

Can Metadata Reveal User Activities?

A Privacy Study of Amazon Ring Cameras

Matias Tacchetti
matias@ucf.edu

University of Central Florida
Orlando, Florida, USA

Juan Gomez
ju323496@ucf.edu
University of Central Florida
Orlando, Florida, USA

Maria Newkirk
ma487111@ucf.edu

University of Central Florida
Orlando, Florida, USA

Alexander Flohr
al520340@ucf.edu
University of Central Florida
Orlando, Florida, USA

1 Abstract

Although smart-home camera's communication is typically secured by using encryption, they may still expose private behavioral data in their network metadata. This paper tests whether encrypted Ring camera traffic can disclose confidential user activities in the form of visible metadata. We purpose ringCap a python based data collection and classification framework. Through a Raspberry-Pi that has been set up as a standalone Wi-Fi access point, the team automatically tracks and labels traffic traces for three functional states: idle, motion detection, and live view. The packet metadata is extracted using tshark, stored in MongoDB, and later analyzed using Python. A scikit-learn Random Forest classifier is trained to infer the state of a device through features like the packet rates, the size of the packet, the protocol mix, and the uplink/downlink ratios. The model is able to perform extremely well in classification (89% accuracy); it can distinguish devices' state of being even when payloads remain encrypted, as metadata in itself can be used to distinguish the different states. These results underscore a significant privacy risk in consumer IoT ecosystems and urge solutions, including traffic padding, timing obfuscation, and network segmentation. Linked: ringCap

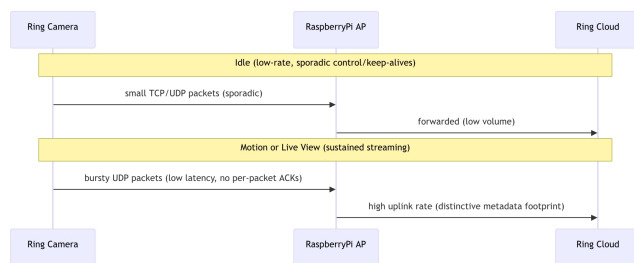


Figure 1: TCP/UDP Network Conversation

2 Introduction

Smart home solutions, most commonly used in residential settings, use cloud-connected cameras such as Amazon Ring to continuously monitor homes and provide remote access. These systems use end-to-end encryption to protect recorded and streamed videos. But even if the payload is secured by encryption, the metadata generated by the device's communications are not shielded from detection.

There is previous evidence that patterns in encrypted traffic, such as packet sizes, timing intervals, or protocol behaviors, have the potential to leak important contextual information despite the fact that the underlying content is hidden.

This work examines the question of whether a simple passive observer in the network can extract the operational state of a Ring camera only from encrypted network data. The team takes into account three user-dependent states: idle, motion detection, and live view. These states yield their own communication behaviors; notably, Ring devices use a large amount of UDP traffic for streaming, which gives rise to very characteristic burst patterns when the camera is active.

To avoid using manual labeling, the team relies on an automated pipeline built using Python to collect and label data then stores it in a mongoDB database. This labeled data is fed into a Random Forest classifier that is fit to classify the 3 states a ring camera can be in: idle, motion, and live-view states.

To illustrate the privacy constraints of encryption-only communications models, this work demonstrates that encrypted IoT devices can leak meaningful data using metadata; as a consequence, defenses are incentivized to mask visible traffic flow.

3 Problem Statement

Smart home cameras, such as those from Amazon Ring, are an increasingly common part of residential security systems that provide the ability to view your surroundings using real-time or cloud-based recorded video. These systems employ encrypted communications to keep user data private. However, the traffic metadata (e.g., packet sizes, time between packets, upload/download ratio and destination) may disclose sensitive context about the device's state and user behavior even when video payload is fully encrypted. There is also historical data, showing that Ring Doorbell devices up to version 3.4.7 misapplied encryption [9] (CVE-2019-9483) which opened the potential for video stream compromise or forgery.

In this project, the team explores whether encrypted Ring camera traffic reveals data that could be exploited by a passive eavesdropper to infer user behavior. The goal of the investigation is to find out whether differences between metadata streams can be exploited for the detection of states (e.g., idle, motion and live view) of devices. The work analyzes only information that is observable at the network layer, without assuming any access to video content or decrypting it.

