TRUstworthy Multi-site Privacy Enhancing Technologies

D2.1
SoA and BSoA for FL-tailored AI models, PET methods and tools

Lead Author: Alberto Pedrouzo-Ulloa (UVIGO)
With contributions from: Carlos García-Pagán (GRAD), Gonzalo Jiménez-Balsa (GRAD), Jaime Loureiro (GRAD), Xavier Martínez (GRAD), Inés Ortega (GRAD), Eva Sotos Martínez (GRAD), Jan Ramon (INRIA), Aleksei Korneev (INRIA), Twaha Mukammel (TVS), Fernando Pérez-González (UVIGO)
Reviewer: Jaime Loureiro (GRAD), Twaha Mukammel (TVS), Jan Ramon (INRIA)

Deliverable nature	**Report (R)**
Dissemination level	Public (PU)
Delivery date	30-09-2023
Version	0.5
Total number of pages	61
Keywords	Federated Learning, AI models, PET methods and tools

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

EXECUTIVE SUMMARY

This document reports the results corresponding to the work realized inside the scope of tasks T2.1 and T2.2 from WP2 of TRUMPET "Privacy-armored AI algorithms and models." These results are classified as two main different outcomes associated to each task:

- T2.1 Identification of AI (Artificial Intelligence) models and aggregation methods for Federated Learning (FL): We survey the state of the art in algorithms and tools for Federated Learning. Then, we identify and provide a list of the most adequate AI models to be used for the implementation of the TRUMPET platform and the different medical use cases.
- T2.2 Research of FL-tailored PET methods and tools: We survey the state of the art in PETs (Privacy Enhancing Technologies), and research their optimal application into the Federated Learning pipeline of the TRUMPET platform.

In relation to the T2.1 outputs, the document surveys the most relevant ML (Machine Learning) algorithms in FL for TRUMPET purposes. In particular, a detailed list of AI models and aggregation methods is presented, and a selection of the best candidates is made taking into account the requirements imposed by the TRUMPET use cases. This selection also includes a review of the most relevant FL frameworks and their adequacy to the needs of TRUMPET.

In relation to the T2.2 outputs, the document contains a description of the state of the art of several modern PET methods which are relevant for the FL scenario. First, a study of the advantages and shortcomings of each individual PET technique for FL is included. The list of PET techniques comprises technologies like Homomorphic Encryption, Secure Multi-Party Computation, Differential Privacy, Private Coded Distributed Computing and Zero-Knowledge Proofs. In addition to the analysis of trade-offs and privacy guarantees provided by each PET technique, the document identifies a list of software tools and libraries for the implementation of each PET method.

Finally, the last part of the document reports the possibility of working with a combination of several of the studied PET technologies. Each technique is classified according to the guaranteed privacy property, and it is discussed how a higher privacy protection can be obtained by combining at least two complementary techniques. The document does not describe all possible combinations, but focuses instead on some combinations which are of particular interest for TRUMPET.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

DOCUMENT INFORMATION

Grant agreement No.	101070038	**Acronym**	TRUMPET
Full title	**TRUstworthy Multi-site Privacy Enhancing Technologies**		
Call	HORIZON-CL3-2021-CS-01-04		
Project URL	https://cordis.europa.eu/project/id/101070038		
EU project officer	Fidel SANTIAGO		

Deliverable	Number	D2.1	Title	SoA and BSoA for FL-tailored AI models, PET methods and tools
Work package	Number	WP2	Title	Privacy-armored AI algorithms and models
Tasks	Number	T2.1 T2.2	Title	Identification of AI models and aggregation methods for Federated Learning Research of FL-tailored PET methods and tools

Date of delivery	Contractual	M12	Actual	M12
Status	version 0.5 ☒ Final version			
Nature	☒ R ☐ DEM ☐ DMP ☐ DEC ☐ ETHICS ☐ OTHER			
Dissemination level	☒ Public ☐ Sensitive			

Authors (partners)	UVIGO, INRIA, GRAD, TVS
Responsible author	Alberto Pedrouzo-Ulloa (UVIGO)
	apedrouzo@gts.uvigo.es

Summary (for dissemination)	*SoA and BSoA for FL-tailored AI models, PET methods and tools contains the results of T2.1 and T2.2. The deliverable contains a selection of AI models and central aggregation methods for the FL setting, as well as an in-depth analysis of the state of the art and improvements of the PET methods and SW tools for FL scenario*
Keywords	*Federated Learning, AI models, PET methods and tools*

VERSION LONG			
Issue Date	**Rev. No.**	**Author**	**Change**
27/01/2023	0.0	Alberto Pedrouzo-Ulloa – Jaime Loureiro	Table of contents
31/05/2023	0.1	Jaime Loureiro – Xavier Martínez – Carlos García-Pagán – Inés Ortega – Gonzálo Jiménez-Balsa – Eva Sotos Martínez – Jan Ramon – Twaha Mukammel	Final T2.1 inputs + intermediate T2.2 inputs of individual PET components
04/09/2023	0.2	Jan Ramon – Aleksei Korneev – Xavier Martínez – Jaime Loureiro – Fernando Pérez-González – Alberto Pedrouzo-Ulloa	Final inputs of combined PET methods + framework and software tools
11/09/2023	0.3	Alberto Pedrouzo-Ulloa	1st complete version of D2.1
22/09/2023	0.4	Jaime Loureiro – Twaha Mukammel - Jan Ramon	D2.1 review
30/09/2023	0.5	Alberto Pedrouzo-Ulloa	Final revision

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

TABLE OF CONTENTS

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

LIST OF FIGURES

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

LIST OF TABLES

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

ABBREVIATIONS AND ACRONYMS

AFL: Armored Federated Learning
AI: Artificial Intelligence
BACC: Berrut Approximated Coded Computing
BFV: Brakerski-Fan-Vercauteren scheme
BGV: Brakerski-Gentry-Vaikuntanathan scheme
CFL: Coded Federated Learning
CKKS: Cheon-Kim-Kim-Song scheme
CNN: Convolutional Neural Network
CRS: Common Reference String
CSA: Cross-Subspace Alignment
DNN: Deep Neural Network
DP: Differential Privacy
FHE: Fully Homomorphic Encryption
FL: Federated Learning
GC: Garbled Circuits
GCSA: Generalized Cross-Subspace Alignment
GDP: Gaussian Differential Privacy
GSW: Gentry-Sahai-Waters scheme
HE: Homomorphic Encryption
LCC: Lagrange Coded Computing
LDA: Linear Discriminant Analysis
LDP: Local Differential Privacy
LWE: Learning with Errors
MA: Moments Accountant
MIMO: Multiple-Input Multiple-Output
MKHE: Multi-Key Homomorphic Encryption
ML: Machine Learning
MSE: Mean Squared Error
NbAFL: Noise before model Aggregation Federated Learning
NN: Neural Network
NTT: Number Theoretic Transform
PCA: Principal Component Analysis
PET: Privacy Enhancing Technologies
PLD: Privacy Loss Distribution
PPFL: Privacy-Preserving Federated Learning
RDP: Rényi Differential Privacy
RLWE: Ring Learning with Errors
RNS: Residue Number System
ROC: Receiver Operator Characteristic

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

SGD:	Stochastic Gradient Descent
SHE:	Somewhat Homomorphic Encryption
SMPC:	Secure Multi-Party Computation
SVM:	Support Vector Machine
SVT:	Sparse Vector Technique
ThHE:	Threshold Homomorphic Encryption
ZKP:	Zero Knowledge Proof

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

1 Introduction

This document reports the results corresponding to the work realized inside the scope of tasks T2.1 and T2.2 from the WP2 of TRUMPET “Privacy-armored AI algorithms and models.” Namely, this work package is in charge of doing research on and developing the FL-tailored AI models and algorithms with privacy guarantees, which will be integrated and validated later in the TRUMPET platform (WP4).

As we can see in Figure 1, this work of WP2 in private-armored FL algorithms has a high dependency on the elicited requirements from WP1 (see next Subsection 1.1) and, similarly to WP3, produces direct outputs for the correct development, piloting and validation of the TRUMPET platform, which will be reported in the corresponding deliverables of WP4.

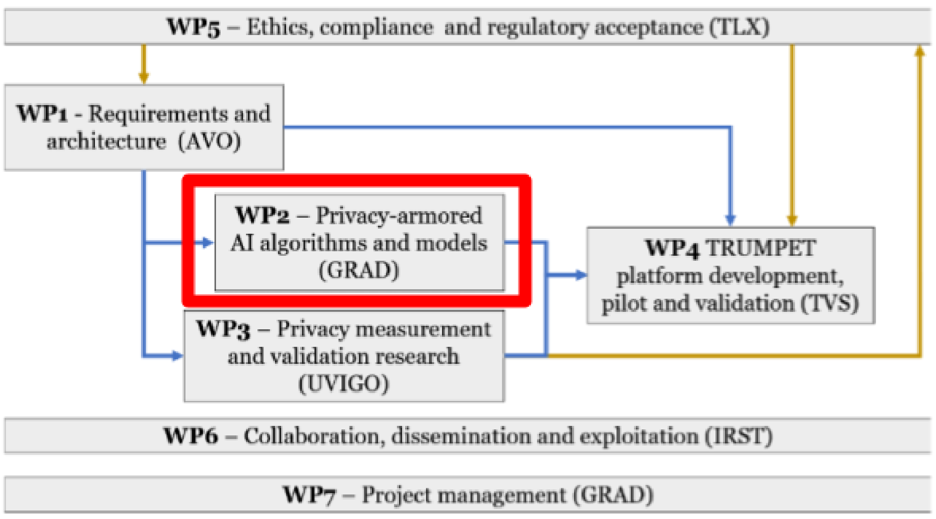
WP5 – Ethics, compliance and regulatory acceptance (TLX)
WP1 - Requirements and architecture (AVO)
WP2 – Privacy-armored AI algorithms and models (GRAD)
WP4 TRUMPET platform development, pilot and validation (TVS)
WP3 – Privacy measurement and validation research (UVIGO)
WP6 – Collaboration, dissemination and exploitation (IRST)
WP7 – Project management (GRAD)

Figure 1 – WP2 and its relationship with other WPs of TRUMPET.

More specifically, and among the initial objectives established for WP2, this document aims at fulfilling the first two WP2 general objectives:

- Survey the state of the art; identify and evaluate AI models to be used in the FL setting.
- Advance the knowledge, expertise and application of different non-encryption/encryption-based PET methods to be included in the armored FL setting.

Consequently, the work produced in this deliverable feeds the privacy-preserving FL algorithms and models (see Figure 2) to the other WP2 tasks T2.3, T2.4 and T2.5, which are instead more focused on their design, development, test and final validation.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

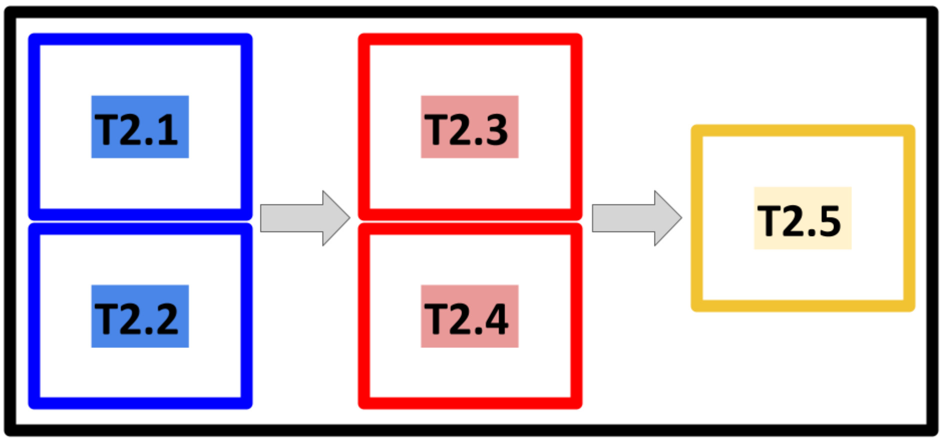

Figure 2 – Dependency between the tasks of WP2.

In summary, the results of this deliverable D2.1 can be divided in two main different types of outcomes:

- Outcomes from Task T2.1 "Identification of AI (Artificial Intelligence) models and aggregation methods for Federated Learning (FL)":
    - A survey of the state of the art in algorithms and tools for Federated Learning.
    - Identify and provide a list of the most adequate AI models and algorithms to be applied on the TRUMPET platform and medical use cases.
- Outcomes from Task T2.2 "Research of FL-tailored PET methods and tools":
    - Survey of the state of the art in PETs (Privacy Enhancing Technologies).
    - Research their optimal application into the FL pipeline of the TRUMPET platform.

1.1 Starting point and structure of the deliverable

In relation to the T2.1 outputs, a set of AI models and algorithms relevant for the TRUMPET use cases is already very succinctly described in D1.1-Sec.3. Also, D1.1-Sec.7 gives a first description of the different use cases together with their datasets.

Section 2 of the present document extends the work carried out in D1.1 and D1.3 by surveying the most relevant ML (Machine Learning) algorithms in FL for TRUMPET purposes. In particular, a detailed list of AI models and aggregation methods is presented, and a selection of the best candidates is made considering the requirements imposed by the TRUMPET use cases. This selection also includes a review of the most relevant FL frameworks and their adequacy to the needs of TRUMPET use cases.

In relation to the T2.2 outputs, D1.1-Sec.6 sketches the first requirements and some important general properties of PETs for TRUMPET. Also, D1.1-Sec.5 sets forth the importance of providing protection against several types of inference attacks in FL.

Section 3 of the present document gives more details and contains a description of the state of the art of several modern PET methods which are relevant for the FL scenario. First, a study of the advantages and shortcomings of each individual PET technique for FL is included. The list of PET

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

techniques comprises technologies like Homomorphic Encryption, Secure Multi-Party Computation, Differential Privacy, Private Coded Distributed Computing and Zero-Knowledge Proofs. In addition to the analysis of trade-offs and privacy guarantees provided by each PET technique, the document identifies a complete list of software tools and libraries for the implementation of each PET method.

Finally, **Section 4** of the document reports the possibility of working with a combination of several of the studied PET technologies. Each technique is classified according to the guaranteed privacy property, and it is discussed how a higher privacy protection can be obtained by combining at least two complementary techniques. The document does not describe all possible combinations but focuses instead on some combinations which are of particular interest for TRUMPET.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

2 A review of the state of the art and beyond in FL

The term Federated Learning (FL) may have somewhat different meanings in the literature corresponding to different areas. At times, it predominantly serves as a synonym for various forms of distributed machine learning, while in other instances, it is closely linked to tasks specifically related to neural networks. Other times, it is associated with forms of distributed machine learning that involve a central coordinator. In the context of TRUMPET, we will use FL to refer to any machine learning using data from multiple data owners and involving a central coordinator.

In this project [PRP+23], we are interested in settings where each of several data owners has a database of sensitive data, and they want to collaborate to perform data analysis without revealing their data to others. This setting is called privacy-preserving federated learning (PPFL), but whenever it is clear from the context, we may just call it FL.

It is helpful to distinguish the data user (sometimes called the “researcher” in TRUMPET) which submits queries to the system, from the coordinator of the computation (also sometimes called the “aggregator”), even if sometimes the same node in the network may fulfil several roles.

The objective of an FL system is the same as the objective of the original ML system that it tries to improve, and which would build the same model on data that is centralized on a single machine. While the objectives are the same, the FL approach poses new challenges.

An FL system makes use of several elements:

- There is the “data owner” entity, which locally computes statistics or trains a partial model on its own data.

- There is the “communication” layer, where the several data owners and the coordinator exchange information, e.g., to aggregate statistics over all data owners.

- There is the “coordinator” entity, where the coordinator decides upon the next actions, and aggregates all information into the final machine learning model.

There are situations where the structure is more complex, e.g., when there are so many data owners that it becomes impractical to let them operate in an unstructured group, or when for cryptographic reasons an external trusted party is needed, but unless explicitly mentioned otherwise we will not need or consider these in TRUMPET [PRP+23].

The way in which these three components of an FL system interact strongly depends on the ML algorithm at hand. We can divide ML algorithms into the following categories:

- *ML algorithms which can be described in terms of averages.* These are algorithms where for the coordinator it is sufficient to know a set of averages (also called *U*-statistics with kernel of degree *1*) [Lee90, CCB16]. Examples of ML models for which such algorithms are available include linear regression, decision trees, Bayesian networks and Gaussian processes. For such algorithms, it is sufficient for every data owner to compute locally a set of statistics on its data, after which the statistics can be averaged over all data owners and the coordinator can construct the ML model from the obtained statistics.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- *ML algorithms which first establish an objective function and then use an optimization procedure to find the optimal values of a set of parameters of that objective function.* Examples of ML models for which such algorithms are available include logistic regression and neural networks [HTF09].

- *ML algorithms which require pairwise access to training instances.* This includes algorithms which can be expressed in terms of *U*-statistics with kernel of degree *2*, e.g., the computation of receiver operator characteristic (ROC) curves and the computation of the Kendall tau statistic [CCB16, CBSC15]. Other algorithms which may need access to pairs of training instances are kernel-based machine learning algorithms in dual space, e.g., certain support vector machine algorithms. If the number of data owners is high, iterating over all pairs of training instances may be expensive, so it is typically better to avoid this, e.g., reverting to approximation algorithms.

- *Others.* There are other ML tasks not fitting in the above categories, e.g., prediction where one party (or collection of parties) has a model and another party an instance on which the model should be applied to obtain a prediction.

In Sections 2.1 and 2.2 we will discuss the first major types of federated machine learning tasks.

2.1 Aggregation

In the task of federated aggregation, all data owners have some data (a scalar, vector, matrix or more compound object) and the goal is to ensure that a data user receives the aggregate (typically sum or average) of these data.

This is a fundamental operation in FL, both for algorithms which can be decomposed directly into averages and for more complex algorithms such as stochastic gradient descent (SGD) algorithms which frequently need to add up gradients computed by data owners.

If there is a trusted central curator, the data owners can simply send their gradients to that trusted party. If data owners have no trust at all, they can add local differential privacy (LDP) noise [DJ13] to their data, but as LDP typically requires a large amount of noise, this usually is not desirable from a utility point of view.

If data owners cannot trust the aggregator, then sending cleartext data without noise is not advisable, even if the data is constituted only of gradients rather than the originally sensitive information [MSCS19, NSH19].

There has been a lot of interest in strategies which do not require to fully trust parties, resulting in several approaches for different security and threat models. One line of work employs a trusted shuffler to create anonymity, which can be used to obtain increased privacy [EFMRT19, GKMP20]. While interesting for theoretical analysis, this approach only replaces the need to trust an aggregator by the need to trust a shuffler, or a shuffling system consisting of several nodes. Another line of work studies secure aggregation methods relying on cryptographic techniques [DKMMN06, CSS12, SCRCS11, BIKMMPRSS17, JWEG18]. All these strategies assume an honest-but-curious (also called semi-honest) security model and/or have a significant computational cost. Recently, more efficient approaches have been studied which at the same time are robust under more malicious models, e.g., when one assumes at most a fixed fraction of the parties is malicious [SBR22].

