Advancing Security and Efficiency in Federated Learning Service Aggregation for Wireless Networks

Zakaria Abou El Houda[1], Diala Nabousli[1], and Georges Kaddoum[1]

[1]Resilient Machine Learning Institute (ReMI), École de Technologie Supérieure (ÉTS), Montréal, Canada
zakaria.abou-el-houda.1@ens.etsmtl.ca; Diala.Nabousli@etsmtl.ca; georges.kaddoum@etsmtl.ca.

***Abstract*—Federated Learning (FL) is a distributed machine learning technique where multiple devices can collaboratively train a model without sharing their data. As a result, FL ensures distinct privacy benefits compared to centralized training approaches. However, despite its benefits, FL remains susceptible to reverse-engineering attacks that can uncover sensitive information about the training data from the local updates sent by each participant. To address this issue, we propose a framework for securely and efficiently aggregating the results of FL on multiple devices. We compare and evaluate the performance of two techniques, Homomorphic Encryption (HE) and Secure Multiparty Computation (SMPC), to determine the best method that allows devices to share their learning results without revealing their raw local data. Ultimately, we propose using SMPC protocol as the most effective solution to secure FL. Our framework is experimentally evaluated, and its effectiveness in terms of security, efficiency, and accuracy is demonstrated.**

***Index Terms*—Wireless Networks; Federated Learning; HE; SMPC.**

I. Introduction

FEDERATED Learning is a method of training machine learning models on decentralized data, where the data is distributed across multiple devices or clients, and the model is trained locally on each device, with the updates being aggregated centrally to update the global model. This approach allows for improved privacy and data security as the raw data never leaves the device and can also handle large and diverse datasets [1]- [12]. FL provides unique privacy advantages compared to centralized training methods. However, FL remains susceptible to reverse-engineering attacks that can uncover sensitive information about the training data from the local updates sent by each participant. To develop a more secure FL framework, it is important to consider various techniques and methods that can protect against reverse-engineering attacks and prevent the extraction of sensitive information from locally computed updates. One possible approach is to use HE, which allows computations to be performed on ciphertext, keeping the data encrypted at all times. Another technique is SMPC which enables multiple parties to jointly compute a function over their private inputs without revealing them to each other [13].

978-1-6654-6483-3/23/$31.00 ©2023 IEEE

Recently, there has been a growing concern about preserving privacy in FL training. Efforts have been made to address this issue and develop methods for protecting the data privacy used in FL training. In [14], Zheng *et al.* proposed a system design that protects the privacy of individual model updates during the FL process; the proposed system incorporates a lightweight encryption and aggregation, and the ability to handle drop-out clients without affecting their participation in future rounds. In [15], Zhang *et al.* proposed a system design that includes a technique to eliminate the identity of individuals model updates by adding random Gaussian noise during the secure aggregation process. This ensures that the identity of each individual is protected and not revealed during the aggregation of model updates. In [16], Zhang *et al.* proposed a system design for FL that enables the implementation of a verifiable privacy-preserving FL by utilizing a lightweight pseudorandom generator, which allows for a secure and efficient way to protect the privacy of individual model updates during the learning process. Finally, in [17], Wibawa *et al.* proposed a novel method for training a convolutional neural network (CNN) to detect COVID-19 using HE and FL. The authors proposed using FL and HE to train a CNN to detect COVID-19 without exposing sensitive patient data. They evaluated the performance of the proposed method and showed that it is able to achieve comparable accuracy to traditional training methods while preserving privacy. In this paper, we present a comparison between two encryption techniques and highlight the benefits of using SMPC as a lightweight method for preserving privacy in FL. This approach does not rely on complex cryptographic primitives and does not introduce additional noise. The work presented in this paper is distinct from previous works in the field as it specifically focuses on this aspect of comparing two well encryption techniques, $i.e.$, HE and SMPC, in the context of FL. In this paper, we propose a framework for securely and efficiently aggregating the results of FL on multiple devices. We compare and evaluate the performance of two techniques, HE and Secure Multiparty Computation (SMPC), to determine the best method for allowing devices to share their learning results without revealing their raw data. Our framework is experimentally evaluated, and its effectiveness in terms of

security, efficiency, and accuracy is demonstrated.

The structure of this paper is as follows: In Section II, we provide background information. Section III outlines our system model. In Section IV, we detail the implementation and evaluation of our proposed framework. Lastly, Section V concludes the paper.

