# Practical Anonymization for Data Streams: z-anonymity and relation with k-anonymity

Nikhil Jha*, Luca Vassio, Emilio Leonardi, Marco Mellia

*Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino 10129, Italy*

Martino Trevisan

*University of Trieste, Piazzale Europa, 1, 34127 Trieste TS, Italy*

**Abstract**

With the advent of big data and the emergence of data markets, preserving individuals' privacy has become of utmost importance. The classical response to this need is anonymization, i.e., sanitizing the information that, directly or indirectly, can allow users' re-identification. Among the various approaches, $k$-anonymity provides a simple and easy-to-understand protection. However, $k$-anonymity is challenging to achieve in a continuous stream of data and scales poorly when the number of attributes becomes high.

In this paper, we study a novel anonymization property called $z$-anonymity that we explicitly design to deal with data streams, i.e., where the decision to publish a given attribute (atomic information) is made in real time. The idea at the base of $z$-anonymity is to release such attribute about a user only if at least $z-1$ other users have exposed the same attribute in a past time window. Depending on the value of $z$, the output stream results $k$-anonymized with a certain probability. To this end, we present a probabilistic model to map the $z$-anonymity into the $k$-anonymity property. The model is not only helpful in studying the $z$-anonymity property, but also general enough to evaluate the probability of achieving $k$-anonymity in data streams, resulting in a generic contribution.

*Keywords:* Anonymization, data streams, scalability, zero delay, k-anonymity.

## 1. Introduction

Big data have opened new opportunities to collect, store, process and, most of all, monetize data. This has created tension with privacy, especially regarding

*Corresponding author

*Email addresses:* nikhil.jha@polito.it (Nikhil Jha), luca.vassio@polito.it (Luca Vassio), emilio.leonardi@polito.it (Emilio Leonardi), marco.mellia@polito.it (Marco Mellia), martino.trevisan@dia.units.it (Martino Trevisan)

information about individuals. For this, legislators have introduced privacy laws to regulate the data collection and market, with notable examples of the General Data Protection Regulation (GDPR) [1] in EU, or the California Consumer Privacy Act (CCPA) [2] in the US.

Anonymization, i.e., generalizing or removing identifying data of individuals, is the classical approach to publish personal information. Thanks to this, Privacy-Preserving Data Publishing (PPDP) has gained attention in the last decade [3]. The birth of data markets, where buyers can access extensive data about individuals, has brought even more attention to the area. In this context, the PIMCity project[1] aims at exploring practical solutions to create transparent online data marketplaces, and this paper is part of our effort to develop practical strategies for anonymization.

Removing the users' *identifiers* (name, social security number, phone number, etc.) is insufficient to make a dataset anonymous. Indeed, an attacker can link users' apparently harmless attributes (such as gender, zip code, date of birth, etc.) called *quasi-identifiers* (QIs) to some external knowledge. With that, the attacker can re-identify the person and thus gain access to other sensible information available in the dataset (disease, income, etc.) – called *sensitive attributes* (SAs) [4]. Famous is the de-anonymization of the Netflix public dataset [5] based on the exploitation of QIs.

Researchers have defined several properties that data should satisfy to avoid re-identification, among which the $k$-anonymity [6], or $k$-anon for short, has become popular. It imposes that every user's released piece of information (a record) should correspond to at least $k-1$ other users, i.e., there are at least $k-1$ other users with the same record. $k$-anon is conceived for tabular and static data; in other words, the dataset must be completely available at anonymization time. Researchers have proposed extensions to a streaming scenario, where continuously incoming records are accumulated, processed in batches, and released after an unavoidable delay [7]. However, for specific applications, it is fundamental to avoid any processing delay. For example, when it is unfeasible to store data for a long time (e.g., for network packets [8]); or for location history, if a real-time (but anonymous) stream shall be used for mobility optimization, e.g., suggesting a free parking spot or the least congested route.

This paper studies the novel anonymization property called $z$-anonymity, or $z$-anon for short, that we previously introduced in [8] in the context of Internet traffic analysis. $z$-anon specifically targets the data stream scenario, aiming to guarantee zero-delay release of data (hence the $z$ instead of $k$). We suppose to receive a raw stream of data in which users' attributes arrive as they are generated. For instance, a new transaction on their credit card, a new position of their car, or a new website they visit. These attributes are QIs, and may allow users' re-identification. To prevent this, a new attribute is released only if it was exposed by at least $z-1$ other individuals in the past window $\Delta t$. $z$-anon is weaker than $k$-anonymity since it cannot guarantee that at least $k-1$ users

[1] `https://www.pimcity-h2020.eu/`

present the same *combinations* of QIs. In this paper, we present a probabilistic model to map $z$- into $k$-anon properties. We find out that $z$ can be tuned to provide $k$-anon with the desired probability. We first study the impact of scenario (number of users, number of attributes) and system parameters ($z$, $\Delta t$) considering a homogeneous and uniform case. Then we relax some assumptions of the model, considering the case of classes of users with different interests or activity levels.

There are various examples of application of $z$-anon. For instance, we have originally proposed it for internet traffic analysis, where high-speed passive monitors process packets that contain QIs (e.g., hostnames of visited websites) in real time [8]. Similarly, the users' browsing history, credit card history, and location history offer rich information that companies may want to access as quickly as possible.

This paper is an extended version of our previous work [9]. In this new version, we improve the model and propose an approximation technique to scale computation. We also extend it to handle classes of users, which now results generic and can be used to evaluate the probability of achieving $k$-anonymity property in data streams. Finally, we quantify the information loss resulting from anonymization, which we measure with the entropy concept as derived from the Information Theory [10].

In the remainder of the paper, after presenting the related work (Section 2), we formalize the $z$-anon property and present an algorithm to enforce it efficiently and in real-time (Section 3). We then propose a probabilistic model to derive $k$-anon properties from $z$-anonymized streams (Section 4). In Section 5, we study the impact of the $z$-anon parameters and system characteristics. We then extend our model to handle different classes of users (Section 6). Finally, we discuss the limitations of our approach, future research direction and draw the conclusions (Section 7). In Appendix A, we detail the implementation and the complexity of the algorithm, in Appendix B we describe an approximation to efficiently use the model in practice, in Appendix C we compare the results obtained by the model to the simulation ones.

## 2. Related work

The problem of providing anonymization guarantees to streaming datasets has arisen with increasing attention to PPDP. Most current solutions work with the concept of data batches, i.e., the incoming data are first accumulated, then processed, and finally released with a sizeable delay. Researchers have proposed several approaches during the years that we summarize in the following.

When working with batches or microbatches, popular approaches aim at guaranteeing $k$-anon independently in each batch. Here, popular solutions are based on trees [11–13] or clustering [7, 14–17]. The rationale behind them is roughly the same: firstly load the incoming records into a structure and secondly release tuples for which $k$-anon is achieved. For instance, authors of [18] design a solution in four steps: cluster arriving tuples, evaluate a noisy centroid for each cluster, control the cluster size to manage concept drifts, and finally release the tuples for those clusters where $k$-anon is verified. Delay is inevitable with clusters needing to accumulate more data before the release. Authors of [19] modify the attributes to steer a microaggregation process. Tuples are not published as is, but they are first aggregated into clusters, whose only centroid is published. The proposal in [20] uses instead a sliding window approach, where tuples are processed to achieve anonymization using a noisy function. Again, data is released only after a window of time.

Other works design approaches based on perturbation. Authors of [21] propose to perturb the output stream as follows. When a user exposes a sensitive attribute, the system publishes $l - 1$ different sensitive attributes so that the attacker can find the actual one with a probability $1/l$. Authors of [22] also propose to replace incoming tuples with sensitive-value sets. To build appropriate sets, they introduce the concepts of semantic and sensitivity diversity. These two techniques allow zero-delay anonymization, but the output streams are largely modified by the perturbation, creating scalability issues too. Furthermore, no guarantees are provided that the resulting release is $k$-anonymized.

Considering suppression and generalization, authors of [23] propose two algorithms to avoid a correlation analysis from transaction items. They use a sliding window approach and aim to provide the data in the current window as output, after guaranteeing it meets privacy constraints. Again, data is published as a continuous stream, but only after one delay window. At last, researchers have spent some efforts to reduce and factor the cost of the delay: Authors of [12] include the delay in the concept of output quality, with a trade-off between data quality and batch size.

To the best of our knowledge, however, only [21] and [22] target the zero-delay goal. In [21], input streams are anonymized immediately with counterfeit values. In [22], $m-1$ random sensitive attributes are published with the original one. Our goal is instead to publish only actual attributes, suppressing those that could allow users re-identification. Here, we present a model that allows one to link the properties of $z$-anon to $k$-anon. None of the previous works provides any means to reach this. This work extends our previous one [9] by refining the $z$-anon model, introducing the case in which different classes of users coexist, and quantifying the information loss. We present a thorough analysis of the impact of scenario and system parameters. Our model is now general and allows one to evaluate the $k$-anonymity probability in data streams, as well as the information loss due to the anonymization process.

