Federated Learning for Privacy Preservation of Healthcare Data From Smartphone-Based Side-Channel Attacks

Abdul Rehman, Imran Razzak, and Guandong Xu

***Abstract*—Federated learning (FL) has recently emerged as a striking framework for allowing machine and deep learning models with thousands of participants to have distributed training to preserve the privacy of users' data. Federated learning comes with the pros of allowing all participants the possibility of creating robust models even in the absence of sufficient training data. Recently, smartphone usage has increased significantly due to its portability and ability to perform many daily life tasks. Typing on a smartphone's soft keyboard generates vibrations that could be abused to detect the typed keys, aiding side-channel attacks. Such data can be collected using smartphone hardware sensors during the entry of sensitive information such as clinical notes, personal medical information, username, and passwords. This study proposes a novel framework based on federated learning for side-channel attack detection to secure this information. We collected a dataset from 10 Android smartphone users who were asked to type on the smartphone soft keyboard. We convert this dataset into two windows of five users to make two clients training local models. The federated learning-based framework aggregates model updates contributed by two clients and trained the Deep Neural Network (DNN) model individually on the dataset. To reduce the over-fitting factor, each client examines the findings three times. Experiments reveal that the DNN model achieves an accuracy of 80.09%, showing that the proposed framework has the potential to detect side-channel attacks.**

***Index Terms*—Federated learning, healthcare, keystroke inference, machine learning, motion sensor, privacy preservation, side chanel attacks, smartphone security.**

I. Introduction

THE smartphone contains Personal Health Records (PHR) comprising data (i.e., family medical histories, past medical and surgical interventions, mental health data, physical activity data, heart rate data, and mood prediction) [1]–[3]. Studies [4]–[6] have shown that PHR data can be stolen using a smartphone's hardware sensor. Regulatory requirements (i.e., General Data Protection Regulation (GDPR) [7], HIPAA [8]) can be met with the help of a newly emerging paradigm, Federated learning (FL), in the field of machine learning. While making use of benefits associated with massively distributed data, FL can mitigate privacy concerns [9]–[12]. FL helps the participants in collaborative training of a global model without sharing their local training data [12]. During each round of communication, all participants train local models based on their training data, and the model is then submitted to the server with updates. A global model is built by the server while employing a secure aggregation using the average of weights associated with local models [13], [14].

FL finds inspiring applications in self-driving cars' image classification, recommendation of services for personalized products, and following word predictors for keyboards [15]. As FL preserves participants' anonymity and the confidentiality of training, adversaries may find attraction in this setting. Moreover, there is a possibility of creating a robust deep-learning model by adversaries needing sufficient training data. They can make such models as they train in an FL framework. In the context of Deep-Learning Side-Channel Attacks (DLSCAs), [16] investigated a new attack vector. Currently, DLSCA is regarded as one of the most effective attacks against cryptographic algorithms' implementation [16].

Although there exist several studies focusing on smartphone-based side Chanel attacks [4]–[6], however, they did not focus on privacy preservation of data and lack providing promising results in regards to achieving higher accuracy when classifying keystrokes. This study addresses the side-channel information associated with smartphone soft keys and vulnerability to leaking by physical implementations. Multi-source training, commonly known for generalization, [17], [18], involves using more than a single profiling device, can reduce the negative impact caused by hardware specifications. However, it results in communication overhead due to the distribution of model training. Besides, the model is expensive to train due to complex data models. Bagging [19], also known as bootstrap aggregation, is another available solution for reducing generalization errors in machine learning. Recently, a diverse set of distinctly trained models have been utilized collectively for voting on the output result [20] and showed superior robustness and predictive performance; however, it reduced the variance

Abdul Rehman and Guandong Xu are with the School of Computer Science & Advanced Analytics Institute Faculty of Engineering & IT, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: abdulrehmanjaved@ieee.org; guandong.xu@uts.edu.au).

Imran Razzak is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: imran.razzak@unsw.edu.au).

compared to stand-alone learning models. Keeping insight into these statements, this paper aims to fill this gap by presenting one such evaluation. By implementing FL, multi-source training (data-level aggregation) and FL (model-level aggregation) are applied to detect side-channel attacks. The following are the key contributions and aspects of this work:

- Develop a novel FL-based framework that exposes the issue of smartphone hardware sensors revealing smartphone users' privacy and preserving the formal privacy while also detecting side-channel attacks.
- Proposed framework aggregates model updates provided by 10 participants. The Deep Neural Network (DNN) is utilized for training on combined side-channel data from 10 decentralized edge devices at the client end, after which the model outputs from 2 clients are aggregated at the server end.
- Experiments show that the FL-based DNN model for side-channel attack detection achieves an accuracy of 80.09%, indicating that the suggested framework can identify side-channel attacks efficiently.

