Learning Context-aware Policies from Multiple Smart Homes via Federated Multi-Task Learning

Tianlong Yu[1], Tian Li[1], Yuqiong Sun[2], Susanta Nanda[2], Virginia Smith[1], Vyas Sekar[1], Srinivasan Seshan[1]
[1]Carnegie Mellon University, [2]Symantec

***Abstract*—Internet-of-Things (IoT) devices deployed in smart homes expose users to cyber threats that can cause privacy leakage (e.g., smart TV eavesdropping) or physical hazards (e.g., smart stove causing fire). Prior work has argued that to effectively detect and prevent such threats, contextual policies are needed to decide if access to an IoT device should be allowed. Today, however, such contextual access control policies need to be manually generated by IoT developers or users via preinstallation or runtime prompts. Both approaches suffer from potential misconfigurations and often fail to provide coverage over the space of policies. In this paper, our goal is to build a machine learning framework to automatically learn the contextual access control policies from the observed behavioral patterns of users in smart homes. Designing such a learning framework is challenging on two fronts. First, the accuracy is constrained by *insufficient data* in some smart homes and the *diversity* of IoT access patterns across different smart homes. Second, since we rely on usage patterns of IoT devices, users will have *privacy concerns*. We address these challenges in designing LoFTI, a *federated multi-task learning* framework that learns customized context-aware policies from multiple smart homes in a privacy-preserving manner. Based on prior user studies, we identify six general types of features to capture contextual access patterns. We build a simple machine learning model with temporal structure to achieve a good trade-off between accuracy and communication/computation cost. We design a custom data augmentation mechanism to address the issue of unbalanced data in learning (i.e., few negative vs. normal samples). We show that LoFTI can achieve low false positives/false negatives, reducing the false negative rate by 24.2% and false positive rate by 49.5%, comparing with the state-of-the-art single-home learning and all-home learning mechanism.**

I. Introduction

While Internet-of-Things (IoT) devices are widely deployed and used in smart homes to benefit people's lives, these devices can also pose severe threats due to their ability to interact with the physical environment (especially for families with baby, people with disabilities, or elderly people who live alone). An increasing number of reported IoT breaches [1], [3], [4], [5], [6], [7], [8], [9], [10] have shown that devices, such as smart TV/camera/heater/stove/window/door/light/speakers, can cause privacy leakage and physical hazards in smart homes. For example, studies show that Samsung smart TVs are susceptible to an eavesdropping attack (privacy leakage) called Weeping Angel [5] and that smart stoves can be remotely controlled to cause fires (physical hazards) [3].

To prevent privacy leakage and physical hazards, prior work [11], [12], [17], [19], [21] has argued the need for contextual information to decide if an access to an IoT device should be allowed or blocked. Today, such contextual access control policies [11], [12], [17], [19], [21] are preinstalled by developers/users or created via runtime prompts. However, such manually generated policies are easy to misconfigure and fail to provide *coverage* over all the contextual policies needed. For example, multiple contextual variables, including temperature, humidity, CO_2 and whether the user is asleep, can impact the policy for opening the window in a smart home. If we rely on developers or users to preinstall the contextual policies, it is hard to determine key factors for a particular smart home. Similarly, runtime prompts with multiple contextual information will place a heavy burden on the users as users need to answer such prompts each time the context changes (e.g., temperature/humidity changes).

A promising alternative is to build a machine learning framework to automate and simplify the process of generating contextual access control policies for IoT devices. The machine learning framework will learn the contextual policies from the historical records of contextual IoT access, and use the learned contextual policies to decide to allow or block a particular IoT access in the future.

However, there are two goals that are hard to simultaneously achieve in designing such a machine learning framework: *accuracy* and *privacy*. To see why, consider two strawman solutions. On one extreme, if each home relies on its own historical records (*single-home learning*), some smart homes may have insufficient data to learn an accurate contextual access control model. On the other extreme, to avoid this insufficient data issue, we could collect all the historical records of all smart homes in a centralized place (e.g., AWS/Azure/Google cloud) and learn one model (*all-home learning*). However, such all-home learning may not be accurate since different smart homes can have diverse contextual access patterns. Also, this all-home learning mechanism requires the historical data of a smart home to be transferred out of the smart home (e.g., to the cloud), and will raise users' privacy concerns.

To address the *insufficient data* issue, *diversity* issue and *privacy concerns*, our insight is that we can leverage a recent advance in distributed machine learning - Federated Multi-Task Learning (FMTL) [22]. To address the *insufficient data* issue, the multi-task learning mechanism can learn the common part of the contextual policies from the historical records from all homes. To address the *diversity issue*, the multi-task learning mechanism can learn the unique part of the contextual policies for each smart home by assigning a customized learning task to each home. To address the *privacy concerns*, we leverage the *federated learning* mechanism. At

978-1-7281-6602-5/20/$31.00 ©2020 IEEE
DOI 10.1109/IoTDI49375.2020.00017

a high level, a federated learning mechanism is an iterative learning process. In each iteration, federated learning performs a local learning step on the raw data within each smart home, then synthesize the model parameters obtained in each local learning to build the contextual access control model at a centralized place (e.g., cloud backend). Since the raw historical data is processed locally within the smart home and is never sent out, it offers better data privacy.

In this paper, we design LoFTI- a framework to Learn contextual access policies via Federated multi-Task learning for IoT devices in multiple smart homes. LoFTI includes an edge computing node in each smart home and a centralized backend server. The basic workflow of such a learning framework includes four steps. In the first step, each edge computing node will collect the IoT access and contextual information records to *construct a historical dataset*. In the second step, the edge computing node will extract a set of key features from the dataset to capture the contextual IoT access patterns. Then, the learning framework will build a machine learning model and push the model to the edge computing nodes and the centralized backend server. After that, the learning framework will use the federated multi-task learning to learn the model across the edge computing nodes and centralized backend server in a distributed and iterative manner.

However, there are several practical challenges in designing LoFTI. The first challenge is how to define a set of key features for contextual IoT access, given a wide range of information that can be collected in smart homes. The second challenge is how to build a ML model with high *accuracy* and low *communication/computation cost*, as complex models tend to be more accurate but need to compute and transmit more model parameters across the edge computing nodes and the backend server. The third challenge is, since the number of benign access is often much greater than the number of malicious access, how to address such unbalanced dataset.

We explore the design space of LoFTI in terms of feature extraction, model building and dataset construction and address the challenges above. We generalize 6 types of key features to capture the contextual IoT access patterns and demonstrate their effectiveness in distinguishing benign/malicious IoT access. We build a simple SVM-based model with extra temporal structures to achieve high detection accuracy and low communication/computation cost. We provide a specialized data augmentation mechanism to balance the contextual access historical record dataset. We show that LoFTI framework can achieve high accuracy (FNR is 0.024 and FPR is 0.086) while addressing user's privacy concerns by guaranteeing that raw IoT data will be stored locally within each smart home.

II. Motivation

In this section, we first walk through a number of IoT breaches causing privacy leakage and physical hazards. Then we demonstrate why contextual policies are needed to prevent such breaches. Finally, we show why the current approaches that manually generate contextual policies are prone to misconfiguration and lack coverage.

IoT devices are susceptible to privacy leakage and physical hazards: Table I shows a list of recently reported IoT breaches. A number of breaches are related to privacy leakage. For Nest cameras, the attacker can steal the private video/image and demand subscribe. For Samsung smart TVs, the attacker can remotely turn on audio recording (via backdoor access) for eavesdropping. Other breaches can cause physical hazards in smart homes. For example, the attacker can open smart doors or smart windows to break into the house. Besides, the attacker can control heaters to overheat the bedroom, or control the stove to cause fire, or remotely control smart toilets to cause water overflow, especially dangerous for families with babies, people with disabilities and the elderly. An interesting case shows that the attacker can even use a drone to turn off smart light to cause a blackout of a house. From these breaches we can see that a wide range of IoT devices are being used to cause privacy leakage or physical hazard. This is due to the fact that IoT devices can interact with the physical environment around them.

