

Membership Inference Attacks on Aggregated Time Series with Linear Programming

Antonin Voyez, Tristan Allard, Gildas Avoine, Pierre Cauchois, Elisa Fromont, Matthieu Simonin

▶ **To cite this version:**

Antonin Voyez, Tristan Allard, Gildas Avoine, Pierre Cauchois, Elisa Fromont, et al.. Membership Inference Attacks on Aggregated Time Series with Linear Programming. SECRYPT 2022 - 19th International Conference on Security and Cryptography, Jul 2022, Lisbon, France. pp.193-204, 10.5220/0011276100003283 . hal-03726234

HAL Id: hal-03726234
https://hal.science/hal-03726234

Submitted on 18 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Membership Inference Attacks on Aggregated Time Series with Linear Programming

Antonin Voyez[1,3][a], Tristan Allard[1][b], Gildas Avoine[1,2][c],
Pierre Cauchois[3], Elisa Fromont[1,5,4][d], Matthieu Simonin[4]
[1] *Univ Rennes, CNRS, IRISA, France*
[2] *INSA Rennes, CNRS, IRISA, France*
[3] *ENEDIS, France*
[4] *Inria, IRISA, France*
[5] *IUF (Institut Universitaire de France), France*
{*antonin.voyez, tristan.allard, gildas.avoine, elisa.fromont*}*@irisa.fr,*
matthieu.simonin@inria.fr, pierre.cauchois@enedis.fr

Keywords: Privacy, Time Series, Membership Inference Attack, Subset Sum Problem, Gurobi

Abstract: Aggregating data is a widely used technique to protect privacy. Membership inference attacks on aggregated data aim to infer whether a specific target belongs to a given aggregate. We propose to study how aggregated time series data can be susceptible to simple membership inference privacy attacks in the presence of adversarial background knowledge. We design a linear programming attack that strongly benefits from the number of data points published in the series and show on multiple public datasets how vulnerable the published data can be if the size of the aggregated data is not carefully balanced with the published time series length. We perform an extensive experimental evaluation of the attack on multiple publicly available datasets. We show the vulnerability of aggregates made of thousands of time series when the aggregate length is not carefully balanced with the published length of the time series.

1 INTRODUCTION

Private organizations and public bodies worldwide are nowadays engaged in ambitious *open data* programs that aim at favoring the sharing and re-use of data (The World Wide Web Foundation, 2017). Opening data is rooted in century-old principles[1] and, with today's data collection capacities, is expected to yield important and various benefits (Deloitte, 2017; The World Wide Web Foundation, 2017; The European Data Portal, 2015; OECD, 2020) (e.g., driving public policies, strengthening citizens trust into governments, fostering innovation, facilitating scientific studies). Personal data are part of the open data movement[2] and an ever-increasing quantity of information about individiduals is made available online, often in an aggregated form (e.g., averaged, summed up) in order to comply with personal data protection laws (e.g., the European GDPR[3], the Californian CCPA[4]) and (try to) avoid jeopardizing individual's right to privacy. In particular, datasets associating various attributes, coming from the same data provider or not, are especially valuable for understanding complex correlations (e.g., gender, socio-economic data, and mobility (Gauvin et al., 2020)).

Time series are considered as valuable data for open data publication. They are unbounded sequences of measures performed over a given population at a given sampling rate. For example, current smart-meters enable the collection of fine-grained

[a] https://orcid.org/0000-0003-3248-1497
[b] https://orcid.org/0000-0002-2777-0027
[c] https://orcid.org/0000-0001-9743-1779
[d] https://orcid.org/0000-0003-0133-3491

[1] See, e.g., the XV^{th} article of the Declaration of the Rights of Man and of the Citizen (1789).

[2] See, e.g., the GovLab Data Collaboratives effort (https://datacollaboratives.org/).

[3] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679

[4] https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121

electrical power consumption with one measure every 30 minutes. Electrical consumption records are then published under a variety of aggregated time series[5]. Today, the publication of such datasets is often encouraged by laws such as the EU directive 2019/1024[6] imposing to the states members to set up open data programs for various information, including energy consumption (Directive 2019/1024 Article 66).

The main protection measure, called *threshold-based aggregation* in the following, consists in guaranteeing that the number of individuals per aggregate is above a given threshold. The threshold is chosen with the goal to preserve the subtle equilibrium between individual's privacy and data utility (Büscher et al., 2017). Threshold-based aggregation may come with additional protection measures (e.g., suppression of outsiders, noise addition) but is often considered as being sufficient in real-life, even for sensitive data (e.g., power consumption in France[7]), provided that the threshold is high enough.

However, and despite its wide usage in real-life, the lack of formal privacy guarantees of threshold-based aggregation, together with the length of the time series allowed to be published, does not allow to formally claim that personal data is indeed protected. In particular, powerful attackers may exist that hold most of the raw data (if not all) and may use them for performing *membership inference attacks*.

This new threat model is relevant in real-life situations, when the raw data are illegitimately accessed after a (partial or total) disclosure, either accidentally by careless employees or following an active attack. Such threats must not be overlooked today as witnessed by the ever-increasing list of major data breaches. For example, the Ponemon Institute reported in 2020 a strong rise in the number of insider-caused cybersecurity incidents (+47% between 2018 and 2020)[8]. As another example, the *2021 Verizon Data Breach Investigation Report* shows the large diversity of industries concerned (from finance and healthcare to energy or transportation) and the distribution of attack patterns having led to the breaches[9].

Membership inference attacks form an important part of the privacy attacks. They aim to infer whether a specific target is present or absent from a dataset using various methods such as statistical tests (Dwork et al., 2017) and machine learning (Rigaki and Garcia, 2020). The presence of a target in a published aggregate is a valuable knowledge for an attacker when the aggregate is, or can be, publicly associated to other, possibly sensitive, attributes.

Contributions. In this paper, we introduce (Section 2) a new threat model where the attacker has access to the published aggregates (e.g., averages, sums) that are, or can be, associated to additional attributes, and to the disclosed raw times series. He/she then aims to determine whether (the time series of) a given target is associated to the additional attribute(s). This can actually be framed as a membership inference attack. We then show (Section 3) that this membership inference attack can be reduced to the subset sum problem, which can be solved as a constraint programming problem when data consist of time series. Each time point serves as a constraint to efficiently solve a subset sum problem. Based on this approach, we perform (Section 4) experiments on real-life data using the Gurobi solver, and show that our membership inference attack is successful in most cases. Our findings demonstrate that breaking privacy when aggregate-based techniques are used to publish time series is much easier than expected. We analyse the limit cases to eventually help publishers enforcing individuals' privacy.