## 3. *z*-anonymity: anonymization for data streams

### *3.1. The z-anonymity property*

We suppose to operate with data streams, where we continuously receive observations that associate users with attributes. We define an observation as a tuple $(t, u, a)$, indicating that, at time $t$, the user $u$ exposes the attribute $a$ from a catalog of attributes $\mathcal{A}$. Attributes can be related to whatever field: a visit to a web page, a purchase, a GPS location, etc.

Here, we assume every attribute $a \in \mathcal{A}$ is a *quasi-identifier*. That is, in the stream there are no *sensitive attributes* – i.e., attributes that contain private information, but cannot bring to re-identification of the user. The users are completely described by the set of quasi identifiers $\mathcal{A}$.

We want to keep private those values of attributes associated with small groups of users, which could ease the re-identification. As presented in [8], we define the property of $z$-private attribute as follows:

**Definition 1.** An attribute $a$ is $z$-private at time $t$ if it is exposed by less than $z$ users in the past $\Delta t$ time interval.

Notice that the same attribute $a$ can be both $z$-private and not $z$-private for different $t$. If the anonymized dataset hides all $z$-private attributes, it achieves $z$-anon.

**Definition 2.** A stream of observations is $z$-anonymized if it does not contain $z$-private attributes at any $t$, given $z$ and $\Delta t$.

In other words, the attributes that are associated with less than $z$ users in the past $\Delta t$ shall be removed or replaced with an empty identifier. The goal is to prevent rare attributes from being published, thus reducing the possibility of an attacker to re-identify a user. In the following, we show how it is possible to achieve $z$-anon in real time efficiently.

We exemplify a mechanism to enforce $z$-anon in Figure 1, while the details of the algorithm and the best-suited data structures to deploy it are discussed in Appendix A. Assume $z = 3$. At time $t_0$ user $u_0$ is the first to expose the attribute $a_0$. The attribute $a_0$ is $z$-private at time $t_0$, hence it shall be obfuscated. Still, the information that $u_0$ exposed the attribute $a_0$ shows its effects for a time equal to $\Delta t$. At time $t_1$, user $u_1$ also exposes $a_0$. Since the number of observations in $\Delta t$ is still smaller than 3, this observation is not released. At time $t_2$ user $u_0$ re-exposes $a_0$, extending the lifetime of the observation, but not changing the number of unique users having exposed $a_0$. At time $t_3$, user $u_2$ exposes $a_0$, making the total users in the past $\Delta t$ equal to 3. Thus the attribute $a_0$ is not $z$-private at time $t_3$ and the observation $(t_3, u_2, a_0)$ can be be released. At time $t_1 + \Delta t$ the attribute $a_0$ related to user $u_1$ expires, hence the total user count decreases back to 2. The same happens when $u_0$ observation expires (at $t_2 + \Delta t$), so that when $u_3$ exposes $a_0$ at $t_4$ the observation can no more be released.

$z$ and $\Delta t$ are parameters that can be tuned to achieve the desired trade-off between data utility and privacy. Therefore, $z$-anon can be adapted to the needs of the desired use case. A large $z$ and a small $\Delta t$ result in the majority of attributes to be anonymized, while a small $z$ and a large $\Delta t$ allow rare values to be possibly released.

The $\Delta t$ parameter can be set based on how often the system administrator randomizes the identifiers of users – such that a user $u$ is no more related to the same identifier after a time period $\Delta t$: its choice may depend on several aspects, such as the nature of the application, the input stream rate, the system

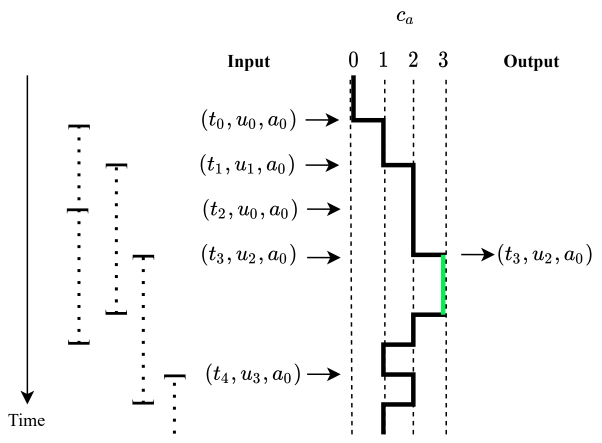


Figure 1: A graphical example of $z$-anon concept with $z = 3$: a tuple is released only if at least other $z - 1 = 2$ different users have exposed the same attribute in the previous $\Delta t$.

memory (the larger $\Delta t$, the larger the memory requirement to store the users' information), and so on.

Notice that $z$-anon considers individual attributes, not on their combinations, as for the $k$-anon property. Hence, it is interesting to study which guarantees the $z$-anon algorithm offers in a global perspective, i.e., which guarantees it is possible to give on the privacy properties in terms of $k$-anon of the output. We study this relationship between $z$-anon and $k$-anon in Section 4.

### *3.2. Modeling z-anonymity*

To fully understand the effect of applying $z$-anon, we model the input data stream as a stochastic process and we show how anonymization modifies it. The code implementing the model is available at [24], while we describe the algorithm to achieve $z$-anon in Appendix A. Table 1 summarizes the terminology we use throughout the following sections.

#### *3.2.1. Modeling the data stream*

We consider a system in which a set of $\mathcal{U}$ users can access the catalog $\mathcal{A}$ of attributes. Let $U = |\mathcal{U}|$ and $A = |\mathcal{A}|$. Users generate a stream of information, exposing in real-time the attribute they have just accessed. The system collects *tuples* in the form $(t, u, a)$, i.e., at time $t$, the user $u \in \mathcal{U}$ exposes the attribute $a \in \mathcal{A}$.

For now, we assume that users are homogeneous and generate independent tuples, so that the probability of exposing a specific tuple depends only on $a$. We will relax this assumption by considering classes of users in Section 6. We assume any user $u$ exposes the attribute $a$ according to a homogeneous Poisson process with rate $\lambda_a$. Hence, the number of times a user exposes the attribute $a$ in a time period $\Delta t$ is modeled as a Poisson distributed random variable $R_a$ with parameter $\lambda_a \cdot \Delta t$, i.e., $R_a \sim Poisson(\lambda_a \cdot \Delta t)$.

Table 1: Terminology used to model $z$-anon and $k$-anon.

| | |
|---|---|
| $\mathcal{U}, U$ | Set and number of users |
| $\mathcal{A}, A$ | Set and number of attributes |
| $\Delta t$ | The time interval length used for evaluating $z$-anon |
| $\lambda_a$ | Exposing rate for attribute $a$ |
| $R_a$ | Random variable counting number of times a user exposes attribute $a$ in $\Delta t$. $R_a \sim Poisson(\lambda_a \cdot \Delta t)$ |
| $X_a$ | Random variable representing whether a user exposes attribute $a$ in $\Delta t$. $X_a \sim Bernoulli(p_a^X)$ |
| $O_a$ | Random variable representing whether a tuple $(t, u, a)$ is published when exposed. $O_a \sim Bernoulli(p_a^O)$ |
| $Y_a$ | Random variable representing whether a user published at least once attribute $a$ in $\Delta t$. $Y_a \sim Bernoulli(p_a^Y)$ |
| $\overline{Y}$ | Set of random variables $\{Y_a\}_{a \in \mathcal{A}}$. $\overline{Y} \sim Bernoulli(p_{\overline{y}})$ |
| $Q_{\overline{y}}$ | Random variable representing the number of users $u \in \mathcal{U}$ with the same realization $\overline{y}$ in $\Delta T$. $Q_{\overline{y}} \sim Binomial(U - 1, p_{\overline{y}})$ |
| $p_{k-anon}$ | Probability that a realization of $\overline{Y}$ satisfies $k$-anon property |

Table 2: The default values used for the model.

| **Variable** | **Default Value** |
|---|---|
| $U$ | 1 000 |
| $A$ | 20 |
| $\lambda_{a_r}$ | 0.2 / r |
| $z$ | 150 |
| $k$ | 2 |
| $\Delta t$ | 12 |

In our analyses, we assume a small set of popular attributes and a large tail of infrequent ones. This allows us to represent systems where users are more likely to expose top-ranked attributes, but there exist a large catalog, a condition which is often observed in real-world systems that are governed by power-law distributions [25]. The usability of the model does not depend on these assumption, which are just considered to match several real-world scenarios. As such, we choose that the $\lambda_a$ for all attributes follows a power law in function of their rank. Let us suppose attributes are sorted by rank, where the most popular attribute is $a_1$ and the least popular $a_A$. In the implementations we will show, we impose $\lambda_{a_1} = 0.2$ and set the remaining $\lambda_a$ as the power-law function $\lambda_{a_r} = {}^{0.2}/_{r}$, where $r$ is the rank of attribute $a_r$. The default parameters used in this article are collected in Table 2.[2]

We denote as $X_a$ the random variable describing whether a user exposed at least once the attribute $a$ in a time interval $\Delta t$. $X_a$ assumes value 1 if the user exposes $a$ in $\Delta t$, 0 otherwise. We note that, by construction, $X_a \sim Bernoulli(p_a^X)$, where $p_a^X$ denotes the probability that a user has exposed attribute $a$, at least once, in the past $\Delta t$. It is straightforward to compute $p_a^X$ given $\lambda_a$ and $\Delta t$ as:

[2]We change the default parameter values from [9] to limit the computational complexity induced by the new model. See Appendix B for a discussion of model scalability.

$$p_a^X = P[R_a \geq 1] = 1 - P[R_a = 0] = 1 - \exp(-\lambda_a \cdot \Delta t) \tag{1}$$

Notice that the different attributes are independent and $p_a^X$ is not a distribution probability mass function, hence the sum of $p_a^X$ over $a \in \mathcal{A}$ can be different from 1.

