ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΠΡΟΓΡΑΜΜΑ ΔΙΔΑΚΤΟΡΙΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Τεχνολογίες κατανεμημένου καθολικού (DLT) και Μηχανική Μάθηση (Machine Learning)

Ανδρέας Ρόναλντ Σορτ

ΑΙΓΑΛΕΩ

Ιούνιος 2024

UNIVERSITY OF WEST ATTICA

DEPARTMENT OF INDUSTRIAL DESIGN AND PRODUCTION ENGINEERING

PROGRAM OF DOCTORAL STUDIES

PhD THESIS

Distributed Ledger Technologies (DLT) and Machine Learning

Andrew Ronald Short

ATHENS - EGALEO

JUNE 2024

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Τεχνολογίες κατανεμημένου καθολικού (DLT) και Μηχανική Μάθηση (Machine Learning)

Ανδρέας Ρ. Σορτ

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ: Ελένη Αικατερίνη Λελίγκου, Καθηγήτρια Τμ. ΜΒΣΠ, ΠαΔΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Ελένη Αικατερίνη Λελίγκου, Καθηγήτρια Τμ. ΜΒΣΠ, ΠαΔΑ
Παναγιώτης Καρκαζής, Αναπληρωτής Καθηγητής, Τμ. ΜΠΥ, ΠαΔΑ
Παναγιώτης Τρακάδας, Αναπληρωτής Καθηγητής, Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνων

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

(Υπογραφή)

ELENI AIKATERINI LELIGKOU Digitally signed by ELENI AIKATERINI LELIGKOU Date: 2024.10.02 14:45:47 +03'00'

Ελένη Αικατερίνη Λελίγκου,
Καθηγήτρια ΠαΔΑ

(Υπογραφή)

PANAGIOTIS KARKAZIS Digitally signed by PANAGIOTIS KARKAZIS Date: 2024.10.09 13:01:13 +03'00'

Παναγιώτης Καρκαζής,
Αν. Καθηγητής ΠαΔΑ

(Υπογραφή)

PANAGIOTIS TRAKADAS
09.10.2024 16:14

Παναγιώτης Τρακάδας,
Αν. Καθηγητής ΕΚΠΑ

(Υπογραφή)

ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ Digitally signed by ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ Date: 2024.10.09 17:45:54 +03'00'

Ευάγγελος Πάλλης,
Καθηγητής ΠΑΔΑ

(Υπογραφή)

[illegible] Digitally signed by ΜΙΧΑΗΛ ΠΑΠΟΥΤΣΙΔΑΚΗΣ Date: 2024.10.05 11:33:15 +03'00'

Μιχαήλ Παπουτσιδάκης,
Καθηγητής ΠαΔΑ

(Υπογραφή)

THEOFANIS ORFANOUDAKIS
09.10.2024 12:20

Θεοφάνης Ορφανουδάκης,
Καθηγητής ΕΑΠ

(Υπογραφή)

ANDREAS PAPADAKIS
09/10/2024 10:56

Ανδρέας Παπαδάκης,
Καθηγητής ΑΣΠΑΙΤΕ

Ημερομηνία εξέτασης ----

PhD THESIS

Τεχνολογίες κατανεμημένου καθολικού (DLT) και Μηχανική Μάθηση (Machine Learning)

Andrew Ronald Short

SUPERVISOR: Helen Catherine Leligou, Professor, Department of IDPE, UniWA

THREE-MEMBER ADVISORY COMMITTEE:
Helen Catherine Leligou, Professor, Dep. IDPE, UniWA
Panagiotis Karkazis, Associate Professor, Dep. ICE, UniWA
Panagiotis Trakadas, Associate Professor, National and Kapodestrian University of Athens

SEVEN-MEMBER EXAMINATION COMMITTEE

(Signature)

ELENI AIKATERINI LELIGKOU
Digitally signed by ELENI AIKATERINI LELIGKOU
Date: 2024.10.02 14:46:12 +03'00'

Helen Catherine Leligou,
Professor UniWA

(Signature)

PANAGIOTIS KARKAZIS
Digitally signed by PANAGIOTIS KARKAZIS
Date: 2024.10.09 16:11:20 +03'00'

Panagiotis Karkazis,
Assoc. Professor UniWA

(Signature)

PANAGIOTIS TRAKADAS
09.10.2024 16:15

Panagiotis Trakadas,
Associate Professor UoA

(Signature)

ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ
Digitally signed by ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ
Date: 2024.10.09 17:46:19 +03'00'

Evangelos Pallis,
Professor UniWA

(Signature)

[illegible]
Digitally signed by ΜΙΧΑΗΛ ΠΑΠΟΥΤΣΙΔΑΚΗΣ
Date: 2024.10.05 11:33:48 +03'00'

Michael Papoutsidakis,
Professor UniWA

(Signature)

THEOFANIS ORFANOUDAKIS
14.10.2024 11:15

Theofanis Orfanoudakis,
Professor EAP

(Signature)

ANDREAS PAPADAKIS
09/10/2024 10:58

Andreas Papadakis,
Professor ASPETE

Examination Date ----

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Ο/η κάτωθι υπογεγραμμένος Ανδρέας Ρόναλντ Σορτ του Δαυίδ, υποψήφιος διδάκτορας του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:
«Είμαι συγγραφέας και δικαιούχος των πνευματικών δικαιωμάτων επί της διατριβής και δεν προσβάλω τα πνευματικά δικαιώματα τρίτων. Για τη συγγραφή της διδακτορικής μου διατριβής δεν χρησιμοποίησα ολόκληρο ή μέρος έργου άλλου δημιουργού ή τις ιδέες και αντιλήψεις άλλου δημιουργού χωρίς να γίνεται αναφορά στην πηγή προέλευσης (βιβλίο, άρθρο από εφημερίδα ή περιοδικό, ιστοσελίδα κ.λπ.). Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

[illegible]

ΠΕΡΙΛΗΨΗ

Η παρούσα διδακτορική διατριβή ερευνά τα οφέλη που προκύπτουν από τον συνδυασμό τεχνολογιών από τους ακόλουθους τομείς: α) ομόσπονδη μάθηση (Federated Learning, β) τεχνολογίες κατανεμημένου καθολικού (Distributed Ledger Technologies) όπως είναι τα Blockchain δίκτυα και γ) βιομηχανικές εφαρμογές.

Η συνεχώς αυξανόμενη χρήση εφαρμογών τεχνητής νοημοσύνης έχει καταστήσει εμφανές ότι η ποιότητα των μεγάλων δεδομένων για εκπαίδευση μοντέλων μηχανικής μάθησης επηρεάζει την απόδοση των τελικών μοντέλων. Για το σκοπό αυτό, η ομόσπονδη μάθηση αποσκοπεί στη συμμετοχή πολλαπλών οντοτήτων που συμβάλλουν στη διαδικασία μάθησης με τοπικά δεδομένα, χωρίς να απαιτείται η κοινή χρήση των πραγματικών δεδομένων. Σε πραγματικές περιπτώσεις χρήσης, οι ανταμοιβές που λαμβάνουν οι χρήστες για τη συμβολή τους στη διαδικασία μάθησης θα πρέπει να εξαρτώνται από τη ποιότητα της συμβολής τους. Έχοντας λοιπόν υπόψιν τη σπουδαιότητα των δεδομένων εκμάθησης, η διατριβή σχεδίασε και στη συνέχεια υλοποίησε μηχανισμό ομόσπονδης μάθησης ο οποίος συντονίζεται από δίκτυο blockchain. Στη συνέχεια, εστιάζει σε τρόπους ενίσχυσης της αφοσίωσης των χρηστών, προσφέροντας "δίκαιες" ανταμοιβές, ανάλογες με τη πραγματική βελτίωση του μοντέλου (ως προς την ακρίβεια) που προσφέρουν. Επιπλέον, για να καταστεί δυνατή η αντικειμενική κρίση της ποιότητας της συνεισφοράς, σχεδιάστηκαν ειδικά έξυπνα συμβόλαια που λειτουργούν πάνω σε τεχνολογίες blockchain. Πιο συγκεκριμένα, έχει αναπτυχθεί ένας αλγόριθμος επαλήθευσης που χρησιμοποιείται για την αξιολόγηση της απόδοσης των συνεισφορών των χρηστών, συγκρίνοντας την ακρίβεια του συνολικού μοντέλου με ένα σύνολο δεδομένων επαλήθευσης. Ο αλγόριθμος έχει σχεδιαστεί για να εκτελείται μέσα σε ένα έξυπνο συμβόλαιο και να καταγράφει τις επιδόσεις (με τη μορφή συστήματος πόντων) σε ένα κατανεμημένο εδάφιο.

Η παρούσα διατριβή ανέπτυξε λύσεις για την βελτίωση ασφάλειας σε βιομηχανικούς προγραμματιζόμενους λογικούς ελεγκτές (PLC), επιτρέποντάς τους να λαμβάνουν εντολές και παραμέτρους από ένα δίκτυο blockchain. Το δίκτυο σε αυτή τη περίπτωση αναλαμβάνει δύο ρόλους: χρησιμεύει ως αμετάβλητη βάση καταγραφής ενεργειών ελέγχου (audit trails) καθώς και ως αξιόπιστη πηγή για κρίσιμες εντολές και παραμέτρους. Σε αντίθεση με τον συμβατικό έλεγχο των PLC από συσκευές διεπαφής ανθρώπου-μηχανής (HMI), αυτή η νέα προσέγγιση δεν απαιτεί πρόσβαση εγγραφής στο επίπεδο του PLC, ελαχιστοποιώντας έτσι την επιφάνεια επίθεσής του και συμβάλλοντας

στην προστασία από γνωστές ευπάθειες και νέες ευπάθειες (zero day) που εμφανίζονται συχνά σε κυβερνοεπιθέσεις. Το δίκτυο blockchain χρησιμοποιείται επίσης ως βάση δεδομένων που υποστηρίζει ιχνηλάτηση για τις ενέργειες των χρηστών, ιδιαίτερα χρήσιμο για εφαρμογές που επιβάλλουν την καταγραφή των ενεργειών των χρηστών ή για παράδειγμα όπως ορίζουν οι καλές πρακτικές παραγωγής (GMP).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τομείς χρήσης τεχνολογιών ομόσπονδης μάθησης (Federated Learning) σε συνδυασμό με τεχνολογίες κατανεμημένου καθολικού (Distributed Ledger Technologies). Ανάπτυξη λύσεων με χρήση έξυπνων συμβολαίων για την αυτόματη απόδοση επιβραβεύσεων σε χρήστες που συνεισφέρουν ποιοτικά δεδομένα για την εκμάθηση μοντέλων μηχανικής μάθησης. Χρήση των τεχνολογιών αυτών σε βιομηχανικές εφαρμογές που απαιτούν ακεραιότητα δεδομένων και αυξημένη ασφάλεια.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: blockchain, federated learning, industrial applications, PLC and SCADA devices, smart contract, ethereum networks, machine learning, rewarding algorithms, audit trail databases, security

ABSTRACT

This PhD thesis explores the benefits arising from the combining technologies from the following areas: a) Federated Learning, b) Distributed Ledger Technologies (DLTs) such as blockchains and c) Industrial applications.

The ever-increasing use of Artificial Intelligence applications has made apparent that the quality of the training datasets affects the performance of the models. To this end, Federated Learning aims to engage multiple entities to contribute to the learning process with locally maintained data, without requiring them to share the actual datasets. In real use cases, the rewards that users get for their contribution to the learning process should depend on the characteristics/quality of their contribution. With this in mind, the thesis first designed and implemented a Federated Learning process that can operate on a blockchain network. The thesis then focuses on ways to strengthen user engagement by offering “fair” rewards, proportional to the model improvement (in terms of accuracy) they offer. Furthermore, to enable objective judgement of the quality of contribution, special smart contracts have been designed that operate on blockchain technologies. More precisely, a verification algorithm has been developed that is used to evaluate the performance of users’ contributions by comparing the resulting accuracy of the global model against a verification dataset. The algorithm is designed to run inside a smart contract and record performance (in the form of a point system) on a distributed ledger.

This thesis later investigates solutions that can empower Programmable Logic Controllers (PLCs) by enabling them to query a blockchain infrastructure for commands and setpoints. The blockchain assumes a dual role in this context: serving as an immutable audit trail database as well as a trusted source for critical commands and setpoints. In contrast to the conventional paradigm of controlling PLCs through Human Machine Interface devices (HMIs), this novel approach does not require write access at the PLC level, thus minimizes its attack surface and helps protect against known and zero-day vulnerabilities often used in cyberwarfare such as in the case of the notorious Stuxnet worm. The blockchain network is used as an audit trail database for user actions, useful for applications that enforce the logging of user operations as Good Manufacturing Practices (GMP) or when required for compliance reasons. Any attempt to maliciously circumvent the logging operation would not affect the operation of a critical process. Real-world applica-tions and use cases are explored to demonstrate the tangible benefits of this approach in industrial settings. Additionally, a prototype implementation is developed in order to examine the feasibility and collect performance indicators.

SUBJECT AREA: Exploration of Federating Learning techniques. Coordination through blockchain networks and the use of smart contracts in order to offer fair rewards to users contributing quality data for machine learning. How these technologies can offer advantages in industrial applications to build upon integrity in audit trail databases and improve security.

KEYWORDS: blockchain, federated learning, industrial applications, PLC and SCADA devices, smart contract, ethereum networks, machine learning, rewarding algorithms, audit trail databases, security

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis advisor, Dr Helen C. Leligkou, for the invaluable guidance, unwavering support, and insightful feedback throughout the entire research process. Her expertise, patience, and encouragement have been instrumental in shaping this thesis.

Special thanks go to the University of West Attica, where much of the research was conducted, for providing access to resources, facilities, and the necessary infrastructure for this study as well as to the Greek Universities Special Account for Research Funding for funding part of this research.

I am indebted to my family for their unconditional love, understanding, and constant encouragement during this journey. Their belief in my abilities has been a constant source of motivation. Special thanks to my children Daphne and Steve for the happy breaks!

I would also like to acknowledge the support of my friends and colleagues who have provided valuable insights and encouragement throughout this thesis project.

Finally, I am grateful to all the participants and fellow PhD students who generously shared their time and expertise, without whom this research would not have been possible.

Λίστα Δημοσιεύσεων / List of Publications

- **Andrew R. Short**, Theofanis G. Orfanoudakis, Helen C. Leligou. (2024). PLCBlox: Using blockchain-based audit trails to generate secure PLC commands. International Journal of Innovative Research and Scientific Studies, doi: 10.53894/ijirss.v7i4.3449
- Theocharis E, Papoutsidakis Michail, **Short, Andrew**, Zisimou, Konstantia. (2023). Low-Cost Solution for Adding Safety Functions to Programmable Logic Controllers (PLCs), WSEAS TRANSACTIONS ON SYSTEMS AND CONTROL. 18, doi: 10.37394/23203.2023.18.28
- **Short Andrew**, Orfanoudakis Theofanis, Helen, Leligou. (2021). Improving Security and Fairness in Federated Learning Systems. International Journal of Network Security & Its Applications, doi:10.5121/ijnsa.2021.13604
- **A. R. Short**, H. C. Leligou and E. Theocharis, "Execution of a Federated Learning process within a smart contract", 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2021, pp. 1-4, doi: 10.1109/ICCE50685.2021.9427734
- **A. R. Short**, H. C. Leligou, M. Papoutsidakis and E. Theocharis, "Using Blockchain Technologies to Improve Security in Federated Learning Systems", 2020 IEEE 44th Annual Computers, Software, and Applications Conference, doi: 10.1109/COMPSAC48688.2020.00-96

Table of Contents

List of Figures

List of Tables

List of Algorithms

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Blockchain network	Δίκτυο κατανεμημένου καθολικού
Distributed Ledger Technologies	Τεχνολογίες κατανεμημένου καθολικού
Federated Learning	Ομόσπονδη μάθηση
Machine Learning	Μηχανική μάθηση
Programmable Logic Controller (PLC)	Προγραμματιζόμενος λογικός ελεγκτής
Human Machine Interface (HMI)	Διεπαφή ανθρώπου μηχανής
Smart Contract	Έξυπνο συμβόλαιο

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ - ACRONYMS

AI	Artificial Inteligence
API	Application Programming Interface
BC	Blockchain
CFR	Code of Federal Regulations
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technologies
DP	Differential Privacy
FDA	Food and Drug Administration
FL	Federated Learning
GDPR	General Data Protection Regulation
HLF	Hyperledger Fabric
HMI	Human Machine Interface
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Interface
IP	Internet Protocol
JSON	JavaScript Object Notation
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MPC	Secure Multiparty Computation
OPC	Open Platform Communications Unified Architecture
pBFT	Practical Byzantine Fault Tolerance
PLC	Programmable Logic Controllers
PoS	Proof of Stake
PoW	Proof of Work
RAM	Random Access Memory
ReLU	Rectified linear unit

ROM	Read only memory
SCADA	Supervisory Control and Data Acquisition
SD card	Secure Digital Card
SGD	Stochastic Gradient Descent
VM	Virtual Machine

1. Introduction

1.1 Machine Learning and Federated Learning Systems

Federated Learning (FL) [1], [2] is a relatively new field of study, whereby collaborating parties can jointly improve a model which is maintained by the coordinator (usually a central server). In this setup, the training is performed individually using locally available data, and the participating entities transmit their model updates to the coordinator. The entities (either end-devices such as mobile phones, or businesses contributing with their data) usually share the same goal, i.e. to make use of the trained model or can otherwise be motivated by financial rewards.

Machine Learning (ML) aims to build a mathematical model by inference, using training data. ML can either be supervised or unsupervised, depending on whether the training data set is labeled or not [3]. In a typical centralized Machine Learning setup, the training data needs to be collected at a central location, which can pose security and privacy risks. Other concerns include the unavailability of training data of good quality, either because of data protection regulations or because of the unwillingness of users to share their private data due to privacy concerns.

Today, large amounts of data are being generated at the edge. FL overcomes the need to transfer user data to central servers, at the same time safeguarding user privacy. Practical implementations of FL already exist, such as the FL powered application Gboard [4], a keyboard from Google for Android devices, which is used to predict words based on previous input and is improved by the small contributions of each device. With the advent of edge computing, it is foreseen that more machine learning applications will run on edge devices. This is evident by the use of Artificial Intelligence (AI)-specific CPUs in the latest mobile devices from manufacturers, such as Apple and Samsung.

FL improves on privacy, since participating entities contribute with model updates rather than raw data, however as training is performed at the edge, a new attack surface is created. An adversary may intentionally send false model updates in order to affect its performance, a method known as model poisoning [5]. Since high quality datasets are an important factor in training, it may be beneficial to reward users whose model updates improve the overall model performance.

However, a set of security attacks have already been witnessed and countermeasures either rely on adding intelligence or resort to Blockchain technology. Blockchain belongs to the family of Distributed Ledger Technologies (DLT) and as such allows for the operation of an immutable database which can be distributed (and replicated) among multiple nodes. These types of databases offer improvements on security and trust, and offer a high level of transparency. For these reasons, they can be exploited to defend against FL-targeting attacks.