Notation	Description
I	Set of individuals
$\mathcal{T}$	Set of timestamps
$\mathcal{S}$	Full multiset of time series
$\mathcal{S}^{\mathcal{A}}$	Multiset of time series participating in the aggregates
$\mathcal{A}$	Vector of aggregates published
θ	Time budget of the attacker
p	Pool size (number of solutions asked to the solver)
$\mathcal{P}$	Set of solutions found by the attacker
$\mathcal{G}$	Adversarial guesses
$\lvert\cdot\rvert$	Cardinality

Table 1: Notations

2 PROBLEM STATEMENT

2.1 Time Series

A time series is a timestamped sequence of data. In the following, we consider fixed-length univariate integer-valued time series. We represent the full mul-

[5]See, e.g., the open data portal of the French national electrical network manager (https://data.enedis.fr).

[6]https://eur-lex.europa.eu/eli/dir/2019/1024/oj

[7]https://www.legifrance.gouv.fr/codes/id/LEGISCTA000034381202/2017-04-08

[8]https://www.observeit.com/cost-of-insider-threats/

[9]https://www.verizon.com/business/resources/reports/dbir/2021/data-breach-statistics-by-industry/

tiset of time series as a $|I| * |\mathcal{T}|$ matrix $\mathcal{S}$ (all notations are defined in Table 1) where the $|I|$ rows represent the time series of each individual and the $|\mathcal{T}|$ columns represent the timestamps (i.e., a sequence of integers from 1 to $|\mathcal{T}|$ for simplicity, see eq. 1). Each cell $\mathcal{S}_{i,t} \in [d_{min}, d_{max}]$ thus represents the value of the i^{th} individual at the t^{th} timestamp. Without loss of generality, and unless explicitly stated, we consider in the rest of the paper that $d_{min} = 0$.

$$\mathcal{S} = \begin{bmatrix} \mathcal{S}_{1,1} & \cdots & \mathcal{S}_{1,|\mathcal{T}|} \\ \vdots & \vdots & \vdots \\ \mathcal{S}_{|I|,1} & \cdots & \mathcal{S}_{|I|,|\mathcal{T}|} \end{bmatrix} \quad (1)$$

2.2 The Publishing Environment

Publisher. As shown in Figure 1, the publishing environment considered consists of a *publisher* and an *attacker* (described in Section 2.3). The publisher collects from a set of *individuals* I a multiset of time series (where each individual produces a single fixed-length time series) associated to, possibly personal and/or sensitive, additional attributes contextualizing the time series (e.g., income, house surface, socio-demographic information, electrical appliances).

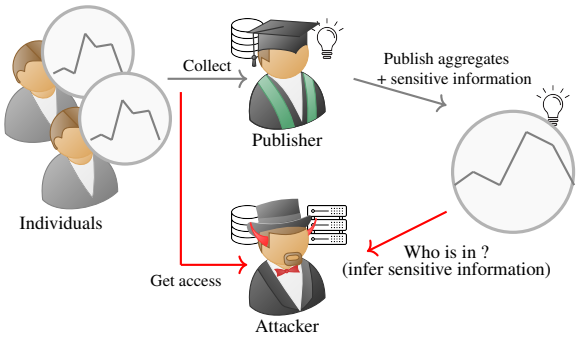


Figure 1: The SUBSUM Attack

Published Aggregates. The publisher selects a sub-matrix $\mathcal{S}^{\mathcal{A}}$ of $\mathcal{S}$ and computes and publishes a vector of aggregates $\mathcal{A}$ over $\mathcal{S}^{\mathcal{A}}$. In this paper, we focus on aggregates that are *sums* (we use the two terms indistinctly below). Note that all sum-based aggregation methods (e.g., average) can also be tackled by our proposed attack with minor changes in the attack constraints. Each aggregate is computed for a specific timestamp t by summing the values of each individual in $\mathcal{S}^{\mathcal{A}}$: $\mathcal{A}_t = \sum_{\forall i} \mathcal{S}^{\mathcal{A}}_{i,t}$. The set of individuals selected in $\mathcal{S}^{\mathcal{A}}$ represents the *ground truth*. We model the ground truth as a vector of probability telling, for each individual of $\mathcal{S}$, which one is part of $\mathcal{S}^{\mathcal{A}}$. The probabilities of the i^{th} individual is set to 1 if and only if the i^{th} individual is in $\mathcal{S}^{\mathcal{A}}$, and to 0 otherwise. It is worth noting that the sequences of aggregates $\mathcal{A}$ usually come along with the timestamps and the number of time series aggregated.

Published Additional Attributes. To make the published aggregates more valuable, they are usually associated with additional information that characterizes the subpopulation in the aggregates. Such characterizing attributes can be for example the average surface of the households in the aggregates, the average age of the individuals, the incomes, the region, etc. To illustrate this point, a practical case is the publication of the average daily electricity consumption (time series) per region (attribute).

2.3 Threat Model

The usual threat model obviously considers an attacker who aims to retrieve from the published aggregates, protected through various privacy-preserving measures, the identity (or associated valuable information) of the people concerned by these data (Jayaraman et al., 2021) (Bauer and Bindschaedler, 2020) (Pyrgelis et al., 2018). A step further consists in considering a threat model where the attacker has access both to the (published) aggregated time series and to the (leaked) raw time series (while the additional attributes remain secret and the goal of the attack remains unchanged). Two points should then be discussed: the relevance of this scenario and the attacker's motivation to perform an attack in such a scenario.

Due to legal requirements and widespread security best practices, publishers usually store direct identifiers and raw time series in separate databases[10] applying for example traditional pseudonymization schemes. Dedicated access control policies, archival rules, or security measures can then be applied to each database (e.g., retention period limited to a couple of years for fine-grained electric consumption time series). Isolating direct identifiers from time series indeed prevents direct re-identifications from malicious employees or external attackers exploiting a leak[11]. The same security principles might apply to the storage of the additional, possibly sensitive, attributes. Note that variants of the above context can be easily captured by our model. An example of a vari-

[10] We use the term *database* for simplicity without any assumption on the underlying storage mechanisms.

