IoT Sentinel: Correlation-based Attack Detection, Localization, and Authentication in IoT Networks

Dianshi Yang*†, Abhinav Kumar†, Stuart Ray†, Wei Wang†, Reza Tourani†
*Digital Security Department, EURECOM, Sophia Antipolis, France
dianshi.yang@eurecom.fr
†Department of Computer Science, Saint Louis University, St. Louis, USA
{dianshi.yang, abhinav.kumar, stuart.ray, wei.wang.5, reza.tourani}@slu.edu

2023 32nd International Conference on Computer Communications and Networks (ICCCN) | 979-8-3503-3618-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCCN58024.2023.10230209

***Abstract*—Security issues have become one of the major challenges for Internet-of-Things (IoT) networks. To overcome this challenge, the recent commonly-used approaches mainly focus on conducting encryption on IoT communication or performing continuous authentication for IoT devices by using pre-shared credentials (e.g., passcode and wireless channel signatures). However, these mechanisms are deemed insufficient, in part, due to the increasing number of data breaches and the recent proliferation of sensitive IoT devices and applications.**

We present *IoT Sentinel* – a novel security system that explores the correlation between IoT devices to effectively and efficiently secure IoT networks. Specifically, our system (i) detects potential attacks, (ii) localizes the attacker, and (iii) conducts dynamic implicit authentication at the same time. Moreover, instead of requiring full physical-layer access to IoT devices for *fine-grained measurement* of the wireless signal, IoT Sentinel uses only *coarse packet-level* device correlation information to secure IoT networks with negligible overhead to the network. Thus, making our approach compatible with existing constrained IoT devices. We extensively evaluate the efficacy of IoT Sentinel in different scenarios and settings. The experiment results show that our approach achieves around 96% attack detection accuracy, more than 70% attacker localization accuracy, and around 100% device authentication accuracy.

***Index Terms*—Internet of Things, Attack Detection and Localization, Authentication**

I. Introduction

The proliferation of Internet-of-Things (IoT) devices across various domains, such as smart buildings, precision agriculture, and medical IoT [1], have significantly improved the quality of life. However, IoT devices also significantly expand the attack surface, offering adversaries ample opportunities to orchestrate stealthy attacks. For instance, by impersonating a legitimate device, the attacker can hijack communication sessions or inject false data into the IoT networks, which introduces severe security concerns [2], [3]. This calls for the design of novel and lightweight security measures to thwart such threats effectively.

To secure IoT networks, the common trend is using pre-shared security credentials for access control and secure communication [4]–[9]. For instance, in [6], IoT devices in the network can generate multiple encryption keys according to the time information to secure the data-sharing process. On the other hand, recently, continuous authentication-based approaches have also been proposed to defend against various threats for IoT networks [10], [11]. For example, [10] allows

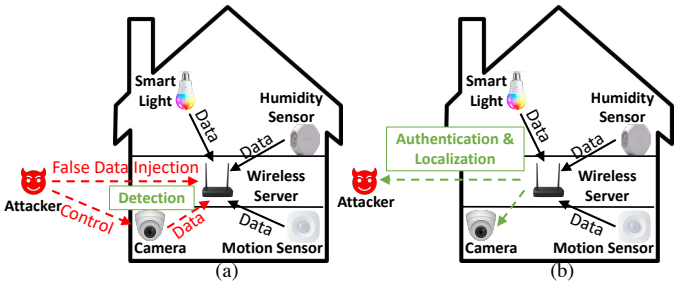


Fig. 1. An example of our proposed approach: (a) An active attacker can inject false data into the IoT network or take control of the legitimate IoT device. IoT Sentinel accurately detects these attacks. (b) After detecting attacks, IoT Sentinel can authenticate vulnerable devices and localize the attacker.

IoT devices to leverage the physical-layer information, such as Channel State Information (CSI), to identify the changes in the network. Interestingly, the context information gathered from IoT devices has been successfully used for device authentication [11]. These approaches work well in their application scenarios.

However, the increasing number of social engineering attacks and data breaches, along with the proliferation of IoT devices and applications, have significantly increased the possibility of leaking security information, such as secret keys or passwords. For instance, the authors in [12] have shown the success of social engineering attacks in obtaining secure credentials for attacking IoT networks. Another important factor in the insecurity of IoT networks is the human errors in choosing weak or redundant credentials, which is in part due to the number of devices and heterogeneity of IoT applications [13]. As a result, existing secret key-based approaches are deemed unreliable due to their vulnerabilities to a wide range of attacks. On the other hand, although continuous authentication schemes use devices' implicit and unique patterns to authenticate legitimate devices, these approaches either use fine-grained physical-layer information (e.g., CSI [10]) or hardly collected information [11] to conduct authentication. In practice, it is difficult for the user always to have access to the physical layers of all IoT devices. In addition, hardly collected information may introduce overhead to IoT networks. More importantly, current approaches primarily focus on attack detection and defense, which has left the adversary's localization unexplored. Therefore, even if some approaches successfully detect potential attacks, the attacker can still remain stealthy

979-8-3503-3618-4/23/$31.00 ©2023 IEEE

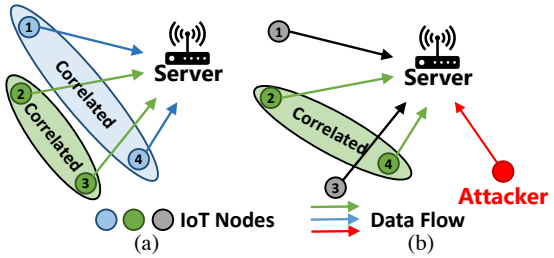


Fig. 2. A simplified example to show the effect of the attacker. (a) When the IoT network is not under attack, IoT nodes 1 and 4 are correlated, while nodes 2 and 3 are correlated. (b) The attacker breaks the original correlation relationships when the IoT network is under attack.

and keep trying to attack IoT networks.

In this paper, we introduce *IoT Sentinel* (Figure 1) – a novel IoT authentication system that **(i)** detects potential active attacks, **(ii)** localizes the malicious IoT device, and **(iii)** conducts implicit dynamic authentication in IoT networks simultaneously. The design of IoT Sentinel does not require full physical-layer access to IoT devices and further incurs negligible overhead. In designing IoT Sentinel, we need to overcome the following three main challenges. **First,** how to detect potential active attacks (e.g., impersonation and reply attack) without requiring full physical-layer access to IoT devices? **Second,** how to find the location of the adversary? **Third,** how to authenticate IoT devices without introducing much overhead to the network?

