Privacy-Preserving Machine Learning

Raoul Grouls, 8 December 2025

Learning goals

- The students understands the defintions of privacy and identification
- The students can define and understands up- and downsides of the methods covered:
    1. Anonymisation
    2. ϵ-Differential Privacy
    3. Secure Multi-Party Computation
    4. Homomorphic encryption
    5. Federated Machine Learning

Definitions

Privacy-Preserving Machine Learning
What is privacy?

- Database privacy balances
    1. “private” functions we wish to hide and
    2. “information” functions whose values we wish to reveal.
- **Identification**: It is not possible to link someones identity to a database entry

Dinur, I., & Nissim, K. (2003, June). Revealing information while preserving privacy. In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 202-210).

Privacy-Preserving Machine Learning

What is privacy?

- Private inputs: $\mathscr{D} = \{d_1, \ldots, d_n\}$
- Some function f: $\mathscr{D} \rightarrow y$, where y depends on the problem at hand (eg it could be a number, category, etc)

Dinur, I., & Nissim, K. (2003, June). Revealing information while preserving privacy. In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 202-210).

Privacy-Preserving Machine Learning

What is privacy?

Privacy:

- several parties compute a function $f\colon \mathscr{D} \rightarrow y$ with private inputs $d_1, \ldots, d_n$
- privacy is protecting each party's private input
- such that other parties can not deduce information that is not already deducible from the output y

Dinur, I., & Nissim, K. (2003, June). Revealing information while preserving privacy. In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 202-210).

Methods

Privacy-Preserving Machine Learning

An overview of techniques

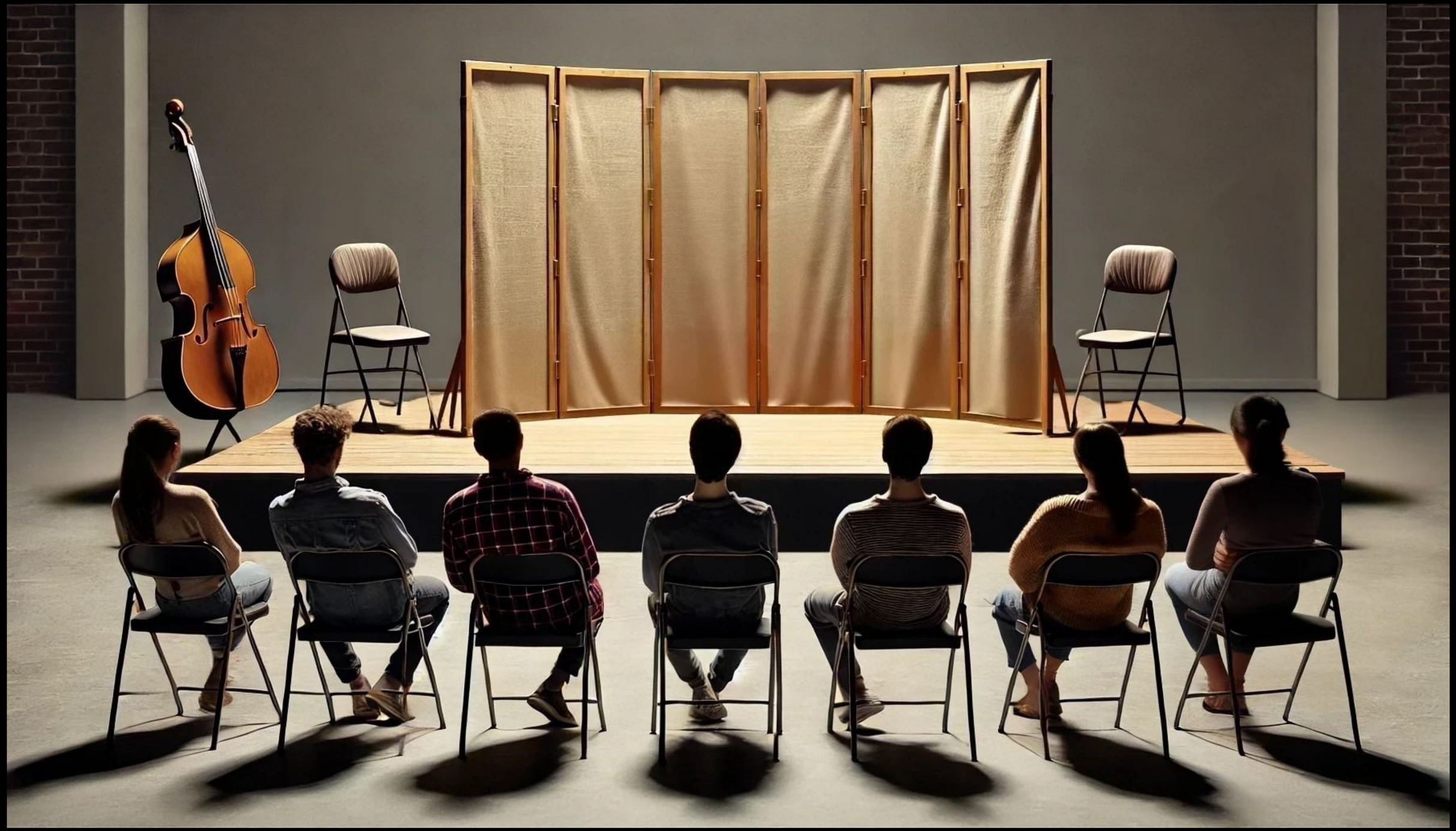
- Anonymisation
- ϵ-Differential Privacy
- Secure Multi-Party Computation
- Homomorphic encryption
- Federated Machine Learning

Anonymisation

Anonymisation

Anonymisation (like removing names and social security numbers) is often not enough to protect privacy since there are other indirectly identifying attributes

Sweeney, L. (2000). Simple demographics often identify people uniquely. *Health (San Francisco)*, *671*(2000), 1-34.



