IMPROVING SECURITY AND FAIRNESS IN FEDERATED LEARNING SYSTEMS

Andrew R. Short[1], Theofanis G. Orfanoudakis[2] and Helen C. Leligou[3]

[1]Department of Industrial Design and Production Engineering,
University of West Attica, Athens, Greece
[2]School of Science and Technology, Hellenic Open University, Patras, Greece
[3]Department of Industrial Design and Production Engineering,
University of West Attica, Athens, Greece

ABSTRACT

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. Since the parameter server does not have access to the actual training datasets, it becomes challenging to offer rewards to users by directly inspecting the dataset quality. Instead, this paper 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 judgment of the quality of contribution, we devise a point system to record user performance assisted by blockchain technologies. More precisely, we have developed a verification algorithm that evaluates the performance of users' contributions by comparing the resulting accuracy of the global model against a verification dataset and we demonstrate how this metric can be used to offer security improvements in a Federated Learning process. Further on, we implement the solution in a simulation environment in order to assess the feasibility and collect baseline results using datasets of varying quality.

KEYWORDS

Incentive schemes, Federated Learning, Blockchain, Rewards, Incentive schemes.

1. INTRODUCTION

Federated Learning (FL) is a subfield of Machine Learning, first introduced by Google in 2016 [1]. It allows participants to collaboratively engage in model training while not requiring them to move data to a central location. This alone offers significant privacy improvements, since the data remains on the user's device, allowing for many more use cases and helps overcome restrictions imposed on data usage by regulations such as GDPR [2] and HIPAA [3].

Considering that an overabundance of data is generated at end devices (e.g., smartphones), it is often preferable to perform training at the edge. Bearing in mind that many newer mobile devices are equipped with dedicated hardware for machine learning, training at the edge is becoming very efficient. Organizations may be motivated to participate in FL, knowing that a collectively built model will perform better than one trained with only locally generated data. If this motivation is not enough, rewarding schemes can be designed, in order to incentivize users to contribute using their data. One such example is Google Keyboard [4] which offers an improved typing experience to users that contribute using data created on their device (i.e. through typing). In the simplest and most popular form of FL, the datasets that belong to participating entities have

similar feature sets. This is the case when organizations have related operations such as datasets of customers belonging to Bank A and Bank B respectively. In this example it is desired that the sample spaces do not overlap between datasets. This type of FL is referred to as horizontal FL [5]. Vertical FL can be performed in the case that the datasets do not share same feature-sets but at least share some common sample-space.

In a FL environment, the joint global model evolves through the continuous contributions of participating parties, usually in the form of model weights [1], [6]. These updates are created when local training is performed on end devices. However, the server that maintains the parameters does not have access to the actual training data, and therefore cannot assess its correctness and quality. Thus, a new attack surface is created, whereas a malicious actor could either attack the training data, the training process, or the resulting model updates. In all of these cases, the intention of the attacker would be to make the global model deviate as much as possible from its intended direction. Two popular attacks are label-flipping [7], [8] and backdoor formation [9], [10].

The label-flipping attack involves manipulating the training data set. In particular, training labels are intentionally altered, while the data remains unchanged. Let's consider the following example: The popular MNIST database of handwritten images is used for a typical digit classification application, and an attacker is motivated to make the machine learning model classify the handwritten number 3 as the number 8. The attacker would then craft a malformed training data set which would include handwritten images of number 3 but would falsely labeled as 8. In a federated learning setup, the resulting model updates would hinder the performance of the global model and depending on the scale of the attack, the configured learning rate and the averaging algorithm used, could even render the model unusable and lead to a denial-of-service attack.

Backdoor-based attacks involve the installation of a hidden feature in the machine learning model. The machine learning model initially appears to function correctly, until the attacker uses a specific pattern to trigger the backdoor, at which point the model's behavior is altered. For this attack to be successful, the attacker trains the model so that a certain prediction is made when a specific pattern appears as an input. In the case of an image classifier, the attacker would for example embed an image pattern (e.g., a watermark), or in the case of a word predictor, a certain sequence of words would be used as the trigger.