[11] The wealth of information contained in pseudonymized time series still allows massive re-identifications, but publishers seldom apply destructive privacy-preserving schemes to the time series that they store for preserving their utility.

ant, common in real-life, is when the additional attributes come from another data provider collaborating with the publisher for computing cross-database statistics. Whatever the scenario, this results in a set of databases, isolated one from the other, either for security reasons in order to mitigate the impacts of data leaks, or simply because multiple data providers collaborate. Because data leaks are common, whether they are due to insider attackers or to external ones, we believe that considering that raw data may eventually leak is a conservative approach that deserves to be explored. Obviously, if the identifiers, the time series, and the additional attributes all leak together, attacking the published aggregates is useless. However, in cases where only the raw times series leak, attackers need to deploy membership inference attacks for learning the association between a target individual (or more) and its attributes. In the following, we consider this threat model. We show on a real-life example that an attacker is able to efficiently retrieve the attributes of a target using linear programming.

More formally, the adversarial background knowledge consists of the full multiset of time series $\mathcal{S}$. Given his/her background knowledge $\mathcal{S}$, the vector $\mathcal{A}$ of timestamped aggregates, and the number of time series aggregated, the attacker outputs a vector of guesses $\mathcal{G}$ over the participation of each individual to the vector of aggregates. The vector of guesses is similar to the ground truth vector s.t. the i^{th} value of the vector of guesses represents the probability of the participation of the i^{th} individual to the aggregate. The closer to the ground truth the guesses, the better for the attacker.

3 THE SUBSUM ATTACK

The SUBSUM attack is based on solving the subset sum problem (Kellerer et al., 2004) over the vector $\mathcal{A}$ of aggregates given the set of time series $\mathcal{S}$.

3.1 Constraint Programming and Solvers.

Constraint programming (CP) is a computation paradigm that can be used to model mathematical optimization problems such as the subset sum problem. The problem is modeled by a set of variables and formulas, called *constraints*, that describe and bound the problem, and an optimization objective. The objective can be to prove (or disprove) the feasibility of the problem, or to find real solutions matching the constraints, if any. *Integer Programming* (IP) focuses on problems where all the variables are integers. An integer programming set of constraints for solving the subset sum problem has the following form :

$$\sum_{i \leq n} (X[i] \cdot V[i]) == s \tag{2}$$

where s is the sum to reach, V is a vector of integer values and X a vector of Boolean values indicating whether to add a specific weight to the bag (i.e., set to 1) or not (i.e., set to 0).

A wide variety of solvers are able to solve such problem (e.g., Gurobi (Gurobi Optimization, 2020), Choco (Charles Prud'homme et al., 2016), OR-Tools (Laurent Perron, 2019)).

3.2 Attack Formulation

The SUBSUM attack solves a well known NP-hard problem and would be highly inefficient and not scale in the general case. This is, in part, due to the fact that for a given timestamp, an important set of individuals may have similar values which may further increase the combinatorics. By exploiting the different time points (with different aggregate values but originating from the same set of individuals), the solver is able to prune the unsatisfactory solutions much faster and converge, with enough constraints, to possibly unique solutions if all individuals do not have identical values as another individual in all time points.

Algorithm 1 thus combines multiple subset sum problems, one for each aggregate (each timestamp), into a single set of IP constraints. Let X be a vector of $|\mathcal{S}|$ Boolean variables denoting the time series that are (possibly in case of multiple solutions) part of the aggregate. We define one constraint per aggregate in $\mathcal{A}$:

$$\sum_{i \leq n} (X[i] \cdot \mathcal{S}_{i,t}) == \mathcal{A}_t \tag{3}$$

Once these constraints are defined, we define one final constraint representing the attacker knowledge of the number $|\mathcal{S}^{\mathcal{A}}|$ of individuals present in the aggregates.

$$\sum_{i \leq n} X[i] == |\mathcal{S}^{\mathcal{A}}| \tag{4}$$

Our objective is to find less than p solutions (ideally $p = 2$) to infer meaningful information about the aggregate members within a reasonable time (the wall time θ). The solver runs until one of the following conditions is satisfied: (1) all the existing solutions are found, (2) the maximum number of solutions, p, is reached, (3) the time budget θ is exhausted. The solver outputs a set $\mathcal{P}$ of up to p candidate solutions together with a status code. Each solution is a vector

Algorithm 1: SUBSUM attack

Input: The set of time series ($\mathcal{S}$), the aggregates ($\mathcal{A}$), the time budget (θ), the number of solutions to look for (p)
Output: The solutions found ($\mathcal{P}$), the guesses ($\mathcal{G}$), the status code (`status`)

```
set_time_limit(θ)
set_pool_size(p)
for i ∈ |S| do
  | add_variable(X_i, {0,1})
end
for t ∈ |A| do
  | add_constraint(∑_{∀i∈S} X_i S_{i,t} == A_t)
end
solve()
P = get_solutions()
G = to_probability_vector(P)
status = get_status()
return P, G, status
```

Functions used in the algorithm :

`add_variable(bounds)`: Define an integer variable bounded to a given set of values.

`add_constraint(constraint)`: Define a constraint binding the model.

`set_pool_size(`p`)`: Define an upper bound p on the number of solutions to search for.

`set_time_limit(`θ`)`: Set the time budget θ of the solver.

`solve()`: Solves the problem.

`get_status()`: Get the status of the solver after its termination. We assume that at least the two following statuses are available : `OPTIMAL` (i.e., either p solutions are found or less than p solutions are found but no more solutions exist), `TIMELIMIT` (i.e., the time budget θ is over) and `INFEASIBLE` (i.e., the model cannot be solved).

`get_solutions()`: Get the solutions found by the solver (if any).

of probability, similarly to the ground truth vector, indicating the individuals from $\mathcal{S}$ that, when summed up together on the right timestamp(s), solve the given set of constraints X. Finally, the attacker computes the frequency of each individual in the set of solutions in order to obtain his guesses $\mathcal{G}$.

Parameter	Value
$\mathcal{S}$	{ISSDA-30m, London-30m}
$\lvert\mathcal{S}\rvert$	{1000, 2000, 3000, 4000, 4500}
$\lvert\mathcal{S}^{\mathcal{A}}\rvert$	$\{0\%, 10\%, \ldots, 100\%\} \times \lvert\mathcal{S}\rvert$
$\lvert\mathcal{A}\rvert$	$\{0\%, 10\%, \ldots, 100\%\} \times \lvert\mathcal{S}\rvert$
θ	{1000, 2000, 4000, 8000, 86400}
p	{2, 100}

Table 2: Values of the parameters used in our experiments.

4 EXPERIMENTS

We perform an extensive experimental survey of the attack using two open datasets: the *CER ISSDA* dataset (CER, 2012) and the *London Households Energy Consumption* dataset. Our results show that the SUBSUM attack is highly efficient (e.g., absolute success) when it's requirements are met, i.e., when the vector of aggregates published contains a sufficient number of aggregates (sufficient number of constraints) and when the attacker has sufficient computing power.

