Blockchain-Enabled Secure Big Data Analytics for Internet of Things Smart Applications

Prakash Tekchandani, Indranil Pradhan, Ashok Kumar Das, *Senior Member, IEEE*, Neeraj Kumar, *Senior Member, IEEE*, and Youngho Park, *Member, IEEE*

***Abstract*—Smart devices in an Internet of Things (IoT) generate a massive amount of big data through sensors. The data is used to build intelligent applications through machine learning (ML). To build these applications, the data is collected from devices into data centers for training ML models. Usually, the training of models is performed on central server, but this approach requires the transfer of data from devices to central server. This centralized training approach is not efficient because the users are much less likely to share data to the centralized data centers due to privacy issues and bandwidth limitations. To mitigate these issues, we propose an efficient hybrid secure federated learning approach with the blockchain to securely train the model locally on devices and then to store the model and its parameters into the blockchain for traceability and immutability. A detailed security and performance analysis is presented to show the efficacy of the proposed approach in terms of security, resilience against many security attacks, and cost effectiveness in computation and communication as compared to other existing competing schemes.**

***Index Terms*—Authentication, big data analytics, blockchain, Internet of Things (IoT), key agreement, security.**

I. Introduction

THE RAPIDLY increasing volume and complexity of big data have led to a significant growth of data [1]. With the fast evolution of technology, enormous amount of data is produced from Internet of Things (IoT) smart devices, healthcare, manufacturing, banking, and other applications. In big data, we extract knowledge and value from various forms of data. This field encompasses statistics, data analysis, and other advanced methods used to understand and analyze actual processes using data. Organizations are embracing the power of big data, using powerful analytics for driving strategic business decisions, identify opportunities, and speed up performance. At the same time, big data security concerns rise with the vast increase in data usage and utilization. Finally, the big data adoption comes down to one question for many enterprises: "How can we use big data potential while effectively reducing security risks?."

Big data is primarily dependent on infrastructure, cloud, and edge computing which impose data security risks [2], [3], [4], [5]. Untrusted applications, which lacks security measure defined by standard authority and data governance, can easily impose risks into enterprise. Not only this, increase usage of mobile and IoT devices to work from anywhere at any time utilizing the cloud network adds to the threat of big data security. Additionally, depth of the user's access to sensitive data is another concern in terms of security. The top preference for any enterprise must include defining the secured privilege access for users. Well-defined security protocols and measures must be defined to avoid potential misuse of data keeping in mind that some users have access to highly sensitive data in some business processes.

As the big data keeps expanding, the risk of data sharing and data source authentication is another point of discussion. Parties involving data sharing do not completely trust on each other during the data transaction. For example, government has a fear of violation in national security and is more cautious about data sharing as well as individual face privacy concern about their personal information. Furthermore, multistakeholder involvement has resulted in poor control over data transfer which creates the hinderance in the development of the big data industry. Big data collected from IoT devices and other applications is stored in data centers for training machine learning (ML) models to build intelligent applications. Centralized training is not efficient because users are much less likely to share data to centralized data centers due to privacy issues and bandwidth limitations.

We propose a novel hybrid secured federated learning approach with blockchain to securely train the model locally on devices and store the model and its parameters in blockchain for traceability and immutability. In federated learning model training, IoT devices learn collectively from a shared model. The server trains the shared model with proxy data and send model to IoT devices that improves the model through local or federated data training. The updated trained

Manuscript received 17 April 2022; revised 29 August 2022; accepted 2 December 2022. Date of publication 6 December 2022; date of current version 24 March 2023. This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2020R1I1A3058605, and in part by the BK21 FOUR Project funded by the Ministry of Education, South Korea, under Grant 4199990113966. *(Corresponding authors: Neeraj Kumar; Youngho Park.)*

Prakash Tekchandani, Indranil Pradhan, and Ashok Kumar Das are with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500032, India (e-mail: prakash.tekchandani@research.iiit.ac.in; indranil.pradhan@yahoo.in; iitkgp.akdas@gmail.com).

Neeraj Kumar is with the Department of Computer Science and Engineering, Thapar University, Patiala 147004, India, also with the School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India, also with the Department of Electrical and Computer Engineering, Lebanese American University, Beirut 1102 2801, Lebanon, and also with the Faculty of Computing and IT, King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: neeraj.kumar@thapar.edu; nehra04@gmail.com; neeraj.kumar@lau.edu.lb).

Youngho Park is with the School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea (e-mail: parkyh@knu.ac.kr).

Digital Object Identifier 10.1109/JIOT.2022.3227162

model is summarized before sending back to server. Now the improved model updates are ready for uploads to server. To make federated learning secured, we aim to design an approach to authenticate device and server before any uploads starts. After authentication between device and server the model is sent to the server, the server further aggregates the model to obtain an improved combined model. The exchange of models between device and server is continued for several iterations until a superior model is obtained. In a federated learning environment the model is passed to training data, instead of training data to model. By this the device has full control on data and achieves data privacy. The improved model and its parameters are stored in blockchain for traceability and immutability.

A. Motivation

Data is everywhere and sensor-equipped IoT devices are a major source of data. Mobiles devices connected with Internet generates data on every click and intelligent applications are required for enhanced user experience. Till 2025 there will be 24.6 billion IoT devices, this means data is available everywhere. The data generated by these applications needs to be analyzed through artificial intelligence (AI) and ML. Usually the training happens on a central server having large computational capability, but this breaches data privacy of user. To avoid this federated learning is used where training happens at local devices and just model is shared, but in this also privacy can be compromised [6]. A primary motivation is to design secured federated learning which ensures data privacy by mutual authentication between server and device. Also, to build an efficient customized ML model without compromising privacy of user's data and bandwidth limitation for transferring private data to the server. In proposed design upon successful mutual authentication, a session key is generated for secured communication between server and device. The proposed design is analyzed on security, storage and computational parameters, as well as using formal specifications (in our context, a widely accepted automated formal security validation tool, called Scyther tool) to assert that the proposed design is secured against various well-known attacks. Our proposed ML model is analyzed under various noise insertion attacks, and performance analysis is presented to assert that our proposed model is efficient.

B. Research Contributions

The main contributions of the proposed work are listed in the following.

1) We propose a novel secured federated learning protocol, based on the network model provided in Fig. 2. In this protocol, first enrollment between server and device takes place in offline mode and then device login is permitted. The authentication and key agreement phase allows server and device to have a common session key for secured communication.
2) Next, we provide a detailed mechanism for a new block creation and addition in the blockchain through the practical Byzantine fault tolerance (PBFT)-based consensus mechanism [7].
3) After this, we provide the detailed experimental setup and analysis about the ML model used in our scheme. We also discussed about performance of our model with and without noise insertion attacks. We also provided experimental results on performance of our model with and without attacks.
4) Next, we apply the threat model given in Section IV-B to provide a rigorous security analysis and performed storage and computational analysis of our scheme. We also performed formal security verification using the widely used Scyther simulation tool.
5) We also provided the details of blockchain implementation with PBFT.
6) Finally, a detailed comparative study among other relevant protocols is presented.

C. Paper Outline

The remainder of this article is outlined as follows. Section II discusses the related work in the big data security and federated learning. The preliminaries required to understand important terminologies are described in Section III. The network and threat models are described in Section IV. Section V discusses the proposed security framework for big data analytics. In Section VI experimental results of ML model are presented. Section VII provides rigorous security, storage, and computational analysis. Section VIII provides formal security verification using the widely used Scyther automated software tool through simulation. The blockchain implementation of the proposed scheme is demonstrated in Section IX. Section X gives a detailed comparative analysis among other related schemes. The concluding remarks are provided in Section XI.

II. Related Work

Uchibeke et al. [8] proposed big data security through blockchain access control. In this work, they focused on big data security and privacy issues in the cloud computing environment. They proposed a blockchain role-based access control to address security in big data. In their study, they modeled the blockchain using the Hyperledger composer. In the role-based access control, access is granted on a role-by-role basis. They created smart contracts to access, grant, and verify users based on the roles assigned.

Yue et al. [9] proposed safe circulation of big data resources through blockchain. In their proposed work, there are three entities as data producers, users, and blockchain network. Smart contract keeps track of a large amount of data generated by data producers and store the data in blockchain. Smart contract is also used by users to obtain authorized data from blockchain.

Espostio et al. [10] proposed the blockchain technology for healthcare data in cloud. New block is created as soon as the patient data is generated. This block is distributed among all peers patient network. After the consensus and Proof of Work (PoW) the block is inserted in blockchain. This enables global view of patient history. This view is immutable and traceable, and cryptographically linked for security. If the consensus among peers is not reached, then the block is declared orphan.

Chang et al. [11] proposed an incentive-based blockchain for edge computing. They used the Stackelberg equilibrium to find optimal incentive to miners to participate in process of blockchain and obtain computational resources from the edge service provider.

Jangirala et al. [12] proposed a lightweight blockchain-enabled communication and computation protocol (LBRAPS) for authentication. This protocol is radio frequency identification (RFID)-based and used for manufacturing supply chain in the 5G mobile edge computing (MEC) environment. LBRAPS is included with authentication and key agreement. This protocol is efficient in computation and supports security features which are efficient against active attacks.

Yang et al. [13] proposed distributed blockchain approach for privacy of network topology for MEC in 5G. They implemented mutual networking trust, heterogeneous MEC, and collaborative routing through blockchain. Their scheme improved the efficiency of MEC in 5G.

Roy et al. [14] proposed a secure and lightweight mobile user authentication scheme for mobile cloud computing. It is based on cryptographic hash, XOR, and fuzzy extractor methods. Challa et al. [15] designed an authentication scheme for cloud-assisted Cyber–Physical System. In this a secured communication between smart meter and a cloud server is established through session key-based mutual authentication.

Lu et al. [16] proposed privacy during data sharing through federated learning and blockchain. They mainly focused on minimizing the consensus work in blockchain through federated learning. The data model is shared instead of actual data to achieve privacy of data.

Chai et al. [17] proposed hierarchical blockchain-enabled federated learning for Internet of Vehicles (IoV). They also used game theory for distributed knowledge sharing. They aggregate the knowledge in hierarchical layers and perform correlation. Their approach improved the security of knowledge sharing in IoV.

Kim and Hong [18] focused on the stability and convergence speed of federated learning by minimizing network load, and proposed a blockchain-based method to address these challenges.

III. Preliminaries

A. Artificial Intelligence and Machine Learning

AI [19] is founded around the idea that human intelligence should be described in such a way that a computer can easily imitate it and carry out tasks ranging from the most basic to the most complex [20]. AI's aims include simulating human cognitive function. To the degree that these can be concretely established, researchers, and developers in the area are making remarkably fast progress in simulating tasks, such as understanding, thinking, and perception. Some claim that in the near future, innovators will be able to create robots that can understand and think on any topic faster than people [21].

