# Federated Learning summary

G. Kalitsios, K. Tsechelidou Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr

Version 1.2

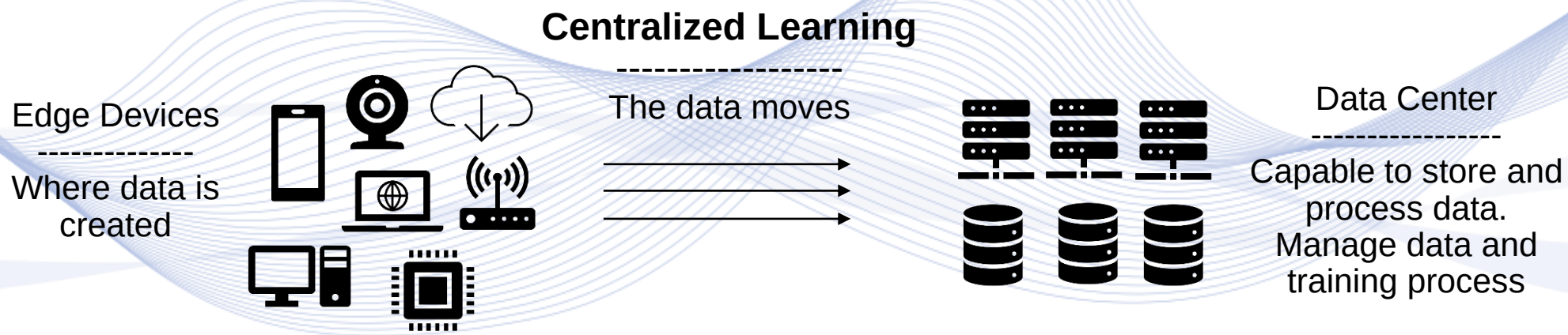
Artificial Intelligence & Information Analysis Lab

# Federated Learning

- Introduction
- Centralized Learning
- Decentralized Learning
- Federated Learning
- Platforms
- Federated Learning Algorithms
- Privacy Principles & Technologies
- Challenges
- Benefits and costs
- Applications
- Open problem areas
- Conclusion

# Introduction

- Most data nowadays are:
  a) created on edge devices such as smartphones, on the internet, IoT sensors attached to industrial equipment or
  b) are controlled by entities, such as institutions (e.g., hospitals) or companies.
- Current practices dictate that for training ML-based models, we move the data to a data center, where the ML model is located.



# Introduction

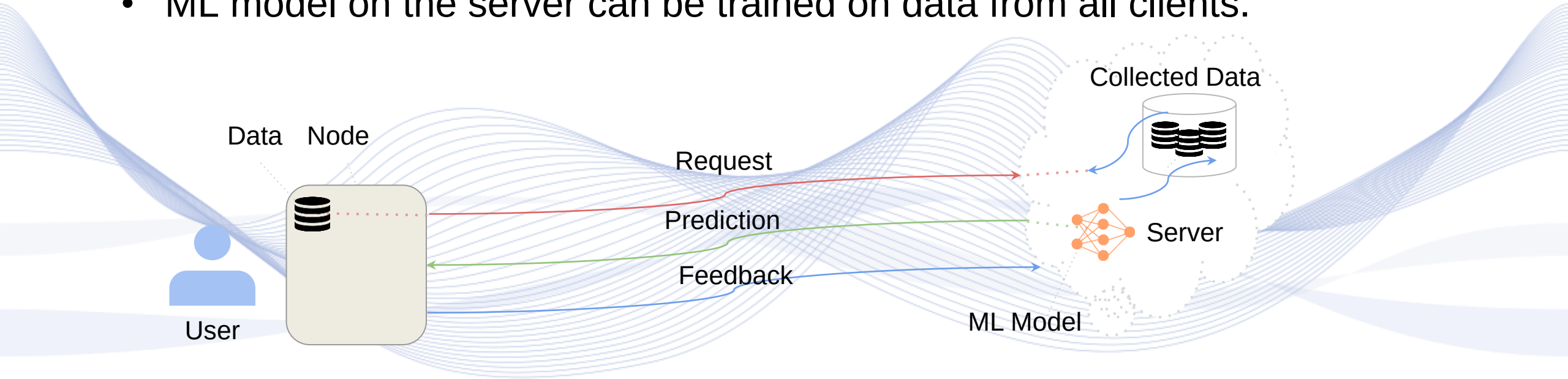
- ***Problems***
- There are strict regulations regarding ***data privacy*** nowadays.
- For this reason, it is usually not practical to collect and upload consumer data into a central location.
- ***Questions Raised***
- How can we train ML systems from decentralized data?
- How can we exploit the relevant information, without direct access to the data themselves?
- How can the data owners of the data be sure that they will not be exposed?

# Centralized Learning

- ML model (to be trained) and all data are in one place.
- Traditionally there is a server in the cloud that hosts the trained ML model and all clients talk to it to make a prediction on their behalf.

***Advantages***

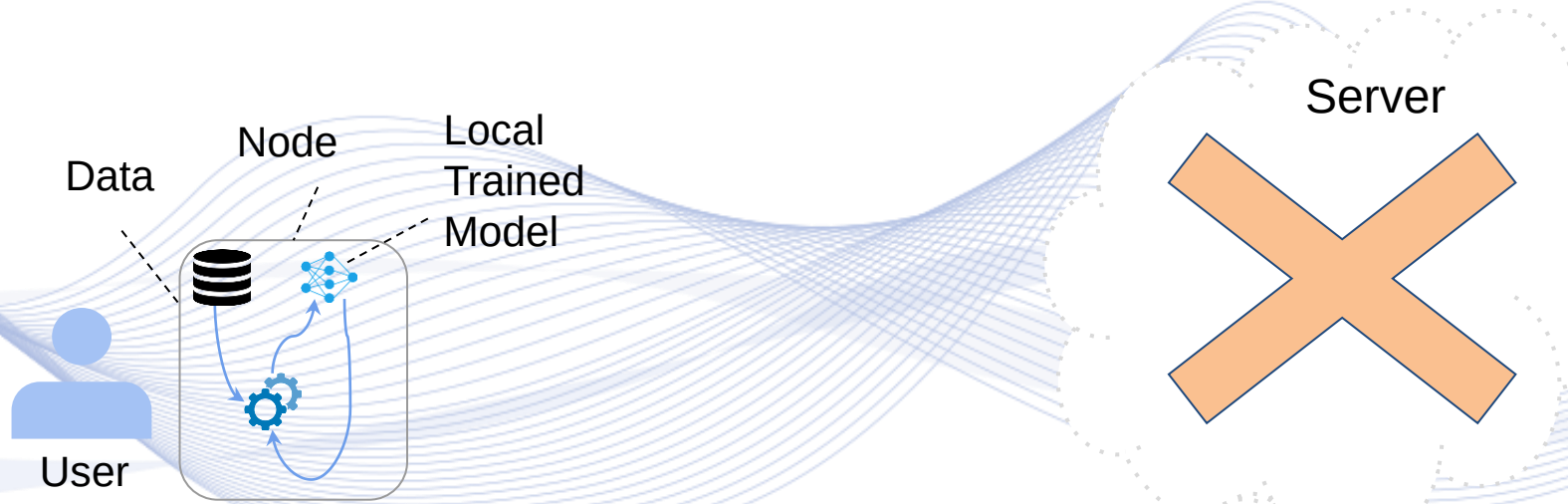
- ML model on the server can be trained on data from all clients.