1.2 Federated Learning and associated security aspects

In contrast to the centralized nature of traditional ML where data is collected in a large dataset (typically in the cloud) and the model is trained centrally, FL proposes that the ML model can reside on client devices and be trained locally. An aggregation server is required in order to transmit the global model and receive the model updates. Although the individual models being trained on each end-device may have an incomplete picture, considering that the insights of all participating parties are accumulated and that this process is repeated on several rounds, the end model tends to be of high quality. Figure 1 shows the steps involved in FL. The workflow starts by transmitting the initial model (initialized with either random or proxy data) to end devices (blue arrow). The end devices can now perform local training using locally available data (yellow arrow). During the training process, gradients are computed e.g. through a variant of Stochastic Gradient Descent (SGD)[6] on a random portion of the training dataset. In the end, these updates are received by the aggregation server (orange arrow), which in turn computes the updated model, typically by aggregating the model updates. This process is repeated until predefined criteria are met.

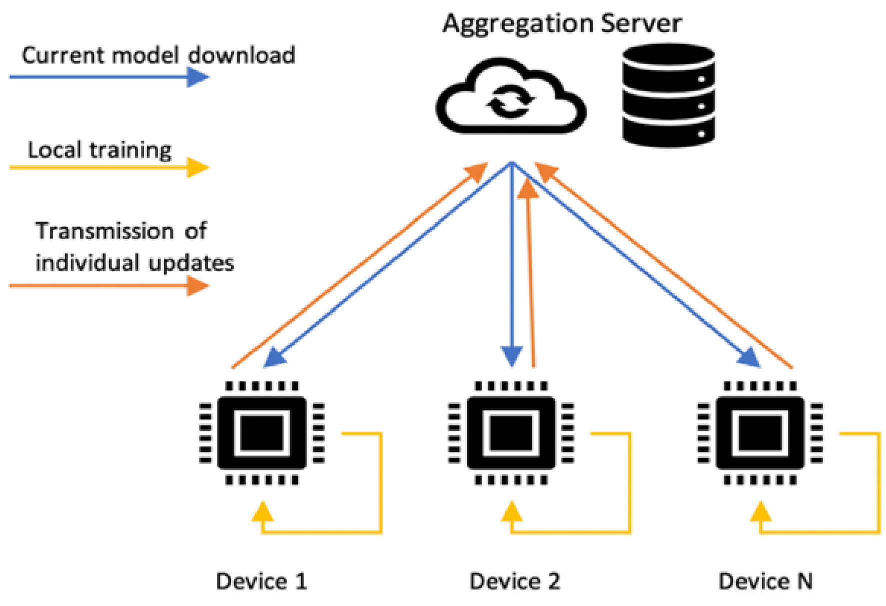


Figure 1 - Federated Learning Process

A major benefit of FL is that the end device is able to use the model even when offline. Although the learning process will be affected, the latest version of the global model will still be available. In an FL setup, the data is kept at end devices and never shared with the server. This alone offers a huge improvement in privacy and helps meet requirements of Regulations such as GDPR [7]. However, since even the model update parameters can reveal sensitive information [8], [9], ongoing research is focused on mitigating privacy concerns. It is also worth mentioning that anonymization alone is not an adequate measure for privacy. The process of anonymization is usually carried out by the server and this trust relationship is not always present. Furthermore, researchers have been able to de-anonymize datasets by using external knowledge [10], [11].

Two of the most widely adopted approaches regarding privacy are Secure Multiparty Computation (MPC) [12] and Differential Privacy [13],[14],[15]. MPC enables participating parties to jointly compute an output of a shared function based on their inputs in a way that no information on the inputs is revealed to each other. Several MPC protocols and their effectiveness under different threat models are analyzed in a study by D. Evans 2018 [16]. Differential Privacy aims to conceal personal information in client model update contributions by adding a small amount of noise. In this approach, there is a trade-off between privacy and model accuracy.

With respect to the security aspects of FL, the main vulnerabilities of this scheme stem from the fact that the learning process occurs on the edge, typically on users' devices. This has created a new attack surface where a compromised device could be manipulated in a way to affect the learning process. Specifically, an adversary could either attack the training data (data poisoning) or the model updates, with the intention of affecting the global model (model poisoning) so that its performance deteriorates and ultimately lead to denial of service [17] or in order to make it perform a certain way. It has been demonstrated that an attacker may use malicious model updates in order to alter the behavior of a global model and be able to control its behavior when certain triggers appear (e.g. certain words in word prediction applications), without otherwise affecting the performance of the model on its main task [5]. Because the model still performs well on its main task, this type of attack cannot be easily detected by the aggregation server. In the worst case, when MPC is used, the source of the attack cannot be identified since the server does not have access to the individual model updates. The same study [5] shows that the attack is possible even with a 1% of compromised devices and can survive multiple training rounds.

1.3 The potential benefits of Blockchain adoption in Federated Learning

Distributed Ledger Technologies are a type of distributed databases collaboratively built (many entities contribute data) and replicated in a number of nodes (replicas of the whole database are kept in many nodes). In essence, they provide a shared and consistent data store. In contrast to traditional databases, the individual data records cannot be updated or deleted and are usually cryptographically interconnected. Most common categories of DLTs are Blockchains and Directed Acyclic Graphs (DAG). Although both of these technologies record transactions in the distributed ledger, they do so in different ways. In a blockchain network, the blocks form a linear chain of transactions in a chronological order [18]. By comparison, new transactions in a DAG network can link to multiple previous transactions. When this link is established, the previous transactions get verified. Due to this branching of existing transactions, the DAG usually resembles a tree. Furthermore, variants of DLT technologies can be further classified as permissioned or permission-less and public or private.

The concept of smart contracts first appeared in 1997 [30] as a way to formalize and secure relationships on public networks. They are formed by machine executable code which is able to release digital assets to untrusted parties when certain pre-defined rules have been met. Although first generation blockchain networks such as Bitcoin [18] are not designed for such use cases, later public and private blockchain technologies such as Ethereum [31] and HyperLedger Fabric are designed to support the development and execution of customized smart contracts [32].

Figure 2 depicts how users of a blockchain network do not directly interact with the distributed ledger; instead, a smart contract is used in order to make queries or add/append data to it.

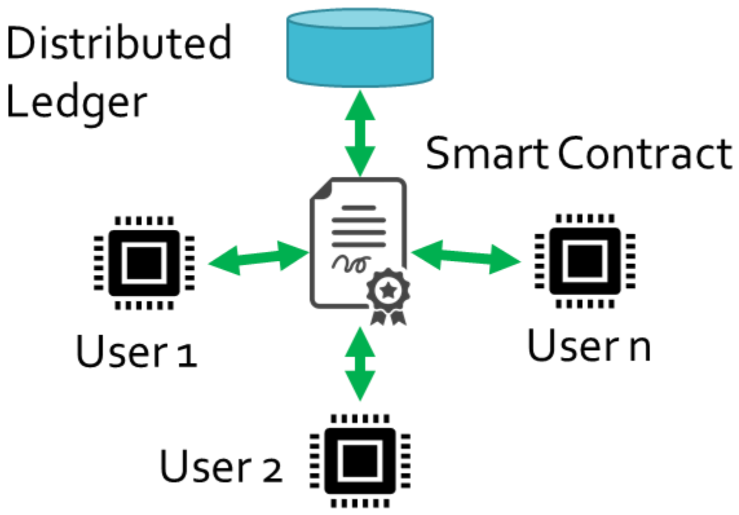


Figure 2 - Smart contract operation principle

It is anticipated the following solutions offer advancements in the following areas:

Data integrity: FL describes a process for collaboratively improving a shared model and does not deal with security aspects. In its basic implementation, the process is coordinated from a single central server. The entity hosting the solution, could alter the data, either with malicious intent or unintentionally. Blockchain networks on the other hand are a proven technology that is inherently secure. All blocks are cryptographically interconnected, so in the case that a block is altered, this would be easily identifiable. In addition to this, due to the decentralized nature of blockchain technologies, the original data will be accessible from the rest of the healthy nodes.

Reliability: Since a blockchain network is decentralized, full copies of the ledger are maintained by multiple nodes. For this reason, there is no single point of failure since the ledger will be available elsewhere in case of a node failure. This is not the case with a minimal FL setup with only one parameter server. Instead, a suitable failover design would need to be implemented. Traditional failover solutions involve the deployment of redundant servers (VMs, Kubernetes), and mechanisms for data replication. We anticipate that the use of a BC network in a FL process will offer improvements in terms of reliability.

Trust: In many cases, the participants in an FL setup might be corporations that wish to work collaboratively to build AI models in either a horizontal or vertical FL system. In a Horizontal FL System, the entities contribute with data of the same structure such as banks contributing with their client lists for fraud detection or the case of hospitals contributing with patient data. In the case of Vertical FL, the entities contribute with datasets containing different features, as in the case of a bank and an e-commerce company. As a part of the fourth industrial revolution, Industry 4.0 embraces data exchange and the creation of digital twins of the manufacturing processes is at its hype. Therefore, it is very likely that FL will become a priority for manufacturing industries. Both of the above learning systems usually require a trusted third party to coordinate the FL process and create the model. A blockchain is a good candidate to act as a trusted coordinator, because of its security and traceability properties. Blockchain solutions make use of a consensus algorithm, which guarantees that all nodes in a distributed ledger will reach an agreement and converge.

Potential for incentives or rewards: In a FL setup, it is important to incentivize users that contribute with quality data. This is crucial as the accuracy of the global model is directly proportional to the quality of the training data. Blockchain is the perfect medium to provide incentives in the form of tokens, which can be exchanged for services or financial rewards.

Auditability - Traceability – Accountability: Every time a piece of data on a blockchain network is updated, this is stored as a new transaction in a separate block. Previous blocks cannot be altered (or deleted). In the context of FL, this can be beneficial, especially in applications that require high level of trust on AI decisions such as in the military sector. Since the blocks on the network include the signature of the user initiating the transaction, the user cannot deny the authorship of this transaction. This property known as non-repudiation, can be used for accountability.

1.4 Blockchain technologies in the context of improving industrial applications

In the era of Industry 4.0 and the ongoing digital transformation, Supervisory Control and Data Acquisition systems (SCADA), Human-Machine Interface devices (HMI), and Programmable Logic Controllers (PLCs) play essential roles in reshaping industrial processes and operations. SCADA systems serve as data hubs that aggregate data from various sensors and machines. HMI devices provide the operator with the ability to interact with complex systems and make informed decisions. PLCs are designed to communicate with a wide range of devices and systems. They serve as the "brains" of automation, orchestrating processes and responding to real-time data. PLCs have limitations in terms of computing resources and processing power, which can make them less suitable for heavy computational tasks compared to more powerful computing platforms. They are primarily designed for real-time control and automation in industrial environments, focusing on reliability, determinism, and stability rather than raw computing power.

SCADA, HMI, and PLCs collectively enable data-driven decision-making by providing real-time insights into operational performance, equipment health, and quality metrics and are integral components of the digital transformation in Industry 4.0.

Audit trail logs

Certain industries such as the pharmaceutical, biotechnology, and medical device industries are required to apply regulations such as those described by the Food and Drug Administration (FDA) in 21 CFR Part 11 [19]. This regulation outlines the criteria under which electronic records and signatures are considered trustworthy, reliable, and legally equivalent to paper records and handwritten signatures. An audit trail is defined as: "A secure, computer-generated, time-stamped electronic record that allows reconstruction of the course of events relating to the creation, modification, and deletion of an electronic record" [20]. In more detail, the logging process should record the user's identity, time-stamps and actions. Additionally, the system should be resilient to tampering and preserve the recorded information for compliance.

A typical architecture describing the flow of information within the components of an industrial application is depicted in Figure 3 below.

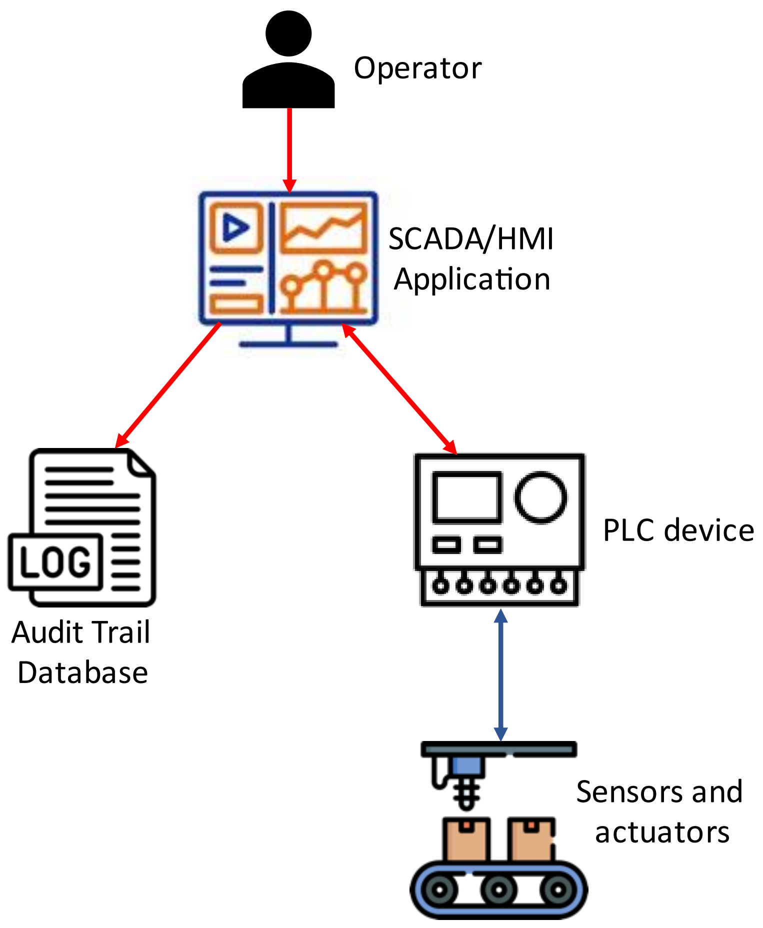


Figure 3 - Typical architecture of industrial applications built upon SCADA/HMI and PLC devices.

Security Considerations

Considering that these systems become more interconnected, cybersecurity is paramount. Industry 4.0 initiatives emphasize robust cybersecurity practices to protect against cyber threats and data breaches.

Many industrial control systems, including SCADA and PLCs, have long lifecycles that can span decades. This means that older hardware and software versions are still in operation, even after newer, more secure alternatives have become available. Some manufacturers discontinue support, focusing their efforts on newer products. As a result, users of these legacy systems are left without access to security patches. In some cases, upgrading to newer SCADA or PLC systems may not even be feasible due to compatibility issues with existing infrastructure, cost constraints, or the need for extensive reconfiguration. Smaller organizations or those with limited resources may struggle to keep pace with technology updates and may continue to use older SCADA and PLC systems. Even when updates are available, organizations may be reluctant to apply them for fear of disrupting critical processes.

In 2010, a highly sophisticated computer worm was discovered that gained international attention for its unprecedented level of complexity and its specific target: industrial control systems, particularly Siemens programmable logic controllers PLCs at the Iranian nuclear program. Once deployed, Stuxnet does not require Internet access for command-and-control. Instead, it is designed to inject rogue code inside PLC devices [21]. Many other exploits compromising legacy SCADA systems have also been well documented [22].

SCADA systems operate by sending data (such as setpoint updates, operator commands) directly to the PLC device and naturally the devices are required to allow such incoming connections. This requirement forms a large attack vector due to the security issues arising from compromised SCADA systems, malware. Because of these security concerns, such deployments usually reside in isolated internal networks without internet connectivity.

Blockchain-based audit trail databases

Blockchain networks make a great medium for audit trail databases. The most compelling reasons are the following:

Compliance: For industries subject to regulatory compliance, blockchain-based audit trail logs can simplify compliance efforts by providing a tamper-resistant and easily auditable record of activities.

Immutability: Once data is recorded on the blockchain, it becomes nearly impossible to alter or delete it. Each new block in the chain contains a cryptographic hash of the previous block, creating a secure and immutable ledger. This feature ensures that audit trail logs remain tamper-proof, providing a reliable historical record of activities.

Security: Blockchain networks employ strong cryptographic techniques to secure data. Public and private keys, digital signatures, and consensus mechanisms help protect the integrity of audit trail logs. Unauthorized access and alterations are exceedingly difficult, if not impossible.

Permanent Records: Audit trail logs on a blockchain can be stored indefinitely. This ensures that historical data is always accessible for compliance, forensic analysis and trend monitoring.

Traceability: Blockchain provides a detailed history of data changes and transactions. Each entry in the audit trail can be traced back to its origin, making it easier to identify the source of any unauthorized actions or anomalies. In the context of audit trails, each action (e.g., setpoint change) can be mapped to a single user.

1.5 Incentive Schemes

In the context of a blockchain assisted FL process, designing effective incentive schemes plays a crucial role in fostering participation and cooperation among competing entities in FL. The utilization of smart contracts within a blockchain network enables the transparent and decentralized recording of each participant's performance metrics. This approach ensures that rewards are allocated fairly and in accordance with predefined criteria, thereby enhancing trust and accountability in the collaborative learning process.

The distributed ledger technology of blockchain, and centralized platforms like Hyperledger Fabric (HLF), support some sort of consensus mechanisms where e.g. a specified percentage of network nodes must agree on transaction validity. By embedding verification algorithms within smart contracts, we can achieve an unprecedented level of transparency. This transparency is especially critical in our context, where algorithmic outputs directly influence incentive calculations, ensuring that rewards are distributed equitably based on measurable contributions.

We believe that incentive schemes are particularly advantageous in environments where trust is paramount, such as collaborative research consortia or industrial alliances engaging in FL. These schemes not only encourage active participation but also incentivize entities to contribute quality data and algorithms, thereby improving the overall performance and accuracy of shared machine learning models.

The deployment strategy for blockchain nodes is adaptable to different FL use cases. In scenarios involving a limited number of participating industries or associations, each entity may opt to host a blockchain node to maintain proximity to their respective data sources and stakeholders. Alternatively, a trusted third-party host may be chosen to oversee the governance and operational integrity of the incentive scheme, ensuring impartiality and adherence to predefined rules.

By integrating robust incentive mechanisms into our FL framework, we aim to create a conducive environment where multiple end users can collaborate effectively, driving innovation and advancing the state-of-the-art in distributed machine learning technologies.

1.6 Research aims and objectives

The primary aim of this thesis is to investigate how Distributed Ledger Technologies and Machine Learning can be integrated to develop innovative solutions that enhance data security, operational efficiency, and collaborative learning in industrial applications.

Firstly, the thesis aims to explore how FL, a decentralized approach to ML model training, can be effectively coordinated using blockchain technology. This includes investigating the mechanisms by which blockchain can facilitate secure and efficient data sharing and model training across multiple entities, ensuring transparency and integrity in the process.

Secondly, the research aims to identify and evaluate the potential industrial applications of DLT and ML, focusing on areas where their synergy can provide significant operational improvements. By examining real-world scenarios, the thesis seeks to demonstrate the practical benefits and address the challenges of implementing these technologies in industrial contexts.

The thesis also aims to optimize user engagement and data contribution quality in FL systems coordinated by blockchain. This involves designing and implementing incentive mechanisms and engagement strategies that encourage high-quality data contributions and active participation from users.

Another aim is to enhance the security and functionality of industrial systems, such as programmable logic controllers (PLCs), through the application of blockchain technologies. This includes developing blockchain-based solutions that can serve as immutable audit trails and command sources for PLCs, reducing their vulnerability to cyber-attacks and improving operational reliability.

The roadmap of this research thesis is outlined below:

1) Develop a framework for the integration of FL and blockchain that ensures secure, transparent, and efficient coordination of model training across multiple entities.
2) Design and implement smart contracts that facilitate the management of FL processes, including data sharing, model aggregation, and reward distribution.
3) Design and implement incentive mechanisms, such as token rewards and reputation systems, to encourage high-quality data contributions in FL systems.
4) Investigate the use of blockchain as an immutable audit trail for programmable logic controllers (PLCs), enhancing their security and traceability.
5) Develop and test a blockchain-based command and control system for PLCs to reduce vulnerability to cyber-attacks and improve operational reliability.