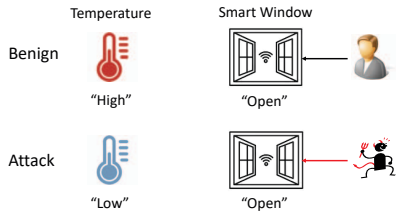

Fig. 1: Benign/malicious smart window scenarios.

Contextual policies are needed: Another interesting observation from the above IoT breaches is that- from a single device perspective, the attackers are accessing these IoT devices similar to how a normal user access these IoT devices. For example, the attacker can turn on the camera or open the smart window just like a normal user. Therefore, it would be hard to distinguish attacker's access from user's access if we only monitor a single device. To prevent privacy leakage and physical hazards, current approaches [11], [12], [17], [19], [21] rely on access control policies with contextual information to decide if an action of an IoT device should be allowed or blocked. In Figure 1, we present a motivating example to show how contextual policy can prevent physical hazards or privacy leakage. A user will access and open the smart window when temperature is high. The attacker, however, will not follow such benign *context* -"temperature is high" and will open the window to break into the house even when the temperature is low. Therefore, if there is a contextual policy that only allows the access to open the window when the temperature is high, this policy is able to prevent the attack.

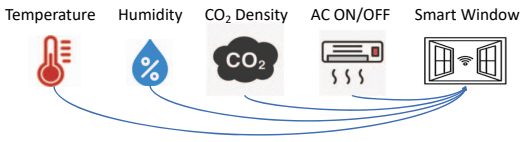

Fig. 2: Complex interactions across smart windows, AC units and the environment.

Device	IoT Breach	Attack Type
Nest Camera	Attacker steal private video/image and demand subscribe [10]	Privacy leakage
Multiple Smart Doors	Attacker open smart door to break into the house [4]	Physical hazards
Multiple Smart Heaters	Attacker can control the heater to overheat the bedroom [8]	Physical hazards
Samsung Smart TV	Attacker can remotely turn on audio recording for eavesdropping [5]	Privacy leakage
Multiple Smart Stoves	Attacker can control the stove to cause fire [3]	Physical hazards
Multiple Smart Window	Attacker can open the smart window to break into the house [7]	Physical hazards
Philips Hue Smart Light	Attacker can use drone to turn off smart light to cause a blackout [6]	Physical hazards
Multiple Smart Toilets	Attack can remotely control smart toilet to cause water overflow [9]	Physical hazards

TABLE I: Recently reported IoT breaches.

Current approaches to generate contextual policies are easy to misconfigure and lack coverage: There are two approaches to generate contextual policies. The first approach is to *preinstall* contextual policies specified by developers or users [17], [21]. The second approach is to use *runtime prompts* with contextual information to let the user make the allow/block decision [15], [17], [19]. Both approaches heavily relies on the knowledge and effort of the developers/users to anticipate the malicious behaviors. Thus is easy to misconfigure and have poor coverage over all contextual policies needed, causing high FPs/FNs. For example, in Figure 2, whether the smart window will be opened depends on multiple factors include temperature, humidity, CO_2 density and whether AC is on/off, and such context can be different between smart homes. Suppose a handcrafted policy is preinstalled to block the access to the smart window when the temperature is above preconfigured thresholds, such thresholds can be easily misconfigured and result in high FPs/FNs. Then, suppose there are runtime prompts with temperature, humidity, CO_2 density and whether AC is on/off, such context can be different every time the smart window is accessed. Considering the number of IoT devices in a smart home, there could be hundreds of prompts per day and cause the user to be unable or unwilling to handle all the prompts.

III. System Overview

In this section, we present an overview of our learning framework - LoFTI. We start by defining the threat model.

Threat model: The goal of the attacker is to access IoT devices to cause privacy leakage or physical hazards. As shown in Figure 3, the IoT devices have user APIs (e.g., open/close window) for common usage and developer APIs (e.g., modify the firmware) for developing functionality. We make the following assumptions about the attacker's capability:

- The attacker can access all the user APIs (e.g., via weak/flawed authentication or backdoor).
- The attacker cannot access the developer APIs to modify the firmware (e.g., via code injection). Such access is easy to distinguish from user access and can be prevented by network defense. Therefore, the attacker cannot spoof the status of the IoT device (e.g., report an opened window as closed) as this requires firmware modification.

Our goal: We aim at learning contextual access control policies from the historical records in smart homes. Suppose the history of IoT accesses with the corresponding context and action is given (e.g., temperature, humidity, the state of the AC and the actions performed on the smart window in Figure 4). The learning mechanism will learn a function F, so

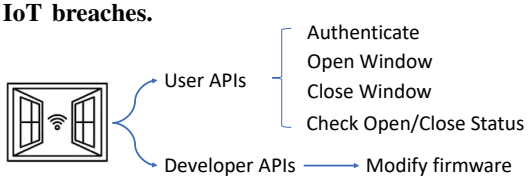

Fig. 3: Two different levels of attacker's capabilities.

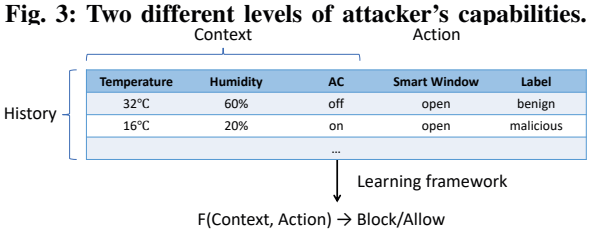

Fig. 4: The goal of the learning framework.

that the F can decide to allow or block an access in the future based on the corresponding context and action. For example, in Figure 4, the history shows that the user will open the window when the temperature and humidity are high (i.e., feels hot), and attack will open the window even when the temperature and humidity are relatively low. Then our goal is to learn function F to decide to block/allow the access to open the window based on the temperature and humidity in the future.

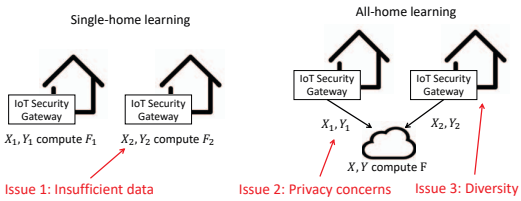

Fig. 5: Two strawman solutions and the issues.

Based on the goal of the learning framework, there are two strawman solutions - single-home learning and all-home learning, as shown in Figure 5.

Strawman 1 - Single-home learning: This strawman solution let each home use its own historical records to learn its own contextual policies. However, some smart homes may have *insufficient data* to effectively learn the contextual policies. For example, in Figure 6, there are two smart homes with smart windows. The user will open the smart window when feels hot in terms of high temperature and humidity. The attacker, however, will open the smart window even when temperature and humidity are low. Each point in Figure 6 is a benign access (blue point) or malicious access (red point) to the smart window. The first row shows the training result and second row shows the testing result for the single-home learning mechanism. In this case, each home learns its own model (the blue line). Also, we can see that all the blue and

red points under the line in the second row for home 2 are misclassified, showing that the FPs and FNs for home 2 is high. This is because home 2 suffers from *insufficient data* with only 12 data points for training, while home 1 has 60 data points for training.

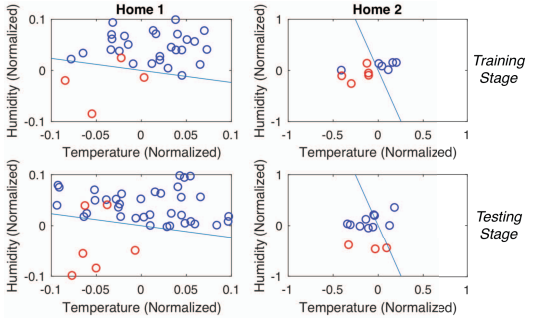

Fig. 6: Insufficient data issue.