# Decentralized Learning

- Why do we even need a server? Can we learn without collecting data.
- Each user is limited to its own data.

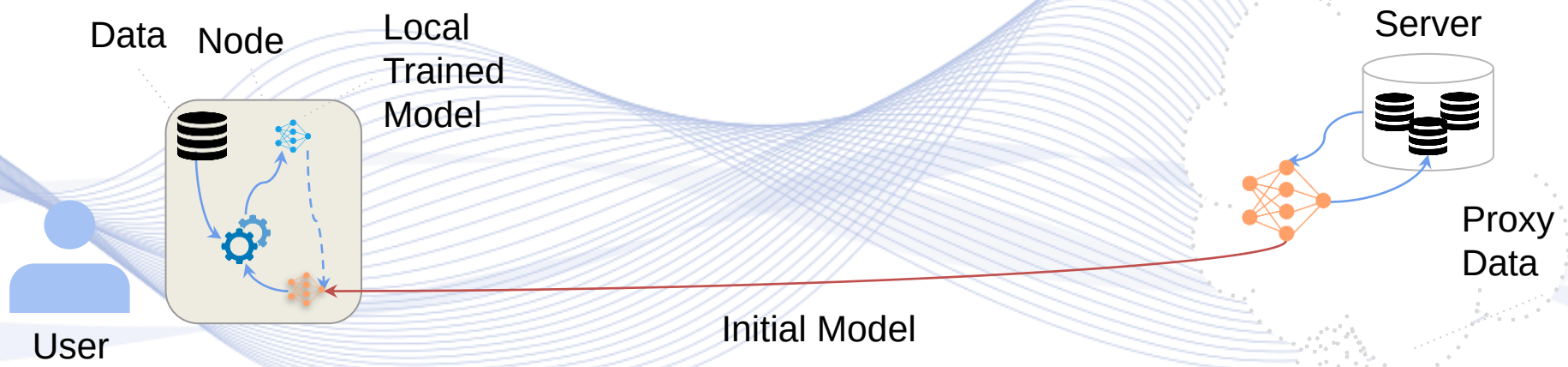
- Each user works autonomously (trains its own model).
- No data leave the user device.



# Decentralized Learning

***One-way decentralized learning scenario.***

- Initial ML model is pre-trained using proxy data on the server.
- It is deployed and further improved (re-trained) on user device.
- Nothing leaves the user device.
- Once deployed, it can not easily learn new ***common*** data patterns.



# Federated Learning

***Federated Learning*** (***FL***) is a ***Distributed Learning*** technique, which is built upon the core idea revolving around ***data privacy***.

- Google introduced Federated Learning in 2017.
- Federated Learning attempts to answer the question: ***Can we train our ML model, without the need to transfer out data over to a central location?***

- Unlike the traditional ML approach, now each local ML model is shared among the servers in the data center.
- No user data are shared.

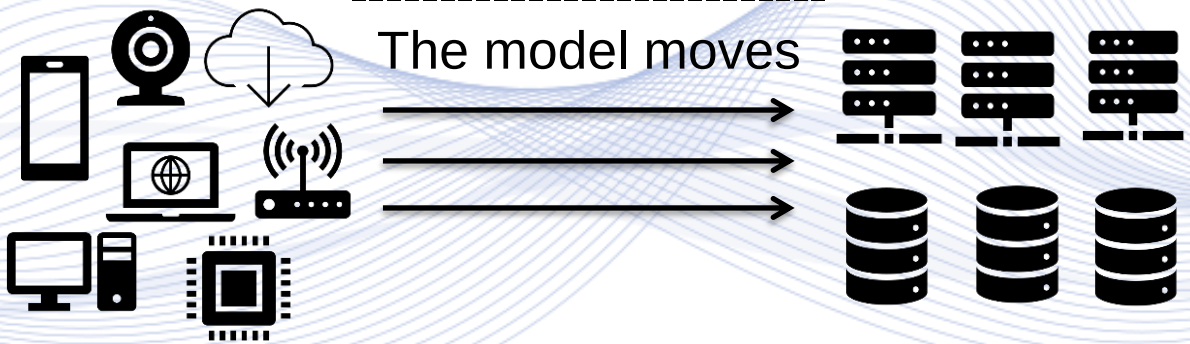
# Federated Learning

- Each user/entity device trains its own ML model locally.
- The server combines the local ML models from different devices into a single federated model and it never has direct access to the training data.

**Edge Devices**

---

Where data is created.
Manage data and train the **local ML model.**



**Federated Learning**

---

The model moves

**Data Center**

---

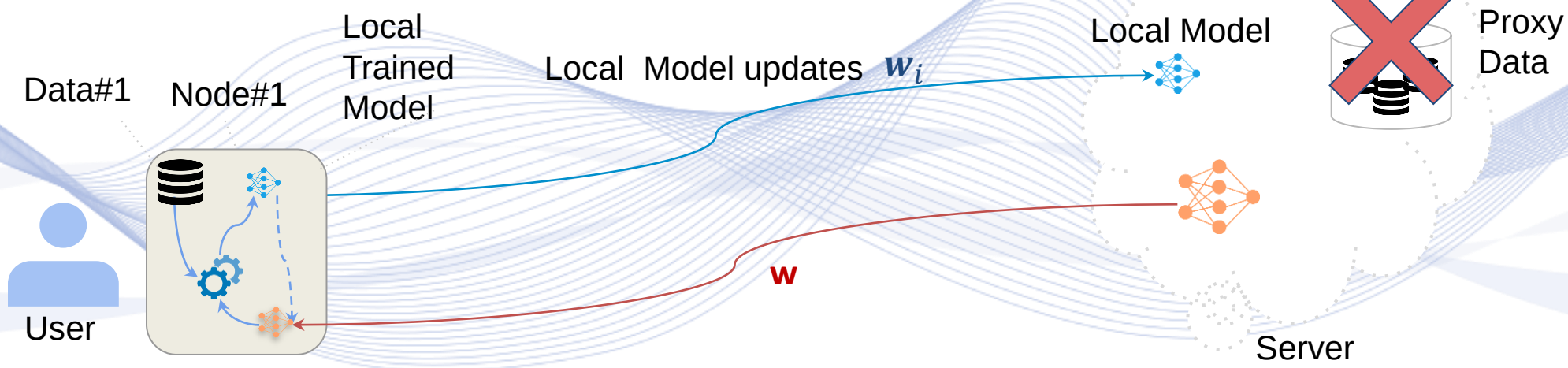
Rules the FL process.
Manage the **global ML model** and the communication with the edge parties.