To overcome these challenges and achieve attack detection and localization, the **key idea** of IoT Sentinel is to explore the hidden features of device correlation in the IoT network. Specifically, we analyzed the delay and link quality correlations between IoT devices and found that these **packet-level correlations** are very sensitive to the changes in the wireless channel. When an active attacker tries to inject false data into the IoT network or conduct replay attacks, the wireless traffic from the attacker will introduce additional noise and fluctuation to the wireless channel, which affects the original correlation between IoT devices. However, since the packet-level correlation only provides the **coarse measurement** of IoT devices and the wireless channel, using these measurements for accurate attack detection and malicious device localization is still challenging. To transfer our key idea into a practical system, we introduce a new Correlation Composition Awareness (CCA) model and Pair Collaborative Localization (PCL) technique to conduct accurate attack detection and localization. Our CCA and PCL are designed based on a novel architecture using Gated Recurrent Units (GRU). To conduct dynamic implicit authentication without imposing significant overhead on the constrained IoT devices, we design a Behavior and Performance Measurement (BPM) scheme, which leverages the heterogeneity of IoT devices' hardware and software stack in identifying suspicious devices.

The *novel contributions* of this paper are as follows:

- To the best of our knowledge, this is the first work that investigates the utility of coarse IoT device correlation information, such as delay and link quality correlations to (i) detect active attacks, (ii) localize the malicious IoT devices and (iii) conduct dynamic implicit authentication simultaneously. The IoT Sentinel's design is generic and introduces minimal overhead to the network, which makes it applicable to existing IoT networks and heterogeneous applications.
- We design a novel Correlation Composition Awareness (CCA) model and a Pair Collaborative Localization (PCL) technique to conduct accurate attack detection and malicious IoT devices localization. We also introduce a Behavior and Performance Measurement (BPM) scheme for low overhead implicit authentication of IoT devices.
- We conduct extensive experiments in different real-world scenarios and settings. The experiment results show that IoT Sentinel can achieve more than **96%** attack detection accuracy, **70%** malicious device localization accuracy, and around **100%** device authentication accuracy.

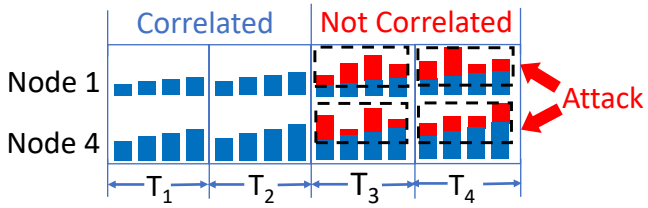


Fig. 3. An example of the insight behind the potential attack: The additional traffic introduced by the attacker affects the original delay correlation between node 1 and node 4.

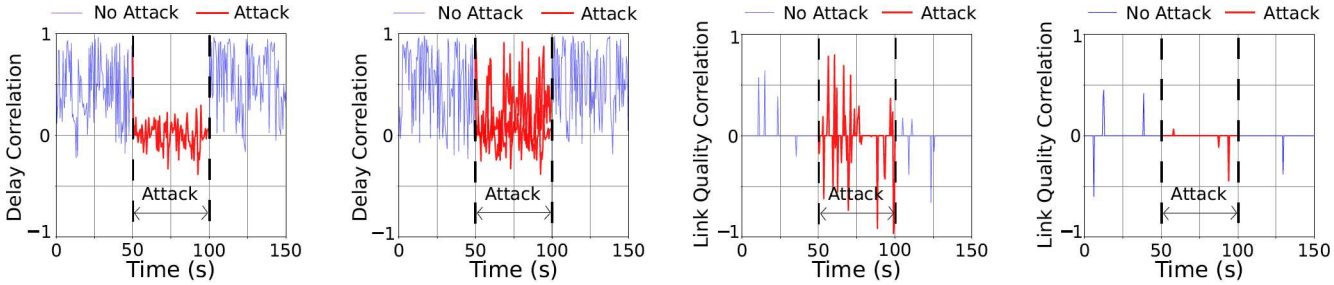
II. Observation

This section shows our observation of the correlation between IoT devices when the IoT network is under attack, which motivates us to design IoT Sentinel, which can detect attacks, localize malicious IoT devices, and conduct low-overhead dynamic implicit authentication at the same time.

A. Analysis of the IoT Correlation Information

In the context of wireless IoT networks, researchers have shown that link quality correlation between IoT devices can be used to significantly improve the network throughput and reduce network latency [14]–[16]. However, little work has investigated the possibility of using the correlation (i.e., link quality correlation and delay correlation) between IoT devices to detect potential attacks and localize malicious devices and intruders. To fill the knowledge gap, this section reports our initial analysis and empirical study on using correlation information for attack detection and localization.

In IoT networks, link quality correlation and delay correlation can be estimated by analyzing the received packet information at the receiver side. In real-world scenarios, due to the instability of the wireless channel, the packet-level link quality and delay may vary according to the changes in the wireless channel. Therefore, when an active attacker joins the network, the additional wireless traffic may introduce sudden changes in the original correlation relationship between IoT devices. A simplified example is shown in Figure 2 (a). IoT nodes 1, 2, 3, and 4 are communicating with the server. Nodes 1 and 4 are highly correlated, while nodes 2 and 3 are highly correlated. However, as shown in Figure 2 (b),



(a) Delay Correlation (b) Delay Correlation (c) Link Quality Correlation (d) Link Quality Correlation

Fig. 4. The delay and link quality correlations between two IoT devices. Our initial experiments represent attack scenarios where the change in delay and link quality correlations are imperceptible ((b) and (d)), which motivates the need for a robust detection and localization machine learning model.

when the IoT network is under attack (e.g., an attacker tries to inject false data to the network), the attacker ***unintentionally*** breaks the original correlation relationship. As shown in Figure 3, the attacker breaks the delay correlation between nodes 1 and 4 in time windows T_3 and T_4. This is because most IoT devices use the Carrier-Sense Multiple Access (CSMA) schemes to avoid potential collisions, while the channel access for CSMA is inherently fair [17], [18]. When an attacker tries to communicate with the server, the additional traffic from the attacker inevitably breaks the delay pattern of other IoT devices. Therefore, the detection of changes in the correlation patterns can lead to the detection of potential attacks and localization of malicious devices.

B. Experiment Setup and Our Observation

To empirically assess our reasoning, we deploy a server (JetsonTX2) and multiple IoT clients (Raspberry Pi3 and Raspberry Pi4) in a smart building on our campus. The IoT clients communicate with the server according to their own working schedule. The distances from the clients to the server vary from $5m$ to $30m$, which is the common setting in a smart building scenario. We also placed a laptop $20m$ away from the server as the attacker to inject false data into the IoT network. Since all the experiments follow similar trends, we only show the correlation between two clients (Figure 4).

From Figures 4 (a) and (c), it is evident that the link quality and delay correlations show different patterns when the IoT network is under attack. Specifically, when the IoT network is not under attack, IoT clients show positive delay correlations. This is primarily because IoT clients can overhear the wireless traffic transmitted from each other, and they are competing for the channel according to the CSMA scheme. However, when the attacker joins the network (from $50s$ to $100s$), these IoT clients have to frequently back off to avoid collisions with the wireless traffic from the attacker, which breaks the original delay correlation. Interestingly, the link quality correlation becomes unstable due to the additional noise introduced to the network between $50s$ to $100s$ time period. The above experiment results motivate us to use delay and link quality correlations for attack detection and localization.

