SNAKE challenge: Sanitization Algorithms under Attack

Tristan Allard, Louis Béziaud, Sébastien Gambs

SNAKE challenge: Sanitization Algorithms under Attack

Tristan Allard
Univ Rennes, CNRS, IRISA
Rennes, France
tristan.allard@irisa.fr

Louis Béziaud
Univ Rennes, CNRS, IRISA, UQÀM
Rennes, France
Université du Québec à Montréal
Montréal, Canada
louis.beziaud@irisa.fr

Sébastien Gambs
Université du Québec à Montréal
Montréal, Canada
gambs.sebastien@uqam.ca

ABSTRACT

While there were already some privacy challenges organized in the domain of data sanitization, they have mainly focused on the defense side of the problem. To favor the organization of successful challenges focusing on attacks, we introduce the Snake framework that is designed to facilitate the organization of challenges dedicated to attacking existing data sanitization mechanisms. In particular, it enables to easily automate the redundant tasks that are inherent to any such challenge and exhibits the following salient features: genericity with respect to attacks, ease of use and extensibility. We propose to demonstrate the main features of the Snake framework through a specific instantiation focusing on membership inference attacks over differentially-private synthetic data generation schemes. This instance of the Snake framework is currently being used for supporting a challenge co-located with APVP 2023 (the French workshop on the protection of privacy).

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; **Data anonymization and sanitization**; • **Software and its engineering** → *Software libraries and repositories.*

KEYWORDS

Challenge, anonymization, sanitization, attack, membership inference attack, synthetic data generation, differential privacy

ACM Reference Format:
Tristan Allard, Louis Béziaud, and Sébastien Gambs. 2023. SNAKE challenge: Sanitization Algorithms under Attack. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3614754

1 INTRODUCTION

Competitions and challenges are commonly used both in the machine learning community – for boosting the design and development of practical and efficient solutions to hard or new problems – and in the security community – for training purposes or for the evaluation of existing infrastructures. In contrast, in the privacy community, there is not a long tradition of holding such challenges. However, in recent years several competitions focusing on data sanitization algorithms (also called *data anonymization algorithms* or *privacy-preserving data publishing algorithms*) have been launched [1, 10, 15, 17, 20].

Some of them, like the 2018 *Differential Privacy NIST Challenge* [20], have focused primarily on the defense aspect. Their main requirements were that the proposed algorithms have to meet formal guarantees such as differential privacy [2], achieve high utility levels on real-life cases while being efficient enough to be run on today's off-the-shelf computer systems. Others, like the *Hide-and-Seek challenge* [10], the *INSAnonym competition* [1] or *PWSCUP* [17] additionally consider attacks on the sanitized datasets generated by the participants. More precisely, these competitions were generally composed of two phases, in which the first one is dedicated to the design of sanitization algorithms (usually focus on a data type and use case) while the second one usually consists in attacking the data sanitized with the algorithms developed during the first phase.

While the sanitization phases of past challenges have been successful, in the sense that they provided insights on the privacy/utility trade-offs or that it has lead to implementations of state-of-the-art or novel sanitization algorithms, the outcomes of the attack phases were usually more mitigated. For example, the organizers of the Hide-and-Seek challenge have reported attack results equivalent to random guesses [10]. The recent *Microsoft Membership Inference Competition* (MICO) [15] consisted only in an attack phase and is therefore an exception to this observation.

We believe that facilitating the organization of attack challenges over sanitization algorithms can greatly benefit the research community – essentially the privacy-preserving database and privacy-preserving machine learning fields – by stimulating research, contributing to strengthening the implementation of sanitization algorithms as well as by generating open source implementations of state-of-the-art attacks. This is precisely the overarching goal of the Snake framework. Snake allows managing and easy automation of the redundant tasks that are inherent to any such challenge (*e.g.*, preparing datasets or running the sanitization algorithms).