ML [22] is the study of computer algorithms that refine themselves over time as a result of experience and data [23]. It is considered to be a component of AI. ML algorithms create a model based on sample data, referred as training data, in order to make projections or decisions without being directly programmed [22]. ML algorithms are used in a broad range of applications, including medicine, email scanning, computer vision, pattern recognition, decision making, and observations of a dynamic environment without explicit instructions [24], [25] where developing traditional algorithms to perform the required tasks is challenging or impossible. Based on the essence of the signal or feedback available to the learning system, ML methods are historically classified into three different categories [26], [27].

1) Supervised Learning: In this machine is given with example inputs and ideal outputs, with the aim of learning a general law that maps inputs to outputs [28].

2) Unsupervised Learning: Unsupervised learning is where the learning algorithm is not assigned labels and is left to find order in its input on its own [29].

3) Reinforcement Learning: A computer program interfaces with a complex world in order to accomplish a specific task (such as driving a vehicle or playing a game against an opponent) [30]. The software is given input in the form of incentives as it navigates its problem space, which it seeks to optimize.

B. Attacks Related to AI/ML

AI/ML, like most networking environments, is vulnerable to a variety of threats. A variety of security attacks on AI/ML and that are based on the adversarial capabilities, knowledge, goals, resources, and the target criteria exists [31], [32], [33]. At a high level, AI/ML attacks can be divided into three categories.

1) Adversarial Inputs: In this carefully designed inputs created with the aim of being reliably misinterpreted in order to avoid detection [34]. Malicious documents intended to evade antivirus and emails trying to evade spam filters are examples of adversarial inputs.

2) Data Poisoning Attack: In this the model trains adversarial data. The most popular form of attack we see is model skewing, in which the attacker tries to contaminate training [35] data so that the line between what the model considers positive data and what the classifier considers poor data swings in his favor.

3) Model Stealing: In this attack, model stealing techniques can be used to steal (i.e., duplicate) templates or restore training data membership by blackbox probing. This attack allows the attackers to gain insight about the training data and enables cloning the model that is close to the actual model [36]. Examples can be to steal stock market forecast models and spam filtering models.

C. Genetic Algorithm Neural Network

Genetic algorithms (GAs) are adaptive heuristic search algorithms that fall under the evolutionary algorithms umbrella [37].

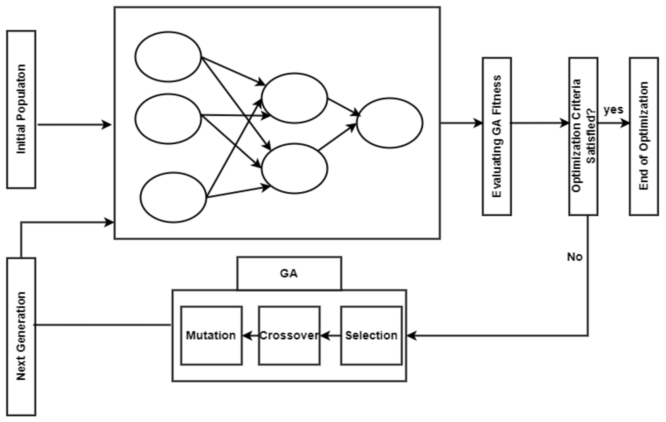


Fig. 1. GANN.

Natural selection and genetics are the foundations of GAs. These are intelligent applications of random search aided by historical data to guide the search to a solution space area with better results. GAs are based on a comparison to the genetic arrangement and action of a population's chromosome. The basis of GAs is as follows.

1) Individuals fight for wealth and mates in a population.
2) Those that are the most successful (fittest) marry in order to produce more offspring than others.
3) The genes of the fittest parent are passed on over the generations, which means that often parents produce children who are superior than either parent.
4) As a result, each subsequent generation is well matched to their surroundings.

The steps of the GA (Fig. 1) can be summarized as follows [38].

1) Initialize populations at random p.
2) Determine population fitness function.
3) Repeat before convergence.
    a) Choose parents from the population.
    b) Crossover to create a new population.
    c) Mutate the new population.
    d) Calculate the health of the new population.

In the neural networks (NNs), hyperparameter tuning is a major problem [39], and GAs can be used to solve this problem [40] as follows.

1) Make a community of several NNs.
2) Assign all the NNs random (in a range) hyper-parameters.
3) For a certain number of iterations, do the following.
    a) Train all the NNs at the same time or one at a time.
    b) Calculate their preparation costs once they have all completed their training.
    c) Calculate each NN's fitness (how well it performed in that iteration) based on its expense.
    d) Determine the population's level of health [required for step e)], can be disregarded depending on the implementation of step e).
    e) Choose two NNs based on their fitness function as determined by a probability system.
    f) Genes from the two NNs are crossed. This will result in an NN that is referred to as a child. Any properties of the first NN and some of the second NN should be present in this NN.
    g) The child's genes will be mutated.
    h) Steps e and g can be repeated with the total number of NNs in the population.

D. Blockchain

The blockchain is a distributed and decentralized ledger that stores data, such as transactions, and that is publicly shared across all the nodes of its network. Blockchains have the following characteristics.

1) Decentralization: A blockchain is distributed peer-to-peer network of nodes, each node stores a whole or part of blockchain [41]. This eliminates the single point of failure and remove dependency on central trusted authority [42]. This quality strengthens the entire system as decentralized information is more difficult to hack than a centralized one sitting in a unique location.

2) Cryptographic Hashing: Every block of blockchain is connected to the previous block through cyptographic hashed connection [43], [44]. This ensures that the content of chain cannot be altered as a single change would affect all subsequent hash values and chain would get invalid [45].

3) Timestamping: Each block in blockchain is timestamped chronologically to provide transaction history, traceability, and transparency. To collect proof of certain information at a particular time, timestamp in combination with hash can also be used [43].

4) Consensus Mechanism: This defines how transactions are validated in distributed environment and added in blockchain. PoW, Proof of Stake (PoS), and PBFT are some of the consensus mechanism [41], [45], [46].

5) Immutability: Cryptographic hashing and consensus mechanism ensure that once data is stored in blockchain it cannot be modified or deleted [47].

IV. System Models

A. Network Model

We used secured federated learning and blockchain to train the model, data privacy, and storing the model and its parameters in blockchain. In federated learning model training, IoT devices learn collectively from a shared model. The server trains the shared model with proxy data and send model to IoT devices. Further, devices improve the model through local or federated data training. The updated trained model is at device end and sent to server. Now the server aggregates the received model with the model present in server and test on test data set. If the accuracy on test data set reaches the threshold value set by server, then the aggregated model is set to be kept in blockchain. To make secured model using federated learning, we proposed a novel approach to authenticate device and server before any uploads starts. After authentication between device and server the model is sent to the server. The server aggregates the model to obtain an improved combined model. The exchange of models between device and server is continued for several iterations until a superior model is obtained. In federated learning the model is passed to training data, instead of training data to model. By this the device has full control on data and achieves data privacy. The improved model and is parameters are stored in blockchain for traceability and immutability. Below are the steps followed in our proposed approach and pictorially described in Fig. 2.

Step 1: Model with random weights or model trained with small data set, is distributed to all the devices, i.e., from central server to A, B, and C as shown in Fig. 2.

Step 2: Each device (A, B, C) has its own data set, and the model received from central server. Each device now train model locally on respective end. Authentication is established between participant and server for secured federated learning.

Step 3: During training of the model on the individual participant data set, the model weights are updated and the updated model weights are then sent to the central server.

Step 4: The central servers collect the model from each device and aggregates the weights using the averaging method

$$\text{AMP} = \frac{\text{CSMP} + \text{PD}}{2}$$

where AMP = Aggregated Model Parameters, CSMP = Central Server Model Parameters, and PD = Parameters from Devices.

Step 5: Steps 1–4 are repeated until the model is trained.

Step 6: After the model is trained, block containing transactions is added to blockchain using PBFT.

B. Threat Model

The following threats have been considered in the proposed scheme.

1) Device can be physically captured in the network, which could lead loss of sensitive information [48].
2) As all the communications except the offline registration phase take place in public channel, any adversary can eavesdrop the communications between server and the devices and intercept the messages [48].
3) The ID and the password are of low entropy. The attacker may guess the ID and password [49].
4) The attacker may delete, manipulate the eavesdropped messages [49].
5) The attacker may be a legitimate user [49].
6) Adversary may initiate data poisoning attack by inserting noise in the data set or flipping the target label and reduce the model performance [50].

V. Proposed Security Framework for Big Data Analytics

For a better understanding of this article, we list the notations along with their significance in Table I. Next, we describe the detailed description of each phase below.

A. Enrollment and Login Phase

The proposed enrollment and login phase is summarized in Fig. 3. The step-wise discussion of this is elaborated as follows.

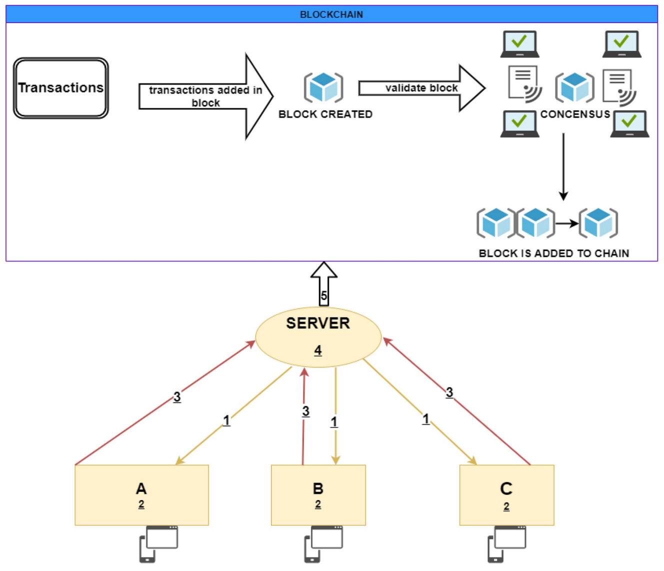


Fig. 2. Network model.

TABLE I
Notations and Their Significance

Notation	Significance
ID_c, ID_i	Identity of an entity
TC	128 bit temporary identity
n	128 bit secret generated by server
r	128 bit secret generated by client
RID, RID', RPW	Variables
RPW', BI, TID, TID'	Variables
IDC, IDC', IDC''	Variables
$H_{IDC}, H_{IDC'}, ACK, ACK'$	Variables
PW_i	Device password
K_{DC}	Shared key between device and server
T_1, T_2, T_3, T_4	Timestamps
R_C, R'_C, R_N, R'_N	Nonces
$h(\cdot)$	"collision-resistant cryptographic one-way hash"
M_1, M_2, M_3	Messages
$LR(X, Y)$	Left rotate X by Hamming distance Y
$RR(X, Y)$	Right rotate X by Hamming distance Y
$HAM(X, Y)$	Hamming distance between X and Y q
SK', SK	Session keys
ΔT	"Maximum transmission delay associated with a message"
$\|, \oplus$	"Data concatenation" and "exclusive-OR" operators

1) Enrollment Phase: In this all the communication between server and device takes place in offline mode and through a secure private channel. The process consists of the following steps.