However, as shown in Figures 4 (b) and (d), due to the randomness of the wireless channel, it is possible that delay and link quality correlations remain similar when the IoT network is under attack. In this case, it is challenging to detect the attacker by using traditional data processing approaches (e.g., using a complex correlation model or designing a data filter). This observation motivates us to design a robust correlation-based attack detection and localization approach using Gated Recurrent Units (GRU) machine learning architecture.

III. Models and Assumptions

A. System Model

Our design considers heterogeneous IoT networks, such as smart buildings or smart farms. The IoT network comprises a set of constrained IoT devices, such as sensors, actuators, and security cameras, deployed in the infrastructure. The IoT devices sense the environment, collect different types of data, and transmit it to the server. The server processes the data received from these devices for analytical and control purposes. The typical communication in IoT networks happens over the wireless medium. As such, we consider a wireless IoT network in which all the devices are equipped with wireless communication modules that allow them to send real-time data to the server. We further consider the IoT network to be heterogeneous in the sense that it consists of IoT devices with a wide range of resources, capabilities, and tasks. In particular, these IoT devices feature different hardware architectures, hardware specifications (e.g., processing unit and memory), and software stacks, including the operating system.

As mentioned earlier, IoT devices transmit their sensory information to the server. We assume that the communication between the legitimate IoT devices and the server is protected through a secure communication protocol, such as TLS or using a pre-shared key. We note that secure communication is a common assumption. More importantly, the design of IoT Sentinel is independent of the presence (or lack of) of any secure communication. We also consider that IoT devices perform measurements and packet transmission according to their applications. For instance, a phasor measurement unit measures the frequency in the power grid around 60 measurements per second [19] or a Nest thermostat measures the temperature once every minute.

B. Threat Model and Assumptions

In this work, we consider an active adversary whose primary objective is to compromise the integrity of the system by injecting false information, aiming to deceive the server while

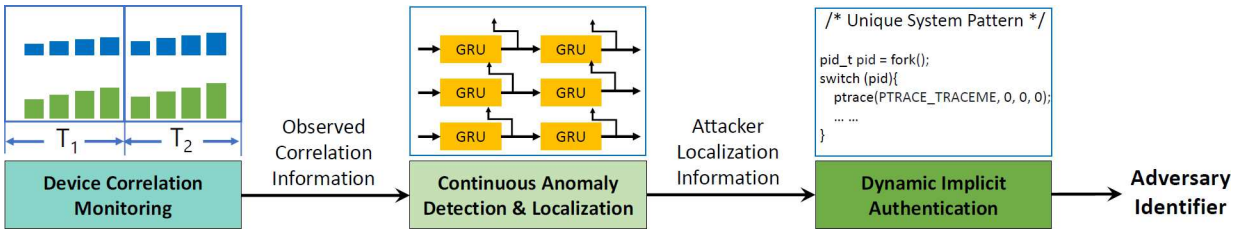


Fig. 5. IoT Sentinel features three core modules: (i) the correlation monitoring module continuously scans the wireless channel and infers correlation information; (ii) the anomaly detection & localization module identifies and localizes an attack; and (iii) the dynamic authentication module implicitly authenticates only suspicious IoT devices to detect the adversary.

concealing its presence. To orchestrate such an active attack, the adversary has to infiltrate the wireless network to be able to transmit false data to the server on the shared wireless channel. To infiltrate the network and inject false information, we assume that the adversary has obtained the requisite credentials (e.g., the network security key) through common attack vectors, such as spear-phishing attacks or using the leaked password files. Thus, the adversary can directly inject false information into the system. Note that this is a reasonable assumption since the objective of this work is to detect attacks aimed at compromising the integrity of the system, rather than those that target the confidentiality of communication.

We emphasize that these attack vectors are among the most common exfiltration approaches, primarily due to the increasing success of social engineering attacks, data breaches, and inherent vulnerabilities in IoT devices. However, we do not consider the scenario in which the adversary compromises a legitimate device and uses it to inject false data.

We neither impose any restriction on the mobility of the adversary nor on the adversary's location, with the exception that the adversary should remain within the transmission range of the server for successful wireless communication. Finally, we do not impose any restrictions on the computing capability of the adversary. We also consider a secure onboarding process, where the server performs individual authentication of the IoT devices for the first time and only allows legitimate IoT devices to join the network. Onboarding an IoT device often requires an interaction between the user and the device, e.g., scanning a QR code or manually pressing a button for device pairing. As such, the adversary can only orchestrate the infiltration and false data injection after the network initiation.

IV. Design Overview

The design goals of IoT Sentinel are to *(i)* detect active attacks, *(ii)* localize malicious IoT devices, and *(iii)* identify the suspicious device(s). To achieve these goals, we designed three modules as the core of IoT Sentinel (Figure 5):

• **Device Correlation Monitoring (§ V-A):** In IoT Sentinel, the server calculates the delay and link quality correlations between each pair of IoT devices according to the received packets. Since IoT devices have different work schedules, the server may receive different numbers of packets from each IoT device in a given time frame. The mismatches between the number of packets affect the correlation calculation, which consequently impacts the detection efficacy. Therefore, instead of calculating the correlation based on a fixed time window, IoT Sentinel dynamically changes the time window according to the traffic rates of IoT devices.

• **Continuous Anomaly Detection and Localization (§ V-B):** According to the correlation information, IoT Sentinel will dynamically detect the potential attack and localize the malicious IoT devices. To cope with the randomness in the wireless channel, we design two novel techniques – *Correlation Composition Awareness (CCA)* and *Pair Collaborative Localization (PCL)* – to support more accurate attack detection and localization under different scenarios.

• **Dynamic Implicit Authentication (§ V-C):** In IoT networks, continuously authenticating all devices incurs significant communication and computation overhead and reduces the life-cycle of battery-operated IoT devices. Thus, we design a device *Behavior & Performance Measurement (BPM)* module for IoT Sentinel to analyze the unique system patterns (e.g., CPU utilization, memory usage, and response time) in the IoT devices, aiming to accurately detect the malicious IoT device, whose behavior is different from the behavior of known devices. According to the outcome of the PCL to the BPM module, IoT Sentinel *dynamically* selects a smaller subset of suspicious IoT devices, and *implicitly* authenticates them by analyzing their performance behavior. Thus, reducing the system's overhead and increasing the lifetime of IoT devices.

V. Detailed Design of IoT Sentinel

A. Device Correlation Monitoring

We first show how to monitor correlation among IoT devices in the network. Specifically, in IoT Sentinel, the server calculates the delay and link quality correlation using Pearson Correlation [20]. Since the traffic rates of different IoT devices may vary according to their applications, the server uses a time window with dynamic sizes to calculate the correlation. Formally, we define the minimal and max window sizes as T_{min} and T_{max}, respectively. If the traffic rate τ_i from a typical IoT device i is lower than the predefined threshold I (I can be determined based on the corresponding smart IoT application), the size of the time window T can be calculated as: $T = a_1(\frac{I}{\tau_i})T_{min} + a_0 T_{max}$, where a_0 and a_1 are decision factors ($a_1, a_0 \in \{0, 1\}$). If the traffic rate from one typical IoT device is too low, increasing the time window size may result in an extremely high correlation monitoring delay. In this case, a_1 and a_0 will be set to 0 and 1, respectively, and IoT Sentinel will use the default max time window T_{max}.

