**Basic Data Anonymization Techniques and K-Anonymity Model**

Table of Contents

[Anonymization Process 3](#_Toc63379923)

[Anonymization Techniques 4](#_Toc63379924)

[Attribute Suppression 5](#_Toc63379925)

[Record Suppression 6](#_Toc63379926)

[Character Masking 7](#_Toc63379927)

[Pseudonymization 8](#_Toc63379928)

[Generalization 9](#_Toc63379929)

[Swapping 10](#_Toc63379930)

[Data Perturbation 11](#_Toc63379931)

[Data Aggregation 12](#_Toc63379932)

[Synthetic Data Generation 12](#_Toc63379933)

[K-Anonymity Model 13](#_Toc63379934)

[Description 13](#_Toc63379935)

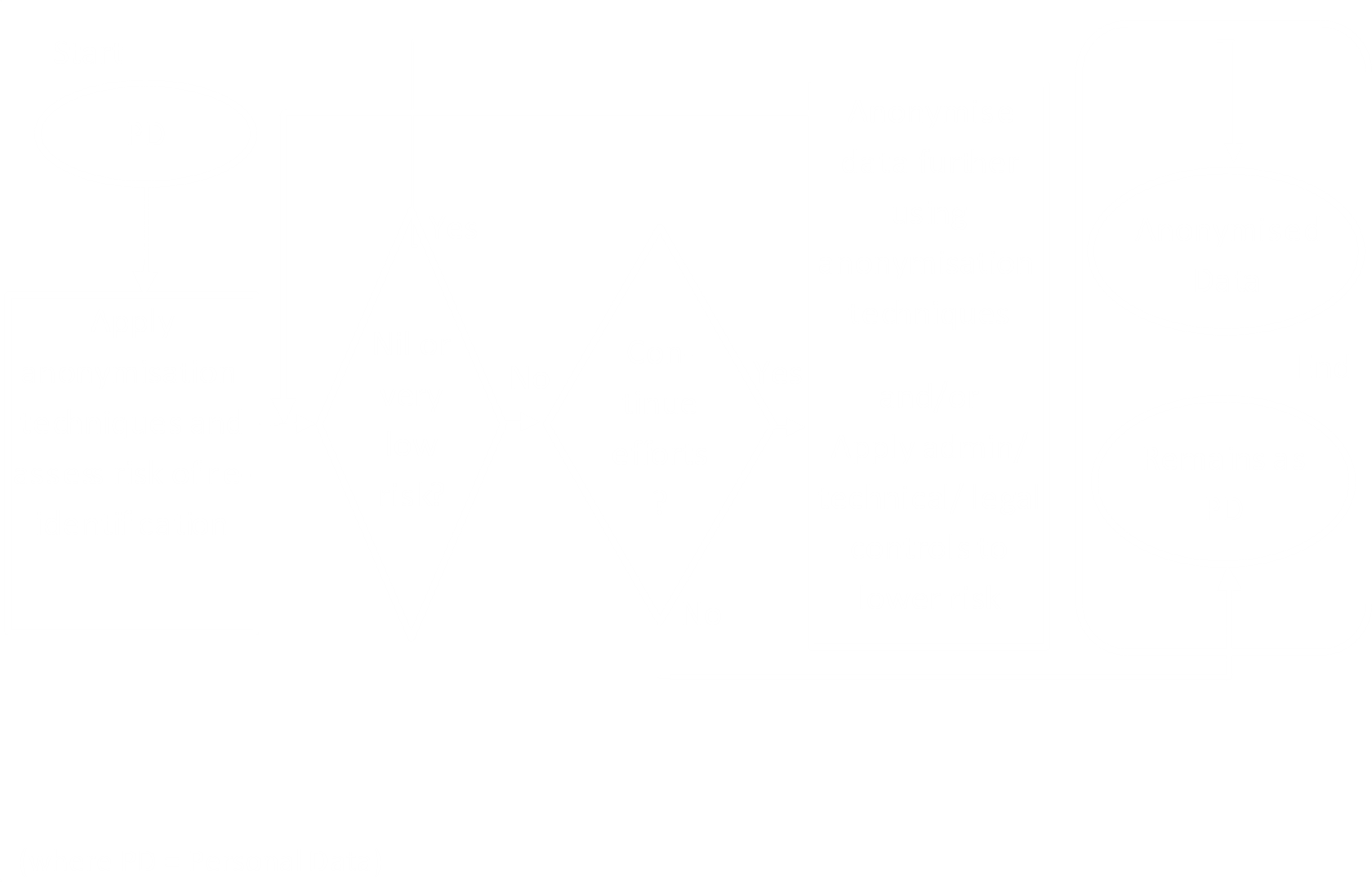
[Attacks Against K-Anonymity 15](#_Toc63379936)

The following section is based on the Guide to Basic Data Anonymization Techniques, published by the Personal Data Protection Commission in Singapore. The complete document can be studied for further details. It is available in the resources section.