Anonymisation

ZIP 60602 consisted primarily of a retirement community and there were less than 12 people of an age under 65 living there.

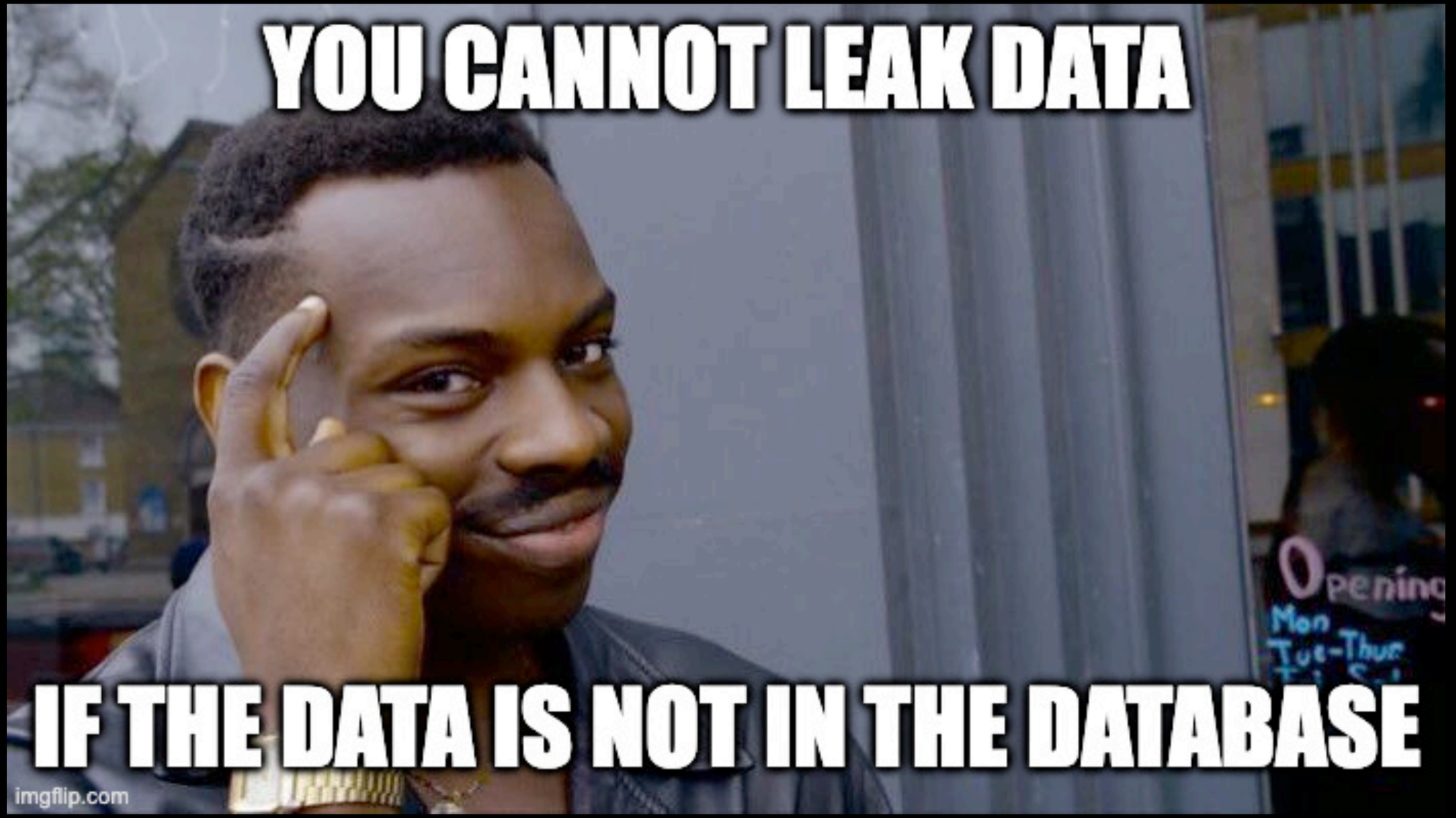
ZIP 60140 is the postal code for Hampshire and there were only two black women who resided in that town.

Likewise, 62052 had only four Asian families.

ZIP	Birth	Gender	Race
60602	7/15/54	m	Caucasian
60140	2/18/49	f	Black
62052	3/12/50	f	Asian

Figure 5 Data that looks anonymous

Sweeney, L. (2000). Simple demographics often identify people uniquely. *Health (San Francisco)*, *671*(2000), 1-34.