This research is organized into several sections. Section II provides a quick overview of the most recent relevant work. Section III provides the network model, dataset, and preliminaries. The proposed framework is presented in Section V. Section VI presents the assessment criteria and outcomes of the recommended approaches. Finally, Section VII concludes the study and concludes with recommendations for further work.

II. Related Work

This section presents the background and existing work side-channel attacks using machine learning and deep learning.

A. Machine Learning Based Side-Channel Attacks Detection

Javed *et al.* [4] explored hardware sensors (like the gyroscope, accelerometer, and magnetometer) to detect typed characters on a smartphone soft keyboard. The authors focused on inferring cross-application keystrokes and developed an Android-based app named *AlphaLogger*. The *AlphaLogger* shows that the smartphone sensor data can be used to predict soft keyboard inputs. They created their dataset with ten individuals using various Android smartphones. The experiments showed that when sensors are used in conjunction with the magnetometer sensor, the *AlphaLogger* performs better, resulting in a 90.2% accuracy. In another work, Cai *et al.* [21] provided three kinds of research in succession to investigate smartphone-based side-channel attacks and investigate the security implications of implicit sensors in smartphone devices. They discuss a broad framework of defense against sensor-sniffing attacks. The work showcases increasingly ubiquitous sensors like (GPS and mouthpieces). The very same researchers explain smartphone-based side-channel attacks in [6]. The authors demonstrate the weakness of a side-channel attack using an Android application named TouchLogger. TouchLogger used ML methods to estimate keystrokes by using the gyroscope sensor. This study was evaluated using a numeric keypad on an HTC Evo 4G smartphone in landscape mode. TouchLogger correctly predicted more than 70% of keystrokes.

Chiappetta *et al.* developed a machine learning framework to discover cache-based side channels [22]. In this detection approach, neural networks have been employed to build models based on the values of Hardware Performance Counters (HPCs) that meet benign and spying processes during the detection of stealth Flush+Reload attacks. Cai *et al.* [23] used machine learning methods for intelligent IoT applications to assess smartphone vulnerability based on the Android OS. They performed an experience study on over 1,406 Android applications to determine the amount of security risk. They used six Machine Learning approaches, and the Random Forest classification algorithm outperformed all others.

B. Deep Learning Side-Channel Attacks Detection

Javeed *et al.* [5] focused on the side-channel cyber-attacks using which hackers can monitor an individual's essential data through the smartphone screen's keystrokes. They proposed *Betalogger* that makes use of a Dense Multi-layer Neural Network (DMNN)., which is built on the sequence to sequence labeling (S2SL) architecture. The *Betalogger* technique employs dense Language Modeling (LM) and DMNN to predict and create lengthy or short phrases written on a smartphone keypad. They improved the dataset used in their previous research [4]. The authors presented a comparative analysis of the proposed DMNN technique with several machine learning approaches, and DMNN outperformed these algorithms with an accuracy of 91.14%. A study focusing on timing analysis [24] established the side-channel attacks showing essential information can be compromised by the cipher's non-constant running time. Power analysis [25] was also introduced, where the input data-driven typical consumption of varying amounts of power is exploited. Presently, power consumption remains a leading one among the most easily comprised side channels. This paper focuses on power analysis. An n-bit key k $\mathcal{E}$ K is intended to be recovered as a result of the side-channel attack. Here, the K denotes the set of all possible keys. For recovering, the attacker uses the physical measurements (e.g., power consumption) and unknown data input (i.e., plaintext). The strategy of divide-and-conquer is usually used where m-bit parts k_i (subkeys) are generated from the division of the key k, which is followed by independent recovery of subkeys, for i $\mathcal{E}$ {1, 2;....n/m}. Typically, m = 8. The two settings for side-channel analysis in deep learning are profiling and non-profiling. The targeted cryptographic algorithm's leakage profile is learned before the actual attack in profiling attacks.

In summary, side-channel attacks have been studied, ranging from traditional desktops to smartphones. The above-discussed studies focused on smartphone-based side-channel attacks [4]–[6] helps to detect side-channel attacks. However, they did not focus on privacy preservation of data and lacked in providing promising results regarding achieving higher accuracy when classifying keystrokes. This paper aims to fill this gap by presenting one such evaluation using FL based DNN model for side-channel attack detection.