4.1 Settings

Experimental Environment. It consists of two software modules[12]. First, the driver module, written in `Python`, is in charge of driving the complete set of experiments : deploying the experiments, launching them, stopping them, and finally fetching the experimental results. Based on the `driver`, we deployed our experiments on our own OAR2 (Linux)[13] computing cluster, allocating 2 cores (2.6 GHz) and 2 GB RAM at least to each experiment. Each experiment is a SUBSUM attack fully parameterized (e.g., full set of time series, aggregates published). We describe it below. The second software module is the solver in charge of performing a SUBSUM attack given a set of parameters. We use in our experiments the Gurobi solver (Gurobi Optimization, 2020), a well-known and efficient IP solver, but any other solver can be used provided that it implements an API similar to the one used in Algorithm 1.

Datasets. We performed our experiments over the real-life *ISSDA CER* public dataset (CER, 2012) which contains power consumption time series of 6435 individuals for over 1.5 years at a 30 mins rate. Removing the time series containing at least one missing value left us with 4622 time series. We changed the unit from kWh to Wh obtaining thus only integer values (required by our solver). We refer to this

[12] Available here: https://gitlab.inria.fr/avoyez1/subsum
[13] https://oar.imag.fr/

cleaned dataset as `ISSDA-30m`. [14]

We replicate the experiments on the real-life *London Households Energy Consumption* dataset (UK Power Networks, 2013) containing 5567 individuals for over 2.5 years at a 30 mins rate. As this dataset contains a lot of missing values (all individuals have at least one missing value), we filled the missing values using the consumption of the previous week. Then, we removed the individuals that have missing values along several weeks. This left us with 5446 individuals. We also changed the unit from kWh to Wh in order to obtain only integer values. We refer to this cleaned dataset as `London-30m`.

Experimental Protocol. We consider three main parameters impacting the SUBSUM attack: the size $|\mathcal{S}|$ of the set of time series, the size $|\mathcal{A}|$ of the vector of aggregates published (number of constraints), and the number of values aggregated in each aggregate $|\mathcal{S}^{\mathcal{A}}|$. Table 2 shows the parameter values we use in our experiments. For each triple of parameters, we generate 20 experiments[15] where each experiment is a SUBSUM attack over (1) a set of time series, (2) a vector $\mathcal{A}$ of aggregates published, and (3) a time budget θ. At each experiment, the full population $\mathcal{S}$, together with the individuals $\mathcal{S}^{\mathcal{A}}$ that are part of the vector of aggregates, are randomly selected. We log for each experiment the ground truth (i.e., the exact set of individuals in $\mathcal{S}^{\mathcal{A}}$), the candidate solution(s) found by the solver, the status of the solver, and the wall-clock time elapsed during the attack.

Success Definition. Our experimental validation considers a flexible success definition. An attack is considered to be a success if the solver finds strictly less than p solutions in the allowed time θ[16]. Indeed, this implies (1) that the solution is part of the pool of solutions and (2) that the solver found all the solutions. We will especially focus on a pool size equals to 2. In this case, a success occurs when the solver outputs a single solution: the membership inference is perfect. However, a pool size equals to 2 is not always sufficient. In these cases, we increase the pool size and analyze the solutions more deeply found by the solver, showing the distribution of the guess vector on the individuals that are part of the ground truth.

[14] https://www.ucd.ie/issda/data/commissionforenergyregulationcer

[15] Performing 20 times the SUBSUM attack on the same set of parameters is sufficient for obtaining a stable success rate for the given set of parameters.

[16] Note that this is a somewhat precautionary success measurement since the attacker will draw conclusions only when he/she is certain that all solutions have been found by the solver. Other success metrics can be considered in which the attacker gains information from incomplete sets of solutions returned by the solver. However, when the solver reaches the time budget or when the pool p is filled, the solver fails to prove the non-existence of other solutions. Inference on incomplete results might be refuted by new, yet undiscovered solutions.

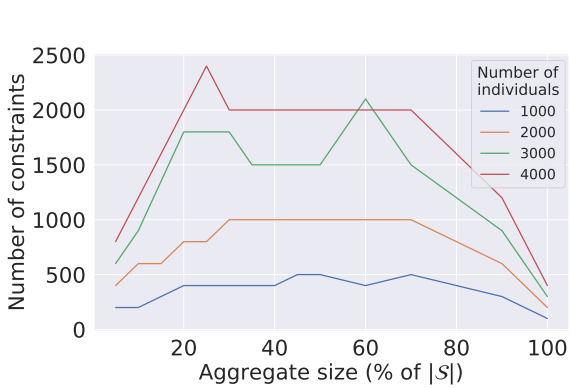
4.2 Experimental Results

Our experiments aim at providing clear insights on the conditions leading to successful SUBSUM attacks. First, we start in Section 4.2.1 by studying the SUBSUM success rate according to the total number of time series in the full population (population size), the number of time series aggregated in the published vector of aggregates (aggregate size), and the length of this vector (number of constraints). We set here the time budget to a value sufficiently high for not interfering with the attack. Through this study, we observe a clear relationship between the population size and the published vector of aggregates (size and length) for a successful SUBSUM attack. This allows us to express both the size of the published vector of aggregates and its length relatively, as a fraction of the population size. Then, we study in Section 4.2.2 the impact of the time budget on the success rate, setting the other parameters to fixed values. Finally, we study in Section 4.2.3 the efficiency of the SUBSUM attack on the full population available in our datasets.

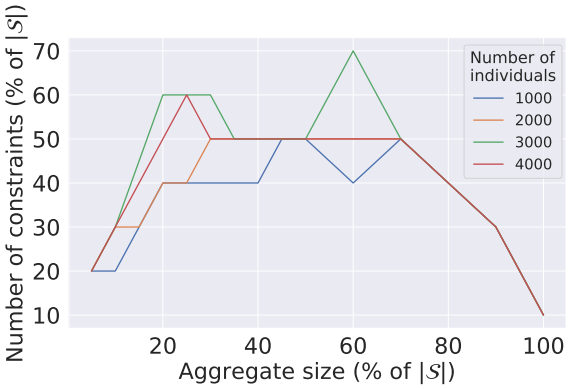
4.2.1 Relationship between the number of individuals and the success

Figure 2 shows the minimal number of constraints required to have at least one success for several population sizes and for a time budget large enough for not interfering (i.e., $\theta = 24h$). We can see that, even if the required number of constraints before getting a success increases with the population size, it follows a parabolic shape for all population sizes. Figure 3 shows the results of attacks with respect to the aggregate size and the number of constraints for a small dataset of 1000 individuals and a small time budget of 1000s (approximately 15 minutes).