II. Background

A. Homomorphic encryption

HE is a form of encryption that allows computations to be performed on ciphertext, producing an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on plaintext. This can be useful when sensitive data needs to be processed without being exposed in plaintext form. The mathematical definition of HE can be represented by the following equation:

$$c_1 = E(m_1); c_2 = E(m_2) \tag{1}$$

$$D(c_1 * c_2) = m_1 * m_2 \tag{2}$$

$$D(c_1 + c_2) = m_1 + m_2, \tag{3}$$

where E is the encryption function, D is the decryption function, m_1 and m_2 are plaintext messages.

There are three different types of HE:

1) Paillier encryption scheme: It is based on the mathematical equation $c = g^m * r^n \mod n^2$, where "g" is a generator, "m" is the plaintext, "r" is a random number, "n" is a large prime number, and "c" is the ciphertext. This scheme allows for homomorphic addition of plaintexts, but not multiplication.
2) ElGamal encryption scheme: It is based on the mathematical equation $c = (g^r, m * h^r)$, where "g" is a generator, "r" is a random number, "h" is the public key, "m" is the plaintext, and "c" is the ciphertext. This scheme allows for homomorphic addition and multiplication of plaintexts.
3) Fully Homomorphic Encryption (FHE) scheme: It is a type of homomorphic encryption that allows for any mathematical operation to be performed on the ciphertext.

B. Secure Multi-Party Computation

SMPC allows multiple parties to jointly compute a function over their private inputs, without revealing any additional information about their inputs to the other parties. SMPC protocols are typically built using a combination of cryptographic techniques such as secret sharing, and zero-knowledge proofs. These protocols enable multiple parties to compute a function over their private inputs in such a way that the output is the same as if the function was computed on the plaintext inputs, but without revealing anything about the plaintext inputs.

III. System Model

In this section, we describe the system model. First, we describe HE in the FL context. Then, we describe SMPC in the FL context.

A. HE in FL

HE can be useful in the context of FL, as it allows the participating devices or nodes to perform computations on their encrypted models without revealing their underlying values to the other parties. FL training using HE includes the following steps:

1. Each agent A_i encrypts its model parameters W_i and b_i using HE scheme such as Paillier or ElGamal (see step 13 in Algorithm. 1).

2. The agents send the encrypted model parameters $(E(W_i))$ and $(E(b_i))$ to a server.

3. The server performs the following steps for each round of FL training:

1) The server receives the encrypted model parameters $(E(W_i))$ and $(E(b_i))$ from all agents (see steps 4 to 7 in Algorithm. 1).
2) The server performs the computations on the encrypted model parameters $(E(W_i))$ and $(E(b_i))$, such as gradient updates, using HE operations, such as homomorphic addition, multiplication, and scalar multiplication. Then the server sends the updated model parameters back to the agents to initiate another round of training (see steps 7 to 9 in Algorithm. 1).

4. Each agent A_i receives the updated model parameters and uses them to update its local model W_i and b_i. This is repeated for a maximum number of rounds r_{max}. Algorithm 1 shows the pseudo-algorithm for FL using HE.

B. SMPC in FL

In the context of FL, the goal of SMPC is to enable multiple agents ($e.g.$, devices or nodes in an FL system) to jointly compute a function over their private inputs, without revealing their inputs to the other parties. This can be formalized using the following equation:

$$f(x_1, x_2, \ldots, x_n) = \mathrm{SMPC}(f, x_1, x_2, \ldots, x_n), \tag{4}$$

where f is the function being computed, $x_1, x_2, \ldots, x_n$ are the private inputs of the parties, and SMPC represents the secure multi-party computation protocol that enables the computation of f over the inputs while maintaining their privacy.

FL training using SMPC includes the following steps:

1. Each agent A_i encrypts its model parameters W_i and b_i using a secure multi-party protocol such as Secure Multi-Party Computation of Sums (SMPCS) or Secure Multi-Party Computation of Inner Product (SMPCIP) (see step 16 in Algorithm. 2).

2. The agents send the encrypted model parameters $(E(W_i))$ and $(E(b_i))$ to the secure aggregators (see step 16 in Algorithm. 2).

3. Each secure aggregator, represented by s_i, where $1 \leq i \leq N$, receives one of the shares of the local model updates from each agent. The global model is then updated by aggregating the received encrypted updates, and the updated global model is sent to the server (see steps 4 to 10 in Algorithm. 2).