### *3.2.2. Applying z-anon*

We show how a stream of data modeled as above appears after being $z$-anonymized. Under $z$-anon, $z$-private attributes at time $t$ are not released. Here, we define the indicator random variable $O_a$ associated to the event that the exposed tuple $(t, u, a)$ is published, whose probability of occurring is denoted with $p_a^O$. $(t, u, a)$ is published if $a$ is not $z$-private at time $t$.

$$p_a^O = P[O_a = 1] = P\left[\sum_{v \in \mathcal{U} \setminus u} X_a \geq z - 1\right] \tag{2}$$

Given our assumption of independence and homogeneity across the users, we are summing up $U-1$ independent and identically distributed random variables, which are distributed as $X_a$. Note that we exclude user $u$, since we are checking the $z$-anon conditionally over the tuple $(t, u, a)$. Hence the current user is already involved by construction.

Since $X_a$ is a Bernoulli with success probability $p_a^X$, its sum, which is Binomially distributed, counts the number of occurrences in a sequence of $U - 1$ independent experiments, $\sum_{v \in \mathcal{U} \setminus u} X_a \sim \mathit{Binomial}(U - 1, p_a^X)$.

Starting from Equation 2 and using the probability mass function of the Binomial distribution we can derive $p_a^O$ as:

$$p_a^O = 1 - \sum_{i=0}^{z-2} \binom{U-1}{i} \left(p_a^X\right)^i \left(1 - p_a^X\right)^{U-1-i} \tag{3}$$

Similarly to Equation 1, we denote as $Y_a$ the random variable describing whether a user published at least once the attribute $a$ in a time interval $\Delta t$. Again, $Y_a \sim \mathit{Bernoulli}(p_a^Y)$, where $p_a^Y$ is simply:

$$p_a^Y = P[X_a = 1] \cdot P[O_a = 1] = p_a^X \cdot p_a^O$$

The set of random variables describing the presence or absence for all the possible attributes $a \in \mathcal{A}$ for a user is denoted as $\overline{Y} = \{Y_a\}_{a \in \mathcal{A}}$. The attacker will not know the random variable $\overline{Y}$, and will observe only realizations of it. Let us denote as $y_a$ a realization of the random variable $Y_a$ and as $\overline{y} = \{y_a\}_{a \in \mathcal{A}}$ a realization of the random variable $\overline{Y}$.

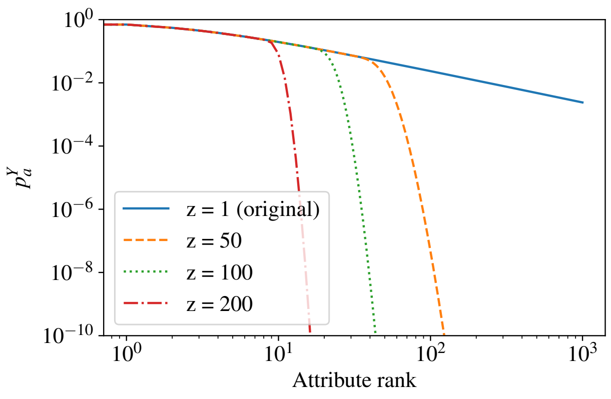


Figure 2: The probability $p_a^Y$ for a user to publish attribute $a$ in $\Delta t$.

### *3.2.3. Impact on released data*

We now qualitatively show the effect of applying $z$-anon on the released data stream. In this experiment, we set $A = 1\,000$ and $U = 1\,000$. We suppose the popularity of attributes follows a power law, with $\lambda_{a_r} = 0.2/r$, where $r$ is the attribute rank.

We study the probability of observing the attribute $a$ in a $\Delta t$, for a given user, in both the original and released data. Figure 2 shows $p_a^Y$ in function of the attribute rank. The blue solid line represents the probability of observing an attribute in case $z = 1$, i.e., no anonymization ($p_a^Y = p_a^X$). The curve appears as a straight line, representing a power law on the log-log plot. When enabling $z$-anon ($z > 1$), we notice that the probability of observing uncommon attributes abruptly decreases with an evident knee. For example, if we observe the curve for $z = 100$ (green dashed line in the figure), already the $13^{th}$-ranked attribute is released with a probability below $10^{-6}$, while it appears on the original stream with probability $10^{-1}$. A higher $z$ moves the knee of the curve closer to the top-ranked attributes.

Conversely, increasing the number of users $U$ increases the lowest-ranked attributes probability to be published (with more users in the system, it is easier to satisfy the $z$ threshold). This would move to the right the curves' knee. We discuss the impact of $U$ later in Section 5.4.

In summary, the figure shows how $z$-anon prevents uncommon attributes from being released. Indeed, those attributes are released only when enough users are exposing them, hence only for popular attributes. In the following, we propose a probabilistic model to study how a $z$-anonymized data stream can result in a $k$-anonymized dataset with controllable probability.

## 4. Modeling *k*-anonymity

We now study the relationship between the $z$-anon and $k$-anon properties. Intuitively, $z$-anon ensures that each published value of an attribute $a$ has been exposed at least by $z$ users in the past time interval, while, with $k$-anon, any given record (i.e., the combinations of all user's attributes) must appear in the published data at least $k$ times. With high-dimensional data, the set of attribute combinations becomes extremely large, thus making $k$-anon tricky to guarantee. Here we show that, with a proper choice of $z$, it is possible to release data in which user results $k$-anonymized.

### *4.1. Getting to k-anon*

Given a specific realization $\overline{y}$ of a user, our goal is to derive the probability to observe at least other $k-1$ users in $\mathcal{U}$ having the same realization $\overline{y}$. If this happens, the user is $k$-anonymized.

Recall that we assume attributes to be independent. Thus each realization $\overline{y} = \{y_a\}_{a \in \mathcal{A}}$ happens with a probability $p_{\overline{y}}$, which results to be:

$$p_{\overline{y}} = \prod_{a \in \mathcal{A}} \left[ y_a \cdot p_a^Y + (1 - y_a) \cdot (1 - p_a^Y) \right]. \tag{4}$$

For any realization $\overline{y}$, the random variable representing the number of users with the same realization in the users' set (which we can model as $Q_{\overline{y}}$) is described by a Binomial distribution with parameters $U-1$ and $p_{\overline{y}}$: $Q_{\overline{y}} \sim Binomial(U-1, p_{\overline{y}})$.

From the point of view of an external observer which only accesses the privatized stream, a user has thus a probability of being $k$-anonymized which can be retrieved from the law of the total probabilities:

$$p_{k-anon} = \sum_{\overline{y}} \left[ 1 - \sum_{j=0}^{k-2} P\left[Q_{\overline{y}} = j\right] \right] \cdot p_{\overline{y}}. \tag{5}$$

In Equation 5, the probability for a user of finding at least $k-1$ other users with the same $\overline{y}$ is evaluated as the opposite of finding up to $k-2$ users. Then, we average this quantity over all the possible realizations of the random variable $\overline{Y}$, summing over all the $\overline{y}$ and multiplying by the respective $p_{\overline{y}}$ to obtain the final $p_{k-anon}$.

It is worth noticing that the cardinality of the possible realization set exponentially depends on the number of attributes $A$. In Appendix B, we discuss how to manage the model computational issue deriving from this, introducing a model approximation technique.

In summary, our model describes the probability that a data stream undergoing $z$-anon results in a dataset which respects the $k$-anon property. Although we can only provide probabilistic guarantees on the $k$-anonymization of the released data, our model allows one to study and control this probability as a function of the parameters.

Moreover, note that, even with no $z$-anon in place (i.e., $z = 1$), the model provides a general way to evaluate the probability of a data stream being $k$-anonymized in a transaction dataset with $U$ users and a catalog of $A$ attributes.

An analysis on the model results as compared to simulation ones is provided in Section 5.1 and Appendix  C.