# Federated Learning Process(1)

A typical Federated Training process includes:

1. **Broadcasting:** The clients download the current model parameters **w**

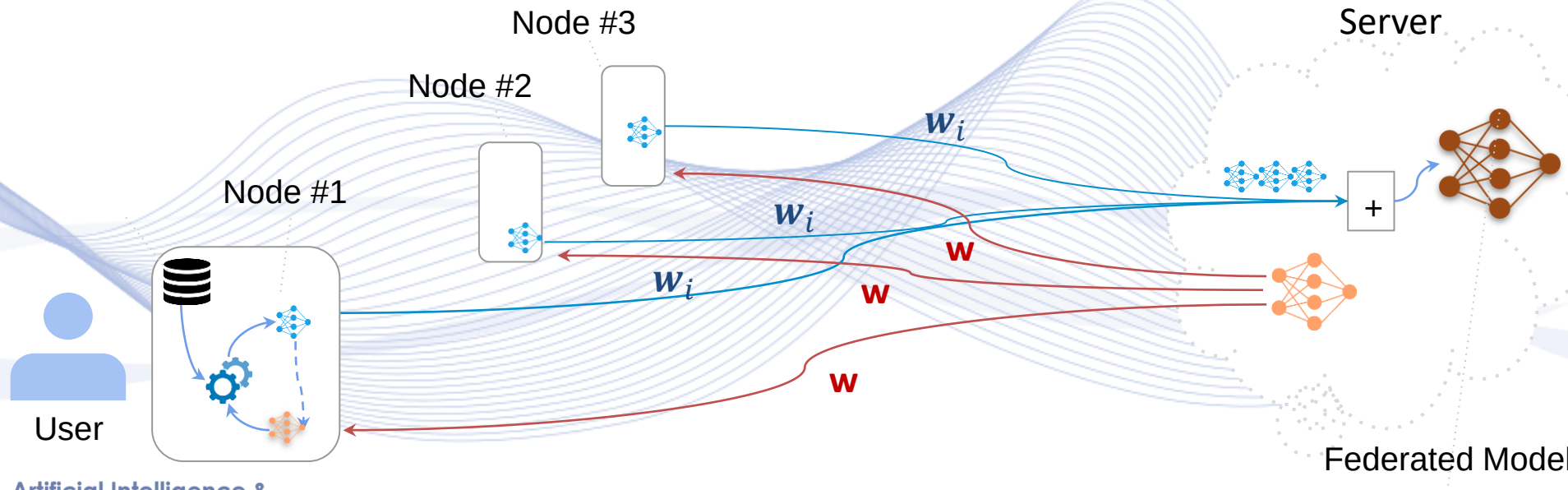
2. **Local training:** The clients locally compute training parameters (or gradients) and send locally trained model updates $\boldsymbol{w}_i$ of i-th client to the server.



# Federated Learning Process (2)

3. **Model Aggregation**: The server performs secure aggregation over the uploaded parameters from the clients without learning local information, to construct an improved global model.

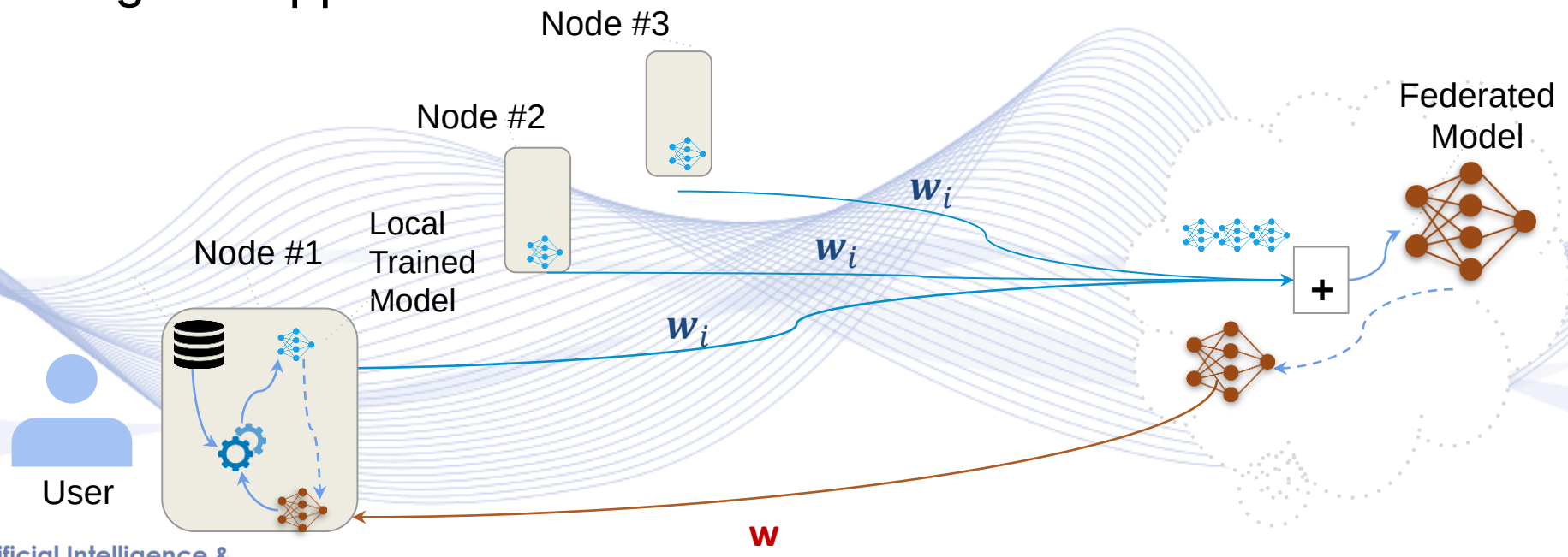
Typically, Federated Model Averaging Algorithm is used in aggregation phase.