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Hence secure aggregation methods exist which, depending on the threat model, can deliver the aggregation of the data owners' data without revealing the individual terms or intermediate results. Sometimes it is undesirable to reveal this aggregate immediately, as in order to achieve statistical privacy (e.g., differential privacy) one may want to add noise first. One challenge is that none of the parties is allowed to know the noise term, as otherwise this party could act as adversary and subtract it from the published result. Fortunately, various strategies exist to generate and add noise privately [SPHR23].

2.2 Federated Optimization

A number of machine learning models, such as logistic regression, neural networks and clustering models cannot be computed in closed form and are usually trained using iterative algorithms. As (deep) neural networks are very popular currently, federated versions of methods such as stochastic gradient descent (SGD) received a lot of attention recently; while our discussion will focus mostly on these methods which can be directly applied to neural networks and logistic regression, very similar strategies can be applied to clustering models.

The most privacy-preserving approach to federated optimization is to encrypt the complete set of computations over all iterations of the algorithm. This is feasible, but rather computationally expensive, especially for very large datasets. The noise needed to be added to the resulting model is then the same as in the trusted curator setting where a trusted party obtains all data, learns the model centrally and privatizes it before publishing [CMS11, KST12].

Sometimes it is undesirable to opt for the trusted curator or complete encryption of the algorithm execution. In such cases, a popular strategy is to securely compute a gradient based on all data and to privatize this gradient before it is received by the aggregator who will apply it to the model. To aggregate securely, techniques such as those described in Section 2.1 can be used. Once privatized gradients can be computed as a subroutine, high-level SGD algorithms mainly need to keep track of the privacy budget, either using Dwork's composition rule [BST14] or more efficiently using Rényi differential privacy [ACGMMTZ16]. Several optimizations are possible on this basic idea to improve the privacy-utility trade-off, e.g., by clipping the gradient norm [MMRHA17, ATMR22] or by adapting the learning rate to the privacy noise properties [KH20, LK18, YLPET19].

Similar approaches have been studied when considering coordinate descent [MBST22] rather than gradient descent, one advantage of federated coordinate descent compared to gradient descent is that the communication cost per iteration is clearly smaller.

2.3 Data Owner side computation

Many machine learning algorithms require computation on the side of the data owner, which often involves both querying the local data and combining it into a local feature or statistic. Examples include the aggregation over the local dataset, or the computation of a gradient of a given model with respect to the local dataset (as is needed in SGD algorithms). This data owner side computation is usually a subroutine of a more high-level federated algorithm such as an aggregation (See Section 2.1) or federated optimization (See Section 2.2).

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

In the semi-honest model, local computation inside the data owner does not pose any risk, as we assume the data owner performs the computations correctly and no information leaves its premises.

In more malicious models, one a priori does not trust the data owner to perform these local computations correctly, and one needs to consider the possible attacks of (a) using incorrect data or (b) making incorrect computations. Both attacks have a similar effect and could be called data or model poisoning. To mitigate these attacks, one can require that the data owner proves that both data and computations are correct.

Proving that computations are correct without disclosing the underlying data is possible using zero knowledge proofs (ZKP).

As the data usually is private, verifying that the data is correct is usually more difficult. Still, a few partial solutions exist. First, one can require ZKP proving that the data is in the appropriate domain, e.g., a range proof to prove that input data is in a specified interval [MKWK19]. Second, one can compute the distribution of values over many records or data owners and compare whether this distribution is significantly different from what can be expected for such values [SPHR23]. This allows to detect outliers or outlier data owners but cannot be used to detect incorrect individual records as exceptional records do exist. Third, one can require that data owners commit to input data. While this has no immediate effect in a computation, it implies, in a next computation using the same basic input data, that the same value must be used. Such consistency requirement helps to limit the opportunity for data poisoning.

2.4 Frameworks, tools and software for Cross-Silo FL

Several state-of-the-art frameworks and tools for FL are listed below. They are divided in two categories, Research-oriented software and Production-oriented software.

2.4.1 Research-oriented software

- **TensorFlow Federated** [TensorFlow19] specifically targets research use cases, providing large-scale simulation capabilities as well as flexible orchestration for the control of sampling.

- **FedML** [HLS+20] is a research-oriented library. It supports three platforms: on-device training for IoT and mobile devices, distributed computing, and single-machine simulation. For research diversity, FedML also supports various algorithms (e.g., decentralized learning, vertical FL, and split learning), models, and datasets.

- **PySyft** [RTD+18] is a Python library for secure, private Deep Learning. PySyft decouples private data from model training, by using federated learning, differential privacy, and secure multi-party computation (SMPC) within PyTorch. PySyft aims to be extensible such that new FL, Secure Multi-Party Computation, Homomorphic Encryption, or Differential Privacy methods can be flexibly and simply implemented and integrated.

- **Sherpa.ai** [RSJ+20] Federated Learning and Differential Privacy Framework is an open-source federated learning and differential privacy framework which provides methodologies, pipelines, and evaluation techniques for federated learning.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- **PyVertical** [RHP+21] is a project focusing on federated learning with data partitioned by features (also referred to as vertical partitioning) in the cross-silo setting.
- **LEAF** [CWL+18] is a modular benchmarking framework for learning in federated settings. It includes a suite of open-source federated datasets, a rigorous evaluation framework and a set of reference implementations.
- **DecLearn** [Declearn22] is a modular open source, INRIA-developed library, currently mostly focused on stochastic gradient descent (SGD) and coordinate-descent algorithms and their variants in a federated setting (one data user/researcher node, many data owner nodes).

2.4.2 Production-oriented software

- **FATE** [FATE19] (Federated AI Technology Enabler) is an open-source project intended to provide a secure computing framework to support the federated AI ecosystem.
- **PaddleFL** [PaddleFL19] is an open-source federated learning framework based on PaddlePaddle. In PaddleFL, several federated learning strategies and training strategies are provided with application demonstrations.
- **Clara** [NVIDIAClara19] Training Framework includes the support of cross-silo federated learning based on a server-client approach with data privacy protection.
- **IBM Federated Learning** [LBT+20] is a Python-based federated learning framework for enterprise environments, which provides a basic fabric for adding advanced features.
- **Flower** framework [BTM+20] supports implementation and experimentation of federated learning algorithms on mobile and embedded devices with a real-world system conditions simulation.
- **Fedlearner** [Fedlearner20] is an open-source federated learning framework that enables joint modelling of data distributed between institutions.
- **OpenFL** [RGF+21] is a Python 3 framework for Federated Learning. OpenFL is designed to be a flexible, extensible and easily learnable tool for data scientists.
- **SubstraFL** [SubstraFL23] is a federated learning Python library that leverages Substra to run federated learning experiments at scale on real distributed data. Its main usage is therefore in production environments.
- **FedBiomed** [CVC+23] is a federated system mostly using existing techniques such as libraries like DecLearn listed above, for use in medical applications.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

2.4.3 Which frameworks suit the needs of TRUMPET?

After studying the above-mentioned FL software tools, PySyft and FATE seem to be the FL tools which better suit the needs of TRUMPET [PRP+23]. Below we outline our findings with a comparison between both and the justification for our preference for PySyft.

2.4.3.1 PySyft

- Pros:
    - Modularity and Flexibility: PySyft is designed to integrate with popular deep learning libraries like PyTorch, enabling users to leverage its features within their existing workflow.
    - Research Focus: PySyft is developed with an emphasis on academic and research-oriented applications, making it suitable for exploring novel privacy-preserving techniques.
    - Extensive Library of Privacy Tools: PySyft provides a wide range of privacy-enhancing tools, including federated learning, secure multi-party computation, and differential privacy mechanisms.
    - Community Support: PySyft is backed by the OpenMined community, which consists of researchers, developers, and practitioners interested in advancing privacy-preserving AI.
    - System Design: The design of the PySyft open-source code base is well structured and developed on segregated interfaces, which makes easier to implement and incorporate new PET methods. Also, the system architecture is built to enable that different service components can be used or not based on the specific needs.
- Cons:
    - Learning Curve: PySyft's flexibility might lead to a steeper learning curve for newcomers, especially those who are not familiar with the underlying concepts of privacy-preserving techniques.
    - Limited Industrial Focus: While PySyft can certainly be used for industrial applications, its primary focus has been on research and exploration, which might result in fewer pre-built solutions tailored for specific industries.

2.4.3.2 FATE

- Pros:
    - Industrial Applications: FATE is explicitly designed to cater to real-world industrial use cases, making it a suitable choice for enterprises looking to implement privacy-preserving AI solutions.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- Multi-Language and Multi-Framework Support: FATE supports multiple programming languages and popular deep learning frameworks, offering greater flexibility for different development environments.
- Ready-Made Algorithms: FATE offers a range of pre-implemented federated learning algorithms and privacy-preserving techniques, saving time and effort in algorithm development.

- Cons:
  - Less Flexibility: FATE's focus on providing a comprehensive platform might lead to less flexibility compared to PySyft, particularly for researchers and developers who want to experiment with custom privacy techniques.
  - Complexity for Beginners: FATE's extensive features and algorithms could be overwhelming for beginners or those who are new to the field of federated learning. In addition, the documentation for the most recent functionalities remains insufficient.

2.4.3.3 **Comparison: PySyft vs FATE**

- Use Case Focus:
  - PySyft: Primarily suited for research and experimentation, making it attractive to academia and projects exploring cutting-edge privacy methods.
  - FATE: Geared towards industrial applications and practical implementations, particularly in industries like finance.
- Flexibility vs. Comprehensive Platform:
  - PySyft offers high flexibility for customised privacy solutions but might require more expertise and time for implementation.
  - FATE provides a comprehensive platform with a wide range of pre-built algorithms and techniques, suitable for faster deployment but with potentially less room for customization.
- Community and Support:
  - PySyft is backed by the OpenMined community, known for its active involvement in privacy-preserving AI research.
  - FATE is supported by WeBank and has lesser community support.
- Learning Curve:
  - PySyft might have a steeper learning curve due to its customizable nature and the limited documentation which is available for the most recent versions.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- FATE's focus on providing ready-made solutions might make it more approachable for those new to the field.

Adequacy of Pysyft: PySyft is a modular federated learning (FL) framework that exhibits a high degree of modularity, offering various modules that can be combined and extended to build customized FL solutions. This modular design enhances flexibility, scalability, and ease of integration. Next, we delve into this justification with an example:

Example for Secure Aggregation in PySyft: Consider a scenario where a healthcare organization wants to perform federated learning on medical data distributed across different hospitals. They are concerned about data privacy and security. PySyft's modularity comes into play:

1. Secure Aggregation Module: PySyft provides a module specifically designed for secure aggregation. The secure aggregation module takes care of basic cryptographic protocols and privacy-preserving mechanisms.

2. Data Privacy Module: In the healthcare scenario, data privacy is paramount. PySyft offers basic privacy-preserving modules like Differential Privacy to ensure that sensitive patient data is not compromised during the training process.

3. Custom Communication Protocols: Hospitals might have specific communication requirements based on their network infrastructure. PySyft's modular communication protocols allow organizations to choose the most suitable protocol for their setup, whether it is WebSockets, HTTP, or others.

4. Flexible Model Architectures: PySyft's modular approach extends to model architectures as well. Hospitals can implement custom model architectures using PyTorch, and then integrate them seamlessly with PySyft's FL framework, incorporating differential privacy or secure aggregation as needed.

5. Community Contributions: The modular design of PySyft encourages the community to contribute additional modules. Suppose a hospital requires a specialized privacy-preserving technique for its use case. They can collaborate with the PySyft community to develop and integrate a new module that caters to their requirements.

6. Scalability and Parallelism: With PySyft's modular architecture, organizations can distribute the training workload across multiple nodes. They can leverage PySyft's built-in support for parallel execution to achieve faster training times without compromising privacy.

2.5 Selection of AI algorithms and models

Based on the above literature study and the requirements reported in D1.1/D1.3, we selected which models and algorithms will be considered in our research (WP2) and our platform (WP4) through the work in T2.1.

As a reminder, D1.1 concluded that the functionalities required for the TRUMPET use cases [PRP+23] could include computation of statistics, linear regression, logistic regression, decision trees, random forests, survival analysis (e.g., Cox regression), feature selection, dimensionality reduction (e.g., clustering) and validation.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

We can categorize the algorithms needed or typically used for these tasks into a number of major categories:

- Methods which can be decomposed into the computation of *U*-statistics (which is possible using direct secure aggregation): computation of statistics, linear regression (on small domains), decision trees and random forests.
- Iterative optimization strategies, often based on stochastic gradient descent (SGD), which are popular to address linear regression (on high dimensional data), logistic regression, Cox regression, and clustering for dimensionality reduction. Of these, only the latter is not gradient based, and only Cox regression requires a comparison operator as part of the core calculation (which may affect the choice of the PET strategy, see Section 3).
- Wrapper strategies, such as for cross-validation, hyperparameter search or similar purposes.
- Specific (other) algorithms: randomization-based validation, and optionally, principal components analysis (PCA).

Next to these models/algorithms required for the TRUMPET use cases, it is useful to support in general other popular methods, as this will be beneficial for the exploitation of the platform. Therefore, we decided to also support (simple) neural networks. The additional effort needed is relatively moderate, as SGD is the most popular algorithm for training neural nets. This also gives the additional option to support with some additional effort a (variational) auto-encoder based dimensionality reduction, offering an alternative to the PCA and clustering algorithms mentioned above.

To determine which exact variant of these models/algorithms is best suited for the use cases (and hence are minimally required), pre-pilot studies on public data or studies on patient data approved by ethical committees will be carried out, so the right variants can be implemented by the time the federated learning platform will be validated.

Currently, we expect the following models and algorithms to be important for the TRUMPET platform, even though further fine tuning may be needed depending on the results of further steps:

2.5.1 Relevant AI models

- Linear regression (including iterative algorithms):
  - Model: A linear model between features and target value is of the form $\boldsymbol{y} = \boldsymbol{w}^T\boldsymbol{x}$ with $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \mathbb{R}$ where $\boldsymbol{w}$ is the column vector of model parameters. A common strategy is to find the $\boldsymbol{w}$ which minimizes $\sum_i(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \lambda||\boldsymbol{w}||^2$ where λ is a regularization (hyper)parameter.
  - Algorithms:
    - If the *U*-statistics $\boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{X}^T\boldsymbol{Y}$ are feasible to be computed (where $\boldsymbol{X}^T = [\mathbf{x}_1 \ldots \mathbf{x}_n]$ and $\boldsymbol{Y}^T = [y_1 \ldots y_n]$), i.e., when the dimension of $\boldsymbol{X}^T\boldsymbol{X}$ is not too large, then one can compute the optimal $\boldsymbol{w}$ in closed form: $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$.
    - If the dimension is too large to compute/store the correlation matrix $\boldsymbol{X}^T\boldsymbol{X}$, then one can follow an iterative approach, e.g., using stochastic gradient descent (SGD).
- Logistic regression:
  - Model: A logistic regression model is of the form $y = S(\boldsymbol{\theta}^T\boldsymbol{x})$ where $\boldsymbol{\theta}$ is the parameter vector and $S(t) = 1/(1 + e^{-t})$ is the logistic function.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

   - Algorithms: In contrast to linear regression, a closed form solution is not possible here, so one normally uses an iterative algorithm, e.g., SGD.
- Random forests (including decision trees):
   - Model: A random forest is an ensemble of decision trees (which are learnt with some randomized elements). A decision tree is either a predicted target value or an “IF *test* THEN t_1 ELSE t_2” structure, where *test* is a binary function on instances, and t_1 and t_2 are decision trees.
   - Algorithms: random forests are typically learnt by iteratively learning decision trees and then building a model which averages their predictions. A decision tree is typically learnt top-down, where in each iteration one decides whether a sufficiently interesting test can be found, if not one generates a leaf (a fixed prediction); else, one selects a test (according to a heuristic and some randomization) and then sorts all instances into a subset on which the test returns and a set on which the test returns, upon which one recursively learns decision trees on the subsets.
- Survival analysis:
   - Model: In survival analysis models are also considered, e.g., linear or logistic regression, but data is censored in some ways. The classic example is a medical study where patients may leave the study before an event of interest occur, so one only knows that the event happened after a given time (or did not happen at all). A popular model is Cox regression.
   - Algorithm: the best training strategy depends largely on properties of the application at hand. At the time of writing, we are in the context of (pre-)pilot studies still examining which training strategies will be the most useful to implement.
- Neural networks:
   - Model: Neural networks are networks of neurons connected by links. The input is provided to the input nodes which propagate the value via their outgoing links; intermediate nodes that received values from all their incoming neighbors compute their output and send it to their outgoing neighbors until all output nodes have computed their values. Neural networks can also be potentially used in a survival analysis context.
   - Algorithms: SGD is a popular strategy to train neural networks. Dedicated algorithmic strategies exist for survival analysis.
- Dimensionality reduction:
   - Model: Dimensionality reduction offers transformations from high dimensional feature spaces into lower dimensional spaces. This is typically useful when the number of features is much larger than the number of instances, e.g., when using genomic data.
   - Algorithms: Several options can be considered, e.g., principal component analysis (PCA), linear discriminant analysis (LDA), or even auto-encoder based strategies. At the time of writing, it is still unclear which strategies are best suited for the TRUMPET use cases.
- Variational auto-encoders:
   - Model: auto-encoders are models that first map instances into vectors in a lower dimensional space and then map them back into the original instance space, i.e., they are models of the form $f = D \circ E$ where $E: \mathbb{R}^n \to \mathbb{R}^m$ (with $m < n$, where $\mathbb{R}^m$ is called the latent space) and $D: \mathbb{R}^m \to \mathbb{R}^n$, where the objective is that instances of the dataset are mapped by f on nearby instances. Variational auto-encoders follow a probabilistic approach, where one is looking for a model that maximizes the likelihood of the given dataset being generated when transforming a sample drawn from the prior on the latent space. While variational auto-encoders can be used as generative models and often provide better results for tasks as synthetic data generation, in the TRUMPET

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

pilots we at most need decent dimensionality reduction of genomic data, and hence we will need to investigate which model is both sufficiently simple (not over-fitting on our small dataset) and sufficiently effective.
  - Algorithms: auto-encoders and related models are normally modelled and trained as neural networks, e.g., using SGD style algorithms.

2.5.2 Relevant FL algorithms

- Secure aggregation:
  - Section 2.1 provides a discussion of (secure) aggregation algorithms. As secure aggregation is a basic building block, we will use multiple existing implementations depending on the threat model, and implement new ones where appropriate, especially to improve robustness against inference attacks by colluding adversaries. A further extension seems needed for survival analysis where secure sums over pairs of instances are required.
- SGD:
  - We discussed SGD already in Section 2.2. We expect to make further improvements to the state of the art, especially in terms of statistical privacy, both to improve the utility for a given privacy budget and to consider new privacy metrics developed in WP3.
- Wrapper algorithms (e.g., validation):
  - To provide a complete researcher dashboard, it is important that the researcher can perform more than stand-alone algorithms, as they are often described in the literature. For example, algorithms like cross-validation and other processes, which are essentially repeated executions of baseline machine learning algorithms.
- Comparison operator for use in core computation of models (e.g., survival analysis).

