Expert Systems With Applications 235 (2024) 121139

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

SecFL – Secure Federated Learning Framework for predicting defects in sheet metal forming under variability

Mario Alberto da Silveira Dib [a,*], Pedro Prates [b,c,d], Bernardete Ribeiro [e]

[a] Centre for Informatics and Systems of the University of Coimbra (CISUC), Coimbra, Portugal
[b] Centre for Mechanical Technology and Automation (TEMA), Department of Mechanical Engineering, University of Aveiro, Aveiro, Portugal
[c] Intelligent Systems Associate Laboratory (LASI), Portugal
[d] Centre for Mechanical Engineering, Materials and Processes (CEMMPRE), ARISE, Department of Mechanical Engineering, University of Coimbra, Coimbra, Portugal
[e] Centre for Informatics and Systems of the University of Coimbra (CISUC), Department of Informatics Engineering, Coimbra, Portugal

ARTICLE INFO

Keywords:
Federated Learning
Digital Envelopes
Machine Learning
Defect Prediction
Data Poisoning
Sheet Metal Forming

ABSTRACT

With the ongoing digitization of the manufacturing industry and the ability to bring together data from specific manufacturing processes, there is enormous potential to use machine learning (ML) techniques to improve such processes. In this context, the competitive automotive industry can take advantage of the ML power by predicting defects before they occur, aiming to reduce the scrap rate and increase the robustness and reliability of the production processes. In a real world scenario, small and medium size companies do not have the amount of data the big companies have, which can prevent the usage of ML models in this vital niche for the industry. A collaboration in terms of data usage to develop powerful and general industry solutions is hindered by data privacy concerns despite similar problems. This paper addresses these concerns by providing a framework based on the Federated Learning (FL) method combined with Digital Envelopes (DE) to allow the ML models training while keeping the data of the partners and the models parameters private and protected against external cyber-attacks, which is one of the weaknesses of FL as of now. A case study was carried out to demonstrate the effectiveness of the proposed framework on handling data poisoning attacks to the training data and also the models' weights.

1. Introduction

Sheet metal forming process is one of the necessary processes used in the automotive industry to produce car components. In this process, metal sheets are plastically deformed into a desired shape by the action of forming tools, which typically consist of a punch, a die, and a blank holder. Firstly, a blank (i.e., non-deformed metal sheet) is placed over the die, and then it is pressed into the die by the motion of the punch to obtain the desired shape; the flow of the sheet material into the die is typically controlled with a blank holder (Hattalli & Srivatsa, 2020). In general, sheet metal forming processes allow obtaining high quality components with high cadence and low cost; however, the variability inherent to mechanical properties, tool geometry and process parameters makes formed components often prone to defects such as wrinkling, tearing, excessive thinning and springback.

Forming processes, in general, are often susceptible to the occurrence of defects, which can make the overall procedure very costly; the numerous variables involved in forming processes, related to the material properties, tooling geometry and process parameters makes it difficult for engineers to accurately predict the occurrence of forming problems, such as springback (Firat, 2007), among others. In this context, ML techniques may aid to solve this issue, since they can be trained with available data for building accurate defects prediction models. The rationale is that the models can generalize in unseen data and successfully identify and check such defect patterns.

Although ideal, the applicability and success of ML models to solve problems are highly dependable of the amount and quality of the available data. This may prevent medium and small companies to fully adopt the concept of smart manufacturing, which could mean a lost in competitiveness in an ever-increasing competitiveness in the automotive industry that has demanded very high quality and robustness requirements, particularly in components that have a direct impact on occupant safety.

One possible solution would be the collaboration among manufacturing companies, where they would share their data to create a

* Corresponding author.
E-mail addresses: mariodib@outlook.com (M.A. da Silveira Dib), prates@ua.pt (P. Prates), bribeiro@dei.uc.pt (B. Ribeiro).

https://doi.org/10.1016/j.eswa.2023.121139
Received 30 March 2023; Received in revised form 2 July 2023; Accepted 6 August 2023
Available online 12 August 2023

Table 1
Experiment result obtained by executing the FL algorithm with corrupted datasets and SecFL activated.

Reference	Sub-section	Outcome
Inamdar et al. (2000)	Machine Learning for Defect Prediction	ANN + bending forming process + real data
Gisario et al. (2011)	Machine Learning for Defect Prediction	ANN + V-shaped bending forming process + experimental data
Liu et al. (2007)	Machine Learning for Defect Prediction	ANN and GA + U-shaped bending forming process + real data
Miranda et al. (2018)	Machine Learning for Defect Prediction	ANN + press break air bending forming process + experimental data
Lei et al. (2021)	Machine Learning for Defect Prediction	SVM + sheet forming process + real Data
Chen et al. (2022)	Machine Learning for Defect Prediction	SVM and KNN and RF + sheet forming process + real Data
Romero et al. (2021)	Machine Learning for Defect Prediction	SVM + Mold forming process + real Data
Dib et al. (2019a)	Machine Learning for Defect Prediction	Single and ensemble classifiers + U-channel forming process + experimental Data
Han et al. (2013)	Machine Learning for Defect Prediction	ANN and PSO algorithm + incremental forming process + experimental data
McMahan et al. (2016)	Federated Learning	Introduction of federated learning algorithm
Shokri and Shmatikov (2015)	Federated Learning	SGD model used for privacy-preserved deep learning algorithm
Dib et al. (2021b)	Federated Learning	Federated learning for springback defect prediction in U-channel forming process
Hao et al. (2020)	Federated Learning	Privacy optimization of federated learning parameters
Phong et al. (2018)	Federated Learning	Privacy optimization of federated learning parameters
Zhang et al. (2017)	Federated Learning	Privacy optimization of federated learning parameters
Bonawitz et al. (2017)	Federated Learning	Privacy optimization of federated learning parameters
Karankar et al. (2014)	Digital Envelopes	Novel for digital envelopes + digital signatures to provide data integrity, confidentiality, and non-repudiation
Guo et al. (2010)	Digital Signature	Scheme of signed digital envelopes
Sun et al. (2020)	Data Poisoning	Efficient method to perform a poison attack on federated learning
Yerlikaya and Bahtiyar (2022)	Data Poisoning	Analysis of the robustness and performances of several machine learning algorithms against data poisoning attacks

common solution to be used by them. However, one major obstacle is the data privacy and data security policies of each company and their desire to not shared their sensitive information with other companies. One method that would help in this scenario is the FL, which decouples the model training from the need for direct access to the raw training data (McMahan et al., 2016). FL allows users to collectively reap the benefits of shared models trained from private data, without the need to centrally store it, where each client train their models with local datasets, that are never sent to the central server, and only the updates, or trained weights, are communicated, empowering models with the amount of data that was not possible to be used before (McMahan et al., 2016).

FL offers a way for companies to collaborate and create solutions while preserving their data privacy. However, despite FL's default privacy protection, there are still concerns regarding data security as the FL approach can be vulnerable to cyber-attacks such as data poisoning (Sun et al., 2020). These attacks can disrupt the training process and even result in the theft of sensitive information.

The main contribution of this paper is establishing a framework, called SecFL, that combines the traditional FL method with DE to provide a private and secure environment for a distributed ML training. The structure of the paper is as follows. First, a description of related works is presented. Second, the used algorithms are described. Third, the proposed approach and experimental settings are presented and explained. Fourth, the proposed work is applied in a case study and the findings are discussed. Finally, the experiences, limitations of the proposed framework and future works are reviewed, concluding this work.

2. Background and related works

The background and related works of this study are presented in the sub-sections below. In order to contextualize the work undertaken in this paper, Table 1 shows an overview of related works comparison addressed in this section.

2.1. Machine learning for defect prediction

ML has been used in manufacturing with different aims such as process optimization and manufacturing defect detection. Table 1 lists the available works that can be seen as using ML for defect prediction with their corresponding objectives. They mainly focus on using ML to improve or optimize the process parameters to springback predict defects in sheet metal forming, which is the focus of this work, considering different setups.

Inamdar et al. (2000) used artificial neural networks (ANN) to the control of the sheet metal bending process in an attempt to restrict springback occurrence and consequently the final angle of bend to within a small tolerance. Gisario et al. (2011) predicts and control the springback in V-shape bending of aluminum alloy sheets by also using ANN, although using experimental data. Liu et al. (2007) applied a generic algorithm (GA) to optimize the topological of an ANN structure to optimize a U-shaped bending forming process, preventing the occurrence of springbacks. Miranda et al. (2018) combined ANN with simulated data that enabled modeling the complex nonlinear behavior of the forming process and springback effect, including the validation of results obtained to predict springback effects in Press brake air bending process.

Lei et al. (2021) proposed a forecast method based on support vector machines (SVM) to accurately predict the maximum thinning ratio of the sheet forming process of zirconium alloys, to avoid the occurrence of cracks. Chen et al. (2022) explored a method based on hyper-parameters search in the field of defect depth classification, using k-nearest neighbors (KNN) algorithm, random forest and SVM to also overcome the lack of crack detection in steel. Romero et al. (2021) also used SVM algorithm, however his goal is to allow the prediction of the geometrical accuracy of molds manufactured via single point incremental forming using aluminized steel sheets.

