

# On the Feasibility of Reasoning about the Internal States of Blackbox IoT Devices Using Side-Channel Information

WEI SUN\*, University of California San Diego, USA

YUWEI XIAO\*, University of California San Diego, USA

HAOJIAN JIN, University of California San Diego, USA

DINESH BHARADIA, University of California San Diego, USA

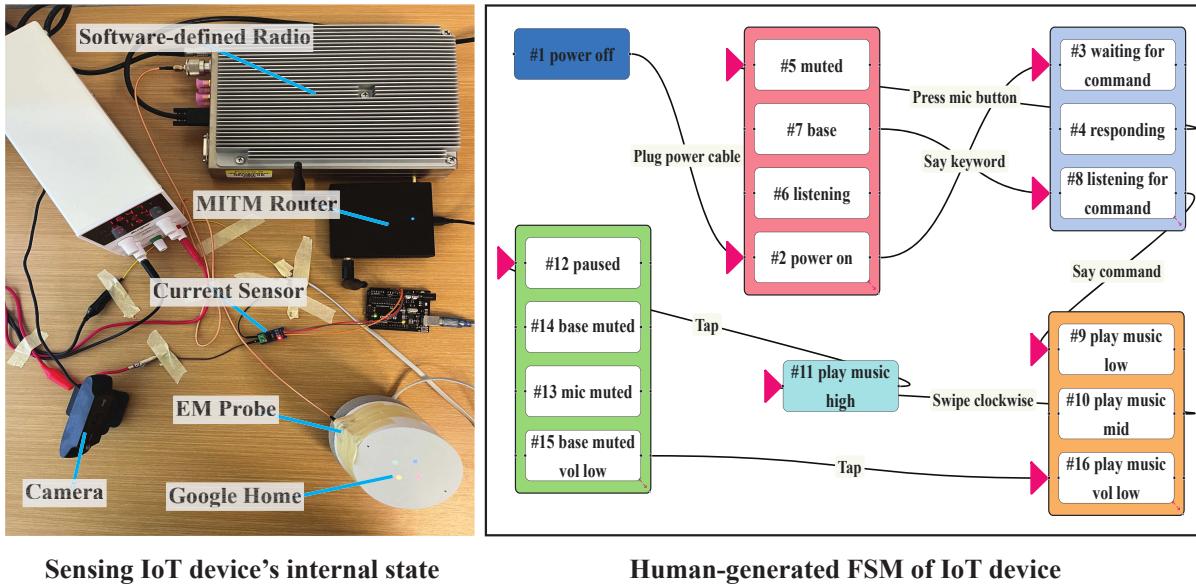


Fig. 1. IoTProsector helps users reason the internal states of black-box IoT devices (e.g., Google Home smart speaker) by sensing side-channel information (e.g., power, emanations, and network traffic) and incorporating human understandings with a user interface design, resulting in the human-generated finite state machine (FSM) of the IoT device.

Internet of Things (IoT) devices are typically designed to function in a secure, closed environment, making it difficult for users to comprehend devices' behaviors. This paper shows that a user can leverage side-channel information to reason fine-grained internal states of black box IoT devices. The key enablers for our design are a multi-model sensing technique that fuses power consumption, network traffic, and radio emanations and an annotation interface that helps users form mental models of a black box IoT system. We built a prototype of our design and evaluated the prototype with open-source IoT devices and black-box commercial devices. Our experiments show a false positive rate of 1.44% for open-source IoT devices' state probing, and our participants take an average of 19.8 minutes to reason the internal states of black-box IoT devices.

CCS Concepts: • Human-centered computing → Ubiquitous and mobile devices; Graphical user interfaces.

Additional Key Words and Phrases: Side-Channel sensing, sense-making, mental model, system images

\*Both authors contributed equally

Authors' addresses: Wei Sun, University of California San Diego, USA, w5sun@ucsd.edu; Yuwei Xiao, University of California San Diego, USA, yux075@ucsd.edu; Haojian Jin, University of California San Diego, USA, haojian@ucsd.edu; Dinesh Bharadia, University of California San Diego, USA, dineshb@ucsd.edu.

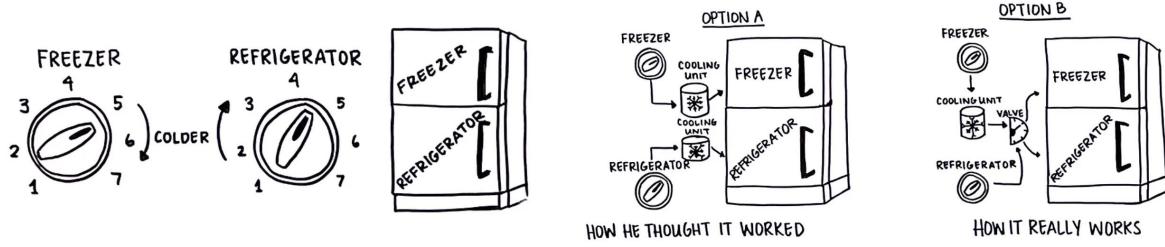


Fig. 2. Don Norman owned a two-compartment refrigerator with two controls: a freezer and a refrigerator. To his surprise, when he tried to make the freezer colder, it also made the refrigerator colder - even when he didn't change the refrigerator dial. Indeed, his mental model (option A) of how the cooling unit worked differed from how it actually worked (option B): there was only one cooling unit with a valve that dispersed the air to each compartment, instead of two respective cooling units. We adapt the image from [44].

## 1 INTRODUCTION

In "The Design of Everyday Things [38]", Don Norman used an example of a refrigerator (Fig. 2) to illustrate that users often have wrong mental models of how products work. As described by Norman, users generate a mental model for how their interactions affect the system and how the system affects them through reading product descriptions and observing system behaviors. However, unlike the refrigerator example, many low-level system behaviors in IoT devices are intangible. As a result, users have various questions regarding the internal states of these devices, such as: "Do Amazon Echo devices really stop listening, when a user presses the mute button [28]?", "Why do cameras stop recording after 30 minutes [21]?"

A number of HCI studies have tried to understand the mental models of smart home users. For example, Clark et al. [19] found that different smart home abstractions have significant priming effects on users' mental models. Blase et al. examined whether trigger-action programming (e.g., IFTTT programs) captures smart home behaviors that users actually desire [45]. In contrast to these studies, our work focuses on building a new tool to support users in forming mental models of IoT devices. Particularly, we consider the side-channel information as a new channel for understanding intangible system behaviors and designing interactions accordingly.

This paper presents IoTProsector, a system that leverages multimodal side-channel information (i.e., power consumption, network traffic, and electromagnetic emanations) to help users understand the fine-grained internal states of IoT devices. IoTProsector first records how an experimenter interacts with a target IoT device and captures the side channel information generated along the process. By clustering the sensor data distributions and analyzing the temporal transitions, IoTProsector then derives a finite state machine and helps experimenters align the IoT device's internal states with their mental models (Fig. 1).

IoTProsector has two key enablers. The first is a multi-model sensing technique that correlates the side-channel information with the IoT device's internal states. When the IoT device stays in different states, its power consumption and generated network traffic differ. For example, the Nest Cam's power consumption remained almost identical when in "indicator-off" mode (340 mA) as when fully operational (370 mA) [41]. This slight reduction correlates with the disabling of the LED power light, given that LEDs typically draw 10-20mA. Further, the electromagnetic emanations are amplitude-modulated clock signals caused by the computation activities on the IoT device, which can exhibit periodic spikes in the frequency domain, and their power spectral densities correlate with the IoT device's internal states. Therefore, we can fuse this side-channel information and leverage machine learning models (e.g., k-means) to probe the IoT device's internal states.

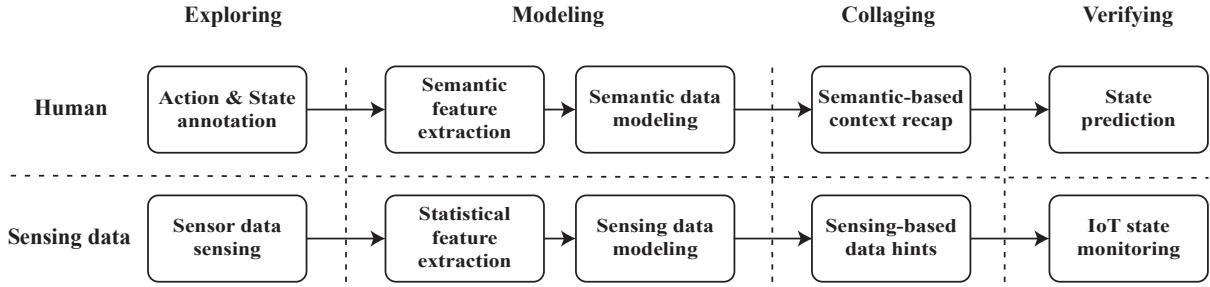


Fig. 3. IoTProsector’s workflow consisting of state exploration, sensing, modeling, collaging, and step-wise verification fuses the sensing model and human mental model for accurate and semantic IoT internal state probing.