III. Network Model, Dataset and Preliminaries

A smartphone-based application is developed to collect data from smartphones, The data collection process is performed with the help of 10 individuals, and five smartphones included *Samsung J7, Huawei Honor, Samsung Grand Prime, Oppo F3, and Oppo F1*. This dataset is the extension of the previously developed dataset [4]. The core purpose of generating this dataset is to provide a high-quality federated learning-oriented dataset and produce side-channel attacks. The dataset also included the user postures, such as noise and movements while typing (i.e., walking, sitting, and standing). The participants were asked to type on a soft keyboard in these three postures. The individuals participating in the experiments hold the smartphone in portrait mode and type with both hands' thumbs. Two parameters are set to ensure the dataset's quality: what data is required and how often is required during the data collection process. The required information is collected with a constant 40 instances/per second (ps) frequency from the keyboard. Several hardware sensors are configured on each smartphone. However, some of the smartphones that participated in the dataset development were not equipped with a gyroscope sensor, and some of the smartphones were used without a magnetometer. We keep only the dataset that has been collected from all three sensors. The developed dataset files consist of 26 alphabets. Each of the alphabet on the keyboard is pressed continuously for almost 2 minutes, and the keyboard readings are stored in a comma-separated (CSV) file. In addition, to each keyboard reading, we assigned a timestamp to ensure that the keyboard readings are well-structured.

The collected raw data is transformed into a sensor event window. From each participant file, a sample of 500 windows is selected. The window size chosen is diverse enough for the classification methods and capturing required data [4]. Therefore, we manually assigned labels to the collected data as ground truth and according to the alphabet. The manually labeled data correctly record the sensor measurements, and map recorded data with corresponding pressed alphabets. After this, based on 160,000 raw sensor data, a feature matrix reading is constructed. Each reading included $3-axis$ of three sensors in the constructed feature matrix.

IV. Background

In recent times, federated learning has been regarded as a machine learning approach with promising results. It has proven itself by leveraging multiple nodes' distributed personalized datasets, such as mobile devices, resulting in better privacy preservation and improved performance. The wide distribution of training data can be seen in federated learning, maintained by workers on mobile devices. A central aggregator updates a global model as it collects local updates from these devices while using the local training data for training the global model during each iteration.

A global model across local data samples held in various decentralized edge servers or devices is trained by FL [26]. Federated learning methods can be categorized into Vertical [27], [28], and horizontal [26], [29]. Datasets sharing a label space while having different feature spaces are handled using procedures of vertical federated learning. Horizontal federated learning [29] is preferred for datasets sharing feature space but differing in samples. The third category of federated transfer learning also exists for datasets differing in both feature and label space.

For exchange and verification of model updates, blockchain was also leveraged in block-chained federated learning introduced by [30]–[32]. For side-channel attacks, horizontal federated learning appears applicable as similar features are shared by traces captured from various devices having the same plaintext and key. Fig. 1 differentiates between centralized learning framework (A) and federated learning framework (B) in the context of data training.

Client updates are combined to produce a new global model when the server uses the algorithm. A subset k of client devices receives a global model w at the training round t. For a particular case, $t = 0$, the same global model trained or initialized randomly on proxy data gives the starting point to client devices. n_k examples exist for local datasets of every client for a given round. Here, k denotes the participating clients' index. In Gboard studies, the typing volume of users relates to n_k. All clients calculate the average gradient g_k with their current model wt on their local data using stochastic gradient descent (SGD).

$\omega_t^k + 1$, which is the local client update is given for any client learning rate ϵ as calculated in Eq. 1.

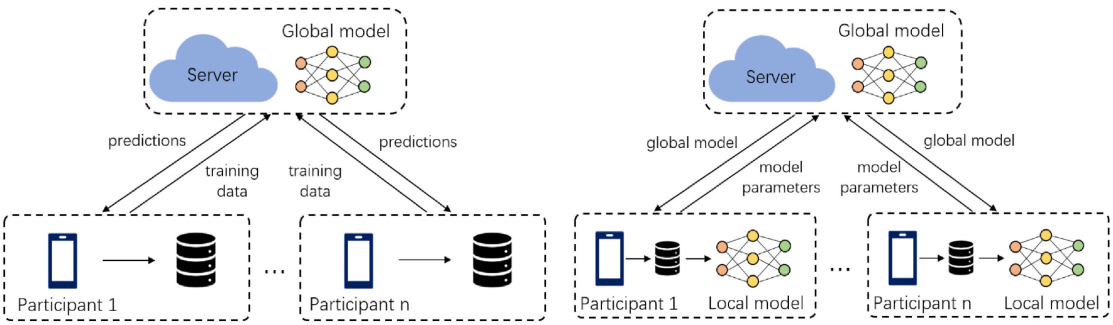
$$\omega_t - \epsilon g k \rightarrow \omega^{k_{t+1}} \tag{1}$$