Dib et al. (2019a) experimented springback prediction in U-channel and square cup forming processes using different ML methods, but the ANN method seems to be preferred and has been proven to be a great

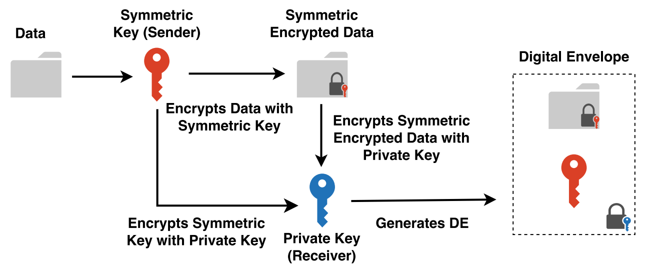

Data
Symmetric Key (Sender)
Symmetric Encrypted Data
Encrypts Data with Symmetric Key
Encrypts Symmetric Encrypted Data with Private Key
Digital Envelope
Encrypts Symmetric Key with Private Key
Private Key (Receiver)
Generates DE

Fig. 1. Standard digital envelope approach.

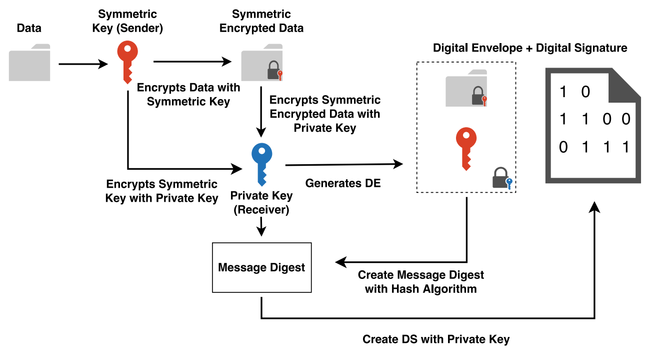

Data
Symmetric Key (Sender)
Symmetric Encrypted Data
Digital Envelope + Digital Signature
Encrypts Data with Symmetric Key
Encrypts Symmetric Encrypted Data with Private Key
1 0
1 1 0 0
0 1 1 1
Encrypts Symmetric Key with Private Key
Private Key (Receiver)
Generates DE
Message Digest
Create Message Digest with Hash Algorithm
Create DS with Private Key

Fig. 2. Digital envelope combined with digital signature.

tool for solving defect prediction in manufacturing because of its ability to approximate non-linear functions in the absence of closed form solutions by providing a way to solve complex, non-linear, polytropic and complex springback problems (Han et al., 2013).

The common approach of the mentioned works is the usage of the centralized learning method, that requires large amount of data for training and validation and where all the data must be aggregated in a single node and then used in the ML training, not preserving the privacy of the data.

2.2. Federated learning

The term federated learning was first introduced by McMahan et al. (2016) and the idea is to provide a central node, or central server, to orchestrate the communication among all the clients' nodes, that would be part in the ML training. The orchestrator creates the first version of the Stochastic Gradient Descent-Based (SGD) ML model and make it available for all clients, that train their own version of the model with their own private data. The choice for the SGD model can be explained by the fact that it can be parallelized and executed asynchronously and updated in the mentioned central server, as described by Shokri and Shmatikov (2015). The FL method also uses the SGD approach, and once the local models are trained inside each client, in parallel, only the models' parameters, or weights, are sent back to the central server, where the weights will be aggregated by the named federated averaging algorithm.

This approach preserves the privacy of the data once it is never exposed to external audience, remaining inside the clients' nodes. The FL proposal is intended to allow the collaboration of different groups with similar problem to solve, regarding data usage, and provide a powerful solution that would be, otherwise, very difficult to achieve. Although the FL method was not initially created focusing on manufacturing, Dib et al. (2021b) adapted it to predict springback defects in U-channel forming process achieving satisfactory results, but the remaining issues of cyber-attacks were not explored. Several privacy optimized FL frameworks were proposed. Hao et al. (2020), Phong et al. (2018), Zhang et al. (2017) and Bonawitz et al. (2017) focused on enhancing privacy by protecting only the models' parameters, where the data was still exposed inside the clients' nodes.

2.3. Digital envelopes

Digital Envelope is a data container used to its content over untrusted network based on asymmetric cryptography and consists in the usage two layers of encryption to secure a message (Karankar et al., 2014). First, a random generated key, called secret key or symmetric, is created and used by the sender to encrypt the message, namely symmetric encryption. Then, the symmetric key is encrypted by the receiver's public key. Finally, the encrypted document and the encrypted symmetric key are packaged together in a called DE and sent to the receiver, and only the owner of the corresponding receiver's private key has access to the content of the DE, represented in Fig. 1.

A caveat regarding DE usage is the fact that while this approach can provide data privacy, it does not assure data integrity, authenticity, nonrepudiation and undeniability (Guo et al., 2010; Karankar et al., 2014).

2.4. Digital signatures

DE can be combined with a mathematical scheme, called Digital Signature (DS), to fix its flaws. The sender makes the message digest from the data using the hash algorithm, which is a one-way algorithm that reads a message of any length and produces a unique output message of a fixed length and generate the DS with its private key. The receiver retrieves the message digest from the DE also using a hash algorithm and uses the sender's public key to verify if both digests are equal, represented in Fig. 2.

The combination of DS and DE, namely signed DE, provides the missing data authenticity, non-repudiation, undeniability and, while it still cannot guarantee data integrity, it is possible to verify if the original data was not modified.

2.5. Data poisoning

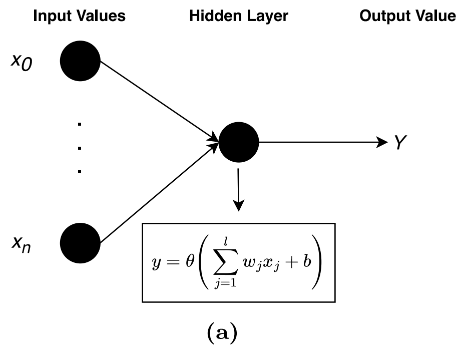
Data poisoning represents external attacks aiming in maliciously manipulate the datasets in order to disturb the ML training and can occur in two ways: causative attacks, where the attacker alters the training process through influence over the training data; and exploratory attacks, where the attacker does not alter the training process, but can use of misclassifications to compromise the model's performance (Sun et al., 2020), such as the change of correctly labeled output with incorrect labels to disrupt performance and preventing the achievement of the desired objective.

Data poisoning attacks has been proved a big threat to ML models and become a topic of interest in the field of adversarial ML (Sun et al., 2020). Yerlikaya and Bahtiyar (2022) analyzed empirically the robustness and performances of several ML algorithms against data poisoning attacks by using four different datasets and three metrics. The final observation is that different types of ML models have different performance results, but all of them sustained decreases for all evaluation metrics.

Sun et al. (2020) provides the most relevant work to our in the context data poisoning and FL, where they effectively launched data poisoning attacks on federated machine learning, by exploiting the existing FL communication protocol.

3. Machine learning algorithms

The main ML algorithms used in this study are presented in the sub-sections below.



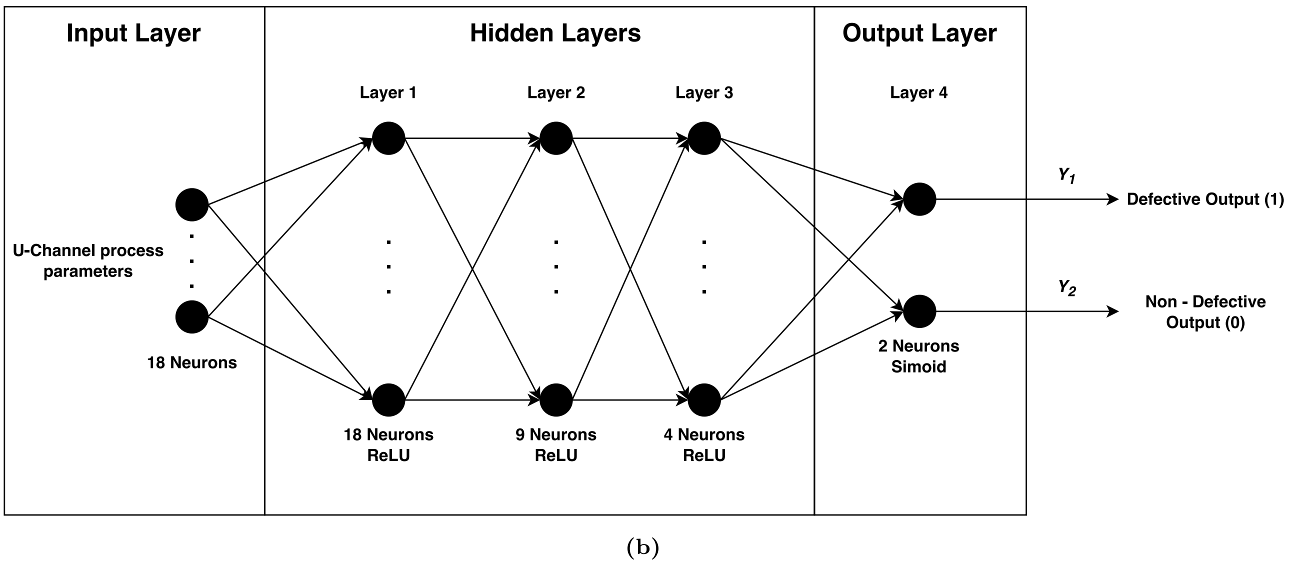


Fig. 3. Scheme of ANNs; Fig. 3(a) base ANN equation; Fig. 3(b) example of a more complex ANN structure that is based on the ANN used in this work.