Second, we develop an annotation user interface to bridge the gap between low-level states and high-level mental models. For example, a smart speaker that plays music at different volumes may have distinct side-channel information, but these states may share similar cognitive meanings. IoTProsector introduces a four-step workflow (explore-model-collage-verify) to guide users in forming mental models using side-channel information. IoTProsector begins by allowing users to blindly interact with the target IoT devices and collecting side-channel information. IoTProsector then identifies unique states using the side-channel information and asks users to merge redundant states. Finally, users verify the correctness of the mental model by interacting with the IoT devices step-by-step.

We built a prototype of our design and evaluated the prototype with open-source IoT devices (Google AIY Voice kits and Vision kits) and black-box commercial devices (Google Home). Our experiments show a false positive rate of 1.44% for open-source IoT devices’ state probing, and our participants take an average of 19.8 minutes to reason the internal states of black-box IoT devices. Notably, it was observed that users’ ultimate mental models, shaped by newly acquired side-channel information, continue to differ. This variation may be partly due to the lasting impact of their initial mental models, which affect the way users revise their understanding [16].

Our main contribution is (1) an IoT probing system that can probe the internal states of the black box IoT devices using the side-channel information; (2) a four-step workflow that can connect the low-level system states to the high-level states in users’ mental model; (3) Our experimental results demonstrate the precision of IoTProsector on probing IoT internal states. Our user study further confirms its capability to assist users in annotating both precise and semantically meaningful IoT states.

## 2 OVERVIEW

In this section, we present the overview system design of IoTProsector, which mainly consists of two components.

**Sensing.** We develop a multi-modal sensing technique that leverages and fuses power consumption, network traffic, and emanations to detect the internal states of IoT devices. Specifically, we extract statistical features from the sensing data and apply the TSNE algorithm [48] to identify the significant features. We then use these features to cluster the IoT device’s internal states through unsupervised machine-learning algorithms.

**Annotation.** Furthermore, we design an annotation user interface to bridge the gap between users’ cognitive understanding and low-level machine states derived from side-channel information. We illustrate this process shown in Fig. 3 as follows:

- **Exploring.** A user may first blindly explore the possible internal states of the given IoT device by interacting with the IoT device based on the instructions illustrated in IoTProsector’s graphical user interface. During the interaction, the sensing data generated by the IoT device are collected, and the

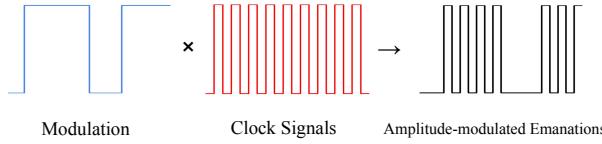


Fig. 4. The clock signals are amplitude-modulated by the computation activity on the IoT device, resulting in the amplitude-modulated clock signals that are termed emanations.

interaction actions, together with the IoT device's behaviors, are video-tapped to facilitate the later sense-making process.

- **Modeling.** In this process, the sensing data collected during exploration are utilized to extract statistical features and then develop the sensing model. In addition, the transition events between the IoT device's states are employed to merge states, thereby reducing redundancy.
- **Collaging.** To further accurately and semantically probe the IoT device's internal states, a visual and interactive display showcasing the relationship between the sensing model and the human mental model, along with the contextual information is provided to enhance the user's understanding and aid in making a collage of the IoT states. By utilizing this information, the user is expected to generate a sense-making FSM that effectively characterizes the IoT device's internal states.
- **Verifying.** After the internal states of the IoT device have been annotated and characterized, the user can verify them using the generated finite-state machine. Specifically, a machine learning classifier is trained based on the collaging results. Then, as the user interacts with the IoT device, the generated sensing data can be classified into one of these internal states. This process enables the user to monitor and analyze the states of the IoT device in real-time.

### 3 IOTPROSECTOR: SENSING

#### 3.1 Side-channel Information Characterization

**3.1.1 Power consumption and network traffic.** Every IoT device drains energy either from the power line or battery and the amount of the energy drained by the IoT device highly depends on the IoT device's state. Even though the network traffic data is usually encrypted, we can still use the network traffic pattern as the side-channel information to probe the IoT devices' internal states. This is because the network traffic introduced by the IoT device highly depends on its state.

**3.1.2 Emanations.** Every IoT device introduces electromagnetic emanations, which are amplitude-modulated clock signals. To identify these emanations from the IoT devices, we can perform a Fast Fourier Transform (FFT) on the collected emanation signals. As a result, the FFT peaks are equally separated in the frequency domain. Then, we use the power spectral density of these FFT peaks as our emanation features for IoT state probing.

**Connecting emanations to internal fine-grained states.** Every IoT device has its own clock for synchronization purposes during the computation. The clock signals (i.e., electromagnetic waves) can emit over the air directly, or go through the electronic components on the IoT device and emit over the air afterwards [43]. These clock signals can be further modulated by the computation activities on the IoT devices resulting in the amplitude-modulated clock signals as shown in Fig. 4. Intuitively, when there is a computation activity, there is clock signal leakage. Otherwise, there is no clock signal leakage. So, these clock signals are amplitude-modulated. We term these amplitude-modulated clock signals as emanations.

Since the emanations are amplitude-modulated clock signals, these emanations can carry sensitive data information about the IoT device's internal states. Specifically, IoT devices' states depend on the computation

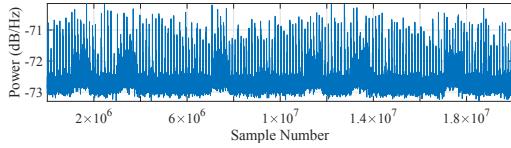


Fig. 5. Time-domain emanation signals from Google Home smart speaker exhibit on-off property, as they are amplitude-modulated clock signals.

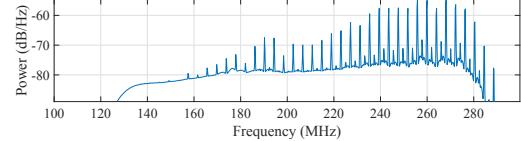


Fig. 6. Frequency-domain emanation signals from Google Home smart speaker exhibit periodic spikes spread across the spectrum, which can be leveraged to sense the IoT device's internal states using the spike's power spectral density.

activities conducted on them, which can be revealed by the emanations. So, we can predict the IoT devices' internal states based on the received emanations. In the time domain, the emanations are squared waves as they are amplitude-modulated clock signals, which can exhibit periodic spikes in the frequency domain.

**Time-domain emanations.** Since the emanations are amplitude-modulated clock signals, they will become the squared waves in the time domain. Therefore, the time-domain emanations present the on-off property. The ideal squared wave using Fourier expansion with a cycle frequency of  $f$  over time  $t$  can be represented as follows:

$$x(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2\pi(2k-1)ft)}{2k-1} \quad (1)$$

To demonstrate the on-off property of the time-domain emanations, Fig. 5 illustrates the emanations of the Google Home smart speaker at the frequency band between 100MHz and 300MHz, which exhibit the on-off property over time. This is because the emanations are amplitude-modulated clock signals.

**Frequency-domain emanations.** The amplitude-modulated clock signals (i.e., emanations) in the time domain are the squared waves. When we do the Fast Fourier Transform (FFT) on the time-domain emanations, we will have frequency-domain emanations, which will have one fundamental harmonic and other multiple harmonics with decreasing power spectral densities. Specifically, the frequency-domain emanations can be represented as follows:

$$x(f) = \sum_{k=-\infty}^{\infty} \frac{2\sin(2\pi kf_0T)}{k} \delta(f - kf_0) \quad (2)$$

where  $f_0 = \frac{1}{T}$  is the frequency of the fundamental harmonic, and  $\delta(f - kf_0)$  indicates the harmonic component at frequency of  $kf_0$  with amplitude of  $\frac{2\sin(2\pi kf_0T)}{k}$ . As we can see, the squared wave consists of an infinite number of sine wave components. Moreover, since the emanations are amplitude-modulated, they will spread over the spectrum. Said differently, each sine wave component acts as a different carrier for the modulation signals. Fig. 6 showcases the frequency-domain emanations of the Google Home smart speaker at the frequency band between 100MHz and 300MHz. As we can see, each peak in the plot represents the intermediate frequency of the harmonics, which are equally separated as the emanations are the amplitude-modulated clock signals.

The emanations propagate inside and outside the IoT device's circuit. Suppose the leaked emanations at the frequency of  $f_l$  inside the circuit, which can amplitude modulate the clock signals at the frequency of  $f_c$ . The RF transceivers further shape the emanations at the carrier frequency of  $f_{carrier}$ , resulting in the emanations emitted over the air through the transceiver's antennas. Therefore, the frequencies of the emanations received over the air can be as follows:

$$f_r = p \cdot f_{carrier} + q \cdot f_c + r \cdot f_l \quad (3)$$

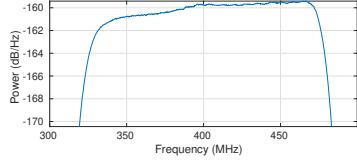


Fig. 7. FFT of IQ samples, when the Amazon Echo Dot is power-off.

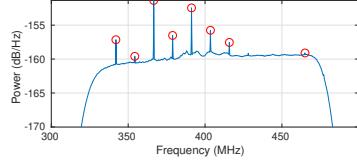


Fig. 8. FFT of IQ samples, when the Amazon Echo Dot is power-on.

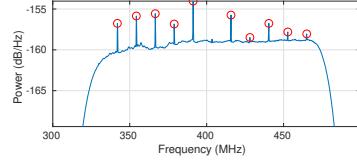


Fig. 9. FFT of IQ samples, when we interact with the Amazon Echo Dot.