Step 1: The Device sends unique identity ID_i to server.

Step 2: The server generates 128-bit secret key r and K_{DC} for each device. Server calculates $RID = h(\text{ID}_i \| r)$. The server sends $\{RID, K_{DC}\}$ to device and stores $\{RID, \text{ID}_i, r, K_{DC}\}$ in its secured database.

Step 3: After receiving $\{RID, K_{DC}\}$ from server, device generates 128-bit secret key n, and enter password PW_i. The device calculates $RPW = h(PW_i \| n)$, $BI = h(\text{ID}_i) \oplus n$ and $RID' = RID \oplus h(IDi \| n)$ and stores them.

2) Login Phase: The login phase consists of the following steps.

Step 1: The Device login with unique identity IDm'_i and password PW'_i. Device calculates $n' = BI \oplus h(\text{ID}_i)$ and

DEVICE	*SERVER*
START OF ENROLLMENT PHASE IN OFFLINE AND SECURE CHANNEL	
Generate ID_i	Generate 128-bit secret r and K_{DC} for each device
$\xrightarrow{\{ID_i\}}$	Calculate *RID*=$h(ID_i \| r)$, store $\{RID, ID_i, r, K_{DC}\}$
(via secure channel)	
	$\xleftarrow{\{RID, K_{DC}\}}$
	(via secure channel)
Generate 128-bit secret n and enters password PW_i	
Store $RPW = h(PW_i \| n)$, $BI = h(ID_i) \oplus n$	
and $RID' = RID \oplus h(IDi \| n))$	
END OF ENROLLMENT PHASE	
START OF LOGIN PHASE IN PUBLIC CHANNEL	
Enter ID'_i and password PW'_i, calculate and fetch corresponding device parameters	
$n' = BI \oplus h(ID'_i)$	$\{RID, ID_i, r, K_{DC}\}$ based on equality of TID
$RPW' = h(PW'_i, n')$	and TID', Accept/Reject
Check if $RPW' == RPW$	Calculate 128-bit temporary identity TC
Accept/Reject	$ID_C = (TC \| r)$
Calculate $RID = RID' \oplus h(IDi \| n))$	$ID_{C'} = ID_C \oplus RID$
$TID = h(RID \| ID_i \| T_S)$	$H_{IDC} = h(ID_C)$
$\xrightarrow{\{TID, T_S\}}$	$\xleftarrow{\{ID_{C'}, H_{IDC}\}}$
Calculate $ID_{C''} = ID_{C'} \oplus RID$	
$H_{IDC'} = h(ID_{C''})$	
Check if $H_{IDC} == H_{IDC'}$	
Accept/Reject	
Proceed ahead with connection	
END OF LOGIN PHASE	

Fig. 3. Summary of enrollment and login phase.

$RPW' = h(PW'_i, n')$. If $RPW' == RPW$, the device proceeds ahead with connection, otherwise the device terminates the connection.

Step 2: Device calculates $RID = RID' \oplus h(IDi \| n))$ and $TID = h(RID \| IDi \| T_S)$ at timestamp T_S. The device sends $\{TID, T_S\}$ to server.

Step 3: Upon receiving $\{TID, T_S\}$ from device, server fetches corresponding device parameters $\{RID, \mathrm{ID}_i, r, K_{DC}\}$ based on equality of TID and TID' ($TID' = h(RID \| IDi \| T_S)$). If no such TID' is found then connection is aborted.

Step 4: Server generates 128-bit temporary identity TC and calculates $\mathrm{ID}_C = (TC \| r), \mathrm{ID}'_C = \mathrm{ID}_C \oplus RID$, and $H_{IDC} = h(I_{DC})$. Server sends $\{\mathrm{ID}_{C'}, H_{IDC}\}$ to device.

Step 5: Upon receiving $\{\mathrm{ID}_{C'}, H_{IDC}\}$ from server, device calculates $\mathrm{ID}_{C''} = \mathrm{ID}_{C'} \oplus RID$ and $H_{IDC'} = h(I_{DC''})$. If $H_{IDC} == H_{IDC'}$, then $\mathrm{ID}_{C''} = \mathrm{ID}_C$ and the device proceeds ahead with connection, otherwise the devices terminates the connection.

B. Authentication and Key Agreement Phase

The proposed authentication and key agreement phase summarized in Fig. 4. The step-wise discussion of this is elaborated as follows.

1) Authentication: In the authentication phase, mutual authentication between server and device is established. This process consists of the following steps.

Step 1: The devices send the $h(\mathrm{ID}_C)$ to server through a public channel after the model is trained at device.

Step 2: The server receives $h(\mathrm{ID}_C)$ from device and verifies against the $h(\mathrm{ID}_C)$ calculated during login phase. If successful, server fetches the corresponding ID_C and K_{DC} of the device, otherwise server terminates the connection.

Step 3: To ensure the authenticity of ID_C that it has been sent by device, the server sends a challenge to device. As a challenge the server generates a nonce R_N at current timestamp T_1. The server calculates $M_1 = LR((R_N \oplus \mathrm{ID}_C \oplus T_1 \oplus K_{DC}), HAM(\mathrm{ID}_C, T_1))$. To ensure the integrity of the message server also calculates $\mathrm{hash}_1 = h(M_1 \| \mathrm{ID}_C \| R_N \| T_1)$. The server sends $\{M_1, \mathrm{hash}_1, T_1\}$ to the device.

Step 4: After receiving message $\{M_1, \mathrm{hash}_1, T_1\}$ from server at time T_{cur}, the device verifies if $|T_{\mathrm{cur}} - T_1| < \Delta T$ or not, where ΔT is the "maximum transmission delay." If the condition is true, device accepts the message and calculates $R_{N'} = RR(M1, HAM(\mathrm{ID}_C, T_1)) \oplus \mathrm{ID}_C \oplus T_1 \oplus K_{DC})$ and $\mathrm{hash}_{1'} = h(M_1 \| \mathrm{ID}_C \| R_{N'} \| T_1)$. The device compares hash'_1 and hash_1. If $\mathrm{hash}_{1'} == \mathrm{hash}_1$, the device has solved the challenge. If $\mathrm{hash}_{1'}! = \mathrm{hash}_1$, the device terminates the connection.

Step 5: To authenticate server and to complete mutual authentication, the device generates a nonce R_C at current timestamp T_2. The device calculates $M_2 = LR((R_N \oplus \mathrm{ID}_C \oplus T_2 \oplus K_{DC}), HAM(\mathrm{ID}_C, T_2))$ and $M_3 = LR((R_N \oplus \mathrm{ID}_C \oplus R_C \oplus K_{DC}), HAM(R_N, \mathrm{ID}_C))$. To ensure the integrity of the message device also calculates $\mathrm{hash}_2 = h(M_2 \| M_3 \| R_N \| R_C \| T_2)$.The device sends $\{M_2, M_3, \mathrm{hash}_2, T_2\}$ to the server.

Step 6: After receiving message $\{M_2, M_3, \mathrm{hash}_2, T_2\}$ from device at time T_{cur}, the server verifies if $|T_{\mathrm{cur}} - T_2| < \Delta T$. If the condition is true, server accepts the message and calculates

$DEVICE$	$SERVER$
Generate H_{ID}=$h(ID_C)$	Verify H_{ID}, fetch corresponding ID_C, K_{DC}
$\xrightarrow{\{H_{ID}\}}$	and calculate M_1=$LR((R_N \oplus ID_C \oplus T_1 \oplus K_{DC}), HAM(ID_C, T_1))$
Check if $\lvert T_{cur} - T_1\rvert < \Delta T$?	$hash_1 = h(M_1 \| ID_C \| R_N \| T_1)$
Accept/Reject and calculate	$\xleftarrow{\{M_1, hash_1, T_1\}}$
$R_{N'}$=$RR(M1, HAM(ID_C, T_1)) \oplus ID_C \oplus T_1 \oplus K_{DC})$	
$hash_{1'} = h(M_1 \| ID_C \| R_{N'} \| T_1)$	
Check if $hash_{1'} == hash_1$, Accept/Reject	Check if $\lvert T_{cur} - T_2\rvert < \Delta T$? Accept/Reject
M_2 = $LR((R_N \oplus ID_C \oplus T_2 \oplus K_{DC}), HAM(ID_C, T_2))$	$\mathrm{R}_{N'} = RR(M2, HAM(ID_C, T_2)) \oplus ID_C \oplus T_2 \oplus K_{DC})$
M_3 = $LR((R_N \oplus ID_C \oplus R_{CDC}), HAM(R_N, ID_C))$	Check if $R_{N'} == R_N$? Accept/Reject
$hash_2 = h(M_2 \|\| M_3 \| R_N \| R_C \| T_2)$	$R_{C'} = RR(M3, HAM(R_N, ID_C)) \oplus R_N \oplus ID_C \oplus K_{DC})$
$\xrightarrow{\{M_2, M_3, hash_2, T_2\}}$	$hash_{2'} = h(M_2 \| M_3 \| R_N \| T_2 \| R_{C'})$
	Check if $hash_{2'} == hash_2$? Accept/Reject
Check if $\lvert T_{cur} - T_3\rvert < \Delta T$?	$SK = h(ID_C \|\| R_C \| R_N \| K_{DC} \| T_1 \| T2)$
Accept/Reject. Calculate	$hash_3 = h(SK \| R_N \| R_C \| T_1 \| T_2 \| T_3)$
$SK' = h(ID_C \|\| R_C \| R_N \| K_{DC} \| T_1 \| T2)$	$\xleftarrow{\{hash_3, T_3\}}$
$hash_{3'} = h(SK' \| R_N \| R_C \| T_1 \| T_2 \| T_3)$	Check if $\lvert T_{cur} - T_4\rvert < \Delta T$?
Check if $hash_{3'} == hash_3$? Accept/Reject	Accept/Reject
calculate $ACK = h(SK \| T_4)$	Calculate $ACK' = h(SK \| T_4)$
$\xrightarrow{\{ACK, T_4\}}$	Check if ACK' == ACK? Accept/Reject
Both $Server$ and $Device$ have common session key SK	

Fig. 4. Summary of authentication and key agreement phase.

$R_{N'} = RR(M_2, HAM(\mathrm{ID}_C, T_2)) \oplus \mathrm{ID}_C \oplus T_2 \oplus K_{DC})$. The server compares $R_{N'}$ and R_N. If $R_{N'} == R_N$, the server verifies that the device has successfully solved the challenge. If $R_{N'}$!= R_N, the server terminates the connection.

Step 7: After verifying the nonce, server calculates the nonce generated by device $R_{C'} = RR(M_3, HAM(R_N, \mathrm{ID}_C)) \oplus R_N \oplus \mathrm{ID}_C \oplus K_{DC})$ and $\mathrm{hash}_{2'} = h(M_2 \| M_3 \| R_N \| T_2 \| R_{C'})$. The server compares hash_2' and hash_2. If $\mathrm{hash}_{2'} == \mathrm{hash}_2$, the mutual authentication between server and client is complete, otherwise if $\mathrm{hash}_{2'}! = \mathrm{hash}_2$, the devices terminates the connection.

