

The 14th International Conference on Ambient Systems, Networks and Technologies (ANT)  
March 15-17, 2023, Leuven, Belgium

# Secure Multi-Party Computation for Magnetic Resonance Imaging Classification

Oleksandr Lytvyn<sup>a,\*</sup>, Giang Nguyen<sup>a,b</sup>

<sup>a</sup>Faculty of Informatics and Information Technologies, STU in Bratislava, Ilkovičova 2, Bratislava 84216, Slovakia

<sup>b</sup>Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, Bratislava 84507, Slovakia

---

## Abstract

Accessing private data always requires a complex negotiation process. This process becomes even more challenging under privacy regulations such as the General Data Protection Regulation and the California Consumer Privacy Act. However, in most cases, the availability of greater amounts of data leads to significant technological breakthroughs. The perfect example is the ImageNet classification challenge, which leads to significant improvements in the image recognition area. The combination of these concerns raises the question of performing computations on data that cannot be seen, and a whole new research field that consolidates privacy-preserving concepts and modern data mining techniques comes into place. This is also the purpose of the work presented in this paper, the discovery of Secure Multi-Party Computation (SMPC) capabilities on protecting privacy during the machine learning process. The main attention is paid towards the combination of SMPC and image classification approach based on the convolutional neural network, especially the secure inference process on the encrypted magnetic resonance images.

© 2023 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0>)

Peer-review under responsibility of the scientific committee of the Conference Program Chairs

**Keywords:** privacy-preserving; distributed secure computation; homomorphic encryption; deep learning; medical image classification

---

## 1. Introduction

Many modern-world problems, which we want to solve with the help of Artificial Intelligence (AI), require direct access to sensitive information, such as personal, medical, business, or security information. In many cases, access to sensitive data is nearly impossible due to privacy-preserving or business-preserving requirements, which push us to deal with privacy concerns. In recent years, numerous approaches have been proposed to preserve data privacy, from randomization and anonymization [1], through differential privacy [2], to homomorphic encryption [3] and Secure Multi-Party Computation (SMPC) [4]. Data privacy must be preserved in all states: data at rest, data in transit, and data in use. Various protocols are widely used to secure data at rest, such as Advanced Encryption Standard (AES) or Rivest-Shamir-Adleman (RSA), and data in transit, such as Hypertext Transfer Protocol Secure (HTTPS) and Secure

---

\* Corresponding author.

Email address: [oleksandr.lytvyn@stuba.sk](mailto:oleksandr.lytvyn@stuba.sk)

Sockets Layer (SSL). However, the data in the use state are not as well ensured in comparison with the previous two states. Here is a gap in ensuring data in use state to close the data security circle, also in compliance with regulations like the General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA).

On the other hand, machine learning (ML) and deep learning (DL) methods are strongly linked to data. The entire modeling and prediction process uses raw data that adversaries can exploit and expose sensitive data. From this point, some questions are raised, such as how to avoid private information disclosure in the ML process and, moreover, how to make predictions using ML models on data without direct access to them.

In this context, the main objectives of this work are as follows: (1) Investigation of synergy between ML and sensitive data protection tools during the computation process; (2) Discover and evaluate the current capabilities of SMPC in combination with image classification using a convolutional neural network (CNN).

The remainder of the paper is organized as follows. An overview of SMPC, including the requirements for SMPC protocols and different security models, is given in Section 2. Section 3 describes the case study on a secure medical image from different perspectives, starting with the analysis of the sample data and the design scenario, through the various aspects of implementation and evaluation with appropriate metrics, as well as the outcome discussion. Section 4 summarizes the results obtained and possible future research direction.

## 2. Secure Multi-Party Computation

SMPC is a cryptographic computing paradigm that enables distributed computation on a network of multiple parties. Its main point is to maintain the security of the input data during the joint computation while ensuring that the participant can obtain only its own output and nothing else. SMPC functionality can refer to many cryptographic and non-cryptographic tasks like ML, authentication, and zero-knowledge proof, and application-oriented tasks like electronic voting and contract signing. Any task that involves multiple parties can be viewed as an SMPC task.

