collection of exemplars, which are used for data science tasks. The advantage of data minimization is that it avoids the drawbacks of large data set analysis such as large storage requirements and high computational expense. An additional benefit is that in the context of privacy, since only the exemplars are used for analysis, a data collector can decide not to share non-exemplars.

To accomplish minimization, filtering using Bayes rule and data clustering has been used. For cross-project defect prediction, Peters et al. [PMGZ13] presented CLIFF, an instance selection

[1] A QID group is a set of instances whose quasi-identifier values are the same because of generalization or suppression.

algorithm which used Bayes rule to rank instances in a dataset. CLIFF assumes that tables of data can be grouped-by the values of the dependent attribute. For example, for a table of defect data containing code metrics, instances are grouped according to defective or not defective dependent values. For each group, CLIFF finds the probability of each attribute value in defective and non-defective instances. The product of each resulting probability is calculated for each instance and those with the highest are selected as exemplars. With CLIFF, data minimization was achieved while maintaining utility. In terms of privacy, the authors used CLIFF as part of a framework, which combines it with another instance selection method and a data obfuscation method [PMGZ13, PML15].

To accomplish obfuscation, the authors presented MORPH, an instance mutator algorithm [PM12, PMGZ13, PML15] based on the perturbation privacy technique (Section 4.1.2). MORPH changes the numeric attribute values of each instance by replacing the original values with MORPHed values. MORPH did not change an instance such that it moves across the boundary between the original instance and instances of another dependent attribute value. For example, if the original instance is defective, its obfuscated version will not become non-defective. The overall results of the combination of CLIFF and MORPH showed that high levels of privacy was possible without damaging defect prediction [PM12, PML15].

For data privacy in effort estimation [QJZ+16], minimization occurs as a side-effect of formatting data into sub-classes or clusters. In other words, Qi et al. [QJZ+16], allowed for outliers to either be discarded or assigned to its nearest sub-class. Their method works as follows: Using effort estimation values, the authors first computed upper- and lower-bounds for each one. The bounds were determined by a tolerance error of 0.25. Since its possible for an instance to be covered by multiple sub-classes, the assignment of each instance is done in ascending order to avoid data imbalance. If there were sub-classes with a single instance, they were either assigned to their nearest sub-class or discarded. The instances which remained were then obfuscated using a modified version of MORPH for effort estimation.

The modified version of MORPH has two main differences from the original algorithm. First, Qi et al. [QJZ+16], used a nearest unlike neighbor selector to determine the boundary up to which an instance can be obfuscated. Because there were multiple sub-classes (more than 2), the selector used the ones which were just before and after as unlike neighbors. These were determined by the effort values. Second, in order to mitigate the possible influence of larger values from one attribute, the authors mapped the original data to intrinsically dimensional space. It is from this space that the nearest unlike neighbor is chosen. The overall results showed that these minimization and obfuscation methods provided privacy for effort estimation while maintaining comparable utility to the original data.

4.3 Data Obfuscation in Software Testing and Debugging

Other data privacy studies in software engineering focus on software testing and debugging [BLJL11, TGX11, LGP14, CCM08, CO11]. In this research area, two privacy concerns were raised: a subject's data was collected in the form of bug reports to guide maintenance after software is deployed and the outsourcing of software testing, which requires test-cases. Work published by Castro et al. [CCM08] sought to provide a solution to the problem of software vendors who need to include sensitive user information in bug reports in order to reproduce a

bug. To protect sensitive user information, they used symbolic execution along the path followed by a failed execution to compute path conditions. Their goal was to compute new input values unrelated to the original input. These new input values satisfied the path conditions required to make the software follow the same execution path until it failed.

As a follow-up, Clause et al. [CO11] presented an algorithm, which anonymized input sent from users to developers for debugging. Like the previous work by Castro et al. [CCM08], Clause et al. [CO11] aimed to supply the developer with anonymized input, which would cause the same failure as the original input. To accomplish this, they first used a novel "path condition relaxation" technique to relax the constraints in path conditions thereby increasing the number of solutions for computed conditions.