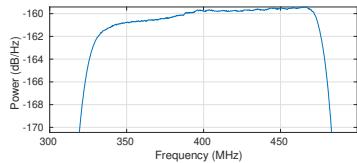


Fig. 10. FFT of IQ samples, when the Google Home smart speaker is power-off.

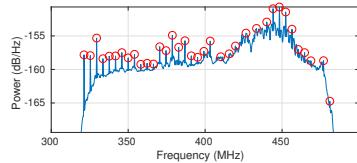


Fig. 11. FFT of IQ samples, when the Google Home smart speaker is power-on.

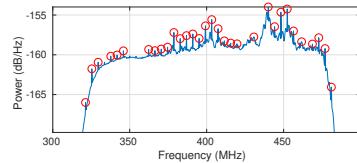


Fig. 12. FFT of IQ samples, when we interact with the Google Home smart speaker.

where  $p$ ,  $q$ , and  $r$  are integers due to the mixing and amplification of the emanations. Note that not all of these multiples are applied, which is highly dependent on the hardware architecture and components (e.g., filters) of the circuit. Some IoT devices may even not have RF transceivers, thereby the emanations are emitted through the data lines (i.e., acting as antennas) in the circuit. As we can see, the  $f_r$  of the received emanations can be highly dependent on  $f_i$  due to the computing activities on the IoT device, which is dependent on the state of the device. Therefore, we can leverage the frequency-domain analysis of the emanations to infer internal IoT states.

### 3.2 Feasibility Study

To demonstrate the feasibility of using the IoT device's emanations for internal IoT state detection, we measure the emanations from the Amazon Echo Dot and Google Home smart speaker. Specifically, Fig. 7, Fig. 8, and Fig. 9 show the frequency-domain emanations from the Amazon Echo Dot, when it is power-off, power-on, and interacting with people respectively. Fig. 10, Fig. 11, and Fig. 12 shows the frequency-domain emanations from the Google Home smart speaker, when it is power-off, power-on, and interacting with people respectively. The red circle indicates the detected FFT peaks on the frequency-domain emanations. As we can see, the Google Home smart speaker and Amazon Echo Dot have different emanation patterns when they are in different states. Moreover, the Google Home smart speaker and Amazon Echo Dot will exhibit different emanation patterns, even though they are in the same state. This is because different IoT devices will have different emanations due to the different hardware architectures.

### 3.3 Machine Learning-based IoT State Probing

Using the side-channel information collected over time during exploration, we compute features from this data as the input for the sensing model to probe the IoT device's internal states. As shown in Table 1, we list nine statistical features that we can extract from this side-channel information. Specifically, we measure the time-series network throughput for network traffic data collection, which will be used to derive the statistical features for network traffic side-channel information. Similarly, we measure the time-series power consumption of the IoT device in each state and derive the statistical features for power consumption side-channel information. However,

Statistical features	Description
MAV	mean absolute value
VAR	variance
RMS	root mean square
Std	standard deviation
MAD	median absolute deviation
Skewness	asymmetry of the data distribution
Kurtosis	shape of the data distribution
IQR	interquartile range
Energy	average sum of the squares

Table 1. Statistical features for time-series power consumption, network traffic, and electromagnetic emanations over frequencies.

for the emanation side-channel information, we first collect the time-domain IQ samples and further conduct Fast Fourier Transform (FFT) to obtain the frequency domain signals. The intuition is that the IoT device’s states are related to the power density of the spikes presented at the frequency domain IQ samples, which is illustrated in the above section. Then, we use the power density of spikes presented in frequency domain signals as the series of data streams to derive the statistical features. After we obtain the statistical features from all the side-channel information, we concatenate them to formulate a vector. Since this three side-channel information could play a different role in IoT state prediction, it’s important to only extract some important features to efficiently train the machine learning models for IoT state prediction. Therefore, we concatenate these features and use the two most important features determined by the t-SNE algorithm [46] as the input of unsupervised classification models (i.e., k-means, DBSCAN, and GMM) for IoT state probing.

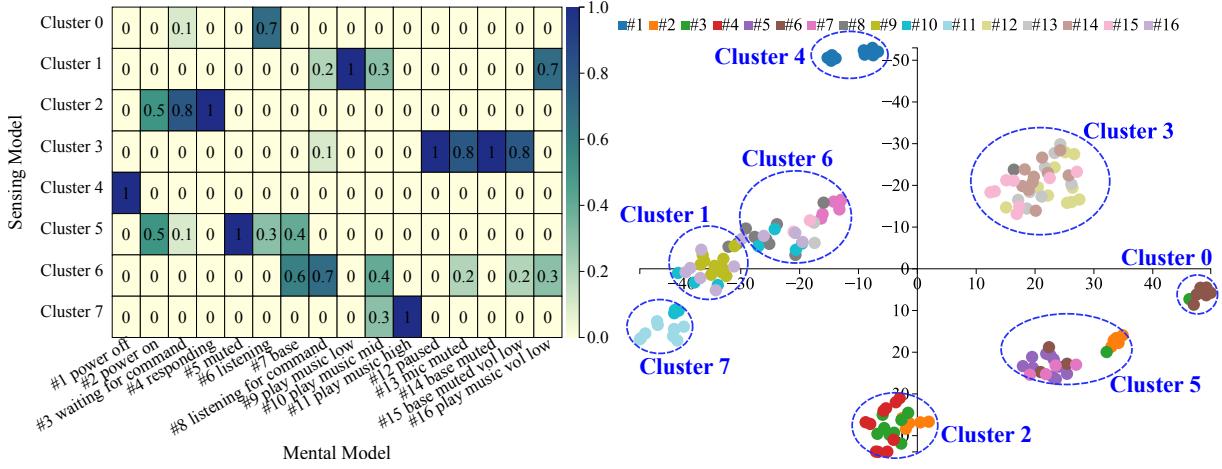
## 4 IOTPROSECTOR: ANNOTATION

### 4.1 Exploring

As the user interacts with an IoT device to explore its possible internal states, the side-channel information of this IoT device (e.g., power consumption, network traffic, and emanations) is automatically generated and collected. Simultaneously, the user-device interactions are recorded, which may cause the IoT device to transit from one state to another. Given each IoT device has a finite set of states and possible transitions between states, a Finite State Machine (FSM) can be used to effectively represent the IoT device’s states and the transition events between states. Specifically, each node in the FSM represents the IoT state and the edge connecting two nodes represents the transition event. This finite state machine can not only show the IoT device’s states and their transitions but also clearly help users monitor the IoT device’s states over time.

Aiming at probing the internal states of the IoT device, we prompt the user to annotate each state following every interaction with the IoT device. While a wide range of states are explored, the annotations of these states depend on the user’s understanding based on their observations of the IoT device’s responses and their interactions. For instance, while interacting with a Google Home smart speaker, one user might annotate the states as ‘question-answering’ and ‘music-playing’ when asking about the weather and playing a song, respectively. Another user, however, might annotate both states simply as ‘responding’.

Therefore, even though more states are probed, they suffer from being less resilient to noise and misannotation, which can introduce confusion on the IoT device’s internal state annotation and verification in the later stage. As a result, the finite state machine is over-semantic or less-semantic. Fig. 13a shows the correlation matrix derived from the sensing model and the human mental model. Each element in the correlation matrix indicates



(a) Correlation matrix of the IoT states derived from the sensing model and human mental model, where each element in the correlation matrix indicates the percentage of the human-annotated states on the cluster generated by the sensing model.

(b) Sensor data representation in 2D plane with TSNE algorithm, when human annotates the IoT states based on their understanding.

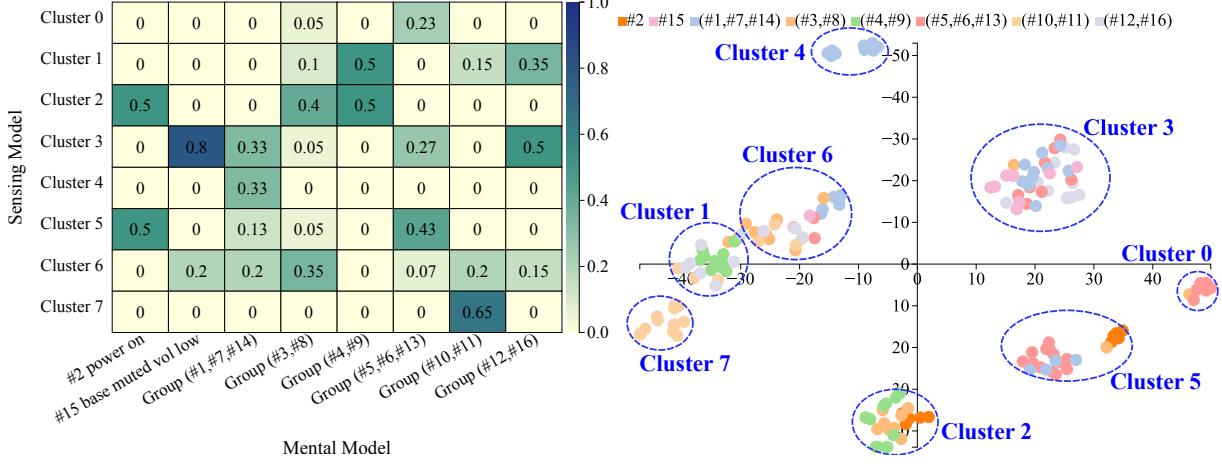
Fig. 13. Understanding human mental model-based IoT state annotation and sensing model-based IoT state representation with correlation matrix and sensor data visualization in the exploring stage.

the percentage of the data points of human-annotated states that belong to the sensing model-generated cluster. As we can see, the human mental model has generated multiple IoT states that are more than the number of IoT states indicated by the sensing model. In other words, each cluster contains multiple human-annotated states. So, there is a misinterpretation of the IoT device’s internal states for the sensing model and human mental model. To be more visual, Fig. 13b shows the processed sensor data in a 2D plane with the TSNE algorithm [48], indicating the IoT states annotated by the human model are over-semantic, as each sensing model-generated cluster may contain multiple human-annotated states with the same semantics.