SMPC must consider the possibility of dishonest behavior of individual participants, as well as the collaboration of corrupted parties [4] with the following requirements to demonstrate the security of SMPC protocols: (1) *Privacy* - none of the participants should be able to access other than its own inputs and outputs; (2) *Correctness* - the protocol should not affect the correctness of output; (3) *Independence of input* - the input of the participants should be independent, despite the existence of corrupted entities; (4) *Guarantee of output* - corrupted entities should not be able to prevent honest participants from receiving the results of the computation; (5) *Fairness* - if all participants receive their outputs, corrupted parties must also receive their outputs.

These concerns define only the essential requirements for the security of the SMPC protocol, which need to be improved for the adversarial behavior defined within the security model. The security model represents the capabilities of corrupted parties that can be handled by the SMPC protocol without affecting the accuracy of the results. SMPC is considered secure and robust within a certain security model if and only if it fulfills security requirements of that model. Currently, the most well-known security models introduced by Zhao et al. in [4] are as follows:

- *Semi-honest Adversary Model* where adversaries execute the protocol correctly, obtain information about corrupted parts of the system, and based on this information, they can attempt to access the private information of other parties.
- *Malicious Adversary Model* where corrupted parties do not follow protocol instructions and can behave unpredictable due to adversary instructions.
- *Covert Adversary Model* where the adversary can attempt to behave maliciously, but has a given probability of being disclosed by honest parties.

The models listed above could be built based on garbled circuits and oblivious transfer concepts in combination with zero-knowledge proof (for Malicious Adversary Model) and cut-and-choose paradigm (for Covert Adversary Model). However, since SMPC is a general concept to secure information during computation time, it could be implemented with more advanced practices and methods. The most promising approaches for the implementation of SMPCs are Fully Homomorphic Encryption (FHE) [5] and Function Secret Sharing (FSS) [6]. Although FHE could handle more complex functions and queries, the computational overhead of current implementations is considerable. FSS has a smaller impact on computation overhead while preserving the capabilities to handle functions of different

complexity. An example is the SyMPC library, which uses one of the implementations of the FSS protocol to perform the computation required during the ML lifecycle, particularly in the inference stage.

**Function Secret Sharing (FSS)** was introduced by Boyle et al. The main idea behind FSS is that functions are converted to secrets, split between multiple parties, and then securely computed without losing the correctness of the results. The authors describe FSS in the following way: FSS  $m$ -party scheme is the way to additively share the function  $f : \{0, 1\}^n \rightarrow \mathbb{G}$ , for some abelian group  $\mathbb{G}$ , into functions  $f_1, \dots, f_m$ , such that  $f(x) = \sum_{i=1}^m f_i(x)$  for every input  $x$ , where  $f_i$  described by the key  $k_i$ , which enables efficient evaluation and every strict subset of keys hides  $f$ . The definition of the FSS for the  $m$ -party computation is presented in [6] as follows:

**Definition 1.** *FSS scheme for  $m$ -parties consists of two algorithms (**Gen**, **Eval**) is defined in the following way:*

- **Gen**( $1^\lambda, \hat{f}$ ) is a probabilistic polynomial-time key generation algorithm, with  $1^\lambda$  as security parameter and  $\hat{f} \in 0, 1^*$  as a description of function  $f$  that produces a  $m$ -tuple of keys  $(k_1, \dots, k_m)$ .
- **Eval**( $i, k_i, x$ ) is a polynomial time evaluation algorithm, with  $i \in [m]$  ( $m$  as party index),  $k_i$  (defines  $f_i : \{0, 1\}^n \rightarrow \mathbb{G}$ ) and  $x \in 0, 1^n$  (input for  $f_i$ ), which outputs  $f_i(x)$  (the  $i$ -th of  $f(x)$ ).

An important part of each encryption scheme is to verify the decrypted ciphertext correctness, which is also applicable for SMPC schemes. The authors provide the definition of correctness depending on the number of parties  $m$  and the security threshold  $t$ . Furthermore, the information leakage parameter *Leak* is introduced to prove that the ideal output with applied *Leak* and the real output are computationally indistinguishable.

In the work [6], the authors present improvements in FSS by simplifying the scheme construction and extending the applicability of FSS to more complex function families. Specifically, the previous notation of a distributed point function was successfully optimized for tensor operations, derived from the construction of the FSS scheme for Decision Trees. This advance opens opportunities to apply FSS to other ML methods. The realization of selected aspects of the SMPC is in an actively evolving stage, in the context of ML and DL. Currently, the three most advanced frameworks are: TensorFlow Encrypted [7], Flower [8] and PySyft [9]. Eventually, the Syft family provided the most advanced implementation of privacy preservation techniques with the PySyft framework for federated learning and differential privacy and SyMPC for SMPC with modern encryption protocols such as Falcon [10], FSS [6] and SPDZ [11].

In recent years, the interest of the SMPC has increased markedly, resulting in various proposals from different perspectives. After narrowing the scope to the combination of SMPC and ML, the number of approaches was significantly reduced. It is worth mentioning that most approaches focus on ML, but not DL. Xiling Li et al. presented a model-agnostic and architecture-agnostic secure feature selection approach based on the Gini index and SMPC [12]. Another approach, presented by A. Resende et al. for the classification of privacy-preserving text in a secure environment based on the Naive Bayes classifier and SMPC [13].

Using deep neural networks, Bittner et al. presented an audio classifier based on deep neural networks with SMPC in the context of semi-honest and malicious adversaries [14]. Haralampieva et al. presented an extensive study of existing frameworks for the combination of SMPC and DL in the work [15]. The authors compare several privacy-preserving frameworks, such as PySyft, CrypTen, TF-Trusted, and HE-Transformer. The experiments are based on the MNIST dataset for the handwritten digit recognition problem. During the experiments, computation time, memory usage, and prediction accuracy are measured within the context of each framework.

In this work, the focus is on the classification of medical images. The proposed solution is based on a more complex DL model that is suitable for images with higher resolution. The complete scenario of secure inference with several parties, including the preparation of data and models, encryption, and decryption of the inference results, is presented as part of applied research focused on the feasibility realization in a secure environment and for certain data types.

### 3. Secured Magnetic Resonance Imaging Classification

Proposed solution contains a case study with the scenario for multi-party encrypted image classification and the corresponding implementation in a simulation environment with further evaluation. Specifically, a convolutional neural network (CNN) is implemented in one training scenario and two inference scenarios called single-party inference and multi-party inference. The experiment was carried out with the publicly available dataset of Magnetic Resonance Imaging (MRI) images (<https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection>) for

brain tumor detection. The dataset consists of 253 images, where the brain tumor is diagnosed in 155 images, and the rest of the images are taken from healthy patients.