For obtaining new global model ω_{t+1}, a weighted aggregation is done by the server as Eq. 2

$$\sum_{k=1}^{K} \frac{n_k}{N}\, \omega_{t+1}^{k} \rightarrow \omega_{t+1} \tag{2}$$

Here $N = P$. In a nutshell, SGD updates are computed by clients locally and then received at the server for aggregation. The number of clients per round (global batch size), number of client epochs, and batch size would make the list of hyperparameters. In contrast to server storage, decentralized on-device computation offers less privacy and security risks, even for anonymized server-hosted data. Direct and physical data control can be assured by keeping data on client devices. Each client communicates the transitory-focused and aggregated model updates to the server. The server never stores these client updates; they are processed in memory and discarded immediately after weight vector accumulation.

Content uploaded is restricted to model weights as it follows the principle of data minimization. Lastly, only the aggregate form of results is used such that many client device updates are combined to improve the global model [33]. It is needed from the users in the presented procedure of federated learning that they trust the fact that individual weight uploads will not be scrutinized at the aggregation server. Server training is preferred because entrusting the server with user data is a difficult choice. To address the trust requirement, it is viable to explore additional techniques. Privacy-preserving techniques like differential privacy and secure aggregation complement FL in past research.



(a) Centralized learning framework

(b) Federated learning framework

Fig. 1. High Level Illustration of Federated Learning vs Centralized Learning.

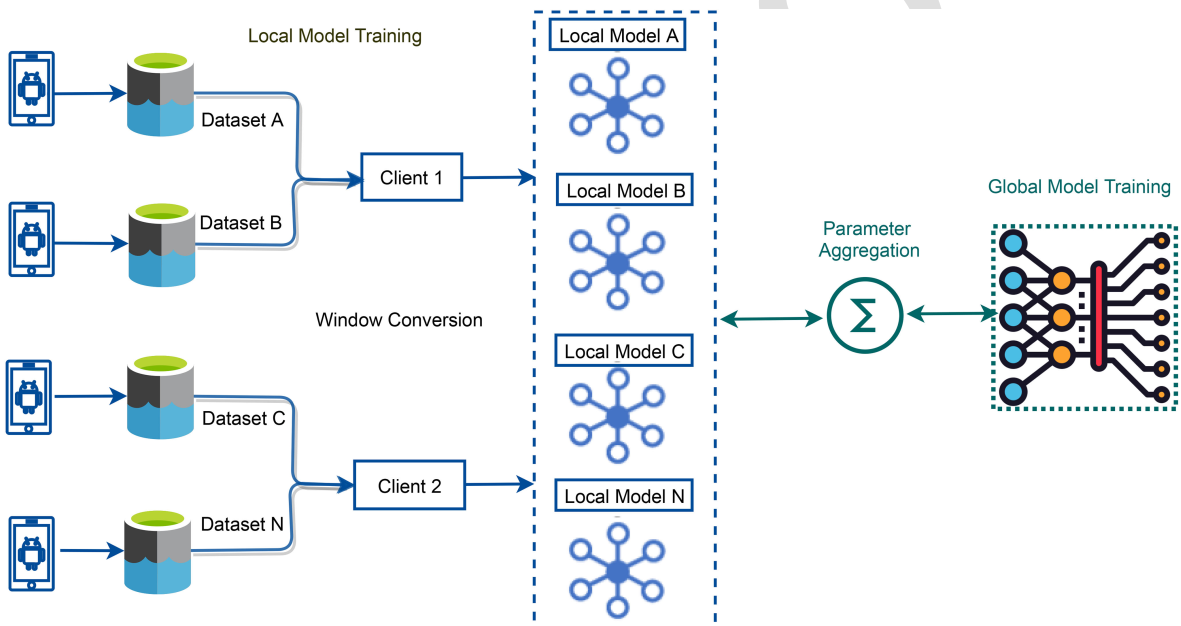


Fig. 2. Proposed Deep Neural Network using Federated Deep Learning framework for Side Channel Attack Detection.

V. Proposed Framework

This section elucidates the concepts of federated learning, network types, and model architecture as they are the building blocks of our proposed methods. Fig. 2 depicts the DNN training process utilizing Federated Learning. It consists of three significant steps. The first step is Training Initialization. Depending on the intended application, the FL server, a cloud server, sets the required data type and training hyperparameters, such as the number of epochs, learning rate, and activation function.