As already seen in Figure 2, Figure 3 shows that the attacks fail in a parabolic area centered around aggregates of half the population size. These failures are due to the fact that the solver reaches the maximum allocated time. Outside this area of failure, the attacks are mostly successful with the exception of a few, randomly distributed, failed experiments which depend on the original aggregate and population samples. In this "light" area, the few failures are due to the existence of several possible solutions (we provide below in Section 4.2.3 an advanced analysis of the cases where several solutions exist). After reaching



(a) Y axis: absolute number of constraints



(b) Y axis: number of constraints (% of the number of individuals)

Figure 2: Minimal number of constraints before getting at least one success. Parameters : Dataset = ISSDA-30m, $|\mathcal{S}| = \{1000, 2000, 3000, 4000\}$, θ=24h, p=2, 20 repetitions.

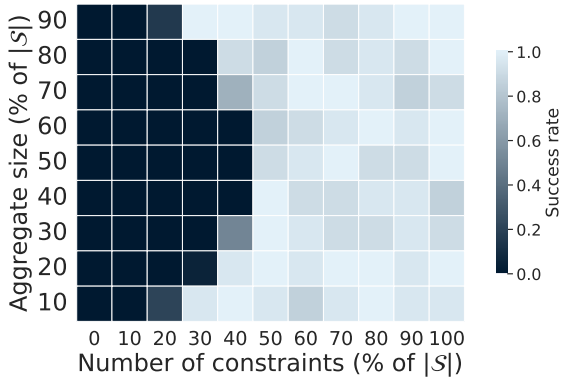
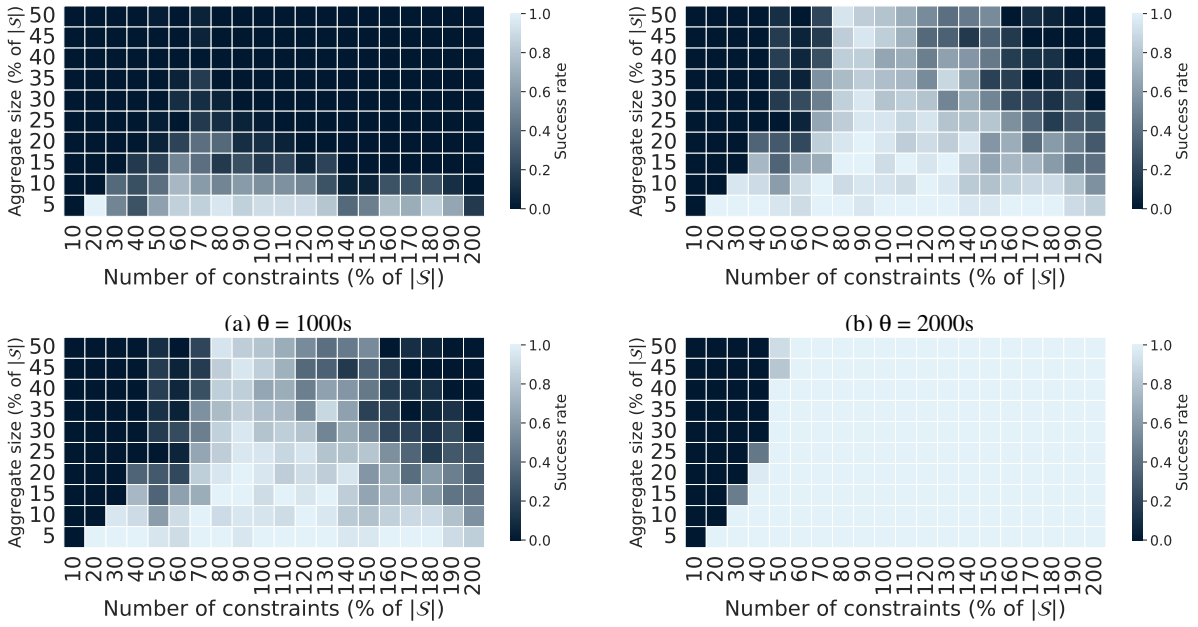


Figure 3: Success rate (0 : all failed, 1 : all succeed). Parameters : Dataset = ISSDA-30m, $|\mathcal{S}|$ = 1000, θ=1000s, p=2, 20 repetitions.

an aggregate size corresponding to 50% of the population, it is expected that the number of constraints needed goes down again which comes from the fact that knowing $\mathcal{S}$, identifying $|\mathcal{S}^{\mathcal{A}}|$ individuals in $\mathcal{S}$ is at least as difficult as identifying $|\mathcal{S}| - |\mathcal{S}^{\mathcal{A}}|$ individuals. With a small aggregate size (lower than 50% of the population size) the number of constraints needed is often twice the size of the aggregate: in general the number of constraints needed to successfully attack a population of size $|\mathcal{S}|$ should be *a*t least in the order of the aggregate size $|\mathcal{S}^{\mathcal{A}}|$. Conversely, if the number of constraints is much lower than the aggregate size, the dataset is deemed safe (relatively to the time budget θ and to the pool size p) to the SUBSUM attack. Building on this result, in the following, both the aggregate size $|\mathcal{S}^{\mathcal{A}}|$ and the length $|\mathcal{A}|$ of the vector of aggregates (number of constraints) are expressed as a fraction of the population size $|\mathcal{S}|$. Note that the length of the vector of aggregates might be larger than the population size, resulting thus in fractions larger than 100% the population size.

