Attack Detection Using Federated Learning in Medical Cyber-Physical Systems

William Schneble
Computing and Software Systems
University of Washington Bothell
Bothell, WA 98011
Email: schneble@uw.edu

Geethapriya Thamilarasu
Computing and Software Systems
University of Washington Bothell
Bothell, WA 98011
Email: geetha@uw.edu

***Abstract*—Medical Cyber-Physical Systems (MCPS) are networked systems of medical devices that provide seamless integration of physical and computation components in healthcare environments to deliver high quality care by enabling continuous monitoring and treatment. As MCPS store sensitive medical data and personal health data, security breaches and unauthorized access to this information can lead to severe repercussions for both the patient and hospital in the form of loss of privacy, abuse, physical harm and liability. The heterogeneity of devices involved in these systems (such as body sensor nodes and mobile devices) introduce large attack surfaces and hence necessitate the design of effective security solutions for these environments. In this paper, we design and implement a massively distributed, machine-learning-based intrusion detection solution for MCPS. Specifically, we explore the concept of Federated Learning to minimize the communication and computation costs involved in traditional machine learning based solutions. We evaluate our design with real patient data and against security attacks such as Denial of Service, data modification, and data injection. Experimental results illustrate that our system achieves high detection accuracy of 99.0% and a False Positive Rate of 1.0% along with a reduced network communication overhead. Lastly, we show that the system can cope with unevenly distributed data and is a scalable solution that leverages the computing resources of many mobile devices.**

I. Introduction

Medical cyber physical systems (MCPS) enables a life-critical, context aware networked systems of medical devices with seamless integration of physical and computational components. Recent advances in Internet of Things, including, wireless sensors, connected medical devices and mobile healthcare make MCPS a promising platform to improve effectiveness of patient treatment and deliver high quality healthcare. MCPS helps constantly monitor and analyze information gathered from medical devices, infer the patient's health condition for diagnosis, and provides timely treatment either through direct feedback from healthcare providers or through automated treatments using medical actuators.

Despite their several benefits, MCPS impose numerous security and privacy challenges. The heterogeneous nature of MCPS and the increased use of wireless and mobile technologies in these systems introduces new attack surface areas and security vulnerabilities. Security attacks in MCPS can result in unauthorized access to sensitive medical and personal health information. Malicious attacks also can lead to false diagnosis and/or incorrect treatment potentially resulting in loss of human life. For instance, settings of a compromised medical device such as a cardiac pacemaker can be modified or entirely shut down resulting in dire consequences. One of the leading cause of vulnerabilities in MCPS is that, although they are designed to be isolated, they are increasingly connected with other systems and networks [1]. Compounding with MCPS's already heterogeneous and ad-hoc nature, security solutions are limited and often lacking in interoperability. In addition to the potential for injury and liability, vulnerabilities in medical devices may also be used to gain backdoor entry into rest of the network.

In this paper, we aim to improve the security of medical cyber physical systems by designing an intrusion detection system (IDS). Existing IDS based security solutions tend to have high false positives rate, often need manual modification, specification and are difficult to scale in the MCPS environment. It is evident from recent research trends that use of machine learning for intrusion detection is very effective in accurately detecting attacks. However, data fusion, pre-processing and resource complexity of machine learning algorithms act as a deterrent in their deployment in a MCPS environment. With these limitations in mind, the main goal of this paper is to design and develop a detection system with high accuracy, low false positives, low communication cost, along with the flexibility and scalability suited for the MCPS environment. Specifically, we explore the feasibility of machine learning approach (using Federated learning) to provide robust attack detection in MCPS.

II. Related Literature

Humayed *et al*. provide a clear abstraction of Medical Cyber-Physical Systems (MCPS) by identifying three major components: communication, computation and control, and monitoring and manipulation (cyber, cyber-physical, and physical respectively) [1]. While MCPS has become invaluable in persistent healthcare monitoring, they also pose a significant security risk given the highly sensitive and valuable information they measure and transmit [2]–[4]. Cyber physcal systems were originally designed to be isolated but their increased connectivity with other systems and networks has contributed

to their components being more integrated. It is at these weak points between *isolated* systems that most attacks occur [1].

Cryptographic security solutions have been proposed for wireless body area networks and medical devices to address the problems of data confidentiality, privacy and authentication. Due to the resource constrained nature of the sensors, existing literature is mostly focused on use of symmetric key cryptography for encryption/decryption and asymmetric cryptography for the key exchange [5]–[7]. These solutions however fail to detect security attacks, where attacker has access to an authorized device, or insider attacks, where a legitimate entity turns malicious. Intrusion Detection Systems (IDS) are often deployed to detect insider attacks or as second layer of defense against external attacks that breach through existing security controls.