YOU CANNOT LEAK DATA
IF THE DATA IS NOT IN THE DATABASE
imgflip.com

ϵ-Differential Privacy

ϵ-Differential Privacy

Idea:

a persons privacy cannot be compromised if their data are not in the database.

We do this by adding noise to the data.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3 (pp. 265-284). Springer Berlin Heidelberg.

ϵ-Differential Privacy

- If the database contains data from a single person, that person's data contributes 100%. If the database contains data from a hundred people, each person's data contributes just 1%
- When a query is made on the data of fewer and fewer people (even with aggregates like average) more noise needs to be added to the query result to produce the same amount of privacy.

**Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis.
In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3 (pp. 265-284). Springer Berlin Heidelberg.**

ϵ-Differential Privacy

$$P[Q(D) \in S] \leq e^{\epsilon} P[Q(D') \in S]$$

Where

- Q is a query function
- D and D’ are two datasets, differing by one element
- S: a possible subset of outputs
- ϵ : a parameter representing the privacy loss. Smaller values give greater privacy.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3 (pp. 265-284). Springer Berlin Heidelberg.

ϵ-Differential Privacy

$$P[Q(D) \in S] \leq e^{\epsilon} P[Q(D') \in S]$$

The goal is to ensure that the inclusion or exclusion of a single individual (D vs D') does not significantly affect the output of a computation, thereby preventing the disclosure of any individual's information.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3 (pp. 265-284). Springer Berlin Heidelberg.

ϵ-Differential Privacy

- The sensitivity of the query is defined as: $\Delta f = \max_{D,D'} \|f(D) - f(D')\|$
- This gives us a guideline for how much noise we should add:

  Noise $\sim$ Laplace $\left(\frac{\Delta f}{\epsilon}\right)$
- Where Laplace refers to the Laplace distribution with pdf:

  $\text{Lap}(x \mid b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$

ϵ-Differential Privacy

Rules of thumb for choosing ϵ

- Strong privacy: range 0.01 - 0.1, probably degrade the data
- Moderate privacy: range 0.1 - 1, some utility loss
- Weaker privacy: range 1 - 10, better data utility

Downside: there is a trade-off between accuracy and privacy

Things to remember about PyDP:

- 🚀 Features differentially private algorithms including: BoundedMean, BoundedSum, Max, Count Above, Percentile, Min, Median, etc.
  - All the computation methods mentioned above use Laplace noise only (other noise mechanisms will be added soon! 😃)
- 🔥 Compatible with all three types of Operating Systems - Linux, macOS, and Windows 😃
- ⭐ Use Python 3.x.

https://github.com/OpenMined/PyDP

Secure Multi-Party Computation

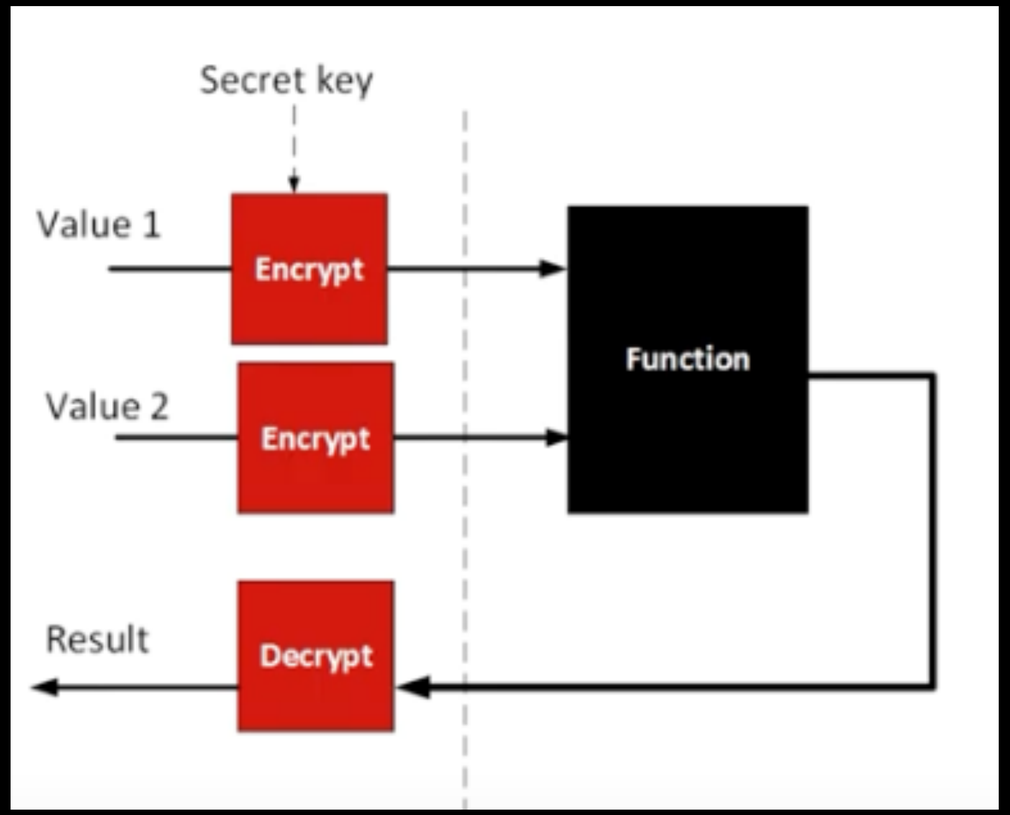
Secure Multi-Party Computation

- A subfield of cryptography
- Enables multiple parties to compute a function over inputs while keeping those inputs private.
- There is a multitude of cryptographic techniques to achieve this