As illustrated in Figure 1, at the protocol level, Ring camera traffic is largely UDP-based rather than TCP during streaming sessions. This is significant because UDP itself is a stateless protocol and does not require the ongoing acknowledgments and handshakes that are standard with TCP. This reduces the latency and lets the video respond a bit faster but generates some baseline packet patterns: quick streams of small packets can help determine when the camera is streaming or detecting movement. These features render the UDP metadata particularly useful in identifying camera states without revealing content, even with encryption playing a role.

Threat model: an attacker on the local network capable of sniffing Wi-Fi or LAN traffic (e.g., a malicious router or man-in-the-middle device) is assumed. The adversary is passive; it cannot forge traffic and only stores packet-level information. In this context, the team explores if metadata can be utilized to predict events based on the amount of data being transferred and correlate it to events recorded by the Ring infrastructure (when applicable).

This study is conducted in a lab environment, where a Ring Battery Doorbell Plus (one of the most popular models) is connected to a bridged Wi-Fi network using an intermediate device. Each test case including idle, motion event and live view is carefully analyzed using the Wireshark utility to review the network conversations in-depth.

The project will address a critical research question: “*Can metadata from encrypted Ring camera traffic be used to infer user activities, even without decrypting its content?*” Demonstrating this hypothesis would raise a privacy concern in the consumer IoT landscape. The purpose of this work is to understand what metadata features carry the most behavioral signals, assess the accuracy in state inference and discuss some mitigations such as packet padding, timing obfuscation or network segmentation. At a minimum, this research aims to inform end-users and IoT developers about the privacy limits that are exclusively dictated by encryption-driven design and to put forward guidelines to reduce metadata leakages.

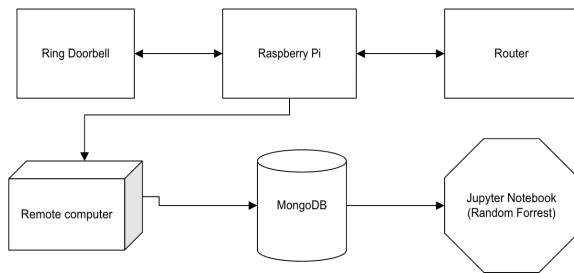


Figure 2: Data collection layout

4 Related Work

Previous work shows that encryption alone does not prevent attackers from inferring user information from IoT traffic. Apthorpe et al. demonstrated that smart home devices, including cameras, speakers, and sensors, emit distinctive metadata patterns even when all payloads are encrypted [1]. Their study showed that packet-size

distributions, timing intervals, and traffic bursts reveal when devices are idle, sensing events, or actively streaming, despite the use of TLS. These observations closely mirror our findings for Ring: the Ring doorbell exhibits clear shifts in traffic direction, volume, and protocol mix that correlate with the device state.

Mohit et al. conducted a traffic privacy study on smart-home IoT applications and evaluated several traffic-shaping defenses, including Independent Link Padding (ILP), Stochastic Traffic Padding (STP), and a dynamic traffic padding (DTP) scheme [7]. Although DTP was more effective than static padding, all padding strategies still leaked enough metadata for an observer to infer device states through packet-rate changes and burst structure. Their results confirm that even padded and encrypted traffic may retain highly identifiable metadata patterns.

Rahman et al. introduced UNSW HomeNet, a large-scale data set for AI-based device classification [10]. They collected flows from more than 100 IoT and non-IoT devices and showed that metadata features—packet counts, flow durations, and size statistics—are sufficient to classify device types with high accuracy. Although their goal was device identification rather than activity inference, their results highlight how stable and distinctive metadata patterns are across modern IoT ecosystems.

Beyond device identification, other research shows that IoT traffic can reveal user activities. Wan et al. proposed IoTMosaic, a system that infers user behavior in smart homes by detecting device events from traffic and matching ordered sequences of events to activity signatures [12]. Their approximate match algorithms tolerate missing or out-of-order packets and achieve near-perfect accuracy in 21 household activities. Although our project studies a single device rather than an entire smart-home ecosystem, both works rely on the same principle: encrypted traffic carries enough temporal and directional structure to uncover user behavior.

Complementary work on encrypted application traffic supports this view. Fathi-Kazerooni et al. showed that an external eavesdropper can identify which applications a user is running by analyzing metadata features such as packet counts, inter-arrival times, flow durations, and endpoint patterns [3]. Multiple classifiers—including Random Forests—achieved high precision and recall even when packet padding was applied. Since the Ring camera exposes only a small number of high-level states (idle, motion, live view), this prior work strongly suggests that state inference from Ring metadata should be both feasible and highly accurate, consistent with the patterns we observe in our captures.