Algorithm 1: Pseudo-algorithm for Federated Learning using HE

Input: Data and model parameters of each agent
Output: Updated model parameters for each agent
Initial Phase: Generate a public-private key pair (pk, sk) using a secure key generation algorithm and send the public key pk to the agents along with the initial global model parameters W_0 and b_0
for $r \leftarrow$*1 to* r_{max} **do**
  for *each agent* A *in* A_r **do**
    $W^c_{r+1,A} \leftarrow$ AgentUpdate(r,A)[0]
    $b^c_{r+1,A} \leftarrow$ AgentUpdate(r,A)[1]
  end
  $W^c_{r+1} = \frac{1}{A_r} \sum_{A=1}^{A_r} W^c_{r+1,A}$
  $b^c_{r+1} = \frac{1}{A_r} \sum_{A=1}^{A_r} b^c_{r+1,A}$
  $W_{r+1} \leftarrow \mathrm{D}_{sk}(W^c_{r+1})$
  $b_{r+1} \leftarrow \mathrm{D}_{sk}(b^c_{r+1})$
  Send W_{r+1} to each agent
  Send b_{r+1} to each agent
end
AgentUpdate(A,r): // Each agent executes this function
for *each Local epoch* e *in* E **do**
  $W_{r,A} \leftarrow \mathrm{W}_r - \eta \nabla f_A(W_r)$
  $b_{r,A} \leftarrow \mathrm{W}_r - \eta \nabla f_A(b_r)$
end
$W^c_{r,A} \leftarrow \mathrm{E}_{pk}(W_{r,A})$
$b^c_{r,A} \leftarrow \mathrm{E}_{pk}(b_{r,A})$
return $W^c_{r,A}$ and $b^c_{r,A}$

Algorithm 2: Pseudo-algorithm for Federated Learning using SMPC

Input: Data and model parameters of each agent
Output: Updated model parameters for each agent
Initial Phase: Generate a fixed finite field ($i.e.$, a set of integers from 0 to $P-1$ for a prime number P) for all computations to take place along with the initial the global model parameters W_0 and b_0
for $r \leftarrow$*1 to* r_{max} **do**
  for *each Secure Aggregator* i *in* N **do**
    for *each agent* A *in* A_r **do**
      $W^{c,i}_{r+1,A} \leftarrow$ AgentUpdate(r,A)[0]
      $b^{c,i}_{r+1,A} \leftarrow$ AgentUpdate(r,A)[1]
    end
    $W^{c,i}_{r+1} = \frac{1}{A_r} \sum_{A=1}^{A_r} W^{c,i}_{r+1,A}$
    $b^{c,i}_{r+1} = \frac{1}{A_r} \sum_{A=1}^{A_r} b^{c,i}_{r+1,A}$
  end
  $W^c_{r+1} = \sum_{i=1}^{N} W^{c,i}_{r+1}$
  $b^c_{r+1} = \sum_{i=1}^{N} b^{c,i}_{r+1}$
  $W_{r+1} \leftarrow \mathrm{D}(\mathrm{Decode}(\mathrm{W}^c_{r+1}))$
  $b_{r+1} \leftarrow \mathrm{D}(\mathrm{Decode}(\mathrm{b}^c_{r+1}))$
  Send W_{r+1} to each agent
  Send b_{r+1} to each agent
end
AgentUpdate(A,r): // Each agent executes this function
for *each Local epoch* e *in* E **do**
  $W_{r,A} \leftarrow \mathrm{W}_r - \eta \nabla f_A(W_r)$
  $b_{r,A} \leftarrow \mathrm{W}_r - \eta \nabla f_A(b_r)$
end
$W^c_{r,A} \leftarrow \mathrm{E}(\mathrm{Encode}(\mathrm{W}_{r,A}))$
$b^c_{r,A} \leftarrow \mathrm{E}(\mathrm{Encode}((\mathrm{b}_{r,A}))$
Split $W^c_{r,A}$ into N shares
Split $W^c_{r,A}$ into N shares
return $W^{c,i}_{r,A}$ and $b^{c,i}_{r,A}$

4. The server decrypts the model parameters and starts a new round until the desired level of accuracy is achieved or a stopping criterion is met (see step 11 in Algorithm. 2).