In addition, the FL server initially builds a global model. Specifications and various hyperparameters are sent to participating DNN models (clients). It is worth noting that the FL server determines both the learning rate and the model epochs. The second step is the training of the DNN model. Each client begins collecting new information and changes the parameters of its local model (L_x^y), and it depends on the global model (M^y), where y is the index of the current iteration. Each client also seeks ideal settings to reduce the loss. Now we send the updated parameters regularly to the FL server. Step three is global model aggregation. In this step, we aggregate the results of multiple clients at the server end and send back the updated parameters to each client. The FL server's goal is to reduce the mean global loss function by using this (3).

$$Loss(M^y) = \frac{1}{N}\sum_{x=1}^{x=N} Loss(L_x^y) \tag{3}$$

It is noticeable that these steps are performed till the desired accuracy is obtained or the loss function gradually decreases.

A. Network Types

A variety of objectives, privacy settings, and network types can be chosen for FL. In addition to other machine learning-related complex criteria, network type significantly impacts the definition of a federated learning system's performance and security-related benefits. It can be categorized into broad

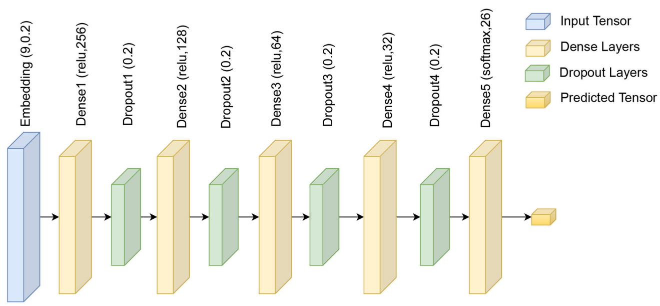


Fig. 3. Proposed Deep Neural Network Structure.

categories of cross-silo and cross-device. For the cross-silo learning, a smaller number of clients, usually ranging from 10 to 100, would pursue shared objectives through cooperation. Connectivity in such a case is likely to be more reliable, the client has powerful computing resources, and data sets are much more significant. On the other hand, the intelligence services of a central provider are used by a more significant number of client devices (up to millions) in cross-device learning. In this category, small client data sets are usually used while there is a high intermittence of network connectivity.

B. Model Architecture

The number of layers and neurons is critical in modeling neural network structures. The dimension of the training set predetermines the number of input and output neurons in a DNN. Various clients train the DNN model. The DNN model's structure consists of the input layer, multiple hidden layers, and an output layer. We use a sequential DNN model composed of a single input layer. The input dimension of this layer is nine, and the last dense layers are composed of 26 output classes. After an input layer, we used a dense layer with 256 units and a relu activation function. The DNN model architectures comprise 5 dropout layers and four dense layers. The dropout layers are used to reduce the over-fitting of the model. The dropout layers value is 0.2. The four hidden layers are composed of the relu activation function and $256, 128, 64, and 32$ units. The fully connected layer used the softmax activation function to predict a multinomial probability distribution. It is used for multi-class classification problems. The DNN model used adam optimizer to reduce the loss and to calculate the loss, the DNN model used categorical_crossentropy. Each dense layer used the relu activation function, and the fully connected layer used the softmax activation function to solve the multi-class classification problem. Fig. 3 presents the DNN structure used in this study for experiments.

VI. Experimental Results and Analysis

This section presents the experimental results and analysis of the proposed framework. In addition, we examine the effect of various parameters on the performance of our framework. The experiments were conducted on a proposed dataset gathered from 10 Android smartphone users. One server and two clients are involved in the experiments. Starting with random weight initialization, the proposed dataset is utilized for training the DNN model specified in Section V-B. Initially, we collected 12,999 dataset samples. We split this dataset into two parts. 75% of the data is used to train the model, and the remaining 25% is used for testing purposes. The 2 clients trained the DNN model on the N number of side-channel datasets. To minimize the loss, we evaluate each client's outcomes three times. The experiments were performed on 26 classes; we used a label encoder to convert the labels into a numeric form into the machine-readable form. The results of N clients were combined on the server-side. The end standard evaluation metrics, including accuracy, precision, recall, and F1-score, were utilized in the experiments. After aggregation at the server end, the DNN model achieves an accuracy of 80.09%.