3.1. Artificial Neural Network

Artificial Neural Network (ANN) is a network containing layers nodes, often called neurons, that connect with the neurons of the next layer from the beginning to end, where the neurons of the same layers cannot be interconnected. The network architecture includes one input layer, an arbitrary number of hidden layers, that are determined by the size and complexity of the relationships being modeled, and one output layer. The main goal of an ANN is to adapt the nodes values to obtain a minimal difference between the network output and the desired output. This is achieved by using three components: the weighted sum of the inputs' values, an activation function θ and a bias b. Eq. (1) shows the base ANN equation, where y is the output of the correspondent node, while the graphic representation of the equation is shown in Fig. 3(a).

$$y = \theta\left(\sum_{j=1}^{l} w_j x_j + b\right) \tag{1}$$

The input values x_j in each node from previous connected nodes are multiplied with a weight w_j, and then summed. The result is then added by a bias b, that is an additional parameter in the neural network that works like the intercept added in a linear equation, which is used to adjust the output along with the weighted sum of the inputs to the neuron. The last step is to pass the final value of a node though an activation function, that has the objective to define the output of the given node based on the type of activation, such as ReLU, sigmoid, softmax or Tanh. In this work we focused on ReLU for the hidden layers and sigmoid for the output layer. As shown in Eq. (2), ReLU returns an output equal to the input for all positive input values, while returning zero for all other input values with the advantage of being faster than other activation functions and has no vanishing gradient problem (Dubey et al., 2021).

$$ReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The choice for sigmoid function in the output layer is due to the classification nature of the defect prediction problem and the expected output exists only between 0 and 1 (Dubey et al., 2021). The sigmoid output produce a vector where each element is a probability. In other words, both probabilities of the input to be defective of non-defective, as shown in Eq. (3).

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Due to the complexity of the problems an ANN is designed to solve, a deeper network architecture is needed to achieve the desired results, which means the more input features and/or output features, the more hidden layers will be necessary, that can be optimized through testing. Fig. 3(b) shows an example of the defined ANN architecture for this work.

3.2. Federated Averaging Algorithm (FAA)

The FL algorithm is based on SGD because it allows an adaptation of the loss function to be more amenable to optimization by simple gradient-based methods. In other words, SGD can be applied naively to the federated optimization problem where a single batch gradient calculation is done per round of communication, meaning one for each client, and it is proved to be computationally efficient (McMahan et al., 2016).

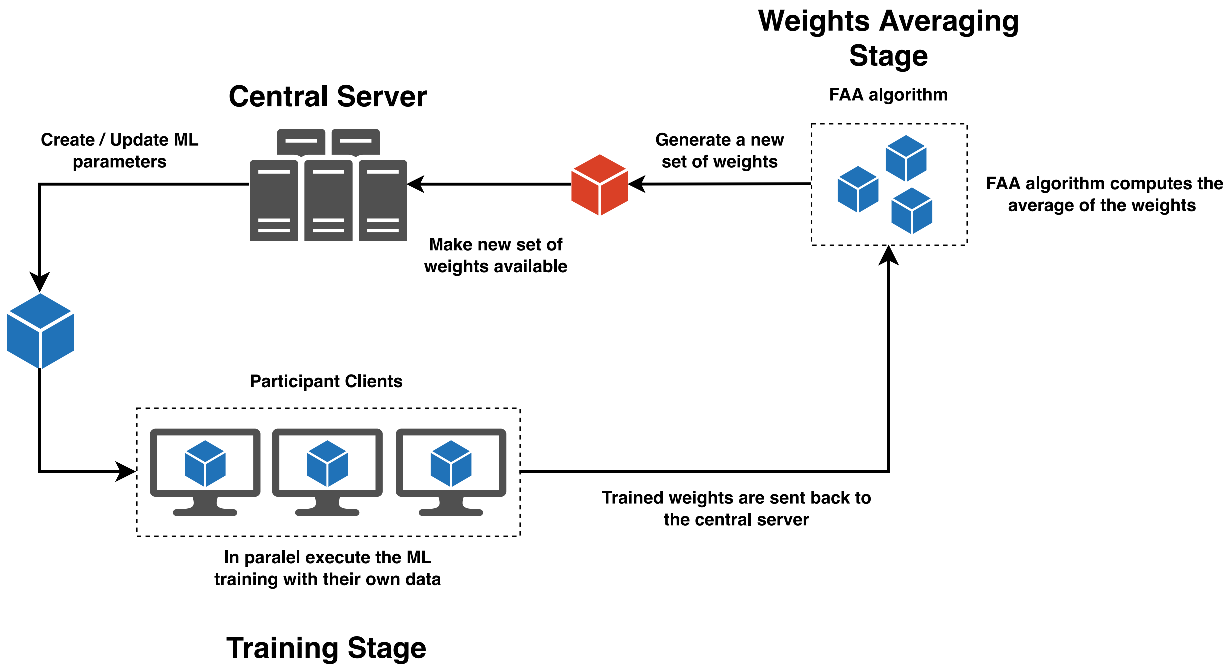

Weights Averaging Stage
FAA algorithm
Central Server
Create / Update ML parameters
Generate a new set of weights
FAA algorithm computes the average of the weights
Make new set of weights available
Participant Clients
Trained weights are sent back to the central server
In paralel execute the ML training with their own data
Training Stage

Fig. 4. Digital envelope combined with digital signature.

The loss function can be understood as the function to validate the best set of weights of a model to solve a given problem, meaning the calculated error of the predictions made using a given set of weights. The SGD based algorithm then changes the weights so that the next evaluation reduces the error, meaning the optimization algorithm is navigating down the gradient of error (Goodfellow et al., 2017). In this work, the expected output of the ML model is binary, for defective and non-defective samples, consequently the loss function used is the log loss (LG) or logarithmic loss, described in Eq. (4) (Ho & Wookey, 2020).

$$LG = -\frac{1}{M}\sum_{m=1}^{M}\Big[Z_m \log(P_{Z_m}) + (1 - Z_m)\log(1 - P_{Z_m})\Big] \tag{4}$$

Where M is the number of training samples, Z_m is the true label for training sample m, P_{Z_m} is the probability of output 1 or defective and $1 - P_{Z_m}$ is the probability of output 0 or non-defective.

To optimize the model's weights, the FL method occurs in two stages, where the decentralized training is first one and happens inside the clients' nodes, and the second step is to perform the Federated Averaging Algorithm (FAA) to consolidate the results into one single and powerful model capable of performing predictions at a high level, as shown in Fig. 4.

During the training stage, initially the central server creates the model structure and the first set of random weights (w_0) and make them available for all participant clients. For each communication round T, a subset (K) of clients (k) belonging to client pool (C) is selected to perform the ML training with their own batched datasets (B) and the defined number of epochs (E). Both B and E can control the amount of computation power required to run the training by regulating their size, where B is the number of training samples utilized in single ML model's training iteration and E is the number of times that the model will work through the entire training dataset. The selected clients then, for each epoch E and each subset of batch B, compute the average gradient on its local data at the current model w_k with a fixed learning rate n, as described in Eq. (5) (McMahan et al., 2016).

$$w_t^k = w_{t-1}^k - n\nabla l\left(w_{t-1}^k; b\right) \tag{5}$$

After each client locally takes one step of gradient descent on the current model using its local data, their weights w_t are transferred back to server, the server then apply the FAA, showed in the Eq. (6) (McMahan et al., 2016), to retrieve average weights of the resulting models.

$$w_t = \sum_{k=1}^{K}\frac{n_k}{n}w_t^k \tag{6}$$

4. Proposed approach

The framework proposed in this work, as shown in Fig. 5, consists in a secure environment capable to provide privacy and protection not only for the data used in distributed ML training, but also to the models' weights. The objective of the mentioned framework is not to prevent cyber-attacks to occur or to identify the type of the cyber-attack, which would be very complex and expensive, but rather to identify when or if an attack happened, the clients and artifact affected and to take the correct actions to halt possible negative effects this activity may cause to the FL process.