Data anonymization is the process of converting personal data into anonymized data so that individuals can no longer be identified using the data. This is done through a range of techniques.

Ideally, data anonymization is irreversible. However, there may be situations in which the organization performing the anonymization is also able to recreate the original data from the anonymized data, thus making the process reversible. In those situations, of course, the organization has to be able to explain the purpose behind the process being reversible.

## Anonymization Process



The diagram above shows the complete anonymization process. Essentially, personal data is taken and some anonymization techniques are applied. Afterwards, the risk of re-identification is assessed. If the risk is low or non-existent, the process is complete. Otherwise, more efforts need to be made. Here, it is possible to apply one’s expert opinion or legal controls to lower the risk.

## Anonymization Techniques

The different data anonymization techniques we will be looking at include:

* Attribute Suppression
* Record Suppression
* Character Masking
* Pseudonymization
* Generalization
* Swapping (Shuffling and Permutation)
* Data Perturbation
* Data Aggregation
* Synthetic Data Generation

Sometimes, multiple techniques could be used one after another if it is determined that the data is not anonymized enough to prevent re-identification.

### Attribute Suppression

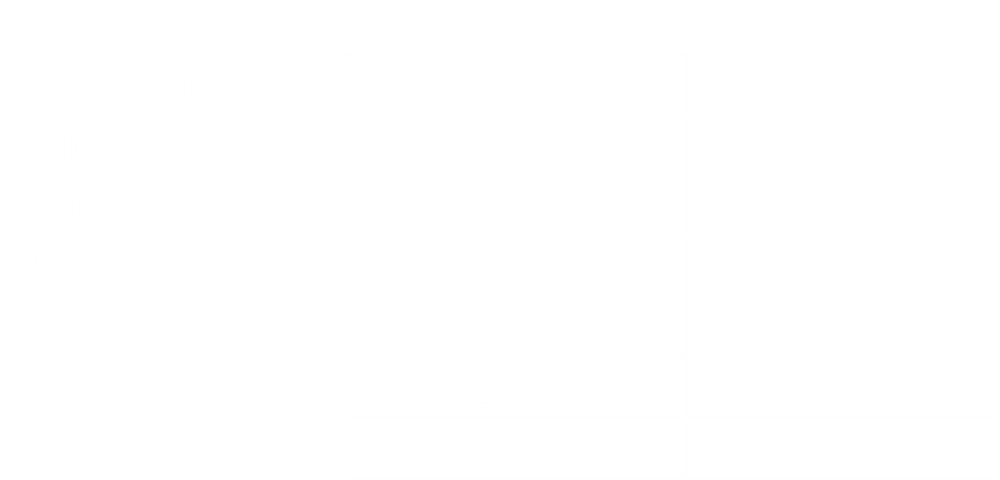
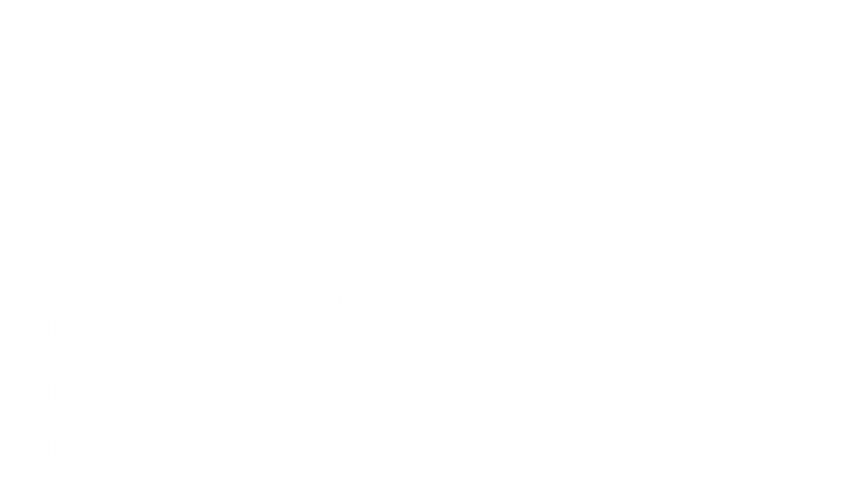
Attribute suppression works by removing a part of the data. In a database or spreadsheet, this would be like removing a column. Remember that this is complete removal not just hiding the attribute. As such, attribute suppression is a form of irreversible data anonymization.