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

3 Privacy Enhancing Technologies for Armored FL

In the TRUMPET project [PRP+23] we will apply several Privacy Enhancing Technologies to the pipeline of FL so as to construct an Armored FL Framework with better privacy guarantees. These techniques can be applied either individually (see Section 3.1) or can be combined (see Section 4) to get a better trade-off between privacy and accuracy. This section focuses on the former, by describing in detail each of the individual PET techniques. We also include a list with relevant available software tools and libraries for each discussed technique (see Section 3.2).

3.1 Study of stand-alone PET components for AFL

We review next the current state of the art of several types of PET technologies. They will be used in TRUMPET to strengthen the baseline privacy provided by the FL algorithms presented in Section 2, namely Homomorphic Encryption, Secure Multi-Party Computation, Code-based Encryption and Differential Privacy.

3.1.1 Homomorphic Encryption

Homomorphic Encryption (HE) deals with the problem of computing with encrypted data. A HE scheme basically consists of an encryption scheme with the peculiarity that some sort of computation can be directly performed over the ciphertext. HE schemes allowing to homomorphically compute only one type of arithmetic operation (addition or multiplication) were known since 1970 [RAD78], and in fact, schemes like Paillier [Paillier99] or ElGamal [ElGamal85] were widely used for that reason. The breakthrough came in 2009, when Gentry [Gentry09] showed that it was possible to construct a HE scheme where both additions and multiplications could be computed under encrypted data. Although the first proposed schemes were very far from being practical, the field evolved very rapidly, and subsequent schemes became faster at an ever-increasing rate.

While FHE (Fully Homomorphic Encryption) schemes allowing for arbitrary computations are perfectly feasible, they are not practical enough yet. So, it is common practice to relax conditions and work instead with SHE (Somewhat Homomorphic Encryption) schemes, on which the number of operations to be evaluated is bounded beforehand.

It is worth mentioning that current schemes are based on the LWE (Learning with Errors) [Regev09] and RLWE (Ring Learning with Errors) [LPR13] assumptions. So, as a byproduct of being based on Lattice-based problems, they are known to be resistant against quantum computers.

Consequently, HE methods provide a very high level of privacy because the most privacy-sensitive information can be encrypted and, if further operations are needed, they can be directly done while the data is still encrypted. Additionally, computation can be performed, in theory, without having any accuracy loss on the results.

On the contrary, HE methods also have some limitations:

- Loss on precision: in many scenarios the computational cost could be very high, so it could be better to reduce the accuracy of the results as a trade-off to improve the efficiency.

- Message expansion: The size of the encrypted data can be much larger than the size of the data in the clear.

- Engineering cost: Taking full advantage of the HE features can be very challenging [SKN+20]. In order to really enhance performance, it is usually required to make non-trivial changes on the data pipelines and algorithms, hence requiring the help of a HE expert. For example, current implementations use very refined representations by combining the RNS (Residue Number System) with NTTs (Number Theoretic Transforms) [PTP17, BPNB23].

Even so, HE is starting to become very practical and there are important efforts to bring HE into real applications. A clear example is the recently started process for the standardization of homomorphic encryption primitives [ACC+19].

The standard covers important aspects of HE, ranging from concrete estimates of the bit security [APS15] together with parameter recommendations, the design of an API to easily develop HE-based applications and discussions regarding several potential application fields. The main HE schemes described are BGV [BGV12], BFV [Brakerski12, FV12], CKKS [CKKS17] and TFHE [CGGI16], where there is no definitive best choice and performance of each one is very dependent on the particular scenario.

There is also a wide list of available HE libraries implementing the previous schemes and some more advanced functionalities. Some of the most important libraries are Microsoft SEAL [SEAL], HElib [HElib], HEAAN [HEAAN], PALISADE [PALISADE], Lattigo [Lattigo], NFLlib [FV-NFLlib], TFHE [TFHE], Concrete [Concrete], OpenFHE [OpenFHE], THFE-RS [TFHE-rs]. We can also find GPU-based implementations as cuHE [cuHE] and a high-level compiler CHET [DSC+19].

Regarding the use of HE to implement ML or FL applications [GLN12], we can already find in the literature several works dealing with important ML tools applicable to the different scenarios. Some examples include logistic regressions [WZD+16, KSKLC18, CGGT19], inference with neural networks as CryptotNets [GDLLNW16], and more complex training phases as the case of a SVM classifier [PBLCL20].

3.1.1.1 Basic HE building blocks

Here we introduce a **high-level primitive description of a common HE scheme.** We also include some extra procedures and functions which are used in most of the lattice-based constructions. Thanks to this, our exposition for the multi-key variants of HE will be simplified in Section 4.1, and there we will only remark which are the main changes needed when working with several keys.

Learning with Errors (LWE): The LWE distribution corresponds to ($\boldsymbol{a}$, b = <$\boldsymbol{a}$, $\boldsymbol{s}$> + e), where $\boldsymbol{a}$ is vector of size n, uniformly random in the integers modulo q, $\boldsymbol{s}$ is the secret and e is sampled from the error distribution. See [Regev09] for more details.

Ring Learning with Errors (RLWE): The RLWE distribution corresponds to (a, $b = a\, s + e$), where a is a uniformly random polynomial in R_q, s is the secret and e is a polynomial sampled from the error distribution. See [LPR10, LPR13] for more details.

General HE primitives: A HE scheme is composed of a tuple of algorithms HE = (KeyGen, Enc, Eval, Dec), where *lambda* is the security parameter.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- HE.**KeyGen**(1^*lambda*) → (*pk*, *sk*, *ek*)
- HE.**Enc**(*pk*, *m*) → *c*
- HE.**Eval**(*ek*, *f*, *c*, *c'*) → c_{eval} , where $c_{eval} = f(c, c')$
- HE.**Dec**(*sk*, *c*) → *m*

Table 1 – Notation used for HE

HE NOTATION	
Initials or symbol	**Meaning**
pk	Public Key
sk	Secret Key
ek	Evaluation Key
pp	Public Parameters
lambda	Security Parameter
m	Plaintext
c	Ciphertext

3.1.2 Secure Multi-Party Computation

Secure Multiparty Computation (SMPC) protocols allow a group of parties to jointly compute a function while also protecting the privacy of the participants' inputs. It was first introduced by Yao [Yao82] for the two-party case as a solution to the Millionaires Problem, and later generalized to the multi-party scenario by Goldreich, Micali and Wigderson [GMW87]. An important distinctive feature which distinguishes SMPC from other more conventional cryptographic schemes is that SMPC also seeks to protect against adversaries coming from the system itself, hence protecting the privacy of the participants' inputs from each other.

The SMPC field covers a wide list of cryptographic techniques [EKR18] ranging from several higher and lower-level tools as Garbled Circuits (GC) [BMR90], Zero-Knowledge proofs [GMW87], commitments, oblivious transfer [Kilian88] and secret sharing [Shamir79], to name just a few. Actually, it is also worth mentioning that strictly speaking, homomorphic encryption techniques can be seen as a subset of SMPC, and many SMPC protocols make use of HE to work properly. Even so, there is a current trend to separate them because SMPC is usually seen as a mechanism to implement concrete policy enforcements among several parties which are interactively performing computation, while HE is usually seen as a mechanism allowing a data owner to securely outsource computation on his/her data. With this in mind, an exhaustive classification of the different SMPC protocols would require to take into account several parameters, as for example, the number of parties involved (mainly two-party or multi-party), the type of corruption (passive, active, covert), the number of corrupted parties (honest majority, dishonest majority), the mobility of the adversary (static, adaptive corruption), etc.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Although SMPC originally started as a theoretical curiosity, since the mid 2000's important improvements were made showing that SMPC applications were in fact possible [MNPS04]. The first important example of deployment in a real scenario was the implementation of the sugar beet auction in Denmark in 2008 [BCD+09]. Currently, the most practical general-purpose protocols are based on additive secret sharing, and the use of Beaver triples [Beaver91] to perform interactive multiplications. They divide computation in two main phases: (1) an offline phase on which some sort of correlated randomness is computed and distributed among the parties, and (2) an online phase on which given some participants' inputs the previous correlated randomness is consumed to perform the intended secure computation.

The most representative example of the previous schemes is the case of the SPDZ protocol [DPSZ12] together with its different variants MASCOT [KOS16], Overdrive [KPR18] and SPDZ2k [CDESX18]. A common trait of all of them is their low computational cost. On the contrary, the number of interactions, and hence communication cost, grows with the number of parties and circuit depth, which makes the latter its weakest point in real scenarios.

SMPC technology has made a huge progress in the last few years [MPCalliance], and there are already many companies offering SMPC-related solutions as Sharemind [Sharemind], Sepior [Sepior], Zcash [Zcash], Unbound [Unbound]. This also includes several frameworks and software developments as the SCAPI library [Libscapi], SCALE-MAMBA SMPC-system [Scale-Mamba], MP-SPDZ [Keller20], Rmind, Jana relational database from Galois [Jana], etc. A very detailed list of several software solutions is available on the web of the TPMPC workshop [TPMPC].

Many of the above schemes have been used to implement important ML or FL tasks as the ones required for TRUMPET use cases [PRP+23]. Actually, there is also a rapidly growing list of works specifically designed to deal with training and prediction/classification phases. Some of the most recent are ABY [MR18], SecureNN [WGC18], Quotient [ASKG19], Gazelle [JVC18] and Delphi [MLSZP20].

3.1.3 Private Coded Distributed Computing

Private Coded Distributed Computing (PCDC) is a set of techniques that belong to the coded distributed computing paradigm. This paradigm uses coding theory to efficiently inject redundancy to mitigate fundamental bottlenecks in large-scale distributed computing, while preserving the privacy of the inputs [UAGJTT21]. In general, the goal of these schemes is to compute a function over several distributed nodes, while keeping the input dataset private. To do this, nodes receive assigned coded versions of the inputs (shares) created using some encoding functions, and then, each node computes the goal function using its particular shares.

One important feature of PCDC techniques is that, in general, they are very light in terms of computation, which makes them attractive for tasks as computationally expensive as the training of machine learning models. In these types of distributed schemes there exist fundamental trade-offs between privacy, computation cost, and communication cost. Consequently, some researching work in this field has focused on the reconstruction of the goal function when stragglers are present [LLPPR18, TLDK16, YLMA17]. The main goal of these works is to reduce the minimum number of

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

nodes that is required to carry out the computation of a function, which reduces the communication overhead associated with these techniques.

One limitation of PCDC is related to the fact that computing complex functions over the coded domain might excessively increase the number of nodes needed to decode the result, to the point that it might be impossible to do it. Another relevant limitation is that the combination of these coding techniques with verification technologies, to ensure the correct execution of the protocol by the participants, becomes quite difficult. PCDC is an emerging field which has undergone some of its most significant advances very recently, which causes the absence of libraries oriented to the implementation of the most recent coding techniques. However, since the most relevant PCDC schemes to the field are often based on simple mathematical functions and protocol flows, their implementation does not pose a great technical challenge.

3.1.4 **Differential Privacy**

Originally designed to deal with the problem of privacy-preserving database analysis, Differential Privacy (DP) addresses the conflicting objective of protecting individual records while still allowing to learn useful information regarding the whole database [DR14]. Consequently, its goal is to quantify and bound the individual information which is leaked on the aggregated database by means of the use of different mechanisms. Informally speaking, DP guarantees that an adversary cannot find out more information regarding an individual than can be inferred from a database lacking the individual.

One of its clear advantages when comparing to other privacy measures is that it presents strong privacy guarantees, even when there are multiple releases and an adversary has access to side information, hence avoiding linkage attacks. Data privacy through DP is based on the addition of noise using random variables through different mechanisms. The main parameter in these mechanisms is the privacy budget (ϵ). This parameter indicates the degree of noise added in the privatization: a small privacy budget value guarantees a greater privacy, but it entails a deterioration in the usefulness of the algorithm. This inconvenience causes having to find a value low enough to guarantee privacy while preserving its usefulness. Thus, numerous investigations have been performed to create techniques and tools with the aim of achieving said balance.

Mechanisms are classified according to their probability of failure. If a mechanism guarantees the correct privatization offered by DP, that is, it has no probability of failure, it is an ϵ-DP mechanism. The most common one in this category is the Laplacian mechanism [DKMMN06]. On the contrary, if a mechanism has a probability of failure δ, it is denominated as (ϵ, δ)-DP mechanism. Within this category, it is possible to find the widely used Gaussian mechanism [DR14]. According to the party which applies the differential privacy mechanism, nowadays two important threat models can be considered [APB+23]: (1) each data owner applies DP mechanisms before collecting and aggregating the data (local DP), and (2) a trusted party collects all the records and then applies a DP mechanism, whose output is then released (curator DP).

In view of the above-mentioned features, DP becomes a powerful method to guarantee the database privacy while still providing some utility. However, taking into account the Fundamental Law of Information Recovery, it still presents some drawbacks: the released database or result suffers a degradation on its accuracy depending on the number of queries that must be answered. In general, the amount of leaked information and accuracy degradation depends on several factors such as the required level of privacy, the amount of data available, the number and type of queries, the threat model and the complexity of the data. As an example of the possible trade-offs, if we fix the privacy level, better accuracy results can be achieved with the curator DP model. However, this model goes with the need of a trusted party to apply the mechanism directly on the private data, which for many cases is a very strong assumption [DR14]. Consequently, more recent work does not consider anymore a trusted curator but uses secure multi-party computation to obtain the output without revealing intermediate results. On the other hand, the overhead on computational cost caused by DP mechanisms is usually not high, typically the smallest among all the privacy methods discussed in this proposal. As a warning note, DP mechanisms are hard to characterise because the information leaked is highly dependent on the type and frequency of the queries [APB+23]. Additionally, DP mechanisms are sensitive to software defaults, as for example the available arithmetic precision.

There are several resources providing DP tools and exemplifying their use on more realistic scenarios. This includes the efforts of Apple Research on Differential Privacy for iOS and macOS keyboard statistics [AppleDP], Facebook Research on Differential Privacy providing a set of best practices to develop DP platforms and the release of a URL-shares dataset [KMRTZ20], Microsoft for obtaining usage statistics in Windows 10 [DKY17], the Google Differential Privacy Library [GoogleDP] and the Harvard Privacy Tools project [HarvardDP], to name just a few. Finally, a salient example coming from statistical agencies is the use of DP mechanisms to release the 2020 US Census Databases [Census22].

3.1.4.1 DP and Federated Learning

DP mechanisms find immediate application on TRUMPET use cases. First, they provide efficient ways of computing private database statistics. Second, they can complement other suggested privacy methods by either avoiding that the trained FL models can be stolen during the prediction/classification phase, or by providing additional protection for the training datasets thanks to the effectiveness of DP against membership inference attacks [AC19]. In the ideal scenario, secure multi-party computation (or homomorphic encryption or other cryptographic technique) is used to make the computations to avoid leaking intermediate results, and a suitable form of statistical privacy (in worst case classic DP) is applied to the output to avoid that the result itself reveals sensitive information.

From now on, we present different proposals which have managed to reduce the negative impact in accuracy of DP in FL, achieving a balance between privacy and the impact on the remaining utility of the data. The problem of finding a suitable privacy budget is approached from different fronts: (1) some works explore new properties of the mechanisms that allow different ways of accumulating the privacy budget in each iteration; (2) other works develop new mechanisms, aiming at an improvement compared to the usual Laplacian and Gaussian ones; finally (3) another option is to change where the mechanism is used in the FL algorithm.

Funded by the European Union | Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

3.1.4.2 Exploring new properties of mechanisms

Wei et al. [WLDMYFJQP20] consider a Gaussian mechanism with two different variances: (1) one of them is used in the loss function of the weights that goes from the client (data owner) to the server (aggregator); (2) the other, for the aggregation function calculated at the server, is used in an algorithm they called Noise before model Aggregation FL (NbAFL). The presented approach argues that both functions (the loss and the aggregation) have different sensitivities, being the sensitivity of a function a measure of how much the output of the function can change when a single individual data point is changed. These variances are parametrized in terms of: the number of exposures (the number of times it is necessary to send the parameters) of each set of local parameters at the clients, an upper limit for the weights, the number of clients to add in the aggregation function, and the total number of customers. In addition, the study evaluates a modification in which only some clients are randomly chosen for the aggregation function. They calculate the loss function with several combinations of parameters and do a comparison with the algorithm without DP. By doing this they achieve better results with the method of selecting K clients and manage to achieve similar results to the case without DP using high privacy budgets.

Another way of optimizing the privacy budget is presented in [Mironov17]. Using the Rényi divergence [Renyi61], Mironov [Mironov17] defines that a randomized mechanism is considered as (α, ϵ)- Rényi Differential Privacy, or (α, ϵ)-RDP, if it is applied to two neighbouring datasets (i.e., datasets that differ in only one entry) that do not diverge by more than ϵ for a given α used in the divergence calculation formula. If a mechanism is (α, ϵ)-RDP, it can be applied a new form of calculating the privacy budget that accumulates in each iteration of the FL algorithm, and that is better than the usual which consists in adding the privacy budget used at each iteration. Likewise, this method can be applied to transform an ϵ-DP mechanism into an (ϵ', δ)-DP mechanism.

Dong et al. [DRS19] proposed *f*-DP as a different view to define DP. They start from the idea of a null hypothesis test which, for two neighbouring datasets, if a data point belongs to one of them, it verifies the null hypothesis; while if it belongs to the other one, it will verify the alternative hypothesis. The objective is to find a trade-off function that makes these two hypotheses indistinguishable, i.e., that confuses false positives with false negatives. In practice, it is difficult to find a function with these characteristics, which leads to define μ-GDP, a type of DP based on a trade-off function between two normal distributions, one with mean *0* and another with mean μ. So, if a mechanism is *f*-DP, a different procedure can be applied in each iteration to calculate the accumulation of the privacy budget.

Another property of a mechanism is its privacy loss [DPgoogle22, DR16], which measures its effectiveness in generating outputs close to the outputs of neighbouring datasets. If the mechanism is ϵ-DP or (ϵ, δ)-DP, the privacy loss will be upper-bounded by the privacy budget. Its study has facilitated the use of the so-called Advanced Composition Theorems, representing an improvement over the previously developed composition approach. In this case, the Privacy loss is seen as a random variable and its distribution is studied, referred to as the Privacy Loss Distribution (PLD).

A similar idea to the previous one is to see the privacy budget as a random variable dependent on the mechanism. This is one of the premises for Moments Accountant (MA), introduced in Abadí et al. [ACGMMTZ16], that allows the use of a small portion of the privacy budget for each of the uses of the noise. The MA method is an alternative to the composition method for the total privacy budget

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

calculation and starts from the idea of calculating the moment bound of the random variable privacy budget. This moment bound is like a limit on growth of the moment. So, using the logarithm of the moment generating function of this random variable and its moment bound to study how much privacy budget must be used each time. However, one of the difficulties presented by the MA method is precisely to find the moment bound. For now, it is only found for the Gaussian mechanism. As a solution, Wang et al. [WBP19] propose years later an analytical calculation of the MA with which they obtain a tail bound that could be used for mechanisms that are (α, ϵ)-RDP. Their experiments show an improvement with respect to previous methods of calculating the privacy budget.