2. State of the art and related work

Researchers are exploring the integration of Federated Learning (FL) and blockchain because it addresses critical challenges in privacy, security, and data integrity, particularly in an era where data is increasingly decentralized and sensitive. Traditional machine learning requires large amounts of centralized data, which can pose risks to privacy and compliance with regulations like GDPR. FL enables training models without sharing raw data, but it is still vulnerable to issues like data poisoning and trust concerns among participants.

Blockchain's decentralized, secure, and transparent nature makes it a natural complement to FL. It ensures the integrity of model updates, prevents tampering, and provides an auditable record of contributions. Additionally, blockchain allows the creation of smart contracts and incentive mechanisms, motivating participants to contribute high-quality data. This combination is particularly valuable in sensitive fields like healthcare, finance, and industrial applications, where security, transparency, and privacy are paramount.

2.1 Blockchain-based Federated Learning approaches

H. Kim et al [23] have proposed a solution (BlockFL), where a blockchain network is used instead of a central server to facilitate sharing of model updates from client devices. The proposed consensus algorithm is based on Proof of Work (PoW). The blocks on the network contain the model updates and miners are used to verify them and add them to the network. The advantages include incentives for devices that contribute to the training process with larger amounts of data, as well as solving the single point of failure in the case of a central server outage. They also study the effects of a miner's malfunction, imposed energy constraints and number of participating devices in respect to end-to-end latency and robustness. This work has been extended to offer rewards to valuable updates with smart contracts, using a Cross-Sampled Validation-Error Scheme (CSVES) [33].

U. Majeed et al [24] have proposed a similar architecture (FLchain) that includes features from Hyperledger Fabric and Ethereum, in which a separate fabric channel is used for each global learning model. The global model state is calculated after each new block generation. The suggested consensus algorithm is a modified version of Practical Byzantine Fault Tolerance (pBFT) and Proof-of-Word (PoW). The main focus in this solution is to improve auditability and governance.

D. Preuveneers et al [25] have implemented FL on a blockchain solution for intrusion detection systems in computer networks, in order to explore auditability and accountability. Their setup relies on a permissioned block-chained network, in order to orchestrate machine learning models using FL. These models are then used to classify traffic for Intrusion Detection. The non-repudiation property of the blockchain network allows for enhanced accountability of contributing parties. The implementation is based on MultiChain, an opensource blockchain platform. The calculated overhead of the proposed solution in relation to traditional FL is estimated between 5%-15%.
More recently, similar research was performed by J. Weng et al [26] who implemented a blockchain assisted FL that focuses on incentive mechanisms and auditability. The setup is implemented on Corda V3.0 (a blockchain network sharing features of Bitcoin and Ethereum) and uses a custom consensus protocol based on the work of Algorand [27]. The learning environment is based on TensorFlow and the results show how the training accuracy increases with more participating parties.
Kang et al [28] have proposed a reputation-based approach that acts as an incentive mechanism in a FL setup. A reputation blockchain network is utilized in order to store weighting reputation opinions from recommenders. Based on these reputation weights and contract theory, an incentive mechanism is designed in order to motivate high reputation workers.
FedCoin [34] has been recently proposed by Liu et al, where a blockchain network is used in order to offer incentives for miners that verify blocks on the network. Specifically, a proprietary consensus protocol is developed based on Shapley Values, in order to promote high quality data from participants, and provide incentives.
Although researchers are focusing on methods to offer incentives to participants [23], [26], [28], [34], these are not calculated based on the actual value of model updates. The work of I. Martinez et al. [33] calculates rewards based on the performance of contributions, however the scheme used (CSVES) offers the same rewards to all users that achieve a performance above a specific threshold. The solution proposed in this paper, is able to adjust the size of the reward based on the improvement that it offers to the joint global model and to our knowledge is unique.
The main contributions of this paper can be summarized as follows: a) It extends the capabilities of our (previous) model update verification algorithm (presented in [35]) in order to provide a metric proportional to the model update contribution and therefore increase fairness in reward allocation. b) it described the mplementation of the

aforementioned verification algorithm in a simulation environment in order to verify feasibility and provide baseline results, based on previous work regarding the specifications of executing a FL process within a smart contract [36].

2.2 Benefits of using blockchain-based audit trails to generate secure PLC commands.

In a recent study, researchers introduced BlockTrail as an innovative blockchain architecture designed to address the space and time complexity challenges inherent in traditional blockchain systems [37], [38]. They employed a hierarchical structure in BlockTrail, utilizing the nature of transactions to partition the system into multiple layers capable of processing transactions simultaneously. This architectural approach aimed to reduce space overhead and accelerate the validation process by minimizing the number of active replicas. The researchers also implemented additional security measures in BlockTrail to strengthen defense capabilities and facilitate the detection of faulty replicas.
Researchers C. Regueiro et al [39] enhanced current audit trail solutions by developing a Blockchain-based mechanism that prioritizes security and usability. They leveraged Blockchain's intrinsic security features, including integrity, traceability, availability, and non-repudiation, to ensure a high level of security in audit trails. To enhance usability, they incorporated a Blockchain monitor, isolating users from the complexities of Blockchain use. The resulting prototype contributes to more reliable, secure, and user-friendly audit trails, with identified improvements over existing methods. The mechanism is designed for general-purpose use, applicable across various ecosystems.
Authors S. Suzuki and J Murai [40] have proposed a scheme using blockchain technology for applications requiring strict access control and auditing, such as medical record queries. Traditional server-side logging is deemed insecure due to potential tampering, leading to the proposal of a blockchain-based request-response channel for client-server systems. A proof-of-concept system is implemented on a public blockchain testbed, demonstrating the viability of using blockchain transactions as an auditable communication channel. The authors suggest broad applicability, including shared key delivery mechanisms, content encryption key delivery, and envision extending the scheme to smart contracts for multi-party conditional schemes and access delegation in healthcare systems.

Researchers W. Pourmajidi and A. Miranskyy [41] address the challenge of log tampering in Cloud solutions by proposing a blockchain-based log system named Logchain. They emphasize the critical nature of logs during incidents and propose immutability through blockchain to ensure the integrity of log data. Logchain, as a Log Chain as a Service (LCaaS), cryptographically seals logs and adds them to a hierarchical ledger, preventing tampering and providing an immutable platform for log storage. The system aims to establish trust among Cloud participants (providers and users) by offering verifiable logs through a hierarchical ledger.

Factom, introduced in a whitepaper by P. Snow et al. [42], addresses the scarcity of trust in the global economy by offering a precise, verifiable, and immutable audit trail, reducing the need for blind trust and enhancing efficiency. Factom introduces a solution by leveraging blockchains to lock in data, providing a distributed and independently auditable mechanism. While Bitcoin's blockchain is a trusted immutable data store, Factom extends blockchain technology to businesses without the complexities associated with cryptocurrencies.

Recent work by P. Schmid et al. [43] addresses security and privacy issues in the Internet of Things (IoT) networks, particularly focusing on Programmable Logic Controller (PLC) systems. Acknowledging the potential of blockchain technology to enhance security, resilience, and trustless authentication in IoT ecosystems, the paper proposes a novel approach. The suggested model integrates a proof-of-work-based blockchain into the PLC IoT ecosystem, facilitating the secure transmission and data logging of binary data from PLC networks. This method aims to address issues like memory, tracing problems, and resource efficiency. The authors highlight the security challenges of integrating blockchain into IoT systems and suggest evaluating alternative consensus mechanisms, managing storage requirements, and exploring integration with emerging technologies like edge computing and machine learning for improved efficiency. They also emphasize the importance of interoperability and standardization efforts for broader adoption.

Researchers A. Vick et al. [44] have explored the integration of blockchain technology, specifically Solana, into Industrial Robot Control systems using a virtual PLC with OPC UA interface. They propose a software gateway connecting Solana Blockchain Cluster and OPC UA server, enabling data exchange between blockchain and industrial equipment. The smart contract deployed on Solana implements control logic, demonstrated by driving an Industrial Robot Handling Process in a laboratory setting. Test results reveal varying runtimes for different steps, influenced by network latency and transaction processing. The cost analysis shows acceptable expenses for implementing a trusted third-party industrial robot control service on Solana. The blockchain ensures end-to-end data security, allowing deployment of third-party control services as smart contracts securely. Additional data security measures are recommended for communication between the gateway client, shop-floor, and automation system operator, especially when not in the same segment or location.

A recent study by S. Loss et al. [45] addresses the regulatory challenges faced by pharmaceutical manufacturing in Brazil, particularly the requirements set by the National Health Surveillance Agency (ANVISA) that mandates that pharmaceutical systems ensure the integrity, security, and traceability of product information to safeguard consumers. The paper proposes a blockchain-based microservice for audit trail management, aiming to automatically record and secure all pharmaceutical system operations, ensuring data immutability. A case study is presented, demonstrating the applicability of the proposed microservice in pharmaceutical systems. The results indicate the microservice's capability to handle 100 to 200 simultaneous users with good throughput, making it suitable for small or medium-sized companies.

3. Design of Federated Learning Systems on blockchain networks

During this part of the research, we introduced an algorithm that evaluates each model update against a known verification dataset before it is aggregated into the global model. The verification process does not rely on the size of the training dataset, which is often unknown or can be falsified by adversaries. Using the MNIST dataset, simulations show that the proposed algorithm effectively identifies and discards malicious updates. The global model maintains high accuracy and convergence even with a significant percentage of adversaries participating.

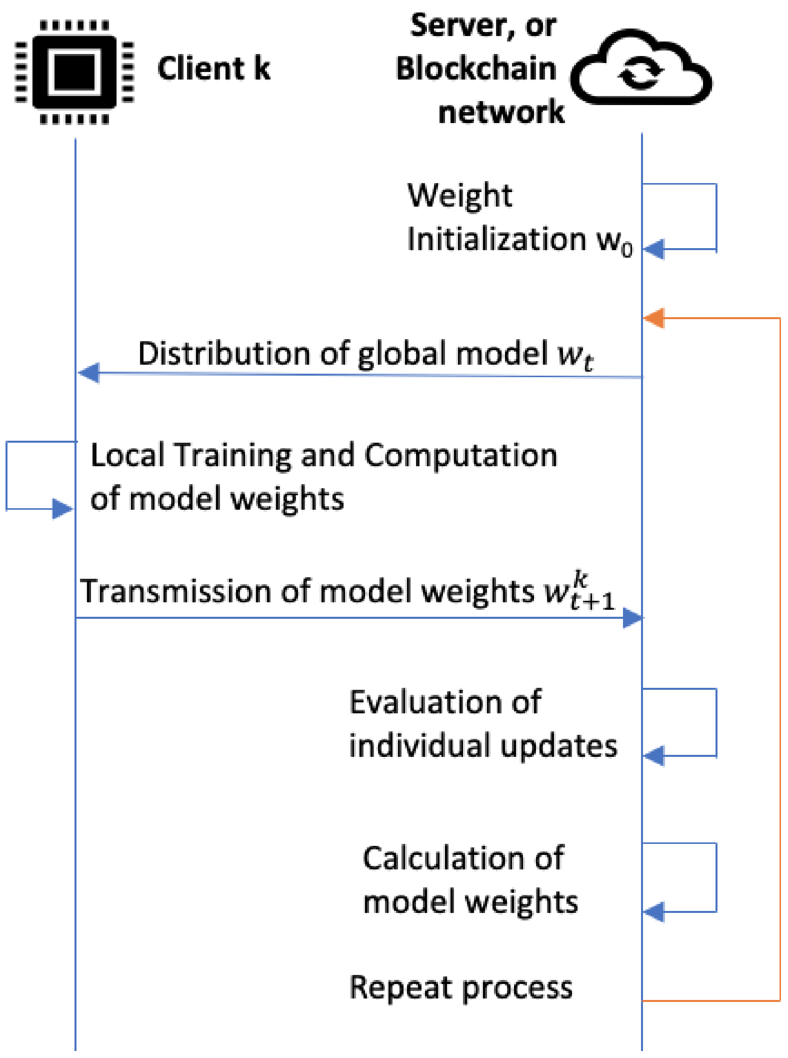
By using this approach, we show that the proposed scheme improves on the following aspects:

- Ensures data integrity and reliability by leveraging the immutability and traceability features of blockchain.
- Provides a decentralized and transparent method to secure FL processes, eliminating the need for a trusted central server.
- Offers potential for incentive mechanisms using blockchain tokens to reward participants for quality contributions.

3.1 Implementation of an algorithm to assess the quality of contributed model updates

In this section, we propose an algorithm for FL, that can be incorporated in a blockchain environment and run inside a smart contract, in order to facilitate the learning process and provide protection against model poisoning. We are also presenting a high-level description of the algorithm and results based on its implementation in open-source tools: Keras and TensorFlow.

The steps involved in the operation of the algorithm are shown in Figure 4. As in the case of traditional FL, the process starts with the initialization of the global model and its relevant weights (with either random values or using proxy data). The global model is then distributed to participating parties. Local training is performed on end devices resulting in the generation of model updates in the form of weights. The coordinator upon receiving the updates, evaluates each update separately against a known good validation data set and records the accuracy. If the accuracy increases, the specific model update is considered reliable. If the accuracy decreases, the update is discarded. A qualitative measure for each update can be derived by measuring the distance between the global model accuracy and the accuracy of the global model when averaged with the update.



Client k
Server, or Blockchain network
Weight Initialization w_0
Distribution of global model w_t
Local Training and Computation of model weights
Transmission of model weights w_{t+1}^k
Evaluation of individual updates
Calculation of model weights
Repeat process

Figure 4 - Steps involved in the proposed algorithm

The pseudocode in Algorithm 1 is a high-level overview of the implementation, based on the FL algorithm and the use of stochastic gradient descent (SGD). During averaging, it intentionally does not take into consideration the clients' data sample size, since in real use cases this would not be known by the server and could be easily falsified by an adversary in order to maximize his effect on the global model. This approach does not measure the quality of each update; instead it only measures if each update increases accuracy in a Boolean logic. Further improvements could be made in order to measure the quality and use this as a metric for an incentive or rewards mechanism.

3.1.1 Algorithm 1 – Discarding of unreliable model updates

Variables used in the algorithm:

- w_0 the weights of the initial model
- λ the learning rate of the global model
- η the learning rate of the local model
- K contains all Clients
- Pk contains all data samples of client k
- B the local batch size
- GetAccuracy a function which returns the accuracy of the specified model against the verification dataset
- r_{t+1}^{k} the received (candidate) update from end-device k to be used in the next model update
- w_{t+1} the weighted average of all verified updates.

```
procedure SERVER
    initialize $w_0$
    for each round t = 1,2,… do
        for each client k in parallel do
            $r_{t+1}^{k} \leftarrow ClientUpdate(w_t)$
            if VerifyUpdate($w_t$,$r_{t+1}^{k}$) then
                $w_{t+1}^{k} = r_{t+1}^{k}$
            end if
        end for
        $w_{t+1} = \lambda w_t + (1-\lambda) \sum_{k=1}^{K} \frac{1}{K} w_{t+1}^{k}$
    end for
end procedure
procedure ClientUpdate($w_t$)
    B ← split Pk to smaller sets
    for all b ∈ B do
        W←w-$\eta\nabla$f($w_t$,b)
    end for
    return W (computed weights obtained by minimizing loss function f($w_t$,b))
end procedure
```

```
procedure VerifyUpdate($w_t$,$r^k_{t+1}$)
    $p_{t+1} = \lambda w_t + (1-\lambda) r^k_{t+1}$
    a = GetAccuracy($p_{t+1}$)
    b = GetAccuracy($w_t$)
    if a > b then
        return True
else
return False
end if
end procedure
```

Algorithm 1 - Discarding of unreliable model updates

The *SERVER* procedure initializes the module weights by either random numbers, or created using proxy data. Then, for each round, it distributes the current model version (w_t) to all clients. After the local training process is finished, the individual candidate model updates r^k_{t+1} are retrieved and verified by the procedure *VerifyUpdate*. In the simplest form, this procedure will return a Boolean value, depending on whether the model update improved the overall model performance against a verification dataset. The new model version w_{t+1} is calculated by averaging all the model updates for which the *VerifyUpdate* procedure returned True. The configurable parameter λ is used to specify the learning rate, affecting the impact of the training round on the global model.

The *ClientUpdate* procedure is responsible for the local training process. First, the data samples Pk are split into smaller batches of size B, which are then used to minimize a loss function f. The reason for choosing a smaller batch size for training is because it speeds up the training process significantly, especially when the end device is low powered such as a mobile phone. The method of splitting a large sample size in smaller batch sizes for training is known as Stochastic Gradient Descend (SGD)[6]. The configurable parameter η, is used to specify the local learning rate, and affects the impact of the training process on the specific model update.

The *VerifyUpdate* procedure is used to compare the performance of the current model w_t, against the performance of the weighted average of w_t with the candidate update r^k_{t+1}. The performance is evaluated against a verification dataset which is chosen before-hand, and remains the same throughout all rounds.

The novelties of the proposed scheme are:

a) we propose an alternative way to evaluate the model updates, i.e. based on the accuracy improvements they bring, which obviates the need to know the dataset size each device possesses which can be falsified,
b) due to the execution of the logic in a smart contract, the logic cannot be compromised,
c) we exploit the traceability capacity of the blockchain to discourage malicious users which could try to poison the model.

When implementing the algorithm inside a smart contract, the following should be considered with respect to the blockchain framework:
a) the smart contract must be able to execute external tools, such as libraries used for model evaluation,
b) all nodes must have access to the same verification dataset and
c) this dataset should be kept private from clients.

The above requirements suggest that a private permissioned blockchain network such as Hyperledger Fabric is a better candidate.

3.1.2 Experimental setup and simulation results

The following simulation uses Tensorflow and Keras to measure the effectiveness of the above algorithm. We perform FL using the popular MNIST database consisting of a database of 60.000 handwritten digit images and 10.000 verification images and corresponding labels. The model consists of an input layer (receives flattened images of size 28x28), a fully connected layer, and an output layer of size 10 which is used as the classifier. The optimizer selected is Adam[46], an adaptive learning rate algorithm that is based on SGD with default base learning rate of 0.001. The simulation assumes 10 participating parties. In each simulation round, a (changing) subset of the nodes are honest and the rest are adversaries. The training images are first split into 10 chunks which are used by the 10 parties respectively. While both honest and adversaries train on their corresponding training images, the honest use the correct training labels while the adversaries use an altered label set. The labels in this malformed set were assigned to a static number. The intention of the adversary in this scenario is to make the global model always predict this number. During simulation, we experimented with additional malformed label sets such as randomly chosen labels; however, the static number approach was found to have greater negative impact on the model accuracy. Each simulation runs for 10 rounds and each local training is performed over 1 epoch. After each round, the model weights are averaged and distributed to the other parties.

Figure 5 - Performance with traditional FL algorithm in the presence of adversariesshows the degradation of accuracy in this FL setup with varying percentage of adversaries when individual updates are not verified (traditional FL). We notice degraded model performance, both in terms of speed (training rounds) and convergence. Specifically, when more than 10% of participants are adversaries, the model accuracy is below 90%, with little or no convergence.