# Federated Learning Process (3)

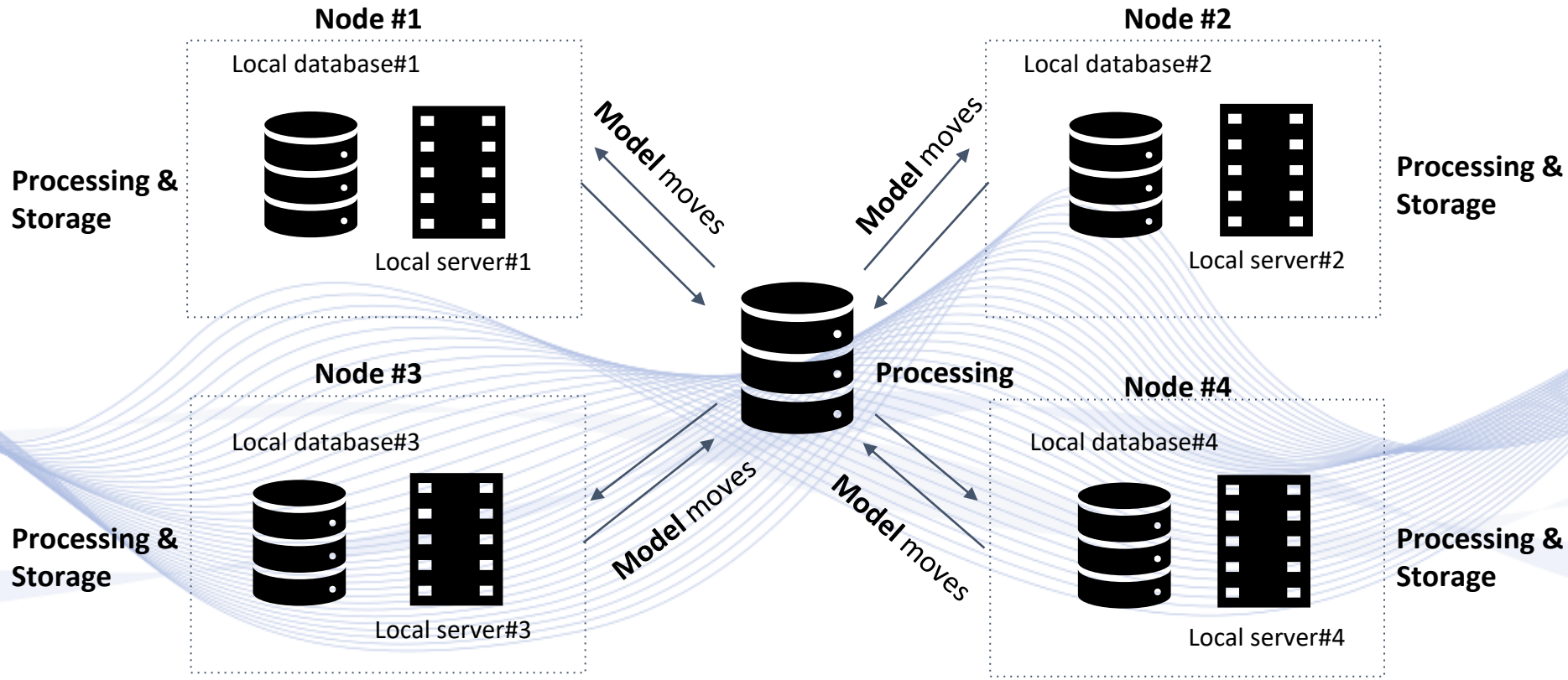
4. **Model updating**: The server locally updates the model based on the aggregated updates computed from the clients that participated in the current round.

The server orchestrates the training process, by repeating the previous steps until training is stopped.



# Federated Learning

- An example of a ***star-like FL topology*** is given here, where each client process and stores data locally (no data sharing happens), while only model updates travel through the network.



# Federated Learning Definition

Let us consider a typical FL system consisting of one server and N clients. Let $D_i$ denote the local database held by the client $C_i$ , where $i \in \{1, 2, \ldots, N\}$. The server aggregates the weights (local models' parameters) received from the N clients as:

$$\mathbf{w} = \sum_{i=1}^{N} \frac{|D_i|}{|D|} \boldsymbol{w}_i$$

where $\boldsymbol{w}_i$ is the parameter vector trained at the i-th client, **w** is the parameter vector after aggregating, $\frac{|D_i|}{|D|} \geq 0$ with $\sum_{i=1}^{N} \frac{|D_i|}{|D|} = 1$ and $|D| = \sum_{i=1}^{N} |D_i|$ is the total size of all data samples. As an optimization problem, this can be formulated as:

$$\mathbf{w^*} = \arg min \sum_{i=1}^{N} \frac{|D_i|}{|D|} f_i(\boldsymbol{w}, D_i)$$

where $f_i$ is the local loss function of the $C_i$.

# Federated Averaging Algorithm

***Federated Averaging Algorithm*** is the first FL algorithm introduced by Google. Its steps are:

- We start with FL manager choosing some clients to participate in one federated round and send them the initial ML model.
- FL manager job is to orchestrate this process and feed the same initial model to all the nodes.
- Each client trains the model locally using its own local data.
- Then each client send back their new modified local model weights to the FL manager.

# FedProx Algorithm

***FedProx algorithm*** is a modification of Federated Averaging Algorithm.

- With some simple modifications, it achieves better performance and heterogeneity.
- Each device that participates in a federated cycle has its own performance and its own limitations, so it does not make sense to expect all devices to do the same job.
- The FedProx algorithm can accommodate both partial and uniform work.
- In this way, FL system heterogeneity is accounted for.

- FedProx algorithm is more stable than Federated Averaging Algorithm.

# FedMA Algorithm

***FedMA algorithm*** is used to implement federated learning with Neural Networks.

- This algorithm builds the shared global model with a layer-by-layer mapping and averaging values of hidden layer weights with similar feature extraction signatures.
- First, the FL manager collects the client first layer weights and performs one-layer mapping to obtain the first layer weights of the federal model.
- The FL manager then sends these weights to the clients, who train all the other layers using their own datasets.

- FedMA is implemented in PyTorch.

# Federated Learning Concerns

- Although the local data of the clients is kept locally in FL, this is not ensure the privacy protection of them.