Strawman 2 - All-home learning: Another strawman solution is to use the historical records of all homes to learn one contextual access model. The issue for such all-home learning mechanism is that some smart home may have *diverse* contextual access patterns. For example, in Figure 7, we use the same dataset as described in the single-home learning case. The *diversity* here is that, comparing with the user in home 1, the user in home 2 has less tolerance to hot, and will open the window at a lower temperature/humidity. However, the all-home learning mechanism will learn the same decision boundary for home 1 and home 2. Therefore, the contextual policy (decision boundary) learned lacks customization and result in high FPs for home 2, as shown in the second row of Figure 7.

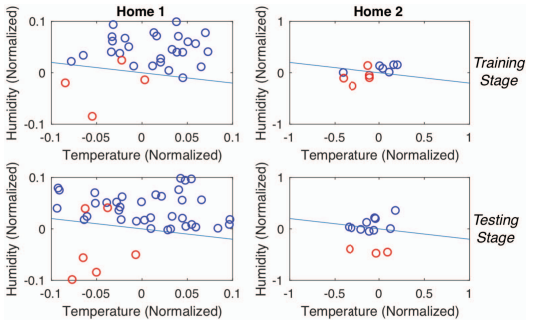

Fig. 7: Diversity issue.

Our insight: We can apply Federated Multi-Task learning [22] to address the above *insufficient data*, *privacy* and *diversity*. The high-level process of Federated Multi-Task learning is shown in Figure 8. In step 0, the cloud backend will initiate the model F and task correlation Ω. In step 1, the cloud backend will send the model parameter w_i to each smart home i. In step 2, the edge node in smart home i will calculate the local model parameter update Δw_i based on local data X_i and model parameter w_i. In step 3, the edge node in smart home i will send the model parameter update Δw_i back to the cloud backend. In step 4, the cloud backend will update the task correlation Ω and the model F based on updates from all smart homes $\{\Delta w_i\}$. Then step 1, 2, 3 and 4 will be executed for multiple iterations until the model F converges.

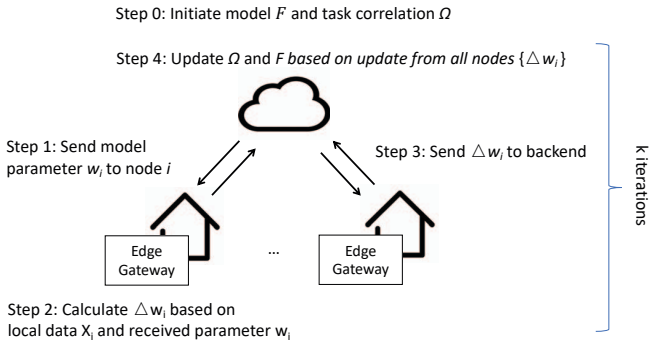

Fig. 8: Federated Multi-Task learning workflow.

Federated Multi-Task learning can address *insufficient data* issue because the model F is learned from the data $\{X_i\}$ from each smart home in a distributed manner. The *diversity* issue is addressed as each smart home is considered as a individual learning task, and the task correlation Ω can capture the diverse correlations (e.g., distinguishing common part and unique part) and allow each task to be customized. There will be less *privacy concerns* for the user as the raw data X_i for smart home i is kept locally within each smart home, only model parameters $\{\Delta w_i\}$ is transmitted to the cloud backend.

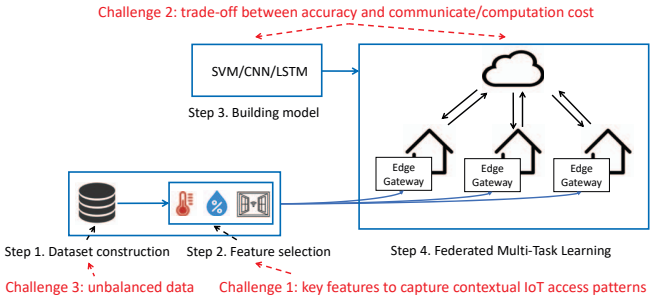

Fig. 9: Basic building blocks for LoFTI.

In this paper, we design LoFTI- a framework to Learn contextual access policies via Federated multi-Task learning for IoT devices in multiple smart homes. LoFTI includes an edge computing node (e.g., an IoT gateway) in each smart home and a centralized backend server. There are several building blocks for LoFTI, as shown in Figure 9. The first building block is the *dataset construction* process for each edge computing node to collect the IoT access and contextual information records. The second building block is the *feature extraction* process for each edge computing node to extract a set of key features from the historical record dataset. The third building block is *building the ML model* and push the model to the edge computing nodes and the centralized backend server. After that, the learning framework will use the federated multi-task learning to learn the model across the edge computing nodes and centralized backend server in a distributed and iterative manner.

Challenges: There are several challenges in designing LoFTI:

Defining key features to extract contextual IoT access patterns: Given the wide range of information in smart homes, it is hard to define a set of key features that are expressive enough to extract various contextual IoT access patterns. For example, a naive solution may use MAC/IP address of an IoT devices as a key feature, which are not effective and similarity in IP address can even be noise to the learning framework.

Building model with high accuracy and low computation/communication cost: More complex ML models (CNN, RNN, LSTM) often have higher accuracy as they can encode more information than less complex models (Naive Bayes, SVM). However, complex machine learning models also result high computation/communication cost, especially in a distributed learning framework like LoFTI. It is hard to achieve a good trade-off between accuracy and cost.

Addressing unbalanced contextual IoT access records: In reality, the historical records for privacy leakage/physical hazards are much fewer comparing with benign IoT access records. For example, there can be more than 10000 records in 100 smart homes about user opening the smart window in the historical records, with less than 100 records about attacker opening the smart window. As a result of such unbalanced dataset, the decision boundary will be pushed towards the attack space and result in high FNs.

We will show how LoFTI addresses the above challenges in Section IV.

Adversary against LoFTI: In this part, we discuss the potential attacks against LoFTI itself. An attacker can be inside the cloud backend or perform man-in-the-middle attack to observe the data sent to the cloud backend. For such attackers, LoFTI's federated learning mechanism ensures that the attacker is not able to observe the raw IoT data in smart homes. An attacker can be inside a smart home and perform adversarial machine learning techniques trying to corrupt LoFTI's model. For such attackers, LoFTI's multi-task learning mechanism ensures that each smart home will learn a customized model so the corrupted data in one smart home will have little impact on other smart homes. An attacker may be able to observe the model parameters sent to the cloud backend and infer private information from the model parameters. In this paper, we show that many sensitive IoT information (e.g., device location, firmware version) can be masked without impacting LoFTI's detection accuracy (Section V). To fully address this type of attacker, we envision that techniques such as *differential privacy* can be incorporated with LoFTI in the future. Finally, we assume that there are mechanisms that prevent the model parameters being modified by attacker and the edge computing node is secured.

IV. LoFTI Design

In this section, we discuss the design of LoFTI. Specifically, we discuss (1) how to select key features that are expressive enough to capture various contextual IoT access patterns, (2) how to build a learning model to achieve a good trade-off between accuracy and computation/communication cost; and (3) how to construct contextual IoT access dataset to address the unbalanced dataset issue.

A. Defining key features for contextual IoT access

LoFTI framework extracts a set of key features from the historical record dataset to capture the contextual IoT access patterns. Given a wide range of information we can collect from IoT devices, it is challenging to identify the salient features.

To address this challenge, we leverage the concept of *contextual integrity* (*CI*) [20], which claims that appropriate information flows (i.e. IoT accesses, in our case) are the ones that conform with the *contextual norms*. Here *contextual norms* refer to five independent parameters: data subject, sender, recipient, information type, and transmission principle. Now the issue is how to define the *contextual norms* that should regulate IoT accesses.