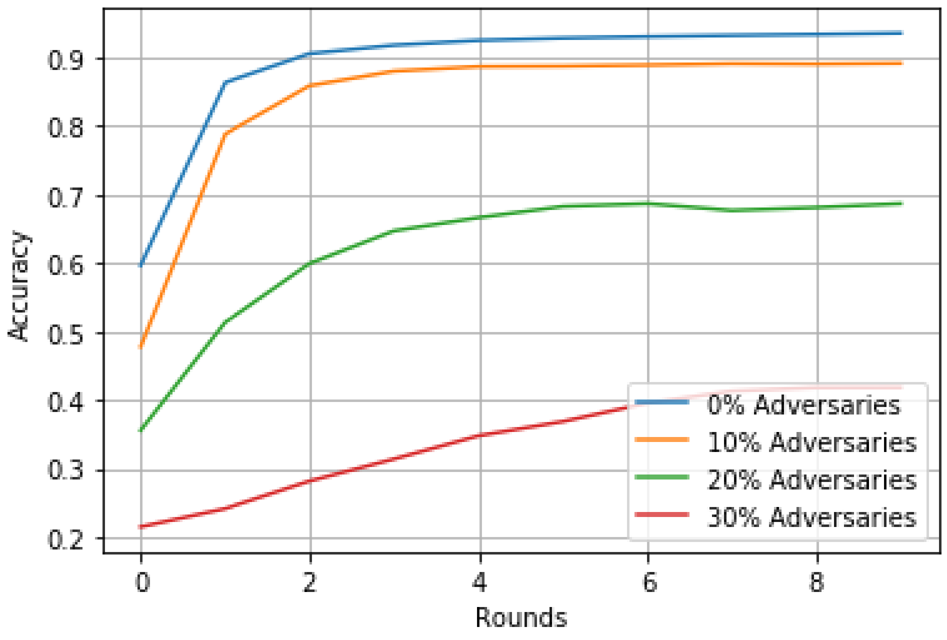


Figure 5 - Performance with traditional FL algorithm in the presence of adversaries

Figure 6 shows how the algorithm has correctly identified the falsified model updates and the model accuracy is able to converge in the presence of 30% adversaries. During the first 2 rounds, the difference in accuracy levels is noticeable and is attributed to the fact that the model is trained with less data (since a considerable amount is discarded). The situation is rectified with each successive round.

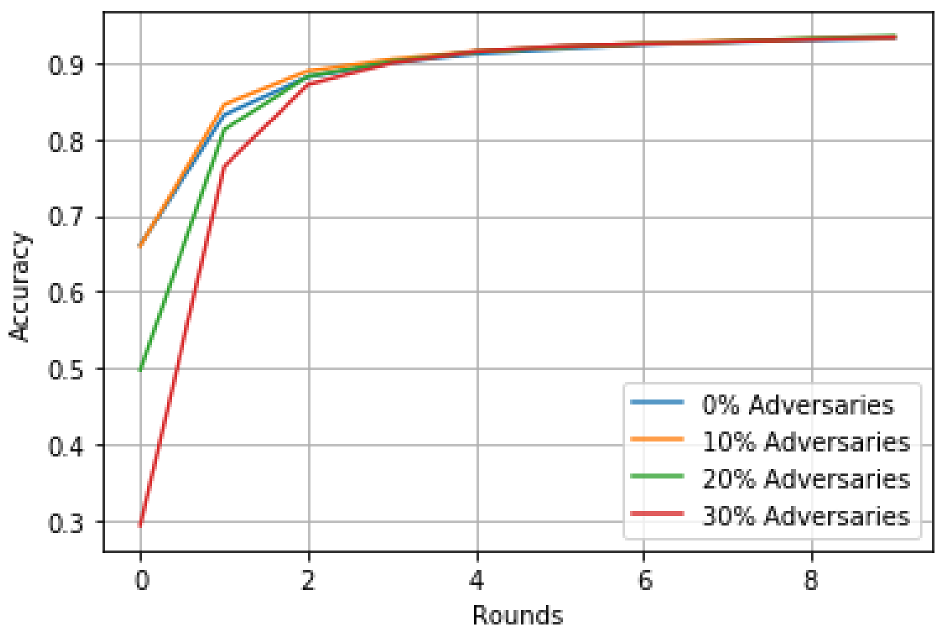


Figure 6 - Performance with model update protection in the presence of adversaries

3.2 Design of a Federated Learning process that is able to execute inside a smart contract

3.2.1 Verification of contributions

The calculation of the next global model version in a FL process typically involves averaging the model weights received from all contributions while also taking into account the number of samples that each client possesses [47]. However, since training in a FL process is performed at the edge, data is not transmitted and is not collected at a central location. It is therefore challenging to assess the honesty of a user's statement regarding the amount of data used during the training process. An attacker may take advantage of this inability, falsely report that he used large data samples, in order to maximize his influence on the global model. It has been demonstrated that a known-good verification data set (kept private from end users) along with a verification algorithm can be used to evaluate individual model contributions, without relying on data sample size numbers, thus providing a level of protection from the aforementioned type of attack [48]. Our proposed solution also uses the verification algorithm in the decision-making process, depending on the incentive scheme used. In the simplest form, we can offer rewards to users, whose contributions were able to pass the verification tests.

3.2.2 Overview of the learning process (Workflow)

Each training round comprises of three main processes. During the first process, participating users request and receive the weights of the current model version. During this step, the smart contract is invoked, which in turn queries the ledger for the relevant data. It should be noted that the ledger is not altered at this stage.

After downloading the latest model version, the end-user is able to use it, as well as to perform local training. The resulting gradients (in the form of model weights) are then sent back to the blockchain network. During this step, a function of the smart contract is invoked, which is responsible for the following tasks:

a) receives the gradients as an input from the end-user,

b) verifies the quality of the update by executing a model update verification algorithm,

c) depending on the results of the verification algorithm, either discard or save user contributions on the ledger.

The last process in the FL workflow is to perform federated aggregation e.g. by averaging all meaningful contributions stored in the ledger and use the result to create the next model version. This process should be triggered to execute at pre-defined intervals depending on the specific use-case e.g. when a certain contribution goal has been reached or when a certain time has elapsed. If the training round is very short, there might not be enough contributions to produce a model with significant improvements. On the other hand, if the training round is very long, users will be delayed in downloading a better-performing version of the model and as a consequence the learning performance will decrease. This last process concludes the training round.

Possibilities for Rewards

After each training round is complete, it is possible to record the performance of each user (in terms of useful contributions) in the ledger. During the training process, user contributions are evaluated against a verification algorithm. The output of this verification function can be used as a metric of users' performance. In the simplest form, the distributed ledger will store increments, that represent the number of successful contributions (i.e. model updates that were deemed useful by the verification algorithm). These scores can be perceived the same way as cryptocurrency tokens, in the sense that they are stored on a secure medium, and can be exchanged for rewards. In a commercial setting, the tokens can be exchanged with money in order to compensate the user for his effort in performing local training and for supplying the FL process with useful model updates. In other use cases, tokens could be exchanged for additional services e.g. access to download and use supplementary Machine Learning (ML) models. Alternatively, the lack of tokens could be perceived as an indication of misbehaving or unproductive users and therefore potentially be used to prohibit them from further participating in the FL process.

3.2.3 Smart Contract Functionality

This section discusses the functionality that should be implemented in the smart contract as a minimum, in order to support the workflow discussed in the previous section.

The first function is responsible for enabling participating users to acquire the latest model version, during the first phase of the training round. The function is defined as GetLatestModelParameters(), queries the ledger and returns a multidimensional table of weights corresponding to the latest model version. This function is invoked once by each participating user. However, since only a simple query is performed against the ledger, we expect that the computational requirements will be low.

Figure 7 below shows interactions between users, smart contract and the distributed ledger during the early stages of the training round (download of model weights).

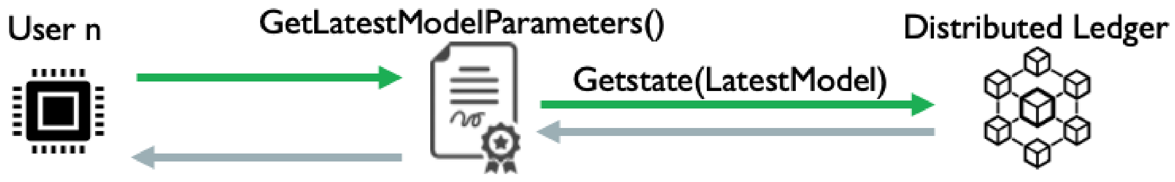


Figure 7 - Interactions between users, smart contract and the distributed ledger – Request of current model version (model weights)

The second smart contract function SubmitModelWeights(weights) is required to facilitate the functions of the second step of the training round (i.e. when a training round is active). When executed, it will receive model updates from the user as an input (in the form of weights), it will then run the verification algorithm, and finally store the contributions to the ledger. We define "weights" in the function above as a multidimensional table of the model weights, corresponding to an individual contribution. This function is also invoked by the user, and can optionally return the contribution assessment result which is derived by the verification algorithm. Two approaches are considered for storing contributions to the ledger. The first approach involves only storing successful contributions i.e. those that improve the accuracy of the model. Another option is to store all contributions along with a metric (derived by the verification algorithm). Depending on the specific use case, we may choose the first approach because it offers performance benefits (since the ledger is being written less often), or the second method which offers higher auditability. In both of these cases, it is anticipated that a moderate amount of computational resources will be required, since the function is called by each participating user, calls external CPU-intensive ML verification libraries, and is expected to affect the state of the ledger.

Figure 8 below shows interactions between users, smart contract and the distributed ledger during the submission of model updates from the users.

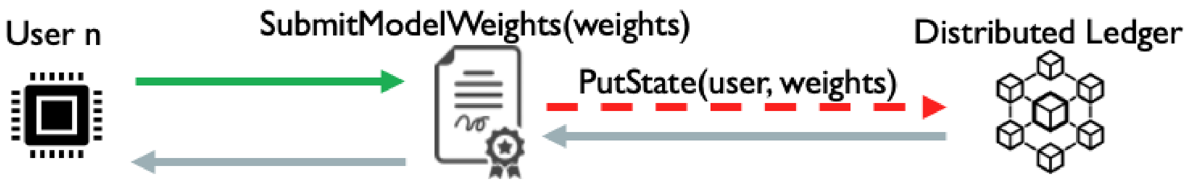


Figure 8 - Interactions between users, smart contract and the distributed ledger – Submission of model weights (user contributions)

The last function CalculateNextModelVersion() is required in order to create the next model version and by doing so, end the current training round. Unlike the previous two functions, it is not invoked by an end user. For this reason, an external trigger should be created as discussed in the previous section. The next model version is created by performing federated averaging among contributions saved in the ledger. The resulted multidimensional matrix is stored in the ledger in order for it to be used during the next training round. Although this function writes to the ledger, it is called only once every training round. Along with the fact that it uses simple mathematical matrix calculations, it is not anticipated to require many computational resources.

Figure 9 below illustrates the interaction between the smart contract and the distributed ledger during the creation of the next model version.

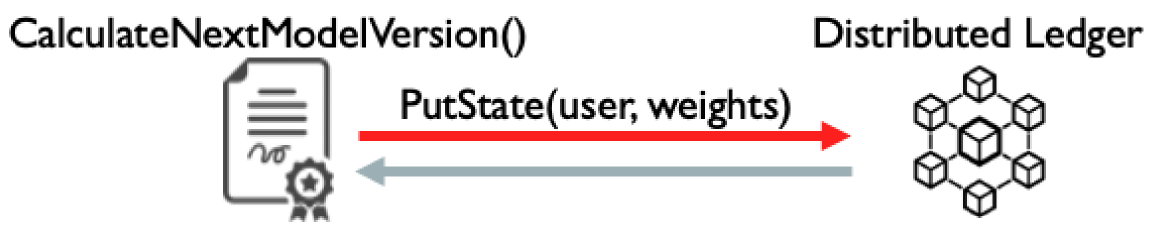


Figure 9 - Interactions between smart contract and the distributed ledger for the creation of the next model version (end of training round)

Table 1 below summarizes the characteristics of the smart contract functions, which are used to predict resources requirements. TR denotes the Training Round.

Resources Requirements for Functions

Function	Attributes		
	Execution Frequency	Alters the ledger	Anticipation of resources required
GetLatestModelParameters	once per user per TR*	NO	LOW
SubmitModelWeights	once per user per TR*	YES	HIGH
CalculateNextModelVersion	once per TR*	YES	LOW

*Training Round

Table 1 - Resources Requirements for Functions

3.2.4 Distributed Ledger operations and data structure

The proposed architectural diagram is based on the concept of key-value pair databases. Such type is used as a state database by HyperLedger Fabric (HLF). Smart contract operations are used to read or write to the current state of the database. Each ledger transaction will either create, update or delete (and therefore modify) key-value pairs in the ledger. However, since HLF belongs to the blockchain family, an immutable sequence of blocks contains all transactions that have affected the current state of the database. The copy of the ledger is propagated to all network members.

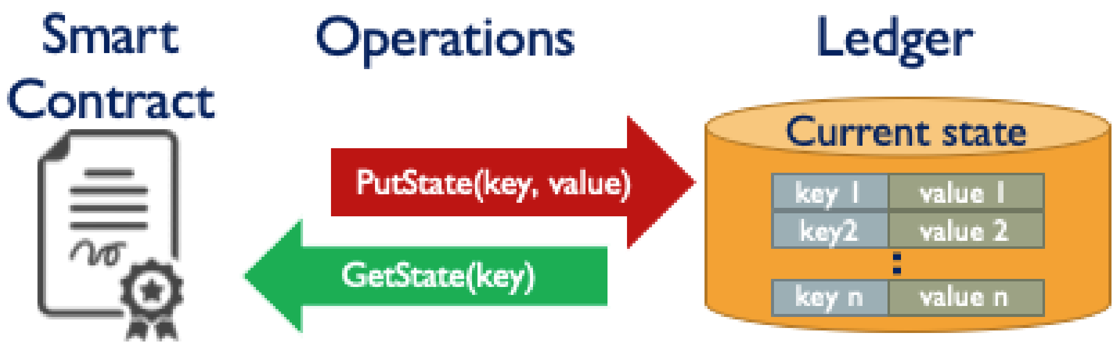


Figure 10 - Operations that access/modify the distributed ledger

Table 2 below summarizes the key-value pairs that will be required as a minimum for the operation of the FL process as well as for recording rewards. The key “identity” below refers to array values. For example, we will provision one key-value pair of *ModelUpdate* and *Reward* variables for each participating user.

Key	***Value***
ModelWeights	Multidimensional matrix containing weights for current model version
ModelUpdate(identity)	Multidimensional matrix containing weights (one for each [identity])
Reward(identity)	Value proportional to quantity/quality of user contribution (one for each [identity])

Table 2 - Key-value pairs for the FL process

3.3 Improving Security and Fairness in Federated Learning Systems

3.3.1 The rewarding algorithm

Building upon previous work [35], [36], we design a rewarding algorithm in order to provide incentives and therefore promote participants with higher quality data. Although our motivation is in line with related work analyzed in section II, we approach this directly from the standpoint of a smart contract so that consensus will be required among multiple nodes. The benefits are twofold: first, the security aspect of the verification algorithm is enforced by all blockchain nodes and trust of the verification algorithm decision is enhanced due to the distributed nature of the architecture. In more detail, we anticipate that the proposed solution will offer advantages in the following areas:

Security enhancements: It is possible that a threat actor may be actively pursuing to degrade the performance of the global model, by either training with malformed data, or transmitting malicious model updates. The solution proposed assumes that a known-good verification dataset is readily available on each blockchain node. Then, we can compute the effectiveness of each model update in terms of accuracy against the verification dataset and conclude weather the individual contribution should be taken into account. We demonstrate that the approach is highly effective against label-flipping attacks, and we intend to assess its performance against other types of attacks (such as backdoor-based attacks or data-poisoning).

Reward calculation on multiple nodes: We propose that the reward calculation is executed inside a smart contact. These will, depending on the blockchain solution used, most likely be run on multiple nodes which will need to reach consensus regarding the reward calculation. Since the rewards will represent the effort of each individual in a FL system, and in some use-cases might be exchanged for money or other services, it is imperative for a user to rest assured that no single network node can alter the ledger where the rewards are kept (either intentionally or due to an error).

Rewards proportional to quality of contributions: Since the parameter server in FL does not have access to the actual data, there is no practical and secure way to acquire or assess the dataset size. In real use cases we cannot assume that a user is honest and will provide the server with a number representing the real amount of data owned, as assumed in other works. Moreover, larger training datasets do not necessarily lead to better models. Instead, we measure the efficiency of each model update by comparing the accuracy of the resulting model against a verification dataset which is used for reference, in order to measure the improvement of the global model performance. The main objective of the rewarding algorithm is to offer fair rewards to participating entities of FL, i.e., analogous to the quality of their model updates.

Principle of Operation

The steps in the federation process and the operation of the algorithm are outlined in Figure 11. The training is performed in rounds. At the beginning of each training round, the current model weights are distributed among all participants. As a result, the participants are able to use the model and perform training using locally available data. During a predefined time, participants are expected to share the derived model weights with the blockchain network.

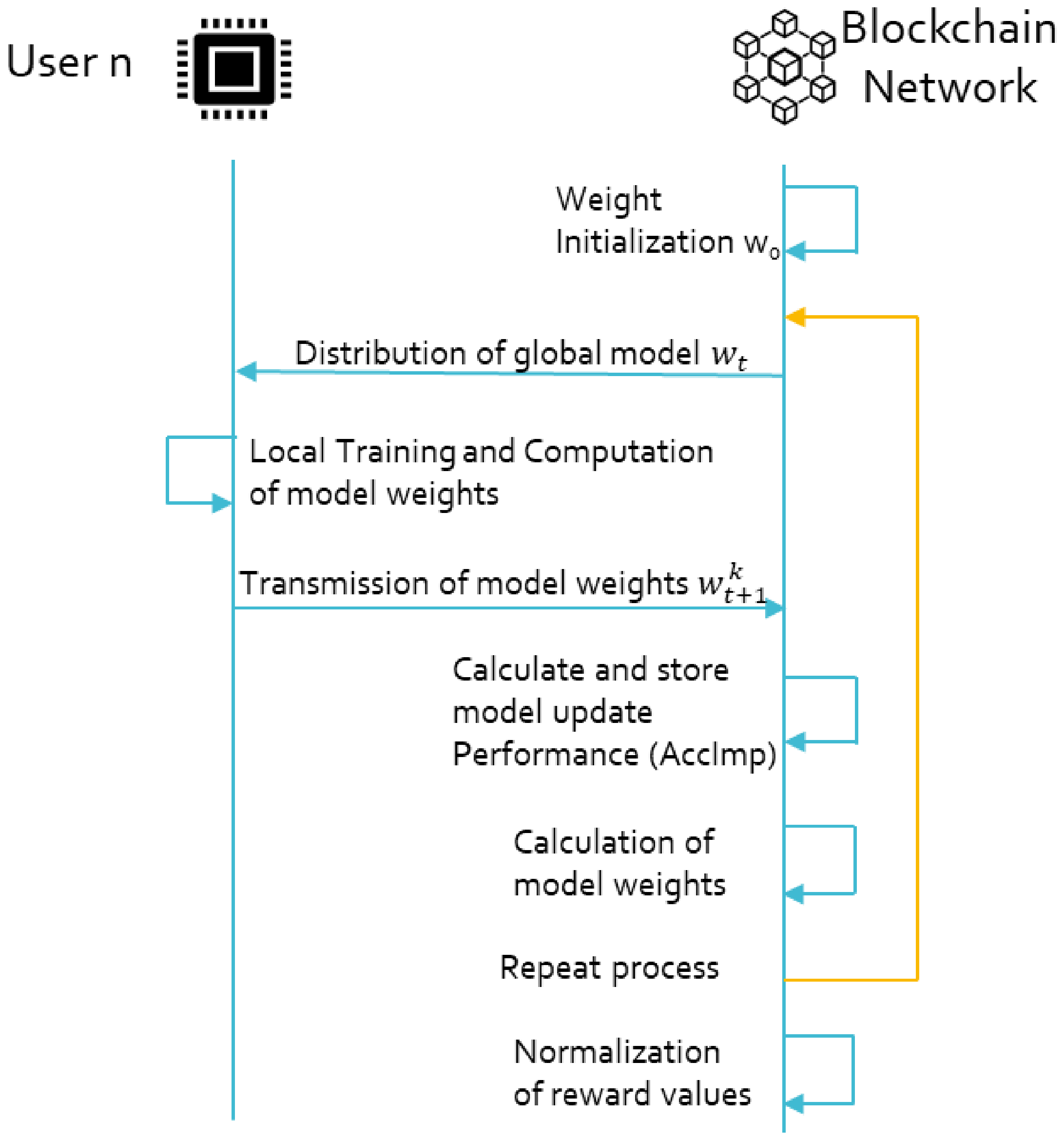

User n
Blockchain Network
Weight Initialization w_0
Distribution of global model w_t
Local Training and Computation of model weights
Transmission of model weights w_{t+1}^k
Calculate and store model update Performance (AccImp)
Calculation of model weights
Repeat process
Normalization of reward values

Figure 11 - Rewarding algorithm, principle of operation

At this stage, all model updates are verified individually in order to calculate rewards. The main steps involved in the process are

a) the model weights are fused with the global model weights,

b) the derived model weights are evaluated against a verification data set and the difference of accuracy is recorded for reward calculation,

c) if the accuracy increases, the specific model update is saved, otherwise it is discarded. Saved contributions are used during global averaging, a process that prepares the model weights for the next global model version which in turn is redistributed in the next training round. This process is repeated for all subsequent training rounds, until a predefined condition is met (e.g., a number of training rounds has elapsed, or the global model accuracy is above a threshold). At the end of the learning process, the metrics stored for each user are normalized, in order to aid reward allocation.