5 Techniques

In order for us to construct a functional machine learning model, we need to collect data. We use a raspberry pi computer running Ubuntu as a malicious access point that our target ring doorbell will use for its internet connection. This allows us to run our data collection framework (ringCAP: <https://github.com/Neozesus81/ringCap>) on the access point and record all the information that is going to and from the doorbell. The block diagram is shown in Figure 2.

It is paramount that the raspberry pi can reach a different computer on the same network in order to send the collected data to be stored there. The compute resources of the pi are limited and using

it to collect and store the data in order to let it keep up with the internet traffic as it is snooping.

Shark.py does all of the heavy lifting when it comes to actually listening to the traffic. It uses a terminal based network protocol analyzer, tshark, [4] to see all the data being sent to and from the ring doorbell. Tshark allows us to specify the network interface we would like to monitor which for us is wlan0. It also allows for define either the length of time we want to capture or the number of packets. Our first approach was to use a time window of 20 seconds per data point of each state. This does not work because when the doorbell is in the idle state it will only send out a packet every 30 seconds. Instead, we decided to force each data point to capture 20 packets which insures we collect traffic for all states of the doorbell.

Mongodb.py defines the connection to the mongoDB [8] running on a computer that is connected to the same router as the raspberry pi. It allows the pre labeled data from tshark to be inserted into the database. We chose to use document based database instead of a relational one because of the nature of our collected data. Not all packets contained all of the same features so it would be difficult to predefine a schemea that would encompass all of the features. Because of this, we only need to specify the data captured in "pcap" and the label in "event_type" in order to insert a data point into the collection. This greatly reduced the cognitive overhead needed to start data collection but made cleaning the data more challenging than expected.

5.1 Data Labels

Our three classes are as follows:

- 1) **Idle:** The Ring camera stays powered and connected, but no motion is triggered, and the application remains closed.
- 2) **Motion Event:** Movement is introduced in front of the Ring camera at recorded times.
- 3) **Live View:** A live stream is initiated in the Ring application, with start and end times recorded.

Main.py combines both the "shark" class and mongoDB.py. First it opens a connection to the database using mongoDB.py; then it will spin off tshark to sniff the data needed and label it with the passed event_type argument. Finally, it will add it to the database. With this pipeline in place, we were able to call this main.py script in a loop while forcing the ring to be in a specific state to collect all of the data needed for this project. While the main.py script is running on the raspberry pi, the remote computer is running a local mongodb server ready to receive all of the data that is being collected.

After the data was collected, we had to clean it for machine learning use in a sklearn [6] random Forrest model. Using a Jupyter notebook, we were able to massage the raw json objects into a usable dataframe. First we had to flatten the extremely nested tshark json object into a single flat object. using a recursive parsing function. The importance of flattening the data is so that a model can easily ingest the data for training and inference. It would be impossible for an off the shelf library model to parse a deeply nested json object. Moreover, with the data being flat it is easier for us to select features that we think give the most information, primarily: ip addresses, port numbers, and time in between packets. Most of the features

Feature Name
frame_frame.time_delta
frame_frame.time_relative
frame_frame.len
frame_frame.cap_len
eth_eth.dst_tree_eth.dst.oui
eth_eth.src_tree_eth.src.oui
eth_eth.src_tree_eth.addr.oui
ip_ip.hdr_len
ip_ip.len
ip_ip.ttl
ip_ip.checksum.status
ip_ip.src
ip_ip.addr
ip_ip.src_host
ip_ip.host
ip_ip.dst
ip_ip.dst_host
tcp_tcp.srcport
tcp_tcp.dstport
tcp_tcp.port
tcp_tcp.len
tcp_tcp.seq_raw
tcp_tcp.hdr_len
tcp_tcp.window_size_value
tcp_tcp.analysis_tcp.analysis.bytes_in_flight
tcp_tcp.analysis_tcp.analysis.push_bytes_sent
tls_tls.record_tls.record.content_type
tls_tls.record_tls.record.length

Table 1: Features Selected for Training

collected could not be used in the model because they could not be encoded effectively. Obviously, the payload data was not usable since it is encrypted. Likewise, the ipv6 addresses are encoded in hex values which can not be casted into numerical values so we omitted them. We were able to encode the ipv4 addresses by removing '.'s from them before casting them to floats. After cleaning the dataset we were left with 27 usable features of the total 409. The other values that were collected were very sparse and did not have significant information gain.