4.2.2 Impact of the time budget

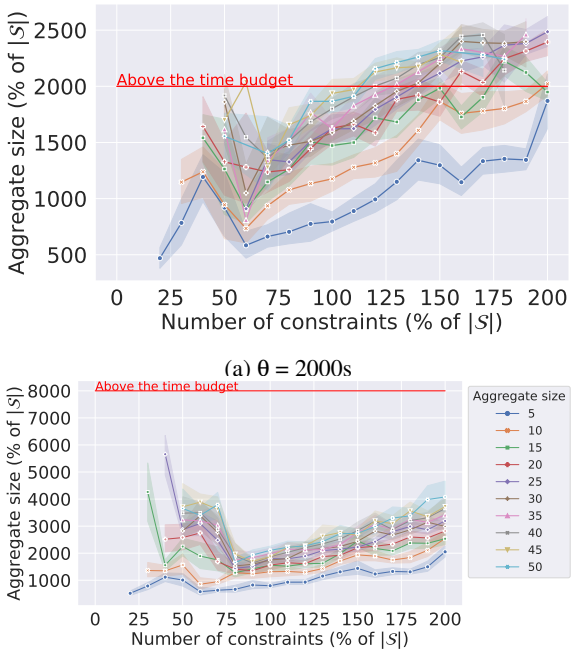
For practical reasons, we evaluate the impact of the time budget on a rather small dataset with $|\mathcal{S}| = 2000$. As can been seen in Figure 4, the time budget θ has an impact on the attack success: the higher the time budget, the higher the success rate. However, as shown in Figure 4d, even with a high time budget, attacking an aggregate with a small number of constraints (the left dark sides of Figures 4b, 4c and 4d) will be unsuccessful. Conversely, if the time budget is too low and the number of constraints is too high (the right black / grey area of the Figures 4a, 4b and 4c), the solver does not have the time to solve the problem with its given parameters. Figure 5 shows the experiments time (in seconds). We note three areas of interest: 1) an area with missing points (before reaching 25% of constraints) corresponding to the, previously defined, parabolic area of failures 2) the failures due to reaching the allowed solver time and 3) when increasing the number of constraints, the sudden drop in execution time until a point where the attack is the fastest for all aggregate sizes. After this point, the execution time increases again linearly with the number of constraints up to the point where the execution time reach the allowed time again (Figure 5b). There is thus a clear compromise to make between the number of constraints used (that needs to be in the order of the aggregate size) and the time budget. If the attacker has a short time budget, he/she might prefer not using all the available constraints at his/her disposal.



(a) θ = 1000s (b) θ = 2000s

(c) θ = 4000s (d) θ = 8000s

Figure 4: Evolution of the success (0 : all failed, 1 : all succeed) depending on the time budget. Parameters : dataset = ISSDA-30m, $|\mathcal{S}| = 2000$, θ = {1000s, 2000s, 4000s, 8000s}, $p = 2$, 20 repetitions.



(a) θ = 2000s

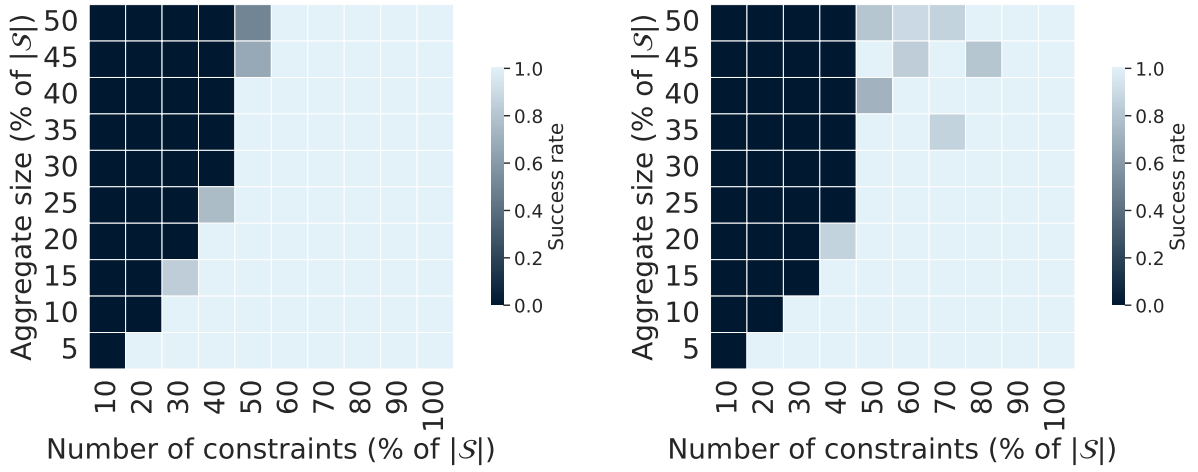
(b) θ = 8000s

Figure 5: Evolution of experiments time depending on the time budget. Parameters : dataset = ISSDA-30m, $|\mathcal{S}| = 2000$, θ = {2000s, 8000s}, $p = 2$, 20 repetitions.

4.2.3 Attacking large populations

The attacks on our largest dataset provide results similar to the ones previously described. However, since the dataset is larger, we need to increase the time budget to 24h to observe the same phenomena. As shown in Figure 6, the relationship between the dataset size, the aggregate size and the number of aggregates required for the attack to succeed is similar to what we observed in the other experiments. In particular, in this case, we need roughly twice more constraints than the aggregate size until we reach an aggregate size of 25% of $|\mathcal{S}|$. After that, the number of constraints needed to reach a success remains stable at 50% of $|\mathcal{S}|$.

It is worth to note that we set the pool size to $p = 100$ in these experiments, meaning that the attack is considered to be successful if the number of solutions found is between 1 and 99. This allows us to study the cases where several solutions are found, which make $\mathcal{G}$ worth analyzing. We chose the experiment in which the solver returned the highest number of solutions and where the intersection of these solutions is the smallest (in other words, where the number of individuals that are part of several solutions is the highest). This case occurred in one of the 20 repetitions of the attack performed with the following parameters: dataset = ISSDA-30m, $|\mathcal{S}| = 4500$, θ = 24h, $p = 100$, $|\mathcal{S}^{\mathcal{A}}| = 225$ (5%) and $|\mathcal{A}| = 900$ (20%). It resulted in a pool containing 3 solutions. We represent in Figure 7 the part of the guess vector limited to the individuals found in the 3 solutions. Figure 7 is a histogram showing the fraction of individuals (Y-axis)



(a) Dataset = London-30m

(b) Dataset = ISSDA-30m

Figure 6: Attack success (0 : all fail, 1 : all succeed) for datasets of 4500 individuals. Parameters : dataset = {ISSDA-30m, London-30m}, $|\mathcal{S}| = 4500$, $\theta = 24$h, $p = 100$, 20 repetitions.

associated to eleven possible ranges of guess in $\mathcal{G}$[17] (X-axis). The figure shows that 224 (98.7%) individuals appear in all the solutions: their membership can thus be completely inferred with certainty while only one individual differs in all three solutions: the membership probability of the corresponding individuals is thus equal to 1/3. This shows that the SUBSUM attack manages to gain significant knowledge about the aggregate members even when multiple solutions are provided.