#### 4.2 Modeling

Before the user engages in refining the FSM, we utilize two types of information to provide a better starting point for the user. First, we leverage the transition events between states. Given that the same transition events are likely to lead to the same state, they can be used as a hint to identify the IoT state with the same semantics. For instance, whenever an user says a keyword (e.g., ‘OK Google’) to start interacting with a Google Home smart speaker, it may always enter the same state, waiting for further commands. Therefore, we merge the states derived from the same transition event to reduce semantics redundancy. Second, we process the sensing data collected during exploration to develop a sensing model as described in Sec 3. This sensing model offers insights from the statistical perspective and classifies initial states into different clusters.

However, while the sensing model itself could serve as a direct characterization of the IoT device’s internal states, it is challenging for an user to understand the meaning of internal state clusters without annotation, which further hinders probing processes. Moreover, the FSM still suffers from inaccurate state representation. This is because the same transition events may lead to different states, meaning the FSM may not accurately correspond to the actual internal states of the IoT device. Fig. 14a shows the correlation matrix of the IoT device’s



(a) Correlation matrix of the IoT states derived from the sensing model and human mental model using transition event as a hint for state annotation, where each element indicates the percentage of the human-annotated states on the cluster generated by the sensing model.

(b) Sensor data representation in the 2D plane with TSNE algorithm, when the states are merged based on the transition events.

Fig. 14. The IoT states merging based on the transition states can be helpful for the semantic IoT states annotation.

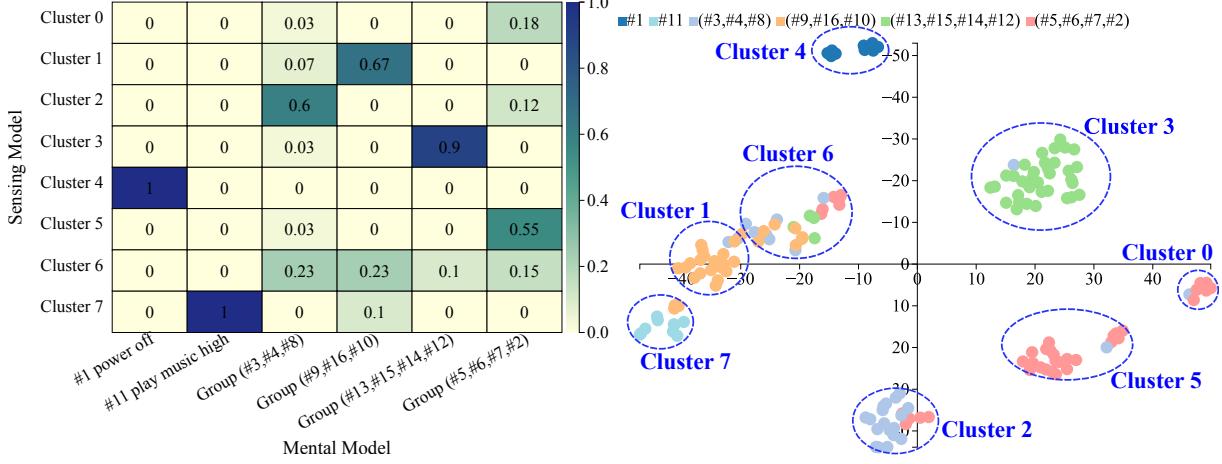
states derived from the sensing model and human mental model after merging the states derived from the same transition event. As we can see, despite the number of human-annotated states decreasing due to the merging of states, merged states are still scattered across different clusters, which can also be demonstrated through the scatter plot of the sensor data with the TSNE algorithm in Fig. 14b.

#### 4.3 Collaging

To further accurately and semantically probe the internal states of the IoT device, we leverage the sensing model since it can indicate the IoT states through the sensor data representation. As such, the sensor data representing the IoT states with the same semantics can formulate a cluster and further be collaged as one state.

To this end, we integrate the sensor data representation from the sensing model with the annotations and transition events from the mental model into a visual and interactive display. This display illustrates the relationship between mental model-introduced states and sensing model-introduced states through a correlation matrix and a sensor data representation. Along with this display, we also provide context information in our user interface (see Fig. 17), which includes the recording of state information and transition events, to help users with state annotation. By interacting with the user interface, the user can leverage the correlation matrix and sensor data representation from both sensing and mental models to further make a collage of the annotated states. Thus, the collages of states remain semantically understandable to users while becoming more reliable with the aid of underlying low-level data information.

Fig. 15a shows the correlation matrix of the IoT states derived from the sensing model and human mental model after the user makes a collage. As we can see, the number of states derived from the sensing model is close to the number of states annotated by humans. This is because each sensing model-derived state has a corresponding human mental model-derived state. This can also be demonstrated through Fig. 15b showing the scatter plot of the sensor data with the TSNE algorithm after making a collage. As we can see, the clusters



(a) Correlation matrix of the IoT states derived from the sensing model and human mental model after making all collages, where each element indicates the percentage of the human-annotated states in the cluster generated by the sensing model.

(b) Sensor data representation in the 2D plane with TSNE algorithm after human finishes making all the collages.

Fig. 15. By fusing the sensing model with the mental model for collaging, the states of generated FSM show high coherence. This indicates that human understanding aligns well with the sensing data derived from the sensing model.

derived from the sensing model mainly consist of one state indicated by the human mental model. As a result, the finalized FSM based on the sensing model and human mental model should have semantic states that are capable of representing the IoT device's internal states.

#### 4.4 Verifying

After the user extensively exploit the IoT device's internal states, IoTProsector can generate a finite state machine of this IoT device. As this IoT device is deployed in the physical environment, its internal states should be indicated through the generated finite state machine. To demonstrate the efficiency of the generated finite state machine, we need to verify that IoTProsector can accurately probe the IoT device's state with the well-fused sensing model and human model obtained in the above sections.

To do so, IoTProsector utilizes the generated finite state machine to train a classifier. Specifically, the statistical features derived from Sec 4.2 are used as data inputs and the annotations of the states serve as labels. During the verification, as the user interacts with the IoT device over time, the generated side-channel information can be used to predict the current IoT state with the well-trained classifier. Fig. 16 showcases the step-wise verification when the user interacts with the Google Home smart speaker. The top figure shows the state transition of the Google Home smart speaker during the interaction and the bottom figure shows the over-time variation of the Google Home smart speaker's finite-state machine. As we can see, when the speaker is powered up, it enters the waiting-for-keyword state. After we say the keyword (i.e., 'OK Google'), it enters the waiting-for-command state. After we ask it to play music (i.e., say a command), it enters the playing-music state. During this process, the colored node in the finite state machine generated by the sensing model and human mental model indicates the Google Home smart speaker's current state and the other states are indicated by the white nodes.

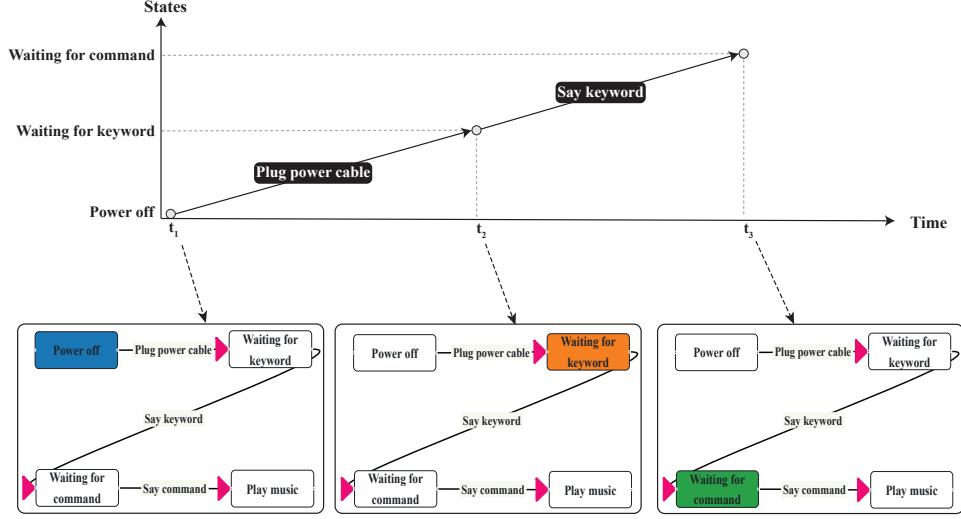


Fig. 16. Step-wise verification for the Google Home smart speaker. The top figure showcases the Google Home smart speaker’s three states when we interact with it over time. The bottom figure shows the finite state machine of the Google Home smart speaker over time. As we interact with the Google Home smart speaker, its state changes over time which is indicated by the colored node in the finite state machine.