Previous efforts [11], [12], [14], [15], [19], [21], [24] have manually crafted different sets of contextual norms and have shown that they are closely related with IoT accesses; however, their definitions have not been very consistent. In this paper, we summarize and generalize these *contextual norms* into six types:

User: The *user* context norm is defined as the role of the person who is accessing the IoT device. Roles can be *parent*, *child*, *household employee* (e.g. a babysitter), or *visitor*. The *user* context norm corresponds to the *sender* in the *CI*, capturing who initiated an IoT access.

Device: The *device* context norm is a combination of device type and device instance identifier. Device type is a combination of device class (e.g., camera), device vendor (e.g., D-Link) and device SKU (Stock Keeping Unit, e.g., DCS-932L). The *device* context norm corresponds to the *recipient* in *CI*, capturing which device is being accessed.

Time of the day: The *time of the day* context norm is a combination of hour of the day, minute of the hour and second of the minute. This context norm corresponds to the *transmission principle* in *CI*, capturing the time condition under which the IoT access should be allowed.

Environment variables: The *environment variables* context norm is a list of variables describing the physical environment of a smart home, such as temperature, humidity, CO_2 density, etc. This context norm corresponds to the *transmission principle* in *CI*, capturing the environmental conditions under which the IoT access should be allowed.

Device states: The *device states* context norm is a list of variables describing the states of the IoT devices. Examples of such states include *smart window opened*, *camera turned on*, and *smart oven turned off*. This context norm also corresponds to the *transmission principle* in *CI*, capturing the conditions of device states under which the IoT access should be allowed.

Action: The *action* context norm is an operation to be performed by the IoT device by the access, such as *open* the smart window and *turn on* the camera. This context norm corresponds to the *data subject* in the *CI*, capturing the subject of the IoT access.

Let's denote the *user* as u, the *device* as d, the *time* as t, the *environment variables* as $\{e\}$, the *devices' states* as $\{s\}$ and the *action* as a. Then the context norms for LoFTI can be expressed as $(u, d, t, \{e\}, \{s\}, a)$.

Next, we demonstrate the expressiveness of LoFTI's contextual norms by showing a list of contextual IoT access patterns LoFTI can capture, as shown in Table II. In the first scenario, a visitor watches a video recorded on the camera (privacy leakage). In this case, the three key features – the user's role as a *visitor*, the device *camera*, and the action of *playing the recorded video* – are captured. In the second scenario, a family member takes shower at night, and the key feature *time* is captured. Similarly, in the third scenario – opening a window when it is hot – the key features temperature and humidity are captured. The fourth scenario covers a cooking activity and the key features – the states of microwave oven and fridge – are captured.

B. Building a model with high accuracy and low computation/communication cost

Considering the distributed learning nature of federated multi-task learning, building a machine learning model for contextual IoT access raises challenging trade-offs between *model accuracy* and *computation/communication cost*.

To understand these trade-offs, we start by exploring the design space for building a machine learning model. At a high level, the machine learning model can be considered as a function F that takes the contextual norms $(u, d, t, \{e\}, \{s\}, a)$ as input x and outputs a binary decision, i.e., whether to block or allow the action a given the context norms.

There are several options to build such a function F, ranging from a simple SVM model to more complex CNN (Convolutional Neural Network) and LSTM (Long Short-Term Memory) models. We discuss how these models perform in terms of *model accuracy* and *computation/communication cost* in detail below.

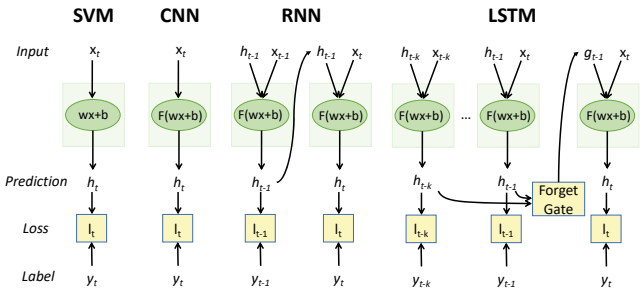

Fig. 10: Comparing the structures of different machine learning models.

To understand the trade-off, it is useful to look into the structures of different machine learning models, as shown in Figure 10. Given input x_t at time t, a (linear) SVM model makes a prediction h_t with a simple function F as $wx + b$. The model then minimizes the loss l_t between the predictions $\{h_t\}$ and labels $\{y_t\}$ by adjusting the parameters w and b in the learning stage to train an accurate function F. CNN models also follow this process, except that they adopt a more complex function $F = f(...f(wx + b))$, where F is a composition of multiple (possibly) non-linear functions (e.g., a sigmoid function). The issue with SVM and CNN is that they have no "memory", i.e. the prediction h_t completely depends on input x_t at the current time t, and cannot encode patterns in history. In contrast, RNN models use previous prediction h_{t-1} as an input (along with other input x_t) to compute the current prediction h_t, thus capturing the sequential patterns in history. However, this model "forgets" very fast as the impact of prior predictions decrease exponentially [18] with each iteration. LSTM addresses this issue by adding a *forget gate* to encode a wide range of predictions h_{t-k} to h_{t-1}. The forget gate can learn to determine how much impact a previous prediction h_{t-i} should have on the current prediction h_t.

Accuracy: Complex models such as LSTM are relatively more accurate as they are able to encode more information than simpler models such as SVM; and more generally, as the models get more complex, their accuracy gets better. However, it is not clear how much information needs to be encoded for contextual IoT access patterns to achieve a satisfying level of accuracy.

Computation/communication cost: Since federated multi-task learning works in a distributed manner, different machine learning models will have different impact on the computation and communication cost. The computation cost includes the CPU and memory consumption on each edge computing node and on the cloud back-end server. The communication cost includes the bandwidth consumption and latency introduced when transmitting the model parameters between the edge computing nodes and the cloud back-end server.

To achieve high accuracy and low communication/computation cost, we can build a simple model with an additional structure that captures the key contextual IoT access patterns. A simple model helps reduce communication/computation cost and a carefully designed additional structure to capture the key contextual IoT access patterns, cuts down the accuracy gap.

More specifically, from Figure 10, we observed that the main difference between a simple model (SVM) and a complex model (RNN and LSTM) is how the temporal information (long-term memory and short-term memory) is captured. The key question here is how much temporal information is needed for capturing contextual IoT access patterns. To answer this question, we make two domain-specific observations:

Short-term correlation is needed to capture contextual IoT access patterns: We observe that all the device states and some discrete environment variables at a given time t are highly correlated with the corresponding device states and discrete environment variables at a time $t-1$. For example, if a camera state transits from `off` to `on`, this short-term correlation from time $t-1$ to time t is needed. The intuition behind this observation is that all the device states and some discrete environment variables are internally following certain state machines, and in a state machine, the next state is highly dependent on its previous state.

Long-term correlation can be aggregated: For environ-

Pattern	u	d	t	$\{e\}$	$\{s\}$	a
Visitor watch camera video record	visitor	camera	...	...	...	play_video_record
Take shower at night	family	shower	21:35:24	...	...	turn_on_shower
Open window when feeling hot	family	window	...	temperature=$35^\circ C$; humidity=60%	...	open_window
Use stove when cooking	family	stove	...	...	microwave_on; fridge_opened	turn_on_stove

TABLE II: Exemplar contextual IoT access patterns that can be captured by LoFTI's contextual norms.

ment variables and device states, capturing the specific long-term sequence in history is not necessary. The aggregated environment variables and device states can be as effective. For example, considering the contextual access pattern that "a user feels hot in the last hour and opens the smart window". Here, the action of open the smart window is only related with the average temperature and humidity in the last hour, not specific temperature or humidity sequence.