Attribute suppression is used when the attribute being removed is not required in the anonymized dataset or when an attribute that can be used to re-identify individuals cannot be suitably anonymized using any of the other techniques. The more unnecessary data we keep around, the easier it is to use the data to identify individuals.

It is best to apply attribute suppression at the beginning of the anonymization process, since it is easier to decrease identifiability at this point.

One limitation of this approach is that the quality of the data is affected to some extent, since we are getting rid of part of the data.

Consider the example below. Our objective here is to analyse the test scores of students under each trainer. As such, we do not need to know which score belongs to which student. Thus, we can remove that attribute.

### Record Suppression

In record suppression, an entire record is removed from the database, i.e. an entire row of data. Thus, record suppression is an irreversible data anonymization technique.

Record suppression is used to remove outlier records or records that do not meet other criteria, such as -anonimity. We shall look into -anonimity in detail later, but essentially, the criterion demands that any given data point be similar to other data points so that that particular data point cannot be used to identify the individual. Essentially, we need to get rid of any record that is uniquely identifiable based on the data itself, i.e. outliers.

Outliers specifically are dangerous to keep around since they are unique and can be used to easily identify individuals. For example, if there is one exceptionally bright student in a class, even if we have nothing other than just the grades of all the students, we will still be able to identify the bright student based on just their grades. Thus, we need to use record suppression here.

Record suppression can be used before or after other data anonymization techniques have been applied. For example, we could remove the record for the exceptionally bright student right at the beginning, or we could perform generalization first and find that there is only one record in the highest graded sub-group, and then remove that record for being an outlier.

Record suppression is also used for pre-processing purposes. For example, if a particular record has a lot of null values, it will not be useful to our analysis. Thus, that record can be suppressed.

### Character Masking

Character masking is the process of hiding a part of a specific data value, such as by using ‘X’s in place of a few digits of a bank account number. Character masking is typically done partially, i.e. to just a few of the characters in the attribute. Since we are replacing characters, character masking is irreversible.

Character masking can be useful in the sense that no one other than the person the message is intended for can identify the value. It is useful when hiding just a part of the attribute is sufficient to provide the extent of anonymity required by the situation.

For example, when account statements are supplied by banks, account numbers are anonymized using character masking. In this way, only the intended recipient will be able to understand the complete account number, since it is their own account number.

Character masking is also used to reduce data granularity. For example, if we have the postal codes of our users, we can identify exactly which area of a city they live in. If we decide to mask the last few digits of the postal codes, we will be unable to tell which part of the city they live in, but we will still be able to identify which city they live in based on the first few digits.

Obviously, character masking can only be used for specific string values, not for general data.

### Pseudonymization

In pseudonymization, data that can be used to identify individuals is replaced with made up values. It is also referred to as coding. For example, we could replace the names of the people in the records with numbers.

Pseudonymization could be irreversible, such that original values are disposed of and the pseudonymization cannot be replicated, or reversible (only by the organization that originally had the data), where the original values are kept securely and can be retrieved and linked to the pseudonymized data using a mapping table if required. For example, say we want to perform some analysis and then inform the participants about how their performance was in comparison to the overall performance. In such a case, if we used pseudonymization, the process would need to be reversible.

Pseudonymization is used when there is a need to uniquely distinguish the data, since we want to perform some analysis based on each record separately, and no other information that can be used to imply the original information is being kept.

To increase security, two levels of pseudonymization can also be used.