To achieve that objective, the combination of the FL method with signed DE that requires one central server and unlimited number of clients. The central server holds its own private key and all the clients' public keys, while the clients hold their own private key, the central server's public key and generate the symmetric key. Each client encrypts its own dataset with the symmetric key, then encrypts the symmetric key with the server's public key, and, combining DE and DS, the clients create the hash of the encrypted document, adding a new layer of protection comparing with the standard DE.

The server first creates the central model with random weights and then sends it to all the participant clients and, before the local training starts, the server verifies for trust issues in the datasets. If negative, the client is excluded from the training, and if positive, the DE is opened, and the standard FL approach continues. Also, to allow the server to have a test dataset available for evaluating the models, a piece of each of the clients' dataset is transferred to the server, as DE, and only aggregated in memory run time, preserving the privacy of its content. The central model is updated with the new weights and the procedure continues until the last defined communication round.

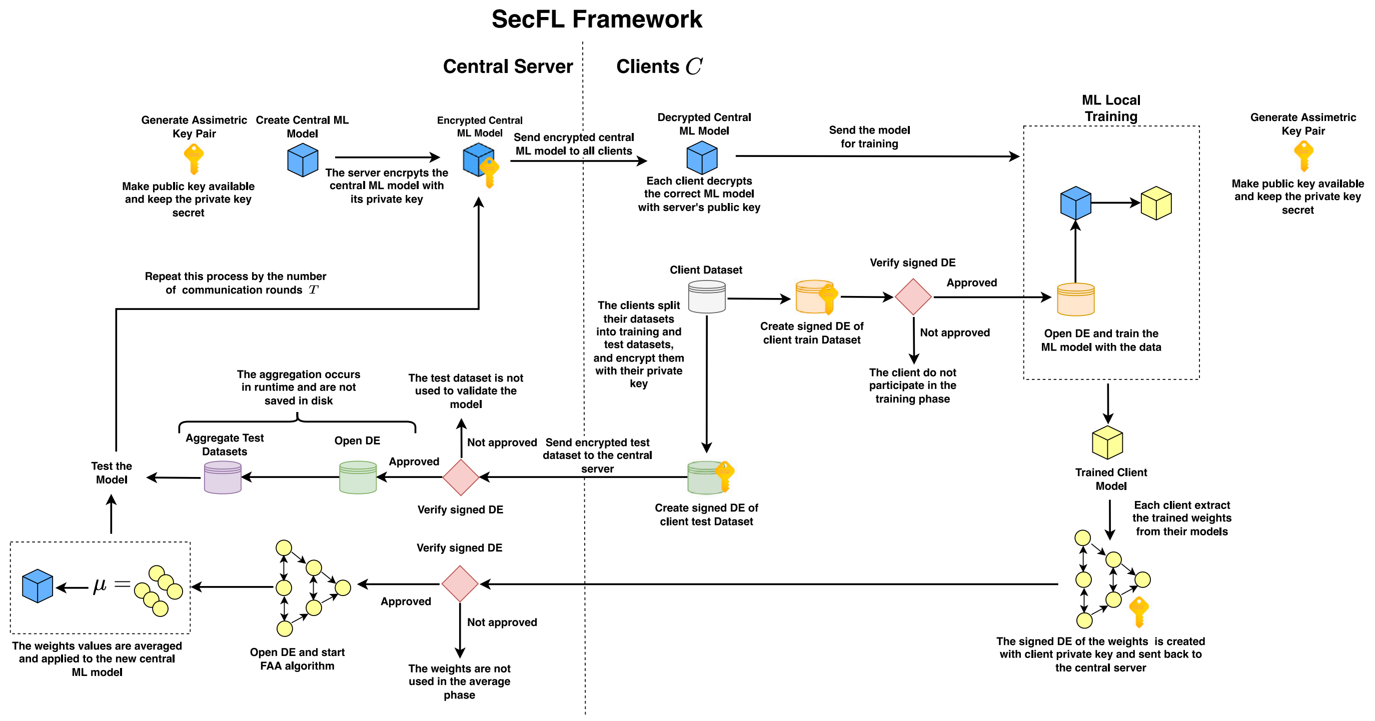
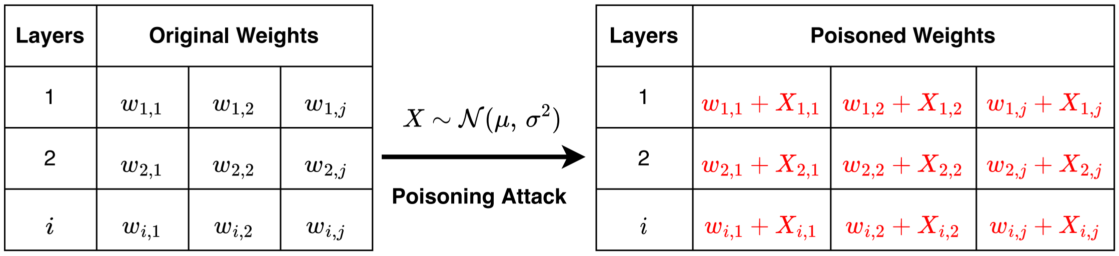

SecFL Framework
Central Server
Clients C
Generate Assimetric Key Pair
Make public key available and keep the private key secret
Create Central ML Model
The server encrpyts the central ML model with its private key
Encrypted Central ML Model
Send encrypted central ML model to all clients
Decrypted Central ML Model
Each client decrypts the correct ML model with server's public key
Send the model for training
ML Local Training
Generate Assimetric Key Pair
Make public key available and keep the private key secret
Repeat this process by the number of communication rounds T
Client Dataset
The clients split their datasets into training and test datasets, and encrypt them with their private key
Create signed DE of client train Dataset
Verify signed DE
Approved
Not approved
The client do not participate in the training phase
Open DE and train the ML model with the data
Trained Client Model
Each client extract the trained weights from their models
The aggregation occurs in runtime and are not saved in disk
The test dataset is not used to validate the model
Aggregate Test Datasets
Open DE
Approved
Not approved
Send encrypted test dataset to the central server
Test the Model
Verify signed DE
Create signed DE of client test Dataset
Verify signed DE
Approved
Not approved
$\mu =$
The weights values are averaged and applied to the new central ML model
Open DE and start FAA algorithm
The weights are not used in the average phase
The signed DE of the weights is created with client private key and sent back to the central server

Fig. 5. Proposed SecFL framework with the combination of the standard FL approach and signed DE.


$X \sim \mathcal{N}(\mu, \sigma^2)$
Poisoning Attack

Layers	Original Weights		
1	$w_{1,1}$	$w_{1,2}$	$w_{1,j}$
2	$w_{2,1}$	$w_{2,2}$	$w_{2,j}$
i	$w_{i,1}$	$w_{i,2}$	$w_{i,j}$

Layers	Poisoned Weights		
1	$w_{1,1} + X_{1,1}$	$w_{1,2} + X_{1,2}$	$w_{1,j} + X_{1,j}$
2	$w_{2,1} + X_{2,1}$	$w_{2,2} + X_{2,2}$	$w_{2,j} + X_{2,j}$
i	$w_{i,1} + X_{i,1}$	$w_{i,2} + X_{i,2}$	$w_{i,j} + X_{i,j}$

Fig. 6. Example of poisoning attack on model's weights.

5. Experimental settings

In this section, the SecFL framework is applied to permit the collaboration in terms of data of multiple parties with similar problem, while keeping the data private and the whole process safe from malicious disruption. This case study exemplifies the utilization of SecFL in a real-world scenario.

5.1. Problem statement

When parties are using FL, connections between them and an external orchestrator must be established to allow the exchange of necessary packages for a proper model training and model updates.

These connections are susceptible to suffer from a wide variety of cyber-attacks from stealing of confidential information to disruption of the ML training process that could prevent the achievement of good results or lead to wrong assumptions. The focus in this study is on the later type of cyber-attack, namely data poisoning.

Two ways of poisoning the data were explored against the standard FL method: dataset poisoning and model's parameters poisoning. Experiments were carried in order to evaluate the negative impact of both approaches to the model's performance and compare with the performance when the SecFL is used to detect secure issues.

5.2. Dataset

The data for this study was generated through numerical simulations using the Finite Element Method (FEM) with the in-house finite element code DD3IMP (Oliveira et al., 2008). This application was specifically developed and optimized for simulating sheet metal forming processes assuming rigid forming tools with surfaces described by Nagata patches (Neto et al., 2017) to solve contact problems between the tools and the deformable sheet metal. The blanks were discretized with 8-node hexahedral solid elements using a selective reduced integration technique.

The sheet metal was modeled as having elastic and plastic properties. For the plastic property, an anisotropic plastic behavior described by the orthotropic Hill'48 yield criterion combined with Swift isotropic hardening law was used. The Hill'48 yield criterion is described in Eq. (7), where the Cauchy stress tensor of a material is defined by its components, namely σ_{xx}, σ_{yy}, σ_{zz}, τ_{xy}, τ_{yz} and τ_{xz} and the anisotropy parameters of the material are represented by F, G, H, L, M and N while Y representing the flow stress.

$$\begin{aligned} Y^2 =& F(\sigma_{yy} - \sigma_{zz})^2 + G(\sigma_{zz} - \sigma_{xx})^2 + H(\sigma_{xx} - \sigma_{yy})^2 \\ &+ 2L\tau_{yz}^2 + 2M\tau_{xz}^2 + 2N\tau_{xy}^2 \end{aligned} \tag{7}$$

The value of Y is determined by the uniaxial tensile stress along the direction of sheet rolling assuming that the summation of G and H is equal to 1, L and M are taken as 1.5 and the anisotropy coefficients

Table 2
Mean value and STD of the features under study. The two levels of blank holder force are 0.28 kN/mm (SD = 0.014 kN/mm) and 1.12 kN/mm (SD = 0.056 kN/mm).

Features		DC06	DP600	HSLA340
C [MPa]	Mean	565.32	1093.0	673.0
	STD	26.85	52.46	32.30
n	Mean	0.259	0.187	0.131
	STD	0.018	0.02	0.011
Y_0 [MPa]	Mean	157.12	330.30	365.30
	STD	7.16	9.646	10.67
E [GPa]	Mean	206	210	210
	STD	3.85	7.35	7.35
v	Mean	0.3	0.3	0.3
	STD	0.015	0.015	0.015
r_0	Mean	1.790	1.010	0.820
	STD	0.051	0.04	0.033
r_{45}	Mean	1.510	0.760	1.070
	STD	0.037	0.03	0.039
r_{90}	Mean	2.270	0.980	1.040
	STD	0.121	0.06	0.061
t_0 [mm]	Mean	0.780	0.780	0.780
	STD	0.013	0.01	0.005
μ	Mean	0.144	0.144	0.144
	STD	0.029	0.029	0.029

are related by $F = \frac{r_0}{r_{90}(r_0+1)}$, $G = \frac{1}{1(r_0+1)}$, $H = \frac{r_0}{(r_0+1)}$ and $N = 0.5 \times \frac{(r_0+r_{90})(2r_{45}+1)}{r_{90}(r_0+1)}$. The Swift hardening law is expressed by Eq. (8), where $\bar{\varepsilon}^p$ is the equivalent plastic strain and C, Y_0 and n are material parameters.

$$Y = C\left[(Y_0/C)^{1/n} + \bar{\varepsilon}^p\right]^n \tag{8}$$

For the elastic property, an isotropic elastic behavior based on the generalized Hooke's law was described, meaning that both properties combined allow the formed metal the possibility to be bent and then return to its original shape to some extent.

The simulation considered three different types of steel sheets, DC06, DP600, and HSLA340 and modeled the U-Channel forming process, which is a 2D bending process commonly used in manufacturing based on the benchmark (Makinouchi et al., 1993). The simulation of a forming process involves three phases. The first phase consists of compressing the sheet against the die using a blank holder. In the second phase, the punch is displaced by 30 mm while the blank holder force (BHF) is kept constant. The final phase involves removing the forming tools, which results in springback.

The simulations were performed using a normal distribution to describe the variability of the parameters C, Y_0 and n (Swift hardening law), Young's modulus E, Poisson coefficient v (Hooke's law) and anisotropy coefficients r_0, r_{45} and r_{90}, with mean and standard deviation values detailed in Table 2. The sheet metal was discretized with one element in the width direction and 150 elements in the length direction, due to the boundary conditions adopted guaranteeing a plane strain state along the width direction. Only half of the U-channel was simulated due to material and geometric symmetries. To complement the dataset, three additional features were considered: initial sheet thickness t_0, friction coefficient μ, and blank holder force BHF, the latter considering two levels that correspond to lower and upper levels of the process window, described in Table 3.

Table 3 exhibits numerical simulation values that were used to determine springback occurrence, when a given output is higher then the presented reference values.

5.3. Data pre-processing and performance metrics

The data is divided into two sets: training (80%) and testing (20%). All datasets undergo data scaling. Each material is subjected to a maximum of 2000 experiments, with 1000 experiments for each BHF level,

Table 3
Springback limit values for the non-defective components.

Material	BHF level	Limit value [mm]
DC06	0.28 kN/mm	5.67
	1.12 kN/mm	2.62
DP600	0.28 kN/mm	11.19
	1.12 kN/mm	8.55
HSLA340	0.28 kN/mm	8.75
	1.12 kN/mm	5.11

resulting in a total of 6000 samples. Before starting each experiment, the data is shuffled and the performance measurements were evaluated using accuracy, precision, recall, and F1 score metrics.

The accuracy metrics evaluates the number of correct predictions made by the model, taking into consideration positives and negatives labels, as shown in Eq. (9).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

Where TP are the true positive instances, TN are the true negative instances, FP are the false positive instances and FN are the false negative instances.

The precision metrics computes the ratio the true positive predictions that are correct positive labels as shown in Eq. (10).

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Where TP are the true positive predictions and FP are the real positive instances.

Recall score represents the model's ability to correctly predict the positives out of actual positives as shown in Eq. (11).

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

Where TP are the true positive instances and FN are the false negative instances.

Lastly, the F1 score can be understood as a function of precision and recall score. F-score is a MLg model performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy, making it an alternative to accuracy metrics. F1 score is computed as shown in Eq. (12).

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

5.4. Weights poisoning

The attack on models' weights is similar compared with dataset poisoning, where original values are changed with fake ones, but the procedure is different. The values on the trained weights' matrix are changed by a random value from a distribution described in Eq. (13).

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{13}$$

Where X is the random number from a distribution $\mathcal{N}$, with $\mu = 0$ and $\sigma = 0.1$. A random number is generated for each weight value of a given ML model, then they are added to the original weight values, as described in Fig. 6, and lastly, they are used during the average phase. The reason for a low and random number to be added to the original weights is to simulate a change pattern that would be difficult to be noted and to be on par with the scale of the original weights.

5.5. Dataset poisoning

To simulate the attack on the datasets, one simple method was used as can be seen in Fig. 7. The outputs of randomly selected datasets were swapped, meaning that the labels which identified the samples as non-defective, in this case the labels 1, are now tagged as 0 and vice versa, and labels which identified the samples as defective are

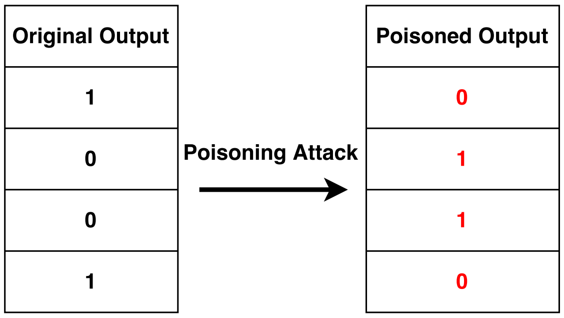


Fig. 7. Example of poisoning attack on data labels.

now labeled as 0. The intent of this is to make the models that train on this data to appear as normal as possible, while providing wrong value during the models' weights aggregation, potentially decreasing the overall performance of the central model.

5.6. Models building approach

The proposed FL approach is based on horizontal FL and is illustrated in Fig. 5. FL offers several advantages over standard ML approaches and one of them is that it enables the distribution of model training among multiple parties, reducing the hardware requirements for this task. It also allows for the analysis of patterns in distributed datasets while ensuring their privacy, which is not possible with standard ML approaches that require centralized training.

Algorithm 1 *KeyPairGeneration*. The central server CV and the K clients generate their public and private key pair. The clients are indexed by k; KL is a defined key length; e is the public exponent; PN is a random prime number from 3 to 36655; p and q are two large random prime numbers where their product, N, cannot exceed KL; d is the private exponent; L is a co-prime number.

```
function GenerateKeyPair()
    $KL \leftarrow 2048$
    $e \leftarrow PN$
    while $q \mod e = 1$ do
        $q \leftarrow KL - \frac{KL}{2}$
    end while
    while $p \mod e = 1$ do
        $p \leftarrow \frac{KL}{2}$
    end while
    $N \leftarrow p \times q$
    $L \leftarrow (p-1) \times (q-1)$
    $d \leftarrow e^{-1} \mod L$
    return $(N, d), (N, e)$
end function

Executed by the server:

$PrivateKey_{CV}, PublicKey_{CV} \leftarrow GenerateKeyPair()$

Executed by the clients:

for each client $k$ in $K$ client's pool do
    $PrivateKey_k, PublicKey_k \leftarrow GenerateKeyPair()$
end for
```

In this study, 9 clients were considered and each one has its own dataset, therefore, 9 datasets, containing different and exclusive samples, were created. As required by a horizontal-based FL, the datasets have the same feature space, which creates a small intersection between them. The central server and all clients generate their pair of public and private keys and distribute them accordingly.