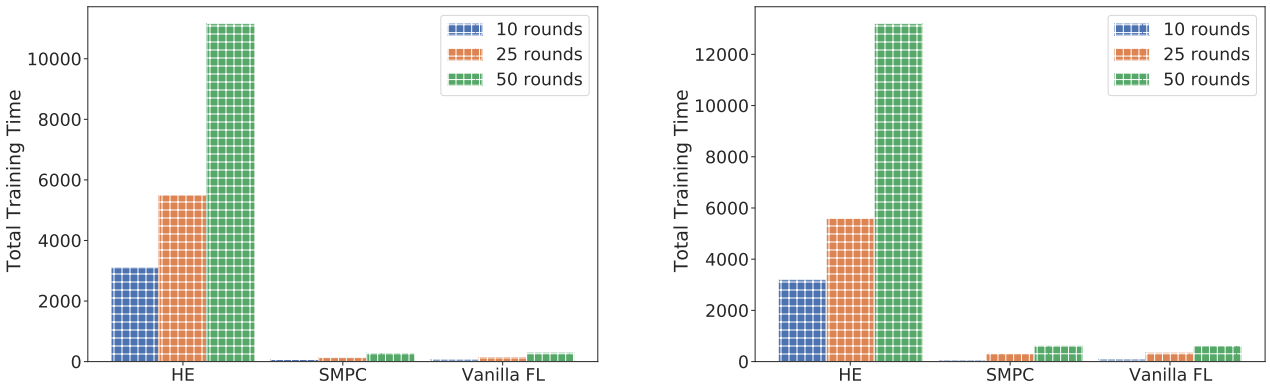
Algorithm 2 shows the pseudo-algorithm for FL training using SMPC.

IV. Implementation

In this section, we first analyze the performance of our proposed framework. Then, we describe our observations on the two techniques.