### Generalization

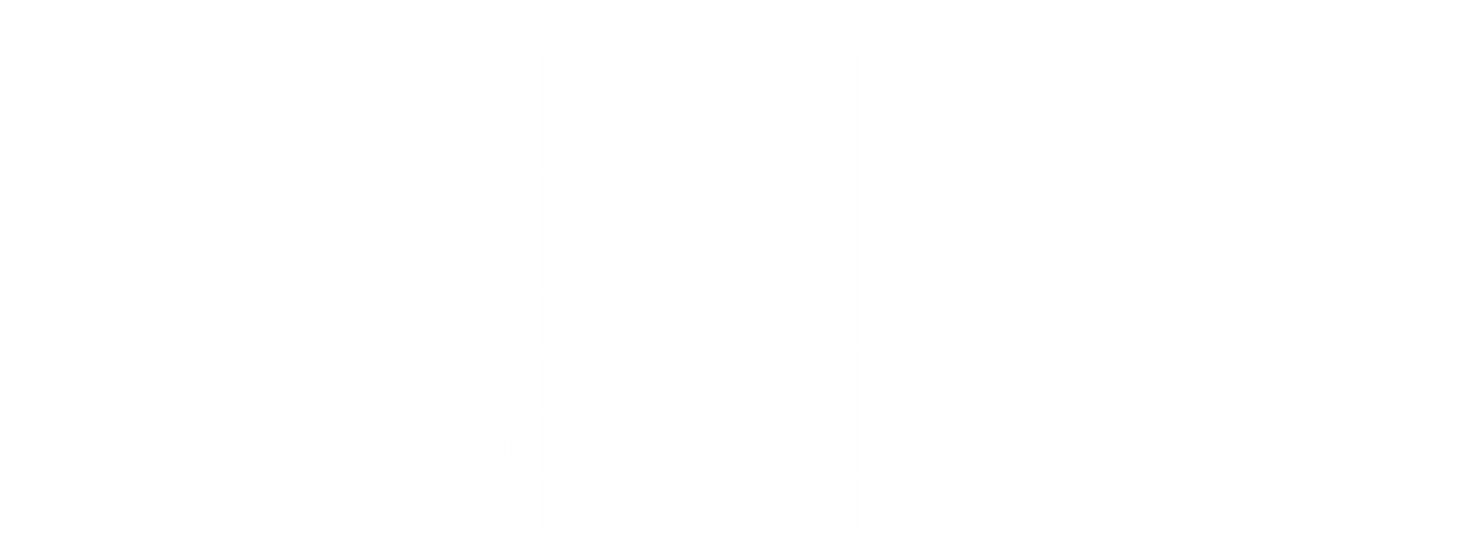
Generalization is the process of deliberately reducing the precision of data. For example, instead of keeping the specific age of an individual, we could store a range in which their age falls, or instead of recording their exact address, we keep the general area in which they live. Generalization is also known as recoding. It is an irreversible technique.

After generalization, if we find there are outliers, meaning records that are sitting alone in a particular range, we could consider record suppression to get rid of these outliers.

### Swapping

In swapping, the data in the dataset is rearranged such that the individual attribute values are all still present, but they do not correspond to the original records they were a part of. Swapping is also called shuffling and permutation. The original dataset is irreversibly changed.

Swapping can be used when analysis will be done at the intra-attribute level only. Consider the example below:

If we want to analyse say, how many of the individuals are from each job field, we can easily do so even after swapping. We would be working with just one attribute, so shuffling it around makes no difference to our results. However, if we wanted to analyse the relationships between different attributes, swapping would prevent us from doing this. For example, if we want to know the average number of visits per month for Lawyers, we cannot do so since the values for average visits per month have been shuffled and no longer correspond to the records that have Lawyer as the job title.