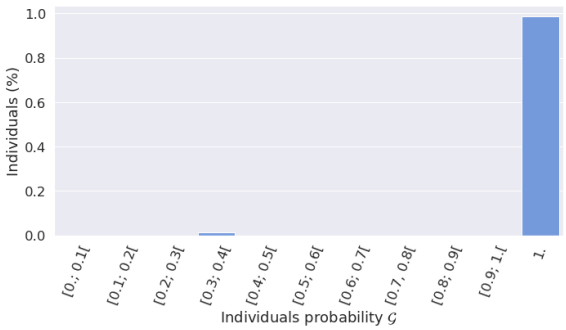


Figure 7: Focus on the experiment in which the solver returned the highest number of solutions and where the intersection of these solutions is the smallest (3 solutions found in this case). We show here the fraction of individuals associated to each possible adversarial guess value (organized within eleven ranges). Parameters : dataset = ISSDA-30m, $|\mathcal{S}| = 4500$, $\theta = 24$h, $p = 100$, $|\mathcal{S}^{\mathcal{A}}| = 225$ (5%), $|\mathcal{A}| = 900$ (20%).

[17]Recall that the vector of guesses $\mathcal{G}$ contains for each individual the probability that the individual participates to $\mathcal{A}$.

5 RELATED WORKS

In the literature, attacks applicable against the privacy-preserving publication of electrical consumption data can be separated in three categories. First, there is the *Linear reconstruction* attacks where the attackers aim at finding back the original dataset from a publication. Secondly, the *Membership Inference Attacks* (MIA in short) looks if a specific individual is part of a publication. Lastly, *NonIntrusive Load Monitoring* (or NILM) tries to infers valuable information (e.g., the devices used) from an electrical consumption time series. We consider our attack as being a membership inference attack (we try to know which individuals take part in a publication) by using methodologies explored by the reconstruction attacks (the usage of linear programming solvers).

5.1 Linear reconstruction attacks

Linear reconstruction attacks (Cohen and Nissim, 2020; Dinur and Nissim, 2003; Dwork et al., 2007; Dwork and Yekhanin, 2008; Lacharité et al., 2018; Quadrianto et al., 2008), illustrated by Cohen & al. (Cohen and Nissim, 2020), are probably the closest from our attack. In these attacks, an attacker is given access to an interface allowing him to generate aggregates from limited queries on the dataset. The attacker sends a set of queries, each of them getting a new aggregate from a randomly selected subset of the dataset. Each query is then used to build a set of constraints that are then solved by a linear programming solver. As for our attack, the goal is to extract private information from a set of constraints. However, the

SUBSUM attack differs in several ways. First, the linear reconstruction attack aims to reconstruct a private dataset (e.g., the values of the dataset elements) while our attack aims to reconstruct private aggregates from known data. The attack aims to infer the presence or the absence of individuals in an aggregate and not their values. Second, due to the different objective of the attack, the attacker does not require the same background knowledge. Our attack does not require access to a query interface and can be performed on a static set of aggregates as they are generally published in open data programs. However, the large number of queries required by the linear reconstruction attack is somewhat balanced by the requirement of the individuals' values.

5.2 Membership inference attacks

Membership inference attacks are another family of attacks where the attacker wishes to infer the presence or the absence of a target inside a dataset or whether a target has played a part in the result of a function (Dwork et al., 2017; Rigaki and Garcia, 2020). Datasets can broadly take the form of aggregates (Bauer and Bindschaedler, 2020; Büscher et al., 2017; Jayaraman et al., 2021; Pyrgelis et al., 2020a; Pyrgelis et al., 2018; Pyrgelis et al., 2020b; Zhang et al., 2020), data used to train machine learning model (Long et al., 2020; Melis et al., 2019; Salem et al., 2019; Shokri et al., 2017; Truex et al., 2019; Yeom et al., 2018), genomics database (Homer et al., 2008; Samani et al., 2015; Sankararaman et al., 2009) or original data used to generate synthetic ones (Stadler et al., 2020).

5.2.1 Membership inference attacks on aggregates

When membership inference attacks are applied to aggregates (Bauer and Bindschaedler, 2020; Büscher et al., 2017; Jayaraman et al., 2021; Pyrgelis et al., 2020a; Pyrgelis et al., 2018; Pyrgelis et al., 2020b; Zhang et al., 2020), as in our approach, the aim is to find whether an individual is part of an aggregate or a set of aggregates. In (Pyrgelis et al., 2018), an attacker has access to a set of aggregates representing the number of people visiting a specific geographic point at a particular time. Similarly to time series, each geographic point generates a new record at given periods. Prior to the attack, the attacker has access to incomplete background knowledge about a target (a single person) telling where it was at a given time. The target can be present in the set of observed points or not. The attacker will then train a machine learning classifier which will be able to infer whether or not the target is present on at least one geographic point at a given time outside his training range. Compared to our attack, this attack required far less background knowledge: only a few points where the target is present or not. However its scale is limited: it is only able to infer the presence or the absence of the target inside a set of aggregates while our attack can reconstruct up to every record present in the aggregate. Membership inference can be applied to more generic data (Bauer and Bindschaedler, 2020; Jayaraman et al., 2021). In these cases a similar method is used: an attack model is trained with similar data as the one attacked. This model is then used to deduce the presence or the absence of a specific target in an aggregate.

5.2.2 Membership inference attack on ML models

These attacks (Long et al., 2020; Melis et al., 2019; Salem et al., 2019; Shokri et al., 2017; Truex et al., 2019; Yeom et al., 2018) aim to find whether an individual is part of the dataset used to train a machine learning model. The attacker has access to unlimited queries to a black box model and to records sampled from the same distribution as the original training data. Usually (Shokri et al., 2017; Truex et al., 2019), the attacker will train an ML-based attack model against multiple shadow models which replicate the behavior of the attacked model and are trained with some known datasets (with and without the target individual). The attack model will then be applied against the real model to attack. Membership attacks on ML models requires knowledge of distribution of the attacked records and large amount of data to train the shadow models. However, they do not need to get access to the real data used by the attacked model. These attacks are, for now, limited to attacking machine learning models and not aggregate data.

5.3 Non Intrusive Load Monitoring

(NILM) introduced by Hart (Hart, 1992) in 1992 is a field of study aiming to extract information contained in electric consumption time series such as the occupancy of a household, the electric devices used and even the TV program watched. NILM algorithms can use a wide variety of methods to reach this goal from finite state machines to knapsack-based algorithms and machine learning classifiers. Despite not being considered as privacy attacks, these algorithms can be used as a tool to retrieve private information contained in time series. To the best of our knowledge, no attempt has been made to adapt NILM principles to extract information contained in an aggrega-

tion of multiple electric load consumption time series. NILM-based attacks may be used as a complement of our attack either to filter a set of loads potentially present in the aggregates or to train a NILM model from the consumption values of the aggregates.

