Secure Federated Learning Based on Coded Distributed Computing

Shaoliang Zhu[1], Alia Asheralieva[1], Md Monjurul Karim[1], Dusit Niyato[2], and Khuhawar Arif Raza[1]
[1]*Department of Computer Science and Engineering*, *Southern University of Science and Technology*, Shenzhen, China,
e-mail:{11710827@mail.sustech.edu.cn, aasheralieva@gmail.com, karim@mail.sustech.edu.cn, khuhawar@mail.sustech.edu.cn}
[2]*School of Computer Science and Engineering*, *Nanyang Technological University*, Singapore, e-mail:{dniyato@ntu.edu.sg}

2021 IEEE Globecom Workshops (GC Wkshps) | 978-1-6654-2390-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/GCWkshps52748.2021.9682063

***Abstract*—Federated learning (FL) enables multiple learning devices to exchange their training results and collaboratively develop a shared learning model without revealing their local data, thereby preserving data privacy. However, contemporary FL models have many drawbacks including limited security against malicious learning devices generating arbitrarily erroneous training results. Recently, a promising concept - coded distributed computing (CDC) has been proposed for maintaining security of various distributed systems by adding computational redundancy to the datasets exchanged in these systems. Although the CDC concept has already been adopted in several applications, it is yet to be applied to FL systems. Accordingly, in this paper, we develop the first integrated FL-CDC model that represents a low-complexity approach for enhancing security of FL systems. We implement the model for predicting the traffic slowness in vehicular applications and verify that the model can effectively secure the system even if the number of malicious devices is large.**

***Index Terms*—Coded Distributed Computing, Federated Learning, Lagrange Polynomial, Vehicular Applications**

I. Introduction

FL is the privacy-preserving machine learning technique that enables multiple learning devices to collaboratively develop a shared learning model without revealing their private data [1], [2]. In particular, in FL, any learning device can estimate the model updates based on the locally-collected and trained data according to the current global model. The model is then consolidated and backed up by a central server in order to allow all learning devices to access the same global model while calculating their latest local updates. Such a process is replicated until the global learning model achieves the targeted accuracy. In this way, the data privacy is preserved because the local training data is not exchanged and, hence, not intercepted on the way from the learning device to the server, as it frequently happens in the conventional distributed learning approaches [1], [2].

Unfortunately, the contemporary FL models suffer from the limited security against untrustable learning generating arbitrarily erroneous training results, which can affect the entire computing process and, thus, accuracy of the global learning model [1]–[4]. Several solutions (e.g., [4]–[6]) have been proposed to solve this problem. One possible solution, e.g. [5], is to identify all untrustable devices that produce inaccurate training results, and perform computation only on trustworthy devices. However, this solution can be rather compute intensive and, hence, unsuitable for realistic FL systems comprising numerous learning devices, as they require periodical verification of the training results of each learning device [6]. Other solutions are based on blockchains, e.g., [7], the blocks containing training results are verified by the blockchain miners to ensure security of the FL system. Nevertheless, due to block verification process, such solutions may significantly increase the delay and energy overheads, as well as computing costs, which makes them impractical for many time sensitive machine learning tasks, e.g., as in mobile and Internet of Things (IoT) applications. A promising concept called CDC has been proposed recently to enhance security of multi-party computations in distributed systems [6]–[9]. One of the low-complexity CDC techniques, Lagrange coded computing (LCC), utilizes the Lagrange polynomial to perform decoding of the polynomial function estimations [10]–[12].

Existing CDC techniques are designated to deal with stragglers (nodes that perform slower than others). However, a few of them tackles with security against malicious nodes. Therefore, compared to other CDC techniques, we use LCC because it provides privacy preservation against malicious nodes, flexibility against straggling nodes, and information-theoretic privacy among colluding nodes [13]. Besides, LCC requires low storage and computation cost, which makes it convenient to implement. As a result, LCC allows the server, i.e., fusion center, to successfully decode the final outcome of distributed computations even when some function estimations by the untrustable parties, e.g., malicious learning devices, are erroneous. Various CDC techniques have been adopted in certain distributed systems, e.g., cloud and edge computing networks. Nevertheless, they have not been implemented in FL models due to the following challenges. First, the architecture of the FL-CDC model based on LCC is not straightforward. In particular, the original LCC model is designed for multi-party computations where the dataset trained by learning devices is produced at the centralized server, i.e., fusion center, whereas in FL model, each device collects its own datasets and uses it to train the model. Second, the original LCC model is designated to operate on polynomial functions, while most common FL models are based on the Neural Networks (NNs) [1], [2], [10]. As such, to apply LCC, the NN function used in FL must be first approximated with a polynomial function.