### Data Perturbation

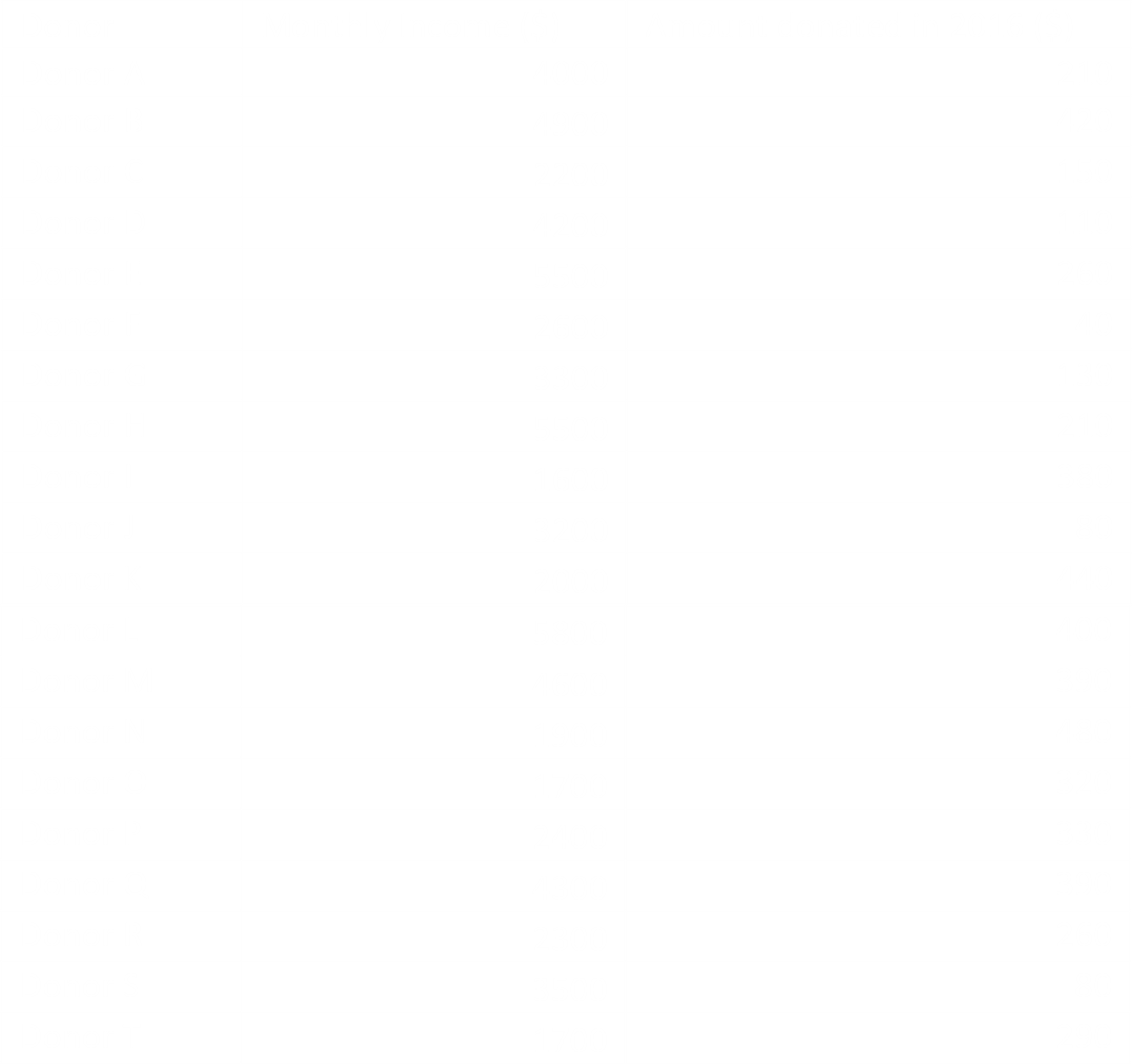
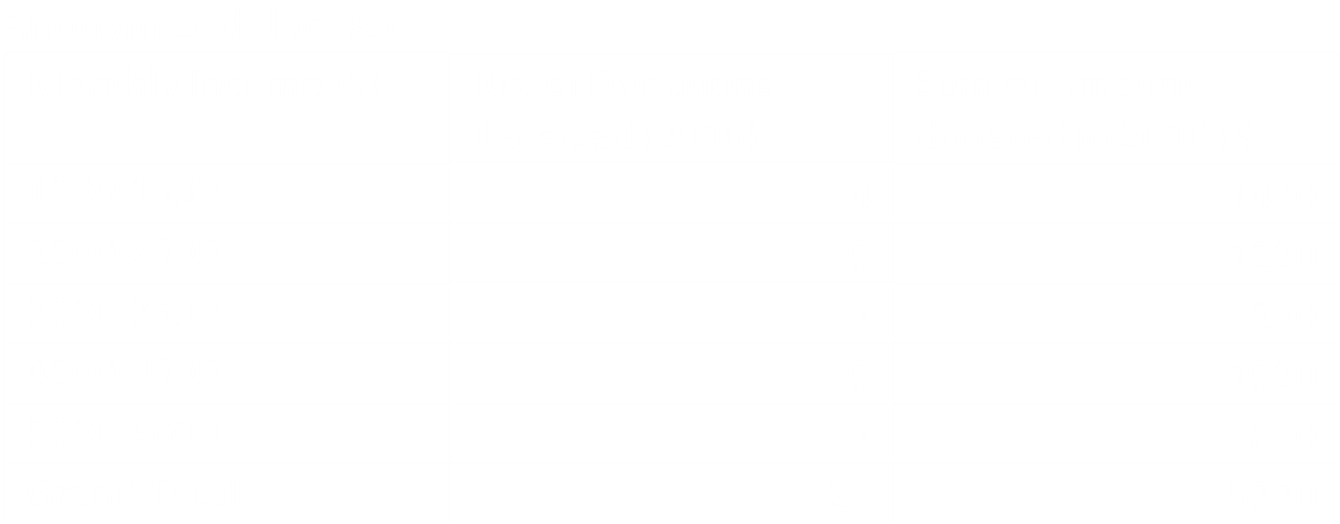
In data perturbation, the original dataset is modified to be slightly different. This can be achieved by rounding data or by adding noise (adding or subtracting a small value). For example, if someone’s age is 28, we could round this to 30 or add noise and make it 27. Since the data is permanently changed, this technique is irreversible.