More precisely, Snake exhibits the following salient features. First, Snake is *generic* in the sense that it is agnostic with respect to the sanitization mechanisms under attack (*e.g.*, aggregated data, l-diverse partitioning algorithms or differentially private synthetic data generation), supports the specification of a wide range of parameters (*e.g.*, privacy levels and hyperparameters) and allows participants to compete in various adversarial settings (*e.g.*, background knowledge and success metrics). Second, Snake is *easy to use* both for participants and for organizers thanks to the Snakemake software [12]. This facilitates both the organization of challenges (for the organizers) and the creation of tasks for local tests during challenges (for the participants). Third, Snake is *extensible*

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614754

as it can support various challenge designs (*e.g.*, tracks, timelines, enabling code submission or not), threat models with respect to participants and the organizers (*e.g.*, colluding participants or not, covert organizers) as well as reproducibility features. As a result, we believe that SNAKE can help nicely complement the current sanitization competitions that focus on the sanitization part, thus resulting in a complete defense-attack pipeline. To the best of our knowledge, SNAKE is the first framework of its kind.

We propose to demonstrate the genericity, usability and extensibility properties of the SNAKE framework through a complete scenario focusing on *membership inference attacks* [5, 7, 8] over differentially-private synthetic tabular data generation schemes [13]. This instantation of the SNAKE framework is currently being used for supporting a challenge[1] co-located with APVP 2023 (the French workshop on the protection of privacy).

2 RELATED CHALLENGES

The closest related challenges to ours are the *Hide-and-Seek privacy challenge* [10] from the Van Der Schaar laboratory, the *INSAnonym competition* [1], the *PWSCup* [17] series of competitions as well as the *Microsoft Membership Inference Competition (MICO)* [15].

The three former challenges differ on the rules, the data and tasks they propose, but they follow the same two-phase structure: (1) a sanitization phase during which the participants implement a sanitization algorithm and run it on the given dataset and (2) an attack phase during which the sanitization algorithms are attacked. The attack phase of these competitions is often short (either by design or in practice) and the algorithms attacked are chosen by the participants. With the SNAKE framework, we aim at facilitating and thus strengthening the attack phase. Compared to Hide-and-Seek, INSAnonym or PWSCup, the sanitization algorithms attacked can be chosen from the literature and/or from the latest sanitization challenges (*e.g.*, the NIST differential privacy challenges) and the number of algorithms under attack can remain small. This gives the opportunity to stress-test state-of-the-art sanitization algorithms and helps participants to focus on their attack algorithm. In particular, we believe that the SNAKE framework complements nicely the other challenges dedicated to designing sanitization algorithms.

Finally, the recent MICO Competition differs from the other three challenges mentioned above in that it consists in a single attack phase. More precisely, it focuses on membership inference attacks, as does the first edition of SNAKE, but differs from SNAKE_1 on the algorithms attacked (classifiers rather than synthetic tabular data generators) and on the background knowledge given to participants (disclosure of the learned model rather than disclosure of only its outputs and hyperparameters). In addition, another difference between MICO and SNAKE_1 is the use of SNAKE to facilitate the creation and management of SNAKE_1. As the objective of SNAKE is precisely to facilitate challenges such as MICO, instantiating the SNAKE framework for obtaining a follow-up of MICO would both be easy and provide various benefits (*e.g.*, automation, reproducibility and participant-side tools).

Besides the use in academia, challenges could be also a way forward for companies to validate their approach in a sound manner. Such examples include the 2006 Netflix challenge, in which privacy issues were highlighted [18], or the 2018 Aircloak bug bounty in which a new class of privacy attacks was proposed [3]. SNAKE can be used to ease the organization of such events, strengthening data sanitization mechanisms and as a first step towards a systematic evaluation of commercial privacy solutions.

[1] https://www.codabench.org/competitions/879/

3 SNAKE FRAMEWORK

SNAKE is not a *one-size-fits-all* standalone tool that would integrate numerous and heterogeneous sanitization algorithms or scoring measures. This would both be non-exhaustive and lead to a heavy and complex tool. Rather, SNAKE consists of *a set of abstractions* and *a flexible technical foundation*. We describe these two levels below before describing in Section 4 SNAKE_1, the first edition of SNAKE, provided as a reference implementation.

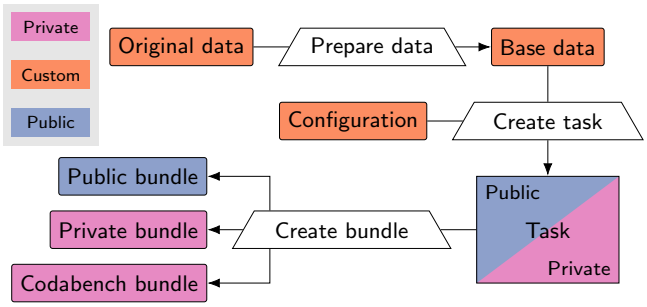

Figure 1: High level view of the workflow. Trapezoids represent rules (*i.e.*, programs) and rectangles data (*i.e.*, files).

Abstractions. The abstractions defined in SNAKE consist in a workflow and its functional specifications. Figure 1 describes the SNAKE workflow. The rule Prepare data creates the main dataset of the challenge, called Base data, from some existing source of data, called Original data. Each task of the challenge is created by Create task, as a set of Public and Private files, with the former shared to participants, and the latter used by the organizers for the evaluation of the attacks submitted by participants. The third rule Create bundle packages all the competition files into (1) a bundle that can be uploaded to the CodaBench competition platform [21] to instantiate the full competition, (2) a public bundle that contains all the data needed by participants (*e.g.*, available online during the competition) and (3) a private bundle that contains the solutions (*e.g.*, available to participants exclusively after the challenge).

Technical Foundations. SNAKE leverages (1) the Snakemake workflow management system [12] for its inner working and (2) the CodaBench competition platform [21] for all the features necessary for running a competition in real-life (*e.g.*, user registration, submission management, leaderboard). Snakemake is a bioinformatics workflow engine which provides a domain specific language implemented as an extension to Python to describes pipelines. Workflows are specified as a directed acyclic graph of rules which transform input files into target files, similar to *makefile* directives. SNAKE uses Snakemake to prepare all files required to define the challenge and outputs a "bundle" which can be uploaded to CodaBench to build a complete challenge. Thanks to the simple rule-based interface

provided by Snakemake, the computational overhead of organizing a challenge using Snake is negligible compared to without it.

Key properties. The design of the Snake framework adheres to the following key goals. These minimal set of properties are both necessary to the successful organization of attack challenges targeting sanitization algorithms and general enough to adapt smoothly to the advances of the field.

Genericity. The Snake workflow (see Figure 1) is high-level and *generic* enough to allow the design of most challenges in which participants attack the privacy properties of a sanitization algorithm. Indeed, any sanitization algorithm inputs prepared data and output sanitized data (*e.g.*, a perturbed model), and any task consists in a challenge available to the participants (public part) and in the answer to the challenge (private part). In particular, the definition of a task as a pair of public and private data allows the framework to support a broad range of attacks scenarios (*e.g.*, varying the background knowledge or sanitization algorithm attacked).

Extensibility. The Snake framework provides the two following levels of extensibility. First, it offers a simplistic challenge overview, which allows for further specification. For example, considering multiple attacks or allowing online attack evaluation is possible without deviating from the framework. Second, steps of the workflow are organized as Snakemake `rules` that (1) allow for generic input, output, script, execution environment, and parameters and (2) can be broken down into sub-rules if the need arises.

Usability. By leveraging the Snakemake workflow management, the Snake framework does not require strong development skills to be used and can be run on commodity hardware. It can thus be used easily by participants during a competition for building a complete local competition environment allowing local executions of the sanitization algorithms and of the participants' attacks. On the organizer side, existing challenges can easily be framed within Snake. Note that using Snake does not generate any computational overhead compared to preparing a challenge without using Snake. Moreover, the Snake framework outputs a packaged bundle compliant with the CodaBench platform, which allows for user-friendly participation (documentation, live ranking and automated scoring using a Python script).

Reproducibility. Snake allows a challenge to be reproduced in full, thanks to the functionalities offered by Snakemake. For example, inputs, outputs, execution environments, parameters and scripts are tracked through hashing. Random seeds used by Create task are saved in the private part of the related task. They can be kept by the organizers and disclosed to participants after the competition. The genericity and reproducibility properties allow the framework to support different *threat models* over both participants and organizers. For example, our framework is designed to allow participants to verify locally that their scores, given by the organizer, are indeed correct. This can simply be done by the participants, through local evaluations of the submissions, performed after the distribution of the private part of the tasks. Preventing any tampering of the private part of tasks could be done for instance by storing their hash in the public bundle and by letting participants check that they match with the private parts distributed by the organizers after the competition.

