Imperial College
London

COMPUTATIONAL
PRIVACY
GROUP

Privacy Engineering

Week 1 - Data pseudonymization and anonymization

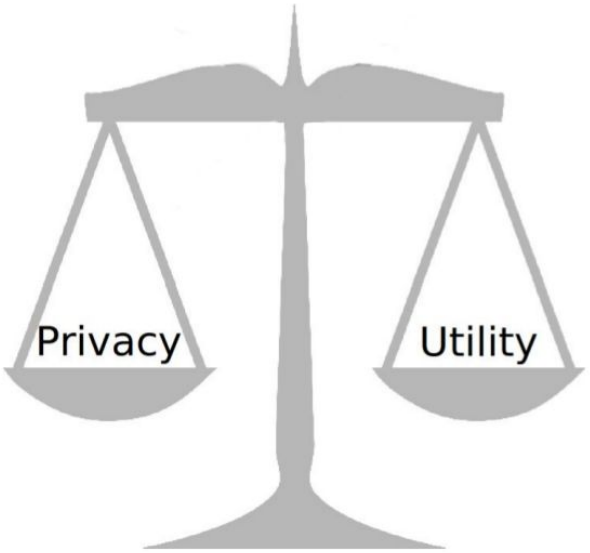
Anonymization: Balancing the use of data in aggregate with protecting people's privacy

Anonymization: disassociating an individual's record from their identity in a particular dataset.

The rationale is that if the data cannot be associated with the individual to whom it relates, it cannot harm that person

This is deeply embedded in data protection laws around the world: the upcoming EU-GDPR and UK-Data Protection Bill, HIPAA (Health data in the US), FERPA (education data in the US)

In short: if the data is anonymous it is not personal data anymore => no consent, purpose, etc



Pseudonymization: The removal of direct identifiers

We remove all **direct identifiers**, e.g. name, phone number, address, social security number etc, and just give each person a unique id that cannot be linked to them.

We could build a **correspondence table** by generating random strings but:

- It quickly becomes cumbersome to save it and keep it up-to-date as new data arrives
- We need to make sure the strings are unique (collisions)

Phone number +44 (0) 77 065	**Pseudonymised ID**
19475	tWILOS3
19476	sK8pERT
19477	AcG7l2j
19478	kdmMsz6
19479	uXxN5OM
...	

We thus define a function to map identifiers (e.g. a phone number) to pseudonymised ID

Using a secret formula

To avoid having to remember and update a correspondence table, one might decide to use the following formula

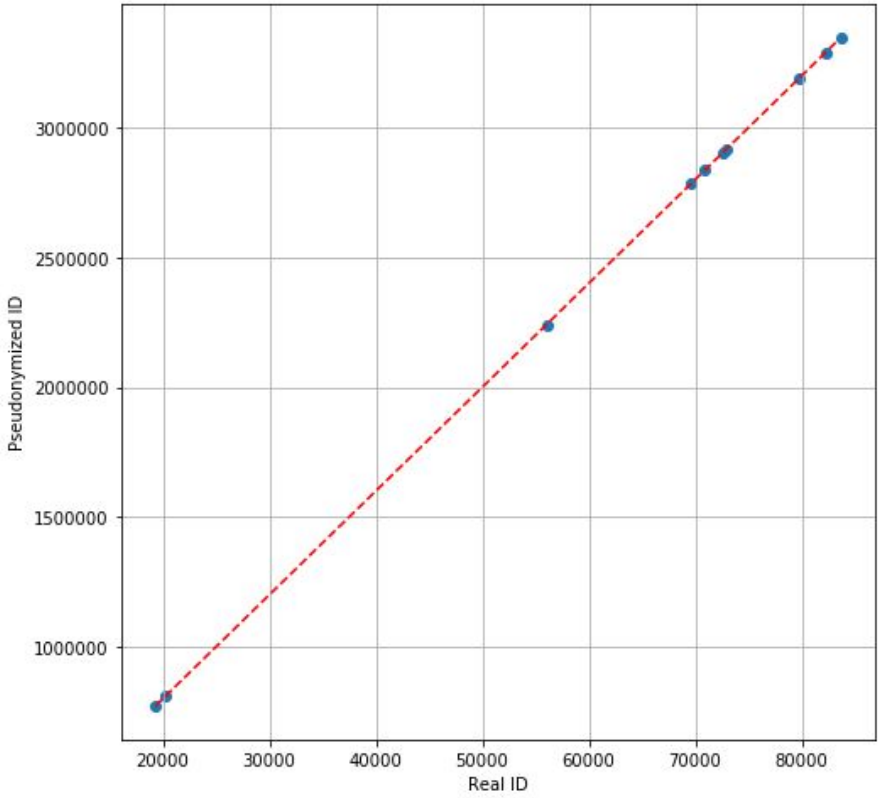
*pseudonymized_ID = phone_number * 17 + 12345*

If we keep the formula secret is this safe? Any idea how to attack data pseudonymised that way?

Phone number +44 (0) 77 065	**Pseudonymised ID**
19475	45410
19476	45427
19477	45444
19478	45461
19479	45478
...	

Now, what if I know the real phone numbers of a couple of people in the dataset?

Phone number +44 (0) 77 065	**Pseudonymised ID**
?	45410
19476	45427
?	45444
19478	45461
19479	45478
...	



Trivial to guess that it's a line and fit a model ($y = \alpha x + \beta$) with $\alpha = 17$ and $\beta = 12345$.
Now, we know that 45444 => 19477

But… nobody would do that

“A digital ticket system e-talons was rolled out in [the] municipal public transport system of Riga, Latvia in 2009. [...] On 25.03.2015 Riga municipality came public with an open data initiative and published all e-talons data on 2 randomly selected months (February 2015, January 2016). Published data is[was] anonymized by replacing real ticketID with another ID using a **proprietary algorithm**.” The database contains a total of **16,423,439** e-talons.