Data perturbation should be used for quasi-identifiers, which are typically numbers and dates that can be combined with other data to identify individuals. By changing these values ever so slightly, we can ensure that they cannot be used to identify individuals indirectly.

Data perturbation should only be used when the slight differences are acceptable and not when the accuracy of the data is crucial. For example, if we wanted to find the average value of some attribute after applying data perturbation, the result we would get would be an approximate result, not the actual one. This reduction of accuracy is one of the drawbacks of this technique.

### Data Aggregation

Data aggregation is simply summarizing the values. For example, instead of storing all the individual ages available in all the records, we could store the average age. Other aggregation functions including sum, min, max, count, median, etc.

Obviously, data aggregation should only be used when individual data is not required and the aggregated data is enough.

### Synthetic Data Generation

Synthetic data generation is not really a data anonymization technique. It is different from the other methods in that it does not even deal with ‘real’ data. It is used to generate synthetic data based on the original data instead of modifying the original data.

Synthetic data generation is typically used when a large amount of data is required to test a system, but very little is actually available. The actual data is not used, but the synthetically generated data is still realistic.

## K-Anonymity Model

The K-Anonymity model aims to include the different data anonymization techniques we have seen so far in such a way so as to make re-identification impossible. The goal is to allow data holders to release a version of the private data that they have collected with the scientific guarantee that the subjects of the data cannot be re-identified, while still leaving the data practically useful.

### Description

We have previously discussed quasi-identifiers, which are pieces of data that can be linked with external sources to identify the data subject. For example, say we have some anonymized medical data that does not reveal the names or addresses of the patients. However, the ZIP codes and birthdates are visible in this medical data. Additionally, say there is a voter list that does have people’s names and addresses on it and also lists their ZIP codes and birthdates. An attacker could use the common information in the two lists, i.e., the ZIP codes and birthdates, to identify exactly which people have particular diseases. In this case, the ZIP codes and birthdates acted as quasi-identifiers.

The K-Anonymity model aims to ensure that in the final released data set, for every tuple there are at least tuples (including the current tuple itself) such that the data for the quasi-identifiers is the same. For example, if , for the example above, there would be at least one other record that had the same ZIP code and birthdate. Since there are then records with the same quasi-identifiers, attackers will be unable to differentiate which one is the one they are looking for. Obviously, the greater the value of , the better anonymity is achieved.

Example

For a release table () of size , with attributes let be the quasi-identifiers associated with the . The will be said to satisfy -anonimity if and only if each sequence of values (the rows basically) for appears with at least occurrences in the .

Consider the table below:



Here, the attributes for Race, Birth, Gender and ZIP are quasi-identifiers. This is because these are common attributes that will most likely be available in other data sources as well. Problem is not a quasi-identifier since this attribute is not a common attribute.

and , and , and , , and and and respectively have the exact same values in the attributes. As such, the value of is here.

This was achieved by using different anonymization techniques. Character masking was applied to the ZIP codes, birthdates were generalized to just the year and attributes like names and NIDs were suppressed. The combination of such things has led to a far greater level of anonymization than the anonymization techniques could achieve individually.