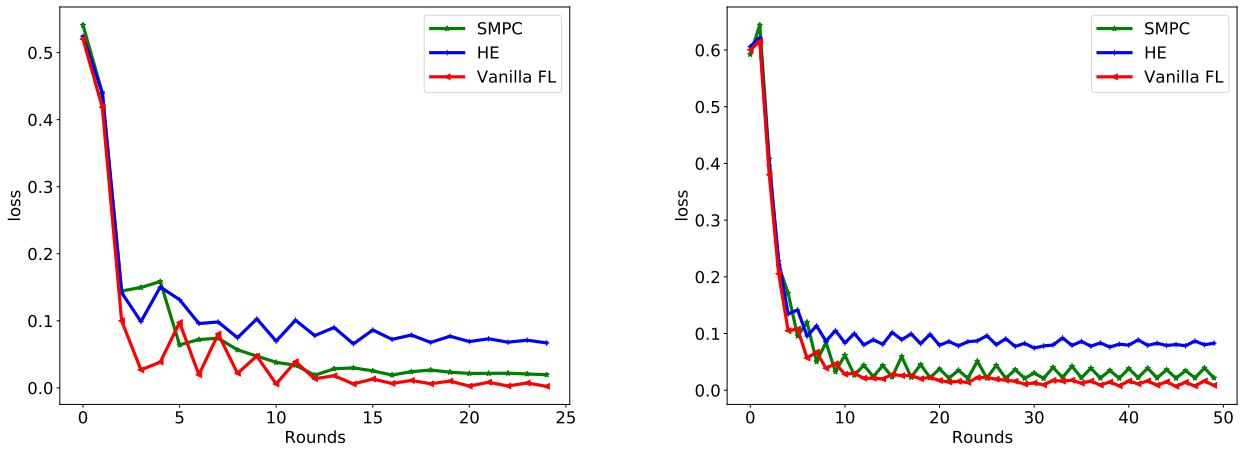
A. Performance evaluation

Our proposed FL framework is implemented using Pysyft [18], is a library for secure and private deep learning and Pytroch [19]. Our proposed FL framework includes a centralized server that distributes partitions of a main dataset to multiple agents/clients. In this study, we use the WSN-DS dataset for intrusion detection in wireless sensor networks, which contains a reasonable number of data samples. The WSN-DS dataset was created using the LEACH protocol, a popular hierarchical routing protocol in WSNs. The dataset was generated by collecting data from Network Simulator 2 (NS-2) and extracting 23 features. Our goal is to create a global model that can accurately classify each sample in the dataset into one of the following five classes: Normal behavior, Constant jamming, Random jamming, Deceptive jamming, or Reactive jamming. We perform experiments involving HE, SMPC, and regular FL (vanilla FL) without encryption. In the first case ($i.e.$, HE), once the local training is done, the model parameters are encrypted using HE (a key size of 64 bits) and then sent to the server. The server uses fed-avg to aggregate the encrypted model parameters. Once done, the model parameters are sent back to each agent, and the training process continues until the desired level of accuracy is achieved or a stopping criterion is met. In the second case ($i.e.$, SMPC), the model parameters were encrypted using a secure multi-party protocol and then sent to secure aggregators; each secure aggregator receives one of the shares of the local model updates from each agent. Next, the global model is updated by combining the encrypted updates received, and then the updated global model is sent back to the server. The third case consists of a



(a) (b)

Fig. 1: Total time of FL process training between HE, SMPC, and vanilla FL for (a) binary classification and (b) Multi-class classification.



(a) (b)

Fig. 2: Training Model losses of HE, SMPC, and vanilla FL on WSN-DS dataset for (a) 25 rounds and (b) 50 rounds.

regular FL (vanilla FL) without the use of encryption.

In this study, we evaluated two distinct scenarios, one for multi-class classification and one for binary classification. We explored various combinations of the number of training rounds (ranging from 10 to 50) and local epochs (varying from 1 to 5). Figs. 1(a) and 1(b) show the total time of the FL process between HE, SMPC, and vanilla FL for binary and multi-class classification, respectively. The total training time includes total tensor encryption and decryption time and aggregation time. In both cases, SMPC results in faster training times when compared to using HE for FL. Furthermore, the time required for SMPC is almost comparable to the time required for traditional FL, indicating that privacy-aware FL can be performed without significant additional computational overhead using SMPC. This makes SMPC a viable option for privacy-preserving FL and is a significant advantage over other methods such as HE, which have been shown to have longer training times. Figs. 2(a) and 2(b) show the training Model losses of HE, SMPC, and vanilla FL on the WSN-DS dataset for 25 rounds and 50 rounds, respectively. The loss decreases and stabilizes for both models and participants, with close-to-zero loss indicating successful learning from each other without data sharing. To demonstrate that using SMPC does not have a negative impact on the performance of the learning model, we conduct experiments to measure the performance metrics of accuracy, precision, recall, F1-score, and training

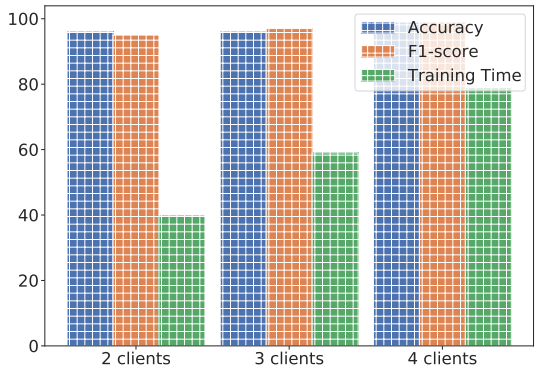


Fig. 3: Comparison of Number of Clients and Computational Costs of SMPC

time (see Table. 1). We varied the number of clients from 2 to 4 clients and measured their performance metrics in Fig. 3. The model achieved a 99% accuracy and F1 score with $69.99s$ training time, which shows that SMPC can be used for privacy-preserving FL without sacrificing the model's performance.

TABLE I: Performance metrics on the WSN-DS dataset

Scenarios	Accuracy	Precision	Recall	F1
SMPC_Binary	**0.99**	**0.99**	**0.99**	**0.99**
FL_Binary	0.99	0.99	0.99	0.99
SMPC_Multi	**0.99**	**0.99**	**0.99**	**0.99**
FL_Multi	0.99	0.99	0.99	0.99

TABLE II: Performance metrics of our FL+SMPC and benchmarks using the WSN-DS dataset

Methods	Accuracy	F1 score	Time (second)
NB	0.72	NA	NA
MLP	0.7	NA	NA
MLP+KSVM	0.94	NA	NA
Adaboost	0.94	0.84	109
Gradient Boost (GB)	0.98	0.97	2880
FL+SMPC	**0.99**	**0.99**	**69.99**