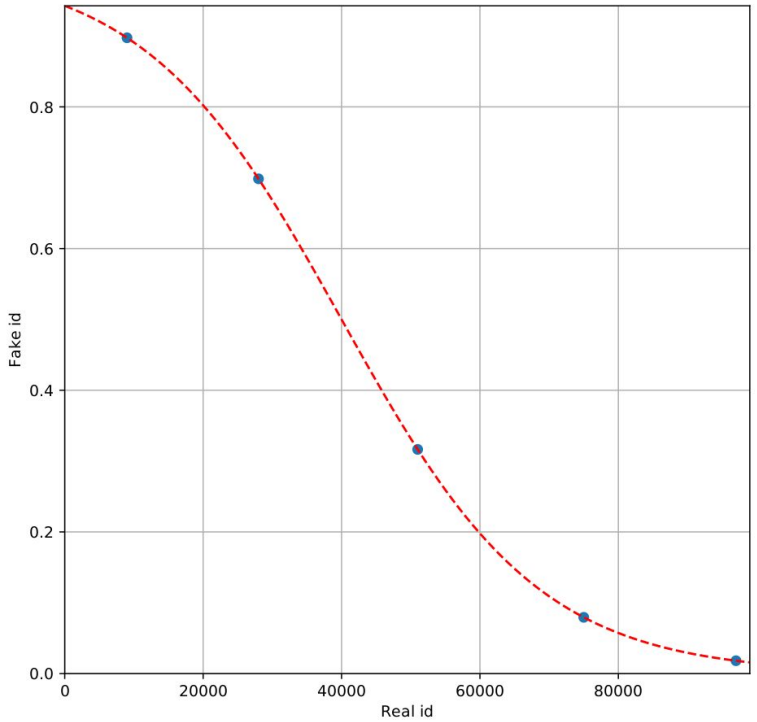
“We used only 2 pairs of ticketID and publishedTicketID that are known to be correctly matched” and...

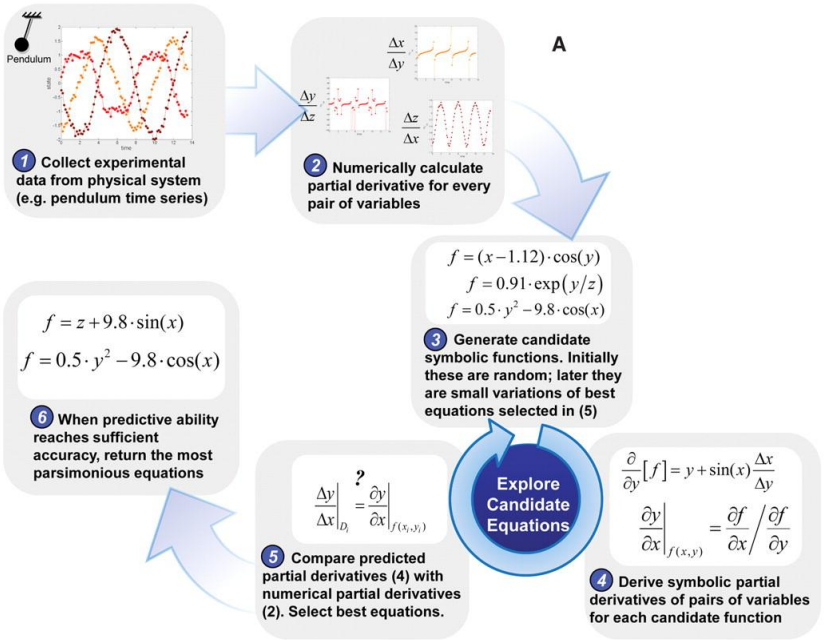
*publishedTicketID = ticketID * 3 - 3072913*

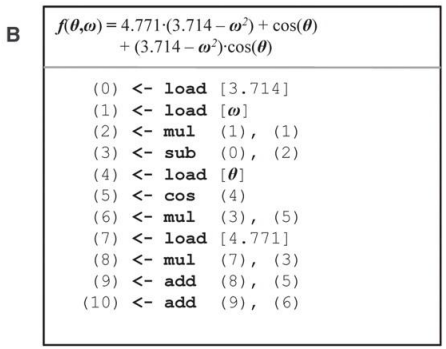
Lavrenovs, A. and Podins, K., 2016, November. Privacy violations in Riga open data public transport system. In *Advances in Information, Electronic and Electrical Engineering (AIEEE), 2016 IEEE 4th Workshop on* (pp. 1-6). IEEE.

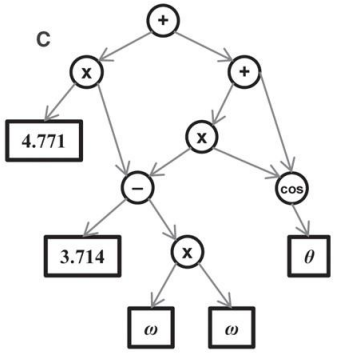
Making the function “more complex” is not going to work either