Several attempts have been made in order to fuse blockchain with federated learning so as to offer advantages, such as protection against model poisoning attacks and the possibilities for rewards, as described in more detail in section 2. Our solution is based on a smart contract architecture and enables the allocation of fair rewards to participants, while at the same we show that it is able to protect against label-flipping attacks. We design a reward mechanism, intended to be run inside a smart contract, which offers rewards to participants based on the quality of contributions (Section 3) and specify the requirements of the smart contract in term of functions and storage structure (Section 4). Later on, we setup a suitable testing environment in order to verify the feasibility of the solution and collect results in simulations with honest participants that own high quality datasets, honest participants that own lesser quality datasets, and adversaries that aim to poison the global model (Section 5).

2. POTENTIAL BENEFITS OF FL COORDINATION THROUGH A BLOCKCHAIN NETWORK – RELEVANT WORK

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 [11]. 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 [12] 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 [11] are not designed for such use cases, later public and private blockchain technologies such as Ethereum [13] and HyperLedger Fabric are designed to support the development and execution of customized smart contracts [14]. Figure 1 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.

In the sequel, we analyze the benefits that result when the FL process is coordinated within a smart contract. In this section we use the term “blockchain” as this is widely used to refer to DLT technologies in total. It is anticipated that these solutions offer advancements in the following areas:

Data integrity: Federated Learning 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 continue to be accessible from the rest of the healthy nodes.

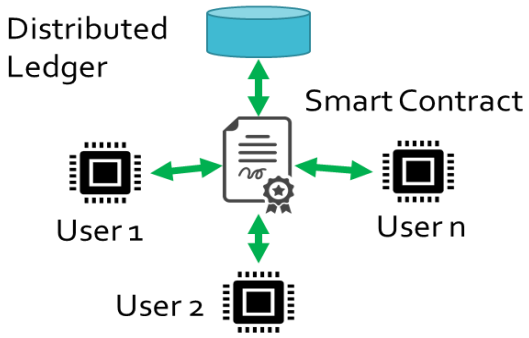

Figure 1. Smart contract operation principle.

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 when only one parameter server is used. Instead, a suitable failover design would need to be implemented. Such 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 federated learning system. In a Horizontal Federated Learning 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 Federated Learning, 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. 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 Federated Learning 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 a 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.

H. Kim et al [15] 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) [16].

U. Majeed et al [17] 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 [18] have implemented Federated Learning 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 federated learning. 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 [19] who implemented a blockchain assisted Federated Learning setup 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[20]. The learning environment is based on TensorFlow and the results show how the training accuracy increases with more participating parties.

Kang et al [21] 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[22] 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.

Han Yu et al [23] have approached the problem of model update evaluation by measuring the accuracy against a reference model and propose a dynamic payoff-sharing scheme. The work emphasizes on different profit-sharing schemes but does not focus on the mechanism which creates the quality metric, nor does it analyze the effectiveness of the approach in the presence of malicious users.

In a study by Xiaoyu Cao et al [24], a mechanism for protection against malicious clients has been proposed in the context of Federated Learning. In this approach, an algorithm creates multiple global models from a random subset of clients. A majority vote between the clients is then used to select the next model version. Image recognition simulations against public database MNIST show that they can achieve 88% model accuracy when 20% of the users are malicious. Results in section 5 of our paper show a higher model accuracy convergence (more that 90%) in the presence of 30% malicious users. However, ourapproach relies on a verification dataset and for this reason results cannot be directly compared.

Although researchers are focusing on methods to offer incentives to participants [15], [19], [21], [22], these are not calculated based on the actual value of model updates. The work of I. Martinez et al. [16] 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 [25]) in order to provide a metric proportional to the model update contributions and therefore increase fairness in reward allocation. b) it describes the implementation 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 Federated Learning process within a smart contract [26].

3. THE REWARDING ALGORITHM

Building upon previous work [25],[26], 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 2, 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 by 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 whether 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. Depending on the blockchain solution used, this process will take place most likely 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 the dataset in order to assess its 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 is 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 federated learning, i.e., analogous to the quality of their model updates.

3.1. Principle of Operation

The steps involved in the federation process and the operation of the algorithm are outlined in 0. 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.

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. This next global model version is in turn 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.

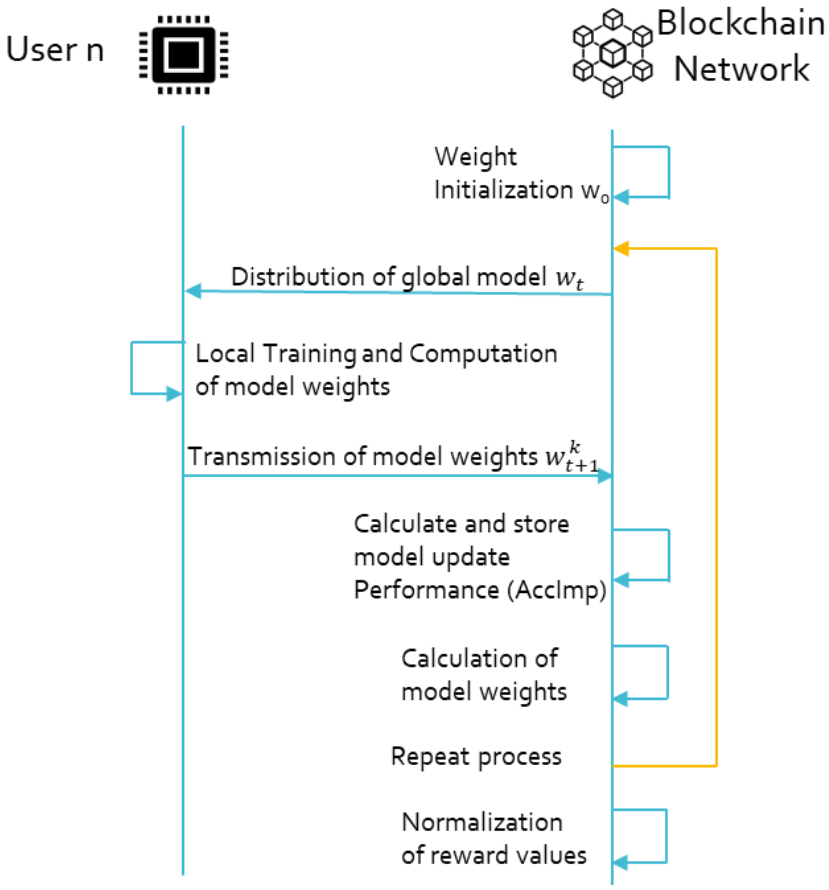

Figure 2. Principle of Operation

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.

3.2. Algorithm 1 – Reward Calculation inside the FL process

The following notation is used in the pseudo-code: 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

---

procedureClientUpdate(w)
    B ← split Pk to smaller sets
    for all b ∈ B do
        W←w-η∇f(w,b)
    end for
    return W
end procedure

---

procedureAccImp(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

Normalization is performed in order to better differentiate between user performamce. During training, it is possible that the reward points collected by a user have a negative value. This can occurif the submitted model updates were determined to affect the global model in a negative way. During our testing, we have opted to zero out these negative values. The following formula has been used for normalization:

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

4. 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 (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.

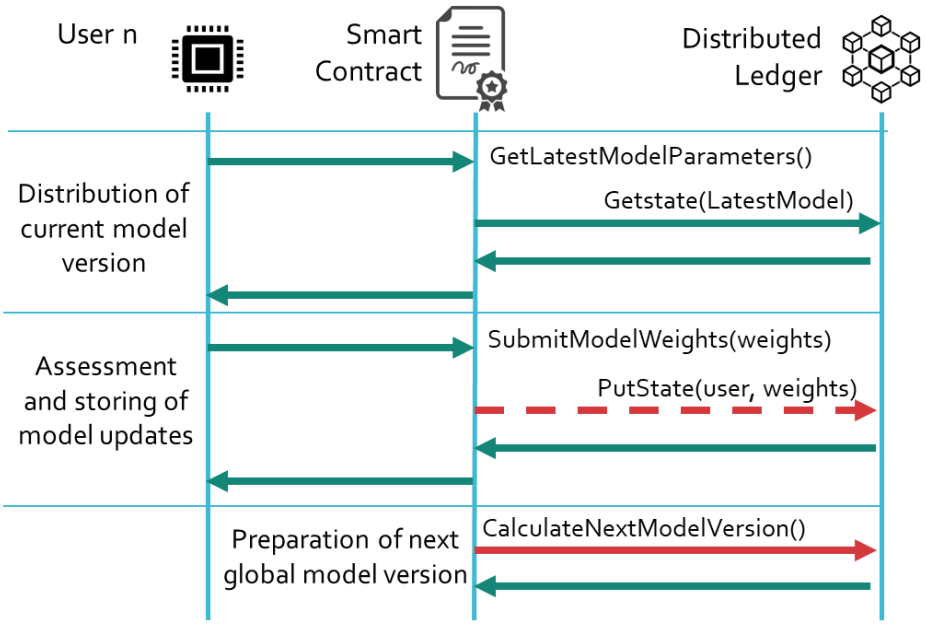

Figure 3. Relationships and Operations.

Furthermore, 0 depicts in greater detail the relationships between users, the smart contract and the distributed ledger, for the implementation of the three main processes that occur in a training round, when federated learning 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 that is specific to HyperLedger Fabric (an 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 in the ledger or to discard it. It is a possibility for failed contributions to be recorded in the ledger, even in the case that they are not used in the next model version, in order to conform with 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 by 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 these two conditions. When the function is triggered, the ledger is queried, and all useful contributions are retrieved. Federated Averaging [1] 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 incur a large overhead.

In order to support the aforementioned functionality, some values need to be stored inside 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 (0). The first key-value pair is required in order to store the weights of the current model version. It is a multi-dimensional array, it is preset before the first training round starts, and it is updated at the end of each training round. In the scope of the FL process, it is referenced by the GetLatestModelParameters() function and is updated when CalculateNextModelVersion() is invoked.

Table 1. 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])

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.

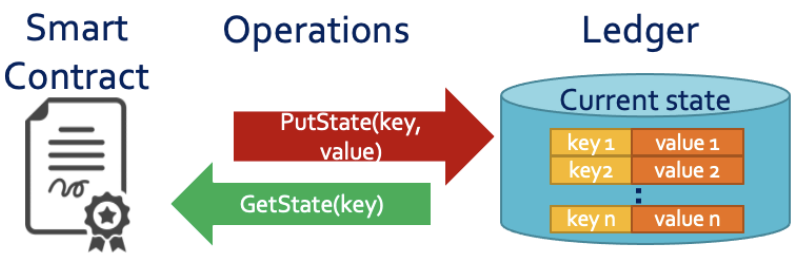

Figure 4. Distributed Ledger operations

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 newer 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.

5. TESTBED

This section includes a description of the tools used for simulating the FL process and collecting measurements. In order to simulate the federated learning process, we have developed scripts in Python language in order to implement algorithm 1 with the aid of open-source frameworks and libraries such as TensorFlow and Kerasas well as publicly available reference datasets from the MNIST database. Before the process starts, the model is configured using the open-source Keras libraries and is initialized 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 in grayscale. The images belong to 10 classes representing the ten numerical digits 0-9 and are commonly used as benchmarking datasets during training and validation. This database is publicly available and used by machine learning developers and researchers in order to record baseline performance.

The developed scripts simulate the federated learning process as depicted in figure 2, support all functions of algorithm 1, and are initially used to collect baseline resultsusing the MNIST database. They also handle necessary functions such as model initialization, model training and the model update aggregation (Federated Averaging [1]). In more detail, the model used throughout this paper is initialized using the Keras libraries, 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.

The experiment is run in the following sequence: First, we carry out performance measurements using a simple federated learning process (also known as vanilla FL i.e., without any

modification to the algorithm) in the presence of adversaries and we collect baseline results (Figure 5). We then use algorithm 1 and repeat the process. We later discuss improvements regarding the security of the global model (Figure 6), as well as show how the point system works (Figure 9) and how it can be used to offer rewards.

We first concentrate on the defense against the malicious nodes. For this reason, during this specific simulation, a federated learning 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 [27]). 0 and 0 show how the global model performs in the presence malicious participants when protections are disabled and enabled respectively. Not surprisingly, when malicious updates are taken into account during model aggregation (0), 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.

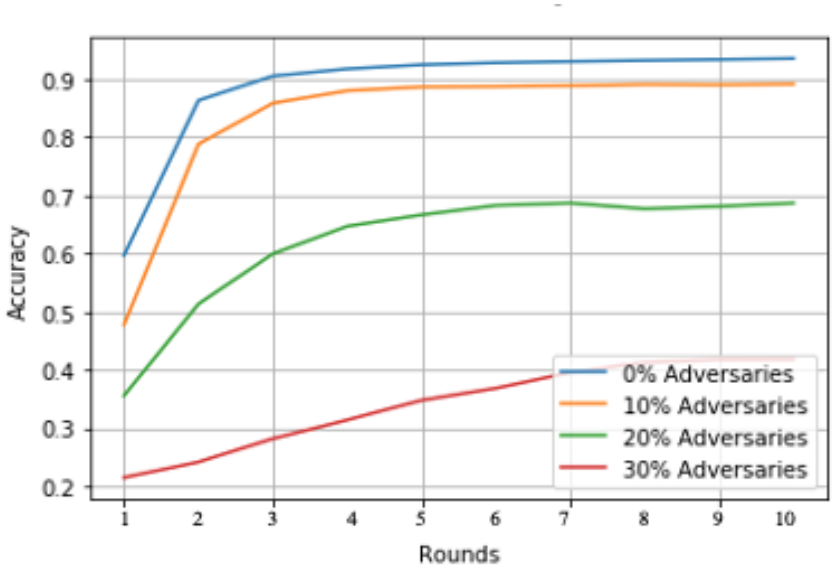

Figure 5. Performance with traditional FL algorithm in the presence of adversaries

When the verification algorithm is configured to discard useless model updates (0) we can observe that it is able to effectively differentiate between useful/harmful model updates and therefore offer a substantial protection against this type of attack.

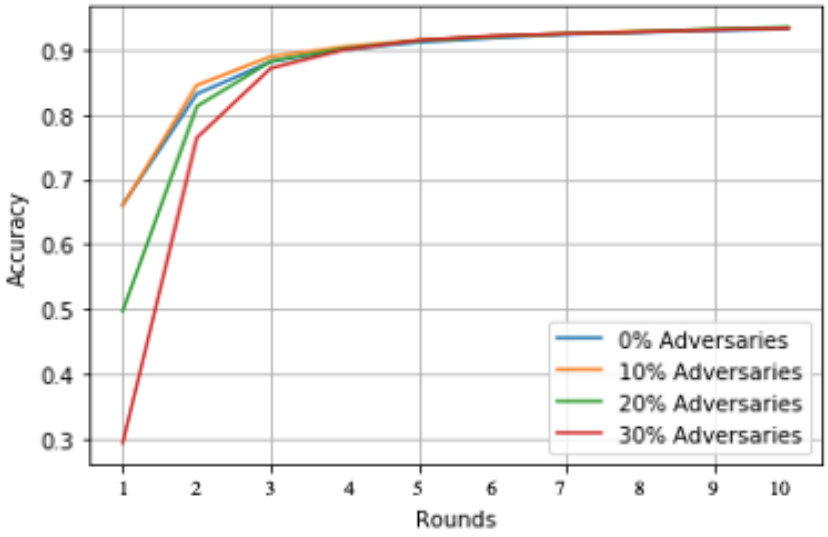

Figure 6. Performance with model update protection in the presence of adversaries.

Then, we turn our attention to the evaluation of rewards, analogous to the model improvement brought by each entity. A simulation of algorithm 1 has been run using the tools described earlier in this section. In this particular setup, 10 participants join an FL process which lasts 10 training rounds. With respect to the training and validation data, we use the popular MNIST database of handwritten images. The training data is then 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 (0). In this simulation, 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 as described in section 1 and are actively trying to degrade the performance of the model.

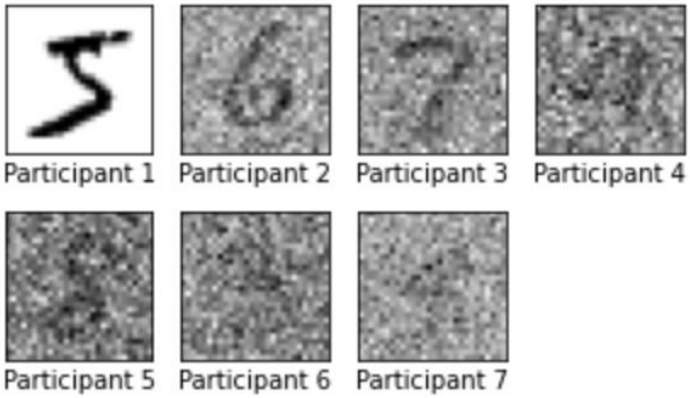

Figure 7. Training images with gaussian noise applied

0 depicts the difference in accuracy between the evaluation model (temporary model p_{t+1} in algorithm 1) and the global model i.e., the result of function AccImp. Even in this extreme simulation consisting of 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).