### *4.2. Modeling Information loss*

Applying $z$-anon to an input stream decreases the amount of information the output stream carries with respect to the original non-anonymized stream. There is a trade-off between data privacy and data usability: if no anonymization is in place, the information provided by the final dataset will be maximum. On the other hand, if the data are anonymized, privacy is protected but information is lost in the process.

Here we consider the entropy as a measure of the amount of information a dataset contains. This metric derives from Information Theory [10], which defines the information brought by the occurrence of a symbol among a set of possible symbols by knowing the probability of such symbol to appear.

In our case, each symbol is a possible realization of $\overline{Y}$. We can hence use Equation 4 to compute the amount of information of the release, by evaluating its entropy as:

$$I = -\sum_{\overline{y}} p_{\overline{y}} \log(p_{\overline{y}}). \tag{6}$$

By using Equation 6, it is also possible to evaluate the information loss caused by anonymizing a data stream, which can be computed as the difference between the information of the non-anonymized stream and the information of the anonymized one.

## 5. Comparing $z$-anonymity and $k$-anonymity

In the following, we use our model to show the impact of the system parameters on the $z$-anon and $k$-anon properties. Our model provides $p_{k-anon}$ as a function of the scenario $(U, A, \lambda)$ and system parameters ($z, k, \Delta t$, which are under our control). As such, this function provides the probability a generic user is $k$-anonymized in the released data. In the following analyses, where not otherwise noted, we use the parameters listed in Table 2.

### *5.1. The impact of z*

We first focus on the impact of $z$. In Figure 3, we report how different values of $z$ result in different probabilities for a given user to be $k$-anonymized – and compare the results to the simulation ones. Different lines correspond to different values of $k$. The larger is $z$, the higher is $p_{k-anon}$. Focusing on $k = 2$ (blue solid line), $p_{k-anon}$ increases starting from $z = 100$. With $z = 250$, the probability of finding at least a user with an identical set of released attributes is already 0.8. When $z > 350$, $p_{k-anon}$ approaches 1, giving the almost certainty

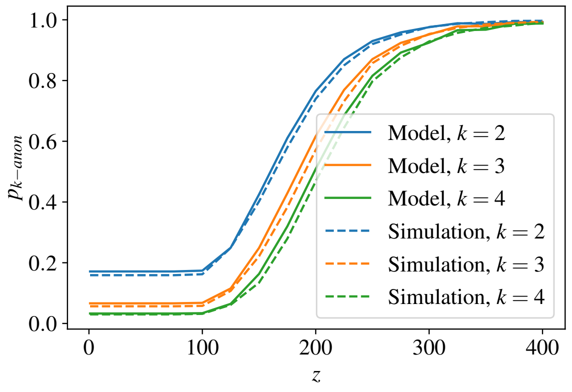


Figure 3: $p_{k-anon}$ changing $z$, for different $k$ values. Exact model results and 10 iterations simulation averages are reported.

that the whole release is $k$-anonymized (with $k = 2$). For $k = 3, 4$ (orange and blue line, respectively), $p_{k-anon}$ exhibits a similar behaviour, with $p_{k-anon}$ decreasing when increasing $k$, as it becomes harder to find $k$ identical users by chance. In summary, one can tune $z$ to enforce a desired $k$ and $p_{k-anon}$ on the released data.

Figure 3 also shows a comparison between the model and the result of simulations with the same parameters. To obtain the dashed lines (i.e., the simulation results), ten simulation are performed, with different seeds. For each of them, we evaluate $p_{k-anon}$ for $k = 2, 3, 4$ and average the outcomes over all the simulations. The dashed lines and the corresponding solid ones follow the same trend and match almost perfectly - the differences being caused by the small number of simulation performed. This result also validates the essential correctness of the model: Appendix C provides more details on the simulation process and on its adherence to the model results.

## *5.2. The impact of* $\Delta t$

The second design parameter one must set is $\Delta t$, the time window on which $z$-anon runs. In Figure 4, we show how $p_{k-anon}$ varies while increasing $\Delta t$, with different values of $k$. Intuitively, the larger the time period, the lower the probability of a user being $k$-anonymized. A large time window allows also unpopular attributes to satisfy the $z$-anon property and be published, thus decreasing the $p_{k-anon}$. Conversely, with a narrow time window, only the most popular attributes result non $z$-private, with a positive effect on $p_{k-anon}$. In summary, $\Delta t$ is another way for tuning the fraction of $z$-private attributes, with a direct impact on the $k$-anon of the released data. Since the effects of tuning $z$ and $\Delta t$ are interchangeable, from now on we will focus on $z$, knowing that acting on $\Delta t$ would have an analogous effect. Note also that the choice of $\Delta t$ impacts

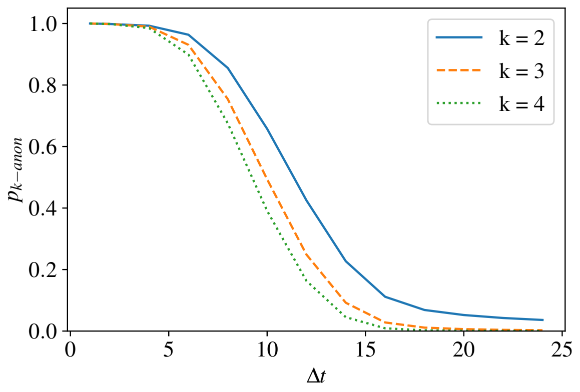


Figure 4: $p_{k-anon}$ changing $\Delta t$, for different $k$ values.

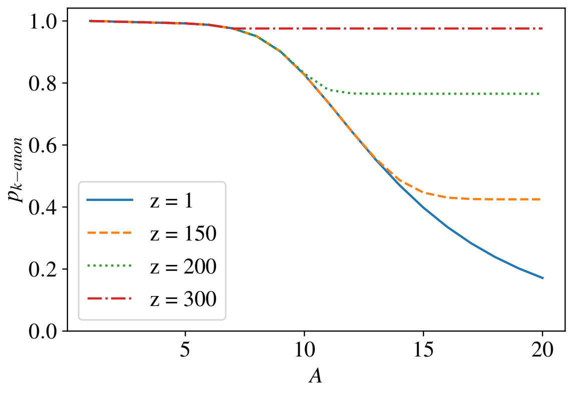
also the system memory and data structure size. As such, one would choose $\Delta t$ on the specific use case, and regulate $z$ for reaching the desired privacy level.

## *5.3. The impact of A*

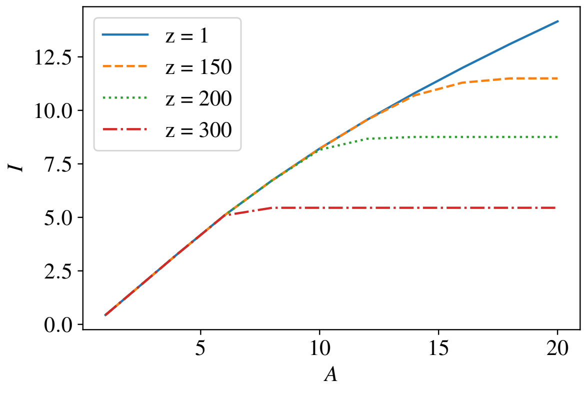
Then, we study the impact of the size of the catalog of attributes $A = |\mathcal{A}|$. In Figure 5a we show how the probability $p_{k-anon}$ of a user being $k$-anonymized varies with $A$ through the impact of $z$-anon. We consider a system where only the top $A$ ranked attributes exist. As such, by increasing $A$, we add more and more infrequent attributes. Intuitively, a large number of attributes makes it hard to find users with the same output realization $\overline{y}$. However, with $z$-anon, the catalog size results limited and thus it plays a marginal role (see Figure 2), and, as such, infrequent attributes are rarely published.

In Figure 5a, $p_{k-anon}$ starts at 1, when few attributes are present, and the number of their possible combinations is low. When $A$ increases, less frequent attributes start to appear. The possible combinations of attributes explode exponentially. With $z = 1$, i.e., no $z$-anon in place, the probability of finding identical users rapidly goes to 0. Enabling $z$-anon, we prevent rare attributes from being released, thus limiting the number of combinations – see dashed lines.

Figure 5b shows the effect of the number of attributes on the entropy of the resulting dataset as defined in Equation 6, for different values of $z$. The entropy increases with the number of attributes $A$. If the attributes were equally probable (uniform distribution), the entropy will scale linearly with $A$. However, given the power law of attributes, the increase is sub-linear. Applying $z$-anon to the input data stream limits the growth of the entropy, preventing the appearance of infrequent combinations. Since least-likely attributes are protected by $z$-anon, they will not be published and will not add information to the final dataset. Therefore, the entropy tends to converge with higher $A$. The higher



(a) $p_{k-anon}$ changing $A$, considering different $z$ values.