2) Key Agreement: In the key agreement phase, a session key is shared between server and device. This process consists of the following steps.

Step 1: The server calculate session key $SK = h(\mathrm{ID}_C \| R_C \| R_N \| K_{DC} \| T_1 \| T2)$ and sends $\mathrm{hash}_3 = h(SK \| R_N \| R_C \| T_1 \| T_2 \| T_3$ generated at T_3 to device. The server sends $\{\mathrm{hash}_3, T_3\}$ to the device.

Step 2: After receiving message $\{\mathrm{hash}_3, T_3\}$ from server at time T_{cur}, the device verifies if $\lvert T_{\mathrm{cur}} - T_3\rvert < \Delta T$. If the condition is true, device accepts the message and calculates $SK' = h(\mathrm{ID}_C \| R_C \| R_N \| K_{DC} \| T_1 \| T2)$ and $\mathrm{hash}_{3'} = h(SK' \| R_N \| R_C \| T_1 \| T_2 \| T_3)$. The devices compares hash_3' and hash_3. If $\mathrm{hash}_{3'} == \mathrm{hash}_3$, the device is assured of the correct session key received from server. If $\mathrm{hash}_{3'}! = \mathrm{hash}_3$, the device terminates the connection.

Step 3: The device generates an acknowledgement $ACK = h(SK \| T_4)$ at time T_4 to inform that it has received session key successfully. The device sends $\{ACK, T_4\}$ to the server.

Step 4: After receiving message $\{ACK, T_4\}$ from device at time T_{cur}, the server verifies if $\lvert T_{\mathrm{cur}} - T_4\rvert < \Delta T$. If the condition is true, server accepts the message and calculates $ACK' = h(SK \| T_4)$. The server compares ACK' and ACK. If $ACK' == ACK$, the device is assured of the correct session key received by device. If ACK' != ACK, the server terminates the connection.

TABLE II
BLOCK STRUCTURE

Attribute	Significance
$timestamp$	the time at which the block was made.
$lastHash$	1hash value of the last block.
$hash$	hash value of the current block
$data$	the transactions that the block holds
$proposer$	the public key of the creator of the block
$signature$	the signed hash of the block
$sequenceNo$	the sequence number of block

C. Blocks Creation and Addition Phase

This section provides detail explanation on creating, verifying, and adding a block in blockchain.

1) Block Creation: In our proposed scheme, as soon as an aggregated model achieves accuracy threshold value set by the server, the model parameters are collected as transactions $Tx_m (m = 1, 2, \ldots, n)$, that are used in block construction by server. These transactions are stored as payload in block. The transaction has the following attributes.

1) *id:* Unique identity of each transaction.
2) *from:* The server address.
3) *input:* The object that contains model and timestamp.
4) *hash:* SHA256 of input.
5) *signature:* The hash signed by the server.

The block structure as shown in Table II has timestamp, lastHash, hash, data, proposer, signature, and sequenceNo as its attributes.

2) Block Addition in Blockchain: In the proposed scheme, we used PBFT algorithm [7] to add a new block in the blockchain. In PBFT the proposed block committed to the blockchain is the most agreed one. The PBFT algorithm is a voting-based consensus algorithm and ensures safety and liveness.

PBFT network consists of nodes say N, ordered from 0 to $N-1$. Among these N nodes, a primary node is selected in round-robin fashion that collect transactions from server and creates a block proposal. All other nodes behave as validators. Each node store list of all member nodes in network, current primary node information, and sequence number of block being processed.

PBFT messages have the following types.

1) *PREPREPARE:* Sent by a primary node after new block proposal. The communication time is $N-1$.
2) *PREPARE:* In a preparing phase this message is broadcast by every node. The communication time is $N(N-1)$.
3) *COMMIT:* In a committing phase this message is broadcast by every node. The communication time is $N(N-1)$.
4) *FINALCOMMIT:* Block is finally committed to chain after $2 \times F + 1$ nodes agreed to commit, where F is the number of faulty nodes. Each node sends confirmation message to primary node and communication time is N.

In summary, the PBFT algorithm completes a round of consensus processes with a time complexity of $O(N^2)$. PBFT has various states and message exchange between nodes to add the block proposed by a primary node. In PBFT, a system of N nodes can tolerate F faulty nodes if $N = 3 \times F + 1$. Each decision in a practical byzantine fault tolerant system needs $2 \times F + 1$ approvals, where F is the faulty nodes. Below mentioned steps are performed by PBFT to add a block in the chain.

1) The primary node broadcasts the proposed block and PREPREPARE message for the proposed block to all the nodes in the network.
2) The validator nodes after receiving the PREPREPARE message move to PREPERPARED phase. Validators broadcast PREPARE message to all other validators.
3) The validators wait for $2 \times F + 1$ PREPARE messages and move to PREPARED state. Validators broadcast COMMIT message to all other validators.
4) The validators wait for $2 \times F + 1$ COMMIT messages and move to COMMIT state. Validators append the $2 \times F + 1$ COMMIT messages received into the block, and add the block into the blockchain.
5) Validators now move an FINALCOMMIT state when the block is inserted in the chain. Subsequently, all nodes increment its sequence number by one and reset phase to PREPREPARE.

D. AI-Based Big Data Analytics Phase

In this section, we will discuss about the data set and ML model used in federated learning. We will also discuss about experimental setup for performance analysis under various attacks.

1) Data Set: For our experiments, the heart disease data set from the Hungarian Institute of Cardiology is selected [51]. The data set is a floating-point data set and has 13 attributes and one target attribute. The attributes are age in years, sex (1 = Male and 0 = Female), chest pain type (1 = Typical angina, 2 = Atypical angina, 3 = Nonanginal pain, and 4 = Asymptomatic), resting blood pressure, serum cholesterol, fasting blood sugar (if fasting blood sugar is greater than 120 mg/dl, then 1, else 0), resting electrocardiographic results (0 = Normal, 1 = Having ST-T wave abnormality, and 2 = Showing probable or definite left ventricular hypertrophy by Estes' criteria), maximum heart rate achieved, exercise induced angina (1 = Yes and 0 = No), ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment (1 = Upsloping, 2 = Flat, and 3 = Down sloping), number of major vessels (0–3) colored by fluoroscopy, thal (3 = Normal, 6 = Fixed defect, and 7 = Reversable defect), and the target attribute is diagnosis of heart disease (0 = Less than 50% diameter narrowing and 1 = More than 50% diameter narrowing). 75% of data set has been kept for training the ML model and 25% data has been kept for testing the accuracy of model. To perform attack, a certain percentage of data set has been poisoned (i.e., noise insertion attack) or a certain percentage of the target label has been flipped (i.e., label flipping attack).

2) Genetic Algorithm Neural Network Architecture: In our work, we have taken the best of both GA and NN and merges them to create a strong classification model. GAs are used mostly to simulate environments and behaviors of entities in a population, whereas NN helps to get the correct hyperparameters [40]. But sometimes it is difficult for NN also to obtain desired hyperparameters. GA combining with NN helps to achieve the desired hyperparameters for NNs and improve the overall efficiency.

GA neural network (GANN) is an ML method designed with the complexities of transcriptional regulation in mind. The key principle is that regulatory regions are composed of features such as consensus strings, characterized binding sites, and DNA structural properties. GANN identifies these features in a set of sequences, and then identifies combinations of features that can differentiate between the positive set (sequences with known or putative regulatory function) and the negative set (sequences with no regulatory function) [40]. Once these features have been identified, they can be used to classify new sequences of unknown function.

3) Machine Learning Model Setup: ML algorithms learn how to map input variables to output variables. As a result, the size and distribution of data drawn from the domain for each variable could differ [22]. Input variables which have different units, implying that the sizes of the variables are different. Differences in scales across input variables can make the problem more difficult to model. To overcome this, in our work we have normalized the data. Normalization is the process of re-scaling data from its original range such that all values fall between 0 and 1 [52]

$$y = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}$$

where the minimum and maximum values are for the normalized value x.

In our experimental setup, the server creates a general GA-based NN model which will be shared among the devices for the training with their data sets. Server creates 15 NNs. Each NN has the same architecture. With four layers as one input layer, two hidden layers, and one output layer. Input layer has 13 neurons and two hidden layers have 20 neurons in each layer. Output layer has two neurons for two output class. Hidden layer uses rectified linear units (ReLUs) as activation function [53]. Output layer uses softmax activation function [54].

After receiving the model instance from server, the device creates list a holding the population weights as vectors, one for each network. Once the parameters of GA is prepared, the GA class is created at the device. This class has ten solutions to be selected as parents in the mating pool, 500 generations, one percentage of genes to be mutated, steady state selection parent type, single point crossover operation, random mutation, and one parent which kept for next population, 0 lower value of the random range from which the gene values in the initial population are selected, 1 upper value of the random range from which the gene values in the initial population are selected. A custom fitness function is created which is a maximization function so that a solution with a high fitness value returned is selected compared to a solution with a low value. Custom call back generation function is used to update model each in each iteration.

4) Noise Insertion: At each device the data has been poisoned by inserting noise at a certain percentage and model is trained on that noisy data. In our setup we used two popular noise function Gaussian noise and Poisson noise. Let the device data set size is D. Randomly Y percentage of the data is chosen from the data set and is poisoned by inserting Gaussian noise or Poison noise, thus $(D * Y\%)$ of the original data is replaced by noisy data in the original data set. The model is trained at the device with this noisy data set and finally we compare the results of the ML model trained on original data with ML model trained with poisoned data under the noise attack.

5) Label Flipping: We have also considered the label flipping attack at the device other than the noise insertion attack. In the label flipping attack the original labels are changed or flipped in the data set. In our case, let there be the total data set size is D for training. Y percentage of the total target label is chosen randomly and those labels are flipped, i.e., label 1 is changed to 0 and label 0 is changed to 1. These flipped labels $(D * Y\%)$ replace the original label in the data set for training.

6) Metrics Used in Experiment: To compare the performance of the model, we have used four metrics, Accuracy score, recall score, precision score, and F1 score. These metrics are calculated using a Confusion matrix. A confusion matrix is table which describes the performance of the classification model based on test data whose true values are known [55]. This matrix contains a number of true positive (TP) class, true negative (TN) class, false positive (FP) class, and false negative (FN) class. TP case is when the prediction is yes and the true value is also yes. TN case is when the prediction is no and the true value is also no. FP case is when the prediction is yes and the true value is no. FN case is when the prediction is no and the true value is yes. A confusion matrix also describes which classes are predicted accurately and inaccurately and errors made.