3.1.4.3 Developing new mechanisms

A different way to improve the accuracy is to develop new mechanisms tailored to the type of data or the algorithm they are using. Two examples of basic mechanisms are the Geometric and the Binomial:

- **Geometric mechanism**: adds noise with discrete values, choosing them through an extended version of the geometric distribution, being able to give as result not only natural numbers but all integers [GRS09].
- **Binomial mechanism**: also adds noise with discrete values, but with a binomial distribution. This mechanism is (ϵ, δ)-DP [MT07].

Other examples of used mechanisms are those developed by Zhao et al. [ZZYWWLNL21], who created an improved version of the ones introduced by Wang et al. [WXYZHSSY19]. These mechanisms are:

- **ThreeOptions:** Taking as input a numeric vector with numbers from *-1* to *1*, returns a value *C*, *0* or *-C* for each of the values with a probability depending on the input value. According to the obtained results, this mechanism outperforms PM-SUB and PM-OUT when it comes to low privacy budget.
- **PM-SUB** and **PM-OPT**: Given a list of numbers from *-1* to *1*, returns another list with numbers from *-A* to *A*, using uniform distributions over a range of values depending on the input. These two mechanisms differ in the way these ranges of values are obtained, being PM-SUB a suboptimal but simpler option than PM-OPT. These two mechanisms can be passed through another mechanism (Discretization Postprocessing) to obtain a discrete output, which is obtained with a Bernoulli distribution. According to the obtained results, PM-SUB works better than ThreeOptions when the privacy budget is high.
- **HM-PT**: This is a combination of ThreeOptions and PM-SUB to take the advantages of both. This mechanism uses PM-SUB with a certain probability, otherwise use ThreeOptions. Authors compare ThreeOptions, PM-SUB and HM-PT mechanisms using the MSE metric with different datasets. They also use different methods: Linear regression, logistic regression and SVM, with various datasets. All three mechanisms achieve reasonably low MSE, but HM-PT slightly outperforms the others.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Agarwal et al. [AKL21] developed a new mechanism that uses the Skellam distribution, a discrete distribution that results from the subtraction of two Poisson distributions. They perform the privacy budget accumulation count in two different ways: one uses the properties of PLD while another uses (α, ϵ)-RDP. Testing the accuracy and MSE, they conclude that the PLD method or using the properties of a (α, ϵ)-RDP mechanism give more efficient results than the previous methods for the Skellam mechanism, the Gaussian, and the discrete Gaussian.

New mechanisms developed for Apple with Standford University in Bhowmick et al. [BDFKR18] are used in an algorithm that applies three mechanisms to strengthen privacy. One of the mechanisms is Gaussian applied in the aggregator. The other two, developed by the authors, are:

- Privatized Unit Vector (PrivUnit): chooses a vector according to a combination of uniform distributions through an input vector.
- Privatize the magnitude with absolute error (ScalarDP): using a scalar, the privacy parameter epsilon, an integer, and an upper bound, obtains a scalar through a combination of two uniform distributions.

3.1.4.4 Changing where we apply the mechanisms

As previously mentioned, another way to achieve a balance between privacy and utility of the algorithm is by selecting where the mechanism is applied. This procedure can be applied in several ways inside FL, since it can be modified to insert the noise in different parts.

Bhowmick et al. [BDFKR18] combine PrivUnit and ScalarDP by firstly separating the vector of weights in direction and magnitude and then, applying respectively PrivUnit and ScalarDP. The combination of these two mechanisms results in a ($\epsilon 1$+$\epsilon 2$)-DP mechanism. In their results the accuracy decreases around *0.1* comparing to not using DP.

Hu et al. [HGLPG20] find a new way to use the mechanism in the algorithm. Authors propose the convergence problem of the FL as an optimization problem to find a minimum required number of iterations. To solve this, the conjugate dual is used. This changes the original optimization primal problem into another one with different variables, called dual variables. In this case, the dual variables correspond to tuples of feature and label, the data with which the network is trained. That is, the primal problem goes from "Finding the weights that minimize the loss function" to the conjugate dual problem “Finding the training data that minimize the conjugate loss function”. Authors propose an algorithm that adds Gaussian noise twice in different steps: first in the dual variables, and subsequently after multiplying the previous result with the matrix of features to obtain the weights. Once the weights with noise are calculated for each client, they are sent to the aggregator using the MA. They compare the objective function of the primal problem, in which loss function has a strong influence, of the problem without privacy preserving methods and with this method.

At last, other algorithms make use of a combination of different techniques. This is the case of NVIDIA in Li et al. [LLSY17], that combines the Sparse Vector Technique (SVT) with Selective Parameter Update. SVT [LSL17] consists in sending only some selected weights with noise, so it can guarantee privacy without consuming privacy budget at each iteration. Selective Parameter

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Update [SS15] is a technique designed to reduce the impact of large weights on the overall computation, where only weights that are greater than a threshold are shared, but these weights are clipped to a pre-established value. The combination of these two techniques gives rise to an algorithm which compares the weights with a pre-established threshold. Each weight is only sent if, when added with parametrised noise using a privacy budget, is greater than a pre-established threshold to which noise parametrizes with a different privacy budget. That is, three noises with three different privacy budgets are used. Comparing the use of these protocols with different privacy budgets and with the ML model without using SVT, differences in accuracy of less than *0.05* are achieved for relatively small privacy budgets.

3.1.4.5 Future perspectives

This section has studied different mechanisms and tools to apply DP to FL settings but with a limited privacy budget. The analysed works provide good results in terms of accuracy, MSE or other ML metrics, but they are not fully tested to verify how well they protect against particular privacy attacks. These features are taken for granted by the DP promises [KMRTZ20] for any privacy budget, so it is not studied whether the used privacy budget is low enough to guarantee privacy in a practical scenario.

For a first approach on the use of DP to protect FL settings in the TRUMPET project, SVT looks easy to implement and provides good results. In addition, the TRUMPET project will also explore the use of new combinations of mechanisms, such as the mechanisms developed in Bhowmick et al. (α, ϵ)-RDP, combined with MA to study if it allows to reduce the privacy budget. Other promising combinations include SVT with Gaussian mechanism or the combination of the Skellam mechanism with Analytic MA.

3.2 Frameworks, tools and software for FL-tailored PETs

The following table lists libraries and frameworks that implement PET methods and that could be used in TRUMPET. They are classified according to the privacy method they implement.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Table 2 – Libraries and frameworks implementing different PET methods.

PETs libraries and frameworks		
Privacy method	**Library/framework**	**Links**
Differential Privacy	Google Differential Privacy Library	https://github.com/google/differential-privacy
	OpenDP	https://projects.iq.harvard.edu/opendp https://github.com/opendifferentialprivacy/
	cuHE	https://github.com/vernamlab/cuHE
Homomorphic Encryption	SEAL	https://github.com/microsoft/SEAL
	HEAAN	https://github.com/snucrypto/HEAAN
	HElib	https://github.com/shaih/HElib
	NFLlib	https://github.com/CryptoExperts/FV-NFLlib
	PALISADE	https://palisade-crypto.org/
	Lattigo	https://github.com/ldsec/lattigo
	TFHE	https://tfhe.github.io/tfhe/
	FHEW	https://github.com/lducas/FHEW
	HE Transformer	https://github.com/IntelAI/he-transformer
	Concrete	https://github.com/zama-ai/concrete
	OpenFHE	https://github.com/openfheorg/openfhe-development
	TFHE-RS	https://github.com/zama-ai/tfhe-rs
Secure Multiparty Computation	Libscapi	https://github.com/cryptobiu/libscapi
	SCALE-MAMBA	https://homes.esat.kuleuven.be/~nsmart/SCALE/
	Additional software and framework resources (e.g., ABY, ABY3, FRESO, CrypTen, Tf-encrypted)	https://github.com/rdragos/awesome-mpc
Private Coded Distributed Computing	Galois	https://github.com/GuildOfWeavers/galois
Zero Knowledge Proof	libsnark	https://github.com/scipr-lab/libsnark
	gnark	https://github.com/ConsenSys/gnark
	Bulletproofs	https://github.com/dalek-cryptography/bulletproofs

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

4 A combination of PET methods for AFL

The techniques presented in the previous section cannot simultaneously protect against all the privacy leaks and attacks that can be present in an FL setting. Some are better for protecting the input privacy and/or the output privacy, and others for protecting the training data privacy or the model privacy. Furthermore, some of them entail a potential loss of utility. So, we must combine them to get a higher protection regarding privacy leaks.

In general, we can distinguish three main categories of PET strategies:

- **Encryption-based computation**: these techniques typically aim at computing an output without any party obtaining more information than the output itself, e.g., secret sharing, homomorphic encryption, garbled circuits, etc.

- **Statistical privacy**: in some cases, even only revealing the final output of an algorithm may leak sensitive information. Therefore, statistical privacy techniques aim at perturbing the output, obtaining an approximate output while hiding sensitive information. Such techniques include differential privacy, pufferfish privacy, some of the new privacy metrics being developed in TRUMPET's WP3, etc.

- **Verification strategies**: these strategies aim at verifying that all parties perform their work correctly, ideally without revealing any additional information but the outcome of the verification. Such strategies often employ zero knowledge proofs, sometimes with the help of specific properties of the encrypted computation techniques, e.g., additive homomorphic encryptions implicitly already allow to verify correct addition without revealing the numbers being added.

The choice of techniques in these three categories is often largely independent, i.e., one can select an encrypted computation strategy, a statistical privacy technique and a verification strategy to build an algorithm, and most combinations are possible even if in some cases some of them offer some additional advantages. Below, we do not describe all possible combinations of these three categories but focus on some combinations and techniques of particular interest.

4.1 Encrypted Computation: SMPC with HE

4.1.1 SMPC vs HE

The main difference between HE and SMPC regarding efficiency relies on the fact that HE requires less interaction but higher computational cost, and for SMPC, the computational cost is much lower, but communication becomes the main bottleneck. Consequently, an interesting path to follow is the combination of both strategies to trade off communication for computation, and vice versa, depending on the available network and computation resources [LJLA17]. A clear example of this idea is the use of Multi-Key Homomorphic Encryption (MKHE) which allows to reduce the costs of bootstrapping in HE by means of interactions among parties [MTP20, AHSL22]. There are already some works exemplifying its use for training machine learning models [FTPSSBH20].

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

4.1.2 Homomorphic Encryption with Multiple Keys

Most existing HE-based solutions for secure computation work under a **single-key approach** in which ciphertexts are encrypted under the same key. However, this approach is **clearly impractical** for **general multi-party computation** with more than two parties. Note that, ideally, independent 2-party computation tasks must be performed between every pair of parties (see an example for the distributed model of computation with HE explained in Figure 2.(b) of [AHSL22]).

On the contrary, remember that TRUMPET [PRP+23] focuses on a **particular computation model given by FL**, in which multiple data owners delegate sensitive data (model updates) to the aggregator to train a collaborative global model. Consequently, while it is true that HE allows us to make general computations over encrypted data, its use must be adapted to the specific needs of FL. Still, many of the current HE-based solutions for secure aggregation work under this **single-key approach**, where the ciphertexts provided by all data owners are encrypted under the same key. This introduces a relevant drawback in FL which consists in the possibility of a **collusion between the cloud and some of the data owners** [PPV22, PPV22HE]. If the aggregator sends the ciphertexts received from one data owner to another one, who also owns the secret key, all inputs provided by the former could be decrypted by the latter (see Figure 3).

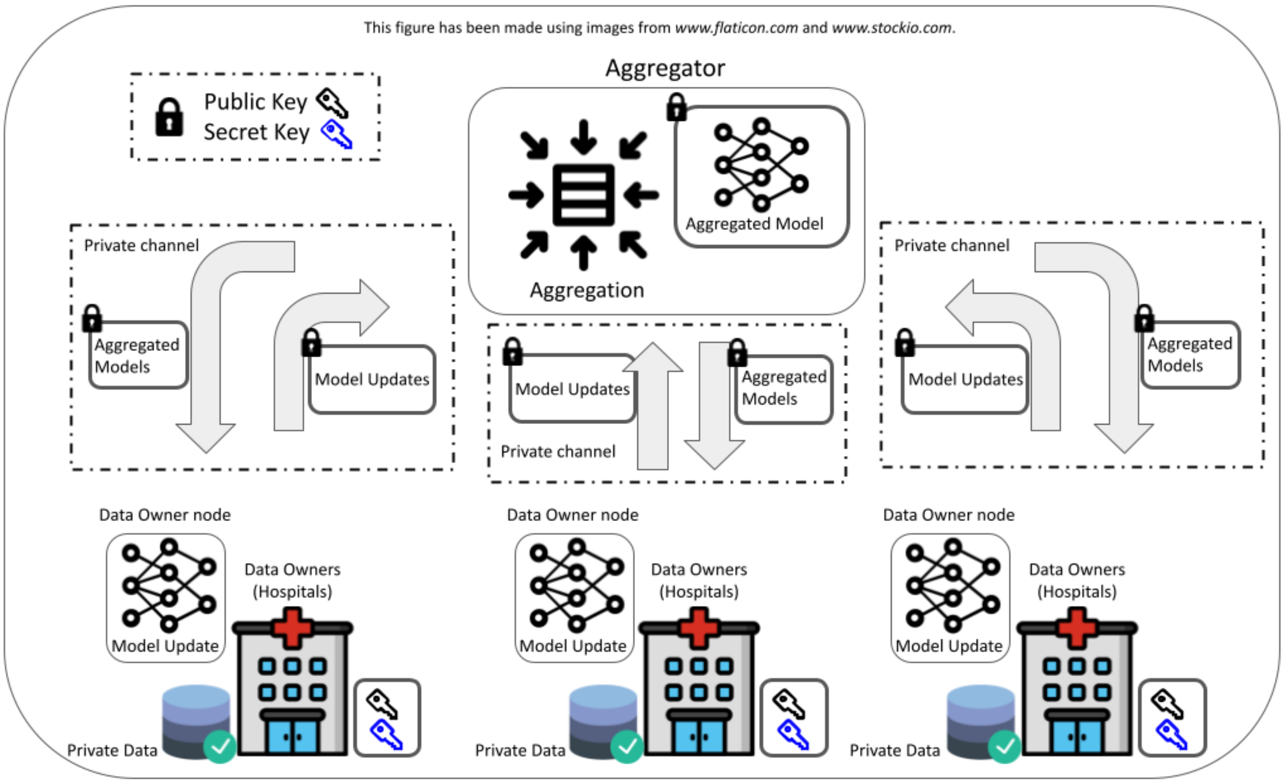

Figure 3 – Secure aggregation by means of Single-Key HE.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

To ensure data privacy in a more practical way, the aggregator should be able to realize secure aggregations on model updates which have been encrypted using different keys for each data owner. Schemes that allow homomorphic evaluation on data encrypted under different keys are known as multi-key HE schemes (see Figure 4).

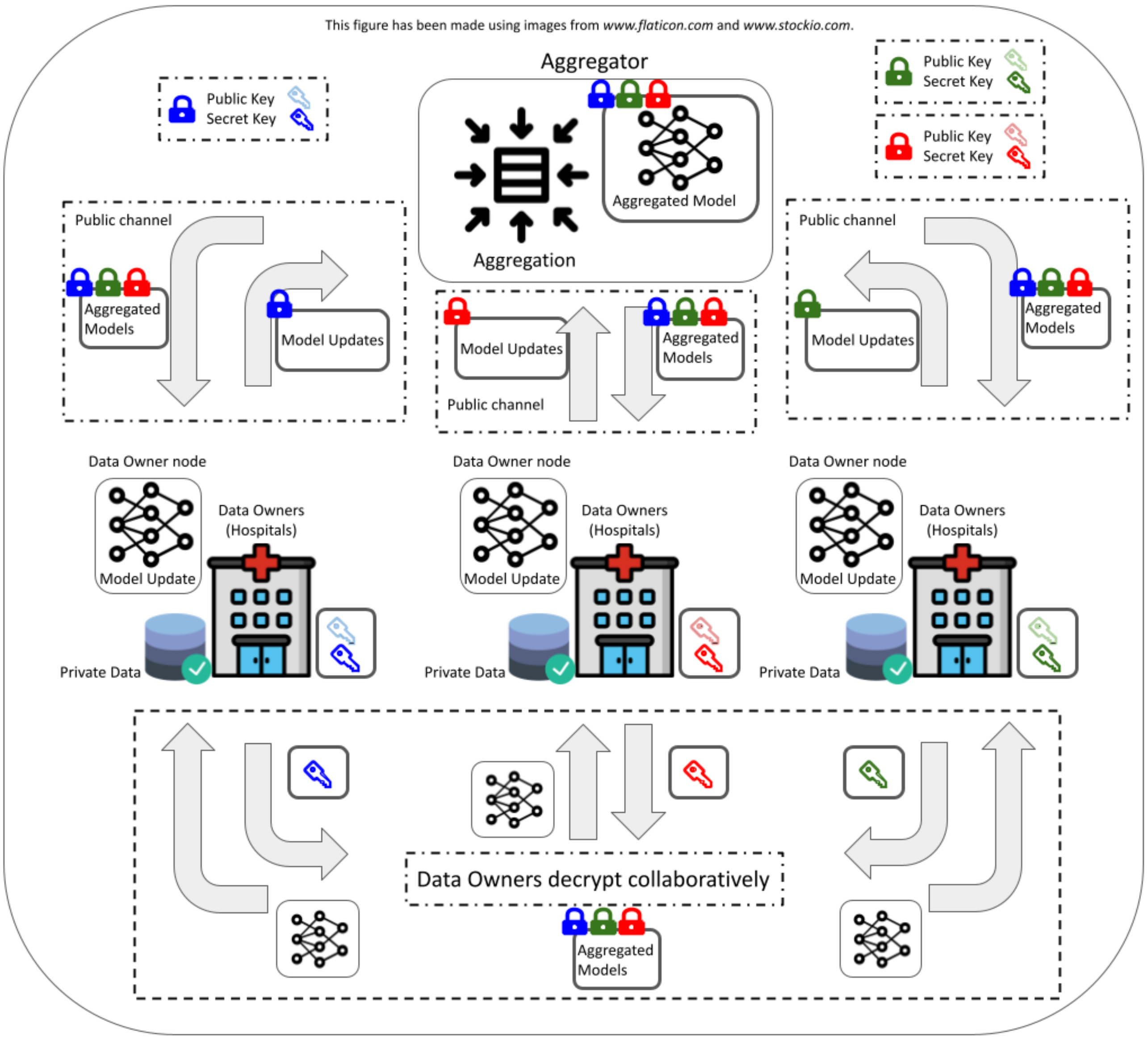

Figure 4 – Secure aggregation by means of Multi-Key HE.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

With a HE whose ciphertexts are encrypted under different keys or their combination, the aggregator could perfectly compute the required aggregation function and return the aggregated models to the final user. Now, by means of a collaborative protocol, parties could re-encrypt the output under the final user's key, or directly decrypt it. **As input ciphertexts are not encrypted under the secret key of the final user, the risk of a collusion with the cloud is removed**.

Next, we briefly detail the existing possibilities to have several keys and HE at the same time. With this aim, the "ideal" concept of a multi-key HE is divided into the two main families: Threshold HE and Multi-Key HE.