- FL raises concerns regarding the goals and capabilities of an **adversary**.
  - ***Model update poisoning***: the adversary controls directly and alters local models' updates sent to the server.
  - ***Data Poisoning Attacks:*** the adversary can only influence the data collection process at the edge of the Federated Learning system. It can be performed for targeted attacks (or back door) and untargeted attacks.
  - ***Inference-Time Evasion Attacks***: the model may be accessible to any malicious client during broadcasting phase.

# Privacy Principles

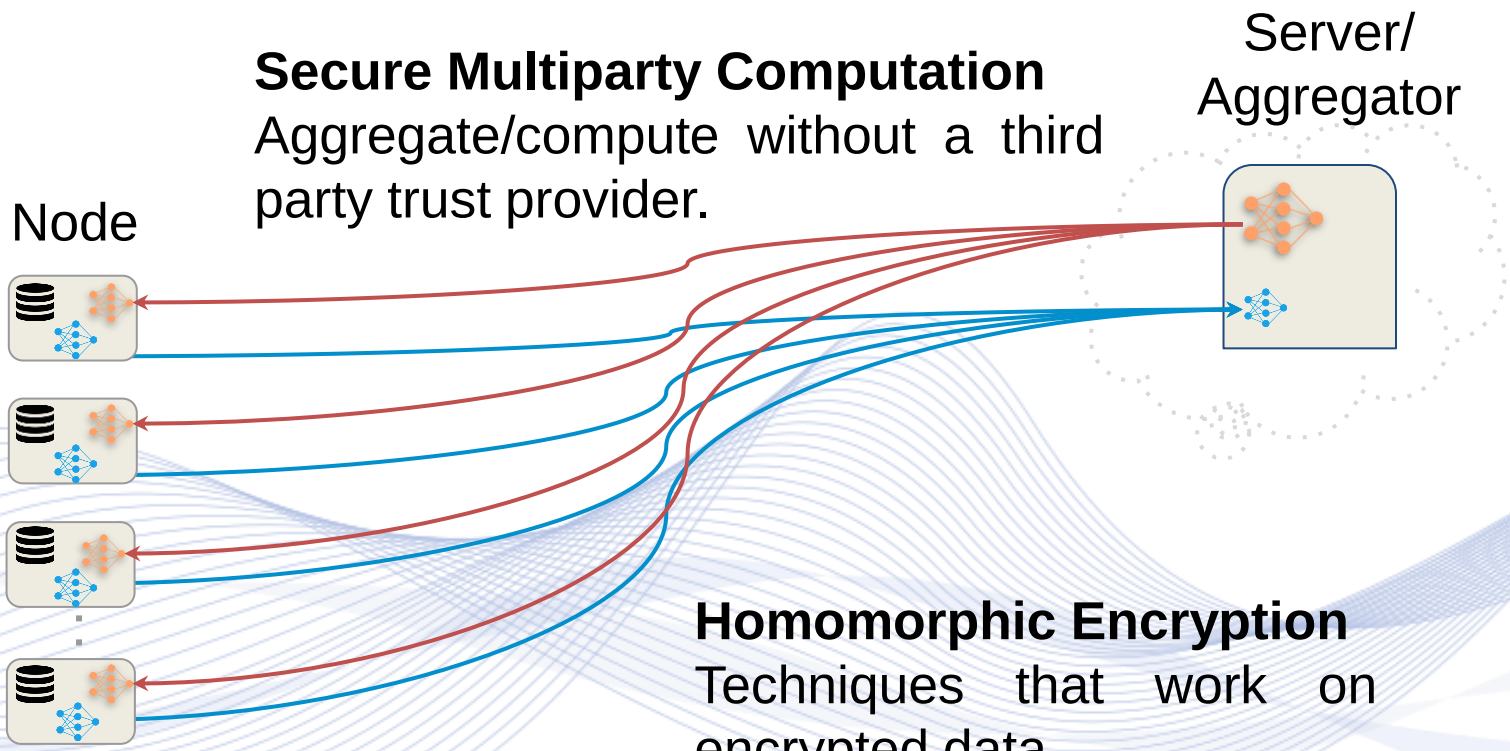
The main considerations of privacy enhancing techniques within the Federated Learning framework include:

- Trade off between ***privacy protection*** and ***ML system performance***.
- Level of trust within the local participants and the global server.
- Mechanisms for information leakage prevention.
- Traceability and accountability that limit the risk of misuse by third parties.

# Privacy Enhancing Techniques

- ***Differential Privacy***: Noise addition to data locally (Local Differential Privacy) or after aggregation step (Global Differential Privacy)
- ***Homomorphic Encryption***: Encryption in the data before they are going to be shared, so that data can be analyzed but its not possible to decoded them into the original format.
- ***Secure Multiparty Computation***: The data analysis is divided into several parts, so that no one can see the full set of inputs.
- ***Zero-knowledge Proof Technologies***: Users can prove their knowledge of a variable (e.g., price), without revealing the variable value itself.

# Privacy Enhancing Techniques in Federated Learning




**Secure Multiparty Computation**
Aggregate/compute without a third party trust provider.

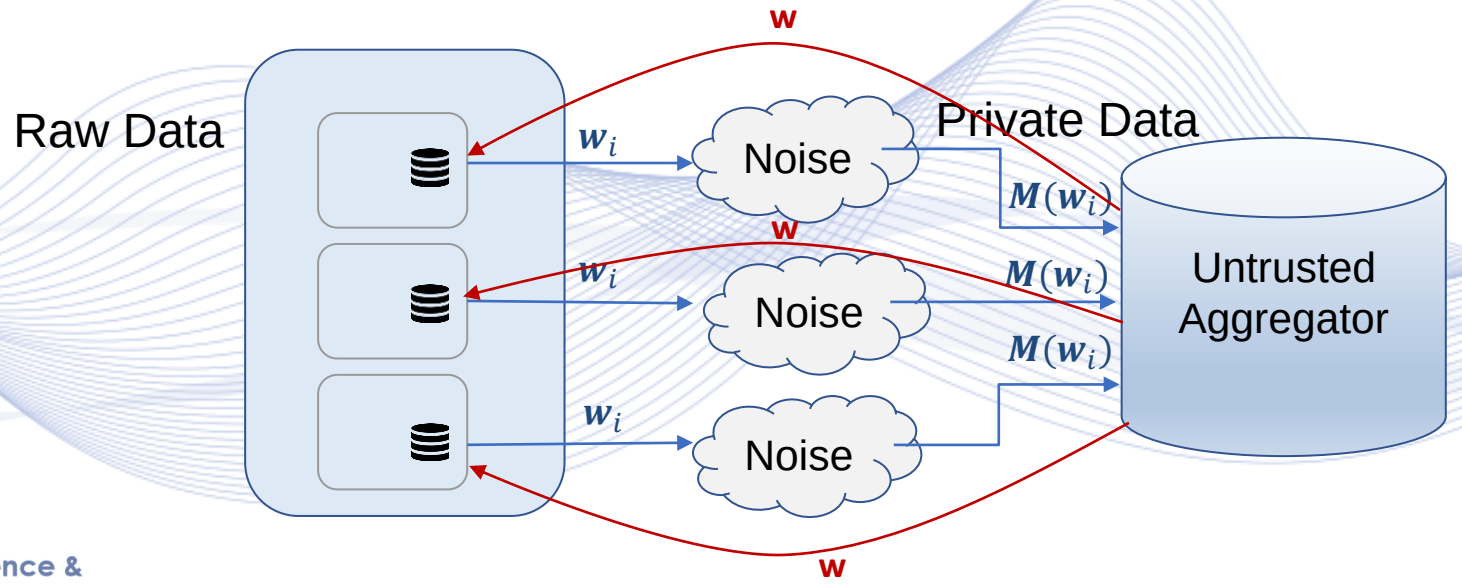
**Differential Privacy**
Adds calibrated noise on the client's data that protects against inference attacks.