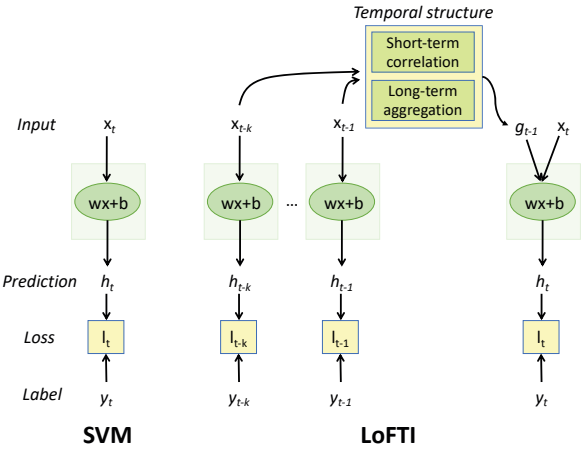

Fig. 11: LoFTI model.

Based on the two observations above, our idea is to add an extra temporal structure on top of simple SVM model, as shown in Figure 11. The extra temporal structure captures the short-term correlation and long-term aggregation. The detailed pseudo code of the temporal structure is given in Figure 12. To capture the short-term correlation, we apply *word2vector* processing on device states and discrete environment variables from time $t-h$ to time t, and generate vectors of *2-grams* to capture the short-term correlation between two adjacent times (Line 10 in Figure 12). To capture the long-term correlation, we aggregate the continuous environment variables from time $t-h$ to time t by its average value (Line 13 in Figure 12), and we aggregate the discrete device states and environment variables from time $t-h$ to time t by the appearance number of *2-grams* (Line 10 in Figure 12). In this way, LoFTI's temporal structure captures the short-term correlation and aggregated long-term correlation.

Next, we compare LoFTI and LSTM in terms of computation cost and communication cost. In distributed machine learning, the computation cost is measured by the number of CPU FLOPS, and the communication cost is measured by the number of model parameters transmitted in each iteration.

We start with LSTM's computation cost. In LSTM, there are 8 matrix-matrix multiplications per layer per time-step. Suppose N is the number of features, M is the hidden unit size, L is the minibatch size. Each matrix-matrix multiplication

```
1  Input: {x_{t-h}, ..., x_{t-1}, x_t} is the sequence of features
2       t is current time, h is history time window
3       m is the number of key features
4  {env} is the environment variable set
5  {state} is the environment variable set
6  Output: g_t is the temporal aggregation
7  for i ← 1 to m
8    if x_{t,i} ∈ {env} or x_{t,i} ∈ {state}
9      if x_{t,i} is discrete
10       g_{t,i} = word2vector({x_{t-h,i}, ..., x_{t-1,i}})
11     end
12     if x_{t,i} is continuous
13       g_{t,i} = mean({x_{t-h,i}, ..., x_{t-1,i}})
14     end
15   end
16 end
```

Fig. 12: LoFTI's temporal structure.

is $A \times B$, where A is a matrix of size $M \times N$ and B is a matrix of size $N \times L$. The matrix-matrix multiplication $A \times B$ uses $ML(2N-1)$ CPU FLOPS. Suppose r is the number of layers and h is the number of timesteps. Then, the total FLOPS for LSTM is $r * h * 8 * ML(2N-1)$.

Next we calculate the FLOPS for LoFTI. We know that the total FLOPS for a linear SVM is proportional to the number of parameters. Suppose N is the number of features and h is the number of timesteps for long-term aggregation for LoFTI (we set it equal to the LSTM's timesteps). In LoFTI's model, it takes $N * h$ FLOPS to calculate the temporal structure and $2 * N$ FLOPS to calculate the model parameters. Therefore, the total FLOPS for LoFTI is $2*N+N*h$. It is obvious that LoFTI's computation cost is much less than LSTM.

Next, we compare the communication cost of LoFTI and LSTM. Since FMTL only transmit parameters, the communication cost is propotional to the number of model parameters. The number of model parameters of LSTM is given by: $4*(N*M+M^2)$, where N is the number of features and M is the number of hidden unit. The number of model parameters of LSTM is given by: $2*N$, where N is the number of features. Therefore, it is obvious that the communication cost of LoFTI is lower than LSTM.

In summary, LoFTI reduces the computation cost by applying a simple model with a concise temporal structure. We will provide an evaluation comparison of LoFTI and LSTM in terms of the computation/communication cost in Section V.

C. Addressing unbalanced contextual IoT access records

A straight forward approach to construct the dataset is to collect the benign and malicious contextual IoT access records as it is. However, while benign users are frequently accessing the IoT devices, attackers only need a few accesses to cause privacy leakage or physical hazards. This means that the number of benign IoT accesses will be much larger than the number of malicious IoT accesses, i.e., the dataset is unbalanced. Therefore, the decision boundary will be pushed

towards the attack samples and causing high FNs. A straw-man solution to balance the dataset would be to copy the attack samples and construct a repetitive dataset. However, the repetitive attack samples cannot capture the variation of attacks and fails to provide coverage. A general approach in machine learning to address unbalanced data is called *data augmentation* [25]. The idea is to introduce variance in the samples to balance out data and improve coverage. However, common data augmentation mechanisms, including cropping, rotation and flipping are closely tied to image representation (as it has originated from computer vision community), and are not applicable for processing the contextual access abstraction for IoT devices.

Therefore, we designed a customized *data augmentation* approach for processing the contextual access abstraction for IoT devices. Specifically, we provide two mechanisms:

Context random sampling: The context random sampling mechanism is inspired by the cropping of an image. But instead of selecting a region of the image (considering locality), our mechanism selects a subset of the environment variables and device states with a fixed ratio r (commonly set to 10% to 20% in sampling papers [16]). The idea is to perform random sampling on the environment variables and device states to narrow down the correlation between environment variables/device states and IoT access with certain action.

Adding contextual noise: The second mechanism is adding noise to the context. This is similar to adding noise to an image, but instead of adding noise to pixels, LoFTI adds noise to the environment variables of the attack sample. The idea is to increase the coverage of environment variables in attack samples. However, deciding how much noise to add can be a challenge. If too much noise is added, the attack sample may deviate to the benign space and cause FNs. If too little noise is added, the coverage will be low. To address this issue, we provide a mechanism to automatically decide the amount of noise being added. The first step is to estimate a rough decision boundary L. Then, for each attack sample, we calculate the distance metric d to the rough boundary. LoFTI ensures that the noise added to the attack sample does not exceed ϵd, where ϵ is a constant within $(0, 1)$ range. In this way, LoFTI provides a heuristic-based method to ensure that the noise is bounded and can increase the coverage without increasing the FNs. The effectiveness of LoFTI's data augmentation mechanism is shown in Section V.

V. Evaluation

In this section, we evaluate LoFTI and show that:

- LoFTI achieves high detection accuracy and low FPs and FNs when compared with single-home learning and all-home learning, reducing FNR by 24.2% and FPR by 49.5%.
- LoFTI achieves high coverage over the contextual policies needed when compared with current approach to manually generating contextual policies.
- LoFTI's key features are effective for detecting malicious IoT access causing physical hazards or privacy leakage.

Cases	Device	Attack	Type
1	Camera	Private video leakage [10]	privacy leakage
2	Smart Door	Break-in when user is asleep [4]	physical hazards
3	Heater	Bedroom overheating [8]	physical hazards
4	Smart TV	Eavesdropping [5]	privacy leakage
5	Stove	Fire hazards [3]	physical hazards
6	Smart Window	Break-in [7]	physical hazards
7	Smart Light	Blackout [6]	physical hazards
8	Smart Toilets	Water overflow [9]	physical hazards

TABLE III: Test cases for malicious IoT access based on reported IoT breaches.