Now we are ready to fit our model. We decided to choose random forest because of its easy of use and its comparable performance to more complex algorithms. XGBoost was another contender for our model architecture but we decided to no use it because of the structure of our data. We do not have that many features to deal with and from our initial results the first handful of features have the highest importance. So using XGBoost features like regularization seemed like overkill.

A Random Forest by nature of its design can have random results. Depending on what feature it selects as it starting point. To minimize the randomness we tested a fit and split 100 times and recorded top 5 features in order of importance. The collected data

was split 70% training data 30% test data. We then took the accuracy of each fit on the test data and found the model to be 89.74% accurate on test data.

6 Evaluation

As identified in our original milestone experiments, idle traffic produced sparse, low-rate packets with minimal variation. Motion traffic resulted in short bursts of mid-sized UDP packets. Live-view captures produced sustained, high-rate traffic with packet sizes frequently between 800 and 1,200 bytes. In our deeper final project analysis, a total of 5,320 packets were collected across three controlled scenarios. The dataset is as follows:

- 1,960 live view packets;
- 1,880 idle packets;
- 1,480 motion packets;

The protocols of the packets are as follows:

- TCP - 2,338 (1.345 MiB)
- UDP - 2,217 (0.985 MiB)
- ARP/ICMP - 765 (0.031 MiB)

The distributions indicate that both signaling and encrypted video transmission were captured in proportions consistent with Ring's operational behavior. Flow analysis revealed stable communication patterns. The Live View captures show sustained UDP flows to a small set of AWS cloud endpoints using high-numbered ports, which is consistent with how WebRTC-based TURN/STUN relays operate for encrypted video streaming. The largest single flow transferred 521,446 bytes from source port 56097 to destination port 40824. Additional high-volume flows exhibited similar directional characteristics. This stability in flow endpoints and port assignments implies that the Ring service does not significantly vary its network paths or port allocations across sessions, increasing the predictability of metadata patterns. This is typical of client-to-cloud IoT behavior and reinforces that the flows we observe are deterministic and easily profileable to malicious actors.

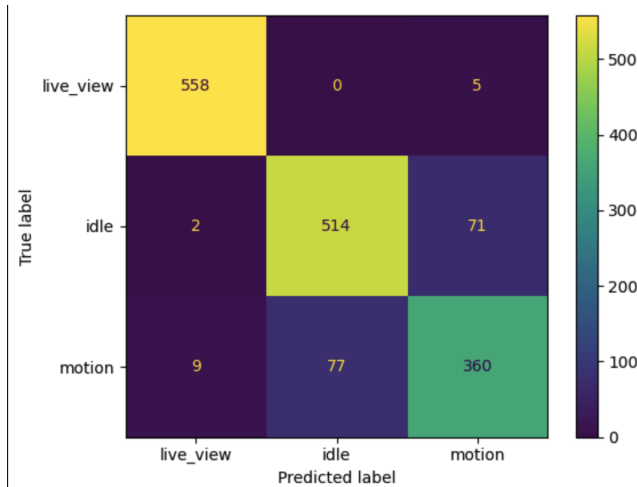


Figure 3: This shows that the motion is the worst performing prediction of the 3 classes.

We utilized a Random Forest classifier which was trained on flattened metadata fields using a seventy-percent training split and a thirty-percent testing split. The model achieved an average accuracy of 89% on the test set. The confusion matrix confirms (Figure 3) that the classifier distinguished live view and idle with near perfect precision and recall. The majority of misclassifications occurred between motion and idle which was expected because both states generate less traffic than the live view. The model correctly identified 2,808 of 2,808 live-view samples, 2,928 of 2,933 idle samples, and 2,204 of 2,239 motion samples.

Feature-importance rankings computed across 100 iterations of model fitting show that destination IP address was the highest-ranked feature. Additional IP addressing fields including ip.addr, dst_host, and ip.host consistently appeared in the top positions as shown in figure 4.