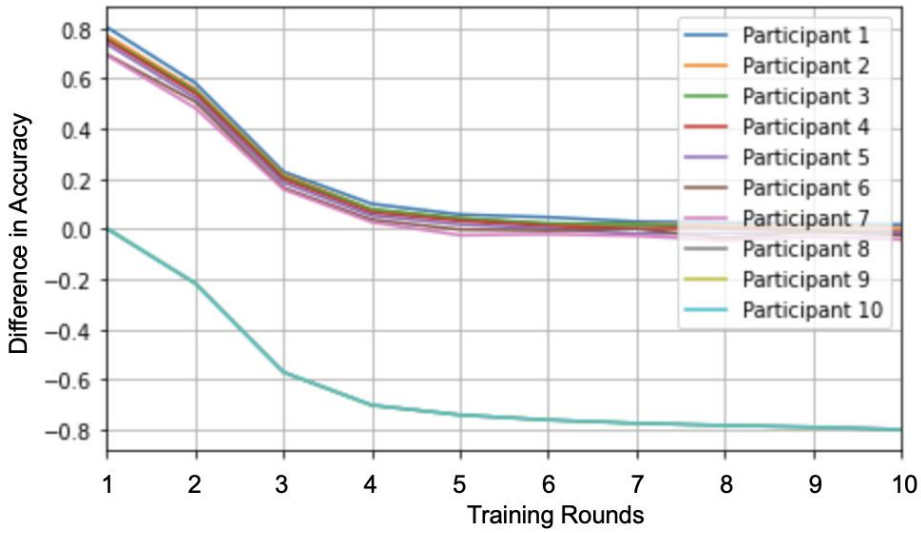

Figure 8. Difference in accuracy of each contribution

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 opt to perform normalization of all values after the training process.

0 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 [28]).

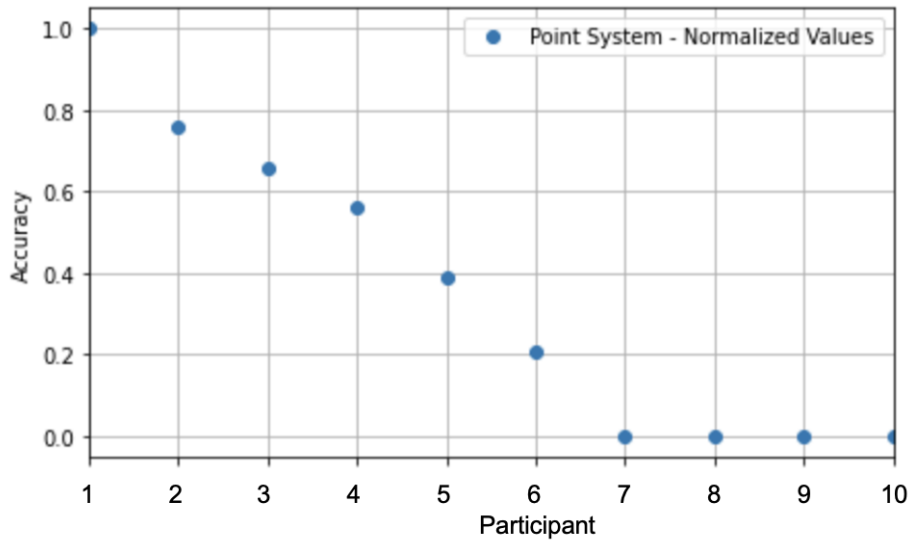

Figure 9. Normalized values, used for rewards

6. CONCLUSIONS

The work presented in this paper involves an FL model update assessment algorithm, designed in a way to be able to run inside of a smart contract. Its functionality has been extended so that it can differentiate between different data quality levels with respect to the datasets provided by users during training. The challenge in real FL use cases is that the coordinator is unable to measure the quality of the training data directly since it is generated and kept at edge devices. Contrary to related work, the algorithm presented in this research does not make any assumptions regarding the presence of this data, nor does it rely on the honesty of participating users. Instead, it measures the performance of the resulting model in terms of its accuracy against a verification dataset. In order to assess its effectiveness and response to the presence of malicious users, a public dataset of training images was modified so that they now contain varying levels of gaussian noise. In this image recognition use case, results show improvements in the following two areas: First, a metric is assigned to each user that represents the quality of the training data used. The authors anticipate that this metric can be used as-is in order to provide incentives to well-behaving users, for example by transferring tokens inside the blockchain network. Secondly, results show that the algorithm successfully protects the global model from malicious behavior by discarding malformed model updates. At the same time, the malicious users can be penalized by recording a low (or negative) performance value on the distributed ledger. As future work, the authors intend to implement the algorithm in a private blockchain network in order to automatically pay out users in the form of blockchain tokens and to further examine the potential of the security protection scheme against different types of attacks.

ACKNOWLEDGEMENTS

All authors would like to thank the University of West Attica for the financial support provided to them to undertake this research project.

REFERENCES

[1] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," Proc. 20th Int. Conf. Artif. Intell. Stat. AISTATS 2017, vol. 54, 2017.

[2] 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)," Off. J. Eur. Union, vol. L 119, pp. 1–88, 2016.

[3] G. J. Annas, "HIPAA Regulations — A New Era of Medical-Record Privacy?," N. Engl. J. Med., vol. 348, no. 15, pp. 1486–1490, 2003, doi: 10.1056/nejmlim035027.

[4] T. Yang et al., "Applied Federated Learning: Improving Google Keyboard Query Suggestions," arXiv, Dec. 2018.

[5] 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.

[6] 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. https://arxiv.org/abs/1610.05492

[7] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, vol. 2019-Decem, pp. 233–239, 2019, doi: 10.1109/ICPADS47876.2019.00042.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," Proc. 29th Int. Conf. Mach. Learn. ICML 2012, vol. 2, pp. 1807–1814, 2012.

[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," Jul. 2018. http://arxiv.org/abs/1807.00459

[10] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," https://arxiv.org/abs/1712.05526, 2017.

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

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

[13] D. Wood, "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER," Ethereum Proj. Yellow Pap., 2014.

[14] E. Androulaki et al., "Hyperledger fabric," in Proceedings of the Thirteenth EuroSys Conference, 2018, pp. 1–15, doi: 10.1145/3190508.3190538.

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

[16] 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), 2019, pp. 50–57, doi: 10.1109/CyberC.2019.00018.

[17] 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), 2019, no. September, pp. 1–4, doi: 10.23919/APNOMS.2019.8892848.

[18] 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," Appl. Sci., vol. 8, no. 12, p. 2663, Dec. 2018, doi: 10.3390/app8122663.

[19] 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 Secur. Comput., vol. PP, no. 8, pp. 1–1, 2019, doi: 10.1109/tdsc.2019.2952332.

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

[21] 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.
[22] 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.
[23] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A Fairness-aware Incentive Scheme for Federated Learning. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES '20). Association for Computing Machinery, New York, NY, USA, 393–399. DOI:https://doi.org/10.1145/3375627.3375840
[24] Xiaoyu Cao, Jinyuan Jia, Neil Zhenqiang Gong. Provably Secure Federated Learning against Malicious Clients. The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), 2021
[25] 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, 2020, pp. 1183–1188, doi: 10.1109/COMPSAC48688.2020.00-96.
[26] 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, In Press.
[27] 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, 2009, pp. 1–4, doi: 10.1109/IWSSIP.2009.5367775.
[28] 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.

AUTHORS

Andrew Ronald Short is member of the teaching staff and Ph.D. candidate at the dept. of Industrial Design and Production Engineering, the University of West Attica in Athens, Greece. He has received his bachelor's and master's degree at TEI Piraeus (Greece) and the University of Sheffield (UK) respectively.

Dr. Theofanis G. Orphanoudakis received his Dipl-Ing. degree in Electrical and Computer Engineering in 1995 and the Ph.D. in Telecommunications in 1998, both from the National Technical University of Athens. Since 2013 he has been an Assistant Professor, in the Department of Sciences & Technology of the Hellenic Open University.

Dr. Helen C. (Nelly) Leligou is currently associate professor at the University of West Attica. From 2007-2017, she acted as an assistant professor at the Technological Educational Institute of StereaEllada. She received both Dip. Ing. and Ph.D. degrees from the National Technical University of Athens, Electrical and Computer Engineering Dept., Greece.