- LoFTI's simple SVM-based model with temporal structure can achieve a good trade off between accuracy and communication/computation cost.
- LoFTI's data augmentation effectively addresses the unbalanced data issue.
- LoFTI can detect new attacks not in the dataset, if there are historical records about the IoT device being attacked.

A. Dataset

To evaluate LoFTI, we use a public large-scale IoT historical record dataset - CASAS [2] with real IoT deployment in more than 400 smart homes in cities including Kyoto, Paris, Milan, etc. The type of IoT devices in the dataset including smart TV/DVD, smart oven/fridge/microwave, smart door/sofa/bed/cupboard, motion/power/temperature sensors, etc. The CASAS dataset includes historical records about how normal human users are interacting with these IoT devices in typical smart home setting. To test how well LoFTI can detect physical hazards and privacy leakage, we constructed a list of physical hazards and privacy leakage cases based on IoT breaches in news reports [1], [3], [4], [5], [6], [7], [8], [9], [10], as shown in Figure 13. We add these cases randomly to each of the 400 smart homes.

B. Accuracy

In this part, we measure the *accuracy* of LoFTI, and compare LoFTI with the state-of-the-art single-home learning approach and all-home learning approach.

The *accuracy* is evaluated by the FNR (False Negative Rate), FPR (False Positive Rate) and F-Score. The FPR describes the occurrence of False Positives, defined as $FPR = \frac{FP}{FP+TN}$. The FNR describes the occurrence of False Negatives, defined as $FNR = \frac{FN}{FN+TP}$. The F-Score reflects the detection performance combing FP and FN, defined as $FScore = \frac{2*Precision*Recall}{Precision+Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

Approach	FNR	FPR	F-Score
Single-home	0.0832	0.1697	0.8867
All-home	0.0318	0.3409	0.8534
LoFTI	0.0241	0.0857	0.9378

TABLE IV: Overall.

Table IV shows the overall FNR, FPR and F-Score comparing single-home learning, all-home learning and LoFTI. The overall result shows that LoFTI achieves low FNR/FPR and high F-Score, reducing FNR by 24.2% and FPR by 49.5% when compared with the best results from single-home learning and all-home learning approaches.

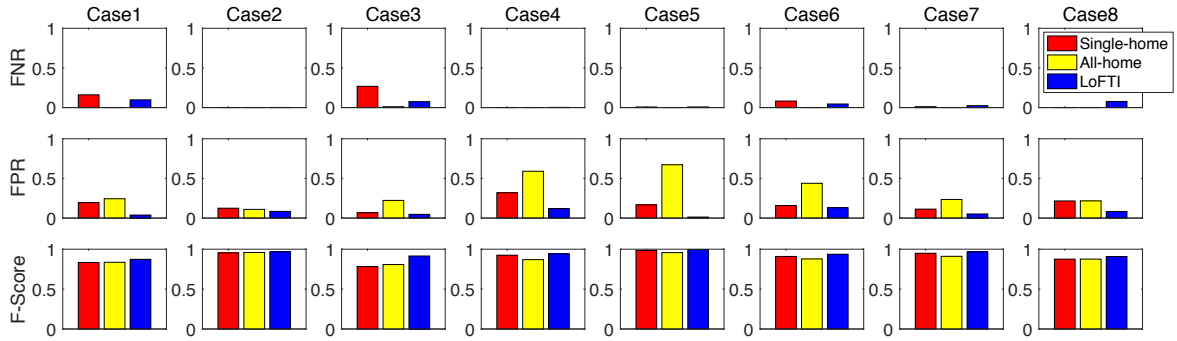

Fig. 13: Comparing single-home learning, all-home learning and LoFTI.

Next, we compare single-home learning, all-home learning and LoFTI case by case. Figure 13 shows the FNR, FPR and F-Score for each case in Table III for single-home learning, all-home learning and LoFTI. The first row depicts the FNR, the second row depicts the FPR for different cases. The last row depicts the F-Score combining the impact of FP and FN. From the result, we can see that single-home learning result in high FNR in case 1, 3 and 6. All-home learning result in high FPR in case 1, 3, 4, 5, 6, 7 and 8. From Figure 13, we can see that single-home learning and all-home learning either result in high FNR or high FPR, and LoFTI achieves low FPs and FNs across all the test cases.

Case	1	2	3	4	5	6	7	8
LoFTI >All>Single	✓	✓	✓					
LoFTI >Single>All				✓	✓	✓	✓	✓

TABLE V: Detailed analysis for the test cases.

Table V shows the detailed analysis for the test cases. We rank single-home learning, all-home learning and LoFTI by the F-Scores they achieve in each case. For case 1,2 and 3, LoFTI achieves higher accuracy (F-Score) than all-home learning than single-home learning. This is because the *insufficient data* issue has a major impact in these cases. To confirm this, we counted the number of IoT access records relevant for distinguishing malicious/benign access in each home (e.g., if malicious access is turn on Camera, we count all the access to Camera). We found that, in case 1,2 and 3, the average relevant IoT access records in each smart home is less than 200, and some smart home have less than 50 relevant IoT access records. This explains the *insufficient data* issue in case 1,2 and 3. For case 5 to 8, LoFTI achieves higher accuracy (F-Score) than single-home learning than all-home learning. This is because the *diversity* issue has a major impact in these cases. For example, in case 6, different users in different smart homes will open their window under different temperature/humidity; in case 5, different users in different smart homes have different interaction patterns with their smart fridge/microwave oven/cupboards before they turn on the oven. In summary, LoFTI achieves low FPs and FNs comparing with single-home learning and all-home learning.

C. Comparison with manually generated policies

In this part, we compare LoFTI with the current approach to manually generate contextual policies. We compare with an user survey [17] that asks 425 users to allow/block IoT accesses (e.g., turn on the camera) based on context norms (e.g., time of the day, whether user is at home or not).

Device	Policy
Camera	If user is at home, block the access to turn on the camera
Smart Door	When user is asleep, block the access to open the smart door
Smart Heater	When temperature $> 16^{\circ}C$, block access to turn on heater
Smart TV	No policy found
Smart Stove	No policy found
Smart Window	No policy found
Smart Light	From 6pm to 8am, smart light can be turned on
Smart Toilets	No policy found

TABLE VI: Contextual policies from user study [17]

To compare with this approach, we collect all the policies (user made allow/block decisions) in the user study [17] that are related with the IoT devices listed in our dataset, as shown in Table VI. We then convert the user generated policies into context-based rules and carefully maps the context from the user study to the corresponding context in our dataset. For example, we map the context that "user is asleep" in the user study to the context that "smart bed is occupied" in our dataset. We then apply the converted context-based rules on our testing dataset and measure the FNR and FPR. We provide no comparison for smart TV, smart window, smart stove and smart toilet, as we cannot find user generated policies for them in the user study.

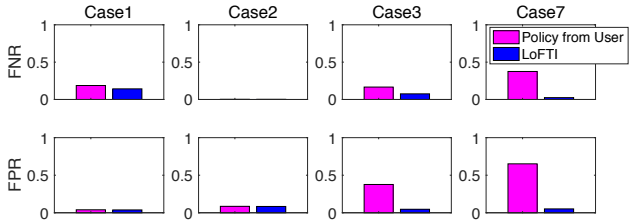

Fig. 14: Comparing user policy from survey and LoFTI.

For the rest 4 types of IoT devices - camera, smart door, smart heater and smart light, Figure 14 shows the FNR and FPR comparing user policy generated from survey and LoFTI. In case 1 and case 2, the FNR and FPR for user policy generated from survey and LoFTI are similar. This is because the scenario is relatively simple. In case 3, the FNR and FPR for user policy are higher for two reasons. First, user policy generated from survey would set a fixed value for temperature, which is not precise and not customized for different smart

homes. Second, user policy generated from survey fails to cover other environment variables that also impact the access of smart heater, e.g., humidity. In case 3, the FNR and FPR for user policy are much higher because it captures the wrong context. The user policy from survey depend on time as context to guide the access of smart light. However, in CASAS dataset, we observed that it is the movement of human sensed by motion sensors that guides the access of smart light. This is a typical case of misconfiguration by manually generated contextual policies.