In the case of companies who did not wish to release actual cases for testing [BLJL11, TGX11, LGP14], they anonymize the test cases before releasing them to testers. Taneja et al. [TGX11] proposed PRIEST, a privacy framework. The privacy algorithm in PRIEST is based on data swapping where each value in a data set is replaced by another distinct value of the same attribute. This is done according to some probability that the original value will remain unchanged. This study [TGX11] followed the work done by Budi et al. [BLJL11]. Similarly, their work focused on providing obfuscated data for testing and debugging. They were able to accomplish this with a novel privacy algorithm called kb-anonymity. This algorithm combined k-anonymity with the concept of program behavior preservation, which guided the generation of new test-cases based on known cases and made sure that new test-cases satisfied certain properties [BLJL11]. The difference with the follow-up work by Taneja et al. [TGX11] was that the data-swapping algorithm maintains the original data and offered individual privacy by swapping values.

For Li et al. [LGP14], data clustering forms the basis of the combination of minimization and obfuscation in test coverage. The authors used a weight-based data-clustering algorithm, which partitions data according to information from program analysis. The information gained from program analysis, indicates how data is used by an application [LGP14]. Once the clusters were formed, centroids and associative rule mining computed and used as constraints to ensure that the centroids were representative of the general population of data in the cluster. The overall results showed that test coverage stayed within a close range of the level of coverage offered by the original data [LGP14].

4.4 Measuring Privacy

Privacy is not a binary step function where something is either 100 percent private or 100 percent not private. Rather it is a probabilistic process where we strive to decrease the likelihood that an adversary can uncover something that they should not know. Privacy metrics allows data collectors to know how easy it is for an adversary to "guess" the original data given the obfuscated data. There are two categories of privacy metrics in the literature: syntactic and semantic [BS08]. Algorithms such as k-anonymity and l-diversity use the syntactic measures. They consider the distribution of attribute values in the obfuscated data set. In comparison, the semantic metrics measure what an adversary may learn or the incremental gain in knowledge caused by the obfuscated data and uses distance measures such as the Earth Movers Distance, KL-divergence and JS-divergence to quantify this difference in an adversary's knowledge [BS08]. Semantic privacy metrics dominate in the sub-disciplines of software engineering. They include

Increased Privacy Ratio (IPR) [PM12, PMGZ13, PML15], *entropy* [CCM08, CO11] and *guessing anonymity* [TGX11, LGP14].

IPR is based on the adversarial accuracy gain, A_{acc} from the work of [BS08]. According to the authors definition of A_{acc}, it quantifies the ability of an adversary to predict the sensitive attribute value of a data subject. A_{acc} measures the increase in the adversary's accuracy after s/he observes an obfuscated dataset and compares it to the baseline from a trivially obfuscated dataset, which offers perfect privacy by removing either all sensitive attribute values or all the other quasi-identifiers. IPR, assumes the role of an adversary armed with some background knowledge from the original dataset and the obfuscated version of that data. When the adversary predicts the sensitive attribute value of a data subject, the original data is used to see if the prediction is correct. If it is, then this is considered to be sensitive attribute disclosure, otherwise it is not [PM12, PMGZ13, PML15].

Other privacy studies in software engineering data science [CCM08, CO11] have used entropy to measure privacy. Entropy (H) measures the level of uncertainty an adversary will have in trying to associate a target to a sensitive attribute [Sha01]. For example, if querying a dataset produces two instances with the same sensitive attribute value then the adversary's uncertainty level or entropy is zero. However, if the sensitive attribute values were different, the adversary had a 50 percent chance of associating the target with the correct sensitive attribute value. The entropy here is one bit, if there are only two possible sensitive values. In general, the entropy of a data set is $H = \sum^{|S|} p(s_i)|log_2 p(s_i)|$, where i=1, which corresponds to the number of bits needed to describe the outcome [Sha01]. Here S is the set of sensitive attribute values.