**Homomorphic Encryption**
Techniques that work on encrypted data.

# Local Differential Privacy

- ***Local Differential Privacy*** (***LDP***) is a state-of-the-art approach that enables statistical calculations, while simultaneously protecting the privacy of each user. If $\mathcal{M}$ is a differentially private mechanism (e.g., noise addition), the model parameter after aggregation is:

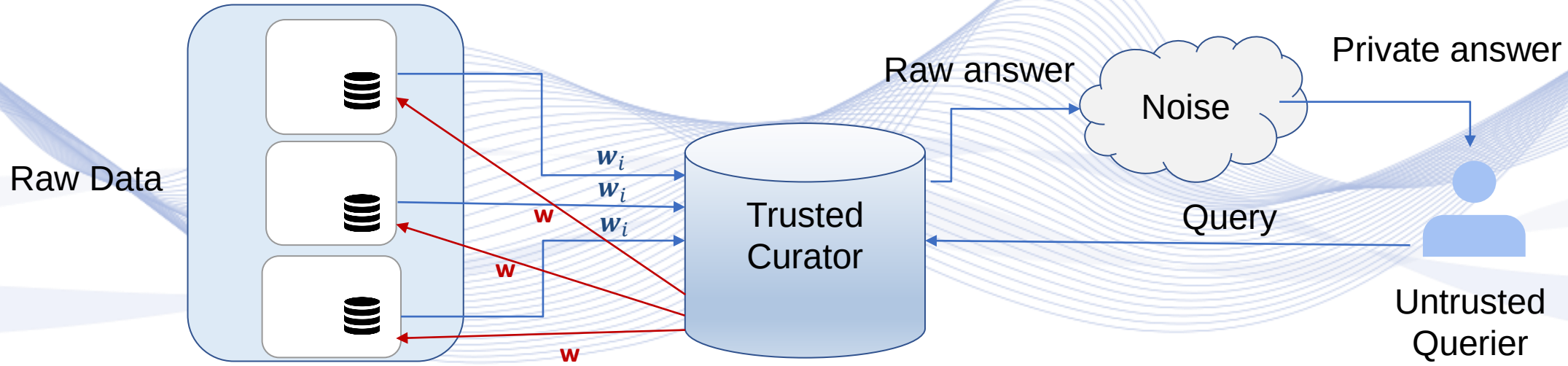
$$\boldsymbol{w} = \text{aggregate}\left(\mathcal{M}(\boldsymbol{w}_i), \mathcal{M}(\boldsymbol{w}_{i+1}), \ldots, \mathcal{M}(\boldsymbol{w}_N)\right)$$



# Global Differential Privacy

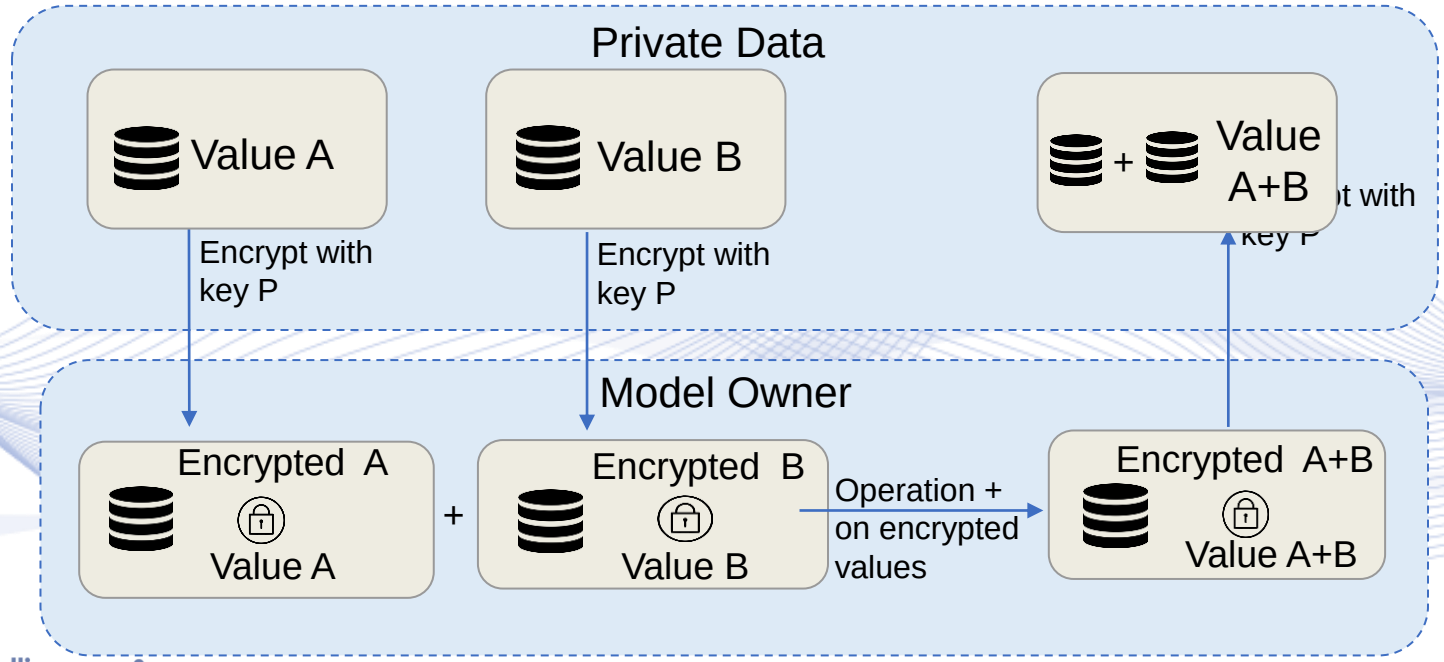
- In ***Global Differential Privacy (GDP)***, a central aggregator is used, who has access to the raw data from each user.
- If $\mathcal{M}$ is a differentially private mechanism (e.g., noise addition), the global parameter vector after aggregation is:

$$\boldsymbol{w} = \mathcal{M}\ (\text{aggregate}\ (\boldsymbol{w}_i, \boldsymbol{w}_{i+1}, \ldots, \boldsymbol{w}_N)$$



# Homomorphic Encryption

***Homomorphic Encryption*** (***HE***) allows computation of encrypted data, creating encrypted results that, when decrypted, match the results of operations, as if they were originally executed.