## 5 IMPLEMENTATION

**Hardware and Software.** IoTProsector utilizes a combination of power consumption, network traffic, and emanations to sense the internal states of IoT devices. The experimental setup is shown in Fig. 1. We use a signal hound [9] for spectrum sensing to extract the emanations from the device. We use a power sensor (i.e., ACS172 [1]) connected with the Arduino to measure the real-time power consumption of the IoT device (e.g., Google Home smart speaker [6], Google AIY voice kit [4]). We use TShark [20] to collect real-time network traffic data from the IoT device. The data is streamed to a desktop, where features are extracted in real-time and used to run the sensing model and human mental model for the IoT device’s internal state probing. To have a well-trained machine-learning model for IoT state probing, we collect 100 measurements across three side-channel information for each IoT device’s state. Then, we simply split the collected dataset into 80% for training and 20% for testing. We evaluate the performance of our system with two white-box IoT devices (i.e., Google AIY voice kit and vision kit) and one black-box IoT device (i.e., Google Home smart speaker). In our evaluation, we explored three states for the Google AIY vision kit and five states for the Google AIY voice kit. Since we conducted a user study to explore the states of the Google Home smart speaker with 10 volunteers, the explored IoT states should be different for different volunteers depending on their understanding of the IoT device’s states.

**User Interface Design.** To assist the users in probing the internal states of IoT devices, as shown in Fig. 1, we develop a graphical user interface (GUI) which is illustrated in Fig. 17 consisting of an interactive chart workspace, context information, and visually plotted sensor data to assist user’s state annotation and collaging. We utilize the React framework [7] in JavaScript for the front-end design and FastAPI [3] in Python for the back-end design. For the front-end design, we employ D3.js [2] and React Flow [8] to illustrate the finite state machine, correlation

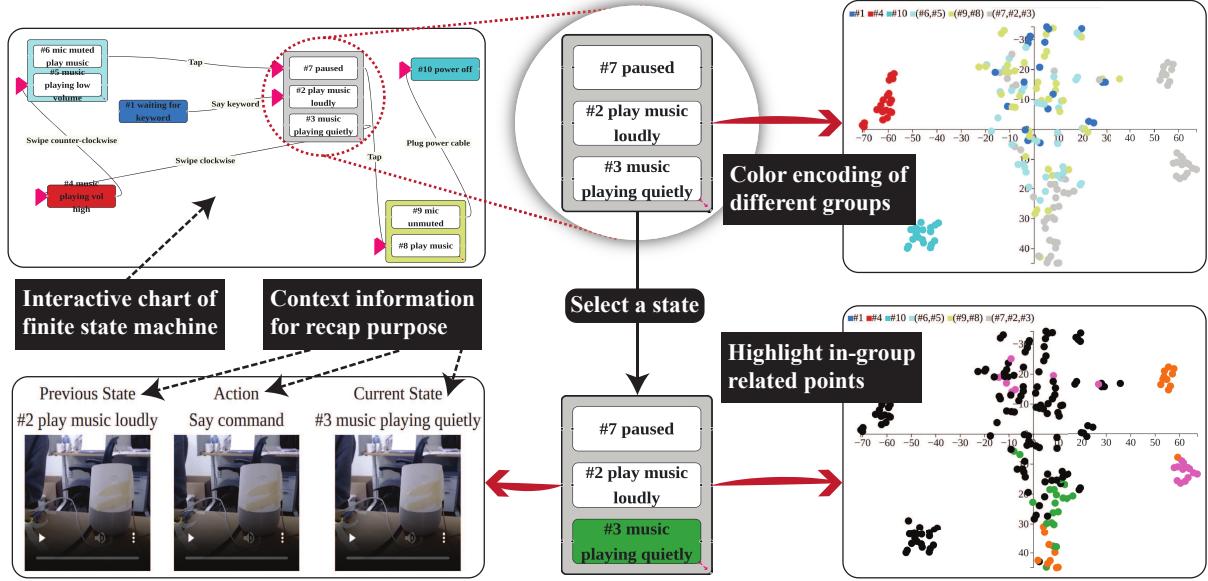


Fig. 17. IoTProsector’s graphical user interface consists of an interactive chart workspace for users to annotate and collage the states, context information, and visually plotted sensor data to assist user’s state annotation and collaging. The zoomed-out figure shows the state collaged by humans based on the sensor data representation and context information.

matrix, and scatter plot that can showcase the data distribution in the 2D plane to assist the user’s state annotation and probing.

**Experimental Settings for White-box IoT State Sensing.** We evaluate the performance of the white-box IoT state sensing with Google AIY vision and voice kits. Since we can program these two kits, we can obtain the ground-truth IoT states based on what pieces of the codes are executed. For the sake of simplicity, we obtained the five ground-truth states of the Google AIY voice kit and three ground-truth states of the Google AIY vision kit. For each IoT state of Google AIY vision and voice kit, we collect the side-channel information 100 times for training and testing. We evaluate the performance of the IoT states probing with precision, recall, F1 score, and confusion matrix using DBSCAN, GMM, and k-means.

**Experimental Settings for Black-box IoT State Sensing.** We evaluate the performance of the black-box IoT state probing with commercial off-the-shelf Google Home smart speaker through a user study. More details can be found in Sec. 6.2.

## 6 EXPERIMENTAL RESULTS

### 6.1 White-box IoT Device Probing Evaluation

To obtain the ground-truth of the internal states of IoT devices, we test IoTProsector with open source hardware (Google AIY voice kit and vision kit). Fig. 29 illustrates the finite-state machine of the Google AIY voice kit. When the Google AIY voice kit is powered up and enters interaction mode, it first accesses the Internet (i.e., Internet access state). Then, it waits for the voice commands/queries (i.e., listening state). After we query the Google AIY voice kit, it enters the speech processing state to understand the commands or queries. Finally, it responds to us by finding the answers from the Internet.

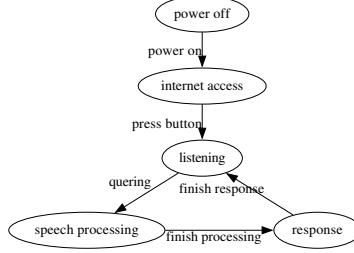


Fig. 18. Finite state machine for Google AIY voice kit during the interaction.

		True Label					True Label					True Label							
		response#1	0.57	0.01	0.3	0.12	0.0	response#1	0.95	0.0	0.02	0.03	0.0	response#1	0.55	0.01	0.29	0.15	0.0
True Label	internet_access#2	0.0	0.99	0.01	0.0	0.0	internet_access#2	0.03	0.97	0.0	0.0	0.0	True Label	internet_access#2	0.0	0.99	0.01	0.0	0.0
	listening#3	0.0	0.0	0.99	0.01	0.0	listening#3	0.03	0.0	0.95	0.02	0.0		listening#3	0.0	0.0	0.87	0.13	0.0
	speech_processing#4	0.0	0.0	0.14	0.86	0.0	speech_processing#4	0.08	0.0	0.07	0.85	0.0		speech_processing#4	0.0	0.0	0.12	0.88	0.0
	power_off#5	0.0	0.0	0.0	0.0	1.0	power_off#5	0.0	0.0	0.0	0.0	1.0		power_off#5	0.0	0.0	0.0	0.0	1.0
		#1	#2	#3	#4	#5		#1	#2	#3	#4	#5		#1	#2	#3	#4	#5	
		Predicted Label						Predicted Label						Predicted Label					

Fig. 19. Multimodal sensor fusion-based IoT state detection with k-means for Google AIY voice kit.

Fig. 20. Multimodal sensor fusion-based IoT state detection with DBSCAN for Google AIY voice kit.

Fig. 21. Multimodal sensor fusion-based IoT state detection with GMM for Google AIY voice kit.

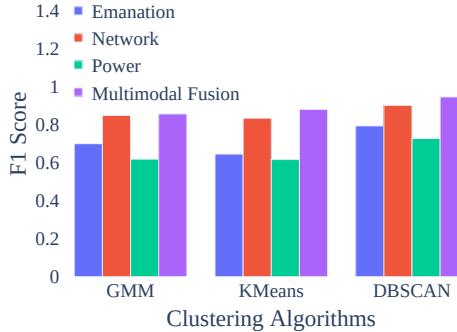


Fig. 22. F1 score of IoT state detection using k-means, DBSCAN, and GMM for Google AIY voice kit.

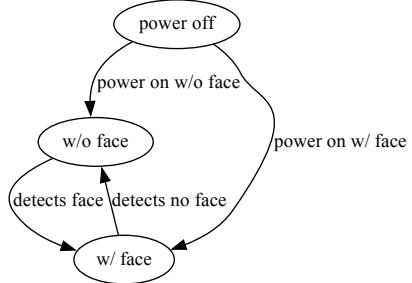


Fig. 23. Finite state machine for Google AIY vision kit.