TABLE I
Client 1 Results

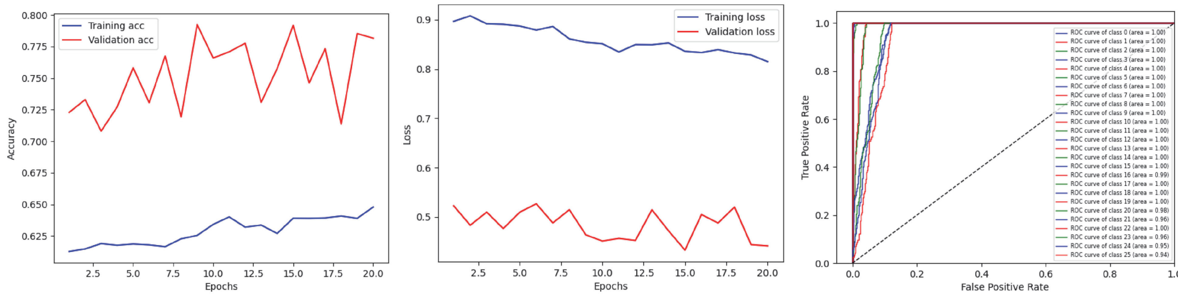
Experiments	Accuracy%	Precision%	Recall%	F1-Score%
Round 1	73	70	73	69
Round 2	68	69	68	62
Round 3	78	81	78	74

A. Server-Based Training With Log Data

An FL system with one central parameter server and two clients is considered. The server manages the selection of each node/client at the start of the model training process and aggregates received model changes. Server-based training of the DNN model relies on data logged. Logs are anonymized and cleansed of personally-identifying information before training. For the server, we initialize various parameters, then we set num_rounds=3, which means that we evaluate our experiments three times. After the FL starts, we go through three rounds. Each round has two stages fit_round and evaluate_round. In the fit_round, the clients send the training results to the server, and in the evaluate_round, both the clients send the testing results to the server, and the server aggregates the results. The above process took 50.57 minutes to complete the experimental process. The server combines the results of N clients and finds that the DNN model has the greatest accuracy of 80.09%. This result demonstrates that the DNN model detects side-channel attacks accurately.

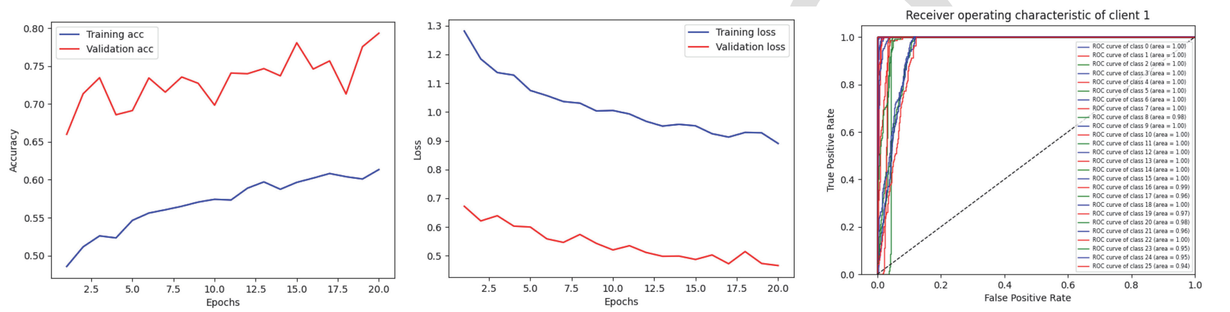
B. Federated Training With Client 1 Caches

Client 1 used a sequential DNN model for experiments. The DNN model contains five dense layers and five dropout layers. Initially, the input dimension is nine, and the last dense layers (fully connected layers) are composed of 26 classes. The model used adam optimizer as an activation function. This model computed loss using categorical_crossentropy. As mentioned in Section VI-A, client 1 evaluates experiments three times and sends the experimental result to a server in two stages fit_round and evaluate_round. The DNN model performed experiments in three rounds. The experimental results of client one are presented in Table I. Experiments show that the results are evaluated three times using standard evaluation measures (accuracy, precision, recall, and F1-score). In the first round, the DNN model exhibits



(a) Training and Validation accuracy of round 3 (b) Training and Validation loss of round 3 (c) Class-wise Receiver Operation Characteristic score of round 3

Fig. 4. Visualization of highest results obtained from client 1.