The performance metrics on the WSN-DS dataset are used to evaluate the effectiveness of the learning model. These metrics include accuracy, precision, recall, and F1-score, which are commonly used to evaluate the performance of a machine learning model. These metrics provide information on the ability of the model to correctly classify data points, the proportion of correctly classified positive instances among all positive instances, the proportion of correctly classified positive instances among all instances that were actually positive, and the balance between precision and recall respectively. The results of these metrics on the WSN-DS dataset give an indication of the effectiveness of the learning model. The results show that the use of SMPC does not negatively impact the accuracy of the model when applied to the WSN-DS dataset (see Table. 1). The model achieved a 99% accuracy and F1 score in both scenarios. This shows that SMPC can be used for privacy-preserving FL without sacrificing the performance of the model in terms of accuracy. We have compared the performance of SMPC with some recent AI-based solutions ($i.e.$, Naive Bayes (NB) [20], Multilayer Perceptron (MLP) [21], MLP+Kernelized Support Vector Machine (KSVM) [22], Adaboost [23], and Gradient Boost (GB) [23]) using the same WSN-DS dataset (see Table. 2). Our SMPC method achieved the best results in the shortest training time.

B. Observations

- **Execution Time**: SMPC results in faster training times by far when compared to using HE. Furthermore, the time required for SMPC is almost comparable to the time required for traditional FL, indicating that privacy-aware FL can be performed without significant additional computational overhead using SMPC.
- **Single point of failure**: One of the downsides of using HE is the risk of a single point of failure. This refers to the fact that the security of the entire system relies on encryption and decryption keys, and if these keys are compromised or lost, the entire system is compromised. In contrast, SMPC is a method that allows multiple parties to perform computations on their own encrypted data while keeping the data private. As a result, SMPC does not suffer from the problem of a single point of failure, as the security of the system is distributed among the different parties. This makes SMPC a more robust and resilient option for privacy-preserving FL than HE.
- **FL model performance**: It is crucial to note that using Secure SMPC for FL should not significantly impact the performance of the learning model. While there may be some slight degradation in performance due to the added complexity of the SMPC protocol, the goal of SMPC is to minimize this degradation as much as possible while still providing privacy-preserving FL. Through our studies, we have shown that SMPC can maintain similar performance compared to traditional FL while providing better privacy guarantees.

V. Conclusion

In this paper, we presented a comparison between two encryption techniques and emphasized the importance of using SMPC as a lightweight method for preserving privacy in FL processes. The proposed SMPC protocol was presented as the most effective solution. The effectiveness of the proposed framework was experimentally evaluated in terms of security, efficiency, and accuracy, and the results demonstrated its effectiveness.

Acknowledgement

The authors would like to thank Mitacs/Ultra Intelligence Communications through project IT25839 and the National Natural Sciences and Engineering Research Council of Canada (NSERC) through research grant RGPIN-2020-06050, for the support of this research.

REFERENCES

[1] Z. A. El Houda, B. Brik, A. Ksentini and L. Khoukhi, "A MEC-Based Architecture to Secure IoT Applications using Federated Deep Learning," in IEEE Internet of Things Magazine, vol. 6, no. 1, pp. 60-63, March 2023, doi: 10.1109/IOTM.001.2100238.

[2] H. P. Phyu, R. Stanica and D. Naboulsi, "Multi-Slice Privacy-Aware Traffic Forecasting at RAN Level: A Scalable Federated-Learning Approach," in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2023.3267725.

[3] A. Meftah, T. N. Do, G. Kaddoum, C. Talhi and S. Singh, "Federated Learning-Enabled Jamming Detection and Waveform Classification for Distributed Tactical Wireless Networks," in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2023.3271578.

[4] Z. A. E. Houda, B. Brik, A. Ksentini, L. Khoukhi and M. Guizani, "When Federated Learning Meets Game Theory: A Cooperative Framework to Secure IIoT Applications on Edge Computing," in IEEE Transactions on Industrial Informatics, vol. 18, no. 11, pp. 7988-7997, Nov. 2022, doi: 10.1109/TII.2022.3170347.

[5] F. Yu et al., "Communication-Efficient Personalized Federated Meta-Learning in Edge Networks," in IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 1558-1571, June 2023, doi: 10.1109/TNSM.2023.3263831.

[6] Z. A. E. Houda, A. S. Hafid and L. Khoukhi, "MiTFed: A Privacy Preserving Collaborative Network Attack Mitigation Framework Based on Federated Learning Using SDN and Blockchain," in IEEE Transactions on Network Science and Engineering, vol. 10, no. 4, pp. 1985-2001, 1 July-Aug. 2023, doi: 10.1109/TNSE.2023.3237367.