6 CONCLUSION

Membership inference attacks aim to infer whether a specific individual belongs to a given aggregate. In this work, we introduced a new technique to perform inference attacks on (threshold-based) aggregated time series. Given a publicly known dataset of time series, and aggregated values (one value per timestamp) calculated from a private subset of that dataset, the adversary that we consider is able to re-identify the individuals belonging to the private subset. To do so, we modeled the membership inference attack as a subset sum problem and we used Gurobi to solve it. We completed experiments on two real-life datasets containing power consumption time series, from UK and Ireland, respectively. We showed that if the number of available aggregated values is larger than the aggregate size (i.e., the number of individuals in the private subset), then the SUBSUM attack is highly likely successful. Interesting future works include relaxing the background knowledge required by the SUBSUM attack (e.g., missing time series, approximate values) and coping with advanced privacy-preserving approaches (e.g., differential privacy).

ACKNOWLEDGEMENTS

We would like to thank Charles Prud'homme (TASC, IMT-Atlantique, LS2N-CNRS) for his help on the choice of the solver. Some of the authors are supported by the TAILOR ("Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization" - H2020-ICT-2018-20) project.

REFERENCES

Bauer, L. A. and Bindschaedler, V. (2020). Towards realistic membership inferences: The case of survey data. In *ACSAC '20: Annual Computer Security Applications Conference*, pages 116–128. ACM.

Büscher, N., Boukoros, S., Bauregger, S., and Katzenbeisser, S. (2017). Two is not enough: Privacy assessment of aggregation schemes in smart metering. *Proc. Priv. Enhancing Technol.*, 2017(4):198–214.

CER (2012). CER Smart Metering Project - Electricity Customer Behaviour Trial. Irish Social Science Data Archive. (accessed may 11 2022). https://www.ucd.ie/issda/data/commissionforenergyregulationcer.

Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca (2016). Choco solver documentation. http://www.choco-solver.org.

Cohen, A. and Nissim, K. (2020). Linear Program Reconstruction in Practice. *J. Priv. Confidentiality*, 10(1).

Deloitte (2017). Assessing the value of tfl's open data and digital partnerships.

Dinur, I. and Nissim, K. (2003). Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 202–210.

Dwork, C., McSherry, F., and Talwar, K. (2007). The price of privacy and the limits of LP decoding. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 85–94.

Dwork, C., Smith, A., Steinke, T., and Ullman, J. (2017). Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4:61–84.

Dwork, C. and Yekhanin, S. (2008). New efficient attacks on statistical disclosure control mechanisms. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conferences*, volume 5157 of *Lecture Notes in Computer Science*, pages 469–480.

Gauvin, L., Tizzoni, M., Piaggesi, S., Young, A., Adler, N., Verhulst, S., Ferres, L., and Cattuto, C. (2020). Gender gaps in urban mobility. *Humanities and Social Sciences Communications*, 7(1):1–13.

Gurobi Optimization, L. (2020). Gurobi optimizer. http://www.gurobi.com.

Hart, G. (1992). Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891.

Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J. V., Stephan, D. A., Nelson, S. F., and Craig, D. W. (2008). Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167.

Jayaraman, B., Wang, L., Knipmeyer, K., Gu, Q., and Evans, D. (2021). Revisiting membership inference under realistic assumptions. *Proc. Priv. Enhancing Technol.*, 2021(2):348–368.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). The subset sum problem. In *Knapsack Problems*, pages 73–115. Springer.

Lacharité, M., Minaud, B., and Paterson, K. G. (2018). Improved reconstruction attacks on encrypted data using range query leakage. In *IEEE Symposium on Security and Privacy, SP*, pages 297–314.

Laurent Perron, Vincent Furnon, G. (2019). Or-tools. https://developers.google.com/optimization/.

Long, Y., Wang, L., Bu, D., Bindschaedler, V., Wang, X., Tang, H., Gunter, C. A., and Chen, K. (2020). A pragmatic approach to membership inferences on machine learning models. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020*, pages 521–534.

Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019*, pages 691–706.

OECD (2020). Oecd open, useful and re-usable data (ourdata) index: 2019. https://www.oecd.org/gov/digital-government/policy-paper-ourdata-index-2019.htm.

Pyrgelis, A., Troncoso, C., and Cristofaro, E. D. (2018). Knock knock, who's there? membership inference on aggregate location data. In *25th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society.

Pyrgelis, A., Troncoso, C., and Cristofaro, E. D. (2020a). Measuring membership privacy on aggregate location time-series. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(2):36:1–36:28.

Pyrgelis, A., Troncoso, C., and Cristofaro, E. D. (2020b). Measuring Membership Privacy on Aggregate Location Time-Series. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(2):36:1–36:28.

Quadrianto, N., Smola, A. J., Caetano, T. S., and Le, Q. V. (2008). Estimating labels from label proportions. *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML 2008)*, pages 776–783.

Rigaki, M. and Garcia, S. (2020). A survey of privacy attacks in machine learning. *CoRR*, abs/2007.07646.

Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. (2019). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019*.

Samani, S. S., Huang, Z., Ayday, E., Elliot, M., Fellay, J., Hubaux, J., and Kutalik, Z. (2015). Quantifying genomic privacy via inference attack with high-order SNV correlations. In *2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015*, pages 32–40.

Sankararaman, S., Obozinski, G., Jordan, M. I., and Halperin, E. (2009). Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy, SP 2017*, pages 3–18.

Stadler, T., Oprisanu, B., and Troncoso, C. (2020). Synthetic data - A privacy mirage. *CoRR*, abs/2011.07018.

The European Data Portal (2015). Creating value through open data : Study on the impact of re-use of public data resources. https://op.europa.eu/en/publication-detail/-/publication/51ec011a-e13b-11e6-ad7c-01aa75ed71a1.

The World Wide Web Foundation (2017). Open data barometer - global report (4th ed.).

Truex, S., Liu, L., Gursoy, M. E., Wei, W., and Yu, L. (2019). Effects of differential privacy and data skewness on membership inference vulnerability. In *First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019*, pages 82–91.

UK Power Networks (2013). SmartMeter Energy Consumption Data in London Households (Accessed May 11 2022). https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *31st IEEE Computer Security Foundations Symposium, CSF 2018*, pages 268–282.

Zhang, G., Zhang, A., and Zhao, P. (2020). Locmia: Membership inference attacks against aggregated location data. *IEEE Internet Things J.*, 7(12):11778–11788.

é