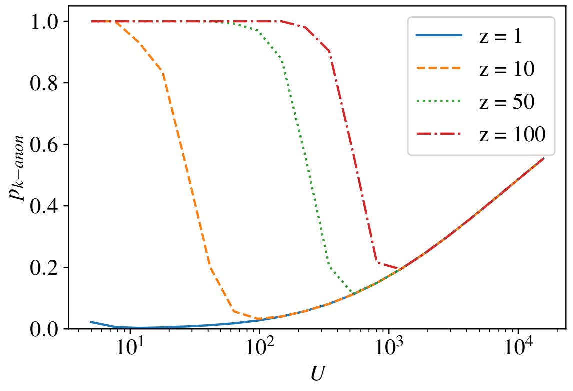
(b) The entropy $I$ of the output $z$-anon dataset changing $A$, for different $z$ values.

Figure 5: The impact of $A$.

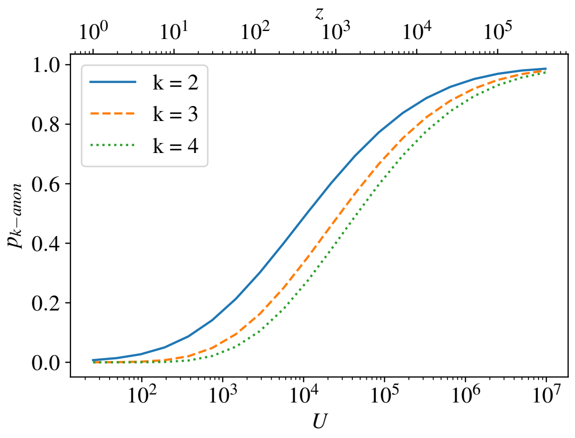
$z$, the fewer the released information. In other words, by tuning $z$ (or $\Delta t$), it is possible to regulate the amount of information in the released data.

## 5.4. *The impact of $U$*

We now study how the number of users $U$ impacts their probability to appear $k$-anonymized in the released data. In Figure 6a, we show how $p_{k-anon}$ varies when increasing $U$, for different values of $z$. We notice the concurrence of two effects: first, with low values of $U$, even a small $z$ ensures that users are $k$-anonymized, as rare realization have few chances to appear. Increasing the number of users leads to a decrease of $p_{k-anon}$. This happens because the large number of users causes even less-popular attributes to overcome the $z$ threshold, hence increasing the number of possible combinations and, thus, decreasing $p_{k-anon}$. At a certain point (depending on $z$) all the attributes are likely to be



(a) $p_{k-anon}$ changing $U$, for different $z$ values.



(b) $p_{k-anon}$ increasing proportionally $U$ and $z$ ($z = U/25$), for different $k$ values.

Figure 6: The impact of $U$ on $p_{k-anon}$.

published, and a second effect steps in: adding new users, each combination has a higher probability of appearing more than once, thus improving $p_{k-anon}$.

To prevent $p_{k-anon}$ from decreasing with a large $U$, we now suppose we set $z$ proportional with $U$. In Figure 6b, we show $p_{k-anon}$ with increasing number of users (bottom $x$-axis) and, consequently, increasing $z$ (top $x$-axis), as we set $z = U/25$. Focusing on the solid blue line ($k = 2$), we notice how $p_{k-anon}$ grows with $U$, reaching values close to 1 with very large $U$ (notice the log $x$-scale). With a higher $k$ (dashed lines), the $p_{k-anon}$ is only shifted to larger values of $U$. The figure shows that a large $U$ leads to better guarantees of $k$-anon as far as $z$ is set proportionally. As such, it is fundamental to consider the number of users in the system to properly set the $z$-anon parameters and, in turn, successfully achieve $k$-anon. Conversely, if $z$ does not grow with $U$, performance guarantees worsen (see Figure 6a).

## 6. Extension to user classes

The model we presented in Section 3.2 relies on the assumptions that the user activity rate is constant over time (i.e., it follows a homogeneous Poisson Process), their behaviour is independent (i.e., users' interactions do not depend one on the other), and homogeneous (i.e., every user acts in the same way). In this section, we relax the last assumption, introducing the concept of *classes* of users. We assume that $C$ classes exist and that each user belongs to one and one only class $c \in \{1, ..., C\}$. Users in the same class behave homogeneously and potentially differently from those of other classes.

We consequently extend our model to consider the dependence on the class $c$ of the user we are considering. In the notation, we will add the subscript of the class $c$ to variables and probabilities. Hence, in each class $c$, there are $U_c$ users.

The attribute exposing rate now depends on the user's class, thus $\lambda_{a,c}$. Consequently, $p^X_{a,c}$ is the probability a user in $c$ exposes $a$ in $\Delta t$. Let $p^O_{a,c}$ be the probability that this attribute $a$ satisfies the $z$-anon constraint. This probability requires a different computation since it depends on the class $c$ of the user and on users in other classes. The $z$-anon constraint is satisfied if there are at least $z - 1$ other users, among all classes, that have exposed $a$ in the past $\Delta t$. This can be written as in Equation 3 as the complementary event where the users exposing $a$ do not add up to $z - 1$.

To this end, we have to find all the possible combinations of users in the different classes exposing $a$. By denoting as $n_i \in \{0, \dots, U_i\}$ the number of users of class $i$ that have exposed a given attribute $a$ in the previous $\Delta t$, we can define the set $\mathcal{C}(z)$ of C-uples, whose sum does not exceed $z - 2$:

$$\mathcal{C}(z) = \left\{ (n_1, \dots, n_C) : \sum_{i=1}^{C} n_i \leq z - 2 \right\}$$

Then:

$$p_{a,c}^{O} = 1 - \sum_{(n_1,\ldots,n_C)\in\mathcal{C}(z)} \left( \prod_{i=1}^{C} P\left[B_{i,c,a} = n_i\right] \right),$$

where $B_{i,c,a} \sim Binomial(U_i - \delta_{i,c}, p_{a,c}^{X})$ is the random variable representing the number of users in class $i$ that have exposed $a$ in the previous $\Delta t$. We remove one user when considering the same class $c$ through the Kronecker delta $\delta_{i,c}$, as we already imposed that such user is exposing $a$.

Consequently, $p_{a,c}^{Y}$ is the probability a user in $c$ publishes at least once $a$ in $\Delta t$, and $p_{\overline{y},c}$ is the probability a user in $c$ has the realization $\overline{y} = \{y_a\}_{a\in\mathcal{A}}$.

Finally, extending $p_{k-anon}$ to $p_{k-anon,c}$ requires a few steps. Similar to what happens in $p_{a,c}^{O}$ the probability for a user of being $k$-anonymized depends on whether it is possible to find in the release at least $k-1$ other users with the same realization – regardless of the class they belong to. We can reuse the definition of $\mathcal{C}$, now with parameter $k$, i.e., $\mathcal{C}(k)$. Then, the probability for a user in class $c$ of being $k$-anonymized follows:

$$p_{k-anon,c} = \sum_{\overline{y}} \left\{ \left[ 1 - \sum_{(n_1,\ldots,n_C)\in\mathcal{C}(k)} \left( \prod_{i=1}^{C} P[Q_{i,c,\overline{y}} = n_i] \right) \right] \cdot p_{\overline{y},c} \right\},$$

where $Q_{i,c,\overline{y}} \sim Binomial(U_i - \delta_{i,c}, p_{\overline{y},i})$ is the random variable representing the number of users in class $i$ with the same realization of attributes $\overline{y}$ in the previous $\Delta t$ as our target user.

In the following, we explore two use cases considering two classes:

- Classes of activity: users belonging to one class are more active than users belonging to the other one;

- Classes of interest: users in different classes have different interests.

## *6.1. Classes of activity*

In this scenario, users of the first class are more active than users of the second class. We define as $\lambda_{a,2}/\lambda_{a,1}$ the level of imbalance between classes. To provide a fair comparison, we want the overall average exposition rate $\lambda_a$ to remain constant. This is verified if the following condition is satisfied:

$$U_1 \cdot \lambda_{a,1} + U_2 \cdot \lambda_{a,2} = (U_1 + U_2) \cdot \lambda_a,$$

where $\lambda_a$ is the overall exposing rate of the users in $\mathcal{U}$ for $a$. Recall that $U_1$ and $U_2$ define respectively the number of users belonging to class 1 and the number of users belonging to class 2.

From the $z$-anon point of view, when $\lambda_{a,2} << \lambda_{a,1}$, this results in users of class 2 exposing few attributes, while the majority comes from users of class 1. Overall, this implies that the "active" population is reduced, thus less tuples

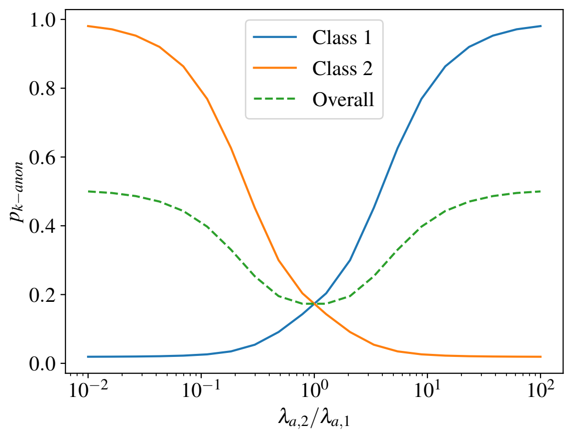


Figure 7: $p_{k-anon}$ with classes of activity ($z = 50$, $U_1 = 500$, $U_2 = 500$).