- **Threshold HE**: This type of scheme requires a setup phase involving the joint participation of all the parties for the key generation process. In general, each party generates his/her own secret key and public key and, afterwards, all of them gather their keys to generate a collective public key.
- **Multi-Key HE**: In contrast to Threshold HE, the setup phase of a multi-key HE scheme does not require the joint collaboration of all the parties. Each user can independently generate its own secret and public keys, afterwards ciphertexts under different public keys can be evaluated "on-the-fly". This results in an output ciphertext, whose decryption will depend on the secret keys of all the input ciphertexts used in the circuit evaluation.

It is worth noting that Threshold and multi-key variants present different advantages and shortcomings depending on the desired key setup complexity, evaluation efficiency, or system scalability, to name just a few practical performance indicators. Actually, in practice, hybrid systems can be designed which combine the best features of threshold and multi-key HE schemes. A general review of the problem of managing several keys can be found in [TCLRLB16], but only for data sharing. A review of the case for several keys in combination with HE, currently can be found in the works [Bongenaar16, AH19, AHSL22].

4.1.3 Threshold Homomorphic Encryption (ThHE)

Supporting secure and privacy-preserving computations on inputs provided by different data owners requires data to be encrypted under multiple keys. The first method which we can consider defining several keys is to make use of threshold encryption [DF89] techniques. With these mechanisms, the parties must **generate the collective key during the setup phase** and before any computation. Note that revoking and changing such a joint key can also be done, but it is required that the setup phase is run again, and afterwards re-encrypt the data under the new joint key.

In a threshold encryption scheme the parties can independently encrypt their data under a joint key but must still cooperate for decryption, which is done by means of a distributed decryption protocol. Shamir's secret sharing [Shamir79] is the basis for a threshold structure *(T, N)*, which allows to define *N* shares of a global secret key to be distributed as individual secret keys among the different parties. The idea is that only *T* parties are required to collaborate for decryption [DHMR07].

The choice of the threshold value *T* allows us to securely deal with two adversary models:

- ***T* ≥ *N/2*** corresponds to the case of **dishonest majority**, when more than half of the users may be corrupted.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

- ***T* < *N/2*** corresponds to the case of **honest majority**, when less than half of the users may be corrupted.

The threshold variant of a HE scheme combines the properties of threshold decryption together with homomorphic computations. Formally, we can define a ThHE scheme as the tuple of algorithms ThHE = (Setup, KeyGen, JointKeyGen, Enc, Eval, PartDec, Combine):

- ThHE.**Setup**(1^*lambda*) → *pp*.
- ThHE.**KeyGen**(*pp*) → (*pk*, *sk*). Each party calls this primitive to generate (pk_i, sk_i)
- ThHE.**JointKeyGen**(pk_1, pk_2, …, pk_N) → *pk**, *ek**. (*Interactive protocol*)
- ThHE.**Enc**(*pk**, *m*) → *c*.
- ThHE.**Eval**(*pk**, *f*, *c*, *c'*) → c_{eval}.
- ThHE.**PartDec**(sk_i, *c*) → p_i. (*Interactive protocol*)
- ThHE.**Combine**(p_1, p_2, …, p_N) → *m*. (*Interactive protocol*)

4.1.3.1 **Joint Key generation**

Some works in the literature assume that the collective public and secret keys are generated by a trusted dealer, removing the JointKeyGen protocol [DF89, CGS97]. Alternatively, the collective public key is also generated in other works by the ThHE.JointKeyGen protocol, having as inputs the different individual parties' keys [RSTVW22].

4.1.3.2 **Distributed decryption**

With respect to the **distributed decryption phase,** we can use ThHE.PartDec followed by ThHE.Combine protocols, or in other works such as [MTBH21] the use of a **collaborative switching key protocol** is proposed. Hence, there exist other methods that avoid the use of a distributed decryption by **re-encrypting the ciphertext under the key of the final receiver**. In Table 3, the reader can consult different threshold HE constructions.

Next, we elaborate more on the peculiarities of Threshold HE schemes depending on the underlying threshold structure.

- **(N, N)-threshold HE**: For this particular access structure, many works try to exploit an **additive homomorphism in the keyspace** which is present in many encryption schemes. Consequently, we search that all shared secret keys sk_i satisfy the relation:

$$sk = \Sigma_i \, sk_i, \text{ for collective public and secret keys } pk \text{ and } sk.$$

As we have seen, once the key setup is done, the rest of evaluation primitives work as in the case of a single-key HE scheme. However, **decryption is impossible if one of the parties does not collaborate**, which can **constitute a point of failure at the time of decryption**.

- **(T, N)-threshold HE**: For *T* < *N*, this type of threshold HE scheme appears as a more flexible construction than (N, N) - ThHE, in which we can decrypt even if some of the users are not present for decryption.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

As an example, in [DF89] the authors propose a general (T, N) threshold version of ElGamal, but it requires a trusted dealer to output the joint public key and the different shared individual keys. Afterwards, [Pedersen91] removed the need of a trusted dealer and introduced an additional validation step in the secret shares for robustness. Even so, all these protocols made use of a trusted authority for decryption. Subsequent works removed this last assumption by relying on Diffie-Hellman key generation [DH76].

A common aspect of these approaches is that they are **based on the use of Shamir's Linear Secret Sharing** [Shamir79] to generate secret key shares for the *N* parties. The idea is to define $sk^* = s^*$ as the constant term of a polynomial $f(x)$, that is, $f(0) = s^*$. The polynomial $f(x)$ has degree $T - 1$, and hence, *T* points are required to reconstruct $f(x)$ and recover s^* by means of Lagrange interpolation. Finally, each sk_i is defined as $sk_i = f(i)$, for $i = 1, \ldots, N$.

Regarding lattice-based HE, [BGGJKR18] proposed the first (T, N)-threshold HE scheme based on LWE. Similarly to previous works, shared keys are defined by means of Shamir's Secret Sharing. Additionally, in [BGGJKR18] a ***universal thresholdizer*** was proposed. This method allows extending any general single-key HE scheme into a (T, N)-threshold one. A more detailed list of threshold HE schemes can be found in Table 3.

As we have seen, both (N, N) - and (T, N) - threshold HE schemes require the participation of all the parties during the "joint key setup." One clear advantage relies on the fact that the size of the joint public key is usually independent of the number of parties. On the other hand, if the joint key must be revoked because users are frequently joining or leaving the system, the whole threshold key setup must be run again and again. If the number of users changes often, we will see next that Multi-Key HE schemes are a more suitable option allowing one to work with multiple keys "on-the-fly", hence avoiding running again the key setup only because a joint key is revoked.

Table 3 – Several Threshold HE schemes (see [AHSL22])

THRESHOLD HE						
Scheme	**Hard problem**	**HE scheme**	**HE type**	**CRS**	**Slot packing?**	**Threshold (t,n) for general t?**
Threshold ElGamal [DF89, Pedersen91]	Discrete logarithm	ElGamal	Partial	No	No	Yes
Threshold Paillier [CDN01]	Composite residuosity	Paillier	Partial	No	No	No
[AJLTVW12]	LWE	BGV	Fully	No	Yes	No
[BGGJKR18]	LWE	GSW	Fully	Yes	Yes	Yes
[MTBH21]	RLWE	BFV/BGV/CKKS	Fully	Yes	Yes	No
[MBH22]	RLWE	BFV/BGV/CKKS	Fully	Yes	Yes	Yes

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

4.1.4 Multi-Key Homomorphic Encryption (MkHE)

A general **description of an "on-the-fly" Multi-Key Homomorphic Encryption** (MKHE) scheme: [LTV12] is the first work to describe an MKHE encryption scheme able to **support homomorphic operations between ciphertexts encrypted under different combinations of secret keys**. Ciphertexts can be encrypted either alone or also under a different combination of individual users' keys *(pk_1, …, pk_N)*. The main difference between ThHE and MKHE relies on the fact that the latter does not require a joint key setup phase among all the participants. For example, now a ciphertext encrypted under pk_1 could be perfectly operated with a ciphertext encrypted under pk_2 and pk_3. However, MKHE still requires a distributed decryption protocol to finally recover the encrypted results. Following the example, the final ciphertext is encrypted under *(pk_1, pk_2, pk_3)*, so the *1-st*, *2-nd* and *3-rd parties* should collaborate during decryption.

General MKHE primitives. We can define a MKHE scheme as the tuple of algorithms MKHE = (Setup, KeyGen, Enc, Extend, EvalKeyGen, Eval, Dec):

- MKHE.**Setup**(*1^lambda, 1^kappa*) → *pp*. Where *kappa* is a bound on the maximum number of different keys allowed.
- MKHE.**KeyGen**(*pp*) → (*pk*, *sk*). Each party calls this primitive to generate (pk_i, sk_i).
- MKHE.**Enc**(*pk*, *m*) → *c*. Given a public key *pk* and message *m*, the algorithm outputs a ciphertext encrypted under public key *pk*.
- MKHE.**Extend**($\{pk_1, \ldots, pk_K\}$, *c*) → c_{ext}. Given a set of *K* public keys with $K \leq kappa$, and a ciphertext *c*, the algorithm outputs the extended ciphertext c_{ext} under public keys $\{pk_1, \ldots, pk_K\}$.
- MKHE.**EvalKeyGen**(pk_{comb}) → ek_{comb}. Given a combined public key pk_{comb}, the output is the extended evaluation key.
- MKHE.**Eval**(pk_{comb}, *f*, c_{ext}, c'_{ext}) → c_{eval}. Given two ciphertexts extended under the same "combined public key" pk_{comb}, outputs the evaluated ciphertext c_{eval}.
- MKHE.**Dec**((sk_1, …, sk_K), c_{ext}) → *m*. Given all the secret keys under which c_{ext} is encrypted, the algorithm outputs its decryption.

At the same time, we can distinguish two different subcategories of Multi-key HE:

4.1.4.1 Single-hop HE

The main feature which characterizes this category of MKHE schemes is that the **primitive MKHE.Extend can only be applied once**. So, if a ciphertext is the result of a homomorphic computation, no further extensions to additional keys can be done. The first single-hop variant was presented in [CM15], where a compiler for multi-identity homomorphic IBE (Identity Based Encryption) was introduced. The compiler, based on the GSW scheme [GSW13] (see Table 5), allows the creation of ciphertexts encrypted under different identities, hence extending the "single-identity" of GSW to multiple identities. The link between this new GSW-based scheme and MKHE relies on the fact that **the concept of identities used in this work can be equivalently seen as the use of different keys in the same ciphertext**.

Additionally, and contrary to ThHE, in the literature of MKHE schemes (e.g., [LTV12, CM15]) many works consider that the decryption primitive MKHE.Dec has access to all the secret keys *{sk_1, …,*

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

sk$_K$}, hence assuming the presence of a trusted party who holds all the required secret keys and performs decryption. Another limitation from [CM15] is that the number of possible keys is bounded by *K*. The work [MW16] fixes the last two shortcomings, in such a way that their improved multi-identity GSW-based scheme (1) does not limit the maximum number of keys a ciphertext can be extended to, and (2) a two-round SMPC-protocol is introduced for collaborative decryption (similarly to the *partial decryption* and *combine* primitives used in ThHE).

4.1.4.2 Multi-hop HE and other properties

Multi-hop MKHE **removes the limitation of single-hop schemes in the number of times we can use the Extend primitive**. **Several redesigns have been proposed** on top of all the **main families of lattice-based HE** used as the baseline scheme. Some relevant examples include multi-hop MKHE variants of GSW [PS16, BP16], BGV [LTV12, CZW17, LZYHLT19], TFHE [CCS19] and both BFV and CKKS schemes [CDKS19]. See more examples in Table 4.

In addition to moving from single-hop to multi-hop, and following the initial work [LTV12], there are several approaches to build MKHE schemes holding more specific properties. Some works improved the expansion in ciphertext size of [LTV12] from quadratic to linear [BP16, CZW17, YKHK18, LZYHLT19, CDKS19]. Some particular schemes do not require CRS in the key generation phase [LTV12, DHRW16]. Lastly it is highlighted that Multi-Key extensions of RLWE-based schemes [CZW17, YKHK18, CDKS19, AH19] preserve the capacity of working with **SIMD (Single-Instruction, Multiple-Data)** encrypted operations thanks to **slot packing**.

Table 4 includes a comparison summary for the main features of MKHE schemes.

Table 4 – Several Multi-Key schemes (see [AHSL22])

MULTI-KEY HE								
Scheme	**Hard problem**	**Ciphertext size grows**	**Protocol rounds**	**Multi-hop**	**# keys is bounded?**	**CRS**	**Slot packing**	**Uses bootstrapping?**
[LTV12]	NTRU	Quadratic	-	Yes	Yes	No	No	No
[CM15]	LWE	Quadratic	-	No	No	Yes	No	No
[MW16]	LWE	Quadratic	2	No	No	Yes	No	No
[PS16]	LWE	Quadratic	2	Yes	Yes	Yes	No	No
[BP16]	LWE	Linear	2	Yes	No	Yes	No	Yes
[DHRW16]	piO	Quadratic	2	Yes	No	No	No	Yes
[CZW17]	RLWE	Linear	2	Yes	Yes	Yes	Yes	No
[YKHK18]	RLWE	Linear	2	Yes	Yes	Yes	Yes	No
[LZYHLT19]	RLWE	Linear	2	Yes	Yes	Yes	Yes	No
[CDKS19]	RLWE	Linear	2	Yes	Yes	Yes	Yes	No

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[AH19]	RLWE	Constant	4	Yes	Yes	Yes	Yes	No
[CCS19]	TLWE	Linear	2	Yes	Yes	Yes	No	Yes

Table 5 – Base Single-Key HE used in each MKHE scheme (see [AHSL22])

MULTI-KEY HE	
Scheme	**Base Single-Key HE scheme which is extended?**
[LTV12]	NTRU
[CM15]	GSW
[MW16]	GSW
[PS16]	GSW
[BP16]	GSW
[DHRW16]	GSW
[CZW17]	BGV, ring-GSW
[YKHK18]	BGV, ring-GSW
[LZYHLT19]	BGV, ring-GSW
[CDKS19]	BGV, CKKS
[AH19]	BFV, ring-GSW
[CCS19]	TFHE

4.1.4.3 Comparison between ThHE and MkHE

Once we have revisited the most prominent constructions for both ThHE and MKHE, next we summarize their strong and weak points.

Threshold HE:

- Flexible access structures for distributed decryption, either dishonest or honest majority. ☑
- Homomorphic computation with encrypted data presents almost no changes and reduced overhead compared to the corresponding baseline single-key HE. ☑
- The setup phase requires a collaborative protocol between all the involved parties. ⊘
- Every time the joint key must be revoked, data re-encryption and a new setup phase must be run. ⊘

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Multi-Key HE:

- Each user can encrypt his/her data with their own secret key. ✓
- The setup phase does not require a collaborative protocol to generate a joint key. ✓
- Homomorphic computation with encrypted data presents significant changes and overhead compared to the corresponding baseline single-key HE. ⃠
- If a group of parties remains the same over time, there is an overhead in ciphertext extension proportional to the number of parties of the group. ⃠

4.1.5 Beyond the state of the art: Tailored MkHE for Average Aggregation

Several works have used single-key HE [PBCSSZ23-HES, MSMGGS21, GSSG22] and even multi-key HE [MTBH21, SPTFBSP21, DCESBPTBH23] as a main building block inside the pipeline of FL. However, a common trait of most of these HE schemes is that they are initially designed to homomorphically evaluate very complex functions which are approximated with high-degree polynomials. Contrarily, in many FL scenarios the considered aggregation functions are usually not as complex, and there is still room for improvement when the homomorphic computation does not have a high multiplicative depth. Following this line of research, it has been shown that it is possible to optimize the use of multi-key HE for the setting of secure average aggregation [PBCSSZ23].

In particular, [PBCSSZ23, PBCSSZ23-HES] propose a lattice-based MKHE scheme especially designed for the average aggregation primitive. Particularizing TrHE and MKHE schemes, we can define a MKAggHE scheme with multiple keys tailored for secure aggregation as the tuple of algorithms MKAggHE = (Setup, KeyGen, Enc, Agg, PartDec, Combine):

- MKAggHE.**Setup**(1^*lambda*) → *pp*, *{share*$_1$*, …, share*$_L$*}*. (*Interactive protocol*)
- MKAggHE.**KeyGen**(*pp*) → (*sk*). Each party calls this primitive to generate *sk*$_i$.
- MKAggHE.**Enc**(*sk*, *share*, *m*, *T*) → *c*. Given a secret key *sk,* a share *share* and message *m* to be encrypted in round *T,* the algorithm outputs a ciphertext encrypted under secret key *sk*.
- MKAggHE.**Agg**(c_1, …, c_L) → c_{agg}. Given a set of *N* ciphertexts encrypted for round *T*, the algorithm outputs the aggregated ciphertext c_{agg} of round *T* under secret keys *{sk*$_1$*, …, sk*$_L$*}*.
- MKAggHE.**PartDec**(*sk*$_i$, *c*) → *p*$_i$. The aggregated ciphertext *c* of round *T* can be partially decrypted with key *sk*$_i$.
- MKAggHE.**Combine**(p_1, p_2, …, p_L) → *m*. (*Interactive protocol*) The partial decryptions must be combined to obtain the aggregated result of round *T*.

In Table 6 we include a high-level comparison between a baseline solution for additive secret sharing, non-optimized single-key and threshold HE, and the recent optimized secret-key variant, which also makes use of additive secret sharing during the setup phase as a building block.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

Table 6 – Comparison of several methods for secure aggregation (see [PBCSSZ23] for details)

SECURE AGGREGATION			
Protection Method N Agg. rounds L Data Owners	**Communication cost**	**Relevant properties**	**Size of the Protected Inputs**
Additive secret sharing (Baseline example from [PBCSSZ23])	O($N \cdot L^2$)	• A new set of shares must be generated per each agg. round. • Avoids collusion with the aggregator	• Same as the plaintext space
Public-Key HE (single-key)	O($N \cdot L$)	• Risk of collusion between the aggregator and secret key owner	• Public-key ciphertexts
Threshold HE (L-out-of-L)	O($N \cdot L$)	• Avoids collusion with the aggregator • Need of generating a collective public key	• Public-key ciphertexts
Tailored Multi-key [PBCSSZ23]	O($N \cdot L$)	• Avoids collusion with the aggregator • Light setup phase to generate a set of additive shares which can be reused in consecutive agg. rounds.	• Secret-key ciphertexts • If RLWE, half the size of public-key ciphertexts. • If LWE, removes a quadratic factor.

Compared to the non-optimized use of HE for average aggregation, MKAggHE [PBCSSZ23] reduces the communication cost per party by a half if based on RLWE, and from quadratic to linear if based on the LWE problem (see Table 6). This benefit of working with secret-key ciphertexts comes from the fact that during the same aggregation round T, the component *"a" from* RLWE or LWE samples is also the same for all submitted ciphertexts. Therefore, this term can be locally precomputed by all Data Owners, so avoiding sending it to the aggregator.

This optimization exploits the same **additive homomorphism in the keyspace** which is present in threshold HE. Consequently, the final aggregated ciphertext is encrypted under a collective secret key *sk* with the form:

$$sk = \Sigma_i \lambda_i \cdot sk_i,$$

being *the* λ_i, the weights associated with the computed aggregation rule.