The calculation of delay correlation is a challenging task primarily due to the lack of strict synchronization between the IoT devices in the network (each IoT device has its own working schedule and timer). As a result, it is relatively difficult for the server to measure the time delay of each packet accurately. To overcome this challenge, IoT Sentinel mainly measures the relative packet transmission delay. Specifically, before transmitting the packet to the server, each packet is assigned a timestamp by the sender. Upon receiving the packet, the server will assign a timestamp to each packet according to the server's timer. Then, the *relative* packet transmission delay D is calculated by the differences in timestamps between the IoT sender and the server. Since IoT Sentinel only needs to calculate the delay correlation, the timers' differences between the IoT sender and server will not affect the correlation results.

Now, to calculate the link quality and delay correlations, we first define the correlation factor as C. Considering Φ represents the number of IoT devices, then the correlation of link quality Q or packet delay D between IoT devices i and j – denoted by $r^E_{i,j}$ – can be calculated as:

$$r^E_{i,j} = \frac{\sum_{i,j=0}^{T}(E_i - \overline{E_i})(E_j - \overline{E_j})}{\sqrt{\sum_{i,j=0}^{T}(E_i - \overline{E_i})^2 \sum (E_j - \overline{E_j})^2}},$$

where $E \in \{Q, D\}$ and T is the size of the time window.

B. Continuous Anomaly Detection and Localization

Our preliminary analysis (Figure 4) shows an evident change in the correlation measurements when the attacker transmits data. However, the impact of the attacker on the correlation drastically changes based on the attacker's behavior, such as traffic rate. Thus, resulting in a non-linear attack detection and localization decision boundary. As such, manually setting a linear threshold for attack detection and localization will be ineffective, particularly when dealing with various scenarios and configurations. For example, when a new legitimate device joins the network, it will also change the correlation between devices. To address this challenge, we propose to use a machine learning-based attack detection and localization framework as supervised learning models effectively learn non-linear decision boundaries and can be consistently *fine-tuned* to adapt to new environments, layouts, configurations (e.g., adding new legitimate devices), and attack signatures.

In IoT Sentinel, we design a Recurrent Neural Network for learning the temporal dependencies of link quality and delay correlations prior to an attack as well as the attacker's impact on these temporal dependencies. In particular, we use Gated Recurrent Units (GRU) since it has fewer parameters than Long-Short Term Memory (LSTM) with a similar performance to an LSTM-based model [21]. To eliminate the need for designing separate attack detection and attacker localization models, we include an early exit to the final machine learning pipeline, resulting in a single memory-efficient neural network.

To train the attack detection model, we generated the training dataset, in which the delay and link quality correlations between pairs of devices are set as input features. We formalize our attack detection dataset ($\mathcal{D}_D$) as:

$$\mathcal{D}_D = \Big\{(\mathcal{X}, \mathcal{Y}_D) : \mathcal{X} = \langle r^D_{i,j}, r^Q_{i,j}\rangle\Big\},$$

where $C^D_{i,j}$ and $C^Q_{i,j}$ are the delay correlation and link quality correlation between device i and device j, respectively. The label ($\mathcal{Y}_D$) will be equal to 0 if there is no attack and 1 otherwise. We build our model with an early exit feature, using two smaller auxiliary classifiers – $C_1\big(F_D(r^D_{i,j}, r^Q_{i,j})\big)$ for attack detection and $C_2\Big(F_L\big(F_D(r^D_{i,j}, r^Q_{i,j})\big)\Big)$ for attack localization. For training the attack localization auxiliary model, we generated a dataset that uses the detection model's outcome. We formally define the Localization dataset ($\mathcal{D}_L$) as:

$$\mathcal{D}_L = \Big\{(\mathcal{X}, \mathcal{Y}_L) : \mathcal{X} = \langle F_D(r^D_{i,j}, r^Q_{i,j})\rangle\Big\},$$

where $\mathcal{X}$ is the output of the detection model, and $\mathcal{Y}_L \in \{Z1, Z2, Z3, Z4\}$ are the localization model's labels, which represent the adversary's zone (the region of the network where the attacker is located).

The detection and localization model consists of GRU layers followed by batch normalization and dropout layers, which allows the model to achieve higher accuracy and avoids overfitting. We trained the model by modifying the multi-exit training algorithm proposed in [22] as shown in Algorithm 1. In training the attack detection and localization model, we take a greedy approach, in which we first divide the early exit model into two auxiliary classifiers and then optimize each one individually. The first classifier, i.e., $C_1(F_D)$, is used for detection and is trained using stochastic gradient descent to update the untrained Classifier (Line 1-2). After this training

Algorithm 1: Detection and Localization Training

Input: $\overline{F} = \overline{F}_L\big(\overline{F}_D(x)\big)$ (untrained), $\mathcal{D}_D$

Output: $F = F_L\big(F_D(x)\big)$

Detection Auxiliary Model Training:

1 **for** *number of the training epochs* **do**

2 Use stochastic gradient descent to update $\overline{F}_D$ on: $\mathcal{L}_{\overline{F}_D}\Big(C_1\big(\overline{F}_D(x_i, y_i)\big)\Big), \forall (x_i, y_i) \in \mathcal{D}_D$

3 **end**

4 **Freeze** F_D ▷ *$C_1(F_D)$ is the fully-trained Auxiliary detection model.*

6 $\mathcal{D}_L \leftarrow F_D(x), \forall x \in \mathcal{D}_D$ ▷ *Generated using $F_D(x)$.*

Localization Auxiliary Model Training:

7 **for** *number of the training epochs* **do**

8 Use stochastic gradient descent to update $\overline{F}_D$ on: $\mathcal{L}_{\overline{F}_L}\Big(C_2\big(\overline{F}_L(x_i, y_i)\big)\Big), \forall (x_i, y_i) \in \mathcal{D}_L$

9 **end**

10 QUANTIZE(F) ▷ *Dynamic-range quantization.*

11 **Return** F ▷ *Fully-trained detection and localization model.*

step, we freeze the layers of the trained detection classifier (Line 4) and use the output of the detection classifier as the training set for the localization auxiliary classifier (Line 7-8). After completing model training, we compress the final model using the 8-bit dynamic range quantization method to reduce the model's memory footprint.

C. Dynamic Implicit Authentication

After attack detection and adversary localization, aiming to identify the intruder's device, IoT Sentinel dynamically selects a subset of devices and implicitly authenticates them using the behavior and performance measurement module. The dynamic device selection process uses the outcome of the correlation abnormality detection module and chooses those devices that are located in the attack zone. The implicit authentication process analyzes the performance and behavior of devices for a particular process to identify a device with a behavior distinct from other known devices. The operation of the authentication process spans two phases – registration and inference. During the registration phase, i.e., IoT device onboarding, the server loads a solving procedure (PROC()) to IoT devices and performs multiple rounds of challenge-response with each legitimate device to create a corpus of ground truth performance and behavior baselines, including memory and CPU utilization along with function calls of the software stack. During the inference phase (Algorithm 2), the server sends a challenge to the selected IoT devices. These devices, using the pre-loaded PROC(), solve the challenge and reply to their responses along with their performance and behavioral measurements. While various challenges can be plugged into IoT Sentinel, in our evaluation, we used factoring large numbers as the challenge.