Accordingly, this paper aims to address the aforementioned challenges in order to apply the concept of LCC to the FL

978-1-6654-2390-8/21/$31.00 ©2021 IEEE

model. The main contributions of the paper are as follows:

- We develop a novel architecture of the integrated FL-CDC model based on LCC. The main challenge of developing the architecture is that the original LCC model is designed to operate in the settings of multi-party computations [10] that are rather different from the FL settings. Hence, to fit the scenario of FL, the proposed FL-CDC architecture specifies the details of data collection and parameter generation by learning devices.
- To adopt LCC in the common FL models, we derive the polynomial approximation of the NN function used in the system. The proposed approximation method is based on the least square approximation [14] where the degree of the polynomial approximation functions is decided to meet the demand of FL-CDC model based on the trade-off between the estimation accuracy and time complexity.
- We present a novel implementation of the FL-CDC model for vehicular applications. The application predicts the slowness of the road traffic based on certain features collected by vehicles. The performance of the FL-CDC model is evaluated based on the extensive simulations conducted under different settings.

The rest of the paper is as follows. In Section II, we review related works. In Section III, we present the system model. In Section IV, we derive the polynomial approximation and develop the proposed integrated FL-CDC model. In Section V, we propose the implementation of the FL-CDC model for vehicular applications and evaluate its performance.

II. Related Work

Existing works in the area of FL can be arbitrarily divided into three categories: (1) FL models (e.g., [11], [12], [15]) to improve the learning efficiency measured in terms of estimation accuracy and/or convergence time. These works include the models designed for supervised and unsupervised learning based on the assumption that all learning devices are trustworthy. (2) FL applications (e.g., [16], [17]) in practical networks, such as mobile and multi-access edge computing networks with the focus on improving the communication efficiency and incentivizing, e.g., using some monetary rewards, the learning devices to complete their learning tasks considering that all learning devices are trustworthy. (3) FL techniques that presume the existence of untrustable, e.g., malicious and malfunctioning, learning devices (e.g., [4], [18]). To deal with such devices, two possible solutions are proposed. The first solution [19], [20] is based on the coding method in which the final results of FL computed at the server can be decoded correctly even if some training results returned by untrustable learning devices are erroneous. The main disadvantage of this method is that it is designed for a very simple FL model based on linear regression, which is rarely used in the state-of-the-art FL algorithms due to low convergence rate and high estimation errors for many learning tasks. Another solution (e.g., [21]–[23]) is based on blockchains where the computing results returned by malicious devices are verified by the set of validators. This solution, however, can increase the delay and computational costs since each validator must apply a verification to each training result generated by learning devices. Consequently, the blockchain-based FL is inapplicable for many delay-sensitive and compute-intensive IoT applications.

III. System Model

A. System Model

We consider a FL system (shown in Fig. 1a) that includes a fusion center, i.e., central server, and N learning devices in the set $\mathbf{N} = \{1, ..., N\}$, each of which produces its local dataset X_n. The dataset X_n can be represented by the data generated by mobile or IoT applications running on the learning device. Each learning device sends their dataset to nearby local wireless node that collects these datasets and converts them into local models. At t^{th} update of the FL model, each learning device n needs to find the new model parameters, i.e., weights, $w_n^{(t)}$ based on the previous model parameters and the gradient of the current local loss function, as [24]

$$w_n^{(t)} = w^{(t-1)} + \alpha_t \nabla C^{(t)}, \forall n \in \mathrm{N}, \tag{1}$$

where α_t is the learning rate; C_n is the loss function of the local FL model of the learning device n. After this, the training results of learning devices are aggregated at the fusion center as [3]:

$$w^{(t)} = \frac{1}{N} \sum_{n \in \mathrm{N}} w_n^{(t)}. \tag{2}$$

Hence, the loss of the global FL model is given by:

$$\nabla C^{(t)} = \frac{1}{N} \sum_{n \in \mathrm{N}} C_n^{(t)}. \tag{3}$$

As such, at each iteration of the FL, each learning device n updates the weights $w_n^{(t)}$ of the local model to minimize its model loss C_n to achieve a certain predefined accuracy θ_n ($0<\theta_n<1$). The FL model converges when the global model achieves some predifined global accuracy ε ($0<\varepsilon<1$). The number of iterations needed for the global FL model to converge is bounded by [24]:

$$J(\varepsilon, \theta) = \frac{O(log(1/\varepsilon))}{1-\theta} = \frac{O(log(1/\varepsilon))}{1-(\sum_{n \in \mathrm{N}} \theta_n)/N}, \tag{4}$$