**Performance with Google AIY voice kit:** Fig. 19, Fig. 20, and Fig. 21 show the confusion matrix for multimodal sensor fusion-based IoT state probing using k-means, DBSCAN, and GMM algorithms for Google AIY voice kit. As we can see, multimodal sensor fusion with k-means, DBSCAN, and GMM algorithms can achieve a high IoT state probing accuracy of around 0.93. Fig. 22 shows the F1 score for IoT state probing using K-means, DBSCAN, and GMM algorithms. From the results, it is clear that multimodal sensor fusion-based IoT state probing performs better than power consumption-based, network traffic-based, and emanation-based IoT state detection. Specifically, the multimodal sensor fusion-based IoT state detection with GMM, k-means, and DBSCAN algorithm

True Label	w/o face#1	0.47	0.53	0.0
	w/ face#2	0.39	0.61	0.0
	power_off#3	0.0	0.0	1.0
Predicted Label	#1	#2	#3	

Fig. 24. Multimodal sensor fusion-based IoT state detection with k-means for Google AIY vision kit.

True Label	w/o face#1	0.91	0.09	0.0
	w/ face#2	0.05	0.95	0.0
	power_off#3	0.0	0.1	0.9
Predicted Label	#1	#2	#3	

Fig. 25. Multimodal sensor fusion-based IoT state detection with DBSCAN for Google AIY vision kit.

True Label	w/o face#1	0.47	0.53	0.0
	w/ face#2	0.4	0.6	0.0
	power_off#3	0.0	0.0	1.0
Predicted Label	#1	#2	#3	

Fig. 26. Multimodal sensor fusion-based IoT state detection with GMM for Google AIY vision kit.

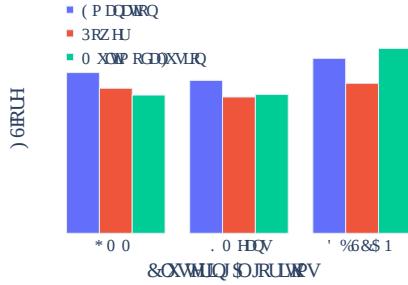


Fig. 27. F1 score of IoT state probing using k-means, DBSCAN, and GMM for Google AIY vision kit.

	Precision	Recall
GMM	0.88	0.86
KMeans	0.91	0.88
DBSCAN	0.95	0.94

Table 2. Precision and recall with k-means, GMM, and DBSCAN for Google AIY Voice Kit’s state probing.

	Precision	Recall
GMM	0.69	0.69
KMeans	0.69	0.69
DBSCAN	0.93	0.92

Table 3. Precision and recall with k-means, GMM, and DBSCAN for Google AIY Vision Kit state probing.

can achieve an F1 score of 0.86, 0.84, and 0.94 respectively, which is larger than the network traffic-based, power consumption-based, and emanation-based IoT state detection. This is because multimodal sensor fusion leverages multiple side-channel information for IoT state probing, thereby differentiating the internal IoT state. Table 2 presents the precision and recall with GMM, k-means, and DBSCAN for Google AIY Voice Kit state detection. As we can see, the best precision and recall are provided by the DBSCAN algorithm for the kit’s state detection, which is 0.95 and 0.94 respectively. This is because the DBSCAN algorithm can automatically detect the cluster number and handle the arbitrary-shaped clusters.

**Performance with Google AIY vision kit:** The Google AIY vision kit has three states: power-off state, without face detection state, and with face detection state. The state machine of the Google AIY vision kit is shown in Fig. 23. Note that the Google AIY vision kit does not introduce any network traffic, as all the computation is locally in the vision kit. This further motivates us to use a multimodal sensor fusion-based method for internal IoT state probing. Fig. 24, Fig. 26, and Fig. 25 present the confusion matrix for multimodal sensor fusion-based IoT state detection with k-means, GMM, and DBSCAN for Google AIY vision kit respectively. As we can see, the classification accuracy for multimodal sensor fusion-based IoT state detection with k-means, GMM, and DBSCAN is 0.69, 0.69, and 0.92 respectively. This is because the DBSCAN algorithm can automatically identify the cluster number and well represent the arbitrary-shaped clusters. Fig. 27 presents the F1 score of IoT state detection using k-means, DBSCAN, and GMM for the Google AIY vision kit. As we can see, the best F1 score of 0.92 is provided by the multimodal sensor fusion method with the DBSCAN algorithm. The multimodal sensor fusion-based methods with k-means and GMM algorithms do not perform better than the single model-based

#	Gender	Education Level	Experience of working with IoT device	Experience with smart speaker
P1	Male	Undergraduate	No	No
P2	Female	Undergraduate	No	No
P3	Male	Graduate	Yes	No
P4	Male	Graduate	No	No
P5	Male	Undergraduate	No	Amazon Alexa
P6	Male	Graduate	No	No
P7	Male	Graduate	Yes	Xiaomi Mi
P8	Male	Graduate	No	No
P9	Male	Graduate	Yes	Google Home
P10	Male	Undergraduate	No	Google Home

Table 4. Information of participants: major, research, or interaction experience with IoT devices and smart speakers.

methods due to the performance limitations of k-means and GMM algorithms. Table 3 presents the precision and recall with GMM, k-means, and DBSCAN for Google AIY Vision Kit state detection. As we can see, the DBSCAN algorithm can provide a precision of 0.93 and a recall of 0.92 for the vision kit’s state detection, which is better than the GMM and k-means algorithms due to its automatic cluster number estimation and arbitrary-shaped clusters characterization.

## 6.2 User Study

To evaluate the effectiveness of the probing process supported by IoTProsector, we conducted a user study with 10 college students. In our study, we selected the Google Home smart speaker as the black-box IoT device to examine the following research questions:

- RQ1** What are IoTProsector’s effects on users’ performance in probing internal states of IoT devices?
- RQ2** What are IoTProsector’s effects on the users’ cognitive load of probing internal states of IoT devices?
- RQ3** How do users think about the probing process using IoTProsector?

**6.2.1 Ethical Considerations.** Our study was approved by our institution’s IRB. All participants consented to have their data recorded and reported in a scholarly publication. Collected data were anonymized after collection and stored in a private location accessed only by the authors.

**6.2.2 Procedure.** The user study consists of four steps: pre-interview, system introduction, probing, and post-interview.

- **Pre-interview.** After the participants signed the consent forms, we first conducted an interview about the participants’ background, demographic information, prior experience of using IoT devices, etc.
- **System introduction.** Then, we set up a Google Home smart speaker with IoTProsector in a typical indoor office environment. We gave each participant a 15-minute tutorial, during which we systematically introduced them to all the functions of IoTProsector, provided them with an instruction table listing all feasible interactions with the Google Home smart speaker [5], and allowed them to try these interactions to become familiar with the device.
- **Probing.** After the participant was familiar with the user interface and system, the participant was presented with the graphical user interface as shown in Fig. 17 and instructed to interact extensively with the IoT device. He/she was encouraged to explore a wide range of interactions, annotate and refine the FSM to match the mental model and sensing model, and subsequently verify the generated FSM.

		Average	Median	Std
Granularity	Number of explored states	8.2	9.0	1.3
	Number of collaged states	4.8	4.5	1.7
Time efficiency (minutes)	Exploring	9.0	9.5	2.4
	Collaging	7.6	8.0	2.5
	Verifying	3.0	3.0	1.0
Total		20.1	19.0	4.5
Correctness	State verification accuracy (correct verification/total verification)	5.8/8.2		

Table 5. Evaluation of user study on probing time in different probing stages, number of explored/collaged states, and state probing accuracy.

	Average	Median	Std
Mental demand	5.6	5	3.5
Physical demand	3.6	2	2.8
Temporal demand	3.7	5	2.5
Performance	4.1	4	2.6
Effort	4.7	5	3.2
Frustration	3.0	2	2.6

Table 6. Evaluation of user study with NASA task load index.

- **Post-interview.** After the probing was done, the participant was asked to complete a NASA TLX test [25]. We also interviewed the participants about their feelings and rationale of operations based on the participant observation of the probing process.

**6.2.3 Participants.** We recruited 10 college students as participants. Table 4 enumerates a breakdown of the participant information. Participants are all majoring in STEM fields (i.e., electrical and computer engineering, computer science and engineering, or data science). One participant self-identified as female, and the remaining students identified as male. Participants P1 and P3 had research experience working with IoT devices. Participants P5 and P8 had used Google Home smart speakers. The other participants either had experience with similar devices (i.e., Amazon Alexa smart speaker, Xiaomi Mi smart speaker) or were not familiar with IoT devices.

**6.2.4 Quantitative Results.** To answer RQ1 and RQ2, we conducted a quantitative analysis.

**Performance.** Table 5 shows the performance of the user study quantitatively. Specifically, we mainly exploit three aspects of the results: (1) quantity of annotated states across different modules, (2) probing time across different modules of IoTProsector, and (3) accuracy of states probing when the participants use IoTProsector for IoT states annotation. As we can see from the table, all Participants finished probing the Google Home smart speaker in a relatively short time (i.e., an average time of 20.1s, a median time of 19.0s, a time standard deviation of 4.5s) with internal states probing (i.e., an average time of 4.8s, a median time of 4.5s, a standard deviation time of 1.7s). Furthermore, participants can verify the IoT states with an accuracy of 5.8/8.2, where 5.8 indicates the average number of correct verified states and 8.2 indicates the average number of states probed by the participants using IoTProsector.