# Zero-knowledge Proof Protocol

***Zero-knowledge proof protocol***.

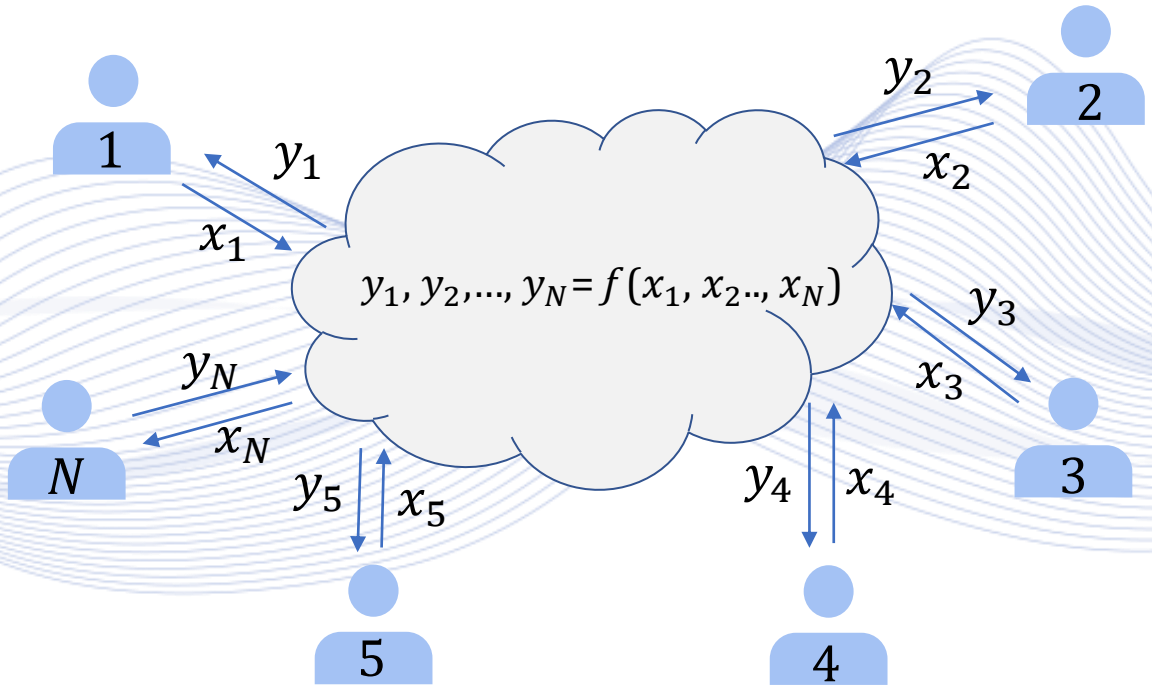
- At the first step, the prover receives ***authenticated private data*** from the trusted authority (on a regular basis or on demand).
- At the second step, the verifier makes a custom request upon the prover's personal data.
- The prover then, at the third step computes the response on the verifier's question and constructs the Zero-Knowledge proof of correct computation.

- At the fourth step, the response and the proof are transmitted to the verifier.

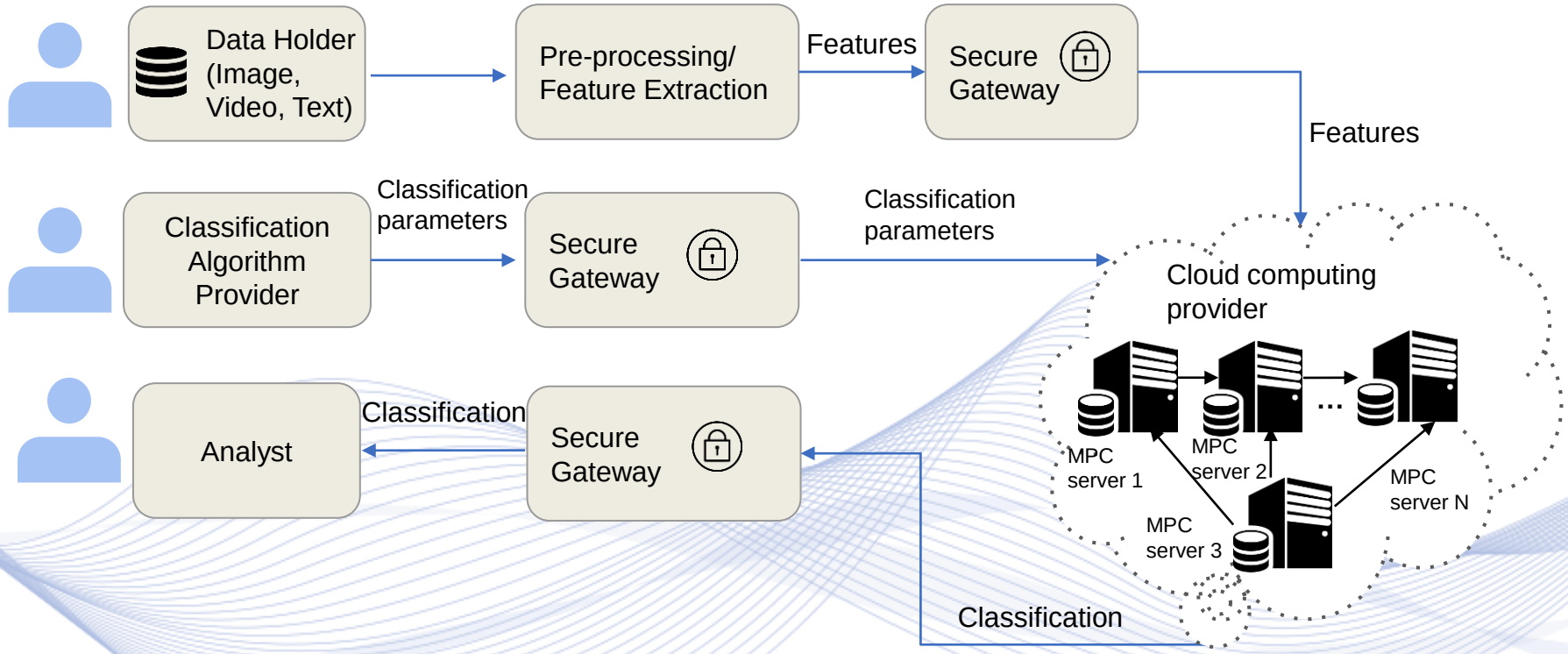
# Secure Multi-Party Computation

***Secure Multi-Party Computation*** (***SMC***).

- $N$ parties compute a function that keeps their own input private. Each party only has access to its own input-output pair.