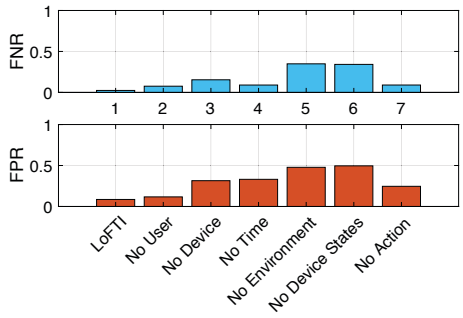

Fig. 15: Effectiveness of LoFTI's key features.

D. Effectiveness of LoFTI's key features

In this part, we evaluate how effective the key features generalized by LoFTI in terms of distinguishing benign and malicious contextual IoT access. To do so, we set the value fields of each key features to default value and measure how much the FNR and FPR will increase, as shown in Figure 15. The result shows that if any of the 6 types of key features defined by LoFTI is set to default, the FNR and FPR will increase. Therefore all 6 types of key features are effective in distinguishing benign and malicious contextual IoT access. We can also see that environment variables and device states cause the highest increase in FNR and FPR. This is not surprising as environment variables and device states are critical features to capture the context. Here the "user" feature is not causing a high increase in FNR and FPR. This is because of the CASAS dataset, where there are not many homes recorded with multiple users.

E. Effectiveness of LoFTI's simple SVM-based model with temporal structure

In this part, we will show that LoFTI's simple SVM-based model with temporal structure can achieve a good trade off between accuracy and communication/computation cost.

Accuracy: We compare LoFTI with the most simple machine learning model - SVM and most complex machine learning model - LSTM, from common machine learning models considered (including SVM, CNN, RNN, LSTM). As shown in Figure 16, we evaluate their detection capability by plotting the ROC (Receiver Operating Characteristic) curve. The x axis is the TPR (True Positive Rate) defined as $TPR = \frac{TP}{TP+FN}$, and the y axis is the FPR defined as $FPR = \frac{FP}{FP+TN}$. If the curve is more close to the $(0, 1)$ point, the corresponding approach has a stronger detection capability. To generate the

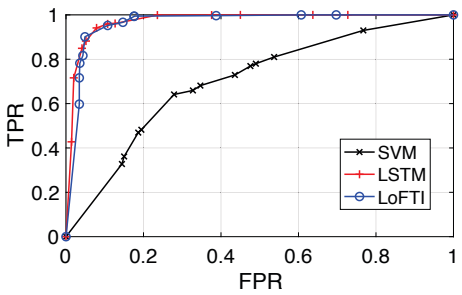

Fig. 16: ROC curve for SVM, LSTM and LoFTI.

curves, we adjust the regularization parameter c for SVM; we adjust the regularization parameter c and the window size of the temporal structure for LoFTI; we adjust the number of layers r, timesteps h and hidden units M for LSTM. From Figure 16, we can see that LoFTI can achieve a detection capability close to LSTM, stronger than SVM. This shows that LoFTI using simple model with temporal structure can achieve a good accuracy.

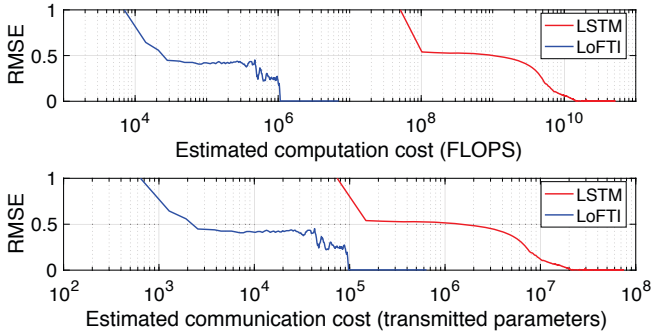

Fig. 17: Estimated communication/computation cost.

Computation cost: In this part, we compare LoFTI and LSTM in terms of computation cost and communication cost. We measure the RMSE (Root Mean Square Error) in each iteration for LoFTI and LSTM. Then we estimate the computation cost and communication cost for LoFTI and LSTM in each iteration. Therefore, we can estimate how much computation cost and communication cost are needed to reduce RMSE (Root Mean Square Error) close to 0 (training complete).

We estimate the computation cost (CPU FLOPS) for LSTM as $r*h*8*M*L*(2*N-1)$, where r is the number of layers, h is the number of timesteps, N is the number of features, M is the hidden unit size, L is the minibatch size. We estimate the computation cost (CPU FLOPS) for LoFTI as $2*N+N*h$, where N is the number of features and h is the time window span (we set it equal to the LSTM's timesteps).

We estimate the communication cost (transmitted parameters) for LSTM as $4*(N*M+M^2)$, where N is the number of features and M is the number of hidden unit. We estimate the communication cost (transmitted parameters) for LoFTI as $2*N$, where N is the number of features.

Figure 17 shows the result. We can observe that the computation cost of LSTM is 4 magnitudes higher than LoFTI; while the communitation cost of LSTM is 2 magnitudes higher than LoFTI. This result is as expected considering the complexity

of neural network layers and gate structures used by models such as LSTM.

In summary, LoFTI can achieve a good trade-off between accuracy and computation/communication cost.

F. Effectiveness of LoFTI's data augmentation

In this part, we evaluate the effectiveness of LoFTI's data augmentation to address the *unbalanced data* issue. As shown in Figure 18, we compare the FNR and FPR with different dataset construction mechanisms including unchanged, repetitive and LoFTI's data augmentation. From Figure 18 we can see that the unchanged dataset result in high FNR. This is because the large number of benign samples are pushing the decision boundary towards the attack samples (to avoid FPs), and result in high FNR. The repetitive mechanisms cannot fully address this issue because it still has a poor coverage over attack samples, so the FNR is still high. LoFTI's data augmentation mechanism successfully reduces the FNR while keep FPR almost at the same level.

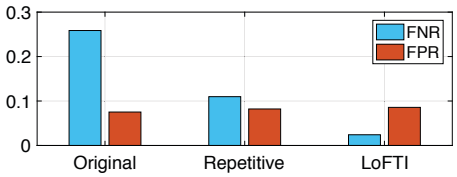

Fig. 18: Different dataset construction mechanisms.

G. LoFTI's capability of handling new attacks

Since LoFTI is a supervised learning approach. One interesting question to ask is whether LoFTI is able to detect new attacks that have never appeared in the training dataset before. To test this, we use 3 types of new attacks and test them against the model learned from dataset without samples of these attacks, as shown in Table VII.

New case	Device	Attack	Type
A	Camera	Eavesdropping	privacy leakage
B	Smart Door	Break-in when nobody at home	physical hazards
C	Smart TV	Copy watching records	privacy leakage

TABLE VII: New IoT attacks for testing.

Figure 19 shows the FNR and FPR for the 3 types of unknown IoT attacks and their corresponding known attacks. For case A and case C, LoFTI still can achieve low FNR and FNR. For case B, the FNR and FNR on unknown IoT attacks are significantly higher. This is because for case A and case C, the context during the new attack is different from the context of the benign access. For case B, the context during the new attack is close to the context of the benign access.

In summary, LoFTI can detect unknown IoT attacks if the context during the new attack is different from the context of the benign access. We also note that LoFTI cannot detect unknown IoT attacks against an IoT device that has no historical record of benign and malicious access in all smart homes. In such case, it is understandable as LoFTI is constrained by the dataset it can learn from.