Algorithm 2 *CreateSignedDigitalEnvelopes*. The k clients generate the DEs for local datasets and the trained weights. SK is the generated symmetric key; SED is the symmetric encrypted message, either a dataset of trained weights; DE is the digital envelope

```
function GenerateSymmetricKey()
    $KL \leftarrow 32$ // Bits
    $passWord \leftarrow getRandomBytes(KL)$
    $symmetricKey \leftarrow encodeBase64(passWord)$
    return $symmetricKey$
end function

function SymmetricEncrypt($message_k, SK_k$)
    $m \leftarrow message_k$
    $rounds \leftarrow 14$ // 256 Bits
    // Divide the plaintext into 4x4 matrix
    $matrix \leftarrow divideIntoBMatrix(m)$
    // List with round key from the main symmetric key
    $roundK \leftarrow getRoundKey(SK)$
    for each $r$ in range($rounds$) do
        for each $m$ in $matrix$ do
            if $r = 1$ then
                $attMatrixKey(RoundK_{r-1}, m)$
            else if $r = 1$ then
                $changeBytes(m)$
                $changeRows(m)$
                $attBlockKey(RoundK_{r-1}, m)$
            else
                $changeBytes(m)$
                $changeRows(m)$
                $changeColumns(m)$
                $attBlockKey(RoundK_{r-1}, m)$
            end if
        end for
    end for
    $symmetricEncryptedDataset \leftarrow reasemble(matrix)$
    return $symmetricEncryptedDataset$
end function

function CreateSignedDE($SK_k, SED_k, PublicKey_{cv}$)
    $e, N \leftarrow PublicKey_{cv}$
    $skEncrypted \leftarrow SK_k^e \mod N$
    $sedEncrypted \leftarrow SDE_k^e \mod N$
    $DE \leftarrow pack(skEncrypted, sedEncrypted)$
    // Make the DS available to the Server
    $DS \leftarrow \left(H\left(DE\right)\right)^{d_k}$
    return $DE$
end function

Executed by the clients:

for each $r$ in range($rounds$) do
    $SK_k \leftarrow GenerateSymmetricKey()$
    $SED_k \leftarrow SymmetricEncrypt(message_k, SK_k)$
    $DE_k \leftarrow CreateSignedDE(SK_k, SED_k, PublicKey_{cv},)$
    delete $SK_k$
end for
```

All clients reserve 10% of its own data for test the model in the central server and generate the symmetric key, that will be used to encrypt and decrypt the datasets and trained weights and is included in the DE. The symmetric key is created from a length of 32 bits and then encoded with base 64.

The symmetric encryption, in this work, consists in divide the message in fixed matrices, change the bytes of the message piece of the

Algorithm 3 *ComputeAverageWeights*. Clients are indexed by k; Communication rounds are indexed by t; E is epoch; b is batch size; n is learning rate; T is the total of communication rounds; w is ML weight; m is the ML model; v and v' are the integers computation of the message and the signature respectively.

```
function ClientTraining($m, w, e, b, n$)
    $B \leftarrow (Splitdataset_k intobatchesof sizeb)$
    for each epoch $e$ from 1 to $E$ do
        for each batch $B$ do
            $w \leftarrow w - n\nabla l(w; b)$
        end for
    end for
    $WDE_k \leftarrow CreateDE(SK_k, w, PublicKey_{cv})$
end function

function VerifyDE($DE_k, N_k, e_k$)
    $v \leftarrow S^e \mod N$
    $v' \leftarrow H(DE_k)$
    if $v = v'$ then
        return True
    else
        return False
    end if
end function

Executed by the server:

initialize $m_0$
initialize $w_0$
$E \leftarrow 50$
$b \leftarrow 10$
$n \leftarrow 0.01$
$T \leftarrow 30$
for each round $T$ do
    $K \leftarrow$ (Random set of clients from $C$) // 3 out of 9
    for each client $k$ in $K$ do
        // Run inside each client
        $result \leftarrow verifyDE(DDE_k, N_k, e_k)$
        if $result = True$ then
            $dataset_k \leftarrow openDE(DE_k)$
        else
            Exclude client $k$ from training phase
        end if
        $WDE_k \leftarrow ClientTraining_k(m_{t-1}, w_{t-1}, e, b, n)$
    end for
    for each client $k$ in $K$ do
        // Run inside server
        $result \leftarrow verifyDE(WDE_k, N_k, e_k)$
        if $result = True$ then
            $w_t^k \leftarrow openDE(WDE_k)$
        else
            Exclude weights $k$ from training phase
        end if
    end for
    $w_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_t^k$
end for
```

matrices, shift the position of the bytes and lastly shift the columns of the matrices. This is done though several rounds, in this case 14, and for each round one attributed key, from the client's private key, is generate and used. The symmetric key and the symmetric encrypted message are encrypted again with CV public key and then packed together in the DE. The DE is then digitally signed, to ensure ownership of the message, and mage available for the central server with its DS, as described in more details in algorithms 1 and 2

Algorithm 3 describes training phase procedure. For each communication round (T) from 1 to 30, 3 clients are selected from the total clients (C) to take part in the ML training using their own local datasets. For optimization purposes, the datasets are further divided into batches (B) that are used to update the ML model weights (w) in each of the defined epochs (e). However, before the training occurs, the server first verifies the authenticity of the dataset's DEs. Since the private key and public key are the inverse, it is possible to compute the inverse message digest integer and compare with the original DS. They must match for the DEs to be accepted. If false in any case, the datasets are excluded from the training phase, or the weights are excluded from the averaging phase.

Table 4
Distribution of material samples over the clients.

Clients	Material
1–3	DC06
4–6	DP600
7–9	HSLA340

6. Results and analysis

The experiments were conducted on a Windows station with an Intel(R) Core(TM) i7-10875H CPU 2.30 GHz and 32 GB of RAM. The cryptographic scheme of SecFL utilizes Python library Pycryptodome and the FL is simulated by TensorFlow, using Python as well.

6.1. Framework prediction performance

In this work a feed-forward ANN based model was used to build the SecFL framework and the framework itself was implemented in Python. The trained model was used to predict samples with and without springback and each client had samples from a specific material type, as shown in Table 4.

It was observed in McMahan et al. (2016) work that using 10% of the total number of clients in each communication round would be a good tradeoff between good results, computational power required to execute the ML trainings and model convergence rate. Since in this work there is a small number of clients, the defined number of participants per communications round is 30%. The hyperparameters values were defined based on Dib et al. (2021b) where several hyperparameter were experimented and their performance were measured and compared. The values used in this work are 50 for epochs (E), 10 for batch size (b), 0.01 for learning rate (n) and 30 communication rounds, with a model containing 558 parameters. To measure the experiment performance with different combinations of datasets and weights, in terms of poisoning level, the same experiment were executed 30 times, meaning that for each experiment, a ML model was trained in 30 communication rounds. The performance metrics were averaged, and the standard deviation was calculated.

Table 5 and Fig. 8(a) show that the higher is the number of clients with poisoned datasets, the worse is the model's performance for all the metrics, taking accuracy as an example, the performance dropped from 0.916 to 0.894 and then to 0.777 and finally to 0.704, when one, two and three clients with poisoned datasets participated in the training, respectively. Surprisingly, when only one dataset is poisoned, the standard FL algorithm was able to overcome that, and its performance was very close compared with the baseline value.

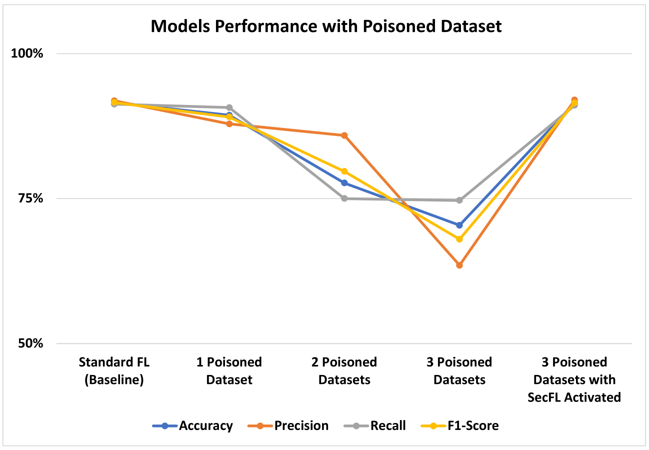
The experiment also shows that when the SecFL is activated, the problematic datasets are identified and correctly removed from the training phase. Even with less data available to be used for the ML

Table 5
Experimental results obtained by training the models with corrupted datasets and SecFL activated.