Accuracy: Accuracy score is the most intuitive performance measure and it is a ratio of correctly predicted observation to the total observations [56]

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}.$$

Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actual class [56]

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Precision Score: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations [56]

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

F1 Score: F1 score is the weighted average of precision and recall [57]

$$F1 - \text{score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}.$$

VI. Experiments

In this section, we performed experiments for two categories and provided comprehensive results: 1) model performance under various attacks like noise insertion attacks (Gaussian noise insertion attack and Poisson noise insertion attack) and label flipping attacks and 2) model performance without any attack.

A. Performance of Model Under Data Poisoning Attacks

After receiving model parameters from device, the server aggregates the model parameters with its own parameter and test the performance with its own test data set. If the accuracy on the test data set is more than threshold accuracy set by server, only then the aggregated model parameters are stored in the blockchain securely. It is possible that the adversary device may want to decrease the performance of the model by sending the bad model parameters to the server. It is also possible device training data set may get poisoned and device trains the model on poisoned data and send the bad model parameters to the server. We will analyze the model performance on the noisy data by applying different percentages of noise on the training data set.

1) *Gaussian Noise Insertion Attack:* Gaussian noise is statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In Fig. 5 the performance of the model is shown by varying the percentage (0%, 10%, 20%, and 30%) of data set being noisy on which the model is trained. As the percentage of noisy data increases the performance of the model decreases (10% and 20%), i.e., accuracy score, recall score, precision score and F1 score decreases. But when 30% of the total data set is noisy, the performance increases, i.e., accuracy score, recall score, precision score, and F1 score have increased. This is because adding noise during training can make the training process more robust and reduce generalization error. Also,

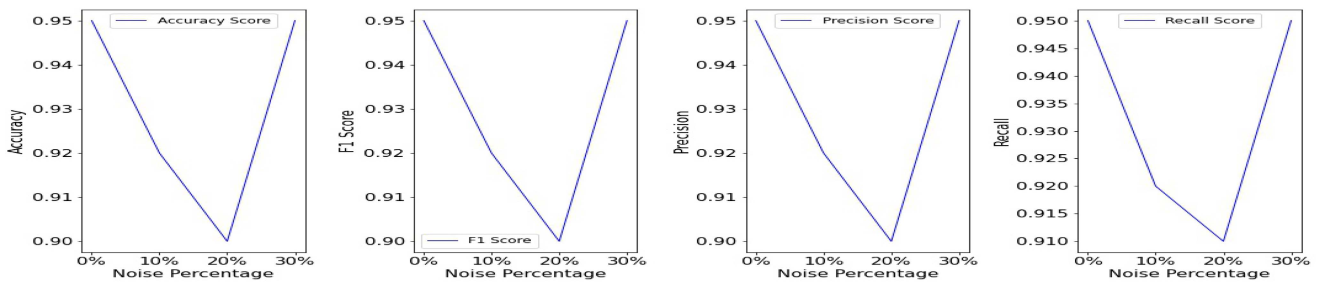


Fig. 5. Experiment results on Gaussian noise attacks.

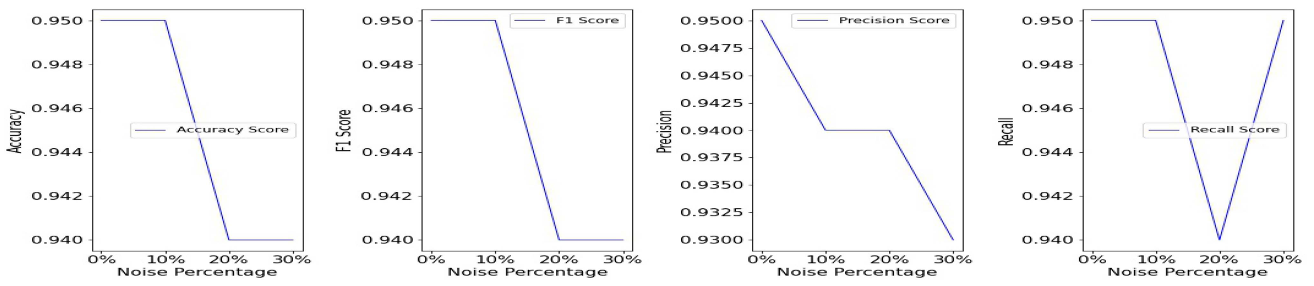


Fig. 6. Experiment results on Poisson noise attacks.

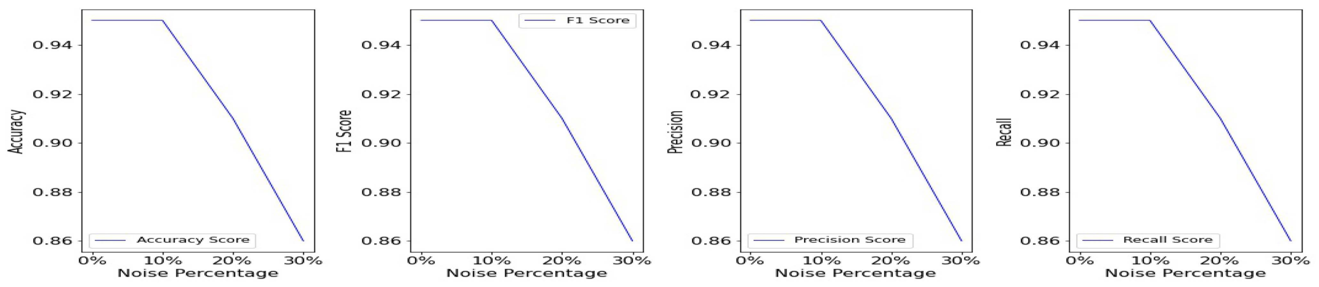


Fig. 7. Experiment results on label flipping attacks.

adding the Gaussian noise can reduce the chance of model getting over-fitted.

2) *Poisson Noise Insertion Attack:* Poisson noise is statistical noise which can be modeled by the group of Poisson processes. In Fig. 6 the performance of the model is shown by varying the percentage (0%, 10%, 20%, and 30%) of data set being noisy on which the model is trained. For the Poisson attack as the percentage of noisy data increases from 0% to 10% and from 20% to 30%, the accuracy and F1 score remains same, but the accuracy score and F1 score decreases overall as the data set becomes more noisy. As the percentage of noisy data increases from 10% to 20%, the precision score remains same, but overall it also behaves like accuracy score and F1 score. The Recall score increases with the increase in the percentage of noisy data set. Recall means the exactness of the label, because of robustness of the data set noise reduces the generalization error, as a result the recall value increases.

3) *Label Flipping Attack:* In label flipping the labels of the training data set are altered. In Fig. 7 the performance of the model is shown by varying the percentage (0%, 10%, 20%, and 30%) of labels being altered on which the model is trained. As more labels are altered in data set, the performance of the model decreases (20% and 30%), i.e., accuracy score, recall score, precision score, and F1 score decreases. In our model if 10% of the total label is altered, the performance does not decrease, i.e., accuracy score, recall score, precision score, and F1 score remain same as the clean data set.

TABLE III
MODEL PERFORMANCE ANALYSIS ON DATA SET WITHOUT ANY ATTACK

Accuracy Score	Recall Score	Precision Score	F1 Score
0.95	0.95	0.95	0.95

1) Model Performance Without Any Attack: The model performance is best when it is free from data poisoning attack. Table III shows the values of accuracy, recall, precision, and F1 score of model when no data poisoning attack are performed. The ML model has better performance in terms of accuracy, recall, precision, and F1 score when there are no data poisoning attacks on the data.

VII. ANALYSIS OF PROPOSED SCHEME

A. Security Analysis

In this section, we discuss the security analysis of the proposed authentication and session key establishment scheme. We have considered server is in a secure place and necessary security measures have been taken on the server side. The adversary may try to capture the device, read the communicated messages, analyze the traffic, try to manipulate the messages. The device can be theft or misused by users. This section discussed resilience of our scheme against these threats and attacks.

1) Off-Line Password Guessing Attack: In our protocol, we have assumed the low entropy PW_i. The attacker can guess them in polynomial time. In our protocol, ID_i and PW_i are not stored in device in plaintext format. We have calculated RPW where $RPW = h(PW_i||n)$. n is 128-bit random secret key generated during login. We have calculated BI to store n where $BI = XOR(h(\mathrm{ID}_i), n)$. We have not stored ID_i in the RID' parameters where $RID' = XOR(RID, h(\mathrm{ID}_i||n))$. RID is calculated by server which is $RID = h(\mathrm{ID}_i||r)$ and r is a 128-bit random secret generated by serve. It is not possible to guess PW_i from RPW.

2) User Anonymity and Untraceability: Like password, we have also assumed low entropy ID_i which can be guessed in polynomial time. During the registration phase, ID_i is sent to the server through the secure private channel. The server calculates RID from the IDi of device where $RID = h(\mathrm{ID}_i||r)$ and r is a 128-bit random secret generated by server and sends RID to device. Device at its end stores RID' where $RID' = XOR(RID, h(\mathrm{ID}_i||n))$. It is impossible to guess ID_i from any of the parameters as we have not stored it anywhere. RID is also calculated by server and is a cryptographic function of ID_i and 128-bit random secret r.

3) Insider Attack: In an insider attack, the attacker knows the password and ID somehow, it can access the device and perform attacks. But in our scheme, the password PW_i is hashed in $RPW(RPW = h(PW_i||n))$. It is impossible to extract the password. Password is used only once during login. Similarly, ID is also stored in device in the parameter $RID'(RID' = XOR(RID, h(\mathrm{ID}_i||n)))$ where $RID = h(\mathrm{ID}_i||r)$ and it is impossible to extract ID because of the hardness of hash function.

4) Device Impersonation Attack: In an impersonation attack if one device has got the secret credentials $\{RPW, BI, RID', K_{DC}\}$ it can act as a verified device. But during login process to calculate the 128-bit random secret n from BI, ID_i is needed as $n = XOR(BI, h(\mathrm{ID}_i))$. Without n it is not possible to calculate RPW and validate the credentials. Also it is needed to calculate TID from RID, ID_i, and T_S as $TID = h(RID, \mathrm{ID}_i, T_S)$. But ID_i is not known to the attacker.

5) Confidentiality: During the registration phase all the transmission of messages take place in the offline mode and within a secure private channel. During authentication phase, the messages $M_1 = \{LR((R_N \oplus \mathrm{ID}_C \oplus T_1 \oplus K_{DC}), HAM(\mathrm{ID}_C, T_1))\}$, $M_2 = \{LR((R_N \oplus \mathrm{ID}_C \oplus T_2 \oplus K_DC), HAM(\mathrm{ID}_C, T_2))\}$, and $M_3 = \{LR((R_N \oplus \mathrm{ID}_C \oplus R_C \oplus K_{DC}), HAM(R_N, \mathrm{ID}_C))\}$ are transmitted through public channel. These messages are hard to decipher as all the messages have random nonce to generate the messages, i.e., R_C, R_N, and K_{DC}. For an attacker to extract the valuable information out of these messages, it should have knowledge about random nonce which is very difficult. $TID(TID = h(RID, \mathrm{ID}_i, T_S))$ is computed using RID which internally uses 128-bit random secret r which is very difficult for an attacker to guess.