$(t, u, a)$ can be published. This in turn will increase the probability of a user being $k$-anonymous. Figure 7 shows this effect showing $p_{k-anon,1}$, $p_{k-anon,2}$ and the resulting overall $p_{k-anon}$. The $x$-axis is log scale, and, as such, the Figure appears symmetric with respect to $\lambda_{a,2}/\lambda_{a,1} = 1$. Users are likely not to expose any attribute for the least active class, with thus a high probability of being $k$-anonymized. Conversely, $p_{k-anon}$ decreases for the most active class, as its users are more active to compensate for the inactive users. Overall, the figure shows that $p_{k-anon}$ benefits when the classes are strongly unbalanced (green dashed line). Conversely, the more similar the class rates become, the more the situation gets close to the single-class scenario. Indeed, when $\lambda_{a,2}/\lambda_{a,1} = 1$, the $p_{k-anon}$ value reaches the same value shown in Figure 3.

### *6.2. Classes of interest*

We consider a second use-case, where users belonging to one class are interested in a set of attributes, while the other users are interested in another set.

One possibility is to divide the attributes into two groups, for which the two classes of users have different interest, which we model with a different probability of publishing such attribute. We create two groups of attributes: recalling that $\lambda_{a_r}$ is the exposing rate of attribute $a$ in position $r$ in the popularity rank, we assign attributes in even positions of the rank to one group, and attributes in odd positions to the other group. Then, we assign the first group of attributes a rate $\eta \cdot \lambda_{a_r}$, $\eta \in [0, 1]$ to the first class of users; and $(1-\eta) \cdot \lambda_{a_r}$ to the second class of users. Conversely, we assign the second group of attributes a rate $(1-\eta) \cdot \lambda_{a_r}$ to the first class of users, and a rate $\eta \cdot \lambda_{a_r}$ to the second class of users. Table 3 formalizes the scenario of the example. If $\eta = 0.5$, the two classes become

| | $\lambda_{a_r,1}$ | $\lambda_{a_r,2}$ |
|---|---|---|
| **Even attributes** $\{a_r : r = 2k, k \in \mathbb{N}\}$ | $\eta \cdot \lambda_{a_r}$ | $(1-\eta) \cdot \lambda_{a_r}$ |
| **Odd attributes** $\{a_r : r = 2k+1, k \in \mathbb{N}\}$ | $(1-\eta) \cdot \lambda_{a_r}$ | $\eta \cdot \lambda_{a_r}$ |

Table 3: Attributes rates for different classes of interest used in the example.

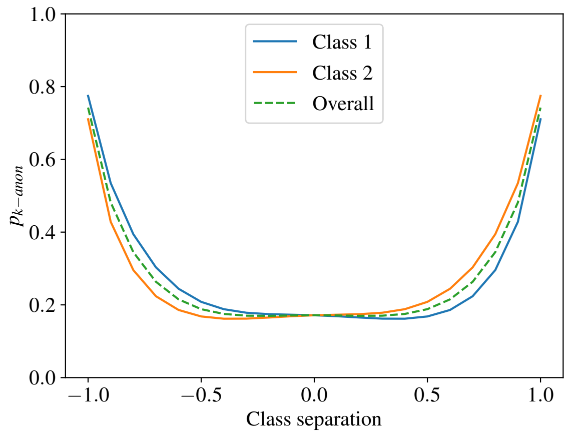


Figure 8: $p_{k-anon}$ with classes of interest ($z = 50$, $U_1 = 500$, $U_2 = 500$).

the same and, consequently, we obtain the same $p_{k-anon}$ as for the single-class scenario. We define *class separation* as the difference $\eta - (1 - \eta) = 2\eta - 1$.

In Figure 8, we show how $p_{k-anon}$ varies with different *class separation* values. Similarly to the previous use case, splitting the users into classes increases the $k$-anon probability. In this case, though, both classes benefit from class separation. Increasing the class separation has the twofold effect of i) reducing the number of attributes that a user is likely to expose and ii) generating two dissimilar groups of users. As such, we conclude that a scenario where multiple groups of users that expose different attributes eases the achievement of $k$-anon.

As shown if Figure 5, the greater the $p_{k-anon}$, the lower the entropy of the released information. Although we do not present the figures for the sake of brevity, the case with classes of users follow the same principle. Where the classes are highly imbalanced in terms of either interest or activity (and the $p_{k-anon}$ is larger) the quantity of information of the output decreases.

## 7. Discussion and Conclusions

In this paper, we studied $z$-anon, a novel anonymization property for data streams. It operates with high dimensional data, organized in transactions (atomic information about users) and with the constraint of zero-delay processing. The idea at the base of $z$-anon is to hide $z$-private attributes, i.e., those associated with less than $z - 1$ other users in the past $\Delta t$, which an attacker could use for re-identification. A data stream undergoing $z$-anon is immediately available with zero delay to the consumer.

$z$-anon only prevents users' re-identification for an attacker that leverages an uncommon attribute. It is designed uniquely to avoid such kind of re-identification. Therefore it does not consider other types of privacy attacks, e.g., targeting the timing or order at which users' attributes appear in the data stream. Moreover, $z$-anon does not consider combinations of $z$-anonymized attributes, treating them independently. Here, we provide a probabilistic model that shows that users can also be $k$-anonymized with a controllable probability even if an attacker knows the entire set of released attributes. With this, we offer guidelines to tune the system parameters and guarantee $k$-anon. As a side achievement, the model can also provide the probability of a user in a set being $k$-anonymized, according to the distribution probabilities of the attributes – whether $z$-anon is in place or not. We are aware that $k$-anon is a limited privacy property, and more sophisticated alternatives exist (notably $l$-diversity, $t$-closeness, and differential privacy); nonetheless, it is the most popular and easy-to-understand one, thus comparing $z$-anon to it is the most obvious choice.

In this context, our model allows the data curator to understand the properties of the released data and manage the trade-off between privacy and data utility. Notice that the model needs to know the statistical properties of the data stream, namely $U$, $A$, $\lambda_a$. Those can be directly estimated from the data in case they are unknown. Once $U$, $A$, $\lambda_a$ have been estimated, the data curator can select the appropriate values of $\Delta t$ and $z$ parameters to obtain a target $k$. In case the stream is not stationary, this process shall be repeated over time.

Our results pave the way towards manifold research directions. First, our probabilistic model can be employed not only to assess how $z$-anon results into $k$-anon, but also to dynamically adjust $z$ to achieve a desired $k$-anon. Moreover, we only considered blurring $z$-private attributes. Alternatively, we could generalize the attributes to pass the $z$-threshold. Furthermore, we argue that we can achieve better data utility while avoiding users' re-identification even if some $z$-private items are released. This can be obtained by introducing perturbations in the released data, e.g., by inserting noise in the data stream or modifying some of the associations between users and attributes, as done in [22, 23].

$z$-anon is weaker than $k$-anon, as it independently operates on users' attributes without considering their combination. Here, we provided a model to evaluate the probability of users being $k$-anonymized, with the $z$-anon filter in place. Within this, the data curator can tune the trade-off between privacy and data utility. We used our model to study the impact of the scenario, such as the number and frequency of attributes or the size of the user base. The model supports heterogeneous users, divided into classes, and we show that an inherent differentiation among the users positively impacts obtaining the $k$-anon. Although we are well aware of the $k$-anon flaws in terms of privacy, we chose to compare $z$-anon to it to provide a wide-known, simple-to-understand benchmark.

## Acknowledgements

The research leading to these results has been funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 871370 (PIMCity project) and the SmartData@PoliTO center for Data Science technologies.

## References

[1] European Parliament and Council of European Union, Directive 95/46/EC. General Data Protection Regulation, `http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf` (Last accessed May 25, 2021) (2016).

[2] California State Legislature, California Consumer Privacy Act of 2018, `https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375` (Last accessed May 25, 2021) (2018).

[3] 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) (Jun. 2010). `doi:10.1145/1749603.1749605`. URL `https://doi.org/10.1145/1749603.1749605`

[4] L. Sweeney, Guaranteeing anonymity when sharing medical data, the Datafly System, in: Proceedings of the AMIA Annual Fall Symposium, American Medical Informatics Association, 1997, p. 51.

[5] A. Narayanan, V. Shmatikov, in: 2008 IEEE Symposium on Security and Privacy (sp 2008).

[6] P. Samarati, Protecting respondents identities in microdata release, IEEE Transactions on Knowledge and Data Engineering 13 (6) (2001) 1010–1027.