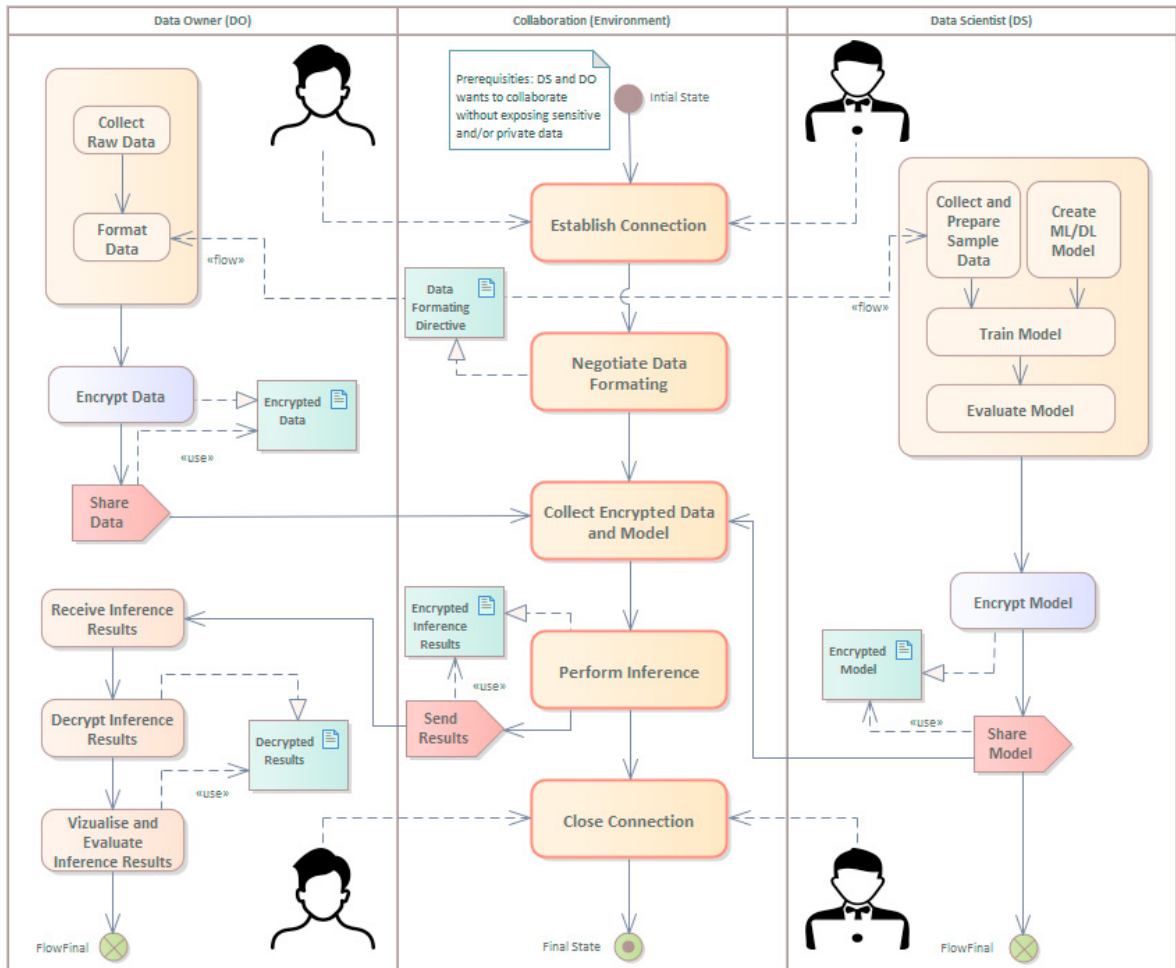


Fig. 1: Secure Inference Scenario Visualization

### 3.1. Design Scenario

The scenario contains two medical institutions (data owners) — *DO1*, *DO2* as participants in the SMPC. Third party acting as the owner of the model (*MO*) or the data scientist (*DS*). Participants (data owners) want to exploit a trained model to help diagnose brain tumors in MRI images. However, the institution's policy prohibits sharing patient personal information with third parties. With the help of SMPC, one can make inferences on the image and preserve the privacy of the entries at the same time.

Figure 1 contains the general scenario for the encrypted inference stage. All participants acting as data owners perform the same actions and consequently, the visualization of the scenario contains only one *DO*. The data and the model must be available within the computation session, making the order of the encryption and sharing steps arbitrary. Unified data formatting is required to overcome the limitations of the SyMPC framework and preserve the accuracy of protocol execution. The scenario can be expanded in different ways to be suitable for other use cases.

The current scenario consists of two main stages: preparation and encrypted inference: (1) The "*Preparation*" stage contains preliminary steps for a successful experiment; (2) The "*Encrypted Inference*" stage consists of individual

steps to perform secured inference using data from multiple parties. Taking into account all previous assumptions, the scenario is proposed as follows.

### (1) Preparation

- (a) Resize, augment, and split the data.
- (b) Implement the CNN model.
- (c) Create a train cycle and a train model.
- (d) Evaluate the accuracy of the model.
- (e) Simulate multiple parties.

### (2) Encrypted Inference

- (a) Establish a session between the parties.
- (b) Negotiate data formatting.
- (c) Encrypt and share the data.
- (d) Encrypt and share the model.
- (e) Perform inference.
- (f) Decrypt and visualize the results.

## 3.2. Implementation

Currently, Syft and SyMPC frameworks are still in the early stages of development. This is the reason why their versions could be incompatible. To preserve the reproducibility of the experiment, it is important to note the exact versions of the software libraries. Concretely: Syft – 0.5.0, SyMPC – 0.5.0rc1, Torch – 1.8.0, Python – 3.9.7. The next important caution is the current limitation of Syft and SyMPC, which allows the computation of two-dimensional convolution only within the FSS security protocol; that is, only two-party computation is supported.

*Preparation.* The first step is to transform the data into the appropriate format. All images are resized to the 64x64 format due to the Virtual Machine's memory limitations with all required frameworks for privacy-preserving, SMPC and DL for proper working. The total number of images was increased with augmentation techniques, such as rotation (by 15,20,25,30 degrees), changing the brightness, adding distortion, and flipping (horizontal and/or vertical). The resulting set contains 1264 images and is divided into training and testing subsets by 80% and 20% respectively.

The second step of the preparation stage is the implementation of the convolutional neural network. The network architecture is shown in Fig. 2. The implementation of the model architecture is not a typical PyTorch model, due to the current limitation of the used versions of Syft and SyMPC supports only MaxPooling2D and ReLu as functions. Therefore, the functions of *torch.nn.functional* must be used. Since SyMPC uses PyTorch's functionality to implement the models, the reference to the *torch* framework must be passed into the model object.

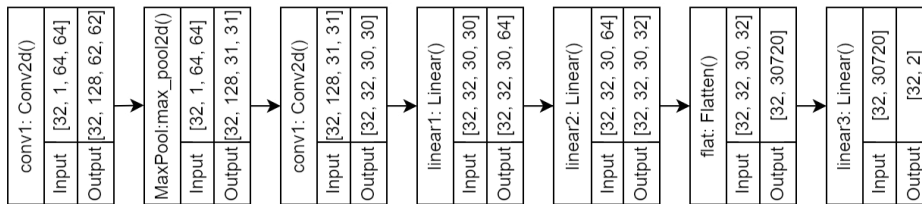


Fig. 2: Architecture of CNN

The next step is the implementation of the train cycle and model training. The train cycle is implemented as regular Torch training cycles with the following hyperparameters: *CrossEntropyLoss* is used as the loss function; *Adam* as an optimizer with a learning rate of 0.0001; and the batch size for training is set to 32. Several model instances were created with the same hyperparameters and trained in different numbers of epochs. Concretely, 10 models were trained from 200 to 2000 epochs, with the 200-epoch step. The best result was achieved by the 9th model, which was trained during the 1800 epoch. The model achieved 0.8893 accuracy and 0.4359 loss in train data.

The last preparation step is to create multiple parties within the simulation environment. SyMPC framework allows one to simulate different numbers of parties to reproduce a wide variety of computation scenarios. At the same time, desired functionality, such as 2D Convolution and MaxPooling, is implemented using the FSS protocol.

*Encrypted Inference.* Firstly, the session needs to be established by specifying the concrete protocol. The security of the session is tuned by setting additional parameters such as the computation ring size, the configuration of the fixed point encoder, and the trusted third party (required by other security protocols).



The next step is to split, encrypt, and share the data. For precise control over the data, each image is shared and encrypted separately. Each tensor must be converted to the `float32` data type to comply with current SyMPC framework implementation. To encrypt the tensors, the `MPCTensor` object must be created whether through the object constructor or the `.share()` function. The output of this step are two arrays – first contain pointers to the encrypted images and second contain labels indicating the tumor presence.

The sharing of the trained model between parties, the inference, and the decryption of the results are illustrated in Algorithm 1. Inference is performed by passing the pointers to encrypted tensors to the model shared within the session. Finally, the encrypted inference is received without disclosing any information about the result of the computation. The decryption of the results is performed by calling the function `.reconstruct()` on the encrypted prediction result pointer. This function can be executed only by the owner of the initial data or by the participant who received the access approved by the data owner (*DO*). The results of the inference (after retrieval) can be visualized as regular images. The results of the SMPC predictions are shown in Figure 3. Above each image are three labels printed: (1) *A* - actual label; (2) *smpc* - prediction of SMPC inference; (3) *raw*: prediction non-encrypted data (without SMPC).

---

**Algorithm 1: Encrypted Inference**


---

```

1 Inference (N, protocol)
   Input   : N: number of workers to participate in inference
             protocol: protocol name for Session establishment
   Output  : predictions: Array of decrypted predictions
   Defaults: SessionManager: class with static methods for Session
             Session: session constructor
             model: pretrained prediction model
             pointers: encrypted images
             data: data for predictions
             torch: instance of Torch framework
2   workers ← GenerateWorkers(N)
3   session ← Session(workers)
4   SessionManager.set_mpc(session)
5   mpc_model ← model.share(session)
6   decrypted_predictions ← Array()
7   foreach pointer ∈ pointers do
8     encrypted_prediction ← mpc_model(pointer)
9     decrypted_prediction ← encrypted_prediction.reconstruct()
10    decrypted_predictions.append(decrypted_prediction)
11  end
12  decrypted_predictions ← torch.cat(decrypted_predictions)
13  decrypted_predictions ← decrypted_predictions.reshape([-1])
14  return decrypted_predictions

```

---

### 3.3. Evaluation

In this experiment, the CNN model is used to perform encrypted image classification based on SMPC approach. At the preparation stage, the dataset is transformed and augmented, the model is implemented and trained, and the individual participants in the computation are created. Furthermore, inferences were made from plain data to compare with the SMPC approach. Prediction comparison was made on the ten randomly chosen images from the test dataset. For proper visualization, the predictions are transformed into Boolean format ("yes" in the case of tumor presence, "no" in the opposite case). Individual images are shown in Figure 3.

Consider Table 1 for a comparison of the inference results. Each cell contains predicted weights for the individual image from raw and SMPC inference. The first weights show tumor absence, and the second is tumor presence. Note that most of the SMPC prediction weights are greater than the raw weights. This is caused by the encryption and decryption operations and does not affect the final classification results.

Consider the prediction for the first image in Table 1: the second values for both tensors are higher than the first, which means that a brain tumor is present in the MRI image and is the correct classification with respect to the visualization of the first image in Figure 3.

Type	1	2	3	4	5
Plain	[-2.5855, 2.5626]	[-3.0288, 3.1859]	[-4.7688, 4.6618]	[ 5.0508, -4.9955]	[-5.1652, 5.1349]
SMPC	[-5.3458, 5.4247]	[-1081.1902, 1103.8638]	[-4.7705, 4.7192]	[ 6.0279, -6.0309]	[-5.3186, 5.2867]
Type	6	7	8	9	10
Plain	[-2.7456, 1.4295]	[-9.0515, 9.0521]	[-8.4364, 9.1571]	[-9.7725, 9.9216]	[ 11.3577, -11.0185]
SMPC	[ 204.0767, -209.7784]	[-10.2065, 10.0856]	[ 1188.5117, -1156.4780]	[-7.6675, 7.7215]	[-209.6427, 213.4298]

Table 1: Comparison of raw and SMPC inference results as tensors

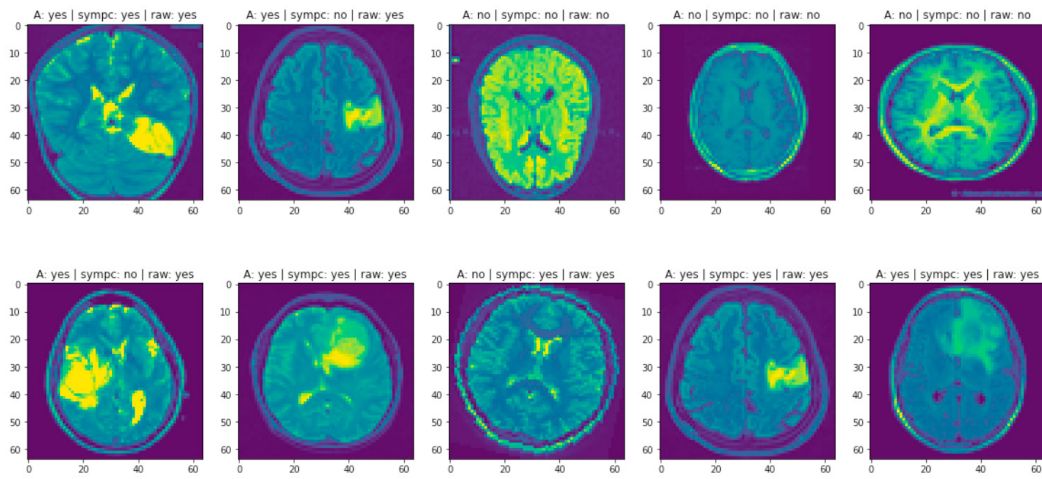


Fig. 3: Visualization results of SMPC inference

	Raw Predictions(RP)	SMPC Predictions(SP)	SP/RP
Accuracy	0.9	0.8	0.9
Raw Number	18	16	16/18

Table 2: Comparison of raw and SMPC accuracies

Table 2 contains the model accuracy comparison for raw data (89%) and encrypted data (79%). The accuracy metric  $ACC = \frac{TP+TN}{TP+TN+FP+FN}$ , where TP and FP are correctly and incorrectly predicted positive values; TN and FN are correctly and incorrectly predicted negative values. In addition to precision, other parameters, such as precision and recall, were measured and used as support metrics. Where the difference between plain and secured metrics is less than 5%, which shows the feasibility of the results obtained. The last column contains the ratio between the raw and SMPC predictions for the loss evaluation. Specifically, 16 correct predictions with SMPC versus 18 correct predictions using the default approach. The loss of accuracy is considered as a trade-off for privacy-preserving computation.

The computation time was also measured within the raw and SMPC inferences, resulting in 2.9917 and 2201.8206 seconds for the classification of 20 images, respectively. The inference time in the entire test set (253 images) takes 28741.7948 seconds. Achieved results show that the computation time depends on the number of operations and the complexity of the CNN model. Another aspect that influences the computation time is the implementation of the FSS protocol. All operations used in the neural network must be modular and follow the additive sharing workflow, which affects the secure inference time.

## 4. Conclusion

The work in this paper presents a privacy-preserving medical imaging classification using a convolutional neural network in combination with SMPC, where the pre-trained model was shared within the session and an inference was made on data from other participants. The study shows that the secure computation between multiple participants became more efficient with recent technological advances within the proposed application scenarios. Although inference with the FSS protocol was successful, several obstacles were identified compared to regular inference, such as an increased inference time (from 3 to 2200 seconds for 20 images) and graceful degradation of prediction accuracy (approximately 10%). This obstacle also shows the high computational requirements of using SMPC during the training stage and encourages further exploration or other solutions. However, SMPC remains a powerful cryptography approach for preserving privacy, including ML and DL, for in-depth study and further investigation efforts.

## Acknowledgements

This work is funded by the European Union through the AI4EOSC project (Horizon Europe) under Grant number 101058593. The publication has also been written thanks to the support of the Operational Programme Integrated Infrastructure for the project: Research in the SANET network and possibilities of its further use and development (ITMS code: 313011W988), co-funded by the European Regional Development Fund (ERDF).