e.g. using *pseudonymized_ID = 1/(1+exp(0.7*(Phone_number/10000 - 4))*







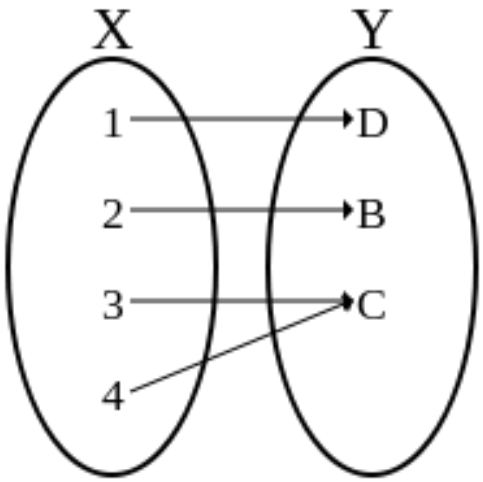


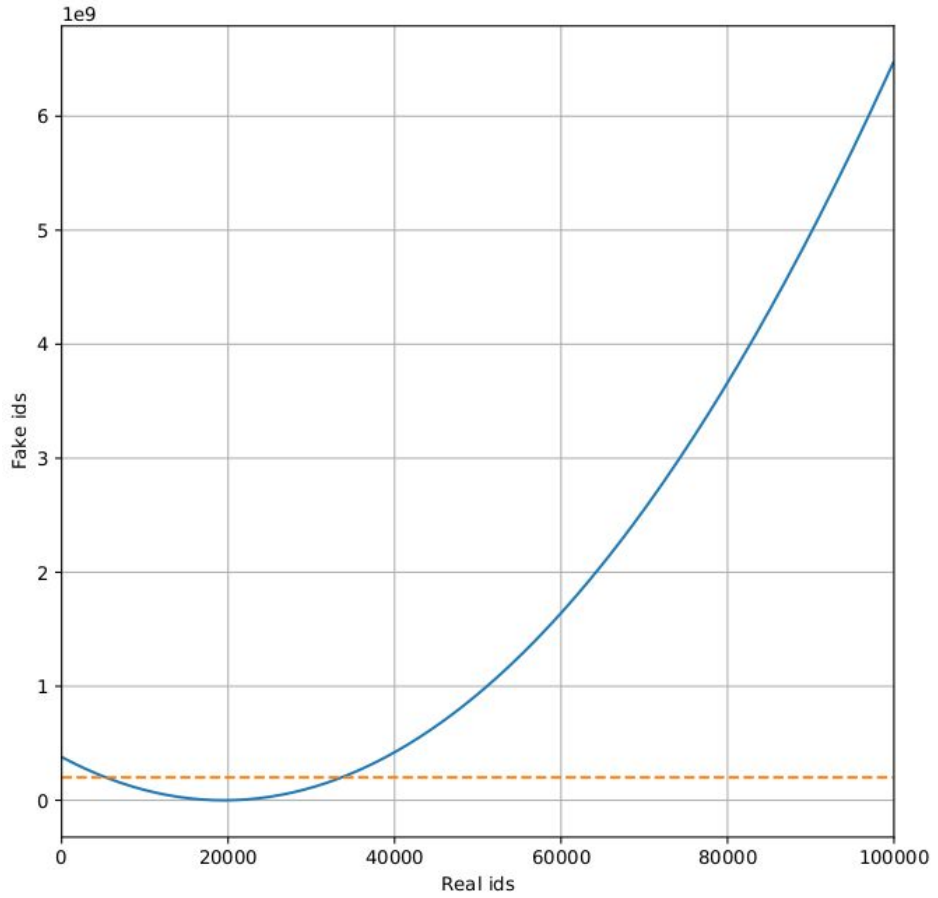
Schmidt, M. and Lipson, H., 2009. Distilling free-form natural laws from experimental data. *science*, *324*(5923), pp.81-85.

And collisions are an issue

Def: A collision occurs when two inputs have the same output value, i.e. if two different IDs are mapped to the same pseudonymised ID.

Using: $y = (x - 19477)^2 + 12$



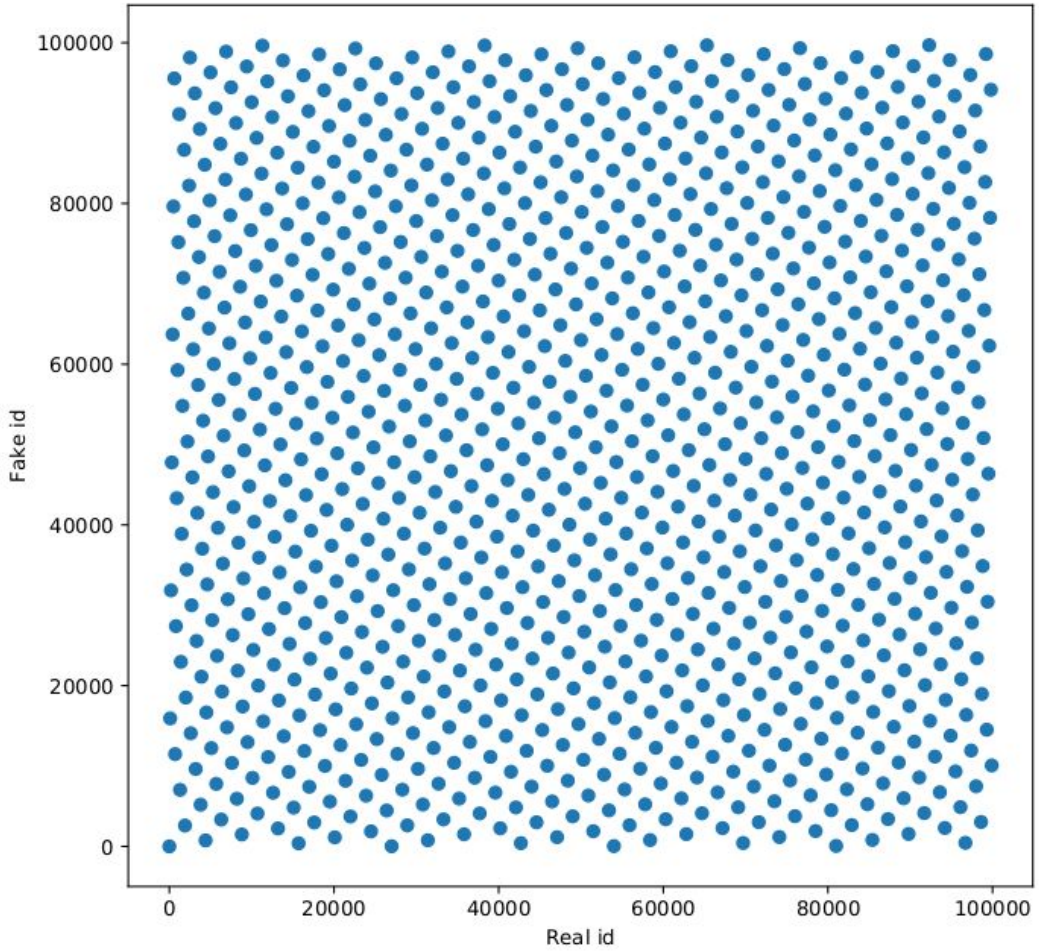


Let’s sum up our requirements

1. We want a **deterministic function** (same inputs are always mapped to same outputs).
2. We want the possible outputs to all have the same length (for simplicity)
3. We want **collisions to be very unlikely** (impossible is too much to ask).
4. We want a function that is fast to compute (i.e. for any input it’s easy to compute its output, for simplicity).
5. We want a function that is **computationally infeasible to invert** (i.e. for any output it’s hard to compute its input). A one-way function.