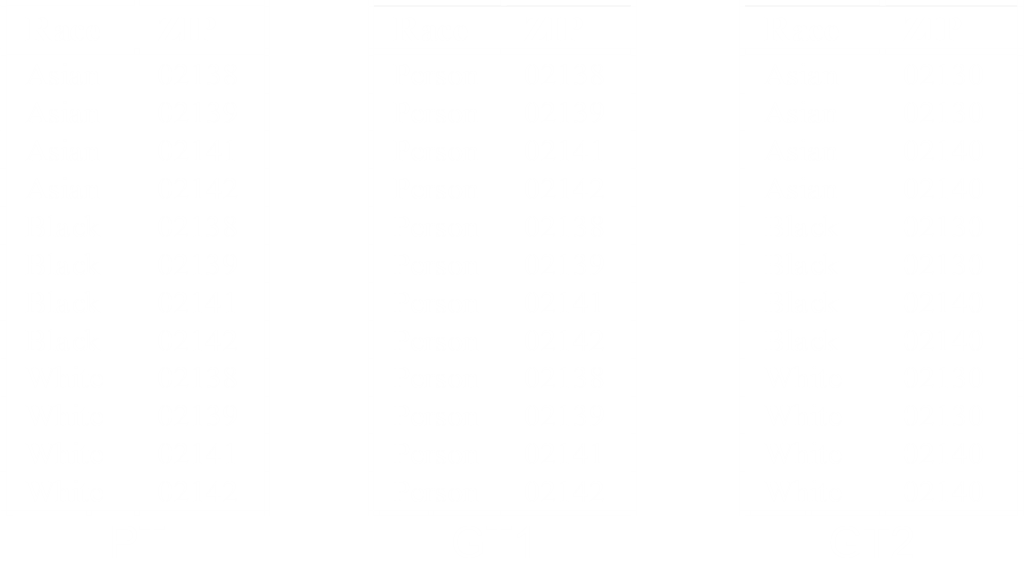
Note that K-Anonymity will only hold for a finite set of data. If we add more tuples to our original data set (the private table ), the same anonymization process we have used might not be enough to ensure that K-Anonymity holds. We would need to go over the process again.

### Attacks Against K-Anonymity

There are a few different ways attackers could attempt to bypass K-Anonymity.

#### Unsorted Matching

Say we have a where the is Race and ZIP. We release the data two different times, once by generalizing the Race of the tuples but leaving the ZIP codes intact, and once by applying character masking to the last digit of the ZIP codes but leaving the Race attribute intact.