**Cognitive Load.** Table 6 shows the statistical results of the participants probing the Google Home smart speaker with the metrics from the NASA task load index on a scale of 1 to 21, where a lower score indicates a lower cognitive load. This index assesses the overall workload of a participant by measuring performance across six dimensions: (1) mental demand, (2) physical demand, (3) temporal demand, (4) performance, (5) effort, and (6) frustration. The average, median, and standard deviation value across these six dimensions is around 4.12, 3.83, and 2.87 respectively. As we can see, the task load measured by these metrics is quite low due to the friendly GUI design, indicating IoTProsector’s capability of helping participants probe the IoT device’s internal states with low cognitive load.

**6.2.5 Qualitative Feedback.** To answer RQ3, we utilized recordings and transcripts of the interviews, conducted inductive thematic analysis [17], and held meetings to review the analysis process and discuss the findings.

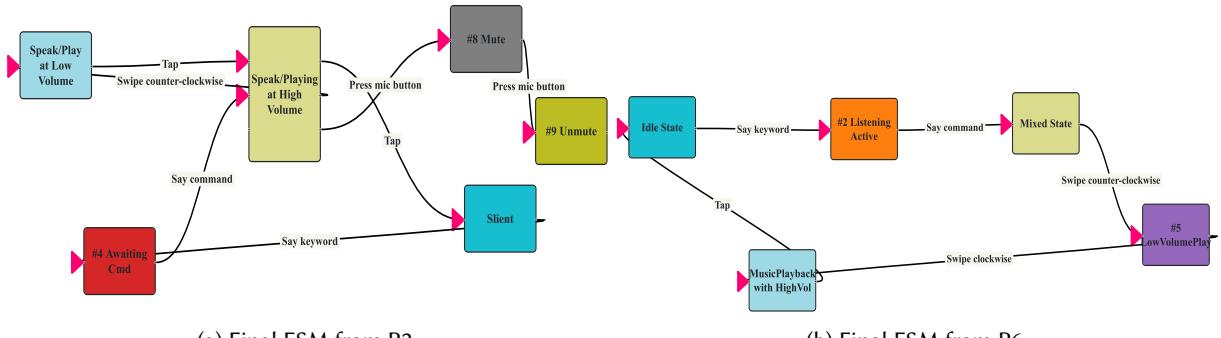


Fig. 28. Users may have different interpretations of the IoT device's internal states. This variation may be partly due to the lasting impact of their initial mental models, which affect the way users revise their understanding [16].

- **Transparency.** IoTProsector's ability to enhance the transparency of IoT devices, especially the efficiency of probing and verifying the black-box IoT device's states was acknowledged by the participants. One of the key aspects they appreciated was the generated FSM, which made the understanding of the device's functioning more accessible. As P9 mentioned, "*I have used the Google Home smart speaker before, but I never got to know what it actually does inside, its operations are often opaque. This system allows me to understand the device's functioning through the finite state machine*". The transparency is further enhanced by the use of transition events. P2 noted the utility of this feature, saying, "*With the transition event, I can easily recall and track the interactions with the device. It also gives me a clearer understanding of how the device responds to different inputs, allowing me to establish a meaningful connection between different internal states of the IoT device*". Another important design is the data representation. As P5 expressed, "*The abundance of data makes me feel a bit overwhelmed at the beginning. Nevertheless, being able to see the low-level data and my annotations side by side truly enhances my comprehension of the IoT device*". P1 also commented on its effectiveness, stating, "*The scatter plot clearly illustrates the low-level data with its interactive attributes and color encoding, which enables me to probe the internal states of the IoT device in a more intuitive way. To be more specific, the interactive features provide me with rapid feedback dynamically during collaging, and the color encoding enhances the clarity of data representation, making it easier to identify and interpret different states*".
- **Learnability.** All participants discussed the learnability of IoTProsector. Some people found this system, especially the friendly graphical user interface, easy to learn and efficient to probe the IoT device's state. P9 said, "*The system is new to me, but the workflow is quite reasonable and intuitive*". Conversely, some participants shared a different opinion that "*It took me some time to realize what I should do*" (P8). This result echos the fact that only three out of the ten participants had experience working with IoT devices.
- **Usability.** When discussing the usability of IoTProsector, the design of the user interface received appreciation from participants (P1-6, P9, P10). P6 noted "*UI is nice and user-friendly*", while P7 mentioned, "*the layout and interaction design of the UI make it easy to use*".
- **Diversity of human mental models.** Participants' answers vary when asked about how they annotate states during the exploration stage. Some participants annotated the states based on their understanding and observation. For example, P1 mentioned that "*It's mainly based on my intuition and understanding*", and P6 noted that "*I annotate them based on my observation about what the device will perform*". However, other participants like P7 relied on different information, saying, "*from the actions I conducted, I can know*

*the state it enters*". This result echoes the situation of semantic redundancy during the exploration stage and implies that the FSM is highly personalized. It also indicates the necessity for a human-in-the-loop design approach, rather than a one-size-fits-all solution.

- **Integration of sensing model and mental model.** Furthermore, to explore how the human-in-the-loop design improves understanding of black-box IoT devices, we asked participants about how they balance the mental model and the sensing model during collaging. A few participants (P1, P3, P4) preferred relying solely on a single model, with statements like "*I collage the states with the same semantics*" from P4, or "*I just collage the states based on the scatter plot*" by P1. However, the majority (P2, P5-10) actively combined both models. For instance, P5 described their approach as "*I make high-level groups based on my understanding, and then refer to the sensing model for verification*". This result underscores the significant roles both the sensing model and the mental model play in the probing process. It also indicates our design objective of incorporating human intelligence alongside sensing techniques within the probing framework.

## 7 RELATED WORK

**Side-channel sensing.** Many studies have investigated the side-channel information of IoT devices, such as power consumption [27], network traffic [12, 15, 23, 39, 47], wireless communication [26, 50], and acoustic emanations [13, 52]. However, these works cannot characterize the fine-grained IoT states. For example, Light auditor [27] leverages power consumption measurements to identify the malicious behaviors of exfiltrating information from smart bulbs, while these malicious behaviors are not classified. Network traffic-based IoT state prediction is straightforward, as every IoT device needs to have network access, and the network traffic pattern can be exploited to predict IoT states. For example, IoTAthena [47] leverages the raw time-stamped IP packets to predict IoT activities in a coarse-grained manner. Emanations from IoT devices have been exploited to detect the existence of IoT devices (e.g., hidden spy camera detection for human privacy [33]), while they have never been used for IoT state prediction. Electromagnetic sensing has been widely explored for human-computer interaction [30–32], which has different intents compared to our EM noise-based IoT state detection. For example, EM-Sense [32] leverages the EM noise generated by everyday electrical and electromechanical objects to achieve touch recognition.

**IoT program debugging.** Programming language technologies have been extensively employed in the realm of IoT for security and privacy analysis [10]. For instance, TZSlicer [49] introduces a framework that automatically identifies code segments requiring protection. IOTA [36], a core calculus for IoT automation, facilitates the development of conflict detection and provenance in home automation programs. IoTSan [37] is another framework that utilizes model checking to pinpoint undesirable cyber states for IoT devices, offering counter-examples to illustrate the underlying causes. SOTERIA [18] extracts state models from IoT code, assisting in the detection of security, safety, and functional errors. However, these studies predominantly address IoT device behaviors through a white-box approach, assuming the availability of the device's source code. In contrast, our work focuses on the underexplored area of black-box IoT devices raised in [24] and novel device mechanisms, aiming to enhance support for black-box debugging requirements.

**Tools for IoT privacy.** Considerable efforts have been devoted to creating new tools that strengthen privacy protection in the IoT domain [22, 29, 51]. For example, Ren et al.[40] utilized multidimensional analysis to characterize information exposure in IoT devices, emphasizing the importance of privacy preservation. Numerous studies[11, 14, 34, 35] have categorized IoT devices by analyzing network traffic patterns for authentication purposes. Additionally, Saidi et al.[42] developed a scalable approach to detecting IoT devices in real-world scenarios using network traffic data. IoT Inspector[23], an open-source tool, facilitates the examination of network traffic for IoT devices in home networks to ensure data privacy. These studies typically examine general IoT

privacy on a macro level, focusing on high-level device behavior and the data transmitted. In contrast, our work aims to support privacy enthusiasts at a micro level, enabling them to debug the fine-grained internal state of each individual device.

## 8 CONCLUSION

In this paper, we design IoTProsector, a system that can probe the fine-grained internal states of black box IoT devices using side-channel information such as power consumption, network traffic, and emanations. Furthermore, we design an annotation interface to semantically probe and verify the IoT device’s internal states. Our experimental results and user study demonstrate the feasibility of using side-channel information to help users form more accurate mental models.

## ACKNOWLEDGEMENTS

This work is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via [2021-2106240007]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. We also thank Raymond Song, Keith Kwong, and Chan Heng Chan for the early experiments.