The Algorithm 1 (shown in the sequel) depicts a high-level representation of the FL process using stochastic gradient descent (SGD). Before the training rounds start, the global model weights w_0 as well as the table holding user rewards R, are initialized. During each training round t, the current model weights w_t are distributed to all participating users k, that perform local training, resulting in individual model updates r_{t+1}^{k}. By using a learning rate λ, the individual model weights are averaged with the global model, and form temporary model weights p_{t+1}. The accuracy of this temporary model is evaluated (using AccImp) against the accuracy of the global model w_t and the difference is integrated in table R. Federated Averaging (FedAvg) is then performed, and the model weights $w_{t+1} =$ are prepared for the next training round. Once all training rounds are completed, the rewards table R is normalized.

Algorithm 2 - Reward Calculation inside the FL process

w_0 is the initialized model weights, λ is the learning rate of the global model, η is the learning rate of the local model, K contains all Clients, Pk contains all data samples of user k, R_k holds reward points for user k, B is the local batch size and GetAccuracy is a function which returns the accuracy of the specified model against the verification dataset.

```
procedure Server
initialize $w_0$
initialize R
for each round t = 1,2,… do
    for each client k in parallel do
        $r_{t+1}^{k} \leftarrow ClientUpdate(w_t)$
        $R_{t+1}^{k}$= AccImp($w_t$,$r_{t+1}^{k}$)
        If $R_{t+1}^{k}$> 0 then
            $w_{t+1}^{k} = r_{t+1}^{k}$
        end if
        $R^k = R^k + R_{t+1}^{k}$
    end for
                $w_{t+1} = \lambda w_t + (1-\lambda)\sum_{k=1}^{K}\frac{1}{K}w_{t+1}^{k}$
end for
normalize(R)
end procedure

procedure ClientUpdate(w)
    B ← split Pk to smaller sets
    for all b ∈ B do
        W←w-η∇f(w,b)
    end for
    return W
end procedure
procedure AccImp($w_t$,$r_{t+1}^{k}$)
    $p_{t+1} = \lambda w_t + (1-\lambda) r_{t+1}^{k}$
    a = GetAccuracy($p_{t+1}$)
    b = GetAccuracy($w_t$)
return a-b
end procedure
```

Algorithm 2 - Reward Calculation inside the FL process

Normalization is performed in order to differentiate between user contributions. During training, it is possible that the reward points collected by a user is a negative value. This occurs when the submitted model updates were determined to affect the global model in a negative way. For testing purposes, negative values are first zeroed out. Then, the following formula is used on all values:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

3.3.2 Smart contract specifications

In the blockchain part of the proposed solution, the users could either be user owned devices (such as mobile phones), machines operating autonomously (for example in industries), or data providers meaning that these could be organizations that are willing to make use of their data in order to assist the training process. The blockchain network further comprises of the smart contract and distributed ledger.

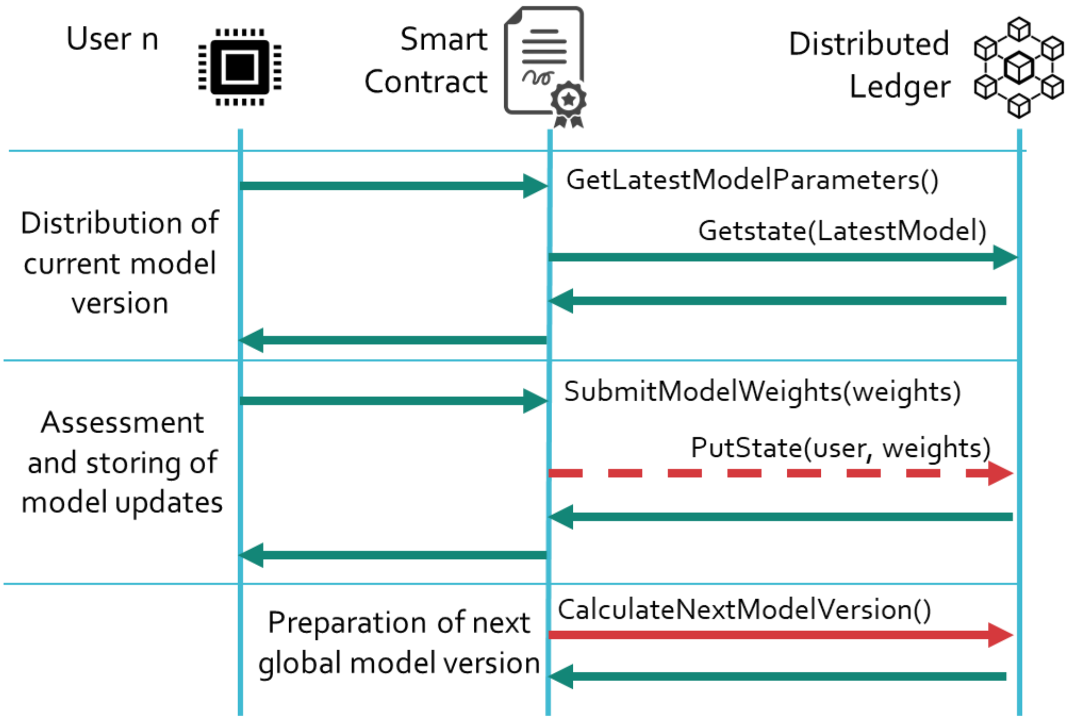

User n
Smart Contract
Distributed Ledger
Distribution of current model version
GetLatestModelParameters()
Getstate(LatestModel)
Assessment and storing of model updates
SubmitModelWeights(weights)
PutState(user, weights)
Preparation of next global model version
CalculateNextModelVersion()

Figure 12 - Blockchain FL, relationships between users, the smart contract and the distributed ledger

Furthermore, Figure 12 depicts in further detail the relationships between users, the smart contract and the distributed ledger, in order to implement the three main processes that occur in a training round when FL is executed inside a blockchain solution. In order to support these main processes, the following three smart contract functions have been considered accordingly. When describing the smart contract functions below as well as the structure of the distributed ledger, we might use some terminology specific to HyperLedger Fabric (opensource private blockchain solution). However, these are included for completeness and should be substituted if implemented on different blockchain technologies.

The first function enables users to asynchronously ask for and retrieve the current model version. Upon being triggered by the user, the function GetLatestModelParameters() queries the distributed ledger for the weights of the current model version which are in turn returned to the user. It is expected that this function will be executed once for every participating user. Since only a query operation is performed against the distributed ledger, the operation is not expected to require many resources.

The next function is responsible for evaluating and storing verified model updates. The function SubmitModelWeights() is once again triggered by the user, as soon as local training is complete. The smart contract at this point invokes the verification algorithm and receives the result. Depending on the result, the smart contract will need to either store the individual contribution to the ledger or discard it. Failed contributions can also be recorded in the ledger, even if they are not used in the next model version, for use cases requiring increased auditability and accountability. This will, however, incur performance costs, as changes made to the ledger will need to be propagated and agreed upon all network nodes.

The last function is required in order to end the training round and prepare the model weights for the next round. Rather than being triggered by a user, the function CalculateNextModelVersion() is triggered by an event. Events are predefined criteria such as when a certain number of contributions is reached, or a predefined time has elapsed or a combination of the two conditions. When the function is triggered, the ledger is queried, and all useful contributions are retrieved. Federated Averaging [49] is then performed and the resulting weights are stored in the distributed ledger. The process of federated averaging is executed once per training round, and therefore it is not anticipated that it will induce a large overhead.

In order to support the aforementioned functionality, some values need to be stored in the ledger. For this reason, we assume a key-value pair database which is common in many private and public blockchain network solutions and we describe its structure. In this type of database, each value is referred to by a unique key (Figure 13). The first key value pair is required in order to store the weights of the current model version. It is a multi-dimensional array, is preset before the first training round starts, and is updated at the end

of each training round. Specifically, it is referenced by the GetLatestModelParameters() function and updated when CalculateNextModelVersion() is invoked.
Another key value pair is defined in order to store contributions from participants. Since the number of received contributions is dynamic, we define it as an array. This array does not need to be persistent across training rounds. It can either be deleted or preserved according to specific use case requirement. In order to calculate user ratings, we need another array which will keep a count of all contributions per user or a value that is proportional to the quality of each user's contributions. This key can be used in order to offer rewards, e.g., in the form of tokens to be exchanged for services.

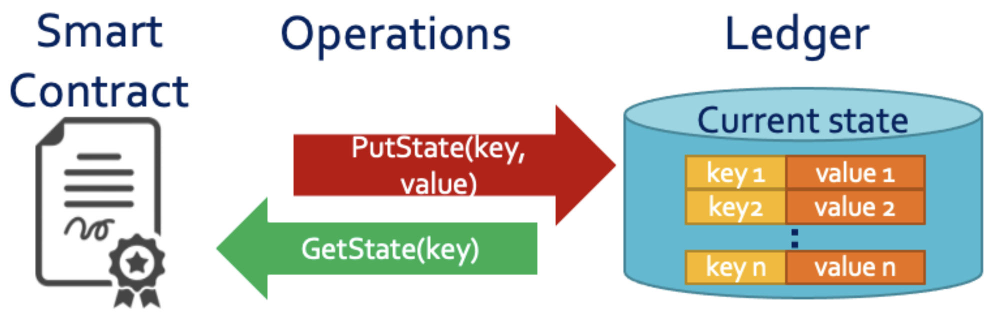


Figure 13- Distributed Ledger operations

For some use cases, it might be desirable to limit access to individual functions beforehand, so only specific users have access. This will most likely increase security and can apply to both private and public blockchain technologies. However, this implies that a central administrating authority will be required in order to select the identities that are able to participate. The following authorization levels are supported:

- **No Access** – The user is not able to access any smart contract function, and therefore cannot use the global model nor participate with model updates.
- **Model use only** – The user is granted access to GetLatestModelParameters() function, but is denied access to SubmitModelWeights(). Therefore, he is able to download the weight of the current model version and use it offline. It is not possible for the user to contribute with model updates.
- **Full Access** – Access is allowed to all functions and therefore the user is able to use and improve the shared global mode.

It is also possible for the authorizations to be assigned dynamically depending on varying attributes. One of these could be user performance. In this scenario, the verification algorithm may be used in order to keep count of the contributions of each user thereby creating a rating. Subsequently, the rating can be used in order to maintain access to the global shared model. For instance, a user may initially have full access, but because he has not participated in the last predefined number of training rounds, he may have his access revoked. As a consequence, he will lose access to new versions of the global model. In other words, this can support use cases where a user is permitted to use the global model as long as he makes meaningful contributions. In a commercial environment, access can be granted on a subscription basis in exchange for money.

DISTRIBUTED LEDGER STRUCTURE

Key	Value
ModelWeights	Multidimensional matrix containing weights for current model version
ModelUpdate(identity)	Multidimensional matrix containing weights (one for each [identity])
Reward(identity)	Value proportional to quantity/quality of user contribution (one for each [identity])

Table 3 - Distributed Ledger Structure

3.3.3 Testbed

This section includes a description of the tools used for simulating the FL process and collecting measurements. In order to simulate the FL process, we developed scripts in Python and made use of opensource frameworks and libraries such as TensorFlow and Keras, as well as publicly available reference datasets from the MNIST database. Before the process starts, the model is setup using the opensource Keras libraries with random values. Model initialization needs to be performed only once, before the first training round, however, the next steps described here are repeated for each subsequent training round. The MNIST database consists of 70,000 images of handwritten digits. The images are 28 by 28 pixels in size and are grayscale. The images belong to 10 classes representing the ten number 0-9 and the set is usually divided into smaller sets for training and validating. This database is publicly available and used by machine learning developers and researchers in order to record baseline performance.

The developed scripts simulate the FL process and are used to collect the measurements presented in this paper. They also handle necessary functions such as model initialization, model training and the model update aggregation. The model used throughout this paper is initialized using Keras, has an input layer of 28x28 in order to receive the handwritten digit images, a dense layer of size 128 with ReLU (rectified linear unit) activation and an output layer of size 10, used as the classifier.

We first concentrate on the defense against the malicious nodes. For this reason, during this specific simulation, a FL process (against the MNIST database) was first run for a duration of 10 training rounds and a varying percentage of malicious participants (following similar strategies applied in other contexts [50]. Figure 14 and Figure 15 show how the global model performs in the presence malicious participants when protections is disabled and enabled respectively. Not surprisingly, when malicious updates are taken into account during model aggregation (Figure 14), the rate of accuracy improvement decreases. Most importantly, in the presence of 20% or more adversarial activity, the model accuracy is not able to converge. This can be attributed to the fact that in label-flipping attacks, the attacker is actively training the model with incorrect labels i.e., aiming to change the direction of the global model predictions.

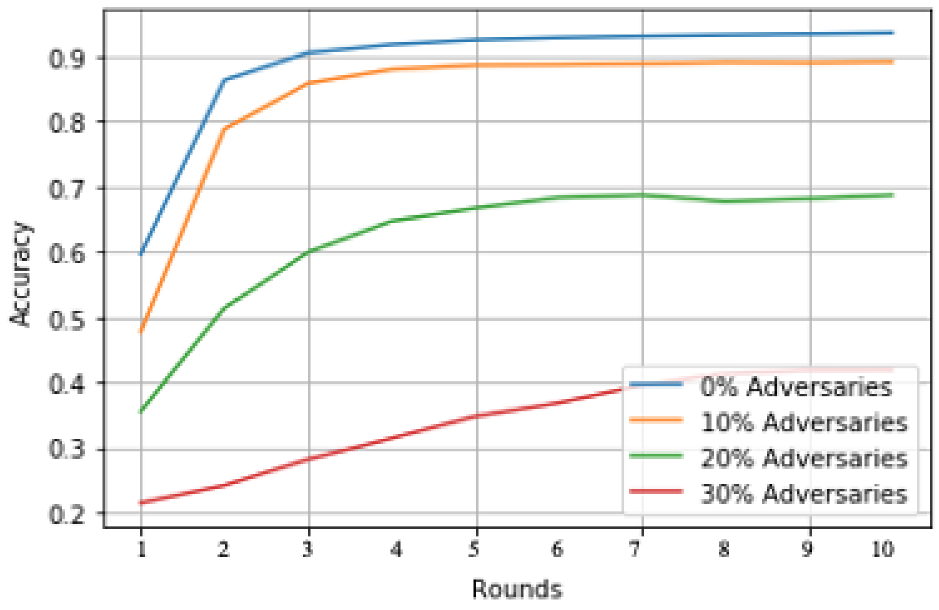


Figure 14 - Performance with traditional FL algorithm in the presence of adversaries

When the verification algorithm is configured to discard useless model updates (Figure 15) we can observe that it is able to differentiate between useful/harmful model updates and offer a substantial protection against this type of attack.

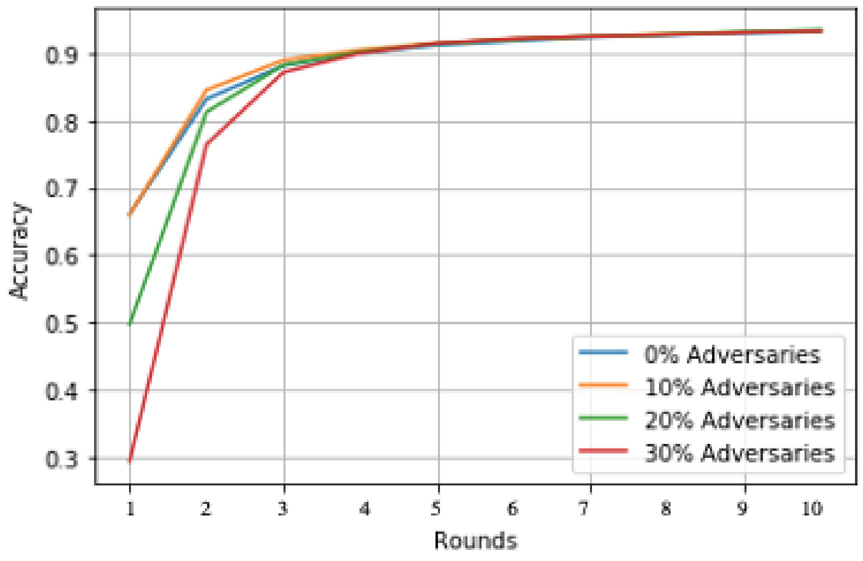


Figure 15 - Performance with model update protection in the presence of adversaries.

Then, we turn our attention of the evaluation of rewards analogous to the model improvement brought by each entity. A simulation of Algorithm 2 has been run using the tools described earlier in this section. In this particular setup, 10 participants participate in the FL process which lasts for 10 training rounds. For training and validation data, we use the popular MNIST database of handwritten images. The training data is further split into smaller chunks, which are allocated to the participants. In order to simulate varying levels of data quality, we create distorted images for training data (Figure 16). Participant 1 receives unmodified training images, whereas participants 2-7 receive images with varying levels of gaussian noise. The variance of the noise ranges between 0.45 and 1.2 and a mean of 0. Participants 8-10 perform label-flipping attack and are actively trying to degrade the performance of the model.