Analyzing these features allows the server to identify an anomalous device whose performance does not match the legitimate known devices. While various challenges can be plugged into IoT Sentinel, in our evaluation, we used factoring large numbers as the challenge. We note that only legitimate devices are equipped with PROC(). However, even if the attacker obtains the challenge-solving procedure and solves the challenge, its performance and behavior measurements would be different from the behavior of a legitimate device, which is primarily due to differences in the processes running on legitimate devices and the adversary.

Per Algorithm 2, the server initiates the authentication process by generating and sending a fixed-length random number (R) as the challenge to the target device ϕ (Lines 1-3). The target device, on receiving R, first initializes requisite structures (Line 4) and then runs PROC(R) to solve the challenge (Lines 5-6). During the challenge-solving procedure, device ϕ measures the performance of the PROC(R) procedure, including its CPU utilization (U), the allocated memory (M), and the list of all system and function calls used by PROC(R) along with their frequencies, i.e., L_f (Lines 7-10). In addition, device ϕ measures the execution time of the PROC(R) procedure (T). These performance metrics are influenced by different factors, including the device's hardware specifications, the operating system, and the processes running on the device. As a result, devices of the same type with a similar configuration show different performance measurements due to variations in their daemon processes or other running applications. Finally, device ϕ creates the response as a 5-tuple (Line 14), including the solution's digest and the measured performance and behavioral information, and sends it back to the server for authentication.

Algorithm 2: Behavior & Performance Detection

Input: Set of target devices (Φ)

Output: Adversary ϕ' $\left(M_{\text{VERIFY}}(\mathcal{R}_{\phi'}) = 0\right)$

1 $R =$ GENERATE-CHALLENGE (1^n)
2 **for** *($\phi \in \Phi$)* **do**
3 SEND (R) to ϕ
4 $U \leftarrow 0, M \leftarrow 0, L_f = \{\}$ ▷ *ϕ operations start.*
5 $T_{start} \leftarrow$ UNIX EPOCH (CURRENT)
6 $PID \leftarrow$ PROC (R)
7 **while** $\left(S \leftarrow \text{PROC (R)}\right)$ **do**
8 $U \leftarrow \frac{U + \frac{U_i^{PID}}{U_i^{total}}}{i+1}$
9 $M \leftarrow \frac{M + M_i^{PID}}{i+1}$
10 $L_f \leftarrow ptrace$
11 **end**
12 $T_{end} \leftarrow$ UNIX EPOCH (CURRENT)
13 $T = T_{end} - T_{start}$
14 $\mathcal{R}_\phi = \langle U, M, T, L_f, SHA256(S) \rangle$
15 SEND$(\mathcal{R}_\phi)$ to server ▷ *ϕ operations end.*
16 $\{0, 1\} \leftarrow M_{\text{VERIFY}}(\mathcal{R}_\phi)$
17 **end**

To collect the system and function calls of the PROC() process, we used function call hooking [23] and function interposition [24] methods. For function call hooking, we used *ptrace* (Line 10 of Algorithm 2), which is a system call in Unix-like operating systems for monitoring, execution control, and memory examination of a given process [25]. The function interposition is a method of replacing the primary implementation of calling a function in dynamic libraries with calls to user-defined wrappers. Thus, allowing the implementation of custom functionalities for tracing the execution of a program in a more granular way. In IoT Sentinel, we intercepted *malloc* and *free* functions by implementing a wrapper around them.Upon the completion of PROC() process, *ptrace* returns a list of system call numbers and their frequencies. We note that the system call names may follow different naming conventions depending on the environment of the operating system and the hardware architecture of the host devices.

In building the dynamic authentication module, we designed a lightweight fully-connected neural network consisting of four layers for learning the behavioral and performance measurement of the IoT devices and predicting the legitimacy of

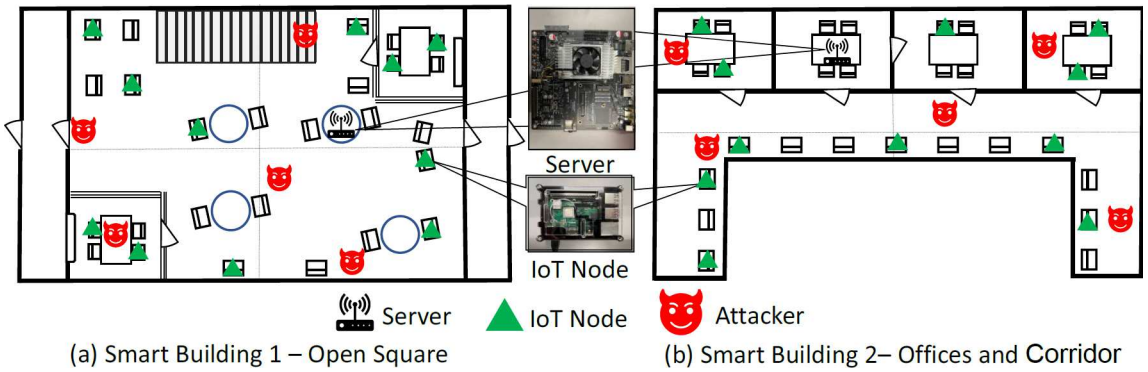


Fig. 6. We conduct experiments in two smart buildings. Each features multiple zones (separated with gray dashed lines).

the IoT device. In particular, we trained a binary classification neural network using the ground-truth data that the server collected during the device onboarding phase and deployed the trained model on the server for device authentication.

VI. Implementation and Evaluation

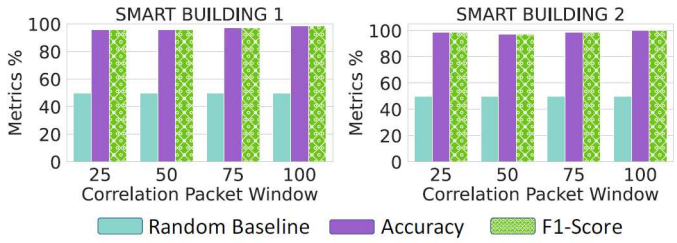
A. Experiment Setup

As shown in Figure 6, we extensively evaluate IoT Sentinel under various settings in two smart building scenarios. Smart building 1 has a relatively open area with fewer walls and obstacles with two main entrances on the left and right sides and frequent human movement. We placed the wireless server (JetsonTX2 and a WiFi AP – Tenda N301) in the open square and used two JetsonTX2 devices, where Jet18 runs Ubuntu18 and Jet20 runs Ubuntu20. We randomly distributed eleven Legitimate IoT devices (Raspberry Pi3 and Pi4) in both the open square and glass meeting rooms. Specifically, Raspberry Pi3 and Pi4 use 32-bit Debian version 8 Jessie and Debian version 11 Bullseye operating systems, respectively. Each IoT device has the same challenge stored in a binary executable format and a daemon to receive the challenge from the server. Each IoT client uses MIRACL library [26] for big-number computing. To communicate with the server, these IoT clients use WiFi channel 1 (2412MHz) to transmit data packets and CSMA schemes to avoid potential collisions. To attack the IoT network, we use MSI GP65 Leopard 9SE to act as an attacker; we chose only one attacker as it represents a stealthier attack scenario. The attacker is placed in different zones and can send packets to impersonate any legitimate devices in the network.