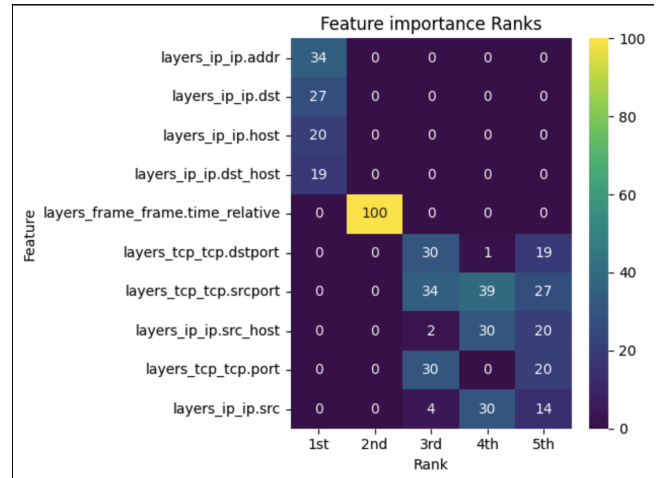


Figure 4: This shows the top 5 features in importance over the 100 fits of the data. The most important feature of each fit was layers_ip_ip.dst 30 out of 100 times. Time relative was always the second most important and the last 5 shared 3rd through 5th.

7 Discussion

In this project we try to determine if the metadata from a ring doorbell is enough information to know what state the camera is in. We found that with some automatic data collection techniques and off the shelf machine learning models one can extrapolate what the camera is currently doing with a significant degree of confidence.

One thing we found surprising is the feature importance of the source and destination ports. They are both around 3-4th in order of importance. After further research Ring has pre defined ports for all of the protocols used on their website. We did not know this while we were creating the model so we did not take that into account during data cleaning and feature selection. Having this knowledge now it would be much easier to track TCP traffic on the target network by limiting it to the ports on Ring's website.

While our data collection set up makes big concessions mainly that someone would connect their ring doorbell to network that they are not familiar with, Being able to classify traffic with a relatively basic machine learning model with 89% accuracy is alarming.

information should only be available to users of the device and not the open internet.

Even still, through this research we hope that users are more aware of the security implications that introducing IoT devices to their home network. Sometimes the modern convince of having a camera on your doorstep is not worth the potential data leakage that come along with it. This goes for all IoT devices as well. The responsibility does not fall exclusively on the consumer to not use these devices either. The companies making these types of projects should also understand the security implications of the devices they make. Having information open to anyone that could increase the opportunities for attackers to infiltrate. Especially if a user of your device needs to expose the ports you list to the open internet in order for you device to work.

8 Conclusion

In conclusion, this project shows that metadata from encrypted Ring camera traffic is enough to reveal device state and related user activities. By using a Raspberry Pi access point with automated capture and labeling, and a Random Forest classifier, we were able to infer idle, motion, and live-view states with about 89.7% accuracy. We achieved this using only packet-level features (metadata).

To summarize, the three states produced different signatures in upload volume, protocol mix, packet size, and packet rate. Idle traffic remained sparse and balanced, while motion events triggered brief but intense upload bursts. Meanwhile, live view generated sustained high-rate UDP streams. These patterns not only separated states in our evaluation but also showed how a passive observer could map network traces to specific activities inside a home.

Our dataset is modest and focuses on a single Ring doorbell in a lab setting. However, these findings still point to wider privacy concerns for consumer IoT camera systems. From metadata alone, an attacker close to the network could infer routines, package arrival times, and periods of absence. For this reason, vendors and users should treat traffic metadata as sensitive data. Additionally, future work should test practical defenses, including padding, timing noise, and network isolation. Finally, protecting smart home privacy requires not only strong encryption but also designs that limit what metadata reveals about life inside a home.

9 Work Distribution

The team members and their expected contributions are:

Matias Tacchetti (Lead):

- Coordinated group communication and assist in organizing project tasks during all the projects. [about 6 hours]
- Handled LaTeX formatting and final submission of all project assignments. [about 5 hours]
- Contributed to researching and collecting academic references for the project. [about 3 hours]
- Performed network traffic capture on Ring camera/doorbell using Wireshark. [about 6 hours]
- Analyzed his captured traffic and summarized results to support other sections of the project. [about 2 hours]
- Wrote the **Techniques** and **Work Distribution** sections of the *Project Proposal*. [about 6 hours]

- Rewrote the **Techniques** section for the *Project Milestone* completely due to the change in project goals, and refined the **Work Distribution** section. [about 5 hours]
- Wrote the **Techniques** and **Conclusion** sections of the final *Project Report*. [about 3 hours]

Maria Newkirk:

- Contributed to researching and collecting academic references for the project. [about 3 hour]
- Performed network traffic capture on Ring camera/doorbell using Wireshark. [about 2 hours]
- Analyzed the captured traffic and reviewed all team captures before the Initial Findings write-up. [about 1 hour]
- Wrote the **Expected Outcomes** section of the *Project Proposal*. [about 1 hour]
- Wrote the **Initial Results** section of the *Project Milestone*. [about 2 hours]
- Wrote the **Related Work** and **Evaluation** sections of the final *Project Report*. [about 14 hours]