These are what a “cryptographic hash function” does (when we say “hash function” in this class we mean cryptographic hash function).

Example of the output of a hash (but not one-way) function



Designed so a “small change in input value generates a large jump in output value”.

For instance it might look like:

$h(k) = \lfloor m * (ka - \lfloor k*a \rfloor) \rfloor$

(called multiplicative hashing) with *k* the input and *a* and *m* constant.

With m = 27813 and a = 3.1415926 this would give us:
19475 => 14348
19476 => 18286

Some actual cryptographic hash functions

- MD5 → 128-bit hash = 32-character string output (hexadecimal)
- SHA (Secure Hashing Algorithm)
  - SHA-1 → 160-bit hash = 40-character string output
  - SHA-224 → 224-bit hash = 56-character string output
  - SHA-384 → 384-bit hash = 96-character string output
  - SHA-512 → 512-bit hash = 128-character string output
- SHA-3 (Secure Hashing Algorithm 3)
  - SHA3-224 → 224-bit hash = 56-character string output
  - SHA3-256 → 256-bit hash = 64-character string output
  - SHA3-384 → 384-bit hash = 96-character string output
  - SHA3-512 → 512-bit hash = 128-character string output

If you know the length of the hash, you can reduce the set of possible hash functions that could have been used

For example, “1945” with:

- MD5: *2d00f43f07911355d4151f13925ff292*
- SHA-1: *f2779feb3682526b35eb6a642e38b67d68c654e4*
- SHA-224: *91bdeaa640cb2983aa7e746a7c1d777cd174406bd840b5bfe77ef482*

Note: vulnerabilities have been discovered both in MD5 and SHA-1 algorithms (collisions), so their adoption is falling. No computationally feasible attacks are known for SHA-224, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512.

End of the story, problem solved?

Phone number +44 (0) 77 065	**Pseudonymised ID**
19475	2d00f43f07911355d4151f13925ff292
19476	1f71e393b3809197ed66df836fe833e5
19477	de03beffeed9da5f3639a621bcab5dd4
19478	7ca57a9f85a19a6e4b9a248c1daca185
19479	36ac8e558ac7690b6f44e2cb5ef93322
...	

Even with a perfect hash function (one that is absolutely impossible to invert), **not necessarily**

If an attacker were to know:

- which hash function we used
  (there are only a handful of possibilities for a given output length)
- the size of our input space
  (e.g. that our original IDs are all 7-digits license plate)

she can easily build a **lookup table** by iterating over all possible IDs!
(a brute-force attack)

Lookup table

Using the hashlib library in Python:

```
def create_lookup_table(N):
  table = {}
  for id in range(10**N):
    h = hashlib.md5(str(id).encode()).hexdigest()
    table[h] = id

  return table
```

Hashed ID	**ID**
cfcd208495d565ef66e7dff9f98764da	00000
….	
1f71e393b3809197ed66df836fe833e5	19476
de03beffeed9da5f3639a621bcab5dd4	19477
7ca57a9f85a19a6e4b9a248c1daca185	19478
...	
d3eb9a9233e52948740d7eb8c3062d14	99999

It happened (again)

“Thanks to a **Freedom of Information** request, Chris Whong received and made public a complete dump of historical trip and fare logs from NYC taxis. It’s pretty incredible: there are over 20GB of uncompressed data comprising more than **173 million individual trips**. Each trip record includes the pickup and dropoff location and time, **anonymized medallion number**.”

NYC taxi licence numbers are 6 or 7-digits long which mean there are ~19M of them.

=> easy to build a lookup table

Medallion information are visible on the cab and can e.g. be found on pictures (Hi Jessica!)

https://tech.vijayp.ca/of-taxis-and-rainbows-f6bc289679a1
https://research.neustar.biz/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset

Making it actually impossible: salt it!

Cryptographic hash functions should not be “secret”: their definition is public and they can be computed by many open source tools. **Good computer security relies on mathematical guarantees and peer-review (is this function really hard to invert), not on “secrecy”**.

See e.g. https://www.schneier.com/essays/archives/2004/10/the_non-security_of.html

Lookup attacks are, however, only possible if you **know the input space** (e.g. license plates, phone number, social security numbers etc) and if this space is **small enough** for you to iterate over all possibilities in a reasonable amount of time.

Def: A salt is fixed string of arbitrary length (but long!) that is added to the identifier before hashing it. Note: salt must be kept secret!

md5(“19477”) => md5(“19477youwillneverguesswhatthissaltis!$”)

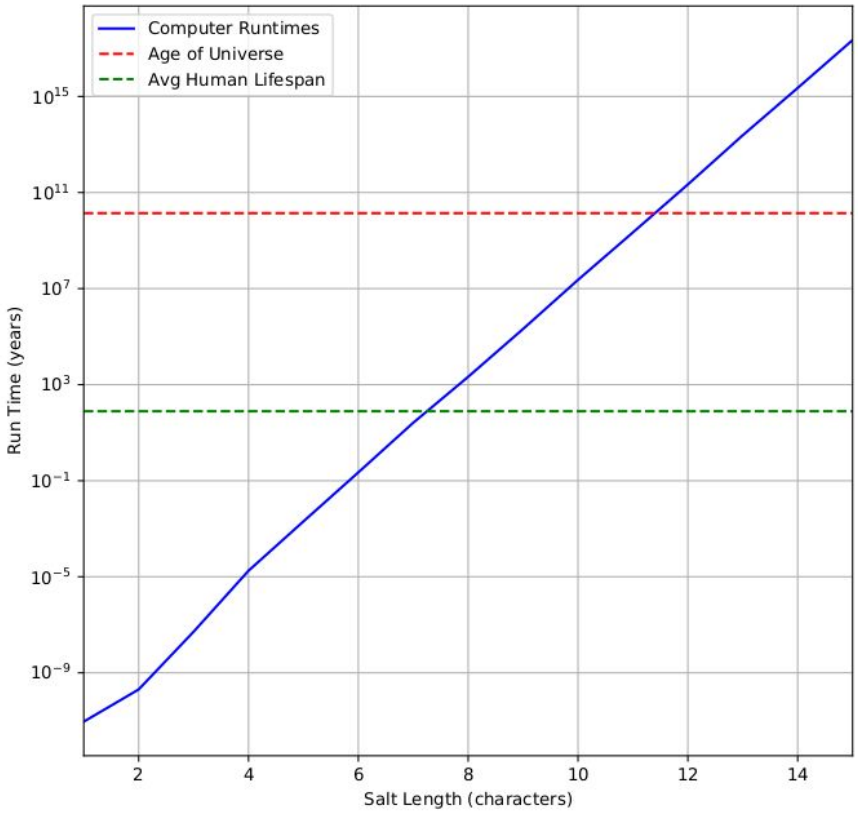
If the salt is long enough (and secret) it cannot be brute-forced