Secure Multi-Party Computation

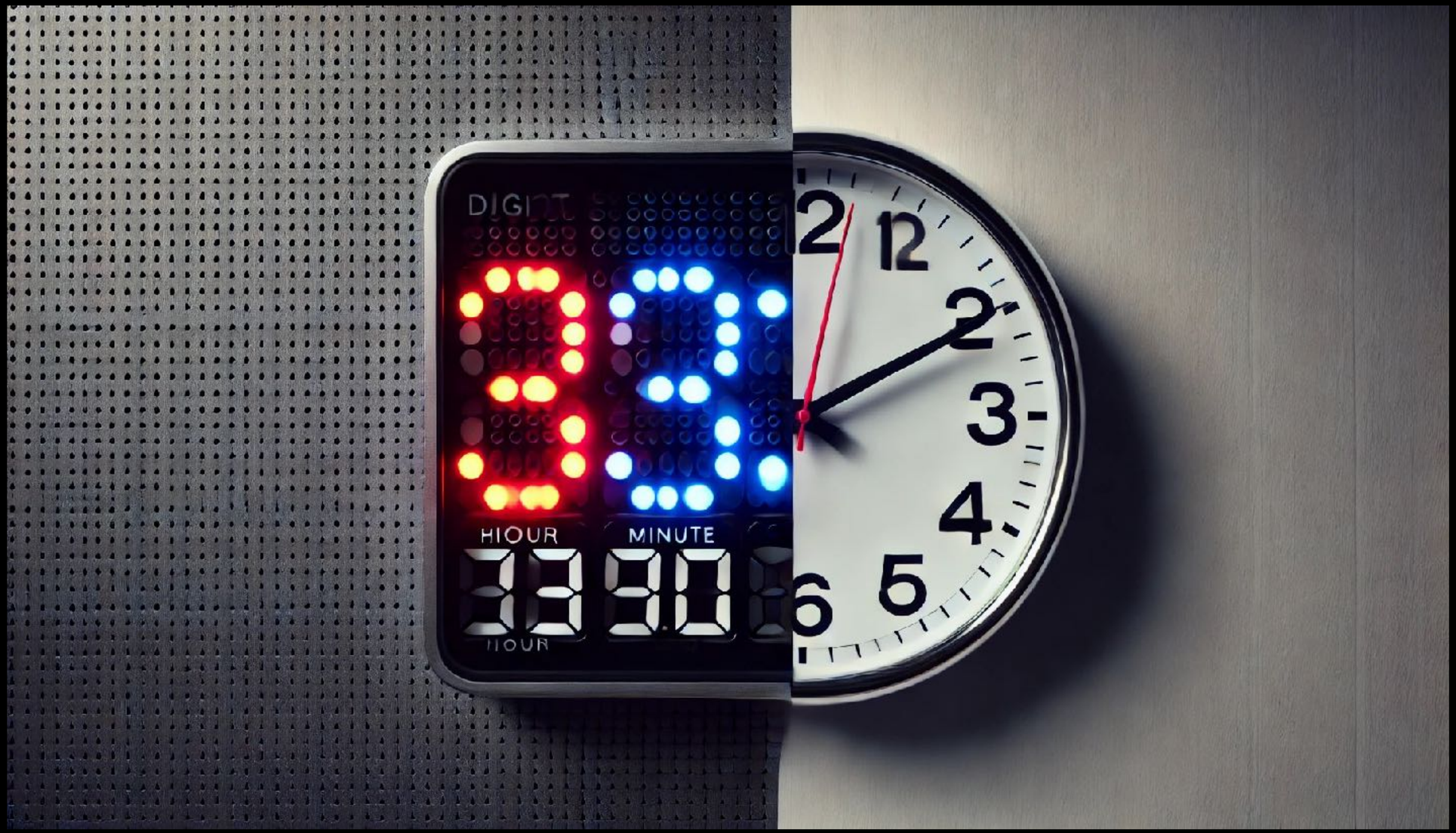
SMPC uses cryptographic tricks to make it impossible to back-engineer the input from the output. The downside of this approach is that:

- It is not always simple to find encryption schemas for your specific function
- Cryptography is a field on its own and not easy to understand, so you will probably depend on someone else having implemented the functions



Homomorphic

- Homos: same
- Morphe: shape
- Homomorphism: a map preserving all the algebraic structures between the domain an the range of a set.