4 SNAKE$_1$: THE FIRST EDITION

Snake$_1$ is the first challenge based on the Snake framework. It focuses on *membership inference attacks* [5, 7, 8]) over differentially private synthetic data generation algorithms [13]), is co-located with APVP 2023, which is taking part during summer 2023. Figure 2 describes the workflow of the challenge for a single task[2].

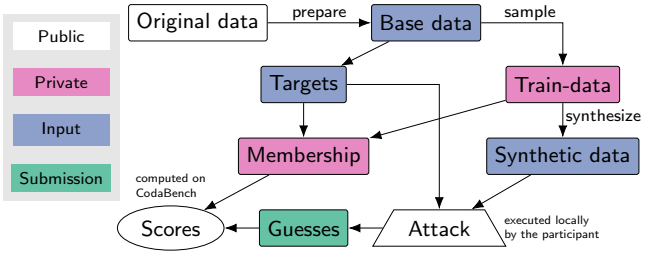

Figure 2: High level view of the workflow implementation for the first edition. Rectangles represent data with rules as edge labels.

First, the data used by the challenge is *prepared* as described afterwards. Each execution of a sanitization algorithm takes as input a private dataset consisting of random *samples* from the base dataset. The attack used as input the *synthetic* data, a set of *targets* for which membership is to be guessed, the *base data* and the parameters of the sanitization algorithm. The participants then perform their attack locally[3] and submit their guesses to the CodaBench space of Snake$_1$. The CodaBench platform scores the tasks by comparing the submission with the (private) ground truth.

The following is an excerpt of the task creation rule (with irrelevant parts removed), with `task_prep` creating most of the public and private files, while synthetic data is generated by a separate rule that uses a specific Conda environment.

```
rule task_prep:
  input: rules.prepare_data.output.data,
  output:
      targets=...,
      train=".../{gen}_{eps}_train.feather",
      seed=...,  # for reproducibility
      truth=...,  # private answers
  script: "scripts/task_prep.py"
```

Attacks algorithm. Each team has to design an attack algorithm as follows.

Input. Attacks algorithms are provided with (1) the *synthetic dataset* generated by an execution of the targeted sanitization algorithm over a private dataset, (2) the *targets* to attack, (3) the *base dataset* from which the private dataset is sampled, and (4) the *parameters* of the execution of the sanitization algorithm attacked;

Output. The output of the attack algorithm is a real number in $[0, 1]$ indicating the predicted probability of each target being within the private dataset or not.

[2]Full implementation available at https://github.com/snake-challenge/snake1.
[3]Participants can use the framework to generate their own versions of the tasks locally and therefore can evaluate their submission on their side before submission.

We detail below the computation of the private datasets, the targets, the sanitization algorithms as well as the attack success measure.

Base dataset and private datasets. SNAKE$_1$ makes use of the publicly available *EPI CPS Basic Monthly* data provided by the Economic Policy Institute [9]. The CPS dataset is divided in years in which a yearly dataset contains more than 10^6 records and 125 columns[4]. A record contains information about a single individual in a household. Our base dataset is built by using a pre-processed sample of columns and rows from the original dataset. Full details are available in the competition online documentation. From this base dataset, we generate one private dataset for each parameterized sanitization algorithm attacked.

Targets and background knowledge. In SNAKE$_1$, any household that contains at least 5 individuals might be a target, with the target consisting of the full set of records of the household. SNAKE$_1$ considers the following background knowledge about each target. The adversary knows (1) the exact records of the household targeted, and (2) the full base dataset[5]. The adversary is also given the information about the sanitization algorithm targeted as well as the parameters used for the executions and has access to its implementation. However, the randomness generated internally during the execution of the algorithm is unknown to the adversary.

Sanitization algorithms under attacks. The sanitization algorithms under attack during SNAKE$_1$ are differentially-private synthetic data generation algorithms. More precisely, we have selected a set of algorithms according to the following two criteria: technical soundness assessed by a rigorous peer-selection process (*e.g.*, published at top-tier conferences or winner of a dedicated competition) and available open-source implementation. In particular, we have used the implementations available in the Reprosyn package [16]. Except for parameters related to differential privacy, we use the default values set in their implementations. The `PrivBayes` algorithm [23] generates synthetic data by capturing the underlying distribution of the private data through a specific Bayesian network. The `MST` algorithm [13] is a generalization of the NIST-MST algorithm, which has won the 2018 NIST Differential Privacy Synthetic Data challenge. It generates synthetic data by perturbing the marginals that capture the data distribution through the Gaussian mechanism and by post-processing them through the `Private-PGM` algorithm [14]. The `PATE-GAN` algorithm [11] is an extension of *generative adversarial networks* [4] based on the *private aggregation of teacher ensembles* framework [19].