Guessing anonymity of an instance in an obfuscated dataset, is the number of guesses that the optimal guessing strategy of the adversary requires to correctly guess an instance used to generate the final obfuscated instance [TGX11]. This is measured as follows: First, a similarity matrix was used to show the similarity between instances in the original data and its final obfuscated version. The similarity is measured as the fraction of attributes whose values are the same between the original and obfuscated instances. A fraction of zero means that there is no similarity between instances, whereas one means that they are the same [TGX11, LGP14].

5 Conclusion

This article suggests that to find the balance between privacy and utility in different sub-disciplines of software engineering, data collectors need to consider two privacy solutions in combination: data minimization and obfuscation. This "marriage" maintains comparable utility and helps data collectors to back-up their data sharing decisions with the knowledge that a large percentage of their data is not shared and what is released, bares little resemblance to the original data. As more and more data is mined from the software engineering process, the combination of minimization and obfuscation can be used to confidently share data with other projects and organizations.

Acknowledgements: This work was funded supported, in part, by Science Foundation Ireland grant 13/RC/2094 and by the European Research Council (Advanced Grant 291652 - ASAP).

Bibliography

[BA08] T. Breaux, A. Anton. Analyzing Regulatory Rules for Privacy and Security Requirements. *Software Engineering, IEEE Transactions on* 34(1):5–20, Jan 2008.

[BLJL11] A. Budi, D. Lo, L. Jiang, Lucia. kb-anonymity: a model for anonymized behaviour-preserving test and debugging data. In *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*. PLDI '11, pp. 447–457. ACM, New York, NY, USA, 2011.

[BS08] J. Brickell, V. Shmatikov. The Cost of Privacy: Destruction of Data-mining Utility in Anonymized Data Publishing. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08, pp. 70–78. ACM, New York, NY, USA, 2008.

[BZH06] M. Barbaro, T. Zeller, S. Hansell. A face is exposed for AOL searcher no. 4417749. *New York Times* 9(2008):8, 2006.

[CCM08] M. Castro, M. Costa, J.-P. Martin. Better bug reporting with better privacy. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*. Pp. 319–328. ACM, New York, NY, USA, 2008.

[CO11] J. Clause, A. Orso. Camouflage : Automated Anonymization of Field Data. *Proceeding of the 33rd international conference on Software engineering*, pp. 21–30, 2011.

[Eur16] Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such data, and repealing Council Framework Decision 2008/977/JHA. 2016.
http://data.europa.eu/eli/dir/2016/680/oj

[FWCY10] B. C. M. Fung, K. Wang, R. Chen, P. S. Yu. Privacy-preserving Data Publishing: A Survey of Recent Developments. *ACM Comput. Surv.* 42(4):14:1–14:53, June 2010.

[GCFX10] M. Grechanik, C. Csallner, C. Fu, Q. Xie. Is Data Privacy Always Good for Software Testing? In *Proceedings of the 2010 IEEE 21st International Symposium on Software Reliability Engineering*. ISSRE '10, pp. 368–377. IEEE Computer Society, Washington, DC, USA, 2010.

[GLK09] C. Giannella, K. Liu, H. Kargupta. On the Privacy of Euclidean Distance Preserving Data Perturbation. 01 2009.

[GLS14] A. Gkoulalas-Divanis, G. Loukides, J. Sun. Publishing data from electronic health records while preserving privacy: A survey of algorithms. *Journal of Biomedical Informatics* 50:4 – 19, 2014. Special Issue on Informatics Methods in Medical Privacy.

[KZDB16] M. Kim, T. Zimmermann, R. DeLine, A. Begel. The Emerging Role of Data Scientists on Software Development Teams. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. Pp. 96–107. May 2016.

[LGP14] B. Li, M. Grechanik, D. Poshyvanyk. Sanitizing And Minimizing Databases For Software Application Test Outsourcing. *IEEE International Conference on Software Testing Verification and Validation*, 2014.

[LLV07] N. Li, T. Li, S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *2007 IEEE 23rd International Conference on Data Engineering*. Pp. 106–115. April 2007.

[MGKV06] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*. Pp. 24–24. April 2006.

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam. L-diversity: Privacy Beyond K-anonymity. *ACM Trans. Knowl. Discov. Data* 1(1), Mar. 2007.