## References

- [1] Ricardo Mendes and João P Vilela. “Privacy-preserving data mining: methods, metrics, and applications”. In: *IEEE Access* 5 (2017), pp. 10562–10582. doi: [10.1109/ACCESS.2017.2706947](https://doi.org/10.1109/ACCESS.2017.2706947).
- [2] Cynthia Dwork, Aaron Roth, et al. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), pp. 211–407. doi: [10.1561/04000000042](https://doi.org/10.1561/04000000042).
- [3] Jung Hee Cheon et al. “Homomorphic encryption for arithmetic of approximate numbers”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437. doi: [10.1007/978-3-319-70694-8\\_15](https://doi.org/10.1007/978-3-319-70694-8_15).
- [4] Chuan Zhao et al. “Secure multi-party computation: theory, practice and applications”. In: *Information Sciences* 476 (2019), pp. 357–372. doi: [10.1016/j.ins.2018.10.024](https://doi.org/10.1016/j.ins.2018.10.024).
- [5] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178. doi: [10.1145/1536414.1536440](https://doi.org/10.1145/1536414.1536440).
- [6] Elette Boyle, Niv Gilboa, and Yuval Ishai. “Function secret sharing: Improvements and extensions”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1292–1303. doi: [10.1145/2976749.2978429](https://doi.org/10.1145/2976749.2978429).
- [7] Morten Dahl et al. “Private machine learning in tensorflow using secure computation”. In: *arXiv preprint arXiv:1810.08130* (2018). doi: [10.48550/arXiv.1810.08130](https://doi.org/10.48550/arXiv.1810.08130).
- [8] Daniel J Beutel et al. “Flower: A friendly federated learning research framework”. In: *arXiv preprint arXiv:2007.14390* (2020). doi: [10.48550/arXiv.2007.14390](https://doi.org/10.48550/arXiv.2007.14390).
- [9] Theo Ryffel et al. “A generic framework for privacy preserving deep learning”. In: *arXiv preprint arXiv:1811.04017* (2018). doi: [10.48550/arXiv.1811.04017](https://doi.org/10.48550/arXiv.1811.04017).
- [10] Sameer Wagh et al. “Falcon: Honest-majority maliciously secure framework for private deep learning”. In: *arXiv preprint arXiv:2004.02229* (2020). doi: [10.48550/arXiv.2004.02229](https://doi.org/10.48550/arXiv.2004.02229).
- [11] Ivan Damgård et al. “Practical covertly secure MPC for dishonest majority—or: breaking the SPDZ limits”. In: *European Symposium on Research in Computer Security*. Springer, 2013, pp. 1–18. doi: [10.1007/978-3-642-40203-6\\_1](https://doi.org/10.1007/978-3-642-40203-6_1).
- [12] Xiling Li, Rafael Dowsley, and Martine De Cock. “Privacy-preserving feature selection with secure multiparty computation”. In: *International Conference on Machine Learning*. PMLR, 2021, pp. 6326–6336.
- [13] Amanda Resende et al. “Fast privacy-preserving text classification based on secure multiparty computation”. In: *IEEE Transactions on Information Forensics and Security* (2022). doi: [10.1109/TIFS.2022.3144007](https://doi.org/10.1109/TIFS.2022.3144007).
- [14] Kyle Bittner, Martine De Cock, and Rafael Dowsley. “Private speech classification with secure multiparty computation”. In: *arXiv preprint arXiv:2007.00253* (2020). doi: [10.48550/arXiv.2007.00253](https://doi.org/10.48550/arXiv.2007.00253).
- [15] Veneta Haralampieva, Daniel Rueckert, and Jonathan Passerat-Palmbach. “A systematic comparison of encrypted machine learning solutions for image classification”. In: *Proceedings of the 2020 workshop on privacy-preserving machine learning in practice*. 2020, pp. 55–59. doi: [10.1145/3411501.3419432](https://doi.org/10.1145/3411501.3419432).