(a) Training and Validation accuracy of round 3 (b) Training and Validation loss of round 3 (c) Class-wise Receiver Operation Characteristic score of round 3

Fig. 5. Visualization of highest results obtained from client 2.

an accuracy score of 73% with 70% precision, 73% recall, and 69% F1-score. Again we evaluate the results to prevent our model from over-fitting; we obtained low results compared to round 1. The results obtained in round 2 are; 68% accuracy, 69% precision, 68% recall, and 62% F1-score. As we set number-of-rounds=3, we re-evaluate the results for round 3. This time we obtained the highest results compared to the previous results of client 1. We obtained the highest accuracy score of 78% in round 3, 81% precision, 78% recall, and 74% F1-score. Round 3 exhibits the highest results from client 1. The highest results are visualized in Fig. 4. The training and validation accuracy is shown in Fig. 4(a). The Figure shows that client 1 in round 3 achieved the highest validation accuracy compared to training accuracy because of the fewer samples in the validation set. Fig. 4(b) shows the training and validation loss; Because of less number of samples in the validation set, the validation loss is less than the training loss. During the training process, the loss decreases on every epoch, which means, on the other hand, the model performance is increasing. In the end, Fig. 4(c) shows the Receiver Operating Characteristic's (ROC) curve of each class. Most classes gained a ROC score of 1, which means that this model performs well on the dataset. The ROC curves, closer to the top-left corner, indicate better performance.

C. Federated Training With Client 2 Caches

Client 2 used the same sequential DNN model in their experiments. This model has the same experimental settings as

TABLE II
CLIENT 2 RESULTS

Experiments	Accuracy%	Precision%	Recall%	F1-Score%
Round 1	67	63	67	62
Round 2	79	78	79	75
Round 3	75	73	75	70

mentioned in Section VI-B. The dense layer of this model also contains 26 classes. The model uses the same Adam optimizer as an activation function and categorical_crossentropy to compute loss. Client 2 also evaluates experiments in three rounds and sends the experimental result to a server in two stages fit_round and evaluate_round. Client 2's experimental findings are given in Table II. The outcomes are examined three times using established evaluation techniques (accuracy, precision, recall, and F1-score). The experiments are performed in three rounds as shown in Table II. In the first round, the DNN model exhibits an accuracy score of 67% with 63% precision, 67% recall, and 62% F1-score. Again we evaluated the results to prevent our model from over-fitting; we obtained higher results than in round 1. The results obtained in round 2 are; 79% accuracy, 78% precision, 79% recall, and 75% F1-score. At the server end, we set the number of rounds =3; we re-evaluated the results for the third round. We obtained an accuracy score of 75% in round 3, 73% precision, 75% recall, and 70% F1-score. Round 2 exhibits the highest results from client 2. The highest results are visualized in Fig. 5. The training and validation accuracy is shown in Fig. 5(a). The Figure shows that client 2 in the second round achieved the highest validation accuracy compared to training accuracy because of the fewer samples in the validation set. Fig. 5(b) shows the training and validation loss; Because of less number of samples in the validation set, the validation loss is less than the training loss. During the training process, the loss decreases on every epoch, which means, on the other hand, the model performance is increasing. In the end, Fig. 5(c) shows the ROC curve of each class. Most classes gained a ROC score of 1, which means that this model performs well on the dataset. The ROC curves, closer to the top-left corner, indicate better performance.

VII. Conclusion

This study proposed an FL-based DNN model for channel attack detection. We have collected a dataset from Android users while typing on a soft keyboard. The dataset is divided into two windows to make two local clients' training models. We have trained the DNN model on two clients, and the results were aggregated on the server-side with 80.09% accuracy. Each client evaluates the findings three times to limit the over-fitting factor. Client 1 achieved the best results in the third round with 78% accuracy, while client 2 achieved the best results in the second round with 79% accuracy. The DNN model obtained a ROC curve score of more than 95% for each class, indicating that the model performed admirably on the provided dataset. The results show that federated learning effectively identifies channel attacks, and the system efficiency study reveals that end-to-end training time and memory cost are both inexpensive and promising for resource-constrained IoT devices. In the future, we intend to examine this phenomenon further by training additional models with other combinations of smartphone devices.

References

[1] A. R. Javed, M. U. Sarwar, M. O. Beg, M. Asim, T. Baker, and H. Tawfik, "A collaborative healthcare framework for shared healthcare plan with ambient intelligence," *Hum.-Centric Comput. Inf. Sci.*, vol. 10, no. 1, pp. 1–21, 2020.

[2] M. Rizwan *et al.*, "Risk monitoring strategy for confidentiality of healthcare information," *Comput. Elect. Eng.*, vol. 100, 2022, Art. no. 107833.

[3] H. Xiong *et al.*, "On the design of blockchain-based ECDSA with fault-tolerant batch verication protocol for blockchain-enabled IoMT," *IEEE J. Biomed. Health Informat.*, to be published, doi: 10.1109/JBHI.2021.3112693.

[4] A. R. Javed, T. Baker, M. Asim, M. Beg, and A. H. Al-Bayatti, "Alphalogger: Detecting motion-based side-channel attack using smartphone keystrokes," *J. Ambient Intell. Humanized Comput.*, pp. 1–14, 2020.