[7] J. Cao, B. Carminati, E. Ferrari, K. Tan, CASTLE: Continuously Anonymizing Data Streams, IEEE Transactions on Dependable and Secure Computing 8 (3) (2011) 337–352.

[8] T. Favale, M. Trevisan, I. Drago, M. Mellia, $\alpha$-mon: Traffic anonymizer for passive monitoring, IEEE Transactions on Network and Service Management (2021) 1–1`doi:10.1109/TNSM.2021.3057927`.

[9] N. Jha, T. Favale, L. Vassio, M. Trevisan, M. Mellia, z-anonymity: Zero-Delay Anonymization for Data Streams, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 3996–4005. `doi:10.1109/BigData50022.2020.9378422`.

[10] L. Brillouin, Science and information theory, Courier Corporation, 2013.

[11] J. Li, B. C. Ooi, W. Wang, Anonymizing Streaming Data for Privacy Protection, in: 2008 IEEE 24th International Conference on Data Engineering, 2008, pp. 1367–1369.

[12] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, Y. Jia, Continuous Privacy Preserving Publishing of Data Streams, in: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, ACM, New York, NY, USA, 2009, p. 648–659. doi:10.1145/1516360.1516435.
URL https://doi.org/10.1145/1516360.1516435

[13] J. Zhang, J. Yang, J. Zhang, Y. Yuan, KIDS:K-anonymization data stream base on sliding window, in: 2010 2nd International Conference on Future Computer and Communication, Vol. 2, 2010, pp. 311–316.

[14] J. Tekli, B. Al Bouna, Y. B. Issa, M. Kamradt, R. Haraty, (k, l)-Clustering for Transactional Data Streams Anonymization, in: International Conference on Information Security Practice and Experience, Springer, 2018, pp. 544–556.

[15] A. B. Sakpere, A. V. D. M. Kayem, Adaptive buffer resizing for efficient anonymization of streaming data with minimal information loss, in: 2015 International Conference on Information Systems Security and Privacy (ICISSP), 2015, pp. 1–11.

[16] A. Otgonbayar, Z. Pervez, K. Dahal, S. Eager, K-VARP: K-anonymity for varied data streams via partitioning, Information Sciences 467 (2018) 238–255. doi:10.1016/j.ins.2018.07.057.
URL https://linkinghub.elsevier.com/retrieve/pii/S0020025518305772

[17] A. Otgonbayar, Z. Pervez, K. Dahal, Toward Anonymizing IoT Data Streams via Partitioning, in: 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2016, pp. 331–336.

[18] M. Khavkin, M. Last, Preserving Differential Privacy and Utility of Nonstationary Data Streams, in: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 2018, pp. 29–34.

[19] J. Domingo-Ferrer, J. Soria-Comas, R. Mulero-Vellido, Steered Microaggregation as a Unified Primitive to Anonymize Data Sets and Data Streams, IEEE Transactions on Information Forensics and Security 14 (12) (2019) 3298–3311.

[20] M. Chamikara, P. Bertok, D. Liu, S. Camtepe, I. Khalil, An efficient and scalable privacy preserving algorithm for big data and data streams, Computers & Security 87 (2019) 101570. doi:https://doi.org/10.1016/j.cose.2019.101570.
URL http://www.sciencedirect.com/science/article/pii/S0167404818313683

[21] S. Kim, M. K. Sung, Y. D. Chung, A framework to preserve the privacy of electronic health data streams, Journal of Biomedical Informatics 50 (2014) 95 – 106, special Issue on Informatics Methods in Medical Privacy. doi:https://doi.org/10.1016/j.jbi.2014.03.015.
URL http://www.sciencedirect.com/science/article/pii/S1532046414000823

[22] S. A. Abdelhameed, S. M. Moussa, M. E. Khalifa, Restricted Sensitive Attributes-based Sequential Anonymization (RSA-SA) approach for privacy-preserving data stream publishing, Knowledge-Based Systems 164 (2019) 1 – 20. doi:https://doi.org/10.1016/j.knosys.2018.08.017.
URL http://www.sciencedirect.com/science/article/pii/S0950705118304131

[23] J. Wang, C. Deng, X. Li, Two Privacy-Preserving Approaches for Publishing Transactional Data Streams, IEEE Access 6 (2018) 23648–23658.

[24] Nikhil Jha, $z$-anonymity model.
URL https://github.com/nikhiljha95/zanonymity

[25] L. A. Adamic, B. A. Huberman, A. Barabási, R. Albert, H. Jeong, G. Bianconi, Power-law distribution of the world wide web, science 287 (5461) (2000) 2115–2115.

[26] A. Meyerson, R. Williams, On the Complexity of Optimal K-Anonymity, in: Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 223–228. doi:10.1145/1055558.1055591.
URL https://doi.org/10.1145/1055558.1055591

[27] Tommaso Bacconi, Nikhil Jha, Martino Trevisan, $z$-anonymity implementation.
URL https://pypi.org/project/zanon/

## Appendix A. Implementation and complexity

Our goal is to design an algorithm to achieve $z$-anon which satisfies the following requirements:

- **Zero delay:** the anonymization property should be achieved without introducing a delay in publishing the anonymized stream. In other words, we want to make an atomic decision. All approaches based on the processing of batches of observations are not applicable, as they need to store and process the entire batch before the release.
- **Efficient algorithm for high dimensional data:** the anonymization property shall be achieved with an efficient algorithm, allowing the deployment at high speed and large volume of data with off-the-shelf computing

capabilities. It is important to carefully build an algorithm working with efficient data structures to obtain the necessary information as quickly as possible. Moreover, users might expose a large set of attributes, whose number is not known *a priori*.

The algorithm we propose generalizes the approach presented in our previous work [8]: the attributes $a$ are stored as a hash table $\mathcal{H}$, with linked lists to manage collisions. Each value $\mathcal{H}(a)$ in the hash table contains three elements:

- metadata about $a$;
- a Least Recently Used list $\text{LRU}_a$ of tuples $(t, u)$;
- a hash table $\mathcal{V}_a$ to track the users that exposed $a$.

The idea is to minimize the time spent searching into the data structures, therefore reducing the memory accesses. By assuming that the number of attributes $a$ is one order of magnitude smaller than the hash structure dimension, collisions are infrequent, and consequently, the total computational cost is $O(1)$ for each incoming observation.

The $\mathcal{H}(a)$'s metadata include the counter $c_a$ and the reference for the $\text{LRU}_a$ first and last attribute. Referring to Algorithm 1, once an observation $(t, u, a)$ arrives, the value $a$ should be inserted in the hash table, if not already present (lines 2-6), otherwise an update should be performed (lines 7-16). The hash value is calculated and the access to the table is done in $O(1)$.

If the user $u$ exposes an attribute $a$ for the first time in the previous $\Delta t$, the user $u$ is inserted into $\mathcal{V}_a$ in $O(1)$, $c_a$ is increased by one and the tuple $(t, u)$ is inserted on top of the $\text{LRU}_a$ in $O(1)$ thanks to the aforementioned references (lines 8-11). If $u$ was already present in $\mathcal{V}_a$ and in $\text{LRU}_a$ with value $(t', u)$, we replace $t'$ with $t$ and the tuple $(t, u)$ is moved on the top of the $\text{LRU}_a$. Again all is done in $O(1)$ (lines 12-14).

Last, to evict old entries and consequently decrease $c_a$, we traverse the LRU in reverse order: we remove each tuple $(t', u')$ where $t' < t - \Delta t$, and we decrease $c_a$ accordingly (lines 18-22). At last, if $c_a \geq z$ the observation $(t, f(t, u), a)$ is released (lines 23-24). The $f(t, u)$ needs an explanation: every $\Delta t$, users' identifiers are rotated, such that the ID related to a user $u$ at a time $t_0$ will no more be related to $u$ at $t_0 + \Delta t$. The user identifiers thus depend on the time at which the tuple is published; the attacker will not be able to track the behaviour of the same user after a $\Delta t$.

Notice that $k$-anon has been proved [26] to be an *NP-Hard* problem. Differently, $z$-anon property can be achieved for each observation with $O(1)$ complexity with properly sized hash-tables. Implementation proposed in [8] allows to manage in real time a 40 Gbit/s stream with common hardware. As part of the PIMCity project, we provide also a Python library to let interested users to adopt $z$-anon [27].