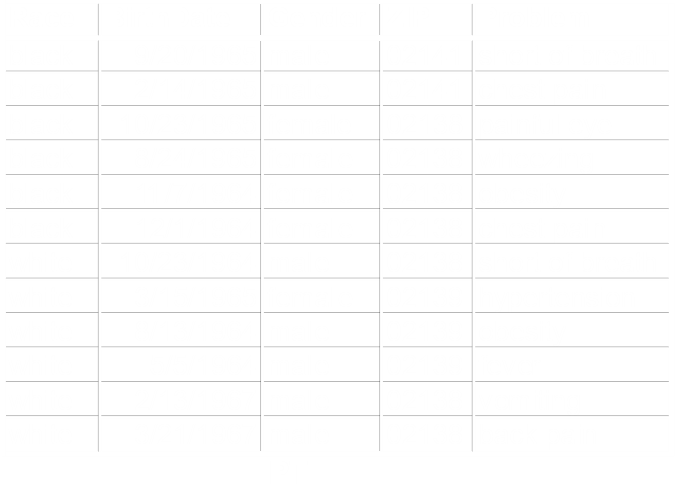


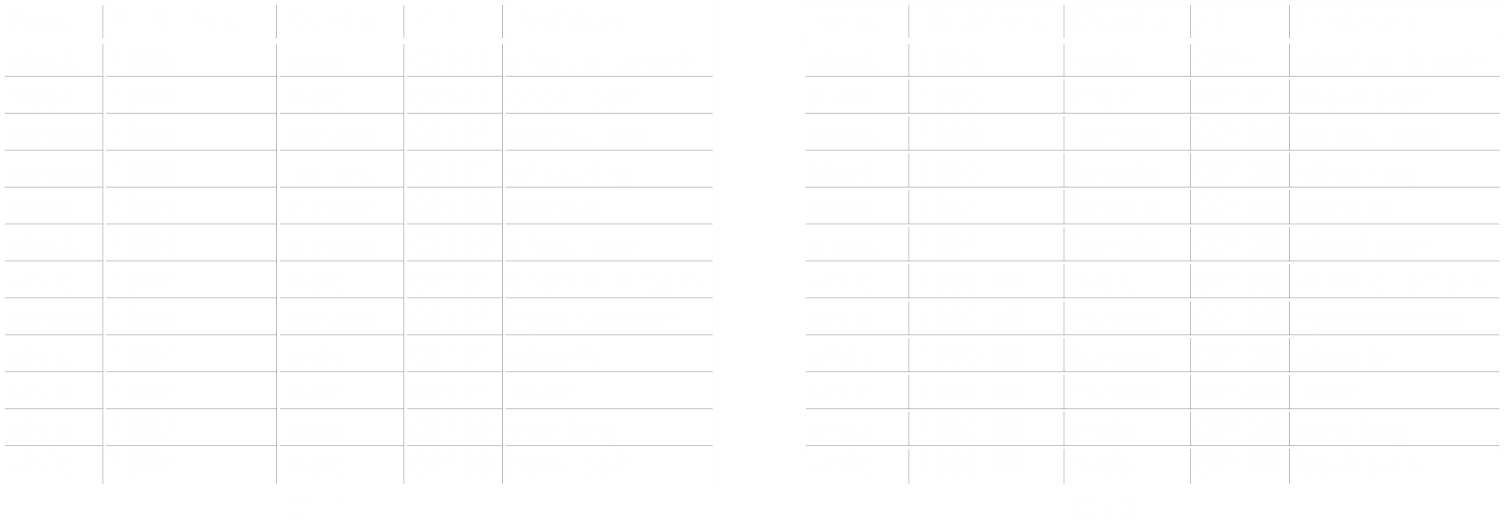
An attacker could now try to match the tuples in the released tables one by one in the order in which they appear. For example, the first tuple from each released table would be matched to reveal both the Race and the ZIP code of that data subject.

An unsorted matching attack is a very naïve attack, so the way to stay safe from one is also simple. We can simply randomly sort the tuples of the released tables. If we sort the first table by ZIP code and the next one by Race, then attackers will no longer be able to use this form of attack, or at least, the data they would retrieve by this method would be wrong.

#### Complementary Release

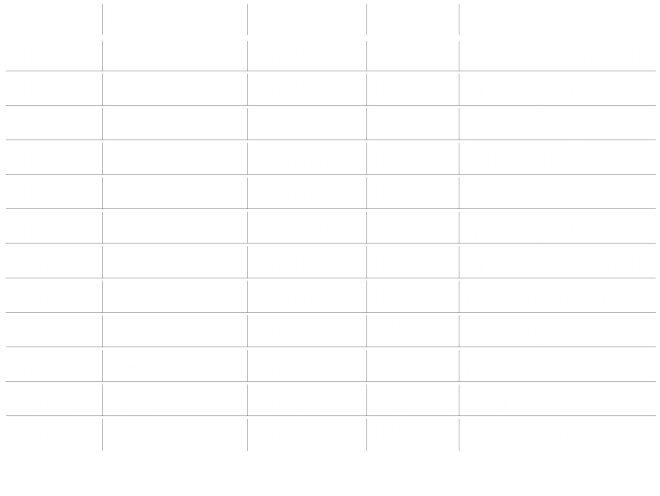
A complementary release attack works by linking different releases of the same .





The has four s, Race, Birthdate, Gender and ZIP. Say we make two releases of the same , perhaps to perform different types of analysis. In the first release, Race and ZIP were anonymized to achieve K-Anonymity. In the second release, Birthdate and Gender were anonymized to achieve K-Anonymity.

An attacker would link both tables using the Problem attribute. This would reveal the linked table.



The linked table has all of the data and does not have K-Anonymity.

The reason the attacker was able to do this is because we used anonymization on different attributes in the different releases. In the first release, we left the Birthdate and Gender attributes unanonymized and in the second release we left the Race and ZIP attributes unanonymized. Thus, the two tables could be combined to reveal all the data.

Because of this, we should always make sure that all quasi-identifiers are anonymized, or at least, that any quasi-identifiers that were anonymized in one release are also anonymized in any future releases.

#### Temporal Attacks

Temporal attacks are similar to complementary release attacks, except that instead of relying on unanonymized attributes from different releases of the same version of the to reveal sensitive information, it relies on unanonymized attributes of different releases from different versions of the .

Say at time we have a , and we release . By time , more tuples have been added to the , so it has become . Now we release . might not respect the anonymization techniques used by , which will allow attackers to again use linking to retrieve supposedly anonymized data.

The solution to temporal attacks is the same, in that we should either anonymize all quasi-identifiers, or we should at least anonymize any quasi-identifiers that were anonymized in any previous release.