In the best case scenario, we know:

- Which hash function was used
- The length of the salt
- That the salt is appended to the end of the real ID
- A data point: (real ID, hashed ID)

We can then enumerate over all possible salts for that real ID until we find the one that matches.

But, **even in that case**, it'll take years to build a lookup table if the salt is long enough!



Q: why is it reasonable to assume the salt is secret but the hash function is not?

Any other case only makes things harder

In reality, the running time of a brute-force attack on salted hashes is often even higher.

An adversary might not know:

1. (likely) The salt length, so she needs to **try all strings of any possible length**.
2. (might not know) A real ID-hashed ID pair so she will have to try **all possible input IDs with all possible salts**.
3. (likely) The salt placement in the real ID string. She will have to to try all possible placements for any salt length.
   Note that we don't recommend to play with this (see link on secrecy) as it doesn't improve the complexity of building a lookup table more than a longer salt

Every unknown parameter (1)-(3) only increases the running time (by a multiplicative factor). A long enough (**and kept secret**) salt is virtually impossible to break

Now the table is properly pseudonymized, is it safe?

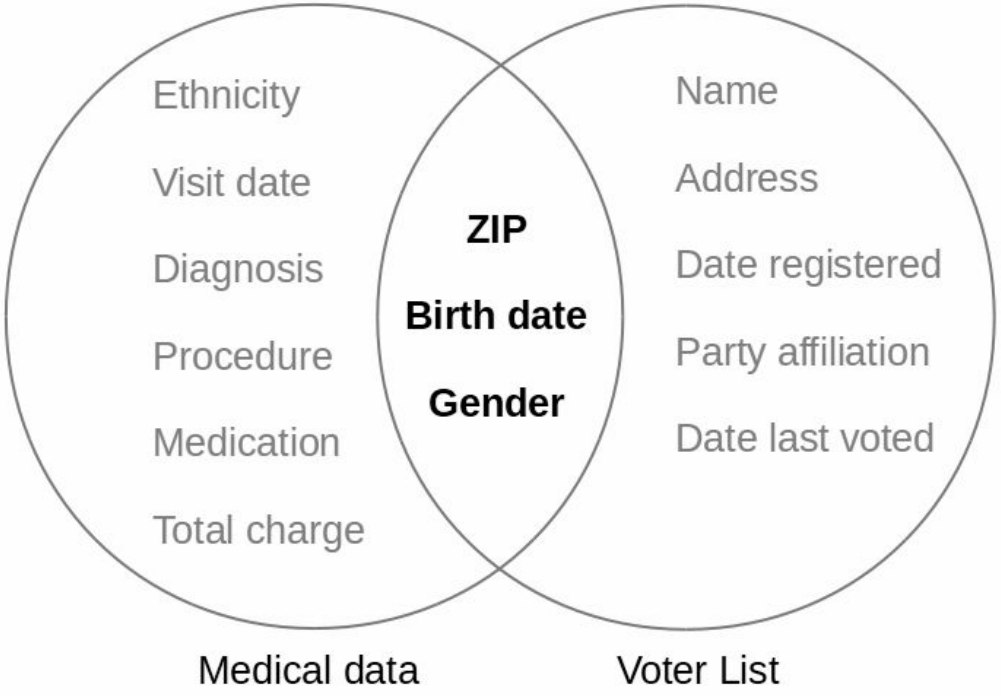
ID	gender	dob	zipcode	disease
800f48...	female	28-09-1955	43221	hypertension
f1f333...	female	06-01-1955	43210	hypertension
3b3c6...	male	08-07-1959	89127	prostate cancer
93db7...	male	10-06-1959	89127	gastritis
e1d54...	male	27-09-1959	89127	skin cancer

What if I were to know that the person I'm searching for is is a man born on July 8, 1959 living in Las Vegas?

But that will never happen in a big database

In 1997, William Weld, Governor of Massachusetts released properly pseudonymized medical records of state employees for research, assuming this protected privacy of individuals…

...a few days later, Gov. Weld's medical records were delivered to his office.



By Latanya Sweeney who was then a graduate student at MIT
(Voter list are public in the US)

A bit of terminology

Direct Identifier

Quasi-identifiers

Sensitive information

Name	Gender	Date of Birth	Zip-code	Employment Status	Education	Marital Status	Disease
Amelia Nixon	female	09-03-1944	60612	retired	bachelors	single	gastritis
1aeef...	female	02-07-1966	94105	employed	highschool	married	multiple sclerosis
cd178...	male	13-02-1945	89136	retired	bachelors	married	hyper-tension
40d10...	female	03-09-1939	10026	retired	bachelors	single	skin cancer
58c41...	female	16-10-1956	60605	unemployed	highschool	single	diabetes

Attributes

Terminology (more formally)

Auxiliary information: Information that could be known to an attacker

Identifier: A piece of information that directly identifies a person (name, address, phone number, ip address, passport number, etc)

Quasi-identifier: A piece of information that does not directly identify a person (e.g. nationality, date of birth). But multiple quasi-identifiers taken together could uniquely identify a person. A set of quasi-identifiers could be known to an attacker for a certain individual (auxiliary info).

Sensitive information: a piece of information about an individual (e.g. disease, drug use) we're trying to protect [but is relevant for the application].

Attacks and uniqueness

What kind of attacks are we trying to protect our data from?