[MKP+13] T. Menzies, E. Kocaguneli, F. Peters, B. Turhan, L. L. Minku. Data Science for Software Engineering. In *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13, pp. 1484–1486. IEEE Press, Piscataway, NJ, USA, 2013. http://dl.acm.org/citation.cfm?id=2486788.2487048

[MZ13] T. Menzies, T. Zimmermann. Software Analytics: So What? *IEEE Software* 30(4):31–37, July 2013.

[OCS+13] I. Omoronyia, L. Cavallaro, M. Salehie, L. Pasquale, B. Nuseibeh. Engineering Adaptive Privacy: On the Role of Privacy Awareness Requirements. In *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13, pp. 632–641. IEEE Press, Piscataway, NJ, USA, 2013.

[Pet16] F. Peters. Becoming Goldilocks: Privacy and data sharing in just right conditions. In Menzies et al. (eds.), *Perspectives on Data Science for Software Engineering*. Pp. 193 – 197. Morgan Kaufmann, Boston, 2016.

[PM12] F. Peters, T. Menzies. Privacy and Utility for Defect Prediction: Experiments with MORPH. In *Proceedings of the 2012 International Conference on Software Engineering*. ICSE 2012, pp. 189–199. IEEE Press, Piscataway, NJ, USA, 2012.

[PMGZ13] F. Peters, T. Menzies, L. Gong, H. Zhang. Balancing Privacy and Utility in Cross-Company Defect Prediction. *Software Engineering, IEEE Transactions on* 39(8):1054–1068, Aug 2013.

[PML15] F. Peters, T. Menzies, L. Layman. LACE2: Better Privacy-preserving Data Sharing for Cross Project Defect Prediction. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*. ICSE '15, pp. 801–811. IEEE Press, Piscataway, NJ, USA, 2015.

[QJZ+16] F. Qi, X. Y. Jing, X. Zhu, F. Wu, L. Cheng. Privacy preserving via interval covering based subclass division and manifold learning based bi-directional obfuscation for effort estimation. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Pp. 75–86. Sept 2016.

[Sha01] C. E. Shannon. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5(1):3–55, Jan. 2001.

[SKM14] S. Sheth, G. Kaiser, W. Maalej. Us and Them: A Study of Privacy Requirements Across North America, Asia, and Europe. In *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014, pp. 859–870. ACM, New York, NY, USA, 2014.

[SS98] P. Samarati, L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical report, CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States), 1998.

[SSK96] M. Shepperd, C. Schofield, B. Kitchenham. Effort Estimation Using Analogy. In *Proceedings of the 18th International Conference on Software Engineering*. ICSE '96, pp. 170–178. IEEE Computer Society, Washington, DC, USA, 1996.

[Swe02a] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10(5):571–588, Oct. 2002.

[Swe02b] L. Sweeney. k-anonymity: A model for protecting privacy. *IEEE Security And Privacy* 10(5):557–570, 2002.

[TBP+12] T. T. Tun, A. Bandara, B. Price, Y. Yu, C. Haley, I. Omoronyia, B. Nuseibeh. Privacy arguments: Analysing selective disclosure requirements for mobile applications. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*. Pp. 131–140. Sept 2012.

[TBPN14] K. Thomas, A. K. Bandara, B. A. Price, B. Nuseibeh. Distilling Privacy Requirements for Mobile Applications. In *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014, pp. 871–882. ACM, New York, NY, USA, 2014.

[TGX11] K. Taneja, M. , R. Ghani, T. Xie. Testing Software in Age of Data Privacy: A Balancing Act. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. ESEC/FSE '11, pp. 201–211. ACM, New York, NY, USA, 2011.

[VBF+04] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, Y. Theodoridis. State-of-the-art in Privacy Preserving Data Mining. *SIGMOD Record* 33:2004, 2004.

[VM14] Y. S. Van Der Sype, W. Maalej. On lawful disclosure of personal user data: What should app developers do? In *Seventh International Workshop on Requirements Engineering and Law*. Pp. 25–34. 2014.

[ZNG+09] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *ESEC/SIGSOFT FSE'09*. Pp. 91–100. 2009.