## REFERENCES

- [1] 2023. ACS 172 current measuring sensor. <https://www.seeedstudio.com/blog/2020/02/15/acs712-current-sensor-features-how-it-works-arduino-guide/>
- [2] 2023. D3.js. <https://d3js.org>
- [3] 2023. FastAPI. <https://fastapi.tiangolo.com/>
- [4] 2023. Google AIY voice kit. <https://aiyprojects.withgoogle.com/voice/>
- [5] 2023. Google Home control guide. [https://support.google.com/googlenest/answer/7072889?hl=en&ref\\_topic=7196346&sjid=8515527651049881371-NC#zippy=%2Cgoogle-home](https://support.google.com/googlenest/answer/7072889?hl=en&ref_topic=7196346&sjid=8515527651049881371-NC#zippy=%2Cgoogle-home)
- [6] 2023. Google Home devices. [https://store.google.com/category/nest\\_speakers?hl=en-US](https://store.google.com/category/nest_speakers?hl=en-US)
- [7] 2023. React. <https://react.dev>
- [8] 2023. React flow. <https://reactflow.dev/>
- [9] 2023. signal hound. <https://signalhound.com/>
- [10] Julius Adebayo, Michael Muellly, Ilaria Liccardi, and Been Kim. 2020. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429* (2020).
- [11] Nesrine Ammar, Ludovic Noirie, and Sébastien Tixeuil. 2020. Autonomous identification of IoT device types based on a supervised classification. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.
- [12] Noah Aphorpe, Dillon Reisman, and Nick Feamster. 2017. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *arXiv preprint arXiv:1705.06805* (2017).
- [13] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE, 3–11.
- [14] Jiaqi Bao, Bechir Hamdaoui, and Weng-Keen Wong. 2020. Iot device type identification using hybrid deep learning approach for increased iot security. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 565–570.
- [15] Bruhadshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral fingerprinting of iot devices. In *Proceedings of the 2018 workshop on attacks and solutions in hardware security*. 41–50.
- [16] Peter A Bibby and Stephen J Payne. 1996. Instruction and practice in learning to use a device. *Cognitive Science* 20, 4 (1996), 539–578.
- [17] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. <https://doi.org/10.1191/1478088706qp063oa>
- [18] Z Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated iot safety and security analysis. In *2018 {USENIX} Annual Technical Conference ({\{}USENIX{\}}{\{}ATC{\}} 18)*. 147–158.
- [19] Meghan Clark, Mark W Newman, and Prabal Dutta. 2017. Devices and data and agents, oh my: How smart home abstractions prime end-user mental models. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–26.

- [20] Gerald Combs et al. 2012. Tshark—dump and analyze network traffic. *Wireshark* (2012).
- [21] HackerNews. 2023. Ask HN: Why do cameras stop recording after 30 minutes? | Hacker News. <https://news.ycombinator.com/item?id=34640107>. (Accessed on 11/15/2023).
- [22] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlene Fernandes, Josiah Hester, and Blase Ur. 2021. SoK: Context sensing for access control in the adversarial home IoT. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 37–53.
- [23] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–21.
- [24] Trung Dong Huynh, William Seymour, Luc Moreau, and Jose Such. 2023. Why Are Conversational Assistants Still Black Boxes? The Case For Transparency. *arXiv preprint arXiv:2306.05218* (2023).
- [25] Load Index. 1990. Results of empirical and theoretical research. *Advances in* (1990).
- [26] Haojian Jin, Cheng Xu, and Kent Lyons. 2015. Corona: Positioning adjacent device with asymmetric bluetooth low energy rssi distributions. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 175–179.
- [27] Woosub Jung, Kailai Cui, Kenneth Koltermann, Junjie Wang, ChunSheng Xin, and Gang Zhou. 2022. Light Auditor: Power Measurement Can Tell Private Data Leakage through IoT Covert Channels. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. 518–532.
- [28] Dennis Knake. 2021. Echo Mute Button: Does Alexa really stop listening? - We speak IoT. <https://www.wespeakiot.com/echo-mikrofon-taste/>. (Accessed on 11/15/2023).
- [29] Ievgeniia Kuzminykh, Bogdan Ghita, and Jose M Such. 2021. The challenges with Internet of Things security for business. In *International Conference on Next Generation Wired/Wireless Networking*. Springer, 46–58.
- [30] Thomas Langerak, Juan José Zárate, David Lindlbauer, Christian Holz, and Otmar Hilliges. 2020. Omni: Volumetric sensing and actuation of passive magnetic tools for dynamic haptic feedback. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 594–606.
- [31] Thomas Langerak, Juan José Zárate, Velko Vechev, David Lindlbauer, Daniele Panozzo, and Otmar Hilliges. 2020. Optimal control for electromagnetic haptic guidance systems. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 951–965.
- [32] Gierad Laput, Chouchang Yang, Robert Xiao, Alanson Sample, and Chris Harrison. 2015. Em-sense: Touch recognition of uninstrumented, electrical and electromechanical objects. In *Proceedings of the 28th annual ACM symposium on user interface software & technology*. 157–166.
- [33] Ziwei Liu, Feng Lin, Chao Wang, Yijie Shen, Zhongjie Ba, Li Lu, Wenyao Xu, and Kui Ren. 2023. CamRadar: Hidden Camera Detection Leveraging Amplitude-modulated Sensor Images Embedded in Electromagnetic Emanations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–25.
- [34] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfiloIoT: A machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*. 506–509.
- [35] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2177–2184.
- [36] Julie L Newcomb, Satish Chandra, Jean-Baptiste Jeannin, Cole Schlesinger, and Manu Sridharan. 2017. IOTA: a calculus for internet of things automation. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. 119–133.
- [37] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V Krishnamurthy, Edward JM Colbert, and Patrick McDaniel. 2018. IotSan: Fortifying the safety of IoT systems. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 191–203.
- [38] Donald A Norman. 1988. *The psychology of everyday things*. Basic books.
- [39] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: Behavior transparency and control for smart home IoT devices. In *Proceedings of the 12th conference on security and privacy in wireless and mobile networks*. 128–138.
- [40] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*. 267–279.
- [41] ABI Research. 2015. Nest Cam Works Around the Clock. <https://www.abiresearch.com/press/nest-cam-works-around-clock/>. (Accessed on 03/09/2023).
- [42] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. 2020. A haystack full of needles: Scalable detection of iot devices in the wild. In *Proceedings of the ACM Internet*

- Measurement Conference*. 87–100.
- [43] Nader Sehatbakhsh, Alireza Nazari, Haider Khan, Alenka Zajic, and Milos Prvulovic. 2019. Emma: Hardware/software attestation framework for embedded systems using electromagnetic signals. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 983–995.
  - [44] Danny Sheridan. 2023. March 14: Mental Models with an example - by Danny Sheridan. <https://www фактотднега1.com/p/march-14-mental-models-with-an-example>. (Accessed on 11/15/2023).
  - [45] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 803–812.
  - [46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
  - [47] Yinxin Wan, Kuai Xu, Feng Wang, and Guoliang Xue. 2021. IoT Athena: Unveiling IoT device activities from network traffic. *IEEE Transactions on Wireless Communications* 21, 1 (2021), 651–664.
  - [48] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. 2016. How to use t-SNE effectively. *Distill* 1, 10 (2016), e2.
  - [49] Mengmei Ye, Jonathan Sherman, Witawas Srisa-An, and Sheng Wei. 2018. TZSlicer: Security-aware dynamic program slicing for hardware isolation. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 17–24.
  - [50] Thomas Zachariah and Prabal Dutta. 2019. Browsing the web of things in mobile augmented reality. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 129–134.
  - [51] Lefan Zhang, Cyrus Zhou, Michael L Littman, Blase Ur, and Shan Lu. 2023. Helping Users Debug Trigger-Action Programs. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–32.
  - [52] Li Zhuang, Feng Zhou, and J Doug Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)* 13, 1 (2009), 1–26.

## A IOTPROSECTOR'S GUI TOOL

### A.1 Code for IoTProsector

We plan to release the code of IoTProsector's design and make it publicly available.

### A.2 Instruction Table for Google Home Smart Speaker

As part of IoTProsector's GUI, we show the instruction table of the Google Home smart speaker to instruct the users to interact with the Google Home smart speaker as shown in Fig. 29.

## B INTERVIEW QUESTIONS

### B.1 General Background Questions

- First name [Study use only]
- Gender
- Education level
- Major

### B.2 Prior Experience of Using IoT Devices

- Do you have experience in working with IoT devices? If "yes", what is it about?
- Do you use any IoT device in your daily life?
- Have you used Google Home smart speaker before? If "no", have you used any other smart speakers before?

### B.3 Open-Ended Feedback Questions

- How do you annotate states during exploring? What criteria do you use?
- What criteria do you use when you collage states?
- Do you find the sensing model useful during this process? Why or why not?
  - When do you refer to the sensing model?
  - What type of information does the sensing model provide to you?

Function	Interaction	Image
Turn off	Plug power cable	
Play or pause	Tap	
Start request	Say keyword	
Do request	Say command	
Turn down volume	Swipe counter-clockwise	
Turn up volume	Swipe clockwise	
Mute or unmute	Press mic button	

Fig. 29. The instruction table shows the users how to interact with the Google Home smart speaker.

- Do you find the mental model useful during this process? Why or why not?
  - When do you refer to the mental model?
  - Do you refer to the context information in the UI? If so, when?
  - What type of information does the mental model provide to you?
- Have you encountered any conflict between your mental model and the sensing model?
  - What is the conflict? When does it occur?
  - How do you deal with this conflict?
- What do you think of IoTProsector?
  - Do you find it easy to use? Why?
  - What do you think of this workflow?
  - Do you think IoTProsector helps you understand the IoT device? If so, how?
  - What do you think can be improved in IoTProsector?