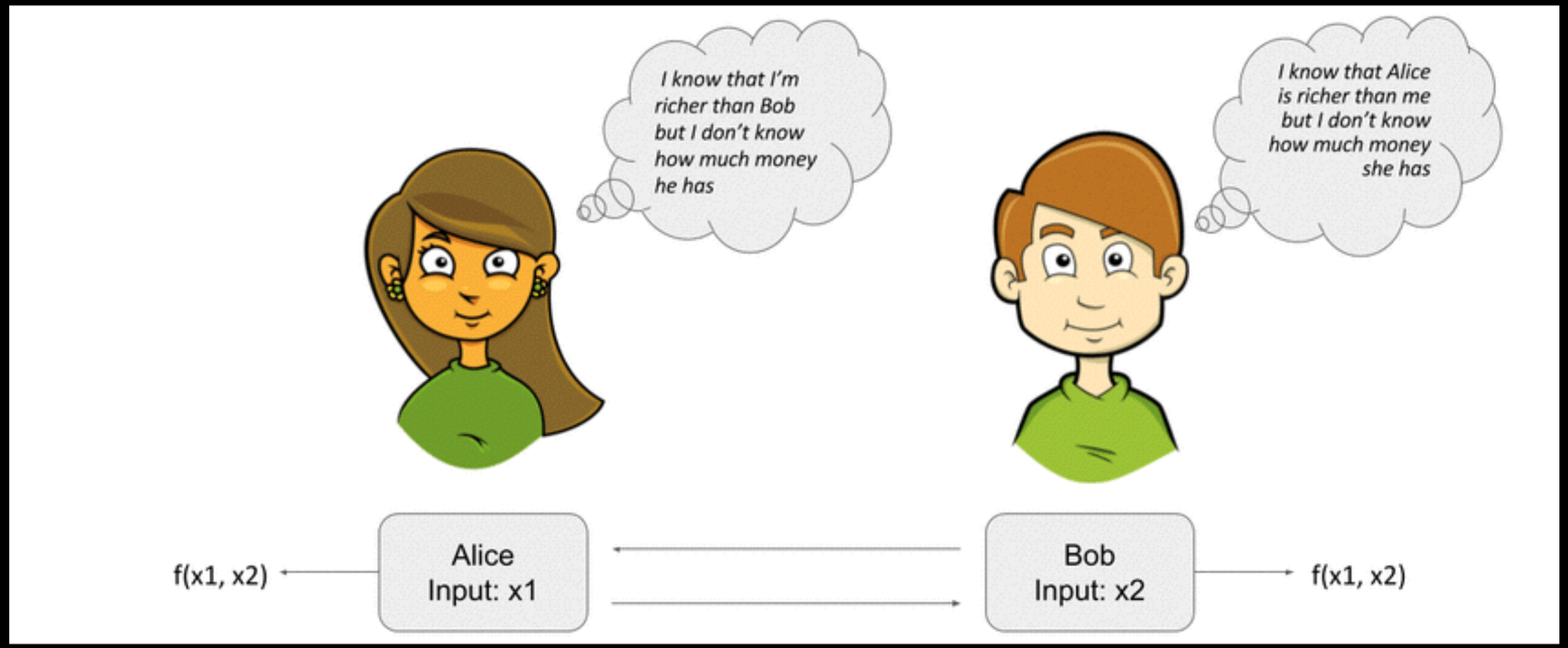
The mapping between the digital and analog time is homomorphic because it preserves the underlying structure (the representation of time) even though the form of representation differs.

Reading is a homomorphism between ideas

Homomorphic Encryption Schemes

Homomorphic encryption allows a third party to perform calculations on the encrypted data while preserving the features of the function and format of the encrypted data

Homomorphic encryption



https://www.researchgate.net/figure/Millionaires-problem_fig1_320290997

Homomorphic encryption

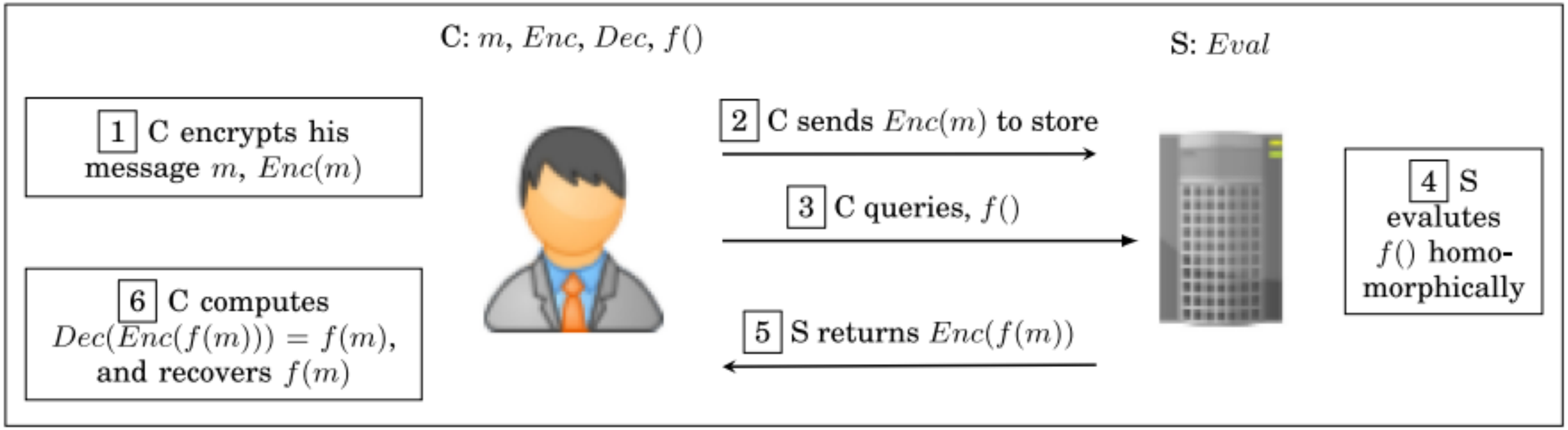


Fig. 1. A simple client-server HE scenario, where C is Client and S is Server.