**Algorithm 1** Pseudo code of the algorithm to achieve $z$-anon.

```
1: Input: $(t, u, a)$
2: if $a \notin \mathcal{H}$ then
3:     $\mathcal{H} \leftarrow \mathcal{H} \cup a$ //new attribute: insert it for the first time
4:     $\mathcal{V}_a \leftarrow \{u\}$ //insert new user u
5:     $LRU_a \leftarrow (t, u)$
6:     $c_a = 1$
7: else
8:     if $u \notin \mathcal{V}_a$ then
9:         $\mathcal{V}_a \leftarrow \mathcal{V}_a \cup \{u\}$ //insert new user u
10:        $c_a \leftarrow c_a + 1$ //add new user
11:        $LRU_a \leftarrow (t, u)$
12:    else
13:        $(t', u) \leftarrow (t, u)$ //update timestamp of user u
14:        move $(t, u)$ on top of $LRU_a$
15:    end if
16: end if
17: //Always evict old users
18: for $((t', u') = \text{last}(LRU_a);\ t' < t - \Delta t;\ (t', u')$=next$)$ do
19:     remove $(t', u')$ from $LRU_a$
20:     remove $(u')$ from $\mathcal{V}_a$
21:     $c_a \leftarrow c_a - 1$
22: end for
23: if $(c_a \geq z)$ then
24:     OUTPUT $(t, f(u, t), a)$
25: end if
```

## Appendix B. Model approximation

As it emerges from Equation 5, the evaluation of $p_{k-anon}$ depends on the number of possible realizations $\overline{y}$ of $\overline{Y}$. In a configuration with $A$ binary attributes, there are $2^A$ of such possible realizations, and their enumeration represents a computational bottleneck. In the following, we propose an approximation strategy to make the computation of Equation 5 practical. We introduce two parameters, $\theta_1$ and $\theta_2$. The first operates to limit the number of attributes to consider. The second limits the number of realizations to evaluate. The rationale is that many realizations have usually a negligible probability to happen, allowing us to neglect them while keeping unchanged the model accuracy.

*Effective attributes*

Focus first on $\theta_1$. Here we leverage the typically heavy-tailed nature of attribute popularity. Intuitively, the least-popular ones will be so rare that no realization $\overline{y}$ would contain them. $z$-anon will exacerbate this, since it will further decrease the publications of such unpopular attributes.

Let the *effective attributes* be those we expect to be exposed at least by $\theta_1$ user in $\Delta T$. Let $A_{eff} \leq A$ be their number. Considering the expected value, we have that an attribute $a$ is effective if $\mathbb{E}[U \cdot p_a^Y] \geq \theta_1$, from which $p_a^Y \geq \theta_1/U$.

We can filter those attributes for which $p_a^Y < {}^{\theta_1}/_U$. In a nutshell, we discard all realizations where non-*effective attributes* appear and, thus, reduce their number from $2^A$ to $2^{A_{eff}}$.

*Effective realizations*

Even the realizations derived from the $2^{A_{eff}}$ attributes may not all be worth an evaluation in Equation 5. We thus design an algorithm to reduce the number of realizations to consider, by enumerating the most likely ones and discarding the rarest ones. To this end, we organize all the possible realizations in a tree. Let the root realization be $\overline{y}_0$. We show a toy example in Figure B.9, for three effective attributes ($A_{eff} = 3$), and use it as a running example. The root node ($\overline{y}_0$) holds the most probable realization. Hence, $\overline{y}_0 = \{y_a\}_{a \in \mathcal{A}}$, where:

$$y_a = \begin{cases} 1, & \text{if } p_a^Y \geq 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{B.1}$$

In our example, $p_a^Y < 0.5, \forall a$, and the most probable realization is $[0, 0, 0]$. $\overline{y}_0$ has three child nodes, each obtained by changing a single attribute. We arrange the children from the most probable to the least probable. The probability of these realizations depends on the distance of $p_{a'}^Y$ to the 0.5 threshold, where $a'$ is the attribute to change. For instance, take attributes $a_1$, $a_2$ and $a_3$. Let $p_{a_1}^Y = 0.49$, $p_{a_2}^Y = 0.1$ and $p_{a_3}^Y = 0.001$. $a_1$ will have a much larger probability of having value 1 than $a_2$ and $a_3$: the probability of the child node with the parent's $a_1$ being 1 will be larger than the one of the child with $a_2$ or $a_3$ set to 1. In a nutshell, we sort the attributes by their probability of changing from their most likely state, i.e., by $|p_a^Y - 0.5|$. Here, $|p_0^Y - 0.5| \leq |p_1^Y - 0.5| \leq |p_2^Y - 0.5|$.

We repeat the procedure recursively on all child nodes, building the three with the depth-first search strategy. When we examine a node, we exclude those realizations already present on parents or siblings. In the example, consider the $\overline{y}_1$ node. In this case, the first attribute cannot be modified again, and $\overline{y}_1$ has only two children, $\overline{y}_{11}$ and $\overline{y}_{12}$. Similarly, when we land on $\overline{y}_2$, we cannot obtain a child by changing the first attribute, as this realization will have been already covered in the sibling $\overline{y}_1$.

We formalize the properties of the realization tree as follows:

- The probability of a parent realization is always greater than the probability of its children.
- The probabilities of siblings' realizations decrease from the most-probable-to-change to the least-probable-to-change.

These two properties allow us to adopt an efficient strategy to neglect unlikely realizations: given a node, its children and siblings on the right side have a lower or equal occurrence probability. Thus, we can efficiently prune the tree: if a node in the tree has a probability to be observed below a threshold, we prune all its children and rightmost siblings, returning to the parent and speeding up the computation.

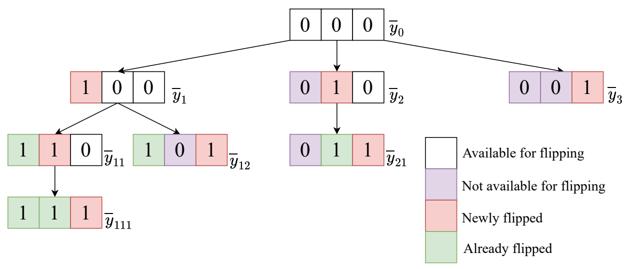


Figure B.9: An example of the realization tree. We assume that the changing probability decreases from the leftmost attribute to the rightmost one.

To set this threshold, we start from the probability of a realization in a scenario where all the $p_a^Y$ are equally probable – hence, all the realizations appear with the same probability: $1/2^{A_{eff}}$. We thus set the threshold to $\theta_2 \cdot 1/2^{A_{eff}}$, where $\theta_2$ allows one to tune the trade-off between execution speed and model accuracy. Indeed, if the threshold is too high, not enough realizations will be considered, and the model will significantly differ from the true value. Conversely, with a low threshold, a multitude of potentially negligible realizations must be evaluated.

Notice that it is easy to recognize a poor choice of $\theta_2$, by summing the probability of considered realizations (those belonging to the three after pruning), and imposing it to be at least – for instance – 0.98. In other words, we impose that:

$$\sum_{\overline{y}:p_{\overline{y}} \geq \frac{\theta_2}{2^{A_{eff}}}} p_{\overline{y}} \geq 0.98 \tag{B.2}$$

In our experiments, we use a greedy algorithm to find $\theta_2$ such that Equation B.2 holds.

## Appendix C. Model validation

To assess the validity of the model, we compare its results with those obtained by simulating the $z$-anon mechanism. To perform the simulation, we randomly generated an input trace that emulates a stream of tuples $(t, u, a)$, with $U = 1\,000$, $A = 20$, $\lambda_{a_r} = {}^{0.2}/_{r}$. We then process the input trace via the $z$-anon mechanism, as described in Algorithm 1. At last, we collect the published tuples and evaluate the fraction of the 1000 users that result 2-anonymized as an estimate of $p_{k-anon}$.

In Figure C.10, we compare the results of the simulation with those of the model, considering both the exact and the approximated version. The complete list of scenario parameters is available in Table 2.

In Figure C.10, each point represents the $p_{k-anon}$ as obtained by each simulation, each with a different seed. The solid blue line indicates the average of the simulations while the solid orange and green lines report the estimation obtained by the exact and the approximated model, respectively. The last two

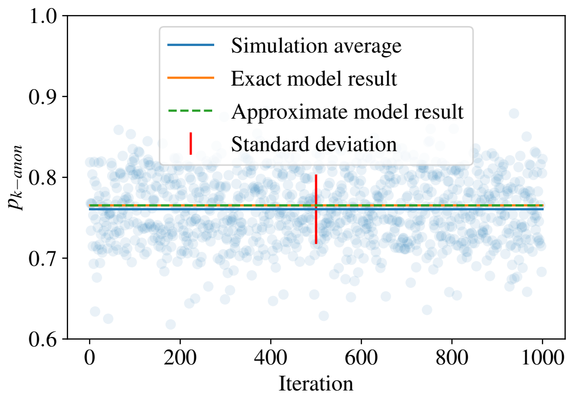


Figure C.10: The $p_{k-anon}$ as evaluated by differently-seeded simulations, compared with the model results.

return the same result. The average $p_{k-anon}$ of the simulation results is within 0.005 from the exact model one, with a standard deviation of 0.04, indicated in Figure C.10 as a vertical red bar.

It is worth noting that for simulation we take care of discarding the initial transient of duration $\Delta t$, during which the system starts accumulating observations, with no eviction happening.