We remind the reader that, while this type of solution can be very efficient for the average aggregation rule, many of these improvements cannot be applied for more complex aggregation functions. For this last scenario, the use of the already mentioned more conventional multi-key HE schemes is still needed.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

4.2 Encrypted Computation: SMPC with PCDC

4.2.1 Secret Sharing as a building block

Secret sharing is a technique where *n* parties each get a part (a share) of a set, such that to reconstruct the secret at least a certain number *t* of these parties are needed, and no information can be obtained by combining only *t-1* shares. Popular secret sharing schemes are additive secret sharing [DFKNT06] and Shamir's secret sharing [Shamir79]. An important advantage of such schemes is that no trusted party is needed, and the system remains secure even when *t-1* parties collude. A disadvantage is that in general secret sharing has a higher cost because all parties need to participate and communicate during the execution of several operations, often leading to a cost of $O(n^2)$. Even so, specific operations such as additions and inner products can be performed much more efficiently. Several libraries exist which implement secret sharing in a reliable way [TPMPC].

4.2.2 Private CDC

The original scheme that provides information-theoretically security is the well-known BGW protocol. It is based on Shamir's Secret Sharing scheme where a master node creates *N* coded shares *T*-secure which are subsequently sent to the workers. There are multiple variants of coded schemes based on Secret Sharing, like leveraging the use of staircase codes [BPR20], using entangled polynomial codes [NNM18] to allow alternative types of input partitions, using polynomial codes [NM21] to massive matrix multiplication, and utilizing GCSA codes [CJWJ21] with noise alignment for batch matrix multiplication.

Secret sharing schemes based on Lagrange polynomial codes, often called polynomial sharing, are very interesting since it leverages the Lagrange interpolation rule to provide *T*-secure information-theoretic privacy to the input data in both nodes and central server. One remarkable alternative to the previous approach, which also leverages Lagrange polynomials, is Lagrange Coded Computing (LCC), which has been proposed as a unified solution for computing general multivariate polynomials. In comparison to the previously mentioned schemes, LCC reduces the amount of storage, communication and randomness overhead.

There are other relevant works on the field, like a proposal of an optimal linear code for private gradient computation using harmonic sequences [YA19], or another that makes use of cross-subspace alignment (CSA) codes [KL19] to propose a private secure coded computation scheme. Some of these works have already been applied to Federated Learning scenarios. For example, CodedPrivateML [SGA21a], that uses LCC to provide information-theoretic privacy guarantees for both the training dataset and model updates, it enables fast training by distributing the computation load effectively, and it secretly shares the dataset and model parameters using coding techniques. LCC has also been leveraged to break a fundamental quadratic barrier for secure model aggregation in Federated Learning [SGA21b]. Coded Federated Learning (CFL) [PDAAH20] has been proposed to mitigate the impact of stragglers (i.e., users who do not provide timely updates) in Federated Learning, and CodedFedL [PDAYTAH21] enables non-linear federated learning by efficiently exploiting distributed kernel embedding. CSA codes [JJ22] are also considered to address the problem of *X*-secure *T*-private federated learning submodel learning.

Some of the most interesting coding techniques to be applied on Federated Learning for providing input privacy are the previously mentioned polynomial sharing configurations, as they are an

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

extension of secret sharing that can be improved to compute complex functions with a low computational cost. They are usually based on Lagrange interpolation, but this technique has some limitations when used in ML scenarios:

- The recovery threshold depends on the degree of the polynomial computed. This means that operations that increase the polynomial degree, like the multiplication of matrices (it doubles the polynomial degree), increase the recovery threshold required to decode the result.
- It does not allow to compute non-linear functions.
- It relies on quantizing the input data into the finite field, which severely affect the precision of the model.
- It becomes numerically unstable when the number of nodes of the network is too big, or if the input data are rational numbers.

4.2.3 Beyond the State of the Art

For these reasons, we have worked on an alternative coding technique to create a secret sharing configuration that overcomes the limitations of Lagrange interpolation. Our proposal is based on Berrut Approximated Coded Computing (BACC) [JM23], which is a coding scheme that allows to approximate non-linear functions with a bounded error, and it is numerically stable. The main downside of the original scheme is that it does not natively provide any form of input privacy.

The main goal of Berrut original scheme is to evaluate an arbitrary function $f: V \rightarrow U$ over some input data $\mathbf{X} = (\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{K-1})$, maintaining the numerically stability with a bounded error, where U and V are sets of real matrices. This scheme works for a decentralized computing configuration with one master node, owner of the data, and N worker nodes, in charge of computing the goal function.

To perform the encoding step over the input data, the next rational function is composed by the master node:

$$u(z) = \sum_{i=0}^{K-1} \frac{\frac{(-1)^i}{(z-\alpha_i)}}{\sum_{j=0}^{K-1} \frac{(-1)^j}{(z-\alpha_j)}} \boldsymbol{X}_i$$

for some distinct values $\alpha_0, \ldots, \alpha_{K-1} \in \mathbb{R}$. In the original scheme it is suggested to choose these values as Chebyshev points of first kind [Xu16], so $\alpha_j = \cos((2j + 1)\pi/2K)$.

This can also be written as $u(z) = \sum_i W_i \boldsymbol{X}_i = \boldsymbol{W} \bullet \mathbf{X}$ with $\boldsymbol{W} = (W_0, \ldots, W_{K-1})$ and being

$$W_i = \frac{\frac{(-1)^i}{(z-\alpha_i)}}{\sum_{j=0}^{K-1} \frac{(-1)^j}{(z-\alpha_j)}}.$$

The master node assigns $u(z_j)$ to node j, so that this node calculates the goal function f over $u(z_j)$, and retrieves the result $f(u(z_j))$ to the master node. In the original BACC scheme, it is suggested to choose z_j as Chebyshev points of second kind, so $z_j = \cos(j\pi/(N-1))$.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

When the master node has a subset of results $f(u(\tilde{z}_j))$ obtained from the n faster nodes denoted as F, it approximately calculates $f(\boldsymbol{X}_j)$ interpolating the value of the function in the point:

$$f(\boldsymbol{X}_j) = r_B(F) = \sum_{i=0}^{n-1} \frac{\frac{(-1)^i}{(z-\tilde{z}_i)}}{\sum_{j=0}^{n-1} \frac{(-1)^j}{(z-\tilde{z}_j)}} f(u(\tilde{z}_i)) .$$

Our work has consisted in introducing random elements into the encoding step, so the input data remains private while the master node is still able to decode the result. To achieve this, we have altered the rational function $u(z)$ used to encode the data, so it introduces some random input $\boldsymbol{R}$ in the composition, therefore $u'(z) = \boldsymbol{W'} \bullet \mathbf{X} + \boldsymbol{V} \bullet \boldsymbol{R}$, being $\boldsymbol{W'}$ and $\boldsymbol{V}$ the coefficients used to encode the input $\mathbf{X}$ and the randomness $\boldsymbol{R}$ respectively. Both $\boldsymbol{W'}$ and $\boldsymbol{V}$ are constructed using some specific set of points $\alpha'_0, \ldots, \alpha'_{K-1} \in \mathbb{R}$ that guarantees that $u(\alpha'_0) = \boldsymbol{X}_0$, as this is a required condition for the interpolation to be feasible.

The privacy guarantees of our scheme have been measured using the notion of mutual information, calculated as the capacity of a MIMO channel. The results of this privacy analysis have shown that not only is the input privacy reasonable for individual shares, but also that mutual information scales logarithmically with the number of colluding nodes. So, increasing the number of colluding parties does not directly imply that new information is inferred from the input. As a result of this work, we have proposed a new scheme that provides a reasonable level of privacy while keeping similar levels of precision compared to original BACC when the number of parties' results used for decoding is close to the total.

The original and the proposed schemes are designed to be used with a unique input owned by some master node that runs the protocol, which implies that in order to make this scheme suitable for Federated Learning, an extension is needed. Therefore, we have proposed a Secure Multi-Party Computation (SMPC) protocol, similar to polynomial sharing, that leverages our encoding technique to distributively compute a function between different nodes, each having its own private dataset. The protocol is defined with the following phases:

- **Sharing**: Each worker node composes $u'(z)$ to encode their own input data $\boldsymbol{X}^{(j)}$, it evaluates the function using Chebyshev points of second kind and exchanges each evaluation (share) with a different node that participates in the protocol.
- **Computation and Communication**: Each worker node evaluates the goal function f over all the shares received by the rest of the nodes and sends this partial result to the master node in charge of decoding the final result.
- **Decoding**: The master node decodes the final result $f(\boldsymbol{X}^{(0)}, \ldots, \boldsymbol{X}^{(n-1)})$ using the interpolation function r_B.

4.3 Statistical Privacy: SMPC with DP

One of the main problems on the use of central DP is that we must trust the aggregator. Pentyala et al. [PRMDMNC22] propose the use of Secure Multi-Party Computation (SMPC) in combination with DP to address this issue. The combination of SMPC with DP offers other benefits, such as the optimization of the use of the privacy budget by each of the computing parties [KOV15] and each

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

party only needs to know its own desired level of privacy, its own function to be computed, and its measure of accuracy.

In a similar way, Böhler and Kerschbaum [BK20] explore the optimization of the computational time for the exponential mechanism, as well as the reduction of the privacy budget. This is done by recursively splitting the data universe into subranges, thereby achieving sublinear running time in the size of the data universe. They focus on utility functions which allow efficient computation of selection probabilities. Specifically, they study the decomposition of the mean function.

At last, Truex et al. [TBASLZZ19] combines DP with SMPC and Homomorphic Encryption in a federated learning setting. When the aggregator requests the data, performs *k* requests, and each request is distributed to all parties through a secure channel. Each party adds Gaussian noise to the information required, with a privacy budget assigned by the aggregator. Moreover, a homomorphic encryption algorithm is included in this schema, where each client/party has a different secret key. The authors perform a study with different AI algorithms in a federated learning setting, including SVMs, Decision Trees and CNNs, adapting them to the requirements of a SMPC system. Thanks to the reduction of the amount of noise used in homomorphic encryption, better accuracy is achieved than simple DP protocols in FL.

4.3.1 Baseline combinations

In general, the baseline combination of SMPC with DP provides, in terms of privacy, similar results to combining HE with DP, although it usually requires considering an additional non-colluding assumption. With respect to efficiency, SMPC provides solutions with less computational cost, but with a much higher number of communication rounds. Regarding leakages on both the training data and the final model, SMPC suffers from the same issues as HE, and consequently DP can help to mitigate them.

4.3.2 DP meets HE

If we focus now on the use of only HE instead of general SMPC, the same considerations as in SMPC follow. We can actually combine DP mechanisms with HE in a similar way [PAHWM18, GSSG22] by adding noise to the inputs before encryption, also during the homomorphic computation, or even directly to the outputs before decryption. However, more recently, some works are exploring more sophisticated combinations of both PET technologies. Modern HE schemes are based on the RLWE and LWE problems (see Section 3.1.1 for more details), which implicitly add noise during the encryption phase. In [Ogilvie23], the author takes advantage of this internal noise used for encryption, and studies whether approximate HE natively provides DP-guarantees; hence aiming at avoiding the addition of an extra noise component during computation.

While in many HE schemes this noise is removed during decryption, some schemes like CKKS are approximate when doing homomorphic operations, so the result is not exact but slightly perturbed. Unfortunately, this HE noise is usually correlated with the computed operations and the encrypted data itself, requiring a careful analysis on how the noise distribution evolves internally [CCHMOP22]. Even so, [Ogilvie23] does show how approximate HE is able to provide some DP-guarantees for homomorphic ridge regression training using gradient descent. Other works [SZSSG23] seek to relax the accuracy of the homomorphic computation with the aim of not only improving the runtime

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

performance, but also providing DP-guarantees. Moreover, DP has been used in [LMSS22] as a tool to harden the security of approximate HE under certain security model assumptions [LM21].

4.3.3 Types of assumptions in DP

Traditional encrypted computation schemes such as secret sharing or homomorphic encryption do not reveal more information than their respective outputs under their security assumptions. If even the exact output of the intended algorithm could reveal too much, one can combine encrypted computation with statistical privacy, such as differential privacy, computational differential privacy [MPRV09], or one of its variants. As most encrypted computation strategies are only computationally secure, i.e., they assume adversaries have bounded computational power, such combinations usually can only achieve computational differential privacy rather than the information-theoretical notion originally proposed by Dwork [DR14].

4.4 Verification Techniques: Zero Knowledge Proofs

In recent years, many new ZKP systems have been proposed to verify arbitrary computations. Since a large number of state-of-the-art ZKP systems were designed to optimize blockchains that process a huge number of transactions, the blockchain infrastructure was naturally reused to organize verifiable federated learning in a plenty of works [QUG+22]. Such blockchain-based approaches are resistant to participant dropout and suitable for anonymous settings, which are often important properties in large-scale systems where anyone can create an account.

Nevertheless, in a cross-silo FL setting such as we consider in the TRUMPET project [PRP+23], blockchain-based infrastructures become computationally inefficient due to the replication of calculations. In addition, blockchain-based FL inherits existing security issues of blockchain [RJVHM23]: *51%* attacks, forking attacks, double spending, and reentrance attacks on the smart contract, amongst others. On the other hand, many machine learning algorithms can be formulated in terms of aggregation operations and various secure aggregation protocols have been developed to satisfy not only the requirements of privacy but also the specific challenges of security [MOBC23]. Authors pay special attention to the verification of both server (aggregator) and data owner actions, e.g., Brunetta et al. [BTLBM21] developed a non-interactive secure aggregation protocol that includes the verification of the result to prevent malicious server attacks. This protocol was later improved in terms of communication cost by Tsaloli et al. [TLBBM21] who proposed a DEVA approach based on the Σ-Protocol for performing proofs. At the same time, several protocols which rely on ZKP schemes were proposed to handle malicious users. Sabater et al. proposed a verifiable secure aggregation protocol [SBR22] based on Σ-Protocols [AC20] that enables users to prove the correctness of their computations. Lycklama et al. proposed the ROFL [LBVKH23] approach with a comparison of Bulletproof [BBBPWM18] and Groth16 [Groth16] schemes for the construction of norm bounds using range proofs. Bell et al. in [BGLLMRY23] later developed a more efficient protocol based on Schnorr proofs and Bulletproof scheme.

Although these works are carried out in the similar research direction, they do not focus on the specific aspects of our cross-silo FL setting where there are a fixed number of identified parties. In

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

the TRUMPET project we should take into account the possibility of collusion between any participants, large amounts of data, the timeline of providing proofs, the fact the identities or parties is known and other features that are often omitted when considering only the aggregation process.

There is a separate line of research dedicated to ZKPs in the scope of the ML-as-a-Service paradigm. This paradigm enables computationally weak clients to train ML models or to compute predictions via powerful cloud infrastructures. Verification of computations allows to guarantee the integrity both for the training and inference stages. Authors develop various techniques to improve verification process efficiency, for example, Feng et al. in [FQZDZ21] proposed a new quantization algorithm to reduce the number of constraints to prove, Huang et al. in [HWCSHX22] applied random sampling to reduce training proof cost, Kang et al. [KHSS22] described a novel arithmetization optimization for DNN inferences.

Nevertheless, such approaches cannot be directly applied in FL settings. In the ML-as-a-Service paradigm, clients share their data with an untrusted party, expecting to receive a proof that the model is being trained/used correctly. In contrast, in FL, the data remains on the user's side and the model is trained between clients. However, optimizations developed in the scope of this paradigm, can often be reused in an FL setting.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

5 Conclusions

This document has reported the final results of the work done during tasks T2.1 and T2.2 of WP2.

Regarding T2.1, a detailed list of AI models and aggregation methods has been presented. For the selection of the best models and methods, we have taken into account the requirements elicited from the TRUMPET use cases. Next, a review of the most suitable FL frameworks for the implementation of the TRUMPET platform has been presented.

Regarding T2.2, a description of the state of the art of modern PET methods has been included, where we have paid special attention to their individual privacy guarantees, available software tools for implementation, and adequacy for the FL setting. Specifically, the list of studied PET methods comprised Homomorphic Encryption, Secure Multi-Party Computation, Private Coded Distributed Computing, Differential Privacy and Zero-Knowledge Proofs.

Finally, the document has delved into the combination of several PET methods as a means to better fulfil the privacy requirements elicited in D1.1 and D1.3. The outcomes reported here will be used as main inputs for the selection of the methods to be implemented and combined in tasks T2.3, T2.4 and T2.5 of WP2, as well as the determination of the FL framework which will underpin the development of the TRUMPET platform in WP4. They will be also used in the tasks of WP3 for the study of the existing privacy leaks, and the application of the privacy metric.

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

REFERENCES

[1] **[PRP+23]** Alberto Pedrouzo-Ulloa, Jan Ramon, Fernando Pérez-González, Siyanna Lilova, Patrick Duflot, Zakaria Chihani, Nicola Gentili, Paola Ulivi, Mohammad Ashadul Hoque, Twaha Mukammel, Zeev Pritzker, Augustin Lemesle, Jaime Loureiro-Acuña, Xavier Martínez, Gonzalo Jiménez-Balsa. "Introducing the TRUMPET project: TRUstworthy Multi-site Privacy Enhancing Technologies." CSR 2023: 604-611.

[2] **[Lee90]** Alan J. Lee. "U-Statistics: Theory and Practice." Marcel Dekker, New York, 1990.

[3] **[CCB16]** Stéphan Clémençon, Igor Colin, Aurélien Bellet. "Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics." J. Mach. Learn. Res. 17: 76:1-76:36 (2016).

[4] **[HTF09]** Trevor Hastie, Robert Tibshirani, Jerome Friedman. "The Elements of Statistical Learning." Springer Series in Statistics, Springer, 2019.

[5] **[CBSC15]** Igor Colin, Aurélien Bellet, Joseph Salmon, Stéphan Clémençon. "Extending Gossip Algorithms to Distributed Estimation of U-statistics." NIPS 2015: 271-279.

[6] **[DJ13]** Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: FOCS (2013).

[7] **[MSCS19]** Melis, L., Song, C., Cristofaro, E.D., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. S&P (2019).

[8] **[NSH19]** Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. S&P (2019).

[9] **[EFMRT19]** Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K.: Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In: SODA (2019).

[10] **[GKMP20]** Ghazi, B., Kumar, R., Manurangsi, P., Pagh, R.: Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In: ICML (2020).

[11] **[DKMMN06]** Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: EUROCRYPT (2006).

[12] **[CSS12]** Chan, T.H.H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Financial Cryptography (2012).

[13] **[SCRCS11]** Shi, E., Chan, T.H.H., Rieffel, E.G., Chow, R., Song, D.: Privacy-Preserving Aggregation of Time-Series Data. In: NDSS (2011).

[14] **[BIKMMPRSS17]** Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical Secure Aggregation for Privacy-Preserving Machine Learning. In: CCS (2017).

[15] **[JWEG18]** Jayaraman, B., Wang, L., Evans, D., Gu, Q.: Distributed learning without distress: Privacy-preserving empirical risk minimization. In: NeurIPS (2018)

[16] **[SBR22]** Sabater, C., Bellet, A. and Ramon,J.: An Accurate, Scalable and Verifiable Protocol for Federated Differentially Private Averaging. Machine Learning 111(11):4249-4293 (2022).