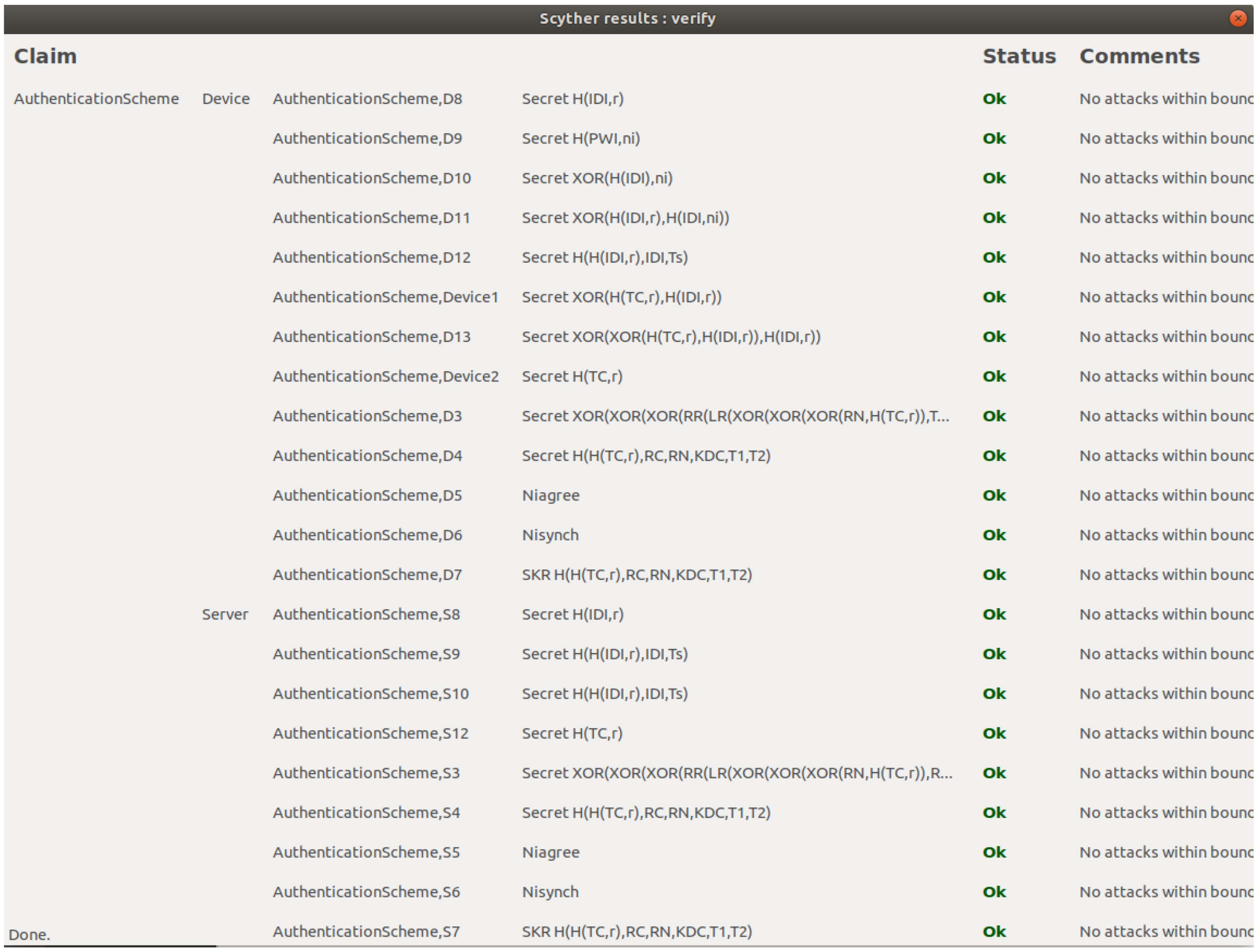
6) Integrity: During authentication phase $M_1\{M_1 = LR((R_N \oplus \mathrm{ID}_C \oplus T_1 \oplus K_{DC}), HAM(\mathrm{ID}_C, T_1))\}, M_2\{M_2 = LR((R_N \oplus \mathrm{ID}_C \oplus T_2 \oplus K_DC), HAM(\mathrm{ID}_C, T_2))\}$, and $M_3\{M_3 = LR((R_N \oplus \mathrm{ID}_C \oplus R_C \oplus K_{DC}), HAM(R_N, \mathrm{ID}_C))\}$ messages are transmitted through the public channel along with the hash of the messages. So if the messages get manipulated during transmission, it is detected by the use of hash of the messages. TID is sent from device to the server. After receiving the TID server matches with its database entry where $TID == h(RID||\mathrm{ID}_i||T_S)$. The integrity of ID_C is also ensured through the hash of the ID_C.

7) Forward and Backward Secrecy: In every session the secret key is generated from fresh newly generated random nonce and random secret. These session keys are unique in every session. If in any session adversary captures the session key, it can not have knowledge about the session keys of future and past sessions as this is totally unique in every session.

8) Replay Attack: The messages M_1, M_2, and M_3 are generated using current timestamp and random nonce in every session. A replay attack using these messages will fail. TID also has random 128-bit secret nonce r underlying. The ACK message is also generated from session key SK which internally uses random nonce R_N, R_C which are unique in every session. TID has internally random 128-bit secret r which is unique in every session. This prevents replay attack.

9) Man-in-the-Middle Attack: The adversary may act as a server for the device and device for the server by placing itself in the middle of the communication channel of server and device. The adversary may manipulate messages M_1, M_2, and M_3. But this fails due to lack of knowledge of random secret nonce R_N, R_C as these are underlying part of the messages. Adversary also can not manipulate TID due the lack of knowledge of ID_i and 128-bit random secret r. This prevents man-in-the-middle attack.

10) Mutual Authentication and Robust Session Key: In our proposed scheme the session key is established through mutual authentication. The server generates random secret R_N which device calculates from the message M_1. The device also let server know the successful calculation of R_N through the message M_2. The device also generates random nonce R_C which server calculates from message M_3. The session key is very strong to decipher and unique in every session as in every session random secrets and timestamps are used to form the session key, i.e., the session key $SK = h(\mathrm{ID}_C||R_C||R_N||K_{DC}||T_1||T_2)$. ID_C has underlying RID, ID_i, random 128-bit secret r which is very difficult to guess. To ensure that the device and server have calculated same session key, the server send the hash of the session key to the device and the device verifies it. The device also acknowledges the server that it has successfully calculated the correct session key through acknowledgment message $ACK(ACK = h(SK||T_4))$.



Scyther results : verify

Claim				Status	Comments
AuthenticationScheme	Device	AuthenticationScheme,D8	Secret H(IDI,r)	Ok	No attacks within bounc
		AuthenticationScheme,D9	Secret H(PWI,ni)	Ok	No attacks within bounc
		AuthenticationScheme,D10	Secret XOR(H(IDI),ni)	Ok	No attacks within bounc
		AuthenticationScheme,D11	Secret XOR(H(IDI,r),H(IDI,ni))	Ok	No attacks within bounc
		AuthenticationScheme,D12	Secret H(H(IDI,r),IDI,Ts)	Ok	No attacks within bounc
		AuthenticationScheme,Device1	Secret XOR(H(TC,r),H(IDI,r))	Ok	No attacks within bounc
		AuthenticationScheme,D13	Secret XOR(XOR(H(TC,r),H(IDI,r)),H(IDI,r))	Ok	No attacks within bounc
		AuthenticationScheme,Device2	Secret H(TC,r)	Ok	No attacks within bounc
		AuthenticationScheme,D3	Secret XOR(XOR(XOR(RR(LR(XOR(XOR(XOR(RN,H(TC,r)),T...	Ok	No attacks within bounc
		AuthenticationScheme,D4	Secret H(H(TC,r),RC,RN,KDC,T1,T2)	Ok	No attacks within bounc
		AuthenticationScheme,D5	Niagree	Ok	No attacks within bounc
		AuthenticationScheme,D6	Nisynch	Ok	No attacks within bounc
		AuthenticationScheme,D7	SKR H(H(TC,r),RC,RN,KDC,T1,T2)	Ok	No attacks within bounc
	Server	AuthenticationScheme,S8	Secret H(IDI,r)	Ok	No attacks within bounc
		AuthenticationScheme,S9	Secret H(H(IDI,r),IDI,Ts)	Ok	No attacks within bounc
		AuthenticationScheme,S10	Secret H(H(IDI,r),IDI,Ts)	Ok	No attacks within bounc
		AuthenticationScheme,S12	Secret H(TC,r)	Ok	No attacks within bounc
		AuthenticationScheme,S3	Secret XOR(XOR(XOR(RR(LR(XOR(XOR(XOR(RN,H(TC,r)),R...	Ok	No attacks within bounc
		AuthenticationScheme,S4	Secret H(H(TC,r),RC,RN,KDC,T1,T2)	Ok	No attacks within bounc
		AuthenticationScheme,S5	Niagree	Ok	No attacks within bounc
		AuthenticationScheme,S6	Nisynch	Ok	No attacks within bounc
Done.		AuthenticationScheme,S7	SKR H(H(TC,r),RC,RN,KDC,T1,T2)	Ok	No attacks within bounc

Fig. 8. Scyther tool verification results.

B. Storage Analysis

1) In our approach the device does not send the data to the server. Instead, device sends only the model parameters which gets stored in the blockchain after successful passing the accuracy test at server side. This reduces the overhead of storing huge data set for training of the AI/ML model.
2) For every end device the server stores only the successfully verified parameters in the blockchain. One block takes 368 bytes in total to store the parameters. Suppose if there are 10 million blocks for each 10 million devices, the total memory needed will be around 3.68 GB in our proposed scheme.
3) In every session the devices and server stores only the session key. After establishing the session key random nonce and time stamp values are deleted from device and server.

C. Computation Analysis

1) During the registration, login, mutual authentication, and key establishment phase the used methods are XOR of two strings, calculating hash of string and rotation of string with the help of hamming distance.
2) The time taken to establish the session key between server and device 0.018 s.
3) In the registration phase there are two hash operations and two XOR operations involved. In the login phase there are four XOR operations and six hash operations. In the authentication and key establishment phase there are 12 hash functions, 12 XOR operations, three left rotations, and three right rotations involved.
4) The size of message M1, M2, and M3 is 52 bytes. The hash of the strings produces 64 bytes hash string. The session key is of 16 bytes (128 bits).