Success measure. The success of a team is computed by first measuring the successes of its attack on each parameterized algorithm attacked (*e.g.*, `MST` parameterized by $\epsilon = 1.0$ and $\delta = 10^{-5}$) and second by aggregating the success measures in a single final score. The success of a given attack for a given parameterized algorithm is evaluated based on the well-known *membership advantage measure* [22]. The number of targets must be at the same time large enough for obtaining a sufficiently stable estimation and small enough for being practical (*e.g.*, $r = 100$).

[4]Full description available at https://microdata.epi.org/variables/.

[5]It provides teams with the knowledge of the population distribution commonly assumed in membership inference attacks.

Execution environment. The design, coding, and executions of attacks are performed locally by each team with its own resources. As a result, there is no restriction on the computing environment used to develop the attacks and the choice of the programming language and the available resources are unconstrained. Note however that we provide to participants the SNAKE$_1$ Python environment.

5 DEMONSTRATION

We propose to demonstrate the key features of the SNAKE framework on several datasets along the following three scenarios.

Organizer side: creating a competition. We demonstrate how one can create a competition with SNAKE, highlighting the benefits of SNAKE for automating the creation of tasks and bundles.

Participant side: overview of the first edition. We use the first edition of SNAKE to give an interactive view of participating in a running competition (*e.g.*, accessing the tasks generated by SNAKE, running a baseline attack and submitting the results).

Extension: code submission. We demonstrate how the framework can be extended with minimal changes using the example of the EKANS toy competition (available on CodaBench[6]) which is the "reverse" of SNAKE$_1$. The objective of EKANS is to propose a way to sample targets that are the easiest to attack. EKANS asks participants to submit *Python code* in charge of sampling "good" targets for membership inference attacks. The CodaBench platform runs the code submitted and performs a basic attack on the sampled targets using the TAPAS package [6]. The score of the participant is the success score of the attack. We believe that the EKANS competition helps illustrate the genericity of the SNAKE framework.

6 CONCLUSION

The SNAKE framework is dedicated to facilitating the organization of challenges stress-testing sanitization algorithms. SNAKE allows to easily automate the redundant tasks that are inherent to any such challenge and exhibits the following salient features: genericity with respect to attacks, ease of use and extensibility.

We illustrate SNAKE through SNAKE$_1$, a specific instantiation focusing on membership inference attacks over differentially-private synthetic data generation schemes. We propose in this paper to showcase SNAKE from the points of view of the organizer and of participants. Overall, we hope that the SNAKE challenges can help gain understanding of the empirical privacy guarantees of sanitization algorithms and of their parameters and pave the way to a greater democratization of sanitization algorithms providing strong privacy guarantees.

ACKNOWLEDGMENTS

We would like to warmly thank the students, engineers and researchers that have helped us to design the SNAKE challenge. This work is partially supported by the ANR 22-PECY-0002 IPOP (Interdisciplinary Project on Privacy) project of the Cybersecurity PEPR, the Canada Research Chair program, a Discovery Grant from NSERC as well as CIFAR through a CIFAR-MILA-IVADO-MITACS Catalyst Grant.

[6]https://www.codabench.org/competitions/746/

REFERENCES

[1] Antoine Boutet, Mathieu Cunche, Sébastien Gambs, Benjamin Nguyen, and Antoine Laurent. 2020. DARC : Data Anonymization and Re-identification Challenge. In *RESSI 2020 - Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information*. Nouan-le-Fuzelier, France. https://inria.hal.science/hal-02512677

[2] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407. https://doi.org/10.1561/0400000042

[3] Andrea Gadotti, Florimond Houssiau, Luc Rocher, Benjamin Livshits, and Yves-Alexandre de Montjoye. 2019. When the Signal is in the Noise: Exploiting Diffix's Sticky Noise. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security '19)*. 1081–1098.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144. https://doi.org/10.1145/3422622

[5] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. 2019. Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. In *Proceedings of the 19th Privacy Enhancing Technologies Symposium (PETS '19)*. 232–249. https://doi.org/10.2478/popets-2019-0067

[6] Florimond Houssiau, James Jordon, Samuel N. Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum D. Mole, Camila Rangel Smith, and Lukasz Szpruch. 2022. TAPAS: a Toolbox for Adversarial Privacy Auditing of Synthetic Data. *CoRR* abs/2211.06550 (2022).