[7] A. Meftah, G. Kaddoum, T. N. Do and C. Talhi, "Federated Learning-Based Jamming Detection for Distributed Tactical Wireless Networks," MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM), Rockville, MD, USA, 2022, pp. 629-634, doi: 10.1109/MILCOM55135.2022.10017755.

[8] Z. A. El Houda, L. Khoukhi and B. Brik, "A Low-Latency Fog-based Framework to secure IoT Applications using Collaborative Federated Learning," 2022 IEEE 47th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 2022, pp. 343-346, doi: 10.1109/LCN53696.2022.9843315.

[9] F. Naeem, M. Ali and G. Kaddoum, "Federated-Learning-Empowered Semi-Supervised Active Learning Framework for Intrusion Detection in ZSM," in IEEE Communications Magazine, vol. 61, no. 2, pp. 88-94, February 2023, doi: 10.1109/MCOM.001.2200533.

[10] Z. Abou El Houda, A. S. Hafid, L. Khoukhi and B. Brik, "When Collaborative Federated Learning Meets Blockchain to Preserve Privacy in Healthcare," in IEEE Transactions on Network Science and Engineering, 2022, doi: 10.1109/TNSE.2022.3211192.

[11] H. P. Phyu, D. Naboulsi and R. Stanica, "Mobile Traffic Forecasting for Network Slices: A Federated-Learning Approach," 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Kyoto, Japan, 2022, pp. 745-751, doi: 10.1109/PIMRC54779.2022.9977882.

[12] Z. A. El Houda, D. Nabousli and G. Kaddoum, "Cost-efficient Federated Reinforcement Learning- Based Network Routing for Wireless Networks," 2022 IEEE Future Networks World Forum (FNWF), Montreal, QC, Canada, 2022, pp. 243-248, doi: 10.1109/FNWF55208.2022.00050.

[13] T. Stacey, B. Nathalie, A. Ali, S. Thomas, L. Heiko, Z. Rui, Z. Yi, "A Hybrid Approach to Privacy-Preserving Federated Learning," in Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, doi: 10.1145/3338501.3357370.

[14] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi and C. Wang, "Aggregation Service for Federated Learning: An Efficient, Secure, and More Resilient Realization," in IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2022.3146448.

[15] X. -Y. Zhang, J. -R. Córdoba-Pachón, P. Guo, C. Watkins and S. Kuenzel, "Privacy-Preserving Federated Learning for Value-Added Service Model in Advanced Metering Infrastructure," in IEEE Transactions on Computational Social Systems, 2022, doi: 10.1109/TCSS.2022.3204361.

[16] Z. Zhang et al., "G-VCFL: Grouped Verifiable Chained Privacy-Preserving Federated Learning," in IEEE Transactions on Network and Service Management, 2022, doi: 10.1109/TNSM.2022.3196404.

[17] Ebrianti Wibawa, Ferhat Ozgur Catak, Murat Kuzlu, Salih Sarp, and Umit Cali. 2022. Homomorphic Encryption and Federated Learning based Privacy-Preserving CNN Training: COVID-19 Detection Use-Case. In Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference (EICC '22). Association for Computing Machinery, New York, NY, USA, 85–90.

[18] PySyft, https://github.com/OpenMined/PySyft

[19] Pytorch Framework, https://colab.research.google.com/

[20] S. Ismail and H. Reza, "Evaluation of Naïve Bayesian Algorithms for Cyber-Attacks Detection in Wireless Sensor Networks," 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2022, pp. 283-289, doi: 10.1109/AIIoT54504.2022.9817298.

[21] A. Iman, B. Al-Kasasbeh, and M. AL-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," 2016 journal of sensors, Hindawi, 2016.

[22] M. Hachimi, G. Kaddoum, G. Gagnon and P. Illy, "Multi-stage Jamming Attacks Detection using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks," 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 2020, pp. 1-5, doi: 10.1109/ISNCC49221.2020.9297290.

[23] M. Nouman, U. Qasim, H. Nasir, A. Almasoud, M. Imran and N. Javaid, "Malicious Node Detection Using Machine Learning and Distributed Data Storage Using Blockchain in WSNs," in IEEE Access, vol. 11, pp. 6106-6121, 2023, doi: 10.1109/ACCESS.2023.3236983.