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, *51*(4), 1-35.

Homomorphic encryption

An encryption scheme is called homomorphic over an operation “$\star$” if it supports the following equation:

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \forall m_1, m_2 \in M$$

Where E is the encryption algorithm and M is the set of possible messages.

Homomorphic encryption

It is sufficient to allow only addition and multiplication operations for “$\star$” because any Boolean circuit can be represented using only XOR (addition) and AND (multiplication) gates.

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \forall m_1, m_2 \in M$$

Pyfhel

Python for Homomorphic Encryption Libraries

Python library for Addition, Subtraction, Multiplication and Scalar Product over *encrypted* integers (BFV/BGV schemes) and approximated floating point values (CKKS scheme). This library acts as an optimized Python API for C++ Homomorphic Encryption libraries.

🎏 **Language**	Python (3.10+), with Cython and C++ (⚠️ *requires a C++17 compiler* ⚠️)
💻 **OS**	Linux, Windows & MacOS.
🔢 **Version**	3.5.0 (stable)
📚 **Docs**	In readthedocs!
✏️ **Demos/Examples**	In the docs with the outputs, sources in the `examples` folder.
🔌 **Backends**	SEAL, OpenFHE (WIP). Shipped alongside Pyfhel.
👷 **Authors**	Alberto Ibarrondo (IDEMIA & EURECOM) and Alexander Viand (ETH Zurich).
🎓 **Original Collaborators**	Melek Onen (EURECOM), Laurent Gomez (SAP Labs).

https://github.com/ibarrond/Pyfhel

Federated Machine Learning

Federated Machine Learning

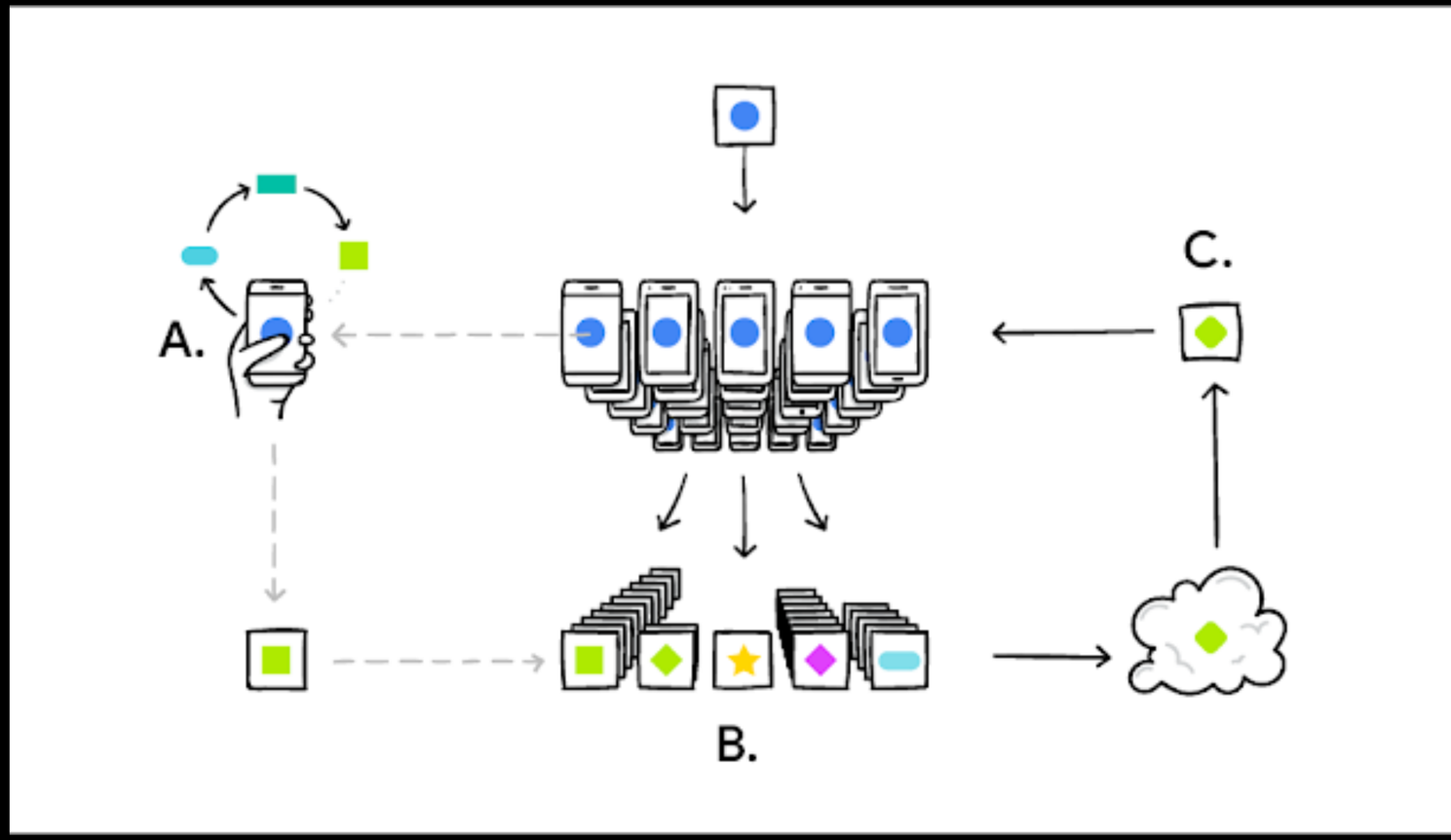
- Define $\mathcal{N}$ data owners $\{\mathcal{F}_1 \ldots \mathcal{F}_{\mathcal{N}}\}$
- They train on data $\{\mathcal{D}_1 \ldots \mathcal{D}_{\mathcal{N}}\}$
- Conventional is to use $\mathcal{D} = \bigcup_{i=0}^{\mathcal{N}} \mathcal{D}_i$ to train a model $\mathcal{M}_{\mathcal{D}}$
- In federated learning, we construct $\mathcal{M}_{FED} = \bigcup_{i=0}^{\mathcal{N}} \mathcal{M}_{\mathcal{D}_i}$
- In addition we want $\left| loss(\mathcal{M}_{\mathcal{D}}) - loss(\mathcal{M}_{FED}) \right| < \delta$