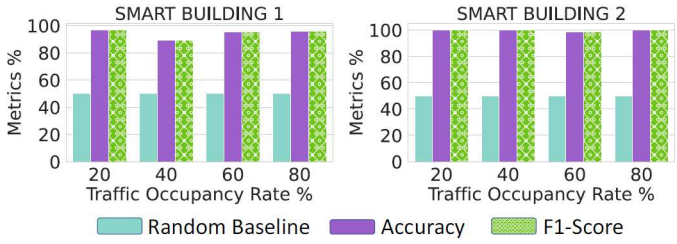
To show the effectiveness of our design, we use the same server, IoT devices, and attacker to conduct experiments in smart building 2, which is shown in Figure 6 (b). Smart building 2 is a pure indoor environment with thick walls and multiple obstacles. This building consists of several office rooms on the side of a long corridor. In this scenario, the wireless server is in the meeting room, while legitimate IoT devices are randomly distributed both in rooms and along the corridor. During the experiment, all the doors of the rooms were closed to simulate a real-world scenario. The attacker was randomly placed in any possible position.

B. Attack Detection Accuracy

To evaluate the effectiveness of IoT Sentinel, in Figure 7(a), we first show the attack detection accuracies under different window sizes in both smart building 1 and smart building 2. In this experiment, we define the time window size varies according to the number of packets received by the server. As we can see from these figures, the attack detection accuracies and F1-score are more than 96% regardless of window sizes, which is much higher than that of random guessing (50%). This is because the active attacker breaks the original correlation relationships between legitimate IoT devices, which is extremely easy to be detected by IoT Sentinel.



(a) Attack detection accuracy under different window sizes.



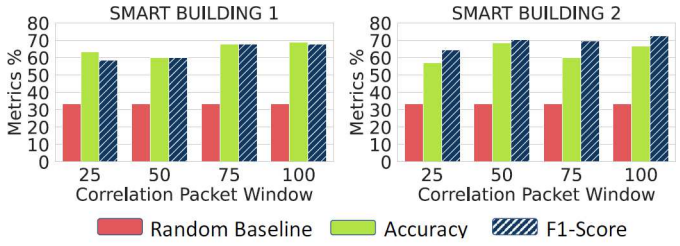
(b) Attack detection accuracy under different network traffic rates.

Fig. 7. Detection accuracy under different window sizes and traffic rates.

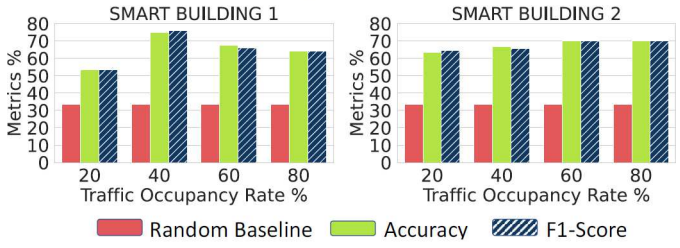
Figure 7(b) shows the attack detection accuracy under different network traffic occupancy rates. As we can see from these two figures, the attack detection accuracy of IoT Sentinel remains relatively stable in most cases. In addition, we also can find that the attack detection accuracy is slightly increased as the traffic occupancy rate increases. This is because the server receives more packets, which makes it easier to update the device correlation in real time. As a result, the attack detection accuracy is slightly increased in this case. **In summary,** IoT Sentinel can accurately detect potential attacks regardless of the time window and traffic rates.

C. Attacker Localization Accuracy

Figure 8 shows the attacker localization accuracy under different window sizes and traffic occupancy rates. Specifi-



(a) Localization accuracy under different window sizes.



(b) Localization accuracy under different network traffic rates.

Fig. 8. Localization accuracy under different window sizes and traffic rates.

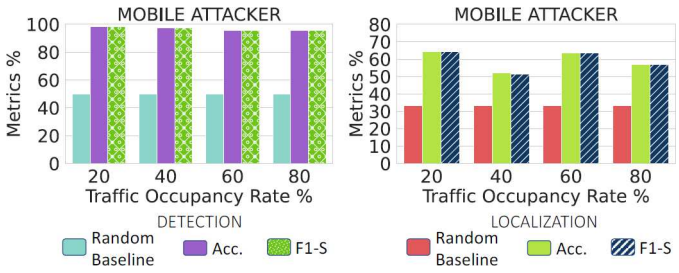


Fig. 9. Detection and localization accuracy for a mobile attacker.

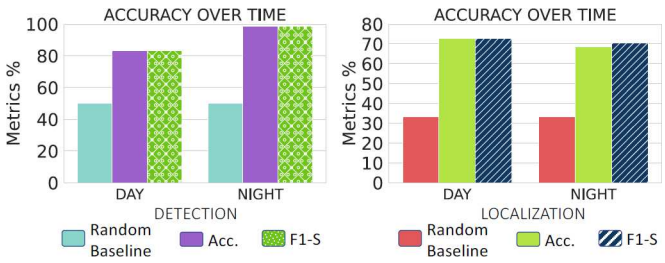


Fig. 10. Detection and localization accuracy over different time periods.

cally, we first show the attacker localization accuracy under different window sizes in Figure 8(a). As we can see from the experiment results, as the window size increases, the localization accuracies increase from 62% to 69% and 58% to 68% in smart building 1 and smart building 2, respectively, which is much higher than that of random guessing (30%). This is because the number of packets received by the server is increased, which makes it easier for the server to measure the wireless channel more accurately.

Figure 8 (b) shows the localization accuracy under different traffic rates. In smart building 1, IoT sentinel is able to achieve more than 72% localization accuracy at a network traffic rate of 40%. While in smart building 2, the localization accuracy increases as the network traffic rate increases. In this scenario, the max localization accuracy is 70% at a network traffic rate of 80%. This experiment proves our analysis – *The higher number of packets received by the server will provide a more accurate estimation of the wireless channel.* **In summary,** IoT Sentinel can effectively localize the potential active attacker. The higher the traffic rate, the better the localization accuracy.

D. Mobility Analysis

We also conduct experiments to study the performance of IoT Sentinel in the mobile scenario. During the experiment, the attacker is moving at a speed of $1m/s$ in random directions in the test environment across different zones. Since the experiments in smart building 1 and smart building 2 show similar trends, in Figure 9, we only show the results in smart building 1. As we can see from these results, the attack detection accuracy and F1-score remain above 90%, while the accuracy and F1-score of attacker localization are about 60%. The mobility of the attacker introduces additional noises to the wireless channel, such as fluctuations in signal strength and packet loss, which affects the accuracy of attack localization. Although localization accuracy is slightly lower than that of a static attack, it is still significantly higher than random guessing, which shows the efficiency of IoT Sentinel. **In summary,** IoT sentinel can effectively detect potential attacks and localize mobile malicious IoT devices in the mobile scenario.