[5] A. R. Javed, S. U. Rehman, M. U. Khan, M. Alazab, and H. U. Khan, "Betalogger: Smartphone sensor-based side-channel attack detection and text inference using language modeling and dense multilayer neural network," *Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 5, pp. 1–17, 2021.

[6] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion." *HotSec*, vol. 11, no. 2011, 2011, Art. no. 9.

[7] P. Voigt and A. V. dem Bussche, "The EU General Data Protection Regulation (GDPR)" *A Practical Guide*, *1st Ed.*, Berlin, Germany: Springer, vol. 10, 2017, Art. no. 3152676.

[8] B. K. Atchinson and D. M. Fox, "From the field: The politics of the health insurance portability and accountability act," *Health Affairs*, vol. 16, no. 3, pp. 146–150, 1997.

[9] W. Luping, W. Wei, and L. Bo, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 954–964.

[10] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov./Dec. 2020.

[11] A. Z. H. Yapp *et al.*, "Communication-efficient and scalable decentralized federated edge learning," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, 2021, pp. 5032–5035.

[12] J. Song, W. Wang, T. R. Gadekallu, J. Cao, and Y. Liu, "EPPDA: An efficient privacy-preserving data aggregation federated learning scheme," *IEEE Trans. Netw. Sci. Eng.*, to be published, doi: 10.1109/TNSE.2022.3153519.

[13] D. Lia and M. Togan, "Privacy-preserving machine learning using federated learning and secure aggregation," in *Proc. 12th Int. Conf. Electron. Comput. Artif. Intell.*, 2020, pp. 1–6.

[14] Z. Lian, W. Wang, and C. Su, "Cofel: Communication-efficient and optimized federated learning with local differential privacy," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[15] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, 2021, Art. no. 106775.

[16] H. Wang and E. Dubrova, "Federated learning in side-channel analysis," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2020, pp. 257–272.

[17] H. Wang, S. Forsmark, M. Brisfors, and E. Dubrova, "Multi-source training deep-learning side-channel attacks," in *Proc. IEEE 50th Int. Symp. Mult.-Valued Log.*, 2020, pp. 58–63.

[18] H. Wang and E. Dubrova, "Tandem deep learning side-channel attack on FPGA implementation of AES," *SN Comput. Sci.*, vol. 2, no. 5, pp. 1–12, 2021.

[19] A. Mosavi, F. S. Hosseini, B. Choubin, M. Goodarzi, A. A. Dineva, and E. R. Sardooi, "Ensemble boosting and bagging based machine learning models for groundwater potential prediction," *Water Resour. Manage.*, vol. 35, no. 1, pp. 23–37, 2021.

[20] P. Perconti and A. Plebe, "Deep learning and cognitive science," *Cognition*, vol. 203, 2020, Art. no. 104365.

[21] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proc. 1st ACM Workshop Netw. Syst. Appl. Mobile Handhelds*, 2009, pp. 31–36.

[22] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Appl. Soft Comput.*, vol. 49, pp. 1162–1174, 2016.

[23] J. Cui, L. Wang, X. Zhao, and H. Zhang, "Towards predictive analysis of android vulnerability using statistical codes and machine learning for IoT applications," *Comput. Commun.*, vol. 155, pp. 125–131, 2020.

[24] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.*, 1996, pp. 104–113.

[25] T. Gellersen, O. Seker, and T. Eisenbarth, "Differential power analysis of the picnic signature scheme," in *Proc. Int. Conf. Post-Quantum Cryptogr.*, 2021, pp. 177–194.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[27] L. Li, Y. Fan, and K.-Y. Lin, "A survey on federated learning," in *Proc. IEEE 16th Int. Conf. Control Automat.*, 2020, pp. 791–796.

[28] C. Wang, J. Liang, M. Huang, B. Bai, K. Bai, and H. Li, "Hybrid differentially private federated learning on vertically partitioned data," 2020, *arXiv:2009.02763*.

[29] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, "Fedu: A unified framework for federated multi-task learning with laplacian regularization," 2021, *arXiv:2102.07148*.

[30] Y. Qu *et al.*, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.

[31] W. Wang *et al.*, "Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3117762.

[32] M. K. Hasan *et al.*, "Lightweight cryptographic algorithms for guessing attack protection in complex Internet of Things applications," *Complexity*, vol. 2021, 2021.

[33] B. Han, R. Jhaveri, H. Wang, D. Qiao, and J. Du, "Application of robust zero-watermarking scheme based on federated learning for securing the healthcare data," *IEEE J. Biomed. Health Informat.*, to be published, doi: 10.1109/JBHI.2021.3123936.