where $\theta = (\sum_{n \in \mathrm{N}} \theta_n)/N$ is the average local accuracy of learning devices, and the number of iterations needed for the local FL model of the learning device $n \in \mathbf{N}$ to converge is bounded by [24]:

$$O(log(1/\theta_n)), \forall n \in \mathrm{N}, \tag{5}$$

From (4), both the local accuracy θ and global accuracy ε of the FL model have a notable impact on its convergence time $J(\theta, \varepsilon)$. In particular, since $\partial J(\theta, \varepsilon)/\partial \varepsilon \leq 0$ and $\partial J(\theta, \varepsilon)/\partial \theta \geq 0$, the convergence time $J(\theta, \varepsilon)$ decreases with ε and increases with θ. For example, if $\theta = \sum_{n \in \mathrm{N}} \theta_n \to 1$, i.e., the average local training result of learning devices is inaccurate (e.g., erroneous), we have $J(\theta, \varepsilon) \to \infty$, i.e., the

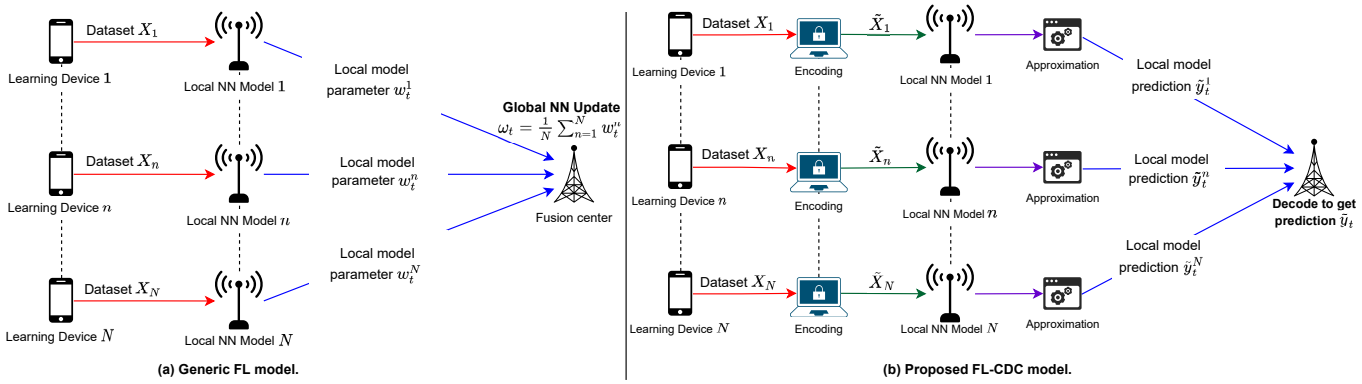

Fig. 1. Comparison of the generic FL and proposed FL-CDC model.

convergence time of the FL model tends to infinity, so that the model will not converge to a fixed-point solution. As such, the performance, i.e., convergence time $J(\theta, \varepsilon)$, of the FL model can degrade significantly due to erroneous training results produced by untrustable (or malicious) learning devices. As explained in Section I, although several solutions, e.g., [4]–[6], have been proposed to address this issue, most of them are rather compute-intensive and can significantly increase the delay and energy costs. Hence, in this paper, we consider an alternative LCC-based approach [10], [12] to deal with untrustable learning devices in the FL system. We begin by presenting the basic idea behind the concept of LCC.

B. Concept of Lagrange Coded Computing

The concept of CDC [6]–[9] has been proposed to solve the main issues of multi-party computations, e.g., untrustable computing devices producing erroneous results. LCC is the suitable solution among current CDC schemes, as it allows a relatively simple implementation for many computing tasks [10], [25], [26]. LCC is utilized to perform the evaluation of a polynomial function f of some degree $deg(f)$ over the dataset $X = \{X_i\}_{i=1}^{K}$ partitioned into K batches in an effective parallel manner by the set $\mathbf{N}$ of N devices, each of which can be untrustable, e.g., malicious, with the goal to compute the final output $Y = \{y_i = f(X_i)\}_{i=1}^{K}$. The central idea behind LCC is to utilize the Lagrange polynomial to encode the input dataset in order to establish the statistical redundancy in a novel coded manner across the devices perfoming computations. This ensures that the final output can be decoded even if some computing results returned by untrustable devices are erroneous [10]. In short, the details of the LCC operation are as follows.

The distributed computing system considered in LCC includes the "master", e.g., fusion center, that offloads computations to N "workers", e.g., learning devices. The set X submitted to workers comprises K intputs, $\{X_i\}_{i=1}^{K}$, and the computing task of each worker is defined by a polynomial function f. In order to encode its data inputs, the master selects K distinct elements, e.g., $\{\beta_i\}_{i=1}^{K}$, and constructs the Lagrange interpolation polynomial u of degree $deg(u) \leq K-1$, defined as [10]

$$u(z) = \sum_{j\in\{1,\ldots,K\}} X_j \prod_{i\in\{1,\ldots,K\}\setminus j} \frac{z-\beta_i}{\beta_j-\beta_i}, \tag{6}$$

so that

$$u(\beta_i) = X_i, \forall i \in \{1,\ldots,K\} \tag{7}$$

holds. After this, the master selects N distinct elements $\{\alpha_i\}_{i=1}^{N}$, such that $\{\alpha_n\}_{n=1}^{N} \cap \{\beta_i\}_{i=1}^{K} = \emptyset$, and then, encodes the data input $\widetilde{X}_i$ sent to worker i as [10]

$$\widetilde{X}_n = u(\alpha_n), n \in \mathbf{N}. \tag{8}$$

Upon receiving its encoded input $\widetilde{X}_n$, the worker computes $\widetilde{Y}_n = f(\widetilde{X}_n)$ and returns the obtained encoded computing result $\widetilde{Y}_n$ back to the master. Accordingly, the master will receive the encoded set $\{\tilde{Y}_n\}_{n=1}^{N}$ which contains N results, up to A of which can be erroneous [10].

Accordingly, to evaluate the final output Y, the master must interpolate the obtained polynomial $f(u)$. Typically, interpolation of the polynomial $f(u)$ of degree $deg(f(u)) = deg(u)deg(f) \leq (K-1)deg(f)$ requires $(K-1)deg(f)+1$ evaluations at distinct points. However, since 2 additional evaluations are required for each erroneous result, the master needs a Reed-Solomon decoder with $2A$ additional evaluations to deal with up to A possibly malicious workers. As a result, the system is A-secure if the final output can be decoded successfully in the presence of up to A possibly malicious workers as long as [10]

$$(K-1)deg(f) + 2A + 1 \leq N. \tag{9}$$

After obtain all coefficients of $f(u)$ through Reed-Solomon decoding, the master evaluates its final decoded output according to $Y = \{Y_i\}_{i=1}^{K} = \{f(X_i)\}_{i=1}^{K} = \{f(u(\beta_i))\}_{i=1}^{K}$.

IV. Integrated FL-CDC Model

A. Polynomial Approximation of NN Functions

Note that since LCC relies on Reed-Solomon decoding, it can only be performed on polynomial functions [10]. However, computations defined in a neural network includes activation

function in neurons and other non-linear functions [27]. Hence, LCC cannot be directly applied to the NNs. In this case, the nonlinear NN functions must be replaced by their polynomial approximations. In paticular, the Weierstrass approximation theorem [28] below provides the necessary theoretical foundation for the feasibility of the approximation.

Theorem 1 (Weierstrass approximation). Suppose g is a continuous real-valued function defined on the real interval $[a, b]$. For every η>0, there exists a polynomial f such that for all x in $[a, b]$, we have $|g(x) - f(x)|<\eta$.

The above theorem states that any continuous function g defined on the closed interval $[a, b]$ can be approximated as tightly as needed by the polynomial function f. However, Weierstrass approximation theorem does not offer a specific algorithm to generate such an approximation. Possible methods to construct the polynomial approximation of the NN functions are through the Taylor series [27] or Chebyshev polynomial [29]. In particular, in this paper, least square approximation is selected because different from other approximation methods, least square approximation is able to achieve small approximation error in a specific input range when the degree of polynomial has already been determined [14]. Least square approximation constructs a group of polynomial coefficients to fit the approximating polynomial function of a given degree. The parameter generation function in the least square approximation enables restricting the input data of the approximating polynomial in a specific range. The time complexity of least square approximation is $max(O((deg(f))^3), O(n(deg(f))^2)$ where n is the number of sample points used in the least square approximation [14]. However, typically, the number of sample points is larger than the expected degree of the polynomial, so that $O((deg(f))^3) \leq O(n(deg(f))^2$ and, hence, the estimated time complexity of the least square approximation is $O(n(deg(f))^2)$.

B. Proposed Integrated FL-CDC Model

Fig. 1b illustrates the proposed model of the integrated FL-CDC system based on LCC. In the model, during the learning process, devices receive encoding parameters predefined by the fusion center and encode the data that they collect according to the parameters assigned to them. The devices apply the function given by the fusion center on the encoded data and send the produced results back to the fusion center. The fusion center updates the machine learning model and sends back the model updates to learning devices.

In general, private local datasets of learning devices are encoded before computation and the NNs used in the FL-CDC model are approximated by polynomials. The fusion center mainly has two functions: i) generating parameters for learning devices and ii) decoding computation results sent back from the devices. At the beginning, the fusion center predefines the parameters $\{\beta_i\}_{i=1}^K$ and $\{\alpha_i\}_{i=1}^N$. Note that, in LCC, each device gets a polynomial function *u* by applying Lagrange interpolation on $\{\beta_i\}_{i=1}^K$ given by the fusion center and data $\{X_i\}_{i=1}^K$ collected by itself. Also, note that unlike the original LCC [10] designed for polynomial computations, when combining LCC with the non-polynomial learning tasks, The approximation of the NN function can only be applied on a specific domain. An element in a worker's encoded data vector can be written as (6).

In our framework, the encoded data must be considered when defininining the polynomial approximation of the NN function. For instance, Taylor expansion of logistic function only convergences in field $[-1, 1]$, and increasing the order of derivative in the approximation does not necessarily reduce the error if the value of input may be very large. The most popular normalizing method is to normalize data into $[0, 1]$ or $[-1, 1]$, the ideal solution is to make encoded data falls into the same filed of original normalized data. An element in a worker's encoded data vector can be further written as

$$u(\alpha) = (c_1, ..., c_K)(X_i, ..., X_K)^T, \tag{10}$$

where $c_j = \prod_{i\in[K]\setminus j} \frac{\alpha-\beta_i}{\beta_j-\beta_i}$.

In the case that data are normalized into $[0, 1]$, as

$$\begin{aligned} &\forall X_i, ..., X_k, (c_1, ..., c_K)(X_i, ..., X_K)^T \in [0, 1] \\ &\quad \Longleftrightarrow c_1, ..., c_k \geq 0 \ and \ c_1 + ... + c_k \leq 1. \end{aligned} \tag{11}$$

In the case that data are normalized into $[-1, 1]$, as

$$\begin{aligned} &\forall X_i, ..., X_k, (c_1, ..., c_K)(X_i, ..., X_K)^T \in [-1, 1] \\ &\quad \Longleftrightarrow |c_1| + ... + |c_k| \leq 1. \end{aligned} \tag{12}$$

The sum of elements in vector $(c_1, ..., c_K)$ given by Lagrange interpolation is always 1, and negative elements always exist if the length of the vector is larger than 1. Therefore, the range of encoded data is always larger than the range of original data. However, the sum of the absolute value of elements can be restricted in any value larger than 1, which means in the case that data are normalized into $[-1, 1]$, the absolute value of encoded data can be restricted in any positive value larger than 1. Once the range of encoded data is determined, the upper bound of approximation error can be determined. For example, for $C > 1$, the range of encoded data is expected to be $[-C, C]$, since

$$\begin{aligned} &\forall X_i, ..., X_k, (c_1, ..., c_K)(X_i, ..., X_K)^T \in [-C, C] \\ &\quad \Longleftrightarrow |c_1| + ... + |c_k| \leq C. \end{aligned} \tag{13}$$

By solving the absolute value inequality, the fusion center generates LCC parameters that is able to restrict encoded data in the expected range whatever the original data is. After the parameters are generated, the fusion center sends them to learning devices that are currently engaged in the learning and waits for their computation results. The fusion center decodes the outcome using the Reed-Solomon decoding. In proposition-1, we estimate the time complexity of the proposed FL-CDC model.

Proposition 1. The time complexity of the FL-CDC model is $O(N\left(K^2 + n(deg(f))^2\right) + K + N^3)$, where n is the number

of sample points used in the least square approximation.

Proof: The execution of the FL-CDC model can be divided into four parts: parameter generation, encoding, approximation, and decoding. The encoding in the FL-CDC model represents interpolation of a polynomial at K points with the time complexity of $O(K^2)$. As mentioned in Subsection IV-B, the time complexity of the approximation of function f is $O(n(deg(f))^2)$. Therefore, the total time complexity of executing encoding and approximation in the FL-CDC model for N devices is $O(N\left(K^2+n(deg(f))^2\right))$. During parameter generation, the center generates $\{\beta_i\}_{i=1}^{K}$ and $\{\alpha_i\}_{i=1}^{N}$, which has the time complexity of $O(K+N)$. Furthermore, the time complexity of the Reed-Solomon decoder is $O((R+2A)^3)$ where R is the recover threshold. Since in our case, $R+2A$ is at most N, the total time complexity of executing parameter generation and decoding in the FL-CDC model at the center is $O(K+N+N^3)=O(K+N^3)$. As a result, the total time complexity of the FL-CDC model is $O(N\left(K^2+n(deg(f))^2\right)+K+N^3)$. ■

V. Implementation for Vehicular Applications

We now describe the implementation of the proposed FL-CDC model based on LCC for vehicular applications aimed to predict the behavior of the urban traffic. The simulation model of the system has been developed in MATLAB. The data used in the application are collected from environments by vehicles in urban area [30]. Totally there are 16 features used to describe the traffic, including immobilized bus, broken truck, vehicle excess, accident victim, running over, fire vehicles, occurrence involving freight, incident involving dangerous freight, lack of electricity, fire, point of flooding, manifestations, defect in the network of trolleybuses, tree on the road, semaphore off and intermittent semaphore. The goal of the application is to predict the slowness in traffic in percentage based on the features collected by vehicles.

The model used in the paper is a single layer NN. Each vehicle participated in the learning owns a local data vector of length K. The model parameter is a weight vector of length K. Due to that data are normalized into $[-1,1]$, the function used to do prediction is

$$f(X)=\frac{1-e^{-wX^T}}{1+e^{-wX^T}}. \tag{14}$$

where w is the model parameter and X is the local data. The fusion center collects the computation results of f from users participating in current learning process.

Cross entropy is used as loss function, given by

$$C=-(y\ln\sigma+(1-y)\ln(1-\sigma)), \tag{15}$$

where $\sigma=\frac{1+f'(X)}{2}$ and y is the real value of prediction. In order to apply LCC in FL-CDC model, f is replaced with its least square approximation f' Besides, in the FL-CDC model, each user encodes its local data vector according to the encoding parameters assigned by the fusion center.

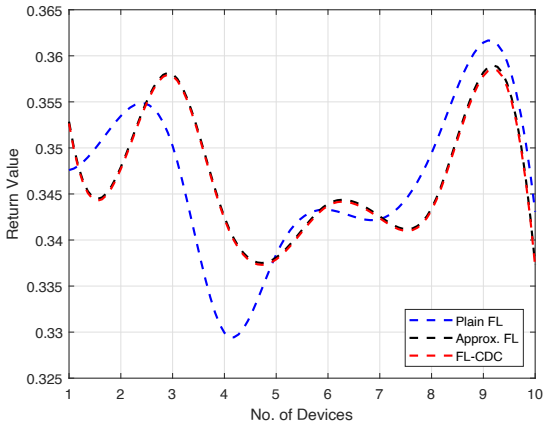

Fig. 2. Return value without malicious devices for N from 1 to 10.

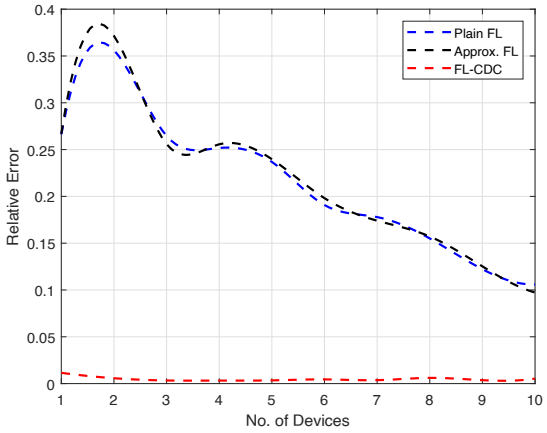

Fig. 3. Relative error without malicious devices for N from 1 to 10.

In Fig. 2, we implement the vehicular application by using plain FL model, approximated FL model and FL-CDC model. Fig. 2 illustrates the return value of the last iteration of the models with the number of workers varying from 1 to 10. In approximated FL mode, the least square approximation is done by using 21 sample points uniformly distributed on $[-2,2]$. We do not introduce malicious devices in the implementation. Due to that the computation results of devices are highly consistent, there is no decode failure in FL-CDC model. Therefore, the two lines representing return values of approximated FL model and FL-CDC model are overlapping in Fig. 2. Fig. 3 illustrates the relative error of the return value between two approximated FL models and plain FL model. The relative error of approximated FL model and FL-CDC model are represented by one line since they are identical. As shown in Fig. 3, the maximum value of relative error is less than 0.045 during simulation.

Fig. 4 and Fig. 5 show the time dynamics of the return predicted value for N=10 devices with no malicious devices (Fig. 4) and with 3 malicious devices (Fig. 5). From Fig. 4, if there are no malicious devices, the value predicted in the FL-CDC model is almost the same as the value predicted in the plain FL model. On the other hand, from Fig. 5, with

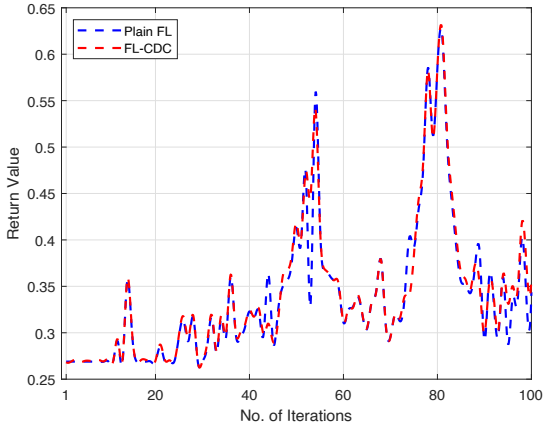

Fig. 4. Return value without malicious devices for N=10.

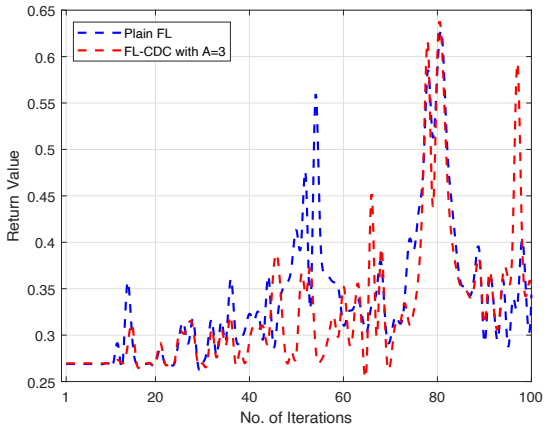

Fig. 5. Return value with 3 malicious devices for N=10.

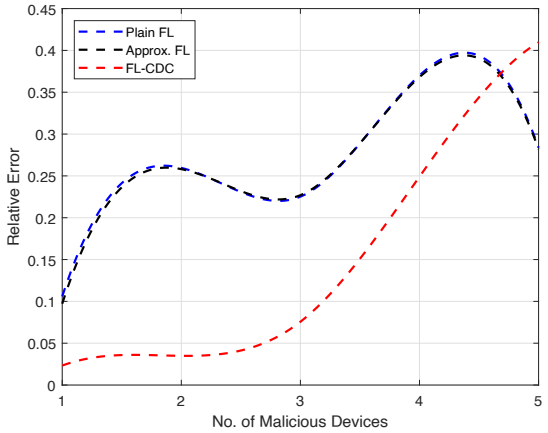

Fig. 6. Relative error with 5 malicious devices for N=10.

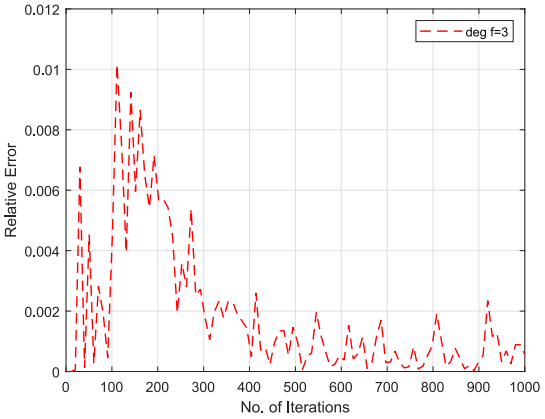

Fig. 7. Relative approximation error with the degree 3

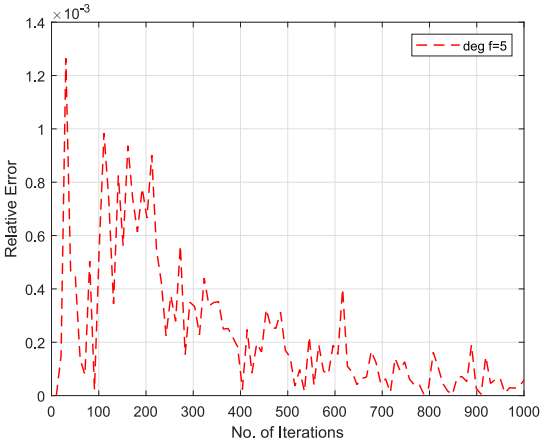

Fig. 8. Relative approximation error with the degree 5

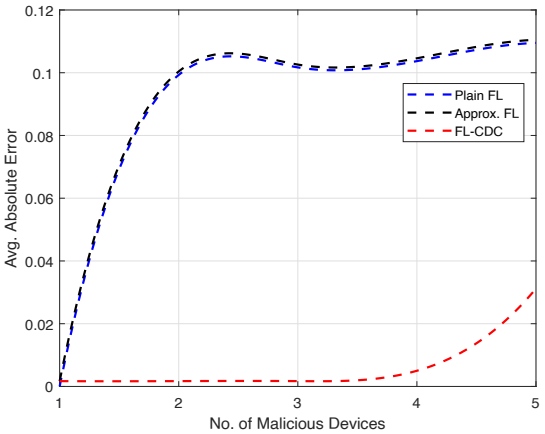

Fig. 9. Avg. absolute error depending on the number of malicious devices.