E. External Environment Interference Analysis

To evaluate the performance of our system under varying levels of environmental interference, we conducted tests during both daytime and nighttime with a window size of 50 packets and a traffic occupancy rate of 100%. During the daytime, there are high levels of interference from pedestrians and external electronic devices, while during the nighttime, there is a quieter environment with fewer pedestrian activities.

Per Figure 10, the level of interference in the environment affects the accuracy of attack detection. During the daytime, when there is a high level of interference from pedestrians and external electronic devices, the attack detection accuracy and F1-score are dropped to about 80%, which is still high enough to detect potential attacks. During the nighttime, the attack detection accuracy is over 95%. On the other hand, the localization accuracy remains around 70% for both daytime and nighttime. **In summary,** the performance of IoT Sentinel is high enough to detect attacks and localize malicious devices regardless of environmental interference.

F. Evaluation of Dynamic Implicit Authentication

To evaluate the efficacy of the BPM module, we analyzed the challenge-solving process on a Raspberry Pi3, a Raspberry Pi4, and two JetsonTX2 devices. We installed different operating systems on these devices to assess the behavioral differences between different software stacks. Figure 11(a) shows that the learning-based BPM module achieves 100% accuracy in differentiating legitimated devices from outsiders. While the BPM module uses a binary classifier for scalability purposes, our investigation revealed that our model could identify whenever a malicious device is introduced to the system with a very high probability (Figure 11(b)). Such

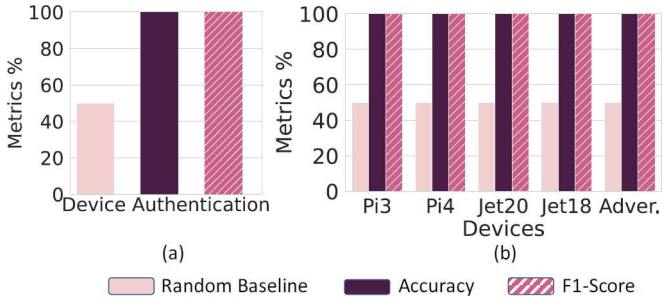


Fig. 11. Accuracy and F1-Score of the device authentication model.

outstanding efficacy is primarily due to evident trends in the performance and behavior of various IoT devices, which we will show in what follows.

From Figure 12, one can observe distinctive performances across all four devices. Even for scenarios where some of the metrics are fairly similar, e.g., execution time and CPU utilization of JetsonTX2 devices, running different operating systems resulted in distinct memory usage (Figure 12(a)). We also compared the efficacy of the BPM module in differentiating two identical Raspberry Pi4 under light and heavy processing workloads (Figure 13). Compared to the light workload scenario with a more homogeneous performance benchmark, the heavy workload scenario has shown more fluctuations in the measured performance, particularly memory usage and CPU utilization. This is primarily because of resource contention among processes and cache effects with multiple threads, which caused performance reduction. We also assessed the impact of different operating systems on the challenge-solving process (the list of function calls in Table I). Despite some similarities across different operating systems, e.g., *write* function, we have observed distinct patterns that can be used in classifying different devices. For instance, some specialized function for ARM64 system like *fstat64* is only used by JetsonTX2 devices, while the ARM32 specialized function *fstat* is used for Raspberry Pi devices. Also, even for the same device type, different versions of an operating system result in different function calls usage, e.g., *nanosleep* or *faccessat*. **In summary,** IoT Sentinel can effectively and accurately authenticate IoT devices using the evident trends in hardware specifications, processing workloads, and software stacks of various devices.

VII. Related Work

We categorize the related work into the following two parts:

• **Wireless Correlation and Localization.** Link quality correlation in wireless networks has been extensively studied [14], [27], [28]. Most of the literature discusses the approach of using link quality correlation to increase network efficiency. For example, in [14], the author mainly leverages the link quality correlation to reduce communication overhead. To be best of our knowledge, little work has been done to use coarse packet-level link quality and delay correlations for attack detection and malicious localization. On the other hand, researchers have also proposed lots of approaches in the field of localization using wireless signals [29]–[31]. For instance, [30] and [31] mainly use RSSI and CSI information

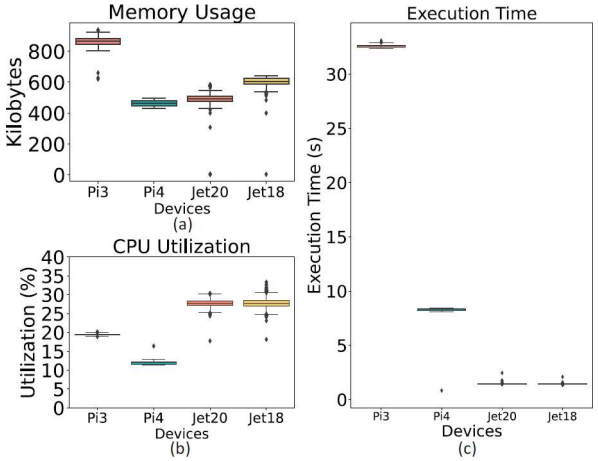


Fig. 12. Performance benchmarking of BPM module across various hardware/software configurations.

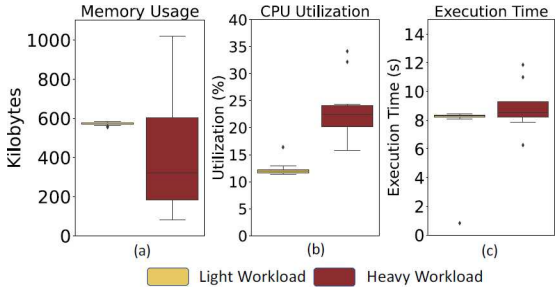


Fig. 13. Comparison of two Raspberry Pi4 with identical hardware and software specifications under different processing loads.

for localization, respectively. Interestingly, EAR [29] can use ambient wireless signals to conduct accurate localization and gesture recognition. Unlike their approaches, IoT Sentinel mainly leverages link quality and delay correlation to detect potential attacks and localize malicious IoT devices.

• **Authentication of IoT Devices.** Researchers have pro-

TABLE I
THE LIST OF FUNCTION CALLS FOR THE PROC() PROCESS ALONG WITH THEIR FREQUENCIES. A DASHED LINE REPRESENTS NO FUNCTION CALL.