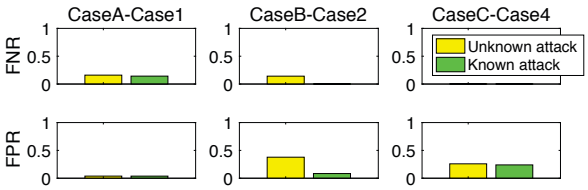

Fig. 19: FNR/FPR on unknown/known IoT attacks.

H. Masking sensitive parameters

The federated learning guarantees that the attacker is not able to see the raw data. However, a strong attacker who can observe the parameters of the model (e.g., via compromising the cloud backend or MITM attack) may be able to infer certain private information about the smart homes.

In this part, we perform a test to see if we can mask some sensitive environment variables in the learning process to prevent the observation from attackers, without significant decrease the detection accuracy. In this test, we mask a number of sensitive information from previous sensitive information tracking literature [11], [12], [23], including device location, device manufacturer, firmware version, device IP address, open ports, zip code and contact list. Then we measure the FNR, FPR and F-Score before and after the mask operation.

From Table VIII, we can see that the FNR and FPR before and after masking are the same. It shows that some of the sensitive information, including device location, device manufacturer, firmware version, device IP address, open ports, zip code and contact list, can be masked without significantly decreasing the detection accuracy. In the next step, we plan to explore other strategies such as *differential privacy* to see if we can mask a wider range of sensitivity information without significantly decreasing detection accuracy.

Approach	FNR	FPR	F-Score
Before mask	0.0241	0.0857	0.9378
After mask	0.0241	0.0857	0.9378

TABLE VIII: FNR/FPR before/after masking.

VI. Related Work

Federated Multi-Task Learning: MOCHA [22] is a novel learning mechanism that combines federated learning and multi-task learning. We build LoFTI based on MOCHA [22].
Manually generating contextual IoT policies: A rich body of previous works are focusing on manually generating contextual policies for IoT devices [15], [17], [19]. He et. al. [17] provide a user study to identify the critical contextual information that should guide IoT access in smart homes. FlowFence [15] and ContexIoT [19] build user-prompt frameworks to prompt detailed contextual information to let users make better IoT access control decisions. FlowFence focuses on data flow and ContexIoT focuses on control flow. The issue with manually generating contextual policies is that it is easy to misconfigure and hard to cover all the contextual policies needed.
Sensitive information tracking: Another group of related works [11], [12], [13], [23] propose to prevent privacy leakage by tracking the sensitive information in mobile applications for IoT devices. However, this approach requires the source

code of the mobile applications which is often not available. Also, since the tracking is based on mobile apps, it can only detect malicious/flawed mobile applications, and cannot handle malicious IoT access directly from the network (via weak authentication or backdoor).

Other IoT access control works: SmartAuth [24] provide an NLP-based approach to check the consistency between the IoT application description and IoT application source code. SmartAuth [24] cannot detect malicious access when there is no consistency issue. ESOs [21] is an integration platform that enables cross-platform enforcement for contextual IoT policies. This related work can be combined with LoFTI to address a broader scope of problems.

VII. Discussion

Beyond smart home: In this paper, we validate LoFTI using smart home datasets. However, as a distributed learning framework to learn contextual security policies, we envision that LoFTI can be applied in other areas beyond smart home settings, such as industry control systems.

Diverse deployment: In this paper, we assume that the deployment of IoT devices in the smart homes are similar or there is a mapping mechanism to mapping different IoT devices and their states (e.g., bedroom camera in home A can be mapped to bedroom camera in home B). In CASAS dataset, the deployment of IoT devices in the smart homes are similar as many of the smart homes are government supported homes for elder people. We will explore the mapping mechanism for diverse deployment in the future.

Model privacy: In this paper, we provide the guarantee that the attacker is not able to see the raw data in the smart home. A strong attacker who can see the parameters of the model (e.g., via compromising the cloud backend or MITM attack) may be able to infer certain private information about the smart homes. However, this put a high bar on the attackers. We envision LoFTI may combine that other techniques such as *differential privacy* to solve this issue.

VIII. Conclusion

The IoT devices can pose severe threats to smart homes, causing physical hazards and privacy leakage. Context-aware security policies are needed prevent physical hazard and privacy leakage. We show that it is possible to build a Federated Multi-Task learning framework to learn the context-aware security policies from multiple smart homes, while preserving the privacy of each smart home. In designing LoFTI, we address the hard trade-off between accuracy and privacy. Our results are promising and show LoFTI can detect realistic physical hazards and privacy leakage generated from reported IoT breaches. Looking forward, LoFTI serves as an interesting proof-of-concept for learning contextual policies for IoT via distributed learning.

Acknowledgment

This work was supported in part by NSF award number CNS-1564009. This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] Alexa has been eavesdropping on you this whole time. https://www.washingtonpost.com/technology/2019/05/06/alexa-has-been-eavesdropping-you-this-whole-time/?noredirect=on.

[2] Casas datasets. http://casas.wsu.edu/datasets/.

[3] Half baked iot stove could be used as a remote controlled arson device. https://hackaday.com/2017/04/20/half-baked-iot-stove-could-be-used-as-a-remote-controlled-arson-device/.

[4] Have a smart lock? yeah, it can probably be hacked. https://www.cnet.com/news/have-a-smart-lock-yeah-it-can-probably-be-hacked/.

[5] How cia mi5 hacked your smart tv to spy on you. https://www.zdnet.com/article/how-cia-mi5-hacked-your-smart-tv-to-spy-on-you/.

[6] Researchers hack philips hue lights via a drone; iot worm could cause city blackout. https://www.computerworld.com/article/3139860/researchers-hack-philips-hue-lights-via-a-drone-iot-worm-could-cause-city-blackout.html.

[7] Sustainability hacks: Automatic window control. https://hackaday.com/2011/09/29/sustainability-hacks-automatic-window-control/.

[8] Target attack shows danger of remotely accessible hvac systems. https://www.computerworld.com/article/2487452/target-attack-shows-danger-of-remotely-accessible-hvac-systems.html.

[9] The ultimate nightmare: Researchers learn how to hack connected smart-home toilets. https://bgr.com/2014/06/12/smart-home-toilets-hacked/.

[10] Watch a hacker access nest cameras and demand people subscribe to pewdiepie. https://www.vice.com/en_us/article/xwb8j7/watch-a-hacker-access-nest-cameras-and-demand-people-subscribe-to-pewdiepie.

[11] I. Bastys, M. Balliu, and A. Sabelfeld. If this then what?: Controlling flows in iot apps. In *CCS*, 2018.

[12] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac. Sensitive information tracking in commodity iot. *Usenix Security*, 2018.

[13] W. Ding and H. Hu. On the safety of iot device physical interaction control. In *CCS*, 2018.

[14] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and P. Bahl. An operating system for the home. In *NSDI*, 2012.

[15] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash. Flowfence: Practical data protection for emerging iot application frameworks. In *Usenix Security*, 2016.

[16] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] W. He, M. Golla, R. Padhi, J. Ofek, M. Dürmuth, E. Fernandes, and B. Ur. Rethinking access control and authentication for the home internet of things (iot). In *Usenix Security*, 2018.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. J. Unviersity. Contexlot: Towards providing contextual integrity to appified iot platforms. In *NDSS*, 2017.

[20] H. Nissenbaum. *Privacy in context: Technology, policy, and the integrity of social life*. Stanford University Press, 2009.

[21] R. Schuster, V. Shmatikov, and E. Tromer. Situational access control in the internet of things. In *CCS*, 2018.

[22] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[23] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia. Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1501–1510. International World Wide Web Conferences Steering Committee, 2017.

[24] Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, and P. Tague. Smartauth: User-centered authorization for the internet of things. In *Usenix Security*, 2017.

[25] A. Zhao, G. Balakrishnan, F. Durand, J. V. Guttag, and A. V. Dalca. Data augmentation using learned transformations for one-shot medical image segmentation. In *CVPR*, 2019.