VIII. Formal Security Verification Using SCYTHER: Simulation Study

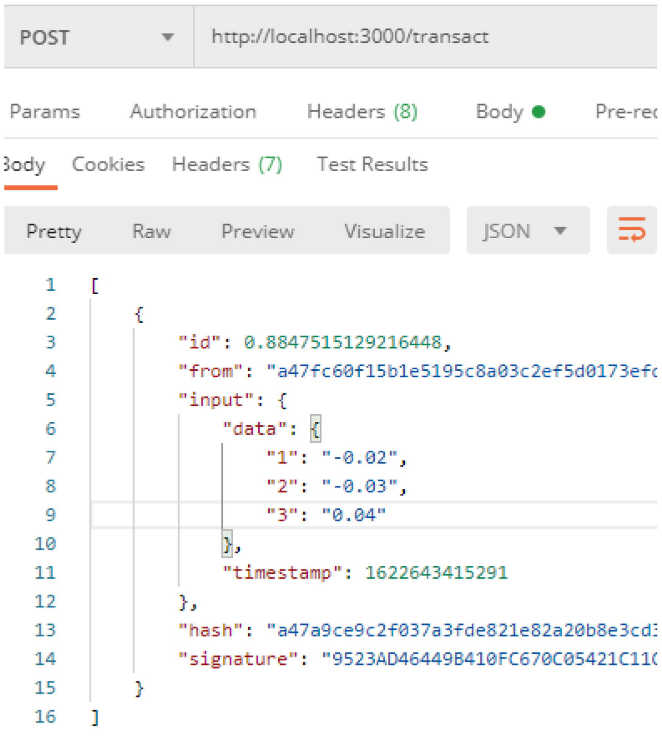
The formal security verification of the proposed scheme is done by using a widely accepted simulation tool, known as Scyther [58]. In this section, we will show that our authentication scheme is secure using a scyther simulation tool. Scyther is a tool that automatically verifies security protocols. It performs good for a large number of nonces and sessions. Scyther is a GUI-based tool that is user-friendly, practical, and simple to understand. In terms of syntax, Scyther is similar to C/Java/Python. Scyther is a programming language that is used to describe protocols that have a specified set of functions. Multiprotocol attacks can also be tested in Scyther. The verification tool is launched by pressing F1 or going to the verify menu, and it displays a result description that explains whether the protocol is attack proof or not. If the protocol is wrong, a button appears next to the argument that can be clicked to learn more about the attacks.

In the proposed protocol, the device and server claim that ID_C and K_{DC} are secret. In the first step of the authentication scheme, device sends *HID* through send_1 function and server receives *HID* through recv_1 function. Server matches *HID* through match function. In second step, server sends $M1$ through send_2 function and device receives $M1$ through recv_2 function. Device matches $hash_1$ through match function. In third step, device sends $M2$, $M3$ through send_3 function and server receives $M2$, $M3$ through recv_3 function. Server matches R_N and $hash_2$ through match function. In the fourth step, session key is generated by server and $hash_3$ is send to device through send_4 function. Device receives $hash_3$ through recv_4 function, matches $hash_3$ through match function and common session key is established. Fig. 8 displays the Scyther tool's findings, which demonstrate that all of the protocol's private data is well-protected.

IX. Blockchain Implementation

We used Node.js to implement blockchain with PBFT. In our implementation, we used host computer having the configuration as: "OS: Ubuntu 18.04 LTS, Processor: Intel i7-8400 (2.80 GHz), Memory: 16 GiB, OS Type: 64 bit, Disk Type: HDD, Disk Size: 1 TB." Our implementation consists of the following components.

1) *Transact:* This is POST API call, it is called by server after the model has been trained to create and store model parameters as data objects into transaction. Fig. 9 shows the transact end point containing one transaction.
2) *Blocks:* This is GET API call, which retrieves blockchain of the transactions. Fig. 10 shows the blockchain using block end point.
3) *Account:* It creates signature of hashed transaction data using elliptical keys.
4) *Transaction:* Create and verify the transactions.
5) *Blockchain:* Add and validate block, identify proposers and create a chain.
6) *Validators:* Generate public keys for validators nodes.
7) *Sockethandler:* Broadcast and receive messages between nodes.



```
[
    {
        "id": 0.8847515129216448,
        "from": "a47fc60f15b1e5195c8a03c2ef5d0173efc
        "input": {
            "data": {
                "1": "-0.02",
                "2": "-0.03",
                "3": "0.04"
            },
            "timestamp": 1622643415291
        },
        "hash": "a47a9ce9c2f037a3fde821e82a20b8e3cd:
        "signature": "9523AD46449B410FC670C05421C11(
    }
]
```

Fig. 9. Transact end point.

X. Comparative Study

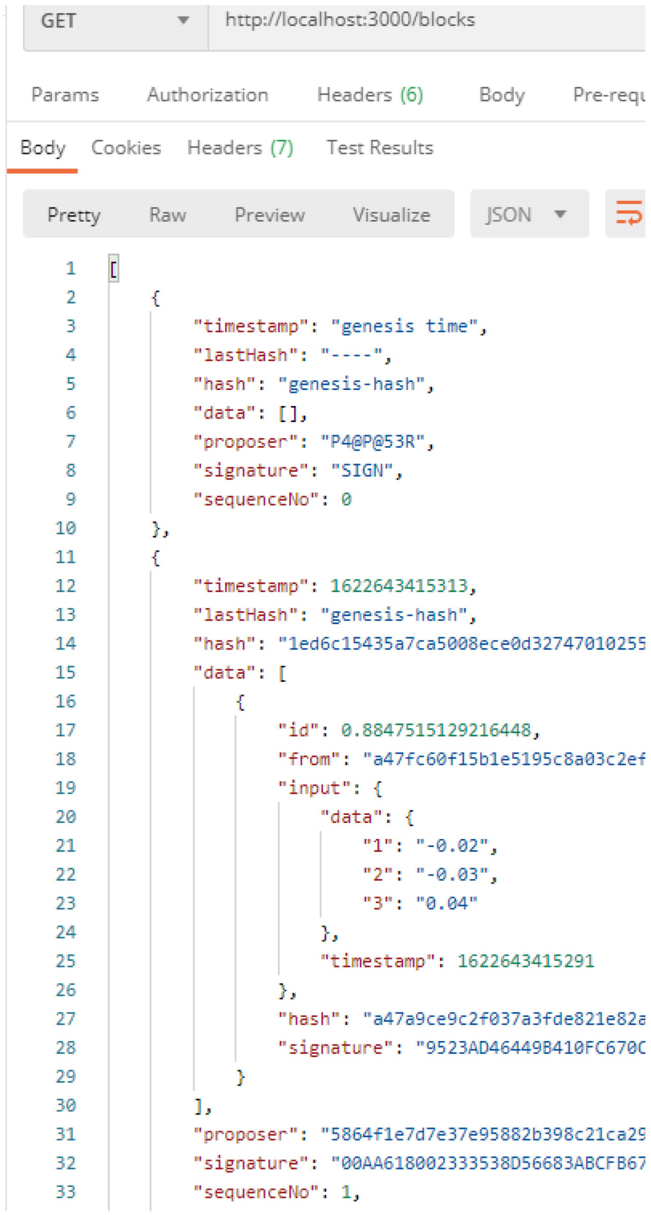
There are previously many studies which have focused on heart disease classification. Among those we have considered three of the studies for comparative study with our model.

1) AI/ML Model Performance: First study is Feature Analysis of Coronary Artery Heart Disease Data Sets [59]. This is focused on the data set of the Hungarian Institute of Cardiology and considered models are C4.5 [60], [61] and fast decision tree (FDT) [62]. Second study is Classification models for heart disease prediction using feature selection and PCA [63]. The results taken from the study are focused on the Hungarian data set. The models used for the analysis are decision tree (DT) [64], gradient-boosted tree (GBT) [65], logistic regression (LOG) [66], multilayer perceptron (MPC) [67], Na´ive Bayes (NB) [64], and random forests (RFs) [68] with Chi-Square [69] and Principal Component Analysis [70]. Third study is Diagnosing Coronary Heart Disease using Ensemble ML [71]. The model is based on the adaptive Boosting algorithm to construct ensemble learning consisting of an optimally weighted majority vote on a number of individual classifiers [71]. The performance of the models has been shown in Fig. 11.

2) Security: In this section, we compare the proposed approach with Lu et al. [16], and Kim and Hong [18] on security features. Security comparison is tabulated in Table IV on various functionality. Table V shows the ML model used for training. In our scheme we used GANN. It is worth notice that the proposed scheme and Lu et al. scheme implement an encrypted model store in blockchain. Moreover, the proposed scheme provides AI/ML-based security for big data analytics and formal security verification using the Scyther tool. Considering all the features our scheme provides better security and more functionality features as compared to those for other existing competing schemes

XI. Conclusion

In this article, we proposed a secured federated learning approach for big data analytics based on authenticated key agreement protocol and blockchain. We used GANN to train our model and proposed secure and efficient authentication and key agreement scheme to ensure secure model



```
[
    {
        "timestamp": "genesis time",
        "lastHash": "----",
        "hash": "genesis-hash",
        "data": [],
        "proposer": "P4@P@53R",
        "signature": "SIGN",
        "sequenceNo": 0
    },
    {
        "timestamp": 1622643415313,
        "lastHash": "genesis-hash",
        "hash": "1ed6c15435a7ca5008ece0d32747010255
        "data": [
            {
                "id": 0.8847515129216448,
                "from": "a47fc60f15b1e5195c8a03c2ef
                "input": {
                    "data": {
                        "1": "-0.02",
                        "2": "-0.03",
                        "3": "0.04"
                    },
                    "timestamp": 1622643415291
                },
                "hash": "a47a9ce9c2f037a3fde821e82a
                "signature": "9523AD46449B410FC670C
            }
        ],
        "proposer": "5864f1e7d7e37e95882b398c21ca29
        "signature": "00AA618002333538D56683ABCFB67
        "sequenceNo": 1,
```

Fig. 10. Block end point.

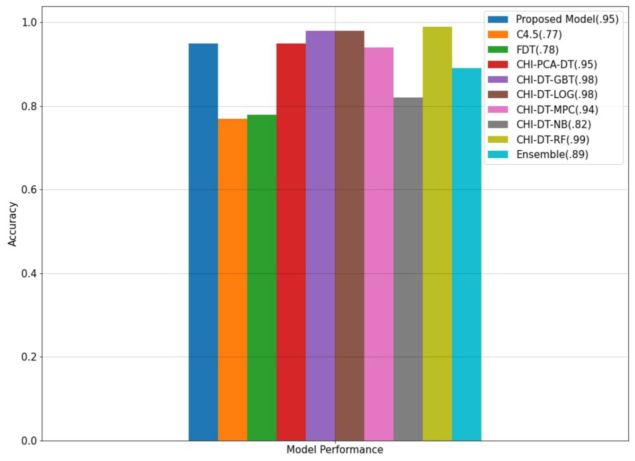


Fig. 11. Comparative study among the models.