Juan Gomez:

- Contributed to researching and collecting academic references for the project. [about 3 hours]
- Performed network traffic capture on Ring camera/doorbell using Wireshark and documented the results. [about 5 hours]
- Wrote the **Problem Statement** for the *Project Milestone* and developed the network flow diagram to show Ring camera communication patterns in idle and active states. [about 2 hours]
- Rewrote the **Problem Statement** section for the *Project Milestone* completely due to the change in project goals. [about 2 hours]
- Wrote the **Abstract**, **Introduction**, and **Problem Statement** sections of the final *Project Report*. [about 3 hours]
- Collaborated with data analysis for the **Evaluation** for the *Project Report*. [about 4 hours]

Alexander Flohr:

- Contributed to researching and collecting academic references for the project. [about 3 hours]
- Created the ringCap repository for the data collection framework. [about 8 hours]
- Performed network traffic capture on Ring camera/doorbell ringCap. [about 8 hours]
- Wrote the jupyter notebook to clean, train, and evaluate our machine learning model. [about 20 hours]
- Created graphics showing the performance of our model. [about 4 hours]
- Reviewed the *Project Milestone* drafts before submission and recommend revisions for grammar and overall coherence. [about 1 hour]
- Wrote the **Initial Discussion** section of the *Project Milestone*. [about 4 hours]
- Wrote **Techniques** and **Discussion** for the final paper. [about 10 hours]
- Proof read final paper before submission. [about 2 hours]

References

- [1] Noah Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. In *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, USA. https://www.ndss-symposium.org/wp-content/uploads/2017/09/NDSS-2017_Paper_Apthorpe.pdf
- [2] Dan Calacci, Jinjun J. Shen, and Alex Pentland. 2022. The Cop in Your Neighbor’s Doorbell: Amazon Ring and the Spread of Participatory Mass Surveillance. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2, Article 400 (2022). doi:10.1145/3555125
- [3] S. Fathi-Kazerooni, Y. Kaymak, and R. Rojas-Cessa. 2019. Identification of User Application by an External Eavesdropper using Machine Learning Analysis on Network Traffic. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. 1–6. doi:10.1109/ICCW.2019.8756709
- [4] Wireshark Foundation. 2025. *tshark: Terminal-based Wireshark*. <https://www.wireshark.org/docs/man-pages/tshark.html>
- [5] Monica Kowalczyk, Johanna Gunawan, David Choffnes, Daniel J. Dubois, Woodrow Hartzog, and Christo Wilson. 2023. Understanding Dark Patterns in Home IoT Devices. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI ’23)* (Hamburg, Germany). Association for Computing Machinery, 1–27. doi:10.1145/3544548.3581432
- [6] Scikit learn Developers. 2025. *sklearn.tree.DecisionTreeClassifier Documentation*. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>
- [7] Mohit, S. Ansari, and A. Kumar. 2021. Traffic Privacy Study on Internet of Things – Smart Home Applications. In *2021 9th International Conference on Cyber and IT Service Management (CITSM)*. 1–6. doi:10.1109/CITSM52892.2021.9588855
- [8] Inc. MongoDB. 2025. *MongoDB Official Website*. <https://www.mongodb.com/>
- [9] National Institute of Standards and Technology. 2019. CVE-2019-9483: Amazon Ring Doorbell Improper Encryption Handling. National Vulnerability Database. <https://nvd.nist.gov/vuln/detail/CVE-2019-9483>
- [10] M. M. Rahman, F. Bouhafs, S. A. Hoseini, and F. D. Hartog. 2025. UNSW HomeNet: A network traffic flow dataset for AI-based smart home device classification. *Computers & Industrial Engineering* 204 (2025), 111041. doi:10.1016/j.cie.2025.111041
- [11] Eimaan Saqib, Shijing He, Junghyun Choy, Ruba Abu-Salma, Jose Such, Julia Bernd, and Mobin Javed. 2025. Bystander Privacy in Smart Homes: A Systematic Review of Concerns and Solutions. *ACM Transactions on Computer-Human Interaction* (2025). doi:10.1145/3731755
- [12] Yinxin Wan, Kuai Xu, Feng Wang, and Guoliang Xue. 2022. IoTMosaic: Inferring User Activities from IoT Network Traffic in Smart Homes. In *IEEE INFOCOM 2022*. 370–379. doi:10.1109/INFOCOM48880.2022.9796908