Federated Machine Learning

- Horizontal Federated Learning: shared feature space
- Vertical Federated Learning: shared ID space
- Federated Transfer Learning: use a limited common sample to learn a mapping between feature spaces

Federated Learning

- Download the current model
- (A) train the model on your local data
- (B) aggregate many users updates
- (C) average the updates to a consensus update
- Repeat



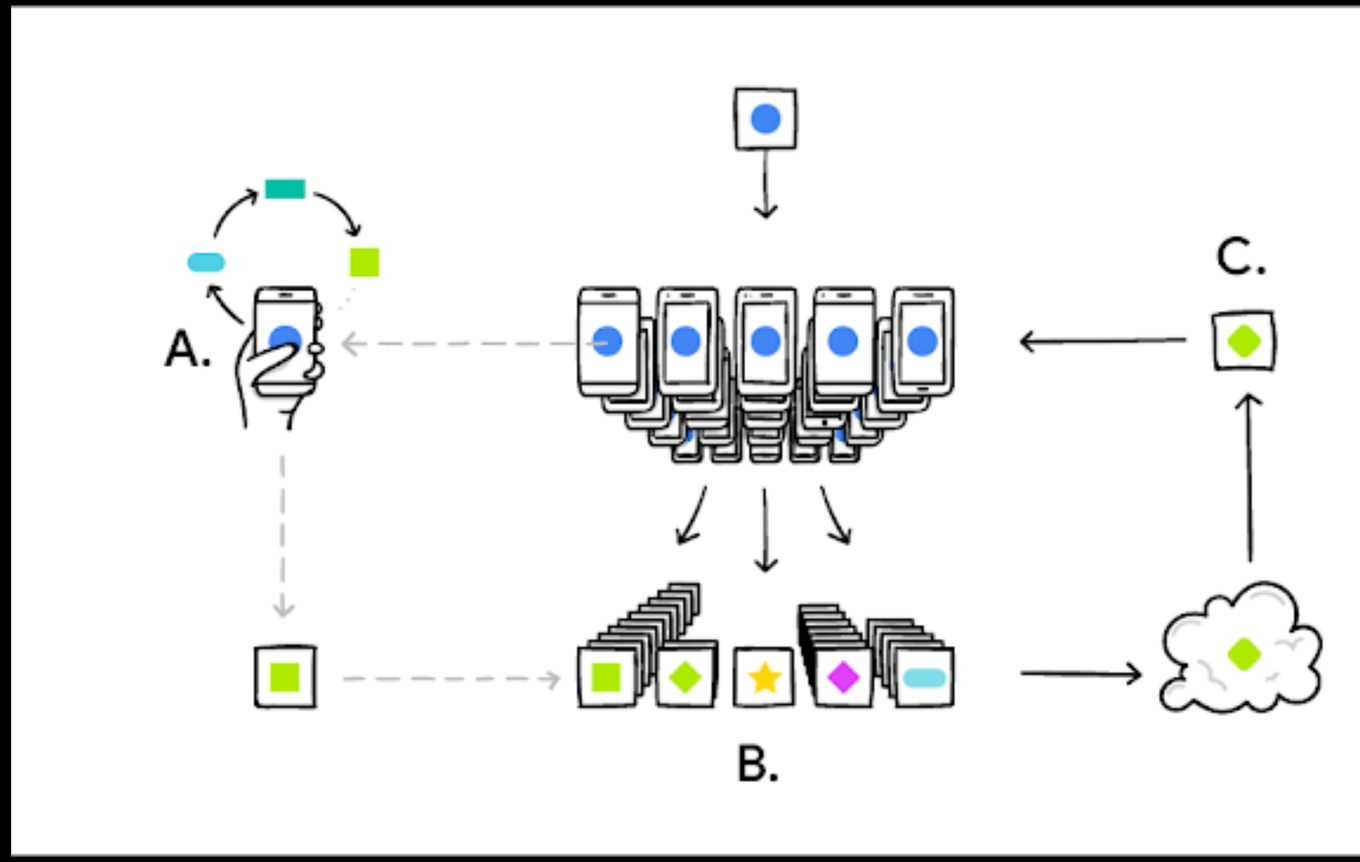
Federated Learning

Typical machine learning are

- Highly iterative
- Low latency
- High throughput

While the federated situation is

- Only intermittently available for training
- High latency
- Lower throughput

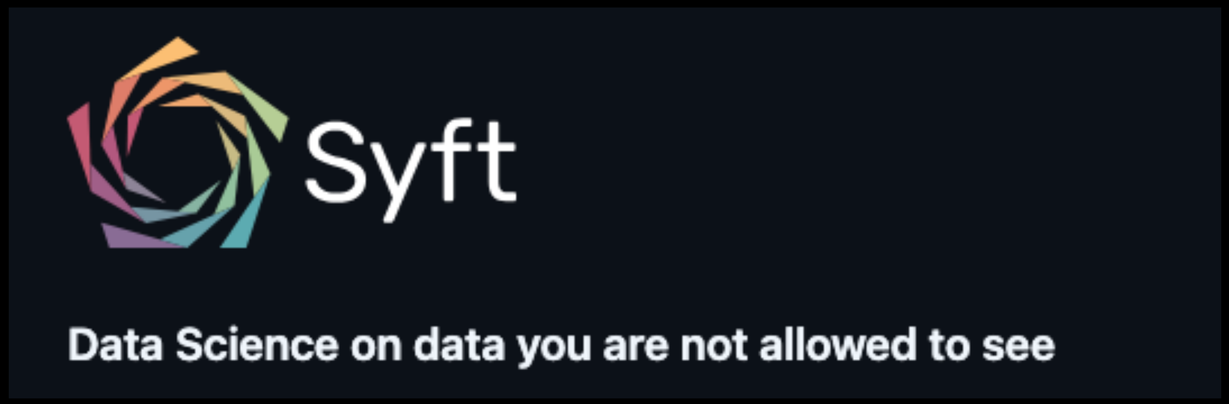