Anomaly based detection methods have been extensively investigated in the literature especially to detect sensor anomalies in wireless sensor networks used for healthcare applications [8]–[13]. In [9], the authors identify unique data states and build a Markov Chain that predicts the probability of a data state transition occurring. The proposed design was able to detect new anomalies and variations in the data but presented high false negatives rate which made it unsuitable for MCPS deployment. Coppolino *et al*. use a local agent on the sensor to detect anomalous behavior which is then sent to a central agent (presumably on the controller) for a final decision [10]. If communication with the central agent is lost, then the node will cycle through neighbors before finally making the local agent decision persistent. Thus, a targeted attacker could jam or denial of service a node to force the node to make an ill-informed decision.

It is noteworthy to mention that these anomaly based detection solutions were not aimed at security attacks, rather on legitimate anomalies due to system and network faults. Centralized anomaly detection solutions have the benefit of increased space for memory and storage as well as computation, but they can be vulnerable to routing attacks [10]. Decentralized models run at the sensor nodes and thus tend to be extremely lightweight in resource consumption and have limited visibility or potentially inconsistent views of the network. For this reason decentralized models often employ other, centralized components such as in [8], [10].

Behavior-Specification based Intrusion Detection (BSID) proposed by Mitchell *et al*. uses a set of behavior states for specifying acceptable behavior of the device [14]. The BSID design utilizes the device behavior rules as input and detects for deviation against the expected behavior. While this solution is one of the very few existing IDS solutions in literature specific to MCPS, it requires domain knowledge to create behavior rules and their performance is heavily correlated to rule coverage. This makes the approach difficult when dealing with complex devices and systems, such as distributed and ad-hoc systems, because the amount of time to complete the specification is high and the odds of missing some machine states is likely. If a behavior state is not identified and it is an unsafe state, the model will not be able to detect this exploit and needs to be updated.

In addition to the many different types of sensors from various vendors, MCPS generates a lot of data such as an ECG application with 288 kbps or 1 Mbps for audio applications [15]. Machine learning algorithms can be highly effective to deal with the large amounts of data in these systems. Odesile *et al*. use mobile agents based machine learning to perform hierarchical anomaly detection at the sensor and at the cluster heads [16]. The flexibility of the mobile agents to move around the network to where data is stored or to computation rich devices makes this approach very attractive. However, cluster heads have greater authority and power in their ability to spawn sensor and detective agents which may make attackers lurk in the network until they become or compromise the cluster head. Salem *et al*. use a Support Vector Machine (SVM) to detect anomalous behavior [17]. This approach is good for noise suppression but is not an ideal solution for isolating faulty sensors in the network.

The literature review reveals that research on machine learning based detection solutions for medical cyber physical systems is very limited and is still in its early stages. To that effect, in this work, we explore and further the research on utilizing machine learning for detecting intrusions in medical cyber physical systems.

III. Network Architecture and Attack Model

A. Network Architecture

We consider a medical cyber physical system architecture as shown in Figure 1. The MCPS network consists of medical devices that are basically wireless body sensor nodes placed on the patient's body; mobile devices that acts as a local gateway to the medical devices and a back-end server at the hospital. The sensor nodes are used to collect patient vitals and administer drugs, such as insulin or anesthetics. The mobile device acts as a gateway for the medical devices. The sensor nodes communicate with the mobile device wirelessly using a short-range communication protocol, such as Bluetooth or Zigbee. The mobile device collects, aggregates, and keeps a history of node measurements, such as blood pressure. To communicate with the hospital server, the mobile device uses IEEE 802.11 to connect to a gateway that connects to the Internet. The server is also connected to the Internet via a wired connection to the hospital's gateway. The server is responsible for handling messages transmitted from the mobile device as well as relaying messages back to patient's mobile devices. The system follows a client-server topology between patient's mobile devices and the hospital server. This ensures scalability as adding more or new mobile devices in the hospital network increases message traffic and logic at the server linearly.

B. Attack Model

The MCPS environment's security is particularly sensitive given the high value of the medical record data and potentially severe repercussions on patient health. Malicious users can use packet header information and payload data to launch attacks

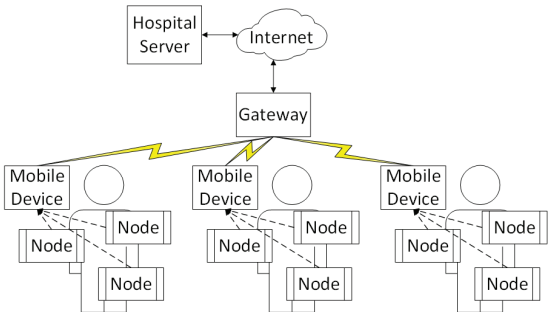


Fig. 1: Network architecture

that are either network or host-based. Below, we describe the common security attacks in the context of medical cyber-physical systems.

Denial of Service (DoS): In a MCPS environment, vulnerabilties in a conencted medical devices can allow the hacker to gain complete control over the device and modify its settings leading to a denial of service attack where the patient cannot receive the necessary medical intervention and treatment. For instance, hacked cardiac pacemakers can be controlled remotely and reprogrammed by hackers disrupting patients heart rhythms and potentially leading to loss of life.

Data Modification: Attackers can launch Data modification attack in MCPS [14], [18] that modifies data (ex: fake an arrhythmia), leading to misallocation and disruption of medical resources as well as potentially leading to denial of required treatment for a patient in need. Data modification attacks can also be used to fool biometric recognition systems and thereby let an attacker gain access to other systems or impersonate an healthcare professional [19].

Data Injection: In Data injection attack, malicious data such as false sensor reading, packet header information or command or query to/from medical device is input to the system, altering the normal course of execution. This attack can drive the system into an unsafe state where data loss, denial of service, and data integrity attacks are made possible.

Eavesdropping: Eavesdropping attack, where data communication is intercepted by unauthorized users, can lead to a breach in privacy; it can also be used to launch other attacks such as replay attacks. Additionally, there are regulations and compliance requirements for the healthcare industry, such as HIPAA, for dealing with sensitive patient data. A breach of protected health information could not only harm patients, but also impose significant risk and liability on the hospital.

IV. Federated Learning for Intrusion Detection

In this work we develop a Federated Learning based Intrusion Detection System (FLIDS) for medical cyber physical systems. Federated Learning is a distributed machine learning algorithm that builds a global model by averaging weights w across many devices over a number of communication rounds t. We modify the original federated learning algorithm from [20] for detecting intrusions in MCPS. The intrusion detection architecture, shown in Figure 2, utilizes the computational resources of the mobile devices and runs the detection module, while the server acts as the central authority and is responsible for registering the mobile devices, calculating the federated model, and storing the model. We describe the design process in detail as follows:

A. Clustering of Patients

The attack detection process begins with a mobile device registering with the server. Devices are then clustered into different groups based on their patient history. Clustering of the mobile devices (see Figure 2) is introduced to separate patients, such as a young healthy person and an elderly person with arrhythmia, who do not share similar behavioral patterns. These differences between groups can normally prevent convergence of the IDS leading to reduced detection accuracy and increased training time. Hence, we enable clustering of users such that similar individuals are grouped together allowing for quicker convergence and more accurate and personalized models. Features such as age, common medical conditions, medications, and history such as smoking/nonsmoking can be used for clustering. Each group has a federated model stored on the server. The mobile device then downloads the federated model from the server and continues to learn and update a new model using the patient's data.

The clustering process occurs during registration of a mobile device with the hospital sever (Figure 2). After a mobile device has been assigned to a group, it only receives and contributes to that groups model via updates as detailed in Algorithm 1. Determining the correct number of clusters will depend on several factors including the number of mobile devices in network and the number of parameters used for clustering process. If the network has a limited number of mobile devices, then it may not be reasonable to use a high number of clusters. For instance, if only 12 patients are in the network, then using four clusters may mean there are groups of only one, two, or three mobile devices. This is problematic because, in the case of a single patient in a group, no benefit of the federated learning process can be claimed. For smaller groups, such as three patients, the entire set of mobile devices is involved in each communication round (in federated learning training) making the process very resource intensive. Furthermore, the fewer patients that are assigned to a group, lesser the data used to train the federated model. This decreases the model's generalization in the long term when new patients enter the group. Thus, determining the correct number of clusters for FLIDS is largely an empirical one that depends on the system and its constraints.

B. Training and Updating the Model

Federated Learning is a distributed machine learning algorithm that builds a global model by averaging weights w across many devices over several communication rounds t. Figure 3 shows an example of weights for a single layer, feed forward neural network. Each neuron in the hidden layer has a transfer function, denoted by f, that takes each feature in a sample $(In_1...In_i)$ and multiplies it by its weight $(IW_{1,1}...IW_{i,1})$ plus a bias (B_1). The weights are modified during training.

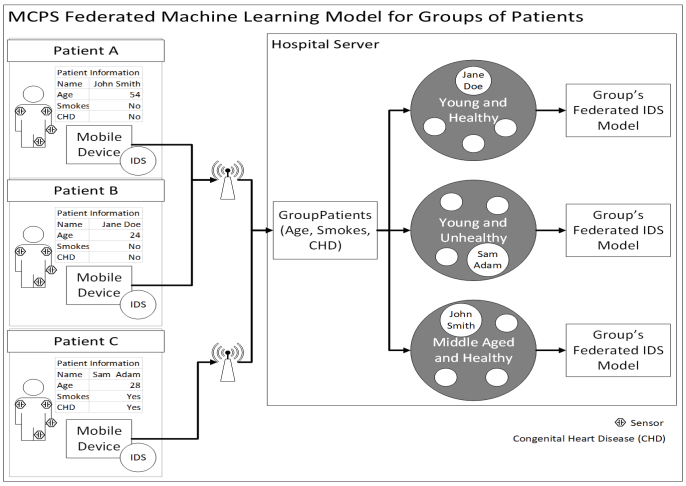


Fig. 2: Federated learning based IDS (FLIDS)

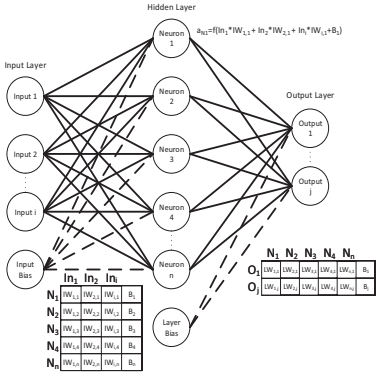


Fig. 3: Single layer, feed forward network with weights and bias.

In this paper, we refer to weights to include both weights and bias (both matrices in Figure 3).

Our modified federated learning algorithm at the server, as shown in Algorithm 1, uses parameter K as the number of clients for a given group and w_k is the mobile device k's model weights. The weights are a learnable parameter and are modified during training of the neural network. In this paper, when we refer to weights we mean all weights and bias. During each round, the server randomly selects a subset of patients from the group. Parameter C denotes the fraction of patients to use for an iteration with $C = 0$ being a random value between $[c_{min}, 1]$ where c_{min} is the minimum fraction of patients to use for a single round. For instance, if $C = 0.5$ and eight patients were in the group then four of them would be selected at random for the round. If $C = 0$, $c_{min} = 0.25$, and eight patients were in the group then a random subset of patients between two and eight will be selected per round. The server has three major responsibilities: registering and grouping patients, calculating and storing the federated models, and distributing the federated model. As previously described, when a patient registers with the server

input : $S_{clients}$ - set of clients maintained by the hospital server
w - the federated model weights
t - the communication round identifier
C - fraction of clients to use in round
output: w^{t+1} - the next communication round's model weights
initialize w
while *True* **do**
  $S_{rclients}$ = selectRandomSubset($S_{clients}$, C)
  K = size($S_{rclients}$)
  for *each client in* $S_{rclients}$ **do**
    w_k^{t+1} = getClientUpdate(w)
  end
  $w^{t+1} = \Sigma_{k=1}^{K} w_k^{t+1}/K$
  for *each client in* $S_{clients}$ **do**
    client.send(w^{t+1})
  end
  t = t + 1
end

Algorithm 1: Federated Learning algorithm at hospital server

they are clustered into a group who share a single IDS model. Figure 4 shows that the server is also in charge of selecting a subset of the model's patients used for each iteration of the federated learning process (as described later in algorithm 1). This subset is chosen randomly and uniformly for a given range. For example, if there are 8 patients registered to group A, and if we do not want only one patient used for an iteration then we can use the range [0.25, 1] to determine how many patients will be used. After determining the number of patients to use, the server selects patients from the group at random and without replacement. The mobile devices of the subset are then sent a message by the server to send their model's weights to the server. The server keeps a record of each mobile device's weights and at an end of a communication round calculates the next federated model w^{t+1}. The federated model

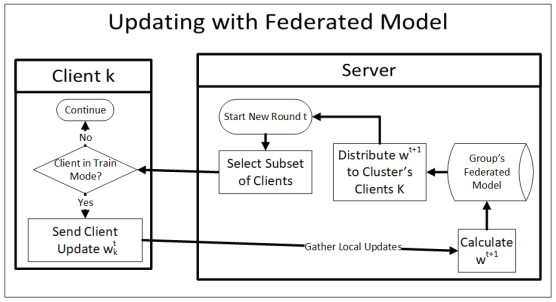


Fig. 4: The updating process

is calculated by summing all of the patients' weights and dividing by the number of patients used for the round. If a patient is unable to send weights to the server, then weight matrices of zeros are added to the sum and the number of patients used for the round, parameter K, is decreased by one (repeat for each missing patient). These new weights are then sent to all mobile devices registered to the group and the process repeats until a specified number of iterations or the change in model's weights at the server are insignificant. After the new model is calculated at the server, it is pushed out to all mobile devices registered in the cluster. The server then sends the newly learned model to all registered patients for the group. This process repeats until the model converges: the weights do not change significantly between an iteration.

The mobile device with the newly learned IDS can be in one of two modes: training or testing. If the device is in training mode, then a patient can make predictions and provide updates back to the hospital server to contribute to the global IDS model. Otherwise, the device is set to test mode where only predictions on new samples are made. Test mode saves communication costs as the IDS weights do not need to be sent to the server, but the mode is primarily used when the mobile device cannot securely collect a baseline for training.

Resource management can also be performed by manipulating parameter C, because use of fewer devices implies less traffic overhead between the devices and server. Fewer mobile devices also decreases the total computation cost used per round without impacting individual training time or detection accuracy. However, training time at the mobile device can be controlled by changing the number of iterations performed, parameter E, and the batch size given by parameter B. The number of iterations is the number of times the gradient is estimated, and the parameters of the model updated. The batch size is the number of samples used per iteration. Lowering the batch size also decreases the computation required to compute the gradient but may increase the number of iterations needed to converge. Increasing the number of iterations would cost more computation power but this can be offset by increasing the number of communication rounds in the federated learning process. Overall, with varying number of mobile devices used per round, the system is highly configurable.

C. Attack Detection

With the latest model obtained, new samples are fed into the IDS and are determined to be either normal or abnormal behavior. A data sample comprises of node values or features, such as heart rate. If the node values are within an expected range and there is no major change in correlation to other node values, then the sample is defined as normal behavior. For example, if the blood oxygen saturation levels are expected to be between 95% and 97% then a normal sample with this feature will have a value in this range. Similarly, if blood oxygen saturation and pulse have a strong and positive correlation then the expected behavior is that when the patient's pulse increases then the patient's blood oxygen saturation increases as well. Abnormal behavior is defined as readings outside of the expected range or unexpected correlations with other node measurements. A data modification attack may change a node value to below an expected value. Data injection may have correct node values but different correlations between the features, such as the pulse increases but blood oxygen saturation decreases. If an abnormal sample is found, then an alert is generated at the mobile device so that medical staff can intervene.

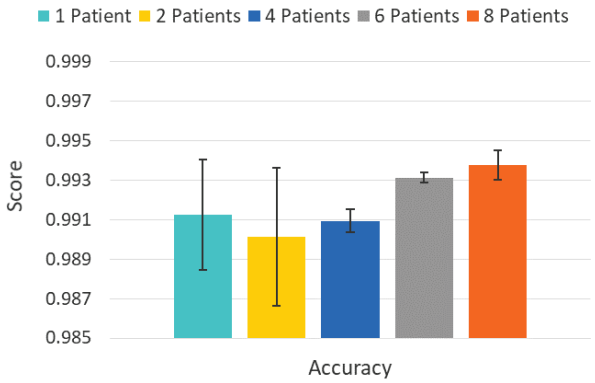
V. Experimental Setup and Results

The MIMIC dataset from PhysioNet is used for evaluation of the proposed system [21]. This dataset has six features, which are typically displayed on an ICU monitor, including elapsed time, arterial blood pressure, heart rate, pulse, respiratory rate, and blood oxygen concentration. The dataset has 121 records with each record having about 35-40 hours of monitored activity. The raw signals include ECG signals that measure the voltage across leads placed on the body. The machine learning was executed using Sci-kit Learn's Multi-Layer Perceptron running on Raspberry Pi's. Each Raspberry Pi is associated with a patient who is generating data for the device to train the IDS. All attack were simulated using new patient data; data the ML model has not been trained on. Half of the data samples are modified to simulate attacks and half remain unperturbed. We also simulated the system using MATLAB by following the same process as above, where the Raspberry Pi's were replaced with MATLAB objects to represent each device. The MATLAB simulation uses a single layer feed forward neural network (MATLABs patternet) and is run sequentially. The time per training round is calculated by taking the total sequential time divided by the number of participating objects. We evaluated the performance of our system using following metrics: Detection Accuracy, False positives rate, recall, F1 score, Training time and communication cost.

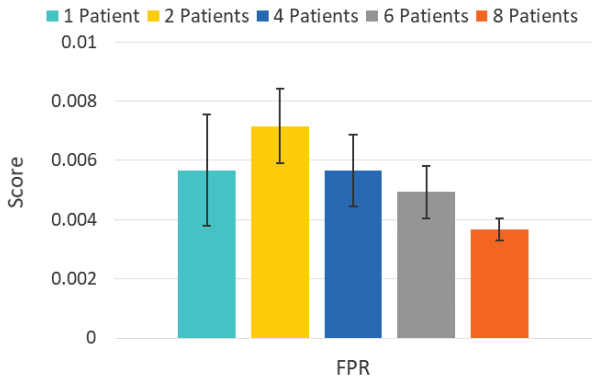
A. Results

In our experiments, we vary the parameter C, (number of mobile devices in the network) for each training round. For instance, $C = 1$, implies the entire set of patients in network were used for training, $C = 0.5$ means half of the patients were used, and $C = 0$ means a random number of patients were used per round of algorithm 1.

We initially run the experiments with $C = 1$, where all registered patients are used for each training round. Figure 5 demonstrates the performance of FLIDS by measuring the

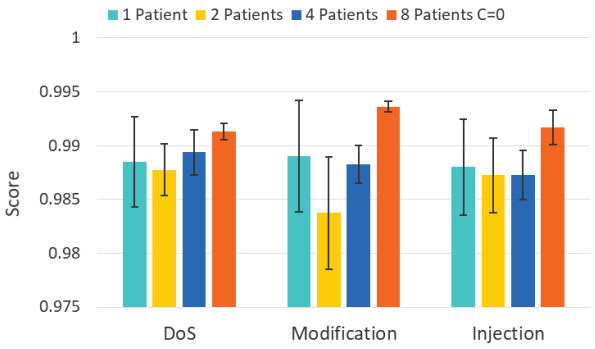


(a) Detection accuracy of FLIDS

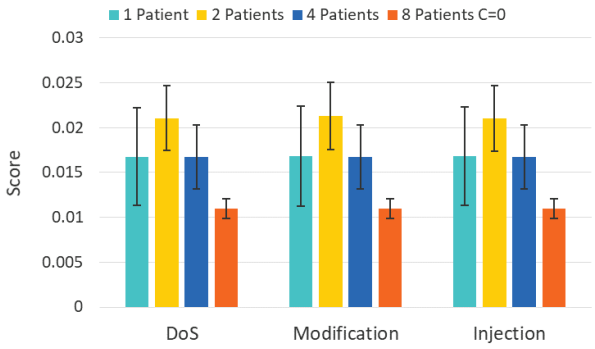


(b) False Positive Rate of FLIDS

Fig. 5: Evaluation of FLIDS with $C = 1$



(a) Attack specific detection accuracy of FLIDS



(b) Attack specific False Positive Rate of FLIDS

Fig. 6: Security attacks evaluation of FLIDS

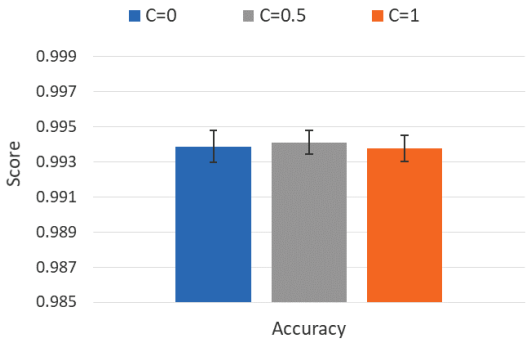


Fig. 7: Eight patients in a group and varying the number of patients used per training round (parameter C).

detection accuracy and false positives rate. The metrics were averaged over five runs and the error bars represent the standard deviation. As we observe from the results in 5a, increasing the number of patients in the network increases detection accuracy by 0.25% and reduces the distribution or variance by 0.2%. Figure 5b similarly indicates that false positives was reduced by 0.2% when using eight patients to generate the federated model compared to a single patient. Overall these results show that more patients in the network means improved performance of machine learning algorithms in terms of detection accuracy and false positives while also reducing variability of the model, the standard deviation, for all metrics.

Next, we vary the values of C to 0.5 where half of the patient population is used for training and to 0, where a random number of patient population is chosen for each round with a minimum of one patient. The results from Figure 7 show that there is no tradeoff in detection accuracy when only half of the patients in a group are being used per round. This result indicates that we can achieve a similarly performing model using fewer patients for training. As a result, we can use half as many patients per training round and reduce our communication and energy cost (see Figure 8).

We simulate the Denial of Service (DoS) attack, data modification and data injection attacks in our network and evaluate the performance. As seen in Figure 6, the accuracy of the model increases and the false positives rate decreases for all attacks when four or more patients are used during the training phase of the federated algorithm. Comparing one and eight patients, the detection accuracy increased by 0.5% while FPR decreased by 0.58% on average. Therefore, we can only conclude that we made a statistically significant and positive impact of the detection accuracy for modification attacks. Using two patients caused a slight decrease in detection accuracy and an increase in FPR. This is due to an equilibrium point

being reached in the federated algorithm where each mobile device starts with the federated model, calculates the update, and receives back from the server the same federated model. This scenario is avoided when more patients are registered to a cluster due to the larger selection of mobile devices during the random subset phase (Section IV-B).

Communication cost of the federated algorithm is measured by the number of bytes transmitted to and from the mobile device. With 20 neurons, 7 features, and binary output, we transmit a total of 1,621 bytes per model exchange. During each training round, a contributing patient trains its model and sends the updated weights to the server, and then receives the averaged weights for a total of two transmissions of size 1,621 for a total of 3,242 bytes per round. There is an additional nine bytes of command information received by all patients for each round resulting in a total of 3251 bytes communicated per round. With eight rounds used in FLIDS, our communication cost is 26,008 bytes. Figure 8a presents the comparison of number of bytes communicated versus number of bytes of data used to train the local patient model. With an average size of data at a patient's device equal to 98,034 bytes we save transmitting 72,026 bytes or reduce the communication cost by a factor of 3.8 using the proposed model.

Figure 8b shows the energy consumption of the federated algorithm. We measure the power consumption using USB e-meter multiplied by the number of seconds spent in each phase of the algorithm. The total energy consumption (in Joules) used for training of the federated model was 265 J while energy consumption on the device with no algorithm was 218 J. Thus, the total energy cost of training the federated model was 46 J, a new patient joining the network uses 5.5 J, and testing new samples uses 5.24 mJ. Our FLIDS consumes 17.5% more energy while training and 3.4% more energy than idling when making predictions.

We then evaluated the performance of our proposed federated algorithm through a comparison with non-federated learning algorithms such as Nearest Neighbor (KNN), Decision Trees (DT), Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), Neural Network (NN) as shown in Figure 9. Figure 9a shows the training time for federated and non-federated models trained with eight patient records on a single patient device. The results show that KNN, DT, and SGD classifiers are much more time performant than SVM, neural networks, and the federated algorithm. However, for a new patient entering the federated network, the final training time is about equal to the SGD classifier and 26.8 times faster than the non-federated NN.

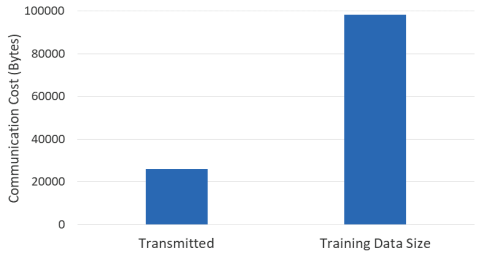
Figure 9b presents a comparison of detection accuracy between federated learning and other learning algorithms for different types of attacks modeled on the system. For the system trained on eight patient records, among the non-federated learning algorithms, we observe that NN has the highest detection accuracy at 90.6% averaged across all attacks. KNN is both accurate and time performant in this scenario with 89.3% and 5.6 times faster than the final federated training round. However, KNN's accuracy is still significantly worse than the federated algorithm that has an additional 7.3% score. Additionally, FLIDS was 6% more accurate on average against the non-federated NN (the best overall non-federated algorithm for detection accuracy). Thus, FLIDS provides the highest detection accuracy while being comparable or better in training time against non-federated algorithms.

Lastly, the scalability of our system was evaluated using the same design, data, and a similar neural network in MATLAB. We recorded the time spent by clients per training round and not the elapsed time at the server. The results can be seen in Figure 10 with the models varying in detection accuracy from 0.9 to 0.98, but all models converge around the same time of 15 seconds. The two patient model took 16.7 seconds to reach round eight while the 64 patient model took 16.9 seconds to reach the same round. Overall, the addition of more patients increases time linearly at the server only while avoiding an increase in a single round's duration or training time.

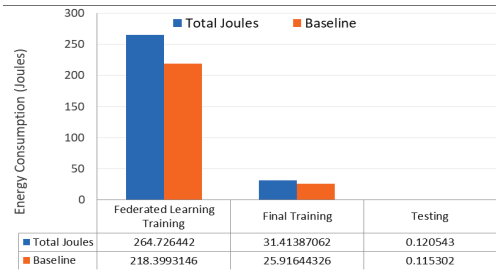
VI. Conclusion

In this work, we implemented a federated Learning based Intrusion Detection System to efficiently detect attacks in Medical Cyber-Physical Systems. Our results indicate that the proposed model improved the detection accuracy and False Positive Rate (FPR) by a statistically significant margin. The low FPR ensures that hospitals spend their staff and resources efficiently while maintaining high quality care for patients. We also observed that the training time for federated model was comparable or better than using a single machine learning instance and training on the same amount of data. Training time for new patients registering with a group were shown to be up to 26.8 times faster than non-federated algorithms. We also showed that adding more patients to the federated model did not increase a round's training time. As a result, we can conclude that a federated learning based IDS can train on more data, increasing accuracy and lowering FPR, while decreasing the amount of time and computation needed of an individual mobile device.

In addition to detection accuracy and lower false positives, our system also ensures privacy because instead of transmitting personally identifiable information of patient across the network only an update vector is communicated. Additionally, size of the update is independent of data size at the mobile device, reducing the network bandwidth by a factor of 3.8. As the amount of communication and computation at each mobile device can be easily configured through parameters, such as the fraction of patients used per round, our scheme has more flexibility in memory restrictive or low bandwidth networks. We note that our proposed solution could be subject to poisoning attacks generated by adversarial machine learning algorithms. Future work includes using Generative adversarial networks in combination with federated learning to provide better protection against adversarial inputs during both training and testing.

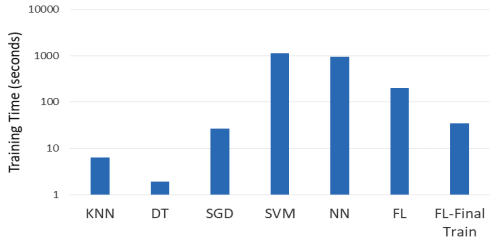


(a) Communication cost of FLIDS versus training data size

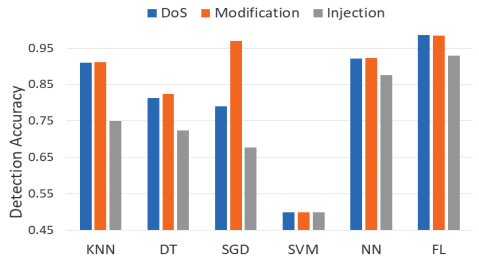


(b) Energy consumption cost of FLIDS

Fig. 8



(a)



(b)

Fig. 9: Comparison of Federated vs. Non-federated algorithms (Patient records=8, $C = 0$)

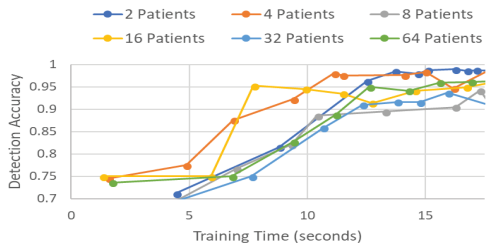


Fig. 10: MATLAB simulation results for C=0

REFERENCES

[1] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-Physical Systems Security – A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.

[2] H. M. J. Almohri, L. Cheng, D. Yao, and H. Alemzadeh, "On threat modeling and mitigation of medical cyber-physical systems," in *CHASE*. IEEE Computer Society / ACM, 2017, pp. 114–119.

[3] K. Zhou, T. Liu, and L. Liang, "Security in cyber-physical systems: Challenges and solutions," *International Journal of Autonomous and Adaptive Communication Systems*, vol. 10, no. 4, pp. 391–408, Jan. 2017.

[4] K. Saleem, Z. Tan, and W. Buchanan, *Security for Cyber-Physical Systems in Healthcare*. Cham: Springer International Publishing, 2017, pp. 233–251.

[5] C. Hu, H. Li, and Y. Huo, "Secure and Efficient Data Communication Protocol for Wireless Body Area Networks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 2, pp. 94–107, 2016.

[6] J.-x. Hu, C.-l. Chen, C.-l. Fan, and K.-h. Wang, "An Intelligent and Secure Health Monitoring Scheme Using IoT Sensor Based on Cloud Computing," vol. 2017, 2017.

[7] C. S. Park, "Security Mechanism Based on Hospital Authentication Server for Secure Application of Implantable Medical Devices," *BioMed Research International*, vol. 2014, 2014.

[8] T. V. P. Sundararajan and A. Shanmugam, "A Novel Intrusion Detection System for Wireless Body Area Network in Health Care Monitoring," *Journal of Computer Science*, vol. 6, no. 11, pp. 1355–1361, 2010.

[9] F. A. Khan, N. A. H. Haldar, A. Ali, M. Iftikhar, T. A. Zia, and A. Y. Zomaya, "A Continuous Change Detection Mechanism to Identify Anomalies in ECG Signals for WBAN-Based Healthcare Environments," *IEEE*, vol. 5, pp. 13 531–13 544, 2017.

[10] L. Coppolino and L. Romano, "Open Issues in IDS Design for Wireless Biomedical Sensor Networks BT - Intelligent Interactive Multimedia Systems and Services," *Smart Innovation, Systems and Technologies*, vol. 6, pp. 231–240, 2010.

[11] S. A. Haque and S. M. Aziz, "False Alarm Detection in Cyber-physical Systems for Healthcare Applications," *AASRI Procedia*, vol. 5, pp. 54–61, 2013.

[12] S. Haque, M. Rahman, and S. Aziz, "Sensor Anomaly Detection in Wireless Sensor Networks for Healthcare," *Sensors*, vol. 15, no. 4, pp. 8764–8786, 2015.

[13] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Sensor Fault and Patient Anomaly Detection and Classification in Medical Wireless Sensor Networks," *IEEE ICC*, pp. 4373–4378, 2013.

[14] R. Mitchell and I.-r. Chen, "Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16–30, 2015.

[15] N. Zakaria, M. I. Sarwar, N. Mustaffa, K. P, T. Wan, and K. Azimi, *Wireless Networks in Mobile Healthcare*. Springer, 2016, no. January 2016.

[16] A. Odesile and G. Thamilarasu, "Distributed Intrusion Detection Using Mobile Agents in Wireless Body Area Networks," in *Seventh IEEE International Conference on Emerging Security Technologies (EST)*, 2017.

[17] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Anomaly Detection in Medical Wireless Sensor Networks using SVM and Linear Regression Models," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 5, no. 1, pp. 20–45, 2014.

[18] P. Kumar and H.-J. Lee, "Security Issues in Healthcare Applications Using Wireless Medical Sensor Networks: A Survey," *Sensors*, vol. 12, no. 12, pp. 55–91, 2011.

[19] B. Biggio, G. Fumera, P. Russu, L. Didaci, and F. Roli, "Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 31–41, 2015.

[20] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated Optimization: Distributed Machine Learning for On-Device Intelligence," in *NIPS*, 2015, pp. 1–38.

[21] A. E. Johnson, T. J. Pollard, L. Shen, L. W. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, pp. 1–9, 2016.