In our model, the attacker assumes that a certain person is in the dataset and wants to find **with certainty** his/her sensitive attribute. If certainty is not possible, the attacker will try to find the sensitive attribute which is *very likely* to be correct (probabilistic attack).

In general, re-identification attacks leverage the *uniqueness* of individuals in the dataset, that is: **Uniqueness w.r.t. A:** fraction of the dataset that is uniquely identified by the set A of quasi-identifiers.

Following her successful re-identification of William Weld, Latanya Sweeney estimated that **87% of the US population is unique given a date of birth, zip code, and sex**. (Sweeney, 2000) [later re-evaluated to be 63% (Golle, 2006)]

Fighting uniqueness with k-anonymity

Def: A table is k-anonymous if every record in the table is indistinguishable from at least k-1 other records, with respect to every set of quasi-identifiers

This means that even if an attacker knows all possible quasi-identifiers, he cannot identify his target uniquely

id	gender	dob	zipcode	disease
800f48...	female	28-09-1955	43221	hypertension
f1f333...	female	06-01-1955	43210	hypertension
3b3c6...	male	08-07-1959	89127	prostate cancer
93db7...	male	10-06-1959	89127	gastritis
e1d54...	male	27-09-1959	89127	skin cancer

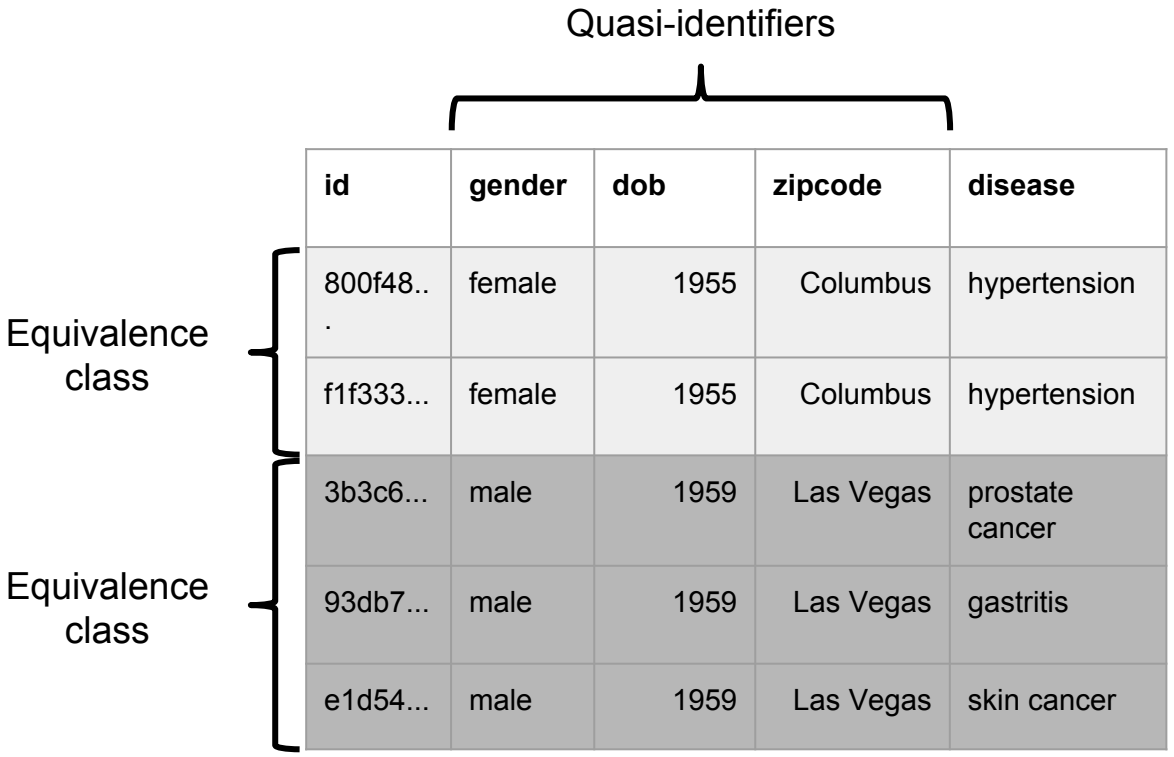
id	gender	dob	zipcode	disease
800f48...	female	1955	Columbus	hypertension
f1f333...	female	1955	Columbus	hypertension
3b3c6...	male	1959	Las Vegas	prostate cancer
93db7...	male	1959	Las Vegas	gastritis
e1d54...	male	1959	Las Vegas	skin cancer

(2-anonymous)

Equivalence class

Def: An equivalence class is a set of records that have the same values for all the quasi-identifiers

Q: What is the minimum and average size of the equivalence classes in a 5-anonymized dataset?



id	gender	dob	zipcode	disease
800f48..	female	1955	Columbus	hypertension
f1f333...	female	1955	Columbus	hypertension
3b3c6...	male	1959	Las Vegas	prostate cancer
93db7...	male	1959	Las Vegas	gastritis
e1d54...	male	1959	Las Vegas	skin cancer

Achieving k-anonymity

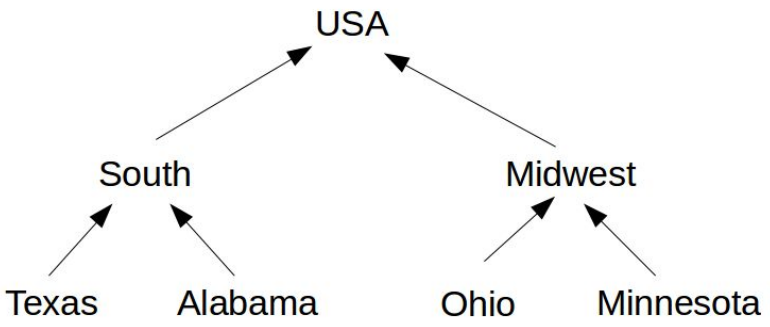
Non-perturbative methods (data truthfulness):

- Generalization: replacing attribute values with more general ones
  - 43221 → 4322** → 43***
  - 57 years old → 5* → >50
  - Tree-based generalization
- Suppression: deletion of a column (attribute) or a row (individual)
  - e.g. remove zipcodes or Clara Hicks from the dataset

Perturbative methods:

- Noise addition: gaussian noise
- Data swapping: swap attributes between individuals
- etc

Tree-based generalization



Here, we will focus on non-perturbative methods

So simple, everyone does this?

"A second data breach within the [AU] federal government in a week has seen a dataset involving **96,000 public servants** pulled from public view [...] Australian Public Service Commission had decided to pull the dataset from the government's open data portal data.gov.au [...] The APSC performs a massive yearly employee census to collect attitudinal data that tracks the views of **staffers about management and workplace conditions**. [...] But it decided to pull the dataset and review the information over concerns matching agency codes to individual responses would make it relatively easy to identify the public service worker who filled out the census. Public servants fill in the employee survey under the condition of anonymity"

Dataset has been downloaded 58 times before being removed.

Reference:
https://www.itnews.com.au/news/govt-pulls-dataset-that-jeopardised-96000-employees-438809

So simple, everyone does this? (2)

"We examined the Chicago Homicide dataset. [...] By combining this data with the Social Security Death Index, also available online, we were able to **successfully determine the identity of 35% of the individuals** [...] in the database. [...]

The dataset includes information on **23,000 victims** and **26,000 offenders**. [...]

The data set included a wealth of other fields that might prove **embarrassing or incriminating for victims and their families**. This included fields such as the relationship between the victim and the offender, the reason for the homicide, previous criminal histories of the victim and offender, as well cause and motivation for the homicide. In addition, the data set includes flags indicating whether the murder involved drugs, child abuse, gang violence, or domestic abuse."

Hard to anonymize properly esp. defining what auxiliary information might be available to an attacker and hence what should be considered a quasi-identifier.

Reference: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.15.7467&rep=rep1&type=pdf

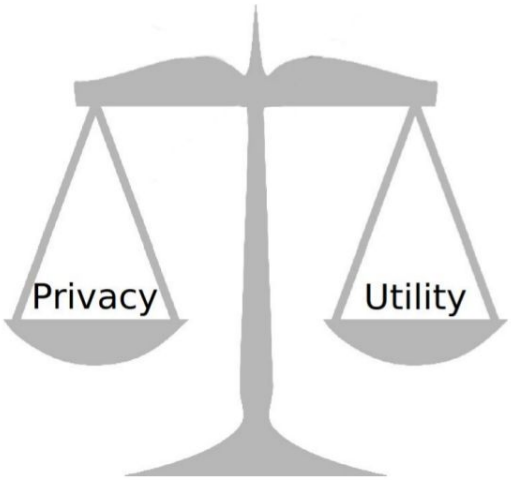
K-anonymizing data while preserving utility

Anonymizing the dataset intrinsically requires you to make the information less precise, complete, and/or accurate. This is an issue when people are then trying to use the data.

People use entropy to measure the amount of information left in the dataset and the **loss of information resulting from anonymizing the dataset**.

Def: Entropy $H(D) = -\sum_{i=1}^{k} \frac{\#C_i}{N} \log \frac{\#C_i}{N}$

Where: N is the amount of rows in the dataset D; C_1 ,..., C_k are the equivalence classes; $\#C_i$ indicates the number of rows that belong to C_i. The higher the entropy, the more information is contained in D.



The entropy $H(D)$ can range between 0 (→ everybody has the same quasi-id vector) and log N (→ nobody has the same quasi-id vector). If we have a dataset D_1 and its anonymized version D_2, we can compute the ratio $H(D_2)/H(D_1)$ to quantify the loss of information.

For instance using generalization

id	gender	age	zipcode	disease
b1871bb...	female	1953-01-12	10001	schizophrenia
2707c18...	female	1953-05-05	10001	Alzheimer's
77402e3...	female	1955-07-30	10001	Alzheimer's
ba0ef09...	female	1955-11-19	10001	skin cancer
4593bc6...	male	1954-05-11	43221	diabetes
7a55289...	male	1954-04-03	43221	skin cancer
c162182...	male	1959-01-28	43221	Alzheimer's
70d4a9c...	male	1959-02-15	43221	schizophrenia

Table 1: D_1

id	gender	age	zipcode	disease
b1871bb...	female	1953	10001	schizophrenia
2707c18...	female	1953	10001	Alzheimer's
77402e3...	female	1955	10001	Alzheimer's
ba0ef09...	female	1955	10001	skin cancer
4593bc6...	male	1954	43221	diabetes
7a55289...	male	1954	43221	skin cancer
c162182...	male	1959	43221	Alzheimer's
70d4a9c...	male	1959	43221	schizophrenia

Table 2: D_2
2-anonymous

$$H(D_1) = -\sum_{i=1}^{8} \frac{1}{8} \log \frac{1}{8} = \log 8 = 3$$

$$H(D_2) = -\sum_{i=1}^{4} \frac{2}{8} \log \frac{2}{8} = \log 4 = 2$$

But, at the end of the day, “utility” depends on the specific application

id	gender	age	zipcode	disease
b1871bb...	female	1950	10001	schizophrenia
2707c18...	female	1950	10001	Alzheimer's
77402e3...	female	1950	10001	Alzheimer's
ba0ef09...	female	1950	10001	skin cancer
4593bc6...	male	1960	43221	diabetes
7a55289...	male	1960	43221	skin cancer
c162182...	male	1940	43221	Alzheimer's
70d4a9c...	male	1960	43221	schizophrenia

Table 1: D_1

id	gender	age	zipcode	disease
b1871bb...	female	1950	10001	schizophrenia
2707c18...	female	1950	10001	Alzheimer's
77402e3...	female	1950	10001	Alzheimer's
ba0ef09...	female	1950	10001	skin cancer
4593bc6...	male	1940	43221	diabetes
7a55289...	male	1950	43221	skin cancer
70d4a9c...	male	1950	43221	schizophrenia

Table 2: D_2
3-anonymous

The table has been 3-anonymized using **record suppression**. But now a researcher who runs analyses only on the anonymized table might conclude that males cannot develop Alzheimer. This is because records to suppress cannot be chosen uniformly at random, hence suppression can introduce **bias** in the data.