# Secure Multiparty Computation




Characteristic use case of Secure Multiparty Computation.

# Federated Learning Challenges

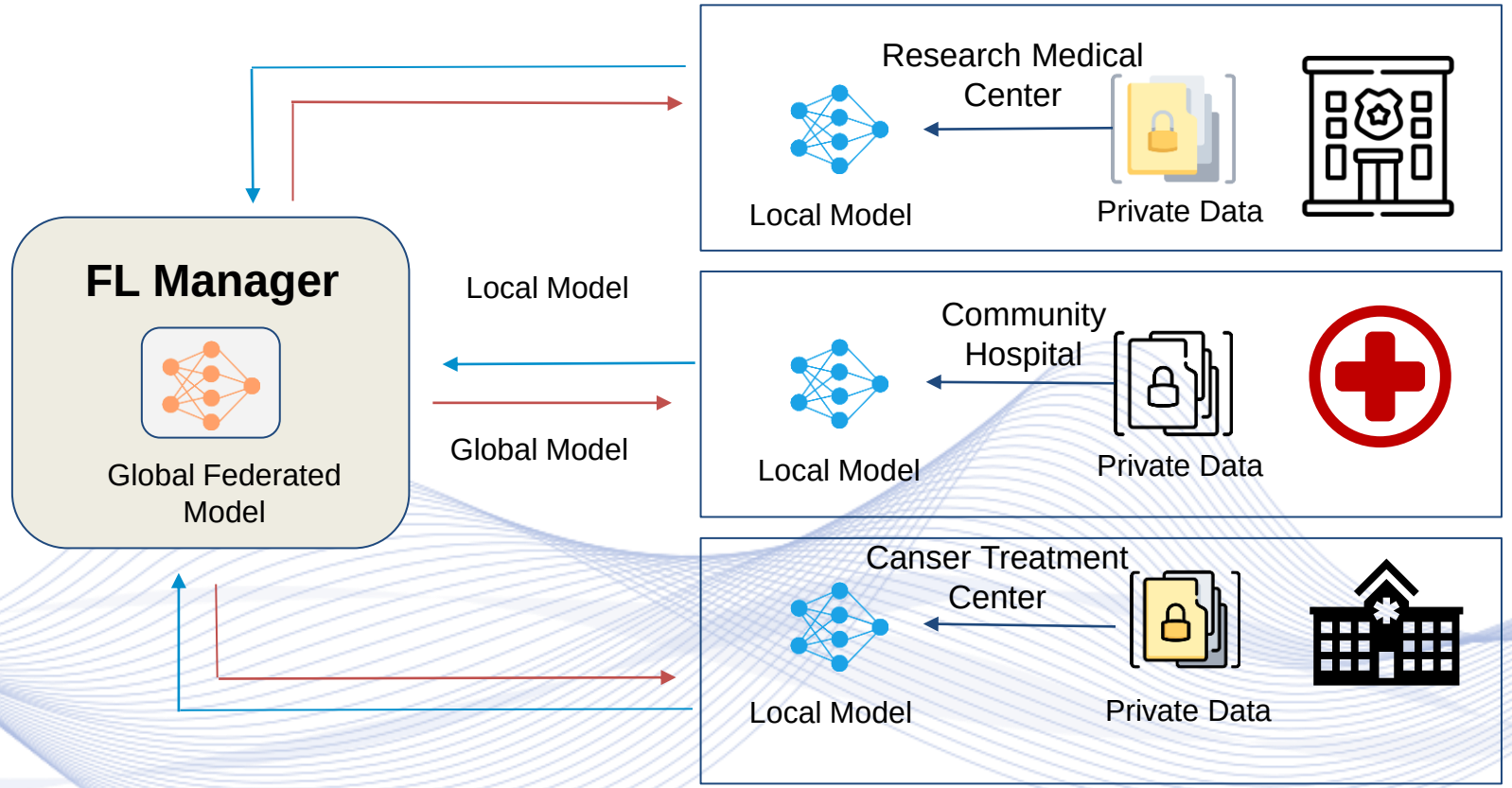
- ***Communication***: A lot of research is focusing on reducing the number of update rounds and the update message itself, i.e., the training derivatives.

- ***System heterogeneity***: FL systems are almost by definition heterogeneous in that the device may vary in storage, computational and communication capabilities.
  - Hence a FL training scheme must be dynamic or conform to the lowest denominator of the devices.

# Federated Learning Challenges

- ***Statistical heterogeneity***: Since ML systems typically rely on the ***Identical Independent Distributed*** (***IID***) data assumption, special techniques must be developed to handle the statistical data heterogeneity, if present.
- ***Privacy***: FL is designed to preserve privacy, but care still need to be given to ensure that sensitive information is not revealed for specific users or devices.

# Federated Learning Applications




Federated Learning application in Healthcare.

# Bibliography


[GOO2016] Goodfellow I, Bengio Y, Courville A, Bengio Y., *Deep learning* MIT press, 2016.
[HAY2009] S. Haykin, *Neural networks and learning machines*, Prentice Hall, 2009.
[BIS2006] C.M. Bishop, Pattern recognition and machine learning, Springer, 2006.
[GOO2016] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016
[THEO2011] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Elsevier, 2011.
[ZUR1992] J.M. Zurada, *Introduction to artificial neural systems*. Vol. 8. St. Paul: West publishing company, 1992.
[ROS1958] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.
[YEG2009] Yegnanarayana, Bayya. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.
[DAN2013] Daniel, Graupe. *Principles of artificial neural networks*. Vol. 7. World Scientific, 2013.
[HOP1988] Hopfield, John J. "Artificial neural networks." *IEEE Circuits and Devices Magazine* 4.5 (1988): 3-10.

# Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr