Federated Learning using Distributed Messaging with Entitlements for Anonymous Computation and Secure Delivery of Model

Monik Raj Behera [1], sudhir upadhyay [1], Robert Otter [1], and Suresh Shetty [1]

[1]Affiliation not available

October 30, 2023

Abstract

Federated learning has become one of the most recent and widely researched areas of machine learning. Several machine-learning frameworks, such as Tensorflow Federated and PySyft and others have gained momentum in recent past and continue to evolve. Some of the frameworks involve techniques such as differential privacy, secure multi-party computation, gradient descent calculation over the network to achieve privacy of underlying data in federated learning. While these frameworks serve the need for a general-purpose federated learning model as per certain framework, in this paper we present a solution based on distributed messaging with appropriate entitlements that enterprises can leverage in a managed and permissioned network. The solution implements access controls on message source and destination in a decentralized network, which can implement any given data science model in the federated network to facilitate secure federated learning.

Federated Learning using Distributed Messaging with Entitlements for Anonymous Computation and Secure Delivery of Model

Monik Raj Behera
Blockchain Engineering
JPMorgan Chase & Co
Bengaluru, India
monik.r.behera@jpmorgan.com

Sudhir Upadhyay
Blockchain Engineering
JPMorgan Chase & Co
New York, United States of America
sudhir.x.upadhyay@jpmorgan.com

Robert Otter
Blockchain Engineering
JPMorgan Chase & Co
London, United Kingdom
robert.otter@jpmorgan.com

Suresh Shetty
Blockchain Engineering
JPMorgan Chase & Co
New York, United States of America
suresh.shetty@jpmorgan.com

***Abstract*—Federated learning has become one of the most recent and widely researched areas of machine learning. Several machine-learning frameworks, such as Tensorflow Federated [1], PySyft [2] and others have gained momentum in recent past and continue to evolve. Some of these frameworks involve techniques such as differential privacy [3], secure multi-party computation [4], gradient descent [5] calculation over the network to achieve privacy of underlying data in federated learning. While these frameworks serve the need for a general-purpose federated learning model as per certain framework, in this paper we present a solution based on distributed messaging with appropriate entitlements that enterprises can leverage in a managed and permissioned network. The solution implements access controls on message source and destination in a decentralized network, which can implement any given data science model in the federated network to facilitate secure federated learning.**

***Index Terms*—federated learning, machine learning, privacy, distributed, decentralized, blockchain**

I. Introduction

The size of data, growing exponentially year-on-year, has created diversified opportunities to design complex artificial intelligence systems for personal and business usage, at scale. The idea of "big data" and "distributed computing" has together formed the noble field of *distributed machine learning* [6]. In the said method, the data from local devices are generally sent over to powerful and high compute oriented servers for combing through huge data, and deriving the global results. The resultant models are sent back to the individual devices [7]. This has made the low powered edge computers and mobile devices to use machine learning for predictive and classifying abilities, without worrying about the training process. Even though distributed machine learning enables a very efficient mode of computation with heterogeneous devices, it comes with its own challenge of data privacy and security. In the domain of finance, defence, government, health care, enterprise and many more, where data privacy is of utmost importance, distributed machine learning can't be used, where collaboration among the participants is expected, for the improvement of models, but data can't be shared.

With the advancement in computing resources across storage, cloud and mobile devices, machine learning in decentralized network is viable, along with improving the experience in artificial intelligence, both in enterprise and personal lives. Google first published a paper on the concept of "Federated Learning" [8] [9] in the year 2017, a special scenario of "collaborative learning", with multiple decentralized computing devices holding local data samples, without exchanging data for training. This satisfies the need for *data privacy*, which was missing in the distributed machine learning arrangement earlier.

In an operator managed, decentralized, peer-2-peer network, various regulations and compliances mandate participant's data privacy, from both "other participants" and "the network". However, by the same regulations and compliances, the network operator still is required to ensure the normal operation of the network by identifying any malicious activity and actors on the network. Further, the network participants want to learn and benefit from other participants' learning without knowing about their data. This poses a challenge to the network operator and diminishes the value for a participant to be part of such a network.

One of the potential solutions to this is federated (machine) learning, that allows for user data to be private and localized to itself while still allowing the analytics to be done on the individual data, without exposure. There are multiple federated learning frameworks - such as Tensorflow Federated [1], PySyft [2], etc. However, most of these frameworks are still evolving and have been adopted in enterprise networks and systems for their individual use-case, which is tightly

coupled, and needs a lot of re-engineering for accommodating new machine learning models.

Federated learning works with both IID (*independent and identically distributed*) and Non-IID (*non-independent and identically distributed*) data [10]. Based on the data features, federate learning can be broadly classified into two groups. If the data across devices are consistent with the same feature set, it would be horizontal federated learning. And if data's feature set is not the same across all the devices, it would be vertical federated learning. We are proposing a generic framework in this paper, where homogenous models across the device are compatible for network-wide orchestration through a distributed messaging pattern. We are going to discuss both "Pub-Sub" and "Blockchain based private transaction" mechanisms of communication, which will maintain anonymity and secure distribution of models and messages. The mechanism is based upon the entitlements on communication channels and encryption of data in transit.

Federated computation is the crucial phase in the federated learning cycle, where the aggregation takes place. Mathematically, there are multiple ways of aggregating homogenous models across the network. "Federated averaging" [8], "Federated learning algorithm with periodic averaging and quantization" [11], "Dynamic model averaging" [12], data masking [18], noise addition [19] and many more are available in terms of federated aggregation algorithms. In our paper, we are going to propose a method to aggregate homogenous models, independently and asynchronously. An important fact is taken into consideration, the reliability of participants sending models for centralized aggregation. Collaborator needs to be aware of the ad-hoc scenarios and needs to adjust the computation dynamically, as per the network.

II. Distributed Messaging

The core of our system, the distributed messaging [13] component, which serves as the communication and orchestration channel among the participant nodes and collaborator node. On the broader side, the nodes are classified as -

- Collaborator node - *for aggregation role*
- Participant node - *participants in the network*

To facilitate the communication between multiple nodes and collaborator independently, we have implemented event-driven architecture. The events are being listened to by an individual threads/processes on the nodes, making it possible for non-collusive message delivery.

We use Apache Kafka [14], a popular pub-sub system for distributed messaging. Fig 1 depicts the communication topology for the same. Since messaging is the core of the whole service, high availability is achieved using a clustered setup of Kafka. We used 3 nodes setup for Kafka and Zookeeper cluster. "Topics" are the logical channels on Kafka, which separates the message flow. Every topic has been assigned a set of producers and consumers, based on the correct entitlements, which is visible in figure 1. We used a total of four topics, out of which two are used for plaintext messages and two for encrypted messages. We also implemented communication using blockchain, which allows the secure orchestration of models in the network, using encrypted channels.

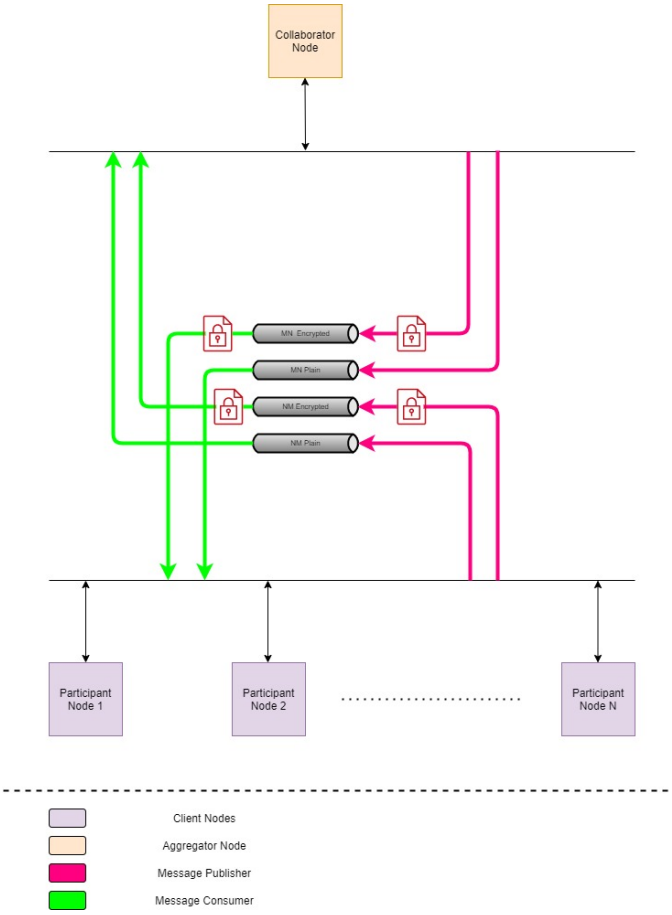

Fig. 1. Communication topology of the federated learning framework

A. Anonymous Computation

Using Kafka admin APIs, we set the role-based access controls (ACLs) on the topics. This essentially defines the which nodes can assume which role on the topics. Message packets don't contain any information about the source of origin in it, hence the *messages are anonymous*. The anonymity of message packets achieves the need for data privacy, as a part of federated learning. No participant node can consume messages from channels, on which participant nodes are producing, thus making it impossible for participants to read any other participating node's data. Security protocols based "Access Controls Lists" allows only valid participants to exchange messages on the messaging system. This takes care of data security. The messages exchanged on the network are encrypted, but still can maintain anonymity, because only one party's RSA key pair (only for collaborator) is static, whereas RSA key pairs generated by participants are rotating for every message. Details of cryptographic implementation have been explained in the subsequent section.

B. Secure Delivery

Along with the distributed message's channel-level security, through Kafka admin API [15] or through permissioned blockchain's channels, *message level dynamic cryptography* provides a secure communication layer with the least key exchange and setup overhead, making participation in the network more fluid in practice.

In "dynamic message encryption-decryption", there are two kinds of nodes involved - *collaborator* and *participant*. For both the sides, encryption and decryption functions are implemented, which is a combination of RSA [16] and AES (CBC mode) [17] algorithms. AES algorithm is used for encrypting the model, and the RSA algorithm is used for encrypting the *key*, being used for symmetric encryption.

In Fig-2, a standard "dynamic message encryption-decryption" life cycle is depicted as DFA state diagram.

Description of the states are as follows -

- S1 : *Algorithm 7* - participant node encrypts the message with collaborator node's public key.
- S2 : *Algorithm 8* - Collaborator node decrypts the message using private key of itself
- S3 : *Algorithm 9* - Collaborator node encrypts the message using participant node's public key
- S4 : *Algorithm 10* - Participant node decrypts the message using the private key generated in the initial state

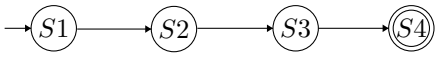

Fig. 2. State diagram *(Life-cycle)* of "dynamic message encryption-decryption"

Based on the cryptographic classification, channels are classified as

- Broadcast Channel - Channels which are being used to broadcast the plain message to all the nodes in the network
- Encrypted Channel - Channels which are being used to broadcast the message with encryption, hence only the valid party can read, even if every consumer receives the message

Based on the data flow direction, channels are classified as

- Collaborator−Participant Channel - Message flows from collaborator to participant nodes only, through entitlement
- Participant−Collaborator Channel - Message flows from participant to collaborator nodes only, through entitlement

Without the "dynamic message encryption-decryption", the total number of *channels* on the network would be defined by equation 1

$$N = 2(P + 1) \tag{1}$$

where N is the total number of channels and P is the total number of participants. The multiplicand 2 signifies that for every logical channel, two sets of physical channels are defined. One for the direction participant node to collaborator node, and other being from collaborator node to participant node. 2 more channels are added which serves as the broadcast channels (both the directions).

Since we are using "dynamic message encryption-decryption", all the messages are being consumed by all the entitled consumers of a given channel. But only the valid consumer, with the right set of RSA private key, can decrypt and read data. With the implementation of secure delivery, the problem of channel maintenance is also decreased to a certain extent. The total number of channels is 4 (2 for broadcast channels and 2 for the encrypted channel). Whenever a new participant is joining the network, or an old participant leaving the network, the only change needs to be performed from an infrastructure point of view is the update in access controlled lists for the channels. With the right set of entitlements, based on roles, the only collaborator can consume from channels, where participant nodes produce. This makes it impossible for any other sibling participant to read any other sibling participant's data.

Algorithm 1 Generate RSA Key Pair

function GENERATERSAKEYPAIR

...

return (K_{pub}, K_{pri}) ▷ Return value computed earlier in function

end function

Algorithm 2 Generate AES Key

function GENERATEAESKEY

...

return K_{AES} ▷ Return value computed earlier in function

end function

Algorithm 3 Encryption

function ENCRYPTION(plainText,key)

...

return $cipherText$ ▷ Return value computed earlier in function

end function

Algorithm 4 Decryption

function DECRYPTION(cipherText,key)

...

return $plainText$ ▷ Return value computed earlier in function

end function

III. FEDERATED LEARNING

In the previous sections, the process of model aggregation orchestration, across the network with multiple *(large)* number of participants is described. The current section describes the federated aggregation of the proposed solution. In the

Algorithm 5 Persisting RSA Key Pairs

function STOREKEY(K_{pub}, K_{pri})
...
return $null$ ▷ Persisting data
end function

Algorithm 6 Removing RSA Key Pairs

function REMOVEKEY(K_{pub}, K_{pri})
...
return $null$ ▷ Removing data
end function

framework proposed, the federation protocol facilitates the partial aggregation in the network [20] [21].

Equation 3 is the objective function, to be minimized on collaborator node. Participants would keep communicating their local models in the given frequency, and collaborator would aggregate the models. Once $g(w)$ converges after multiple aggregation cycles, collaborator node would optimize the resource usage by pausing the global model update, until a better minimization of the $g(w)$ is observed.

$$g(\mathbf{w}) = \sum_{c=1}^{N} p_c G_c(\mathbf{w}), \tag{2}$$

where $0 \leq p \leq 1, \quad \sum_{c=1}^{N} p_c = 1$ and N denotes number of participants.

$$\textbf{Objective: } \min_{\mathbf{w} \in W} g(\mathbf{w}) \tag{3}$$

where W is the set of *weight* matrices in a given model, p is the weight score assigned (statically or dynamically computed)

Algorithm 7 Encryption of message by participant node

Require: $M \in \{collaborator \mid \text{cardinality of set is } 1\}$
Require: $N \in \{participant \mid \text{cardinality of set is } \geq 1\}$
Require: K^M_{pub}; K: Key of RSA pair
Ensure: $p' \neq \emptyset; p'$: plaintext message
$(K^N_{pub}, K^N_{pri}) \leftarrow$ GENERATERSAKEYPAIR
STOREKEY(K^N_{pub}, K^N_{pri})
$K'_{AES} \leftarrow$ GENERATEAESKEY
$c'_{AES} \leftarrow$ ENCRYPTION(p', K'_{AES})
$c'_{RSA} \leftarrow$ ENCRYPTION(K'_{AES}, K^M_{pub})
return $(K^N_{pub}, c'_{AES}, c'_{RSA})$

Algorithm 8 Decryption of message by collaborator node

Require: $M \in \{collaborator \mid \text{cardinality of set is } 1\}$
Require: $N \in \{participant \mid \text{cardinality of set is } \geq 1\}$
Require: K^M_{pri}; K: Key of RSA pair
Ensure: $K^N_{pub}, c'_{AES}, c'_{RSA}$
$K'_{AES} \leftarrow$ DECRYPTION(c'_{RSA}, K^M_{Pri})
$p' \leftarrow$ DECRYPTION(c'_{AES}, K'_{AES})
return (K^N_{pub}, p')

Algorithm 9 Encryption of message by collaborator node

Require: $M \in \{collaborator \mid \text{cardinality of set is } 1\}$
Require: $N \in \{participant \mid \text{cardinality of set is } \geq 1\}$
Require: K^N_{pub}; K: Key of RSA pair
Ensure: $p'' \neq \emptyset; p''$: plaintext message
$K''_{AES} \leftarrow$ GENERATEAESKEY
$c''_{AES} \leftarrow$ ENCRYPTION(p'', K''_{AES})
$c''_{RSA} \leftarrow$ ENCRYPTION(K''_{AES}, K^N_{pub})
return $(K^N_{pub}, c''_{AES}, c''_{RSA})$

Algorithm 10 Decryption of message by participant node

Require: $M \in \{collaborator \mid \text{cardinality of set is } 1\}$
Require: $N \in \{participant \mid \text{cardinality of set is } \geq 1\}$
Require: K^N_{pri}; K: Key of RSA pair
Ensure: $K^N_{pub}, c''_{AES}, c''_{RSA}$
if K^N_{pub} *not present on current system* **then**
...do nothing
return $null$
else
$K''_{AES} \leftarrow$ DECRYPTION(c''_{RSA}, K^N_{Pri})
$p'' \leftarrow$ DECRYPTION(c''_{AES}, K''_{AES})
REMOVEKEY(K^N_{pub}, K^N_{pri})
return p''

to the participant. $G(w)$ is the loss function [22] [23] of the given model, used for federated aggregation. If a subset of participants is unable to send the local model to the federated node, then the weight of that participant would be set to 0.

Weight of the participant plays an important role in federated learning environment [24] [25]. There are various ways to decide weight (priorities) of the participants, depending on the network setup like

- Number of data points in the local data sample
- Logical prioritization based on network and business use case
- Reliability of local data
- Frequency of participant's participation in the aggregation cycle

Federated aggregation, as described in equation 5 gives a more generalized way of aggregation in case of artificial neural network/machine learning models. The framework proposed, supports various federated functions and their respective loss functions, as the aggregation for each machine learning model is independent of each other. The loss function used in the framework, as of now, given in equation 7 and equation 8, are "stochastic gradient descent" [26] and "adam" [27] loss functions respectively. The generic nature and flexibility of the framework supports the inclusion of heterogeneous machine learning models and loss functions, which can be extended, whenever required.

$$W^c = \{\mathbf{w}^c_1, \mathbf{w}^c_2, \ldots, \mathbf{w}^c_K\} \tag{4}$$

$$\mathbf{w}_k^G = \sum_{c=1}^{N} p_c \mathbf{w}_k^c, \quad k \to k^{th} \text{ layer of model} \tag{5}$$

where $0 \leq p \leq 1, \quad \sum_{c=1}^{N} p_c = 1.$

$$W^G = \{\mathbf{w}_1^G, \mathbf{w}_2^G, \ldots, \mathbf{w}_K^G\} \tag{6}$$

where p is the scalar, priority value and $\mathbf{w}$ is the weight matrix, W^c denotes the set of weights of all the layers of a given model in participant c, W_k^G denotes aggregated, global weight of layer k, of the model and W^G is the set of all the aggregated, global weight layers of the given model.

$$\text{SGD:} \quad \mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \sum_{i=1}^{N} \|\hat{\mathbf{y}^i} - \mathbf{y}^i\|^2, \quad t \in \mathbb{R}_{\geq 0} \tag{7}$$

where $\mathbf{w}_t$ denotes weight matrix calculated through *SGD* optimizer for current time cycle. $\mathbf{w}_{t-1}$ denotes weight matrix of previous time cycle, α is the learning rate [26], $\hat{\mathbf{y}^i}$ and $\mathbf{y}^i$ are computed hypothesis and training target vectors respectively.

$$\text{Adam:} \quad \mathbf{w}_t = \mathbf{w}_{t-1} - \frac{\eta}{2} \nabla_{\mathbf{w}} e(\mathbf{w_{t-1}}), \quad t \in \mathbb{R}_{\geq 0} \tag{8}$$

where $\mathbf{w}_t$ denotes weight matrix calculated through *Adam* optimizer for current time cycle, $\mathbf{w}_{t-1}$ denotes weight matrix of the previous time cycle. $\eta \in \mathbb{R}_{\geq 0}$ is the step size and $e(\mathbf{x})$ is the error function [27] [28].

IV. Evaluation

A. Experimental Setup

For validating the proposed solution and it's hypothesis in achieving the acceptable accuracy score in the federated network, we did set up the experimental environment. In the given set-up, we used two kinds of data -

1) Synthetic data generated in a controlled environment, in real-time manner
2) MNIST data [29]

We created a network with 5 participant nodes and 1 collaborator node, installed Apache Kafka with 4 topics, entitlements applied on the users and groups level. The Kafka is run in single-node mode for the simplicity of experimentation. We installed the Quorum permissioned blockchain network for private transactions. Needless to say, both Kafka and Quorum solve the same purpose, provisioning a secure and anonymized channel, hence we used both in separate experiment observations. Each node has RHEL 7.6 OS installed with Python 3.8 installed, along with all the supporting libraries like Kafka-python, Tensorflow 2.0, Web3Py, etc. The services are designed to run as a self-sufficient daemon in the node, which can run the analysis on real-time data. The pipelines are created for data ingestion, pre-processing, deep learning computations and finally visualization through dashboards. Each participant node has 4GB RAM with 2 core processor, whereas collaborator node has 16 GB RAM with 8 core processor. Figure 3 depicts the complete asynchronous network level architecture diagram of the setup, which accommodates the generalized (*large number of*) participant nodes setup. The architecture can be logically thought of a star topology [30], which is depicted in figure 4.

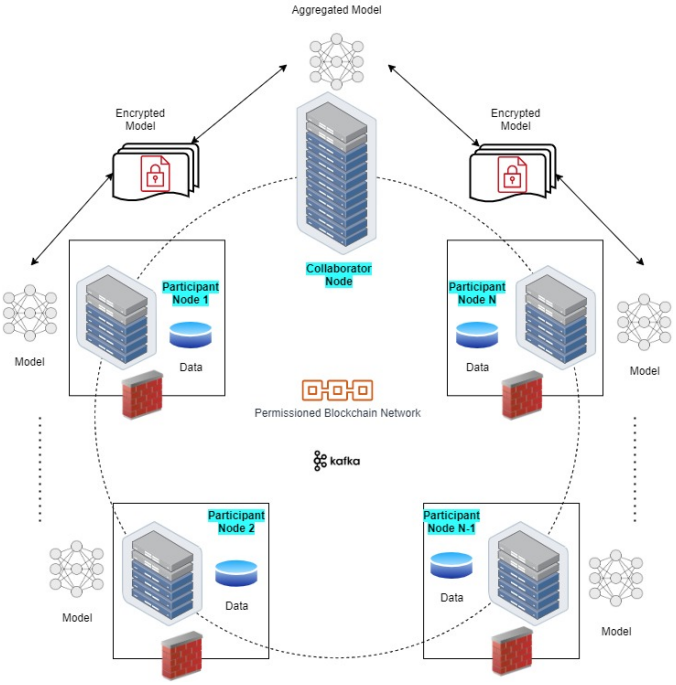

Fig. 3. Physical infrastructure architecture diagram with multiple nodes and collaborator node setup

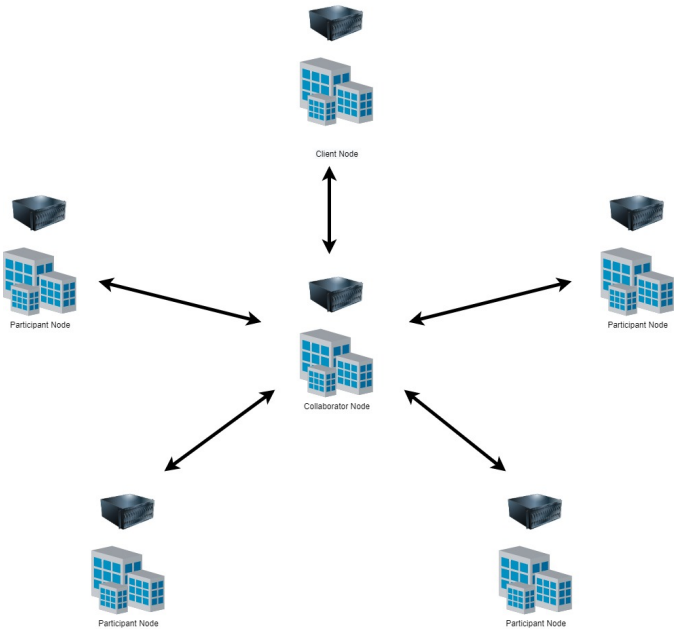

Fig. 4. Logical communication network topology of the setup

B. Federated Learning on Synthetic Data

For testing framework's performance on various real-time data, we came up with a synthetic data generator, a simple python script to generate numeric data with given dimension and sample size. The synthetic data generated follows the *gaussian distribution* [31]. The data is continuously generated with a frequency of $1\ min$ on each node. Since the data has time component in it, we treated it as time series analysis and implemented unsupervised anomaly detection using "Long Short-term Memory" deep neural network [32]. Anomaly detection's scores were then fed into the machine learning pipeline for supervised binary classification between *true positives* and *false positives* values. For binary classification, we used "Logistic regression based classification" technique [33].

Deep learning based anomaly detection is run every $5\ minutes$ on the real-time data generated, and the result, of numeric data type, is passed on to the next stage in the pipeline. The anomaly detection model is *not* included in the federated learning, because it is of unsupervised nature. The model architecture for LSTM network we used is described in figure 5. The anomaly scores are calculated based on equation 9.

Logistic regression model is the one which participates in federated learning. A participant sends it's local model when the request is made by the collaborator. Once the aggregation, based on equation 5 is completed, the new updated global model is requested by the individual participant. Communication is secured through the set of RSA keys and entitlements based on encrypted channels.

$$\text{Loss MSE} = \frac{1}{n}\sum_{i=1}^{n} \|\hat{\mathbf{y}}^i - \mathbf{y}^i\|^2 \tag{9}$$

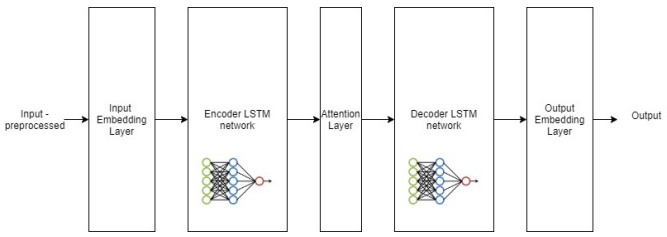

Fig. 5. LSTM Network used for anomaly detection in stage 1 of time series analysis pipeline for synthetic data

For visual analysis of the data and anomalies generated, figure 6 depicts one of the features, which plays the primary roles in synthetic anomaly injection. Figure 7 shows the anomalies generated by secondary features in the data. In figure 8, we can see the anomalies generated and *true vs false positives* decision prediction, which is computed through logistic regression model, using federated learning framework. Figure 9 depicts a common representation of 3 nodes' anomaly loss and logistic classifier's decision (0 *being false-positive* and 1 *being true-positive*)

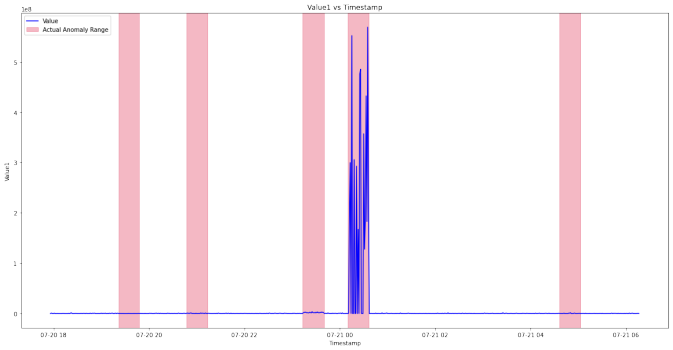

Fig. 6. The "magenta" colour regions in the graph are time regions where anomalies are synthetically generated. Spike in line graph shows the anomalies generated for "feature 1", depicted as "value1"

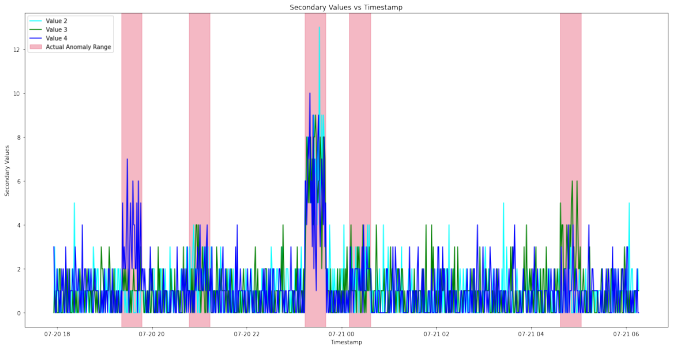

Fig. 7. The "magenta" colour regions in the graph are time regions where anomalies are synthetically generated. Spike in line graph shows the anomalies generated for "feature 2", "feature 3", and "feature 4", depicted as "value2", "value3" and "value4"

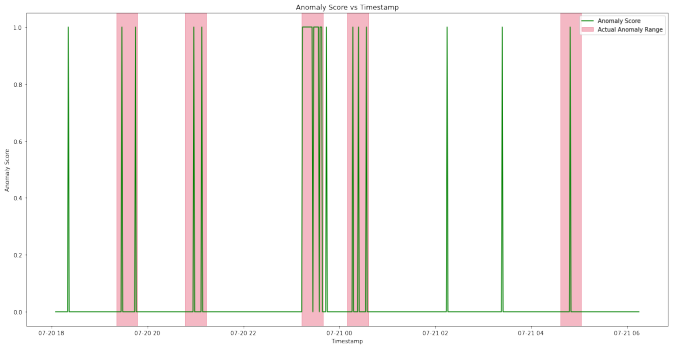

Fig. 8. The "magenta" colour regions in the graph are time regions where anomalies are synthetically generated. Green peaks show the decision of anomalies detected being true positive, through federated learning (logistic regression classifier)

C. Federated learning on MNIST data

In this section, we are describing the methodology of verifying our proposition on standard MNIST data [29]. We have used the same setup, described earlier, added artificial neural

Fig. 9. Network level visualization of multiple nodes' anomaly detection's decision. First row shows the anomaly scores per node. Second row shows anomaly decision, which is the result of logistic regression model through federated learning

network machine learning model in the federated framework. The neural network being used has one input layer, with 128 input neurons, one hidden layer of 2096 neurons and one output layer with 10 neurons. Figure 10 depicts the model graphically.

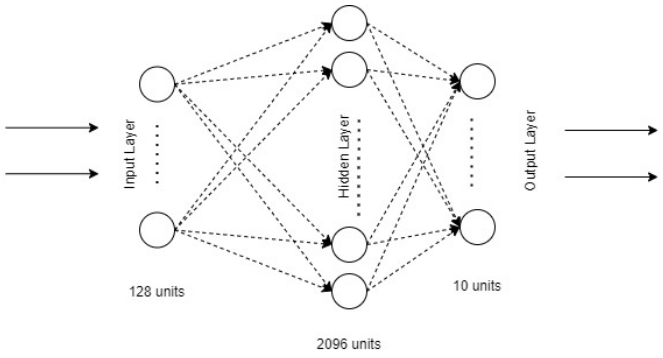

Fig. 10. Architecture diagram of neural network model for hand-written digit recognition used on MNIST data

During experimentation, we studied loss' dependence on learning rate of adam's optimizer, by changing the value of learning rate, as depicted in figure 11. We also studied accuracy and it's dependence on learning rate, as shown in figure 12. Table I represents loss and accuracy of all models in the federated setup. Table infers that using federated computation, we achieved better accuracy over time.

TABLE I
LOSS AND ACCURACY OF LOCAL AND GLOBAL MODEL OVER NETWORK

Node	*Loss*	*Accuracy (in %)*
Global Model - Initial	**2.9559**	**91.12**
Participant 1	1.8994	94.83
Participant 2	1.6606	94.58
Participant 3	1.8350	94.91
Participant 4	1.8994	94.52
Participant 5	1.6724	94.82
Global Model - Aggregated	**0.7036**	**95.39**

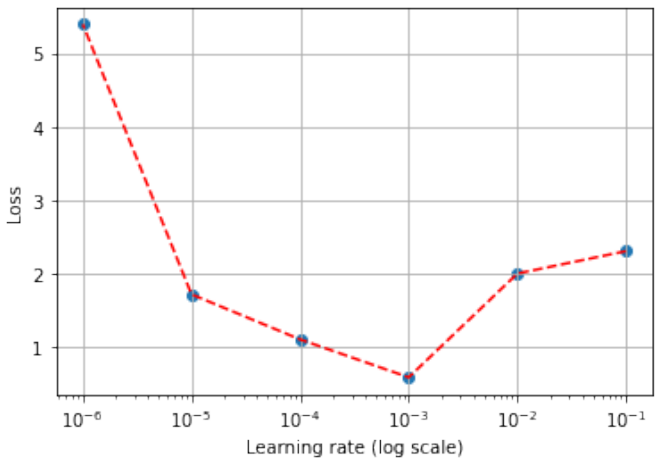

Fig. 11. Loss vs learning rate of the neural network's Adam optimizer

V. CONCLUSION

In this paper, we have presented the current common practices around federated learning in academia and enterprises. To overcome the generic challenge of secure and anonymized orchestration of models in the federated network, we are proposing an elegant and new method of realizing federated learning, independent of machine learning framework using entitled, anonymized and secure channels. We have emphasized the medium of communication would be a distributed messaging system, either like Kafka, NATS.io, pub-sub or like a permissioned blockhain/ledger like Quorum [34]. The system described in the given paper is flexible because of no dependency on any specific framework. It is highly available in nature because of distributed messaging through "highly available cluster" [35] or through "blockchain network"

A. Applications

In terms of application, the proposed federated framework is most applicable in an operator managed private, permis-

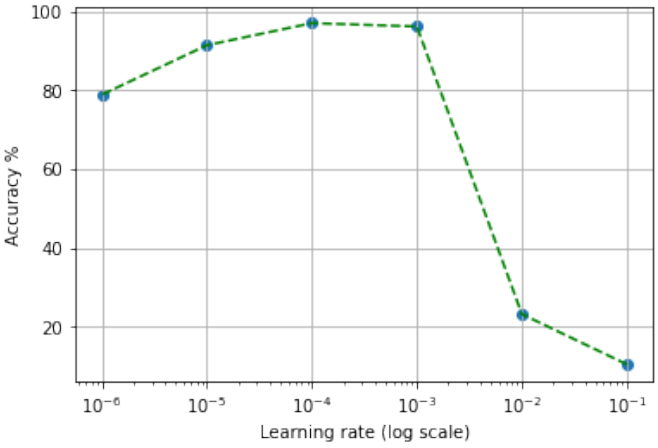

Fig. 12. Accuracy (in percentage) vs learning rate of the neural network's Adam optimizer

sioned, consortium decentralized networks. On a permissioned blockchain, we can categorize most of the activities in two broad categories:

1) Transactional data between two participants. Examples of this may include data related to trades, personal information, token, money transfer etc. The data could be classified as highly confidential and sensitive.
2) Network related meta-data such as count of a specific categories of transactions, frequency of transactions, throughput & latency of such transactions.

Transactional data between two participants can be used to train machine learning models that can learn from such data to identify anomalies or specific patterns in the transactions. Such models, when fully trained, can make predictions of future transactional behaviour on the network.

To further illustrate anomalies, two participants have learnt that they have observed a large number of transactions for a specific currency in the last N hours or days that can be attributed to a specific geo-political event. Interestingly, other pairs of participants have observed similar behaviour/patterns across other events. However, because of privacy and compliance concerns, other than only those participants that were part of those such transactions have visibility into such patterns. So, how do other participants on the network benefit from such a data pattern on the network? The proposed solution can be implemented in this case where the network operator can ensure a secured (perhaps by operational procedures) messaging infrastructure (Kafka, blockchain or similar) and allow network participants to communicate with that messaging infrastructure along with proposed techniques for sharing the model. Further, the same infrastructure can be leveraged for unsupervised anomaly detection on the network.

Some other applications on similar networks can include improving network efficiency by leveraging secured computation of network-wide aggregate metrics, providing transparency in the network without compromising the privacy of underlying data. As the number of enterprise blockchains and decentralized networks continue to evolve, along with increasing the volume of transactions on such networks. Further, it is evident that the more successful networks would provide its participants the benefit of the network effect and learning, without requiring the members of the network to share proprietary and competitive data to other members. It is also observed that most enterprises technology teams are quite familiar with public key infrastructure along with messaging systems, such as Kafka. A federated learning solution based on this technology stack can increase its adoption and usability.

B. Challenges and future scope

The proposed solution opens up scope for stringent safeguarding the security of encryption/decryption keys. Any compromise of the private keys of the aggregator by a malicious operator can potentially leak some data attributes, which in result can compromise data privacy in general over the network. The possible ways to safeguard the keys would be to use key management solutions [36] [37] like a vault.

In the current model, there is only one and predefined collaborator, which is decided by the network operator. In future, the choice of collaborator can be based on a consensus algorithm among all of the participants. One potential algorithm could be RAFT based [38] [39], where the collaborator could be elected by the rest of the participants in the network for a certain duration. In such cases, it is possible to extend the solution to a true peer-2-peer decentralized network where there is no network operator responsible for the governance of the messaging infrastructure, which brings federated learning in more fluid and ad-hoc infrastructure.

ACKNOWLEDGMENT

We would like to thank our colleagues Robert Otter, Suresh Shetty, Brett J Sanford, Sean J Moran, Tulasi D Mova, Jacob Mendel, Antonios Georgiadis, Razi Abbas, Debidutta P Samantaray, Fran Silavong and Sanket Kamthe for reviews and discussions during the experimentation and evaluation. We would also like to thank JP Morgan Chase & Co for supporting us in carrying out the experiments for research.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Ryffel, Theo, et al. "A generic framework for privacy preserving deep learning." arXiv preprint arXiv:1811.04017 (2018).

[3] Sarwate, Anand D., and Kamalika Chaudhuri. "Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data." IEEE signal processing magazine 30.5 (2013): 86-94.

[4] Lindell, Yehida. "Secure multiparty computation for privacy preserving data mining." Encyclopedia of Data Warehousing and Mining. IGI Global, 2005. 1005-1009.

[5] Wang, Shiqiang, et al. "Adaptive federated learning in resource constrained edge computing systems." IEEE Journal on Selected Areas in Communications 37.6 (2019): 1205-1221.

[6] Kraska, Tim, et al. "MLbase: A Distributed Machine-learning System." Cidr. Vol. 1. 2013.

[7] Li, Mu, et al. "Scaling distributed machine learning with the parameter server." 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). 2014.

[8] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial Intelligence and Statistics. PMLR, 2017.

[9] Konečný, Jakub, et al. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).

[10] Zhao, Yue, et al. "Federated learning with non-iid data." arXiv preprint arXiv:1806.00582 (2018).

[11] Reisizadeh, Amirhossein, et al. "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization." International Conference on Artificial Intelligence and Statistics. 2020.

[12] Kamp, Michael, et al. "Efficient decentralized deep learning by dynamic model averaging." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2018.

[13] Irribarren, Roberto. "Distributed messaging system." U.S. Patent No. 5,841,966. 24 Nov. 1998.

[14] Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.

[15] Gupta, Maanak, Farhan Patwa, and Ravi Sandhu. “An attribute-based access control model for secure big data processing in Hadoop ecosystem.” Proceedings of the Third ACM Workshop on Attribute-Based Access Control. 2018.

[16] Bleichenbacher, Daniel. “Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1.” Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1998.

[17] Frankel, Sheila, R. Glenn, and S. Kelly. “The AES-CBC cipher algorithm and its use with IPsec.” (2003).

[18] Liu, Yang, et al. “Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing.” arXiv preprint arXiv:1907.10218 (2019).

[19] Mivule, Kato. “Utilizing noise addition for data privacy, an overview.” arXiv preprint arXiv:1309.3958 (2013).

[20] Liu, Lumin, et al. “Client-edge-cloud hierarchical federated learning.” ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020.

[21] Xu, Zirui, et al. “Elfish: Resource-aware federated learning on heterogeneous edge devices.” arXiv preprint arXiv:1912.01684 (2019).

[22] Bottou, Léon. “Large-scale machine learning with stochastic gradient descent.” Proceedings of COMPSTAT’2010. Physica-Verlag HD, 2010. 177-186.

[23] Culotta, Aron, et al. “Author disambiguation using error-driven machine learning with a ranking loss function.” Sixth International Workshop on Information Integration on the Web (IIWeb-07), Vancouver, Canada. 2007.

[24] Yang, Kai, et al. “Federated learning via over-the-air computation.” IEEE Transactions on Wireless Communications 19.3 (2020): 2022-2035.

[25] Nishio, Takayuki, and Ryo Yonetani. “Client selection for federated learning with heterogeneous resources in mobile edge.” ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019.

[26] Bottou, Léon. “Large-scale machine learning with stochastic gradient descent.” Proceedings of COMPSTAT’2010. Physica-Verlag HD, 2010. 177-186.

[27] Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. “On the convergence of adam and beyond.” arXiv preprint arXiv:1904.09237 (2019).

[28] Bock, Sebastian, Josef Goppold, and Martin Weiß. “An improvement of the convergence proof of the ADAM-Optimizer.” arXiv preprint arXiv:1804.10587 (2018).

[29] LeCun, Yann and Cortes, Corinna. “MNIST handwritten digit database.” (2010): .

[30] Barranco, Manuel, et al. ”An active star topology for improving fault confinement in CAN networks.” IEEE transactions on industrial informatics 2.2 (2006): 78-85.

[31] Aguiar, Ricardo, and M. T. A. G. Collares-Pereira. “TAG: a time-dependent, autoregressive, Gaussian model for generating synthetic hourly radiation.” Solar energy 49.3 (1992): 167-174.

[32] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov. “Unsupervised learning of video representations using lstms.” International conference on machine learning. 2015.

[33] Dreiseitl, Stephan, and Lucila Ohno-Machado. “Logistic regression and artificial neural network classification models: a methodology review.” Journal of biomedical informatics 35.5-6 (2002): 352-359.

[34] García-Pérez, Álvaro, and Alexey Gotsman. “Federated byzantine quorum systems.” 22nd International Conference on Principles of Distributed Systems (OPODIS 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[35] Murphy, Declan J., et al. “Method and apparatus for transparent server failover for highly available objects.” U.S. Patent No. 6,185,695. 6 Feb. 2001.

[36] Lerner, Daniel M., et al. “Encryption key management system and method.” U.S. Patent No. 6,157,722. 5 Dec. 2000.

[37] Buer, Mark L., and Joseph J. Tardo. “Key management system and method.” U.S. Patent No. 7,773,754. 10 Aug. 2010.

[38] https://raft.github.io/

[39] Ongaro, Diego, and John Ousterhout. “In search of an understandable consensus algorithm.” 2014 USENIX Annual Technical Conference (USENIXATC 14). 2014.