But, at the end of the day, “utility” depends on the specific application

id	gender	age	zip-code	ancestry	disease
56b2...	*	1950-1960	NV	Denmark	prostate cancer
3f43...	*	1950-1960	NV	Denmark	gastritis
1e81...	*	1950-1960	NV	Denmark	breast cancer
3ef9...	*	1930-1940	CA	England	skin cancer
86f8...	*	1930-1940	CA	England	prostate cancer

Dataset 2-anonymized to study the relationship between ancestry and disease

id	gender	age	zipcode	ancestry	disease
e320...	F	1951	West	*	Alzheimer's disease
aa4c...	F	1951	West	*	skin cancer
582c...	F	1951	West	*	breast cancer
7c89...	M	1955	Northeast	*	prostate cancer
4f29...	M	1955	Northeast	*	diabetes

Dataset 2-anonymized to study the relationship between age/gender and disease

Can I just release two versions of the dataset, one for each application? Do you see any issues here?

Unfortunately not (if I obtain access to both datasets)

id	gender	age	zip-code	ancestry	disease
56b2...	*	1950-1960	NV	Denmark	prostate cancer
3f43...	*	1950-1960	NV	Denmark	gastritis
1e81...	*	1950-1960	NV	Denmark	breast cancer
3ef9...	*	1930-1940	CA	England	skin cancer
86f8...	*	1930-1940	CA	England	prostate cancer

id	gender	age	zipcode	ancestry	disease
e320...	F	1951	West	*	Alzheimer's disease
aa4c...	F	1951	West	*	skin cancer
582c...	F	1951	West	*	breast cancer
7c89...	M	1955	Northeast	*	prostate cancer
4f29...	M	1955	Northeast	*	diabetes

I can easily see that there is only one person in both datasets born between 1950 and 1960, living in the west part of the US (both quasi-identifiers), and who has breast cancer (sensitive information). Merging the datasets, this person is now a F, born in 1951, living in NV and of Danish ancestry. Not k-anonymous anymore

Ok, it's hard to define quasi-identifier and find a balance but, at least, if I do it I'm ok?

Any attacks you can think of on this (2-anonymized) dataset?

What if I know that the person I'm searching for is a woman, born in 1955, living in Columbus, OH?

→ she suffers from hypertension

id	gender	dob	zipcode	disease
800f48...	female	1955	Columbus	hypertension
f1f333...	female	1955	Columbus	hypertension
3b3c6...	male	1959	Las Vegas	prostate cancer
93db7...	male	1959	Las Vegas	gastritis
e1d54...	male	1959	Las Vegas	skin cancer

Def: A **homogeneity** attack can take place when individuals in the same equivalence class all have the same sensitive attribute value

Preventing Homogeneity attacks: ℓ-Diversity

Solution: ℓ-Diversity (Machanavajjhala, 2006)

An equivalence class is ℓ-diverse if it contains at least ℓ distinct values for the sensitive attributes. A table is ℓ-diverse if every equivalence class is ℓ-diverse

id	gender	dob	zipcode	disease
ae1cd7...	female	1954	Columbus	breast cancer
df0133...	female	1954	Columbus	skin cancer
3b3c6...	male	1959	Las Vegas	prostate cancer
93db7...	male	1959	Las Vegas	gastritis
e1d54...	male	1959	Las Vegas	skin cancer

2-anonymous and 2-diverse dataset

Ok, one release of the dataset that preserves enough utility for all current and future applications and that is both k-anonymous and ℓ-diverse **finally** enough?

Any attacks you can think of on this dataset?

Well, not quite yet...

Def: A **semantic** attack can take place when sensitive attributes of individuals in an equivalence class are distinct but semantically similar. Looking at this table, I learn that all women born in 1954 and living in Columbus have cancer.

Def: A **skewness "**attack" (here it is probabilistic) takes place when the distribution of the sensitive attributes in a class is skewed. In the general population, 99% might test negative for illegal drugs but, in an equivalence class, only 15% test negative. I learned something about people in this class

id	gender	dob	zipcode	disease
ae1cd7...	female	1954	Columbus	breast cancer
df0133...	female	1954	Columbus	skin cancer
3b3c6...	male	1959	Las Vegas	prostate cancer
93db7...	male	1959	Las Vegas	gastritis
e1d54...	male	1959	Las Vegas	skin cancer

2-anonymous and 2-diverse dataset

t-closeness (Li, 2007)

Def: An equivalence class is said to have **t-closeness** if the **distance** between the distribution of a sensitive attribute in this class and the distribution of this attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

There are numerous definition of distances between distributions. Li uses Earth mover's distance (EMD), which quantifies the "effort" needed to to "move" probability mass between points in one distribution in order to make it look the same as the other one.

But now, in a t-close dataset, the likelihood of suffering from cardiac diseases of young women has to be the same than the one of old men!

And even t-closeness is not always enough

Which disease do you think Isabella Lee who is born in 1941, living in the US, and from Finnish origin is suffering from?

Well, we know she's female (even if this has been removed from the dataset) so she can't suffer from prostate cancer...

id	gender	dob	zipcode	ancestry	disease
1efc27...	*	1953	USA	Germany	multiple sclerosis
ce0ae...	*	1953	USA	Germany	diabetes
58c80...	*	1941	USA	Finland	prostate cancer
a4116...	*	1941	USA	Finland	prostate cancer
17513...	*	1941	USA	Finland	Alzheimer's disease

Issues with data anonymization

You can only anonymize a dataset once incl. future data (even if more advanced and complex algorithms exist e.g. Wang, 2006)

It depends on the applications you're considering (very hard for open-data)

Once the dataset is out it's out… magnet:?xt=urn:btih:cd339bddeae7126bb3b...

You need to make strong assumptions on

- What is a quasi-identifier (In the heritage prize dataset anonymization, the blue and red teams' risk of re-identification estimates varied by an order of magnitude solely because of the hypothesis they made)
- What is ok and not ok for an attacker/researcher to learn (people who lived close to a river are more likely to suffer from Weil's disease (contaminated animal urine) and people in Scotland are more likely to report having used cocaine
- What could be inferred but is not available in the dataset (directly or at all)

Questions?