[17] **[SPHR23]** Sabater, C., Peter, A., Hahn, F. and Ramon, J.: Private sampling with identifiable cheaters. In PETS 2023.

[18] **[CMS11]** Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially Private Empirical Risk Minimization. Journal of Machine Learning Research 12:1069–1109, 2011.

[19] **[KST12]** Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private Convex Empirical Risk Minimization and High-dimensional Regression. In Proceedings of the 25th Annual Conference on Learning Theory, pages 25.1–25.40. JMLR Workshop and Conference Proceedings, 2012.

[20] **[BST14]** Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds, October 2014.

[21] **[ACGMMTZ16]** Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar and Li Zhang. Deep Learning with Differential Privacy. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 308–318, 2016. arXiv: 1607.00133.

[22] **[MMRHA17]** H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data, 2017. arXiv:1602.05629.

[23] **[ATMR22]** Galen Andrew, Om Thakkar, H. Brendan McMahan and Swaroop Ramaswamy. Differentially Private Learning with Adaptive Clipping, 2022. arXiv:1905.03871.

[24] **[KH20]** Antti Koskela and Antti Honkela. Learning Rate Adaptation for Differentially Private Learning. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, pages 2465–2475. PMLR, 2020.

[25] **[LK18]** Jaewoo Lee and Daniel Kifer. Concentrated Differentially Private Gradient Descent with Adaptive per-Iteration Privacy Budget, August 2018. arXiv:1808.09501.

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[26] **[YLPET19]** Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy and Stacey Truex. Differentially Private Model Publishing for Deep Learning. In 2019 IEEE Symposium on Security and Privacy (SP), pages 332–349, May 2019. arXiv:1904.02200.

[27] **[MBST22]** Paul Mangold, Aurélien Bellet, Joseph Salmon and Marc Tommasi. High-Dimensional Private Empirical Risk Minimization by Greedy Coordinate Descent, 2022. arXiv:2207.01560.

[28] **[MKWK19]** Eduardo Morais, Tommy Koens, Cees van Wijk, Aleksei Koren. "A Survey on Zero Knowledge Range Proofs and Applications." CoRR abs/1907.06381 (2019).

[29] **[TensorFlow19]** The TFF Authors. TensorFlow Federated, 2019. URL https://www.tensorflow.org/federated.

[30] **[HLS+20]** Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning, 2020.

[31] **[RTD+18]** Théo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, Jonathan Passerat-Palmbach. "A generic framework for privacy preserving deep learning." CoRR abs/1811.04017 (2018).

[32] **[RSJ+20]** Nuria Rodríguez Barroso, Goran Stipcich, Daniel Jiménez-López, José Antonio Ruiz-Millán, Eugenio Martínez-Cámara, Gerardo González-Seco, María Victoria Luzón, Miguel Angel Veganzones, Francisco Herrera. "Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy." Inf. Fusion 64: 270-292 (2020).

[33] **[RHP+21]** Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, Michael A. Hoeh. "PyVertical: A Vertical Federated Learning Framework for Multi-headed SplitNN." CoRR abs/2104.00489 (2021).

[34] **[CWL+18]** Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, Ameet Talwalkar. "LEAF: A Benchmark for Federated Settings." CoRR abs/1812.01097 (2018).

[35] **[Declearn22]** A. Bellet, P. Andrey, N.Bigaud et al. Declearn, a modular and extensible framework for Federated Learning. 2022, https://gitlab.inria.fr/magnet/declearn/declearn2.

[36] **[FATE19]** The FATE Authors. Federated AI technology enabler, 2019. URL https://www.fedai.org/.

[37] **[PaddleFL19]** The PaddleFL Authors. PaddleFL, 2019. URL https://github.com/PaddlePaddle/PaddleFL.

[38] **[NVIDIAClara19]** NVIDIA Clara. Clara Training Framework, 2019. URL https://developer.nvidia.com/clara.

[39] **[LBT+20]** Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Jeremy Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, Mark Purcell, Ambrish Rawat, Tran Ngoc Minh, Naoise Holohan, Supriyo Chakraborty, Shalisha Witherspoon, Dean Steuer, Laura Wynter, Hifaz Hassan, Sean Laguna, Mikhail Yurochkin, Mayank Agarwal, Ebube Chuba, Annie Abay. "IBM Federated Learning: an Enterprise Framework White Paper V0.1." CoRR abs/2007.10987 (2020).

[40] **[BTM+20]** Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Nicholas D. Lane. "Flower: A Friendly Federated Learning Research Framework." CoRR abs/2007.14390 (2020).

[41] **[Fedlearner20]** The Fedlearner Authors. Fedlearner, 2020. URL https://github.com/bytedance/fedlearner.

[42] **[RGF+21]** G. Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, Jason Martin, Brandon Edwards, Micah J. Sheller, Sarthak Pati, Prakash Narayana Moorthy, Hans Shih-Han Wang, Prashant Shah, Spyridon Bakas. "OpenFL: An open-source framework for Federated Learning." CoRR abs/2105.06413 (2021).

[43] **[SubstraFL23]** The SubstraFL Authors. SubstraFL. 2023, URL https://docs.substra.org/en/stable/.

[44] **[CVC+23]** Francesco Cremonesi, Marc Vesin, Sergen Cansiz, Yannick Bouillard, Irene Balelli, Lucia Innocenti, Santiago S. Silva R., Samy-Safwan Ayed, Riccardo Taiello, Laetitia Kameni, Richard Vidal, Fanny Orlhac, Christophe Nioche, Nathan Lapel, Bastien Houis, Romain Modzelewski, Olivier Humbert, Melek Önen, Marco Lorenzi. "Fed-BioMed: Open, Transparent and Trusted Federated Learning for Real-world Healthcare Applications." CoRR abs/2304.12012 (2023). URL https://fedbiomed.gitlabpages.inria.fr/.

[45] **[RAD78]** R. Rivest, L. Adleman, and M. Dertouzos. "On Data Banks and Privacy Homomorphisms." Foundations of Secure Computation, Academia Press (1978).

[46] **[Paillier99]** Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". EUROCRYPT. Springer. pp. 223–238, 1999.

[47] **[ElGamal85]** Taher ElGamal. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." IEEE Transactions on Information Theory. 31 (4): 469–472, 1985.

[48] **[Gentry09]** Craig Gentry. "A fully homomorphic encryption scheme." PhD thesis, Standford University, 2009.

[49] **[Regev09]** Oded Regev: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6): 34:1-34:40 (2009).

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[50] **[LPR13]** Vadim Lyubashevsky, Chris Peikert, Oded Regev: On Ideal Lattices and Learning with Errors over Rings. J. ACM 60(6): 43:1-43:35 (2013).

[51] **[SKN+20]** Imtiyazuddin Shaik, Ajeet Kumar Singh, Harika Narumanchi, Nitesh Emmadi, Rajan Mindigal Alasingara Bhattachar: A Recommender System for Efficient Implementation of Privacy Preserving Machine Learning Primitives Based on FHE. CSCML 2020: 193-218.

[52] **[PTP17]** Alberto Pedrouzo-Ulloa, Juan Ramón Troncoso-Pastoriza, Fernando Pérez-González. "Number Theoretic Transforms for Secure Signal Processing." IEEE Trans. Inf. Forensics Secur. 12(5): 1125-1140 (2017).

[53] **[BPNB23]** Iván Blanco-Chacón, Alberto Pedrouzo-Ulloa, Rahinatou Yuh Njah, Beatriz Barbero-Lucas. "Fast polynomial arithmetic in homomorphic encryption with cyclo-multiquadratic fields." CoRR abs/2304.04619 (2023).

[54] **[ACC+19]** Martin R. Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E. Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, Vinod Vaikuntanathan. "Homomorphic Encryption Standard." IACR Cryptol. ePrint Arch. 2019: 939 (2019).

[55] **[APS15]** Martin R. Albrecht, Rachel Player and Sam Scott. "On the concrete hardness of Learning with Errors." Journal of Mathematical Cryptology. Volume 9, Issue 3, Pages 169–203, ISSN (Online) 1862-2984, October 2015.

[56] **[BGV12]** Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping." In ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. Pages 309-325, 2012.

[57] **[Brakerski12]** Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP." In CRYPTO 2012. Pages 868-886, 2012.

[58] **[FV12]** J. Fan and F. Vercauteren. "Somewhat practical fully homomorphic encryption." Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.pdf.

[59] **[CKKS17]** Jung Hee Cheon, Andrey Kim, Miran Kim, Yongsoo Song. "Homomorphic encryption for arithmetic of approximate numbers." In International Conference on the Theory and Applications of Cryptology and Information Security, pp. 409–437.Springer, Cham. 2017.

[60] **[CGGI16]** I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. "Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016." LNCS, vol. 10031, pp. 3–33, 2016.

[61] **[SEAL]** Microsoft SEAL. URL https://github.com/microsoft/SEAL.

[62] **[HElib]** HElib library. URL https://github.com/shaih/HElib.

[63] **[HEAAN]** HEAAN library. URL https://github.com/snucrypto/HEAAN.

[64] **[PALISADE]** PALISADE Homomorphic Encryption Software library. URL https://palisade-crypto.org/.

[65] **[Lattigo]** Lattigo: A library for lattice-based multiparty homomorphic encryption in Go. URL https://github.com/tuneinsight/lattigo.

[66] **[FV-NFLlib]** FV-NFLlib library. URL https://github.com/CryptoExperts/FV-NFLlib.

[67] **[TFHE]** TFHE: Fast Fully Homomorphic Encryption over the Torus. URL https://tfhe.github.io/tfhe/.

[68] **[Concrete]** Concrete: TFHE Compiler that converts python programs into FHE equivalent. URL https://github.com/zama-ai/concrete.

[69] **[OpenFHE]** OpenFHE - Open-Source Fully Homomorphic Encryption Library. URL https://github.com/openfheorg/openfhe-development.

[70] **[TFHE-rs]** TFHE-rs: A Pure Rust implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data. URL https://github.com/zama-ai/tfhe-rs.

[71] **[cuHE]** cuHE: CUDA Homomorphic Encryption Library. URL https://github.com/vernamlab/cuHE.

[72] **[DSC+19]** Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madanlal Musuvathi, Todd Mytkowicz. "CHET: an optimizing compiler for fully-homomorphic neural-network inferencing." PLDI 2019: 142-156.

[73] **[GLN12]** Thore Graepel, Kristin E. Lauter, Michael Naehrig. "ML Confidential: Machine Learning on Encrypted Data." ICISC 2012: 1-21.

[74] **[WZD+16]** Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin E. Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, Xiaoqian Jiang. "HEALER: homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS." Bioinform. 32(2): 211-218 (2016).

[75] **[KSKLC18]** Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, Jung Hee Cheon. "Logistic Regression Model Training based on the Approximate Homomorphic Encryption." IACR Cryptol. ePrint Arch. 2018: 254 (2018).

[76] **[CGGT19]** Sergiu Carpov, Nicolas Gama, Mariya Georgieva, Juan Ramón Troncoso-Pastoriza. "Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption." IACR Cryptol. ePrint Arch. 2019: 101 (2019).

[77] **[GDLLNW16]** Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, John Wernsing. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy." ICML 2016: 201-210.

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[78] **[PBLCL20]** Saerom Park, Junyoung Byun, Joohee Lee, Jung Hee Cheon, Jaewook Lee. "HE-Friendly Algorithm for Privacy-Preserving SVM Training." IEEE Access 8: 57414-57425 (2020).

[79] **[LPR10]** Vadim Lyubashevsky, Chris Peikert, Oded Regev: On Ideal Lattices and Learning with Errors over Rings. EUROCRYPT 2010: 1-23.

[80] **[Yao82]** Andrew C. Yao. "Protocols for secure computations." 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982): 160–164.

[81] **[GMW87]** Oded Goldreich, Silvio Micali, Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority." STOC 1987: 218-229.

[82] **[EKR18]** David Evans, Vladimir Kolesnikov, Mike Rosulek. "A Pragmatic Introduction to Secure Multi-Party Computation." Found. Trends Priv. Secur. 2(2-3): 70-246 (2018).

[83] **[BMR90]** D. Beaver, S. Micali, P. Rogaway. "The Round Complexity of Secure Protocols." Proceeding STOC '90 Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing. pp. 503-513, 1990.

[84] **[Kilian88]** Joe Kilian. "Founding Cryptography on Oblivious Transfer." STOC 1988: 20-31.

[85] **[Shamir79]** A. Shamir. "How to share a secret.Communications of the ACM." 22(11):612–613, 1979.

[86] **[MNPS04]** Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella. "Fairplay - Secure Two-Party Computation System." USENIX Security Symposium 2004: 287-302.

[87] **[BCD+09]** P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, T. Toft. "Secure multiparty computation goes live." Financial Cryptography and Data Security, Lecture Notes in Computer Science,5628, pp. 325–343. Springer.

[88] **[Beaver91]** Donald Beaver: Efficient Multiparty Protocols Using Circuit Randomization. CRYPTO 1991: 420-432.

[89] **[DPSZ12]** I. Damgard, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In Advances in Cryptology – CRYPTO 2012, volume 7417 of LNCS, pp643–662. Springer, 2012.

[90] **[KOS16]** Marcel Keller, Emmanuela Orsini, Peter Scholl: MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer. ACM Conference on Computer and Communications Security 2016: 830-842.

[91] **[KPR18]** Marcel Keller, Valerio Pastro, Dragos Rotaru: Overdrive: Making SPDZ Great Again. EUROCRYPT (3) 2018: 158-189.

[92] **[CDESX18]** Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, Chaoping Xing: SPDℤ2k: Efficient MPC mod 2k for Dishonest Majority. CRYPTO (2) 2018: 769-798.

[93] **[MPCalliance]** MPC Alliance. URL https://www.mpcalliance.org/.

[94] **[Sharemind]** Sharemind. URL https://sharemind.cyber.ee/.

[95] **[Sepior]** Sepior. URL https://sepior.com/.

[96] **[Zcash]** Zcash. URL https://z.cash/technology/paramgen/.

[97] **[Unbound]** Unbound Technology. URL https://www.unboundtechnology.com/.

[98] **[Libscapi]** LIBSCAPI - The Secure Computation API. URL https://github.com/cryptobiu/libscapi.

[99] **[Scale-Mamba]** SCALE-MAMBA Software. URL https://homes.esat.kuleuven.be/~nsmart/SCALE/.

[100] **[Keller20]** Marcel Keller: MP-SPDZ: A Versatile Framework for Multi-Party Computation. IACR Cryptol. ePrint Arch. 2020: 521 (2020).

[101] **[Jana]** Jana: Private Data as a Service. URL https://galois.com/project/jana-private-data-as-a-service/.

[102] **[TPMPC]** Theory and Practice of Multi-Party Computation Workshop website. URL https://www.multipartycomputation.com/home.

[103] **[MR18]** P. Mohassel, P. Rindal. "ABY3: A mixed protocol framework for machine learning." In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 35-52), 2018.

[104] **[WGC18]** S. Wagh, D. Gupta, N. Chandran. "SecureNN: Efficient and Private Neural Network Training." IACR Cryptology ePrint Archive, 2018, 442.

[105] **[ASKG19]** N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, A. Gascón. "QUOTIENT: two-party secure neural network training and prediction." In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 1231–1247), 2019.

[106] **[JVC18]** Chiraag Juvekar, Vinod Vaikuntanathan, Anantha Chandrakasan. "GAZELLE: A Low Latency Framework for Secure Neural Network Inference." USENIX Security Symposium 2018: 1651-1669.

[107] **[MLSZP20]** Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, Raluca Ada Popa. "Delphi: A Cryptographic Inference Service for Neural Networks." USENIX Security Symposium 2020: 2505-2522.

[108] **[UAGJTT21]** Sennur Ulukus, Salman Avestimehr, Michael Gastpar, Syed Ali Jafar, Ravi Tandon, Chao Tian. "Private Retrieval, Computing and Learning: Recent Progress and Future Challenges." CoRR abs/2108.00026 (2021).

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[109] **[LLPPR18]** K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, K. Ramchandran. "Speeding up distributed machine learning using codes." IEEE Transactions on Information Theory, vol. 64, no. 3, pp. 1514-1529, 2018.
[110] **[TLDK16]** Rashish Tandon, Qi Lei, Alexandros G. Dimakis, Nikos Karampatziakis. "Gradient Coding." CoRR abs/1612.03301 (2016).
[111] **[YLMA17]** Qian Yu, Songze Li, Mohammad Ali Maddah-Ali, Amir Salman Avestimehr. "How to optimally allocate resources for coded distributed computing?" ICC 2017: 1-7.
[112] **[DR14]** Cynthia Dwork, Aaron Roth. "The Algorithmic Foundations of Differential Privacy." Found. Trends Theor. Comput. Sci. 9(3-4): 211-407 (2014).
[113] **[APB+23]** David W. Archer, Borja de Balle Pigem, Dan Bogdanov, Mark Craddock, Adrià Gascón, Ronald Jansen, Matjaz Jug, Kim Laine, Robert McLellan, Olga Ohrimenko, Mariana Raykova, Andrew Trask, Simon Wardley. "UN Handbook on Privacy-Preserving Computation Techniques." CoRR abs/2301.06167 (2023).
[114] **[AppleDP]** Differential Privacy Overview, Apple. URL https://images.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.
[115] **[KMRTZ20]** Daniel Kifer, Solomon Messing, Aaron Roth, Abhradeep Thakurta, Danfeng Zhang. "Guidelines for Implementing and Auditing Differentially Private Systems." CoRR abs/2002.04049 (2020).
[116] **[DKY17]** Bolin Ding, Janardhan Kulkarni, Sergey Yekhanin. "Collecting Telemetry Data Privately." NIPS 2017: 3571-3580.
[117] **[GoogleDP]** Google's Differential Privacy library. URL https://github.com/google/differential-privacy.
[118] **[HarvardDP]** Harvard University Privacy Tools Project. URL https://privacytools.seas.harvard.edu/differential-privacy.
[119] **[Census22]** 2022 Decennial Census. Processing the Count: Disclosure Avoidance Modernization. URL https://www.census.gov/about/policies/privacy/statistical_safeguards/disclosure-avoidance-2020-census.html.
[120] **[AC19]** Mohammad Al-Rubaie, J. Morris Chang. "Privacy-Preserving Machine Learning: Threats and Solutions." IEEE Secur. Priv. 17(2): 49-58 (2019).
[121] **[WLDMYFJQP20]** Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, H. Vincent Poor. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis." IEEE Trans. Inf. Forensics Secur. 15: 3454-3469 (2020).
[122] **[Mironov17]** Ilya Mironov. "Rényi Differential Privacy." CSF 2017: 263-275.
[123] **[Renyi61]** A. Rényi. "On measures of entropy and information." In Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics (Vol. 4, pp. 547-562). University of California Press, 1961.
[124] **[DRS19]** Jinshuo Dong, Aaron Roth, Weijie J. Su. "Gaussian Differential Privacy." CoRR abs/1905.02383 (2019).
[125] **[DPgoogle22]** Differential Privacy Team Google. "Privacy Loss Distributions." 2022. URL https://raw.githubusercontent.com/google/differential-privacy/main/common_docs/Privacy_Loss_Distributions.pdf.
[126] **[DR16]** Cynthia Dwork, Guy N. Rothblum. "Concentrated Differential Privacy." CoRR abs/1603.01887 (2016).
[127] **[WBP19]** Yu-Xiang Wang, Borja Balle, Shiva Prasad Kasiviswanathan. "Subsampled Renyi Differential Privacy and Analytical Moments Accountant." AISTATS 2019: 1226-1235.
[128] **[GRS09]** Arpita Ghosh, Tim Roughgarden, Mukund Sundararajan. "Universally utility-maximizing privacy mechanisms." STOC 2009: 351-360.
[129] **[MT07]** Frank McSherry, Kunal Talwar. "Mechanism Design via Differential Privacy." FOCS 2007: 94-103.
[130] **[ZZYWWLNL21]** Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, Kwok-Yan Lam. "Local Differential Privacy-Based Federated Learning for Internet of Things." IEEE Internet Things J. 8(11): 8836-8853 (2021).
[131] **[WXYZHSSY19]** Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, Ge Yu. "Collecting and Analyzing Multidimensional Data with Local Differential Privacy." ICDE 2019: 638-649.
[132] **[AKL21]** Naman Agarwal, Peter Kairouz, Ziyu Liu. "The Skellam Mechanism for Differentially Private Federated Learning." NeurIPS 2021: 5052-5064.
[133] **[BDFKR18]** Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, Ryan Rogers. "Protection Against Reconstruction and Its Applications in Private Federated Learning." CoRR abs/1812.00984 (2018).
[134] **[HGLPG20]** Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, Yanmin Gong. "Personalized Federated Learning With Differential Privacy." IEEE Internet Things J. 7(10): 9530-9539 (2020).
[135] **[LLSY17]** N. Li, M. Lyu, D. Su, W. Yang. "The Sparse Vector Technique." In Differential Privacy: From Theory to Practice (pp. 93-112), 2017. Cham: Springer International Publishing.
[136] **[LSL17]** Min Lyu, Dong Su, Ninghui Li. "Understanding the Sparse Vector Technique for Differential Privacy." Proc. VLDB Endow. 10(6): 637-648 (2017).
[137] **[SS15]** Reza Shokri, Vitaly Shmatikov. "Privacy-Preserving Deep Learning." CCS 2015: 1310-1321.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[138] **[LJLA17]** J. Liu, M. Juuti, Y. Lu, N. Asokan. "Oblivious neural network predictions via minionn transformations." In Conference on Computer and Communications Security (pp. 619–631), 2017.