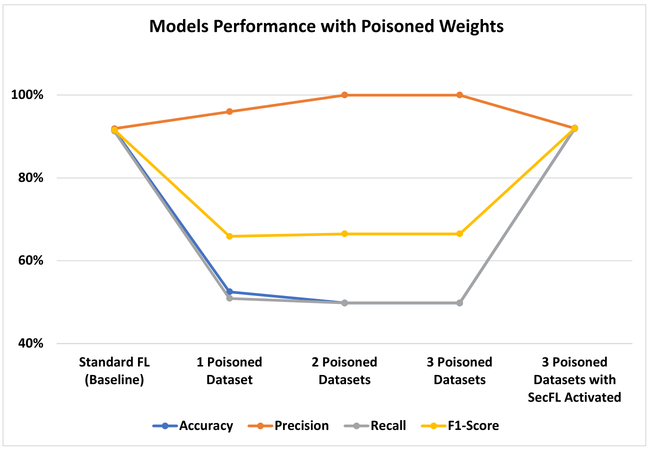
Model	Acc.	Acc. (STD)	Precision	Precision (STD)	Recall	Recall (STD)	F1-Score	F1-Score (STD)
Standard FL Algorithm (Baseline)	**0.916**	0.009	**0.919**	0.011	**0.913**	0.011	**0.916**	0.010
1 Client with Corrupted Dataset	0.894	0.031	**0.879**	0.037	0.907	0.037	0.891	0.0344
2 Clients with Corrupted Dataset	0.777	0.169	**0.859**	0.159	0.750	0.159	0.797	0.149
3 Clients with Corrupted Dataset	0.704	0.282	**0.635**	0.321	0.747	0.321	0.680	0.295
3 Clients with Corrupted Dataset (SecFL Activated)	0.915	0.007	**0.920**	0.011	0.911	0.011	0.915	0.007

Table 6
Experimental results obtained by training the models with corrupted weights and SecFL activated.

Model	Acc.	Acc. (STD)	Precision	Precision (STD)	Recall	Recall (STD)	F1-Score	F1-Score (STD)
Standard FL Algorithm (Baseline)	0.916	0.009	0.919	0.011	0.913	0.011	0.916	0.010
1 Client with Corrupted Weights	0.525	0.101	0.960	0.138	0.509	0.138	0.659	0.136
2 Clients with Corrupted Weights	0.498	0	1	≈ 0	0.498	≈ 0	0.665	≈ 0
3 Clients with Corrupted Weights	0.498	0	1	≈ 0	0.498	≈ 0	0.665	≈ 0
3 Clients with Corrupted Weights (SecFL Activated)	**0.920**	0.008	**0.920**	0.011	**0.919**	0.011	**0.920**	0.009



(a)



(b)

Fig. 8. Performance of ML models on (a) poisoned weights and (b) poisoned weights.

trainings, the performance was virtually the same as the baseline value with 0.915 accuracy.

Differently from the poisoned datasets, it requires only one poisoned set of trained weights to completely disrupt the weight averaging algorithm. Table 6 and Fig. 8(b) show that the models did not converge at all, but one interesting fact is that the precision is very high, varying from 0.960 to a perfect score of 1. This can be explained as the model is assuming all outputs as true or 1 due to the misdirection caused by the poisoned weights. When the SecFL was activated, it also performed achieving 0.920 of accuracy, which is on par with the baseline value.

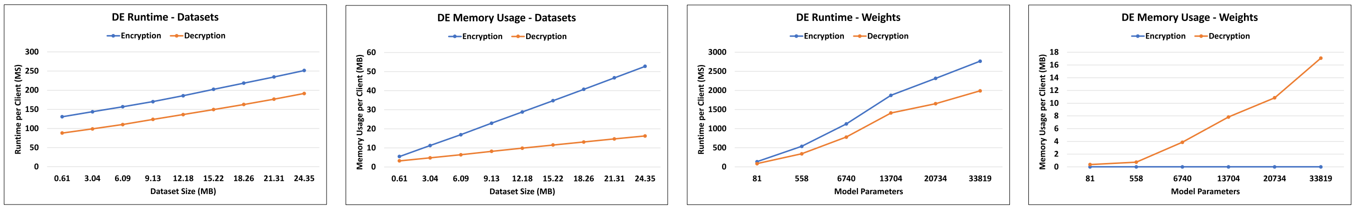
In all scenarios the SecFL was able to provide a secure layer to the standard FL method, which detects when an external attack occurred and prevent the usage of malicious data during the model building.

6.2. Framework functionality

Table 7 presents the functional comparison between SecFl and some state-of-the-art privacy-preserving FL schemes, including PEFL (Hao et al., 2020), PPDL-AHE (Phong et al., 2018), PP-MDL (Zhang et al., 2017) and PSA (Bonawitz et al., 2017). Although all the analyzed privacy-preserving approaches provide a reliable parameter protection, in terms of keep it private, and parameter non-repudiation, which can be understood as the assurance of the integrity and origin of the parameters, the SecFL differs by also provide dataset protection and dataset non-repudiation. This functionality is essential to prevent the ML models from training on corrupt data that would disrupt their performance, considering a realistic threat where the adversary may compromise honest parties and attack their data before the training starts and/or in between the communication rounds. Also, similar to PPDL-AHE and PP-MDL, the performance prediction of SecFL doesn't decrease when compared to the baseline standalone FL prediction. Another main contribution of SecFl is the fact that it provides a general solution to deal with cyber-attacks, providing privacy and protection to both the parameters and datasets against multiple threats, while the others on specific solutions.

Table 7
Functional comparison between SecFL and state-of-the-art approaches.

	SecFL	PEFL	PPDL-AHE	PP-MDL	PSA
Parameter Protection	✔	✔	✔	✔	✔
Dataset Protection	✔	X	X	X	X
Intact Prediction Performance	✔	X	✔	✔	–
Resilient to Attacks	✔	✔	✔	✔	✔
Dataset Non-Repudiation	✔	X	X	X	X
Parameter Non-Repudiation	✔	✔	✔	✔	✔
Multiple Protection	✔	X	X	X	X



(a) (b) (c) (d)

Fig. 9. Computational cost for each client: (a) computational cost for DE creation with different dataset sizes; (b) computational cost for DE extraction with different dataset sizes; (c) computational cost for DE creation with different model's parameter numbers; (d) computational cost for DE extraction with different model's parameter numbers.

Table 8
Framework efficiency results in details for wall-clock runtime and consumed memory, encompassing dataset and weight protection.

Dataset size	Datasets – encryption		Datasets – decryption	
	Runtime (MS)	Memory usage (MB)	Runtime (MS)	Memory usage (MB)
0.61	131	5.48	88	3.19
3.04	144	11.24	99	4.78
6.09	157	16.91	110	6.39
9.13	170	22.94	124	8.19
12.18	186	28.85	136	9.87
15.22	202	34.77	149	11.5
18.26	218	40.7	163	13.11
21.31	235	46.75	177	14.7
24.35	251	52.79	191	16.27
Model params	Weights – encryption		Weights – decryption	
	Runtime (MS)	Memory usage (MB)	Runtime (MS)	Memory usage (MB)
81	137	0.01	87	0.37
558	539	0.01	344	0.75
6740	1125	0.01	780	3.87
13704	1872	0.01	1412	7.84
20734	2317	0.01	1655	10.85
33819	2766	0.01	1990	17.07

6.3. Framework efficiency

To assess the framework efficiency, we focused specifically on the creation and extraction of Data Encryptions (DEs) for both datasets and the protection of models' weights. This aspect represents the main contribution of our work, while the models' training, including the weights average, is not performed on encrypted data. For each selected part, we conducted measurements of the framework efficiency over 30 iterations, ensuring a robust evaluation. The results obtained from these measurements were then averaged to provide a comprehensive assessment of the framework's efficiency.

In the SecFL framework, the creation of Data Encryptions (DEs) follows a general rule that can be divided into four parts for each user. Firstly, there is the creation of the symmetric key. Secondly, the dataset is encrypted using the symmetric key. Thirdly, the symmetric key itself is encrypted. Lastly, a hash signature of the created package is generated. Similarly, during the decryption process, three steps are involved. Firstly, the hash signature is verified. Secondly, the symmetric key is decrypted. Finally, the datasets are decrypted. Both the encryption and decryption operations are sequential and can be impacted by the size of the inputs. Consequently, a linear complexity is expected due to the nature of these sequential operations and their dependence on input size.

Fig. 9(a) shows, indeed, that the wall-clock runtime increases linearly $O(n)$ with the size of the datasets, for either the encryption (creation) and decryption (extraction) of the Digital Envelopes. In terms of memory usage, Fig. 9(b) shows a linear increase, but the impact of dataset size in encryption is much higher than in decryption, where the factors of increase are 6% and 2% respectively.

Figs. 9(c) and 9(d) show the framework efficiency regarding the weights' protection. Differently from the datasets' protection, there is an additional computation because the weights' protection is added to each layer of the model, which has complexity $O(m)$, finalizing the overall computational power for weight protection as $O(n + m)$. The wall-clock runtime is more sensitive to the increase of model's parameters and procedure takes longer time to finish than compared with dataset protection. A similar behavior is observed in memory usage, but the number of parameters did not affect the encryption, where the computational complexity is a constant $O(1)$.