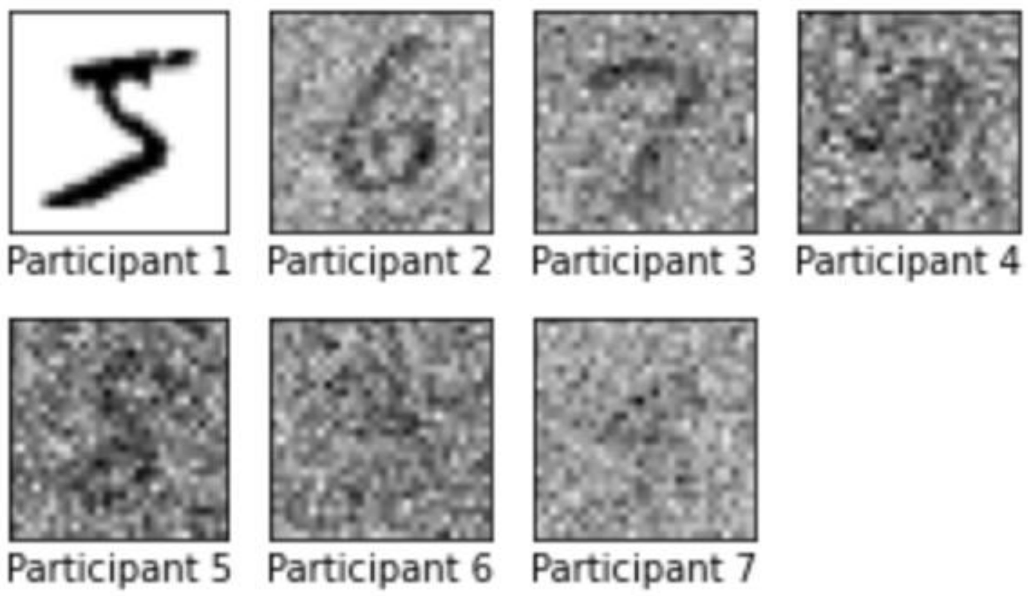


Figure 16 - Training images with gaussian noise applied

Figure 17 depicts the difference in accuracy between the evaluation model (temporary model p_{t+1} in Algorithm 2) and the global model i.e., the result of function AccImp. Even in this simulation with mixed types of participants, the accuracy of the resulting model is still able to reach more that 90% accuracy. In the graph, we can distinguish two different trends. The higher ones show the performance of the honest participants, while the lower ones (overlapping each other) show the performance of the threat actors (participants 8-10).

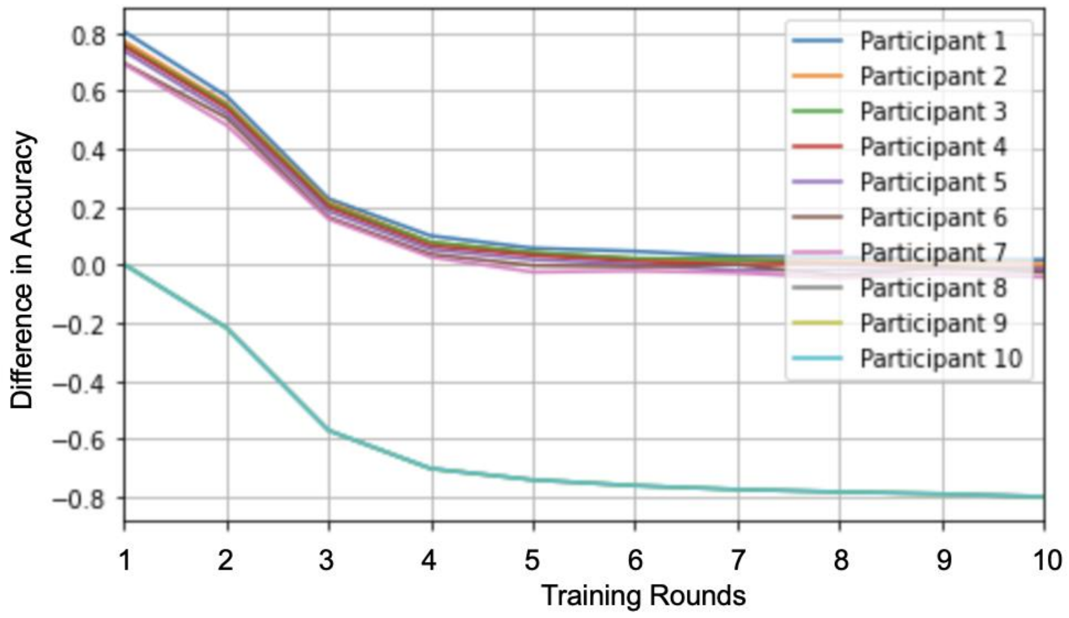


Figure 17 - Difference in accuracy of each contribution

In more detail, we can also extract the following information:

We can observe that the accuracy gains (in terms of accuracy difference against the validation dataset) is high for participants 1-7 during the first training rounds and tend to level off during the end. This curve matches the inverse typical curve of machine learning, i.e., accuracy gains are higher when a machine learning model is untrained and tend to decrease when the performance of the model is higher.

Even participants that used very noisy data (such as 6 and 7), contribute positively to the FL process as evident by the high performance shown in the figure during at least the first 4 rounds. In further rounds, and as the model has become more mature, the model updates of the least performing participants (e.g., participants 6-7) become redundant, the accuracy improvement becomes negative (in terms of the AccImp function) and, as a consequence, are discarded during model aggregation.

Participants with the healthiest data (such as participants 1-2), contributed positively until the last training round. Their contributions improved the accuracy of the global model in respect to the verification dataset, which was correctly identified by the verification algorithm (AccImp function returned positive numbers) and as a consequence all model updates of these participants were used during model aggregation.

Participants 8-10 on the other hand only have zero and negative values. This is attributed to the fact that their model updates test badly against the verification set. So as the global model accuracy improves, the difference in accuracy (AccImp function) becomes negative.

The performance difference between participants 1-7 is not readily distinguishable in the above figure. For this precise reason, we need to perform normalization of all values when the training process is finished.

Figure 18 depicts the points accumulated by each participant, after the values have been normalized on a scale of 0-1. The best performing participant (participant 1) is awarded the highest value 1, while the worst performing participant (participant 7) is assigned the lowest value 0. Participants 2-6 have values in-between, proportional to their contribution. What is interesting here, is that these results correctly correlate to the quality of the training data used. Conveniently, adversaries have been correctly identified (result of AccImp function is negative) and have received no rewards. The malicious model updates were not used during the generation of the global model since this would require the satisfaction of the condition ($R_{t+1}^{k} > 0$). Such a fair reward scheme is necessary both for consumer owned devices and industrial organizations owned devices where the current challenge is to collaboratively train digital twin models of the industrial processes (as discussed in [51]).

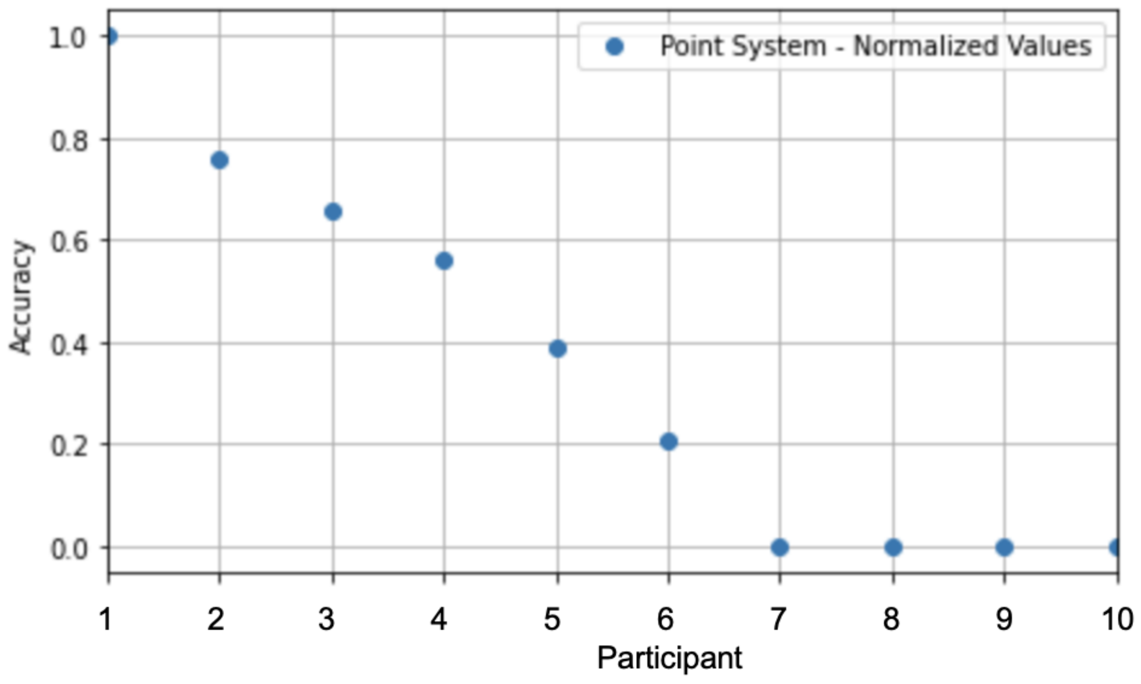


Figure 18 - Normalized values, used for rewards

The initial results demonstrate the effectiveness of the blockchain-based scheme in protecting against model poisoning. Future work includes extensive testing and implementation in open-source blockchain platforms, as well as exploring reward mechanisms based on the quality of contributions.

4. PLCBlox, a blockchain based secure source of commands for PLC devices

Motivated by the fact that industrial applications (machines and processes) generate large amount of data which are potentially valuable for training ML models, the research focused on ways to interconnect these with blockchain assisted FL processes such as those described in chapter 3. It quickly became apparent that an interface would need to be developed in order to allow communication between industrial devices and blockchain networks.

This section introduces PLCBlox, a novel integration of Programmable Logic Controllers (PLCs) with blockchain technology aimed at enhancing security and audit trail capabilities in industrial applications. It begins by contextualizing the roles of SCADA systems, HMI devices, and PLCs in Industry 4.0, emphasizing their critical functions in data aggregation, operator interaction, and real-time control. Security challenges, including the historical Stuxnet worm incident, underscore the vulnerabilities of legacy SCADA and PLC systems, necessitating robust cybersecurity measures.

The proposed PLCBlox scheme innovates by leveraging blockchain networks as secure audit trail databases. Blockchain's immutability and cryptographic security make it ideal for recording and verifying transactional data, ensuring compliance with regulatory standards such as FDA's 21 CFR Part 11. Key benefits outlined include enhanced data integrity, traceability, and resistance to tampering, which are crucial for industries like pharmaceuticals and biotechnology.

PLCBlox operates by enabling PLCs to interact directly with blockchain nodes via smart contracts. This approach decentralizes user authentication and authorization, traditionally managed by SCADA systems, to enhance security. The integration ensures that all changes to PLC parameters and commands are logged immutably on the blockchain, forming a transparent and auditable record.

The research outlines a prototype implementation using Siemens PLCs and an Ethereum-based blockchain network (Ganache), demonstrating feasibility and performance metrics. It addresses resource constraints of PLC devices and evaluates latency and throughput in a simulated industrial environment. Results indicate that PLCBlox effectively minimizes attack surfaces and operational risks associated with traditional SCADA architectures, providing a foundation for further research and development.

We believe that PLCBlox represents a significant advancement in industrial cybersecurity, offering enhanced security, auditability, and compliance capabilities through blockchain integration.

4.1 Principle of operation

The solution presented in this section builds upon the related work presented in the previous section in order to enable PLC devices to query blockchain-based audit trail database as a source for parameters and commands. This approach offers improvements to typical HMI and SCADA applications and to our knowledge has not been already researched.

For reading and displaying process data (information flow from CPU to HMI) the procedure remains unchanged i.e., the CPU is read by the HMI device using vendor specific libraries. In order to update values (parameters or commands) in the CPU area, the user (operator) must first commit a transaction on the blockchain network using his wallet. The CPU asynchronously queries a blockchain node for sets of parameters and setpoints. When changes are identified, they are applied inside the device user memory. The complete workflow is depicted in Figure 19 below.

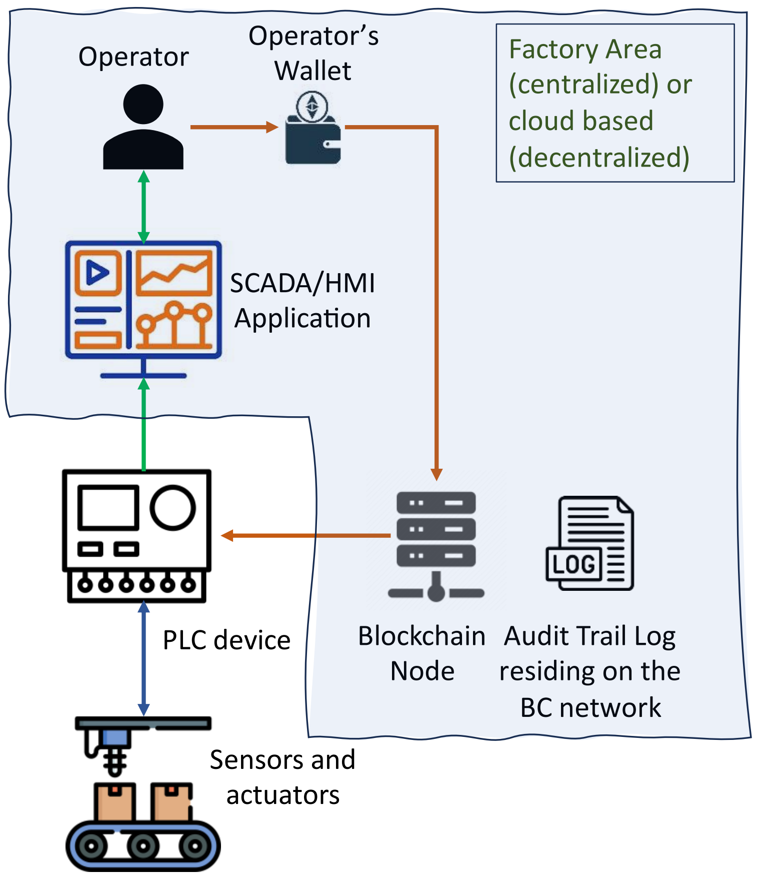

Operator
Operator's Wallet
Factory Area (centralized) or cloud based (decentralized)
SCADA/HMI Application
PLC device
Blockchain Node
LOG
Audit Trail Log residing on the BC network
Sensors and actuators

Figure 19 - Flow of information within PLCBlox

In contrary to traditional HMI systems, the user authentication and authorization tasks are now coordinated by the blockchain network, the smart contract and the user's wallet. Moreover, by leveraging the underlying technology of blockchain, the aspects of the audit trail database are self-contained in the smart contract's storage area. These concepts are summarized in Table 4.

Function	**Typical Architecures**	**PLCBlox**
User Authentication	Local Windows Accounts, Domain Controller Accounts, Accounts configured in SCADA application	User's wallet
User Authorization	Local Windows Groups, Domain Controller Groups, Groups configured in SCADA application	Smart contract

Audit logs	Audit trail database	Blockchain
Source of device (PLC) commands and setpoints	Commands and setpoints/parameters are sent from the SCADA/HMI application to the PLC.	The PLC device queries the blockchain for commands and new setpoints/parameters.
Required permissions on PLC device	Read/Write	Read only

Table 4 - Key differences between typical industrial application architectures and PLCBlox

Improvements

In addition to improvements in terms of scalability and secure audit logs already demonstrated by researchers in the related work section, PLCBlox by itself offers advantages in the following areas:

Security improvements on the PLC device

PLCBlox does not require write access to the PLC device. Updates of command statuses and setpoints are initiated by the PLC device during the querying operation of the blockchain node. It is therefore possible to completely disable write access at the device level. This fact alone leads to a smaller attack surface and renders many known and zero-day attacks unusable [21], [52].

Security improvements at the application layer (SCADA/HMI)

An attack on the device hosting the user application could circumvent the user authentication and authorization checks, allowing anyone to control the end device. Even further, a malicious threat actor could use the compromised operating system as an attack vector, bypass the SCADA software entirely and use vendor specific libraries to communicate with the end device directly as in the case of the Stuxnet worm [21]. With PLCBlox, the SCADA/HMI application is not able to control the PLC directly due to the fact that write access has been disabled. Any compromise of its software components would not affect the behavior of the PLC. Moreover, the authentication and authorization tasks are now coordinated by the blockchain network which is an inherently more secure platform.

Audit trail enforcement

A compromised or malfunctioning SCADA system could be altered in a way that all components are functional except the audit trail logging aspects. In such a scenario a malicious user (e.g., rogue employee) would be able to execute commands and alter setpoints without his actions being recorded. With PLCBlox, any attempt to bypass the audit trail would prove meaningless as it is the only source for command and setpoint updates.

4.2 Prototype implementation

PLC devices are considered unsuitable for performing higher-level tasks such as blockchain transactions due to their limited resources. Their work memory (equivalent to RAM in computers) ranges from a few Kbytes to 32Mbytes for very large systems and code storage is usually also limited. The process of querying a blockchain node does not have high computational requirements such as those associated with the creation of signed transactions. Instead, predefined Application Programming Interface (API) calls are used such as those in the form of JSON-RPC documented for the Ethereum network[1]. Due to the resource limitations of PLC devices, a prototype implementation was necessary to first verify the feasibility of the solution, as well as to record real-world performance indicators and to provide the base for further development.

The complete design of PLCBlox solution consists of four elements, namely, the PLC application, an Ethereum private blockchain network, a smart contract and an Ethereum compatible wallet. A complete list of hardware and software tools used for the development and execution of PLCBlox implementation is summarized in the following table (Table 5).

[1] Ethereum JSON-RPC API - https://ethereum.org/en/developers/docs/apis/json-rpc/

Component	**Tools**
Development of PLC demo application and reference blockchain node client	Siemens TIA Portal V18 (Development IDE for Siemens PLC devices)
PLC device	Siemens S7-1511-1PN
HMI Device for acquiring results	HMI TP900 Comfort Panel (Simulated HMI panel)
Blockchain Network	Ganache private Ethereum blockchain environment
Authentication	Metamask Ethereum Wallet
Ethereum Smart Contract	Remix IDE (Solidity programming language)

Table 5 - List of hardware and software tools

PLC Application

The PLC application code incorporates a recently developed library designed for interfacing with the Ethereum node. Additionally, it includes a demo application that serves as an illustrative example of a streamlined industrial process, along with supplementary code employed for gathering performance indicators. The Ethereum interface library employs essential functions for executing HTTP API requests and implements the Ethereum "eth_call" client-side JSON API.

The demo application presupposes that the PLC device oversees the control of a machine, necessitating one temperature setpoint and one command to configure the operation mode. Consequently, it is imperative to retrieve the following two variables from the smart contract:

OperationMode: This variable signals the intended operational state of the machine.

Temperature: This variable represents the setpoint for the desired process temperature.

Smart Contract

The smart contract runs on the blockchain network and interfaces with the PLC device through the blockchain node. It processes users' requests for updating process parameters which are received in the form of transactions. It needs to also support the authentication and authorization functionality which would otherwise be handled by the SCADA device.

Authentication and Authorization

To showcase the authorization capabilities of PLCBlox, the smart contract incorporates distinct authorization levels outlined as follows:

***Administrator*:** By default, this role is assigned to the user that is deploying the smart contract. This user is able to assign other users to roles but cannot modify any other variables.

User: Users are restricted to modifying the machine's 'operationMode' variable exclusively.