3 malicious devices, the proposed FL-CDC model produces a more accurate prediction than the plain FL model. Next, in Fig. 6 we implement the FL-CDC model by fixing the number of devices as 10 with the number of malicious devices varying from 10% to 50%. Fig. 6 illustrates return values of the last iteration from plain FL model, approximated FL model and FL-CDC model. As shown in Fig. 6, relative error of plain FL model and approximated FL model can reach 0.35 during simulations, while decode failure does not happen in FL-CDC model until the number of malicious devices reaches 40%. It shows that FL-CDC model can decode data successfully when the number of malicious workers does not exceed the recover threshold.

Figs 7 and 8 show the time dynamics of the relative

approximation error with the degrees 3 and 5, respectively. From these figures, the approximation loss reduces with time for both degrees. Furthermore, the highest value of the relative error reached at the beginning of simulations is rather small, i.e., comparable to 0.01. Fig. 9 shows the average absolute average error depending on the number of malicious devices for the total number of devices N=10. From this figure, the proposed FL-CDC model can secure the system against upto 3 (or 30%) malicious devices. Moreover, even when the number of malicious devices is higher than 3, our FL-CDC model still achieves better accuracy than the plain FL and approximated FL models.

VI. Conclusion

We have proposed the first integrated FL-CDC model based on LCC to secure the FL process. The model utilizes the least square approximation to approximate the NN computed during FL. The model has been implemented for vehicular applications. Simulation results show that the proposed model achieves a good approximation accuracy and is able to secure the system against malicious edge devices.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Project No. 61950410603; and in part by the Characteristic Innovation Project No. 2021KTSCX110 of Guangdong Provincial Department of Education. The Corresponding Author is Alia Asheralieva.

References

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, Bhagoji *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *arXiv preprint arXiv:2009.13012*, 2020.

[4] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded computing for distributed machine learning in wireless edge network," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[5] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments." in *Osdi*, vol. 8, no. 4, 2008, p. 7.

[6] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "On batch-processing based coded computing for heterogeneous distributed computing systems," *arXiv preprint arXiv:1912.12559*, 2019.

[7] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," *arXiv preprint arXiv:1808.03949*, 2018.

[8] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely-throughput optimal coded computing over cloud networks," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 301–310.

[9] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato *et al.*, "A survey of coded distributed computing," *arXiv preprint arXiv:2008.09048*, 2020.

[10] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1215–1225.

[11] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[12] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[13] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato, C. Leung, and C. Miao, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1800–1837, 2021.

[14] C. Lanczos and T. Teichmann, "Applied analysis," *Physics Today*, vol. 10, no. 6, pp. 44–46, 1957.

[15] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[16] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[17] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[18] S. Ha, J. Zhang, O. Simeone, and J. Kang, "Coded federated computing in wireless networks with straggling devices and imperfect csi," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 2649–2653.

[19] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[20] L. Chen, Z. Charles, D. Papailiopoulos *et al.*, "Draco: Robust distributed training via redundant gradients," *arXiv preprint arXiv:1803.09877*, 2018.

[21] U. Majeed and C. S. Hong, "Flchain: Federated learning via mec-enabled blockchain network," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[22] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[23] W. Zhang, Q. Lu, Q. Yu, Z. Li, Y. Liu, S. K. Lo *et al.*, "Blockchain-based federated learning for device failure detection in industrial iot," *IEEE Internet of Things Journal*, 2020.

[24] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[25] M. Fahim and V. R. Cadambe, "Lagrange coded computing with sparsity constraints," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 284–289.

[26] A. Asheralieva and D. Niyato, "Fast and secure computational offloading with lagrange coded mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4924–4942, 2021.

[27] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 123–142, 2018.

[28] S. H. Jeffreys and B. S. L. Jeffreys), *Methods of mathematical physics / 3rd ed*. Methods of mathematical physics / 3rd ed, 2002.

[29] M. Vlcek, "Chebyshev polynomial approximation for activation sigmoid function," *Neural Network World*, vol. 22, no. 4, pp. 387–393, 2012.

[30] R. Ferreira, C. Affonso, and R. Sassi, "Combination of artificial intelligence techniques for prediction the behavior of urban vehicular traffic in the city of sã£ o paulo," in *10th Brazilian Congress on Computational Intelligence (CBIC)-Fortaleza, CearÃ¡ â€" Brazil (Nov. 2011)*, 2011, pp. 1–7.