[7] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. 2022. Membership Inference Attacks on Machine Learning: A Survey. *Comput. Surveys* 54, 11s (2022), 235:1–235:37. https://doi.org/10.1145/3523273

[8] Jihyeon Hyeong, Jayoung Kim, Noseong Park, and Sushil Jajodia. 2022. An Empirical Study on the Membership Inference Attack against Tabular Data Synthesis Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM '22)*. 4064–4068.

[9] Economic Policy Institute. 2023. Current Population Survey Extracts. https://microdata.epi.org/ Version 1.0.40.

[10] James Jordon, Daniel Jarrett, Evgeny Saveliev, Jinsung Yoon, Paul W. G. Elbers, Patrick Thoral, Ari Ercole, Cheng Zhang, Danielle Belgrave, and Mihaela van der Schaar. 2020. Hide-and-Seek Privacy Challenge: Synthetic Data Generation vs. Patient Re-identification. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track (NeurIPS '20) (Proceedings of Machine Learning Research, Vol. 133)*, Hugo Jair Escalante and Katja Hofmann (Eds.). PMLR, 206–215. http://proceedings.mlr.press/v133/jordon21a.html

[11] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *Proceedings of the 7th International Conference on Learning Representations (ICLR '19)*.

[12] Johannes Köster and Sven Rahmann. 2018. Snakemake - a scalable bioinformatics workflow engine. *Bioinformatics* 34, 20 (2018), 3600. https://doi.org/10.1093/bioinformatics/bty350

[13] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *Journal of Privacy and Confidentiality* 11, 3 (2021). https://doi.org/10.29012/jpc.778

[14] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning (ICML '19) (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4435–4444. http://proceedings.mlr.press/v97/mckenna19a.html

[15] Microsoft. 2023. MICO: Membership Inference Competition. https://github.com/microsoft/mico

[16] Callum Mole. 2023. Reprosyn. https://github.com/alan-turing-institute/reprosyn

[17] Takao Murakami, Hiromi Arai, Koki Hamada, Takuma Hatano, Makoto Iguchi, Hiroaki Kikuchi, Atsushi Kuromasa, Hiroshi Nakagawa, Yuichi Nakamura, Kenshiro Nishiyama, Ryo Nojima, Hidenobu Oguri, Chiemi Watanabe, Akira Yamada, Takayasu Yamaguchi, and Yuji Yamaoka. 2023. Designing a Location Trace Anonymization Contest. *Proceedings of the 23rd Privacy Enhancing Technologies Symposium (PETS '23)* (2023), 225–243. https://doi.org/10.56553/popets-2023-0014

[18] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of Large Sparse Datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P '08)*. IEEE Computer Society, 111–125. https://doi.org/10.1109/SP.2008.33

[19] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *Proceedings of the 5th International Conference on Learning Representations (ICLR '17)*.

[20] Diane Ridgeway, Mary F Theofanos, Terese W Manley, and Christine Task. 2021. *Challenge design and lessons learned from the 2018 differential privacy challenges*. https://doi.org/10.6028/nist.tn.2151

[21] Zhen Xu, Sergio Escalera, Adrien Pavão, Magali Richard, Wei-Wei Tu, Quanming Yao, Huan Zhao, and Isabelle Guyon. 2022. Codabench: Flexible, easy-to-use, and reproducible meta-benchmark platform. *Patterns* 3, 7 (2022), 100543. https://doi.org/10.1016/j.patter.2022.100543

[22] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *Proceedings of the 31st IEEE Computer Security Foundations Symposium (CSF '18)*. 268–282. https://doi.org/10.1109/CSF.2018.00027

[23] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Transactions on Database Systems* 42, 4 (2017), 25:1–25:41. https://doi.org/10.1145/3134428