SuperUser: A super-user has the capability to alter both variables, namely 'operationMode' and 'Temperature'.

The smart contract employs three variables (admin, user, superuser) to store the public addresses of users corresponding to their designated roles. Authentication relies on the inherent functionality of the blockchain network and the user's wallet. New requests reach the smart contract in the form of transactions which are signed with the user's wallet, utilizing the associated private key. The smart contract then verifies these transactions against known public keys, permitting functions to execute in accordance with the specified authorization level.

Functions

The smart contract provides the following functions to facilitate interaction with the PLC and enable user interactions:

ChangeUser and ChangeSuperUser: These functions are invoked by the admin to assign user and super-user permissions, respectively. The blockchain network address of a new user is passed in the function parameters.

SetTemperature: Super users utilize this function to modify temperature setpoints. The new setpoint value is passed as the function parameter.

SwitchON and SwitchOFF: Users or super users can call these functions to modify the desired operation mode.

Project Repository

In order to lay the ground for further research and development, all components of PLCBlox have been uploaded to GitHub[2], namely:

[2] PLCBlox GitHub repository - https://github.com/andshort/PLCBlox

Controller-side code including opensource communication libraries, the newly developed Ethereum client-side RPC library and the demo process application

Smart Contract source code (bloxtrail.sol)

4.3 Measurements

Developing the prototype implementation has been a vital phase of the research, serving as a testing ground to evaluate feasibility, performance and to record resources consumed. The complete solution, has been successful in verifying the complete feature set shown in Table 4, namely user authentication, authorization, the usage of the blockchain network as an audit trail database, as well as the interface between a PLC device with an Ethereum node. Most importantly, it demonstrates that a set of setpoints and commands can be updated without requiring direct access to the controller.

As discussed earlier, PLC devices are limited in resources. The most important metrics are the size of the actual code as well as the size of work memory. Work memory is the equivalent of RAM in computers but is a fraction of the size. The specific controller used in the implementation has 300kb of work memory. Although the source code storage can be extended by means of an SD card, the work memory cannot. The prototype implementation including all necessary libraries consumes 260kb of storage, as well as 14,5kb of work memory (5% of available).

Regarding the performance and in order for the results to better reflect real-world conditions, the CPU and blockchain network (Ganache) were located in separate locations and were connected through the in-ternet across a link with a moderate network latency of 12ms. Additionally, the Ethereum node calls were configured to run in a loop so as one call finishes, the next one starts immediately. Each call duration is measured by the CPU and an HMI device was configured to display the results. An example of 100 measurement points is shown in Figure 20. Each measurement point reflects the duration of the Ethereum node call and is calculated when the call is finished. In summary, each call completed within a time-frame of 17-28ms with a mean value of 22ms. Subtracting the baseline network latency (12ms with no network load), we can attribute an average cycle duration of 10ms to CPU computation and delays in the Ethereum network protocol.

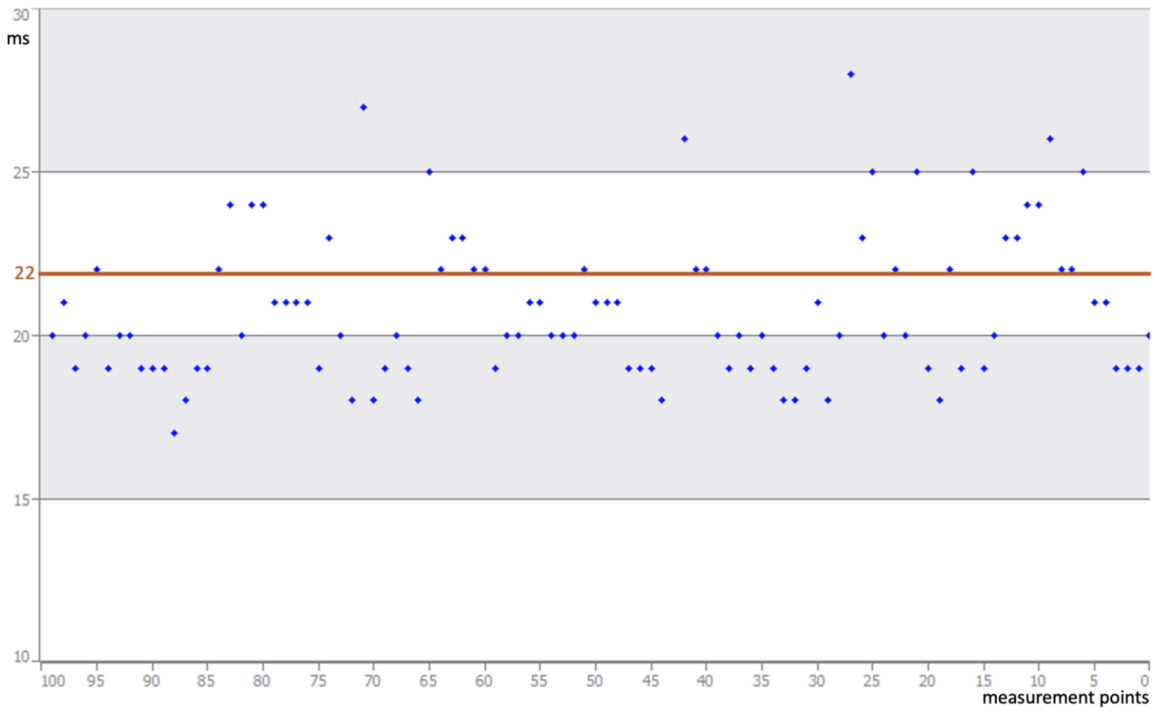


Figure 20 - Latency metrics over a period of 100 measurement points

The resource utilization figures and performance indicators are summarized in Table 6 bellow. The list focuses primarily on the PLC device, as this would act as the bottleneck performance-wise.

Metric	Value
PLC code size on PLC Device	260kb
Work memory (RAM)	14,5kb
Cycle duration (Request to response, end to end)	22ms
Throughput (calculated)	45requests/sec

Table 6 - Utilization figures and performance indicators

Discussion

The innovative approach presented in this research has shown that modern PLC devices are able to interface with common blockchain nodes, opening new possibilities for added functionality and security improvements. Furthermore, it is demonstrated how PLCBlox can take advantage of the intrinsic tamperproof characteristics of an Ethereum network in order to facilitate the storage of audit trail records.

Security and policy enforcement (Audit trail logs)

Authentication and authorization tasks are now handled by the user's wallet and the smart contract respectively. These two components coupled with the underlying blockchain network operation offer higher security standards which have been proven by their use in the finance sector, especially when taking into account the security issues that exist in typical industrial deployments.

Furthermore, by requiring less access permissions on the PLC device (Read only), it is impossible for an outside process to intentionally or unintentionally alter its data, in effect eliminating an attack surface that has been exploited over the years. Even further, due to the design, a compromised SCADA system would not be able to adversely impact the running process. Security requirements of these systems can therefore be relaxed.

The enforcement of the audit trail records, typically handled by the SCADA software on the end device, is now part of the system design. Moreover, there is no meaningful way to circumvent them, providing value to industries that require such policies.

Resource overhead and performance

It should be noted that the performance figures collected in the previous section reflect a mostly ideal setup due to the usage of a local Ethereum network node installation and the stable network connectivity. The authors argue that this setup more closely resembles typical deployments in industries that use on-premises audit trail database servers. Moreover, the small delay overhead imposed by the newly developed communication cycle should not be a concern for most industrial applications. At the same time, PLCBlox's small PLC code footprint and RAM requirements leaves most of its resources available for the operation of the main application.

Concerns when running PLCBlox on a pubilc blockchain network

While the solution is expected to function on public blockchain networks that share a similar RPC communications interface (such as the Ethereum Mainnet), one would need to consider the following factors:

a) longer network delays attributed to both network latency as well as the fact that the blockchain node is simultaneously catering to a wider audience,

b) reduced reliability due to the reliance on stable internet connectivity,

c) costs incurred during the deployment of the smart contract and the creation of transactions (as would be the case when inserting audit trail entries) and

d) privacy concerns arising from the fact that sensitive process information (process values, employee names or ids, justification of user actions) would be stored on a public medium.

For the reasons stated above, the use of PLCBlox in conjunction with public blockchain networks would not meet the requirements of most industrial use cases and has therefore not been examined in this research.

5. Conclusions and future work

The thesis successfully developed a framework for integrating FL with blockchain technology, demonstrating how blockchain can coordinate secure and transparent model training and data sharing across multiple entities. The implementation of smart contracts to manage FL processes, including data contribution, model aggregation, and reward distribution, proved effective in ensuring fairness and transparency. It designed and implemented effective incentive mechanisms, such as token rewards and reputation systems, to encourage high-quality data contributions in FL systems.

The research identified several industrial applications where the integration of DLT and ML can provide substantial improvements. Prototypes developed for use cases such as auditing showed that these technologies could significantly enhance operational efficiency and security. The practical benefits observed in these applications validate the potential of DLT and ML integration in real-world industrial scenarios. The study demonstrated that blockchain technology could enhance the security and functionality of programmable logic controllers (PLCs). By developing blockchain-based audit trails and command systems, the research reduced the vulnerability of PLCs to cyber-attacks and improved their operational reliability. This contribution is particularly valuable for industries relying on PLCs for critical operations.

5.1 Federated Learning process coordinated by a blockchain network rewarding capabilities

In the beginning of the research, we have examined the current work related to the use of Blockchain Technologies in the field of FL and we have identified a potential for blockchain technology to improve security of the FL process. We later on proposed an algorithm that is able to run inside a smart contract of a blockchain network. In contrast to the techniques currently available, where researchers relied on the use of data sample size[23],[26] or reputation[28] as a quality metric, our solution measures the accuracy of the model update directly. In this way, we do not need to assume that participants of the FL process are honest. The first results show that the algorithm provides a high level of protection against model poisoning attacks. We anticipate that results obtained can be considered as a baseline when implementing the algorithm on various blockchain technologies and that the algorithm can be extended in order to offer rewards on a blockchain network, relative to the quality of each contribution.

We lated build the specifications of a blockchain network and associated smart contracts that would allow a private blockchain network to be used in a unique way in order to support a FL process. We showed how it improves trust and security by running a verification algorithm within a smart contract which is executed amongst multiple nodes. The network can be configured to require consensus among a certain percentage of nodes.

In a later stage of this research we designed a FL model update assessment algorithm, designed in a way to be able to run inside of a smart contract and offer rewards. We extended the functionality of the algorithm in order to create a useful metric that can be used for reward allocation. We successfully verified the feasibility of the solution, by performing simulations with specially crafted images, representing datasets of varying quality. The results show that the algorithm is able to record user performance while at the same time makes the system resilient to label-flipping attacks.

The outcome of the devised point system makes it easy to rank participants. More importantly, adversaries are correctly identified and given no points, while the best performing user is ranked first. Although the verification algorithm is designed to run inside a smart contract, the simulations were run off-chain.

Researchers could verify how well the rewarding algorithm would behave in actual blockchain networks, more importanly to verify feasibility and measure the overhead imposed by the blockchain network.

5.2 Blockchain in industrial applications

The latest work focused on security issues with current deployments of Programmable Logic Controllers (PLCs) and Supervisory Control and Data Acquisition Systems (SCADA) in the industry and developed a solution that enables PLC devices to query a blockchain infrastructure for commands and setpoints.

The innovative approach (PLCBlox) presented in this research shows that modern PLC devices are able to interface with common blockchain nodes, opening new possibilities for added functionality and security improvements. Furthermore, it is has been demonstrated how PLCBlox can take advantage of the intrinsic tamperproof characteristics of an Ethereum network in order to facilitate the storage of audit trail records.

Security and Policy Enforcement (Audit Trail Logs)

Authentication and authorization tasks are now handled by the user’s wallet and the smart contract, respectively. These two components, coupled with the underlying blockchain network operation, offer higher security standards, which have been proven by their use in the finance sector, especially when taking into account the security issues that exist in typical industrial deployments.

Furthermore, by requiring fewer access permissions on the PLC device (read only), it is impossible for an outside process to intentionally or unintentionally alter its data, effectively eliminating an attack surface that has been exploited over the years. Even further, due to the design, a compromised SCADA system would not be able to adversely impact the running process. Security requirements of these systems can therefore be relaxed.

The enforcement of the audit trail records, typically handled by the SCADA software on the end device, is now part of the system design. Moreover, there is no meaningful way to circumvent them, providing value to industries that require such policies.

Resource Overhead and Performance

It should be noted that the performance figures collected in the previous section reflect a mostly ideal setup due to the usage of a local Ethereum network node installation and the stable network connectivity. The authors argue that this setup more closely resembles typical deployments in industries that use on-premises audit trail database servers. Moreover, the small overhead delay imposed by the newly developed communication cycle should not be a concern for most industrial applications. At the same time, PLCBlox’s small PLC code footprint and RAM requirements leave most of its resources available for the operation of the main application.

5.3 Future Work

Regarding the proposed solutions for running FL processes in blockchain networks, as well as the algorithms for rewarding functionality, we believe that researchers can extend this research in the following areas:

- Implement the FL and blockchain-based reward system in real-world scenarios across various industries to test its effectiveness and scalability. This will help identify practical challenges and refine the solution.
- Develop and integrate more sophisticated verification algorithms to better assess the quality of contributions. These could include mechanisms to detect and mitigate fraudulent behavior or poor-quality data submissions.

- Investigate advanced privacy-preserving techniques such as differential privacy or homomorphic encryption to further protect user data while maintaining high model performance.
- Explore and implement dynamic reward mechanisms that can adapt to changing data contributions and model requirements. This could include machine learning techniques to predict optimal rewards based on historical data.
- Focus on optimizing the scalability and performance of the solution, particularly in terms of processing speed and resource consumption. This includes testing with larger datasets and more complex models.
- Research and implement additional strategies to enhance user engagement and participation in FL, such as gamification elements or community-building initiatives.

Regarding the published work of PLCBlox as well as general blockchain use in industrial applications, we believe that researchers can conduct extensive scalability testing to evaluate the performance of the blockchain network under varying loads and transaction volumes. This includes simulating high-traffic scenarios to identify potential bottlenecks and optimizing the system to handle increased demands efficiently. The integration of PLCBlox with different blockchain protocols beyond Ethereum, such as Hyperledger Fabric, Corda, or other emerging technologies could further be studied, as it would help determine the most suitable blockchain infrastructure for various industrial applications based on specific needs and constraints. Further studies could focus on investigating advanced security measures to offer additional protection to the blockchain network and to PLC interactions. This could include the implementation of multi-factor authentication, hardware security modules (HSMs), and quantum-resistant cryptographic algorithms.

We also believe that there is space for study in the development of methods to ensure seamless interoperability between PLCBlox and existing legacy systems in industrial environments. This involves creating standardized interfaces and protocols that allow smooth integration without extensive modifications to current infrastructure.

We see benefit also in researchers conducting a comprehensive economic analysis to assess the cost-benefit ratio of implementing blockchain technology in industrial automation. This includes evaluating initial setup costs, operational expenses, and potential savings from enhanced security and reduced downtime.

Conducting a comprehensive economic analysis to assess the cost-benefit ratio of implementing blockchain technology in industrial automation, could also be meaningful, especially if the initial setup costs, operational expenses, and potential savings from enhanced security and reduced downtime are evaluated.

Future work will focus on scaling the proposed solution, optimizing performance, and expanding its application to a wider range of industrial and collaborative environments. By doing so, this research paves the way for more robust, secure, and efficient systems in both machine learning and industrial automation.

6. References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated Machine Learning,” *ACM Trans Intell Syst Technol*, vol. 10, no. 2, pp. 1–19, Jan. 2019, doi: 10.1145/3298981.

[2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated Learning: Strategies for Improving Communication Efficiency,” pp. 1–10, 2016.

[3] Z. Ghahramani, “LNAI 3176 - Unsupervised Learning,” pp. 72–112, 2004.

[4] A. Hard *et al.*, “Federated Learning for Mobile Keyboard Prediction,” 2018.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How To Backdoor Federated Learning,” Jul. 2018.

[6] L. Bottou, “Large-Scale Machine Learning with Stochastic Gradient Descent,” *COMPSTAT*, vol. 19th Inter, 2010, doi: 10.1007/978-3-7908-2604-3_16.

[7] European Parliament and Council of the European Union, “Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (Data Protection Directive),” *Official Journal of the European Union*, vol. L 119, pp. 1–88, 2016.

[8] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning,” *Sp*, pp. 1–15, Dec. 2018, doi: 10.1109/SP.2019.00065.

[9] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” *Proc IEEE Symp Secur Priv*, vol. 2019-May, pp. 691–706, 2019, doi: 10.1109/SP.2019.00029.

[10] A. Narayanan and V. Shmatikov, “How To Break Anonymity of the Netflix Prize Dataset,” 2006.

[11] A. Narayanan and V. Shmatikov, “Robust de-anonymization of large sparse datasets,” *Proc IEEE Symp Secur Priv*, pp. 111–125, 2008, doi: 10.1109/SP.2008.33.

[12] K. Bonawitz *et al.*, “Practical Secure Aggregation for Privacy-Preserving Machine Learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on*

Computer and Communications Security - CCS '17, New York, New York, USA: ACM Press, 2017, pp. 1175–1191. doi: 10.1145/3133956.3133982.

[13] M. Abadi *et al.*, "Deep learning with differential privacy," *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 24-28-Octo, no. Ccs, pp. 308–318, 2016, doi: 10.1145/2976749.2978318.

[14] R. C. Geyer, T. Klein, and M. Nabi, "Differentially Private Federated Learning: A Client Level Perspective," no. Nips, pp. 1–7, 2017.

[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, May 2017, doi: 10.29012/jpc.v7i3.405.

[16] D. Evans, V. Kolesnikov, and M. Rosulek, "A Pragmatic Introduction to Secure Multi-Party Computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2–3, pp. 70–246, 2018, doi: 10.1561/3300000019.

[17] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," *USENIX Security Symposium*, no. November, 2019, doi: https://doi.org/10.48550/arXiv.1911.11815.

[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," in *21st Century Economics*, 2008.

[19] U.S. Department of Health and Human Services Food and Drug Administration, "Part 11, Electronic Records; Electronic Signatures — Scope and Application," Guidance for Industry. Accessed: Nov. 26, 2023. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/part-11-electronic-records-electronic-signatures-scope-and-application

[20] U.S. Food & Drug Administration, "Computerized Systems used in Clinical Trials," Guidance for Industry. Accessed: Nov. 26, 2023. [Online]. Available: https://www.fda.gov/inspections-compliance-enforcement-and-criminal-investigations/fda-bioresearch-monitoring-information/guidance-industry-computerized-systems-used-clinical-trials

[21] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," *IEEE Security & Privacy Magazine*, vol. 9, no. 3, pp. 49–51, May 2011, doi: 10.1109/MSP.2011.67.

[22] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics," *IEEE*

Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1942–1976, Jul. 2020, doi: 10.1109/COMST.2020.2987688.

[23] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained On-Device Federated Learning," *IEEE Communications Letters*, vol. PP, no. c, pp. 1–1, 2019, doi: 10.1109/LCOMM.2019.2921755.

[24] U. Majeed and C. S. Hong, "FLchain: Federated Learning via MEC-enabled Blockchain Network," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, Sep. 2019, pp. 1–4. doi: 10.23919/APNOMS.2019.8892848.

[25] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study," *Applied Sciences*, vol. 8, no. 12, p. 2663, Dec. 2018, doi: 10.3390/app8122663.