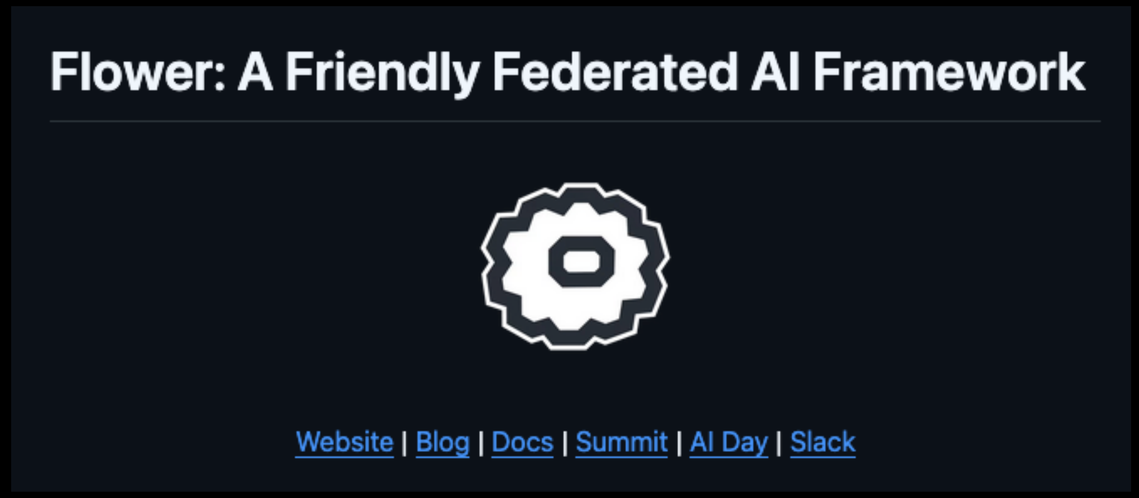
Federated Learning

Note that:

- Federated machine learning does preserves privacy
- However, theoretically, the model could still learn negative behaviour (eg let's say it learns how to keep you "engaged" on eating disorder themed content by inducing a negative body image)
- Now the model can use this to "help" others find the "right" content to "feed" their eating disorder scrolling behaviour
- This might be legally correct, and it preserves privacy, but is it ethical to "share" this information while assuring users that this is "privacy by design"?


FEDERATED MACHINE LEARNING





https://github.com/OpenMined/PySyft

https://github.com/adap/flower