	JetsonTX2-18	JetsonTX2-20	Pi4	Pi3
access	-	-	1	7
cacheflush	-	-	-	1
clock_nanosleep	-	1	-	-
close	3	3	6	8
faccessat	4	1	-	-
fstat	4	4	-	-
fstat64	-	-	7	9
lseek	-	-	-	12
mmap	8	8	-	-
mmap2	-	-	13	22
mprotect	6	6	10	14
munmap	1	1	2	2
nanosleep	1	-	-	1
open	-	-	-	8
openat	3	3	6	-
read	2	2	4	18
readlink	-	-	1	-
rt_sigaction	-	-	-	1
rt_sigprocmask	-	-	-	2
set_tls	-	-	1	1
uname	-	-	-	1
write	1	1	1	1

posed lots of work on Wireless Authentication techniques [4]–[8], [32], [33]. For example, AuthIoT [32] harnesses a learning-based authentication scheme for wireless IoT devices without input interfaces. Move2Auth [33] requires users to perform hand gestures in front of a given IoT device for authentication. On the other hand, continuous authentication techniques [10], [11] are considered more secure since an adversary cannot obtain permanent access to a network. For example, ContexIoT [11] mainly uses fine-grained context identification for continuous authentication. Different from their approaches, IoT Sentinel implicitly authenticates a small number of devices according to attack detection and malicious localization results, which significantly reduces the overhead.

VIII. Conclusion

This paper introduces *IoT Sentinel*, which leverages coarse packet-level correlation information (i.e., link quality and delay correlation) to **(i)** detect active attacks, **(ii)** localize the malicious IoT device and **(iii)** conduct dynamic implicit authentication simultaneously. To do this, we design novel Correlation Composition Awareness (CCA) and Pair Collaborative Localization (PCL) techniques that can conduct attack detection and localization in noisy wireless environments. To reduce network overhead and further improve the security level of IoT networks, we also introduce the Behavior and Performance Measurement (BPM) technique. The real-world evaluation results show that IoT Sentinel can achieve more than 96% attack detection accuracy, 70% malicious device localization accuracy, and 100% device authentication accuracy.

Acknowledgement

This work is partially supported by US NSF awards #2133407, #2148358, and Taylor Geospatial Institute (TGI).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and privacy issues for an iot based smart home," in *International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2017.

[3] Z. Mohammad, T. A. Qattam, and K. Saleh, "Security weaknesses and attacks on the internet of things applications," in *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology*, 2019.

[4] A. Diro, H. Reda, N. Chilamkurti, A. Mahmood, N. Zaman, and Y. Nam, "Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication," *IEEE Access*, 2020.

[5] S. Kumar, Y. Hu, M. P. Andersen, R. A. Popa, and D. E. Culler, "Jedi: Many-to-many end-to-end encryption and key delegation for iot." in *Proceedings of USENIX Security Symposium*, 2019.

[6] H. Shafagh, L. Burkhalter, S. Ratnasamy, and A. Hithnawi, "Droplet: Decentralized authorization and access control for encrypted data streams," in *Proceedings of USENIX Security Symposium*, 2020.

[7] X. Wang, Y. Sun, S. Nanda, and X. Wang, "Looking from the mirror: Evaluating iot device security through mobile companion apps." in *USENIX Security Symposium*, 2019.

[8] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *Proceedings of ACM SIGSAC CCS*, 2018.

[9] T. Mick, R. Tourani, and S. Misra, "Laser: Lightweight authentication and secured routing for ndn iot in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 755–764, 2017.

[10] H. Kong, L. Lu, J. Yu, Y. Chen, and F. Tang, "Continuous authentication through finger gesture interaction for smart homes using wifi," *IEEE Transactions on Mobile Computing*, vol. 20, no. 11, pp. 3148–3162, 2020.

[11] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. Unviersity, "Contexlot: Towards providing contextual integrity to appified iot platforms." in *NDSS*, vol. 2, no. 2, 2017, pp. 2–2.

[12] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Journal of Information Security and applications*, vol. 22, pp. 113–122, 2015.

[13] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[14] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang, "Exploring link correlation for efficient flooding in wireless sensor networks." in *NSDI*, vol. 10, 2010, pp. 1–15.

[15] J. Jun, L. Cheng, L. He, Y. Gu, and T. Zhu, "Exploiting sender-based link correlation in wireless sensor networks," in *IEEE International Conference on Network Protocols*, 2014.

[16] S. I. Alam, S. Sultana, Y. C. Hu, and S. Fahmy, "Link correlation and network coding in broadcast protocols for wireless sensor networks," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2012.

[17] M. Durvy and P. Thiran, "A packing approach to compare slotted and non-slotted medium access control," in *Infocom*, 2006.

[18] W. Wang, X. Liu, Y. Yao, and T. Zhu, "Exploiting wifi ap for simultaneous data dissemination among wifi and zigbee devices," in *IEEE International Conference on Network Protocols*, 2021.

[19] R. Tourani, S. Misra, T. Mick, S. Brahma, M. Biswal, and D. Ameme, "icens: An information-centric smart grid network architecture," in *IEEE International Conference on Smart Grid Communications*, 2016.

[20] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[22] C. Hettinger, T. Christensen, B. Ehlert, J. Humpherys, T. Jarvis, and S. Wade, "Forward thinking: Building and training neural networks one layer at a time," *arXiv preprint arXiv:1706.02480*, 2017.

[23] J. Lopez, L. Babun, H. Aksu, and A. S. Uluagac, "A survey on function and system call hooking approaches," *Journal of Hardware and Systems Security*, vol. 1, pp. 114–136, 2017.

[24] T. W. Curry *et al.*, "Profiling and tracing dynamic library usage via interposition." in *USENIX Summer*, 1994, pp. 267–278.

[25] M. Bagherzadeh, N. Kahani, C.-P. Bezemer, A. E. Hassan, J. Dingel, and J. R. Cordy, "Analyzing a decade of linux system calls," *Empirical Software Engineering*, vol. 23, pp. 1519–1551, 2018.

[26] 2023. [Online]. Available: https://github.com/miracl/MIRACL.git

[27] S. M. Kim, S. Wang, and T. He, "Exploiting causes and effects of wireless link correlation for better performance," in *IEEE Conference on Computer Communications*. IEEE, 2015, pp. 379–387.

[28] Z. Zhao, W. Dong, G. Guan, J. Bu, T. Gu, and C. Chen, "Modeling link correlation in low-power wireless networks," in *IEEE Conference on Computer Communications*. IEEE, 2015, pp. 990–998.

[29] Z. Chi, Y. Yao, T. Xie, X. Liu, Z. Huang, W. Wang, and T. Zhu, "Ear: Exploiting uncontrollable ambient rf signals in heterogeneous networks for gesture recognition," in *Proceedings of the 16th ACM conference on embedded networked sensor systems*, 2018.

[30] S. Sadowski and P. Spachos, "Rssi-based indoor localization with the internet of things," *IEEE access*, vol. 6, pp. 30 149–30 161, 2018.

[31] T.-C. Tai, K. C.-J. Lin, and Y.-C. Tseng, "Toward reliable localization by unequal aoa tracking," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 444–456.

[32] S. Zhang, P. K. Sangdeh, H. Pirayesh, H. Zeng, Q. Yan, and K. Zeng, "Authiot: A transferable wireless authentication scheme for iot devices without input interface," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 23 072–23 085, 2022.

[33] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based iot device authentication," in *IEEE INFOCOM 2017-IEEE conference on computer communications*. IEEE, 2017, pp. 1–9.