TABLE IV
SECURITY FEATURES COMPARISON

Features	Lu et al. [16]	Kim et al. [18]	Proposed Scheme
Security against Offline Password Guessing	No	No	Yes
Confidentiality	No	No	Yes
Anonymity and Untraceability	No	No	Yes
Mutual Authentication and Session Key	No	No	Yes
Forward Secrecy	No	No	Yes
Backward Secrecy	No	No	Yes
Security against Device Impersonation Attack	No	No	Yes
Security against Replay Attack	No	No	Yes
Security against Man-in-the-middle Attack	No	No	Yes
Security against Insider Attack	No	No	Yes
Data Integrity	No	No	Yes
Security against Data Poisoning	Yes	Yes	Yes
Blockchain	Yes	Yes	Yes
Encrypted Model store in Blockchain	Yes	No	Yes

TABLE V
MODEL TYPES

	Lu et al. [16]	Kim and Hong [18]	Proposed Scheme
Model	Regression Model	Mutlilayer Perceptron	Classifier using Genetic Algorithm Neural Network

parameters exchange in federated learning. The transactions created as model parameters among IoT smart devices are securily stored in blocks and are added to blockchain using PBFT. Experiments and performance analysis performed on our proposed ML model shows that it is efficient under various attacks. A detailed formal security analysis and formal security verification with the help of widely used Scyther simulation tool shows that the proposed scheme is resilient against several attacks present in the IoT environment. A rigorous comparative study shows that the proposed scheme provides accurate, secure and resilient federated ML model for big data analytics.

In the future, the work can be extended from the following perspectives: federated learning methods, environment of federated learning, and privacy constraints. In federated learning methods perspective, to address challenges of scalability, heterogeneity, and privacy in case of unsupervised learning where labels in federated networks is undefined or weak. In environment of federated leaning perspective, to address issues, such as when the underlying data-generation model changes over time or when the devices exhibit different behavior at different times. Finally, to explore on enhancing privacy in distributed ML and privacy-preserving methods specifically at more granular level, as privacy constraints may differ across devices or even across data points on a single device.

Acknowledgment

The authors would like to thank the anonymous reviewers and the Associate Editor for their valuable suggestions and feedback on this article.

References

[1] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, "A survey on big data market: Pricing, trading and protection," *IEEE Access*, vol. 6, pp. 15132–15154, 2018.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] W. Yu et al., "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[4] R. S. Bali and N. Kumar, "Secure clustering for efficient data dissemination in vehicular cyber–physical systems," *Future Gener. Comput. Syst.*, vol. 56, pp. 476–492, Mar. 2016.

[5] N. Kumar, R. Iqbal, S. Misra, and J. J. P. C. Rodrigues, "An intelligent approach for building a secure decentralized public key infrastructure in VANET," *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 1042–1058, 2015.

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" 2020, *arXiv:2003.14053*.

[7] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[8] U. U. Uchibeke, K. A. Schneider, S. H. Kassani, and R. Deters, "Blockchain access control ecosystem for big data security," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2018, pp. 1373–1378.

[9] L. Yue, H. Junqin, Q. Shengzhi, and W. Ruijin, "Big data model of security sharing based on blockchain," in *Proc. 3rd Int. Conf. Big Data Comput. Commun. (BIGCOM)*, 2017, pp. 117–121.

[10] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: A panacea for healthcare cloud-based data security and privacy?" *IEEE Cloud Comput.*, vol. 5, no. 1, pp. 31–37, Jan./Feb. 2018.

[11] Z. Chang, W. Guo, X. Guo, Z. Zhou, and T. Ristaniemi, "Incentive mechanism for edge-computing-based blockchain," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7105–7114, Nov. 2020.

[12] S. Jangirala, A. K. Das, and A. V. Vasilakos, "Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7081–7093, Nov. 2020.

[13] H. Yang, Y. Liang, J. Yuan, Q. Yao, A. Yu, and J. Zhang, "Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5G and beyond," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7094–7104, Nov. 2020.

[14] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services," *IEEE Access*, vol. 5, pp. 25808–25825, 2017.

[15] S. Challa, A. K. Das, P. Gope, N. Kumar, F. Wu, and A. V. Vasilakos, "Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems," *Future Gener. Comput. Syst.*, vol. 108, pp. 1267–1286, Jul. 2020.

[16] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020, doi: 10.1109/TII.2019.2942190.

[17] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 3975–3986, Jul. 2021.

[18] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *Proc. 20th Asia–Pacific Netw. Oper. Manag. Symp. (APNOMS)*, Matsue, Japan, 2019, pp. 1–4.

[19] R. Mitchell, J. Michalski, and T. Carbonell, *An Artificial Intelligence Approach*. Berlin, Germany: Springer, 2013.

[20] D. Dobrev, "A definition of artificial intelligence," 2012, *arXiv:1210.1568*.

[21] E. Bryndin, "Robots with artificial intelligence and spectroscopic sight in hi-tech labor market," *Int. J. Syst. Sci. Appl. Math.*, vol. 4, no. 3, pp. 31–37, 2019.

[22] T. M. Mitchell et al., *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.

[23] E. Alpaydin, *Introduction to Machine Learning* (Adaptive Computation and Machine Learning), 3rd ed. Cambridge, MA, USA: MIT Press, 2014.

[24] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *Int. J. Security Netw.*, vol. 10, no. 3, pp. 137–150, 2015.

[25] Q. Xue and M. C. Chuah, "New attacks on RNN based healthcare learning system and their detections," *Smart Health*, vols. 9–10, pp. 144–157, Dec. 2018.

[26] M. Z. Alom et al., "The history began from alexnet: A comprehensive survey on deep learning approaches," 2018, *arXiv:1803.01164*.

[27] Y. Xin et al., "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[28] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009, pp. 9–41.

[29] H. B. Barlow, "Unsupervised learning," *Neural Comput.*, vol. 1, no. 3, pp. 295–311, 1989.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[31] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Inf. Sci.*, vol. 239, pp. 201–225, Aug. 2013.

[32] G. Li, P. Zhu, J. Li, Z. Yang, N. Cao, and Z. Chen, "Security matters: A survey on adversarial machine learning," 2018, *arXiv:1810.07339*.

[33] T. S. Sethi and M. Kantardzic, "Data driven exploratory attacks on black box classifiers in adversarial domains," *Neurocomputing*, vol. 289, pp. 129–143, May 2018.

[34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy (Euro S P)*, Saarbruecken, Germany, 2016, pp. 372–387.

[35] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[36] T. N. Nguyen, "Attacking machine learning models as part of a cyber kill chain," 2017, *arXiv:1705.00564*.

[37] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Cham, Switzerland: Springer Int., 2019, pp. 43–55.

[38] T. V. Mathew, "Genetic algorithm," Dept. Civil Eng., IIT Bombay, Mumbai, India, Rep., 2012.

[39] W. S. Sarle, "Neural networks and statistical models," in *Proc. 19th Annu. SAS Users Group Int. Conf.*, 1994, pp. 1–13.

[40] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *Proc. 3rd Int. Conf. Genet. Algorithms*, Fairfax, VA, USA, 1989, pp. 379–384.

[41] I. Abraham and D. Malkhi, "The blockchain consensus layer and BFT," *Bull. EATCS*, vol. 3, no. 123, pp. 1–22, 2017.

[42] N. Kshetri, "1 blockchain's roles in meeting key supply chain management objectives," *Int. J. Inf. Manag.*, vol. 39, pp. 80–89, Apr. 2018.

[43] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Bus. Rev., Seoul, South Korea, White Paper, 2008. Accessed: Mar. 2022. [Online]. Available: https://downloads.coindesk.com/research/whitepapers/bitcoin.pdf

[44] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, Vienna, Austria, 2016, pp. 3–16.

[45] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, 2017, pp. 557–564.

[46] A. Ghosh, S. Gupta, A. Dua, and N. Kumar, "Security of cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102635.

[47] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—A systematic review," *PLoS ONE*, vol. 11, no. 10, 2016, Art. no. e0163477.

[48] R. Amin, R. S. Sherratt, D. Giri, S. H. Islam, and M. K. Khan, "A software agent enabled biometric security algorithm for secure file access in consumer storage devices," *IEEE Trans. Consum. Electron.*, vol. 63, no. 1, pp. 53–61, Feb. 2017.

[49] V. Sureshkumar, R. Amin, V. R. Vijaykumar, and S. R. Sekar, "Robust secure communication protocol for smart healthcare system with FPGA implementation," *Future Gener. Comput. Syst.*, vol. 100, pp. 938–951, Nov. 2019.

[50] X. Zhang, X. Zhu, and L. Lessard, "Online data poisoning attacks," in *Proc. Int. Conf. Learn. Dyn. Control*, 2020, pp. 201–210.

[51] D. Dua and C. Graff. "UCI machine learning repository." 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[52] T. Lee, V. P. Singh, and K. H. Cho, "Data preprocessing," in *Deep Learning for Hydrometeorology and Environmental Science*. Cham, Switzerland: Springer, 2021, pp. 21–25.

[53] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.

[54] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activiation functions," in *Proc. 8th Aust. Conf. Neural Netw.*, vol. 181. Melbourne, VIC, Australia, 1997, pp. 181–185.

[55] R. Susmaga, "Confusion matrix visualization," in *Intelligent Information Processing and Web Mining*. Berlin, Germany: Springer, 2004, pp. 107–116.

[56] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in *Proc. Austr. Joint Conf. Artif. Intell.*, 2006, pp. 1015–1021.

[57] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[58] C. J. F. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols," in *Proc. Int. Conf. Comput.-Aided Verif. (CAV)*, 2008, pp. 414–418.

[59] R. El-Bialy, M. A. Salama, O. H. Karam, and M. E. Khalifa, "Feature analysis of coronary artery heart disease data sets," *Procedia Comput. Sci.*, vol. 65, pp. 459–468, Dec. 2015.

[60] D. Lavanya and K. U. Rani, "Performance evaluation of decision tree classifiers on medical datasets," *Int. J. Comput. Appl.*, vol. 26, no. 4, pp. 1–4, 2011.

[61] A. Priyam, G. Abhijeeta, A. Rathee, and S. Srivastava, "Comparative analysis of decision tree classification algorithms," *Int. J. Current Eng. Technol.*, vol. 3, no. 2, pp. 334–337, 2013.

[62] J. Su and H. Zhang, "A fast decision tree learning algorithm," in *Proc. 21st Nat. Conf. Artif. Intell. (AAAI)*, vol. 1. Boston, MA, USA, 2006, pp. 500–505.

[63] A. K. Gárate-Escamila, A. H. El Hassani, and E. Andrès, "Classification models for heart disease prediction using feature selection and PCA," *Inform. Med. Unlocked*, vol. 19, Jan. 2020, Art. no. 100330.

[64] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Boca Raton, FL, USA: CRC Press, 2015.

[65] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[66] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2009.

[67] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[68] L. Breiman, "1 Random Forests–random Features," Dept. Statistics, Univ. California, Berkeley, CA, USA, Rep. 567, 1999.

[69] H. O. Lancaster and E. Seneta, "Chi-square distribution," *Encyclopedia of Biostatistics*, vol. 2. Chichester, U.K.: Wiley, 2005.

[70] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[71] K. H. Miao, J. H. Miao, and G. J. Miao, "Diagnosing coronary heart disease using ensemble machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 10, pp. 30–39, 2016.