All the results presented in this study were calculated based on the involvement of a single client. It is important to note that the number of clients participating in the communication rounds can have an impact on the runtime of the weights' decryption process. Since the decryption process is performed sequentially by the server, the expected runtime would be multiplied by the number of clients involved.

However, it is worth mentioning that Table 8 provides a more detailed analysis of the results. Despite the potential impact of multiple clients on decryption runtime, SecFL has achieved reasonable results, particularly considering the data size and computational power typically found in small and medium-sized companies. This indicates that SecFL is suitable for scenarios with medium to large-scale operations within these organizational contexts.

7. Conclusions and future works

In this paper a new FL framework is proposed to provide an alternative and secure way to exchange confidential information through the communication protocols making it possible to provide protection and privacy not only to the model's weights, but also to the datasets used during the training.

Based on the experimental results, it was demonstrated that the standard FL method suffered progressively losses in performance the more corrupted data was using and the SecFL was able to overcome such disruption and achieved the desired results, even when a simple mitigation solution was implemented.

The SecFL framework was designed to be computationally efficient, cheap, easy to implement and to be a general protection against the various cyber-attacks, providing mechanisms to identify corrupted data and to take mitigation actions to prevent its negative effects on the ML models by assuring data integrity, confidentiality, non-repudiation, and authenticity of the data, while not preventing an external attack to occur; however, only data poisoning protection was investigated in this study. Thus, more investigation is needed to better understand the impact and effectiveness of the current framework against different threats. Given that the SecFL is designed to be a general solution, a wide range of scenarios can be considered, which may include the consideration of different cybernetic threats usage. Also, a more sophisticated action point to deal with clients with corrupted data and an optimization of the model structure is a challenge that can be addressed.

CRediT authorship contribution statement

Mario Alberto da Silveira Dib: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Pedro Prates:** Conceptualization, Resources, Writing – review & editing, Visualization, Supervision. **Bernardete Ribeiro:** Conceptualization, Resources, Writing – review & editing, Visualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential

Acknowledgments

This research has received sponsorship from FEDER funds through the COMPETE program (Programa Operacional Factores de Competitividade), Portugal and from national funds through FCT (Fundação para a Ciência e a Tecnologia), Portugal under the following projects:

- UIDB/00285/2020
- UIDB/00326/2020
- UIDB/00481/2020
- UIDP/00481/2020
- LA/P/0112/2020

We would like to express our sincere appreciation for the support received from these funding sources.

References

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (pp. 1175–1191). http://dx.doi.org/10.1145/3133956.3133982.

Chen, H., Zhang, Z., Yin, W., Zhao, C., Wang, F., & Li, Y. (2022). A study on depth classification of defects by machine learning based on hyper-parameter search. *Measurement*, *189*, Article 110660. http://dx.doi.org/10.1016/j.measurement.2021.110660.

Dib, M., Oliveira, N. J., Ribeiro, B., & Prates, P. (2019a). Single and ensemble classifiers for defect prediction in sheet metal forming under variability. *Neural Computing & Applications*, *32*, 12335–12349. http://dx.doi.org/10.1007/s00521-019-04651-6.

Dib, M., Ribeiro, B., & Prates, P. (2021b). Federated learning as a privacy-providing machine learning for defect predictions in smart manufacturing. *Smart and Sustainable Manufacturing Systems (ASTM journal)*, *5*(18), http://dx.doi.org/10.1520/SSMS20200029.

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2021). Activation functions in deep learning: A comprehensive survey and benchmark. http://dx.doi.org/10.48550/arxiv.2109.14545.

Firat, M. (2007). U-channel forming analysis with an emphasis on springback deformation. *Materials & Design*, *28*, 147–154. http://dx.doi.org/10.1016/j.matdes.2005.05.008.

Gisario, A., Barletta, M., Conti, C., & Guarino, S. (2011). Springback control in sheet metal bending by laser-assisted bending: Experimental analysis, empirical and neural network modelling. *Optics and Lasers in Engineering*, *49*, 1372–1383. http://dx.doi.org/10.1016/j.optlaseng.2011.07.010.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. MIT Press.

Guo, W., Chen, Y., & Zhao, X. (2010). A study on high-strength communication scheme based on signed digital envelope. In *Proceedings of the second international symposium on networking and network security* (pp. 190–192).

Han, F., Mo, J., Qi, H., Long, R., Cui, X., & Li, Z. (2013). Springback prediction for incremental sheet forming based on FEM-PSONN technology. *Transactions of Nonferrous Metals Society of China*, *4*, 1061–1071. http://dx.doi.org/10.1016/S1003-6326(13)62567-4.

Hao, M., Li, H., Luo, X., Xu, G., Yang, H., & Liu, S. (2020). Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *Journal of Transactions on Industrial Informatics*, *16*, 6532–6542. http://dx.doi.org/10.1109/TII.2019.2945367.

Hattalli, V. L., & Srivatsa, S. R. (2020). Sheet metal forming processes – recent technological advances. *Materials Today: Proceedings*, *5*(3), 2564–2574. http://dx.doi.org/10.1016/j.matpr.2017.11.040.

Ho, Y., & Wookey, S. (2020). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, *8*, 4806–4813. http://dx.doi.org/10.1109/ACCESS.2019.2962617.

Inamdar, M., Date, P., & Desai, U. (2000). Studies on the prediction of springback in air vee bending of metallic sheets using an artificial neural network. *Journal of Materials Processing Technology*, *108*, 45–54. http://dx.doi.org/10.1016/S0924-0136(00)00588-4.

Karankar, N., Kapoor, V., & Patidar, C. P. (2014). An innovative digital envelope slant for an unsecured channel. *International Journal of Scientific Research in Network Security and Communication*, *2.3*, 9–12.

Lei, C., Mao, J., Zhang, X., Wang, L., & Chen, D. (2021). Crack prediction in sheet forming of zirconium alloys used in nuclear fuel assembly by support vector machine method. *Energy Reports*, *7*, 5922–5932. http://dx.doi.org/10.1016/j.egyr.2021.09.013.

Liu, W., Liu, Q., Ruan, F., Liang, Z., & Qiu, H. (2007). Springback prediction for sheet metal forming based on GA-ANN technology. *Journal of Materials Processing Technology*, *187–188*, 227–231. http://dx.doi.org/10.1016/j.jmatprotec.2006.11.087.

Makinouchi, A., Nakamachi, E., Onate, E., & Wagoner, R. H. (1993). Numisheet'93 benchmark problem. In *1993 2nd International conference on numerical simulation of 3D sheet metal forming processes-verification of simulation with experiment*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. (2016). Communication-efficient learning of deep networks from decentralized data. http://dx.doi.org/10.48550/arXiv.1602.05629.

Miranda, S., Barbosa, M., Santos, A., Pacheco, J., & Amaral, R. (2018). Forming and springback prediction in press brake air bending combining finite element analysis and neural networks. *The Journal of Strain Analysis for Engineering Design*, *53*, 584–601. http://dx.doi.org/10.1177/0309324718798222.

Neto, D. M., Oliveira, M. C., & Menezes, L. F. (2017). Surface smoothing procedures in computational contact mechanics. *Archives of Computational Methods in Engineering*, *24*, 37–87. http://dx.doi.org/10.1007/s11831-015-9159-7.

Oliveira, M. C., Alves, J. L., & Menezes, L. F. (2008). Algorithms and strategies for treatment of large deformation frictional contact in the numerical simulation of deep drawing process. *Archives of Computational Methods in Engineering*, *15*, 113–162. http://dx.doi.org/10.1007/s11831-008-9018-x.

Phong, L. T., Aono, Y., Hayashi, T., Wang, L., & Moriai, S. (2018). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, *13*(5), 1333–1345. http://dx.doi.org/10.1109/TIFS.2017.2787987.

Romero, P. E., Rodriguez-Alabanda, O., Molero, E., & Guerrero-Vaca, G. (2021). Use of the support vector machine (SVM) algorithm to predict geometrical accuracy in the manufacture of molds via single point incremental forming (SPIF) using aluminized steel sheets. *Journal of Materials Research and Technology*, *15*, 1562–1571. http://dx.doi.org/10.1016/j.jmrt.2021.08.155.

Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. In *Proceeding of 53rd annual allerton conference on communication, control, and computing* (pp. 909–910). http://dx.doi.org/10.1109/ALLERTON.2015.7447103.

Sun, G., Cong, Y., Dong, J., Wang, Q., & Liu, J. (2020). Data poisoning attacks on federated machine learning. http://dx.doi.org/10.48550/arXiv.2004.10020.

Yerlikaya, F. A., & Bahtiyar, Ş. (2022). Data poisoning attacks against machine learning algorithms. *Expert Systems with Applications*, *208*, http://dx.doi.org/10.1016/j.eswa.2022.118101.

Zhang, X., Ji, S., Wang, H., & Wang, T. (2017). Private, yet practical, multiparty deep learning. In *Proceedings of IEEE 37th international conference on distributed computing systems* (pp. 1442–1452). http://dx.doi.org/10.1109/ICDCS.2017.215.