[26] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive," *IEEE Trans Dependable Secure Comput*, vol. PP, no. 8, pp. 1–1, 2019, doi: 10.1109/tdsc.2019.2952332.

[27] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand : Scaling Byzantine Agreements for Cryptocurrencies," 2017.

[28] J. Kang, Z. Xiong, and D. Niyato, "Incentive Mechanism for Reliable Federated Learning : A Joint Optimization Approach to Combining Reputation and Contract Theory," *IEEE Internet Things J*, vol. PP, no. c, p. 1, 2019, doi: 10.1109/JIOT.2019.2940820.

[29] D. N. Dillenberger *et al.*, "Blockchain analytics and artificial intelligence," *IBM J Res Dev*, vol. 63, no. 2, 2019, doi: 10.1147/JRD.2019.2900638.

[30] N. Szabo, "Formalizing and Securing Relationships on Public Networks," *First Monday*, vol. 2, no. 9, Sep. 1997, doi: 10.5210/fm.v2i9.548.

[31] D. Wood, "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER," *Ethereum Project Yellow Paper*, 2014.

[32] E. Androulaki *et al.*, "Hyperledger fabric," in *Proceedings of the Thirteenth EuroSys Conference*, New York, NY, USA: ACM, Apr. 2018, pp. 1–15. doi: 10.1145/3190508.3190538.

[33] I. Martinez, S. Francis, and A. S. Hafid, "Record and Reward Federated Learning Contributions with Blockchain," in *2019 International Conference on Cyber-*

Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, Oct. 2019, pp. 50–57. doi: 10.1109/CyberC.2019.00018.

[34] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, "FedCoin: A Peer-to-Peer Payment System for Federated Learning," 2020, pp. 125–138. doi: 10.1007/978-3-030-63076-8_9.

[35] A. R. Short, H. C. Leligou, M. Papoutsidakis, and E. Theocharis, "Using Blockchain Technologies to Improve Security in Federated Learning Systems," in *Proceedings - 2020 IEEE 44th Annual Computers, Software, and Applications Conference, COMPSAC 2020*, IEEE, Jul. 2020, pp. 1183–1188. doi: 10.1109/COMPSAC48688.2020.00-96.

[36] A. R. Short, H. C. Leligou, and E. Theocharis, "Execution of a Federated Learning process within a smart contract," in *39th International Conference on Consumer Electronics IEEE ICCE2021*, 2021, p. In Press.

[37] A. Ahmad, M. Saad, M. Al Ghamdi, D. H. Nyang, and D. Mohaisen, "BlockTrail: A Service for Secure and Transparent Blockchain-Driven Audit Trails," in *IEEE Systems Journal*, Institute of Electrical and Electronics Engineers Inc., Mar. 2022, pp. 1367–1378. doi: 10.1109/JSYST.2021.3097744.

[38] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards Blockchain-Driven, Secure and Transparent Audit Logs," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, New York, NY, USA: ACM, Nov. 2018, pp. 443–448. doi: 10.1145/3286978.3286985.

[39] C. Regueiro, I. Seco, I. Gutiérrez-Agüero, B. Urquizu, and J. Mansell, "A blockchain-based audit trail mechanism: Design and implementation," *Algorithms,* vol. 14, no. 12, Dec. 2021, doi: 10.3390/a14120341.

[40] S. Suzuki and J. Murai, "Blockchain as an Audit-Able Communication Channel," in *Proceedings - International Computer Software and Applications Conference,* IEEE Computer Society, Sep. 2017, pp. 516–522. doi: 10.1109/COMPSAC.2017.72.

[41] W. Pourmajidi and A. Miranskyy, "Logchain: Blockchain-Assisted Log Storage," in *IEEE International Conference on Cloud Computing, CLOUD,* IEEE Computer Society, Sep. 2018, pp. 978–982. doi: 10.1109/CLOUD.2018.00150.

[42] P. Snow *et al.*, “Factom Business Processes Secured by Immutable Audit Trails on the Blockchain,” 2014. [Online]. Available: www.Factom.org

[43] P. Schmid, A. Schaffhäuser, and R. Kashef, “IoTBChain: Adopting Blockchain Technology to Increase PLC Resilience in an IoT Environment,” *Information,* vol. 14, no. 8, p. 437, Aug. 2023, doi: 10.3390/info14080437.

[44] A. Vick, W. Chen, and J. Krueger, “Using Solana Blockchain and OPC UA for Trusted Third Party Industrial Robot Control Services,” *ISR Europe 2023, 56th International Symposium on Robotics, Stuttgart*, pp. 332–337, 2023.

[45] S. Loss, L. Cardoso, N. Cacho, and F. Lopes, “Pharmaceutical Audit Trail Blockchain-Based Microservice,” in *Proceedings of the 16th International Joint Conference on Biomedical Engineering Systems and Technologies,* SCITEPRESS - Science and Technology Publications, Mar. 2023, pp. 368–375. doi: 10.5220/0011685100003414.

[46] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” pp. 1–15, Dec. 2014.

[47] F. Sattler, S. Wiedemann, K. R. Muller, and W. Samek, “Robust and Communication-Efficient Federated Learning from Non-i.i.d. Data,” *IEEE Trans Neural Netw Learn Syst*, vol. 31, no. 9, pp. 3400–3413, 2020, doi: 10.1109/TNNLS.2019.2944481.

[48] A. R. Short and H. C. Leligou, “Novel defense scheme against model poisoning attack,” 2020.

[49] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017,* vol. 54, 2017.

[50] T. Zahariadis, P. Trakadas, S. Maniatis, P. Karkazis, H. C. Leligou, and S. Voliotis, “Efficient Detection of Routing Attacks in Wireless Sensor Networks,” in *2009 16th International Conference on Systems, Signals and Image Processing,* IEEE, Jun. 2009, pp. 1–4. doi: 10.1109/IWSSIP.2009.5367775.

[51] P. Trakadas *et al.*, “An artificial intelligence-based collaboration approach in industrial iot manufacturing: Key concepts, architectural extensions and potential applications,” *Sensors (Switzerland),* vol. 20, no. 19, pp. 1–20, 2020, doi: 10.3390/s20195480.

[52] X. Pan, Z. Wang, and Y. Sun, “Review of PLC Security Issues in Industrial Control System,” *Journal of Cyber Security*, vol. 2, no. 2, pp. 69–83, 2020, doi: 10.32604/jcs.2020.010045.

APPENDIX A - Improving Security and Fairness in Federated Learning Systems Python Code

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

```
from __future__ import absolute_import, division, print_function,
unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
class_names = ['0','1','2','3','4','5','6','7','8','9']
```

```
from keras.datasets import mnist
import random

(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
train_images = train_images / 255.0
test_images = test_images / 255.0
train_labels_adversary = np.array([random.randint(2, 2) for x in
train_labels])

train_images_split = np.array_split(np.array(train_images),10)
```

```
train_labels_split = np.array_split(np.array(train_labels),10)
train_labels_adversary_split =
np.array_split(np.array(train_labels_adversary),10)

average_model = keras.Sequential()
flmodel = list()
average_model = keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(128, activation='relu'),
        keras.layers.Dense(10)
    ])

average_model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True
),
                  metrics=['accuracy'])
#average_model.fit(train_images, train_labels_random, epochs=1)
average_model.save_weights('model2.h5')
flmodel = list()
for x in range(10):
      flmodel.append(keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(128, activation='relu'),
        keras.layers.Dense(10)
        ]))
      flmodel[x].compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True
),
                  metrics=['accuracy'])
```

```
accuracy = [list()]
loss = [list()]
accuracy.append(list())
loss.append(list())


for adversaries in range(0,4):
  average_model.load_weights('model2.h5')
  for x in flmodel:
    x.load_weights('model2.h5')

  for x in range(1):
    newround(average_model.get_weights(), adversaries)
    weights = list()
    for model in flmodel:
      if
evaluatemodel(average_model.get_weights(),model.get_weights()):
        weights.append(model.get_weights())
    #weights = [model.get_weights() for model in flmodel]

    mean_weights = list()
    for weights_list_tuple in zip(*weights):
        mean_weights.append(
            np.array([np.array(w).mean(axis=0) for w in
zip(*weights_list_tuple)])
        )

    avg_model_weights = average_model.get_weights()
    new_weights = [0.8*x + 0.2*y for x, y in zip(avg_model_weights,
mean_weights)]


    average_model.set_weights(new_weights)
    test_loss, test_acc = average_model.evaluate(test_images,
test_labels)
    accuracy[adversaries].append(test_acc)
    loss[adversaries].append(test_loss)
    accuracy.append(list())
    loss.append(list())
    print('\nTest accuracy:', test_acc, 'Adversaries:',
adversaries, 'round:', x)
```

```
import matplotlib.pyplot as plt
for i in range(4):
    plt.plot(accuracy[i],label = '%s%% Adversaries'%(i*10))
plt.xlabel("Rounds")
plt.ylabel("Accuracy")
plt.legend()
plt.
plt.grid()
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 20
fig_size[1] = 16
plt.rcParams["figure.figsize"] = fig_size
plt.show()
```

```
def newround(global_model, adversaries):

  for x in range(0, adversaries):
    print('adversary', x)
    flmodel[x].set_weights(global_model)
    flmodel[x].fit(train_images_split[x],
train_labels_adversary_split[x] , epochs=1)

  for x in range(adversaries , 10):
    print('honest', x)
    flmodel[x].set_weights(global_model)
    flmodel[x].fit(train_images_split[x], train_labels_split[x],
epochs=1)
```

```
def evaluatemodel(global_model_weights, model_update):
  average_model.save_weights('pre.h5')

  average_model.set_weights(global_model_weights)
  test_loss, test_acc = average_model.evaluate(test_images,
test_labels)

  new_weights2 = [0.8*x + 0.2*y for x, y in
zip(global_model_weights, model_update)]
  average_model.set_weights(new_weights2)
  test_loss_candidate, test_acc_candidate =
average_model.evaluate(test_images,  test_labels)
  average_model.load_weights('pre.h5')
  if test_acc_candidate>test_acc:
    print('Model Update is Healthy')
    return True
  else:
    print('Model Update should be discarded')
    return False
```

APPENDIX B – PLCBlox, Smart contract code

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

import "hardhat/console.sol";

contract BloxTrail {

    address private admin;
    address private user;
    address private superuser;

    int16 public Temperature;
    bool public OperationMode;

    modifier isAdmin() {
        require(msg.sender == admin, "Caller is not the admin");
        _;
    }
    modifier isUser() {
        require(msg.sender == user, "Invalid Permissions");
        _;
    }
    modifier isSuperUser() {
        require((msg.sender == user) || (msg.sender == superuser) , "Invalid Permissions");
        _;
    }

    constructor() {
        console.log("BloxTrail contract deployed by:", msg.sender);
        admin = msg.sender; // 'msg.sender' is sender of current call, contract deployer for
a constructor
        user = admin;
        superuser = admin;
    }

    function changeUser(address newUser) public isAdmin {
        user = newUser;
    }
    function changeSuperUser(address newSuperUser) public isAdmin {
        superuser = newSuperUser;
    }

    function SetTemperature(int16 newTemperature) public isSuperUser {
        Temperature = newTemperature;
    }
    function SwitchON() public isUser {
        OperationMode = true;
    }
     function SwitchOFF() public isUser {
        OperationMode = false;
    }
}
```

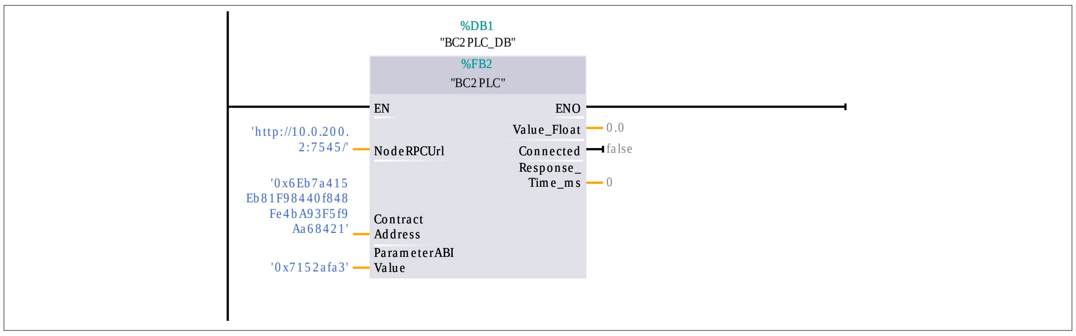
APPENDIX C – PLCBlox, PLC Code

plcblox / PLCBlox [CPU 1511-1 PN] / Program blocks

Main [OB1]

Main Properties					
General					
Name	Main	Number	1	Type	OB
Language	LAD	Numbering	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	Author		Comment	
Family		Version	0.1	User-defined ID	

Network 1:

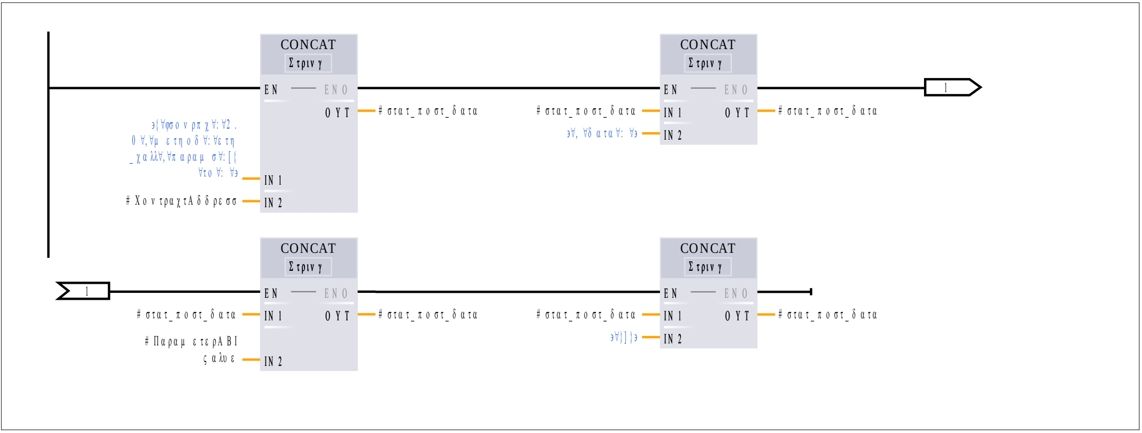


plcblox / PLCBlox [CPU 1511-1 PN] / Program blocks

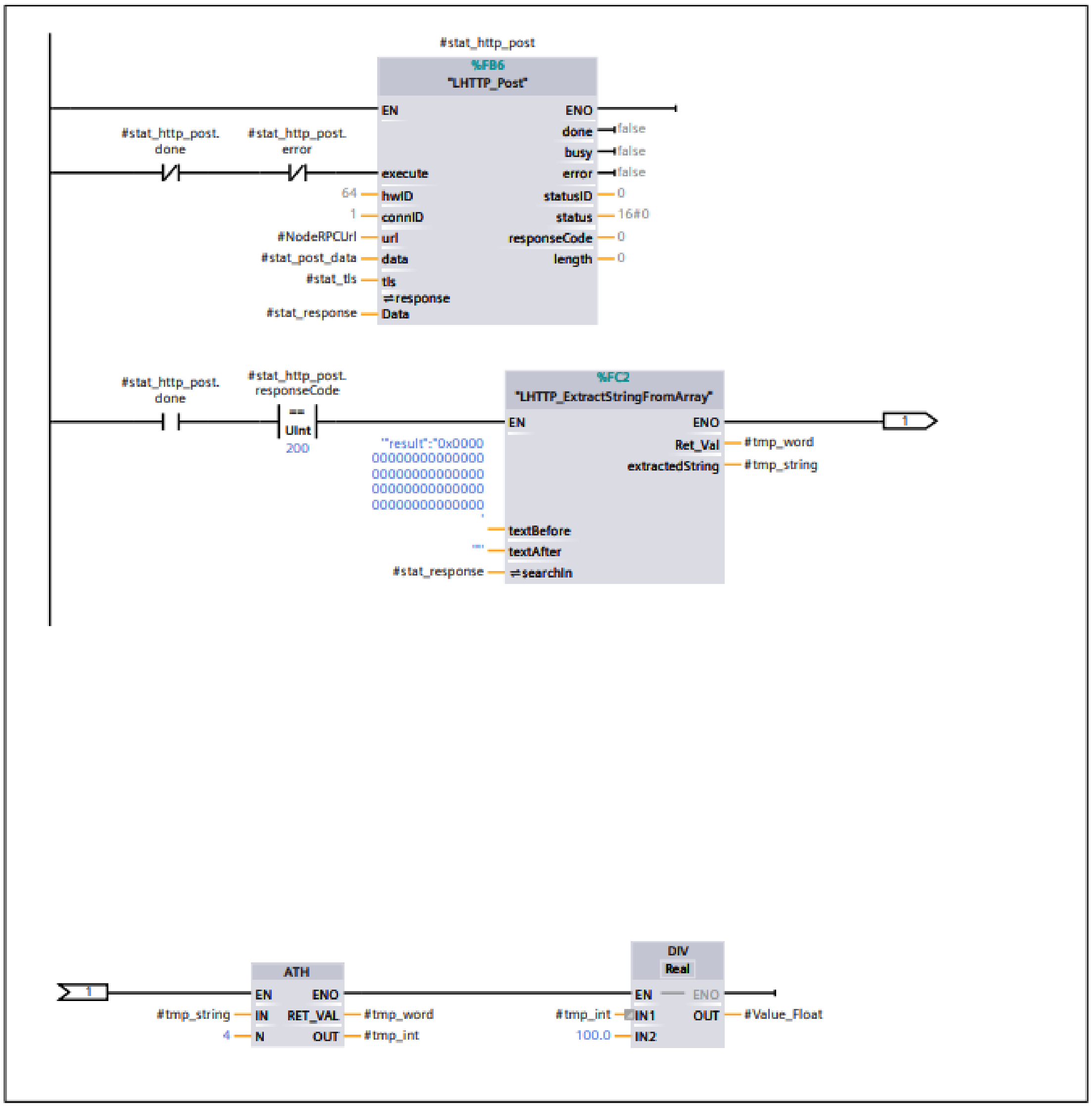
BC2PLC [FB2]

BC2PLC Properties					
General					
Name	ΒΧ2ΠΛΧ	Number	2	Type	ΦΒ
Language	ΛΑΔ	Numbering	Αυτομ ατιχ		
Information					
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

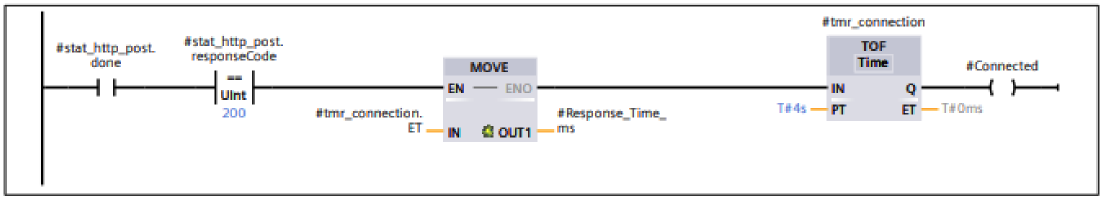
Network 1:



Network 2:



Network 3: Connected


#stat_http_post.done
#stat_http_post.responseCode
==
UInt
200
MOVE
EN
ENO
#tmr_connection.ET
IN
OUT1 #Response_Time_ms
#tmr_connection
TOF
Time
IN
Q
T#4s PT
ET T#0ms
#Connected