[139] **[MTP20]** Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Pierre Hubaux: Multiparty Homomorphic Encryption: From Theory to Practice. IACR Cryptol. ePrint Arch. 2020: 304 (2020).

[140] **[AHSL22]** Asma Aloufi, Peizhao Hu, Yongsoo Song, Kristin E. Lauter: Computing Blindfolded on Data Homomorphically Encrypted under Multiple Keys: A Survey. ACM Comput. Surv. 54(9): 195:1-195:37 (2022).

[141] **[FTPSSBH20]** D. Froelicher, J. R. Troncoso-Pastoriza, A. Pyrgelis, S. Sav, J. S. Sousa, J.-P. Bossuat, and J.-P. Hubaux. "Scalable privacy-preserving distributed learning." CoRR,abs/2005.09532 (2020).

[142] **[PPV22]** Alberto Pedrouzo-Ulloa, Fernando Pérez-González, David Vázquez-Padín: Secure Collaborative Camera Attribution. EICC 2022: 97-98.

[143] **[PPV22HE]** Alberto Pedrouzo-Ulloa, Fernando Pérez-González, David Vázquez-Padín: Multi-Key Homomorphic Encryption for Collaborative Camera Attribution. Poster presentation. 5th HomomorphicEncryption.org Standards Meeting 2022.

[144] **[TCLRLB16]** Jun Tang, Yong Cui, Qi Li, Kui Ren, Jiangchuan Liu, Rajkumar Buyya: Ensuring Security and Privacy Preservation for Cloud Data Services. ACM Comput. Surv. 49(1): 13:1-13:39 (2016).

[145] **[Bongenaar16]** Elena Fuentes Bongenaar: Multi-key fully homomorphic encryption report. Retrieved from https://www.cse.chalmers.se/~elenap/papers/Multi-key%20fully%20homomorphic%20encryption%20report.pdf.

[146] **[AH19]** Asma Aloufi, Peizhao Hu: Collaborative Homomorphic Computation on Data Encrypted under Multiple Keys. CoRR abs/1911.04101 (2019).

[147] **[DF89]** Yvo Desmedt, Yair Frankel: Threshold Cryptosystems. CRYPTO 1989: 307-315.

[148] **[DHMR07]** Vanesa Daza, Javier Herranz, Paz Morillo, Carla Ràfols: CCA2-Secure Threshold Broadcast Encryption with Shorter Ciphertexts. ProvSec 2007: 35-50.

[149] **[CGS97]** Ronald Cramer, Rosario Gennaro, Berry Schoenmakers: A Secure and Optimally Efficient Multi-Authority Election Scheme. EUROCRYPT 1997: 103-118.

[150] **[RSTVW22]** Dragos Rotaru, Nigel P. Smart, Titouan Tanguy, Frederik Vercauteren, Tim Wood: Actively Secure Setup for SPDZ. J. Cryptol. 35(1): 5 (2022).

[151] **[MTBH21]** Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Philippe Bossuat, Jean-Pierre Hubaux. "Multiparty Homomorphic Encryption from Ring-Learning-with-Errors." Proc. Priv. Enhancing Technol. 2021(4): 291-311 (2021).

[152] **[Pedersen91]** Torben P. Pedersen: A Threshold Cryptosystem without a Trusted Party (Extended Abstract). EUROCRYPT 1991: 522-526.

[153] **[DH76]** W. Diffie and M. E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6):644–654, 1976.

[154] **[BGGJKR18]** Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, Amit Sahai: Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. CRYPTO (1) 2018: 565-596.

[155] **[CDN01]** Ronald Cramer, Ivan Damgård, Jesper Buus Nielsen: Multiparty Computation from Threshold Homomorphic Encryption. EUROCRYPT 2001: 280-299.

[156] **[AJLTVW12]** Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, Daniel Wichs: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. EUROCRYPT 2012: 483-501.

[157] **[MBH22]** Christian Mouchet, Elliott Bertrand, Jean-Pierre Hubaux: An Efficient Threshold Access-Structure for RLWE-Based Multiparty Homomorphic Encryption. IACR Cryptol. ePrint Arch. 2022: 780 (2022).

[158] **[LTV12]** Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. STOC 2012: 1219-1234.

[159] **[CM15]** Michael Clear, Ciaran McGoldrick: Multi-identity and Multi-key Leveled FHE from Learning with Errors. CRYPTO (2) 2015: 630-656.

[160] **[GSW13]** Craig Gentry, Amit Sahai, Brent Waters: Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. CRYPTO (1) 2013: 75-92.

[161] **[MW16]** Pratyay Mukherjee, Daniel Wichs: Two Round Multiparty Computation via Multi-key FHE. EUROCRYPT (2) 2016: 735-763.

[162] **[PS16]** Chris Peikert, Sina Shiehian: Multi-key FHE from LWE, Revisited. TCC (B2) 2016: 217-238.

[163] **[BP16]** Zvika Brakerski, Renen Perlman: Lattice-Based Fully Dynamic Multi-key FHE with Short Ciphertexts. CRYPTO (1) 2016: 190-213.

[164] **[CZW17]** Long Chen, Zhenfeng Zhang, Xueqing Wang: Batched Multi-hop Multi-key FHE from Ring-LWE with Compact Ciphertext Extension. TCC (2) 2017: 597-627.

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[165] **[LZYHLT19]** Ningbo Li, Tanping Zhou, Xiaoyuan Yang, Yiliang Han, Wenchao Liu, Guangsheng Tu: Efficient Multi-Key FHE With Short Extended Ciphertexts and Directed Decryption Protocol. IEEE Access 7: 56724-56732 (2019).

[166] **[CCS19]** Hao Chen, Ilaria Chillotti, Yongsoo Song: Multi-Key Homomorphic Encryption from TFHE. ASIACRYPT (2) 2019: 446-472.

[167] **[CDKS19]** Hao Chen, Wei Dai, Miran Kim, Yongsoo Song: Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference. CCS 2019: 395-412.

[168] **[YKHK18]** Satoshi Yasuda, Yoshihiro Koseki, Ryo Hiromasa, Yutaka Kawai: Multi-key Homomorphic Proxy Re-Encryption. ISC 2018: 328-346.

[169] **[DHRW16]** Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, Daniel Wichs: Spooky Encryption and Its Applications. CRYPTO (3) 2016: 93-122.

[170] **[PBCSSZ23-HES]** A. Pedrouzo-Ulloa, A. Boudguiga, O. Chakraborty, R. Sirdey, O. Stan, M. Zuber, “Practical Multi-Key Homomorphic Encryption for Efficient Secure Federated Average Aggregation,” Poster presentation at the 6th HomomorphicEncryption.org Standards Meeting, Seoul, South Korea, 2023, https://gpsc.uvigo.es/sites/default/files/slides/PBCSSZ-HES2023.pdf.

[171] **[MSMGGS21]** A. Madi, O. Stan, A. Mayoue, A. Grivet-Sébert, C. Gouy-Pailler and R. Sirdey, "A Secure Federated Learning framework using Homomorphic Encryption and Verifiable Computing," 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), 2021, pp. 1-8.

[172] **[GSSG22]** A. Grivet-Sébert, R. Sirdey, O. Stan, and C. Gouy-Pailler, “Protecting data from all parties: Combining FHE and DP in federated learning,” CoRR, vol. abs/2205.04330, 2022.

[173] **[SPTFBSP21]** S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. Sa Sousa and J.-P. Hubaux, "POSEIDON: Privacy-Preserving Federated Neural Network Learning," NDSS 2021.

[174] **[DCESBPTBH23]** D. Froelicher, H. Cho, M. Edupalli, J. Sa Sousa, J.-P. Bossuat, A. Pyrgelis, J. R. Troncoso-Pastoriza, B. Berger, J.-P. Hubaux, "Scalable and Privacy-Preserving Federated Principal Component Analysis," IEEE Symposium on Security and Privacy, 2023: 1908-1925.

[175] **[PBCSSZ23]** A. Pedrouzo-Ulloa, A. Boudguiga, O. Chakraborty, R. Sirdey, O. Stan and M. Zuber, "Practical multi-key homomorphic encryption for more flexible and efficient secure federated average aggregation", Proceedings of the 2023 IEEE CSR Workshop on Privacy-Preserving Data Processing and Analysis, 2023.

[176] **[DFKNT06]** I. Damgard, M. Fitzi, E. Kiltz J.B. Nielsen and T. Toft. "Unconditionally secure constant rounds multi-party computation for equality, comparison, bits and exponentiation." TCC 2006, LNCS 3876, 285–304, 2006.

[177] **[BPR20]** R. Bitar, P. Parag, S. El Rouayheb. "Minimizing latency for secure coded computing using secret sharing via staircase codes." IEEE Transactions on Communications, vol. 68, no. 8, pp. 4609–4619, 2020.

[178] **[NNM18]** H. A. Nodehi, S. R. H. Najarkolaei, M. A. Maddah-Ali. "Entangled polynomial coding in limited-sharing multi-party computation." In 2018 IEEE Information Theory Workshop (ITW), Nov 2018, pp.1–5.

[179] **[NM21]** Hanzaleh Akbari Nodehi, Mohammad Ali Maddah-Ali. "Secure Coded Multi-Party Computation for Massive Matrix Operations." IEEE Trans. Inf. Theory 67(4): 2379-2398 (2021).

[180] **[CJWJ21]** Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar. "GCSA codes with noise alignment for secure coded multi-party batch matrix multiplication." IEEE Journal on Selected Areas in Information Theory, vol. 2, no. 1, pp. 306–316, 2021.

[181] **[YA19]** Q. Yu and A. S. Avestimehr. "Harmonic coding: An optimal linear code for privacy-preserving gradient-type computation." In 2019 IEEE International Symposium on Information Theory (ISIT), July 2019, pp.1102–1106.

[182] **[KL19]** Minchul Kim, Jungwoo Lee. "Private Secure Coded Computation." IEEE Commun. Lett. 23(11): 1918-1921 (2019).

[183] **[SGA21a]** J. So, B. Guler, and A. S. Avestimehr. "Codedprivateml: A fast and privacy-preserving framework for distributed machine learning." IEEE Journal on Selected Areas in Information Theory, vol. 2, no. 1, pp. 441–451, 2021.

[184] **[SGA21b]** J. So, B. Guler, and A. S. Avestimehr. "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning." IEEE Journal on Selected Areas in Information Theory, vol. 2, no. 1, pp. 479–489, 2021.

[185] **[PDAAH20]** Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Amir Salman Avestimehr, Nageen Himayat. "Coded Computing for Federated Learning at the Edge." CoRR abs/2007.03273 (2020).

[186] **[PDAYTAH21]** Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Yair Yona, Shilpa Talwar, Salman Avestimehr, Nageen Himayat. "Coded Computing for Low-Latency Federated Learning Over Wireless Edge Networks." IEEE J. Sel. Areas Commun. 39(1): 233-250 (2021).

[187] **[JJ22]** Zhuqing Jia, Syed Ali Jafar. "X-Secure T-Private Federated Submodel Learning With Elastic Dropout Resilience." IEEE Trans. Inf. Theory 68(8): 5418-5439 (2022).

[188] **[JM23]** Tayyebeh Jahani-Nezhad, Mohammad Ali Maddah-Ali. "Berrut Approximated Coded Computing: Straggler Resistance Beyond Polynomial Computing." IEEE Trans. Pattern Anal. Mach. Intell. 45(1): 111-122 (2023).

[189] **[Xu16]** Kuan Xu. "The Chebyshev points of the first kind." Applied Numerical Mathematics. Volume 102, April 2016, Pages 17-30.

Funded by the European Union
Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038

[190] **[PRMDMNC22]** Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson C. A. Nascimento, Martine De Cock. "Training Differentially Private Models with Secure Multiparty Computation." CoRR abs/2202.02625 (2022).

[191] **[KOV15]** Peter Kairouz, Sewoong Oh, Pramod Viswanath. "Secure Multi-party Differential Privacy." NIPS 2015: 2008-2016.

[192] **[BK20]** Jonas Böhler, Florian Kerschbaum. "Secure Multi-party Computation of Differentially Private Median." USENIX Security Symposium 2020: 2147-2164.

[193] **[TBASLZZ19]** S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou. "A hybrid approach to privacy-preserving federated learning." In Proceedings of the 12th ACM workshop on artificial intelligence and security (pp. 1-11), 2019.

[194] **[PAHWM18]** Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption," IEEE Trans. Inf. Forensics Secur. 13(5): 1333-1345, 2018.

[195] **[Ogilvie23]** Tabitha Ogilvie, "Differential Privacy for Free? Harnessing the Noise in Approximate Homomorphic Encryption," IACR Cryptol. ePrint Arch. 2023: 701, 2023.

[196] **[CCHMOP22]** Anamaria Costache, Benjamin R. Curtis, Erin Hales, Sean Murphy, Tabitha Ogilvie, Rachel Player, "On the precision loss in approximate homomorphic encryption," IACR Cryptol. ePrint Arch. 2022: 162, 2022.

[197] **[SZSSG23]** Arnaud Grivet Sébert, Martin Zuber, Oana Stan, Renaud Sirdey, Cédric Gouy-Pailler, "When approximate design for fast homomorphic computation provides differential privacy guarantees," CoRR abs/2304.02959, 2023.

[198] **[LMSS22]** Baiyu Li, Daniele Micciancio, Mark Schultz, Jessica Sorrell, "Securing Approximate Homomorphic Encryption Using Differential Privacy," CRYPTO (1) 2022: 560-589.

[199] **[LM21]** Baiyu Li, Daniele Micciancio, "On the Security of Homomorphic Encryption on Approximate Numbers," EUROCRYPT (1) 2021: 648-677.

[200] **[MPRV09]** Ilya Mironov, Omkant Pandey, Omer Reingold, Salil P. Vadhan. "Computational Differential Privacy." CRYPTO 2009: 126-142.

[201] **[QUG+22]** Youyang Qu, Md Palash Uddin, Chenquan Gan, Yong Xiang, Longxiang, Gao, and John Yearwood. "Blockchain-enabled federated learning: A survey." ACM Comput. Surv., 55(4), nov 2022.

[202] **[RJVHM23]** Nuria Rodríguez Barroso, Daniel Jiménez-López, María Victoria Luzón, Francisco Herrera, Eugenio Martínez-Cámara. "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges." Inf. Fusion 90: 148-173 (2023).

[203] **[MOBC23]** Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, Mauro Conti. "SoK: Secure Aggregation Based on Cryptographic Schemes for Federated Learning." In PETS 2023, 23rd Privacy Enhancing Technologies Symposium, 10-14 July 2023, Lausanne, Switzerland (Hybrid Conference), Lausanne, 2023.

[204] **[BTLBM21]** Carlo Brunetta, Georgia Tsaloli, Bei Liang, Custavo Banegas, Aikaterini Mitrokotsa. "Non-interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning." ACISP 2021: 510-528.

[205] **[TLBBM21]** Georgia Tsaloli, Bei Liang, Carlo Brunetta, Gustavo Banegas, Aikaterini Mitrokotsa. "sf DEVA: Decentralized, Verifiable Secure Aggregation for Privacy-Preserving Learning." ISC 2021: 296-319.

[206] **[AC20]** Thomas Attema, Ronald Cramer. "Compressed $\varSigma $-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics." CRYPTO (3) 2020: 513-543.

[207] **[LBVKH23]** Hidde Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, Anwar Hithnawi. "RoFL: Robustness of Secure Federated Learning." SP 2023: 453-476.

[208] **[BBBPWM18]** Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, Gregory Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More." IEEE Symposium on Security and Privacy 2018: 315-334.

[209] **[Groth16]** Jens Groth. "On the size of pairing-based non-interactive arguments." In Advances in Cryptology – EUROCRYPT 2016, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[210] **[BGLLMRY23]** James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, Cathie Yun. "ACORN: Input Validation for Secure Aggregation." USENIX Security Symposium 2023.

[211] **[FQZDZ21]** Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. "Zen: An optimizing compiler for verifiable, zero-knowledge neural network inferences." Cryptology ePrint Archive, Paper 2021/087, 2021.

[212] **[HWCSHX22]** Chenyu Huang, Jianzong Wang, Huangxun Chen, Shijing Si, Zhangcheng Huang, and Jing Xiao. "zkMLaas: a verifiable scheme for machine learning as a service." In GLOBECOM 2022 - 2022 IEEE Global Communications Conference, pages 5475–5480, 2022.

[213] **[KHSS22]** Daniel Kang, Tatsunori Hashimoto, Ion Stoica, Yi Sun. "Scaling up Trustless DNN Inference with Zero-Knowledge Proofs." CoRR abs/2210.08674 (2022).

Funded by the European Union

Trumpet project has received funding from a Research and Innovation action activity under Horizon Europe Framework Programme with Grant Agreement No.101070038