A Trigger for the Autonomous Decommissioning of Smart Devices

Ravindra Mangar
Dartmouth College
Hanover, New Hampshire, USA
ravi.gr@dartmouth.edu

Jared Chandler
Dartmouth College
Hanover, New Hampshire, USA
jared.d.chandler@dartmouth.edu

Jingyu Qian
University of Illinois
Urbana-Champaign, Illinois, USA
jingyuq2@illinois.edu

Carl A. Gunter
University of Illinois
Urbana-Champaign, Illinois, USA
cgunter@illinois.edu

Timothy J. Pierson
Dartmouth College
Hanover, New Hampshire, USA
timothy.j.pierson@dartmouth.edu

David Kotz
Dartmouth College
Hanover, New Hampshire, USA
david.f.kotz@dartmouth.edu

Abstract

Smart devices are ubiquitous in modern environments, yet their decommissioning phase remains poorly studied and often overlooked in system design. We define *secure decommissioning* as the process by which a smart device securely disconnects from its environment and makes sensitive data inaccessible. If not decommissioned, devices may retain sensitive information – such as security credentials or user-behavior data that could be recovered by an adversary. Unfortunately, some users may forget to decommission a device when they dispose or sell it, and cannot decommission a device that is lost or stolen. This paper investigates a trigger mechanism for individual wireless smart devices to automatically identify conditions requiring decommissioning. Our approach does not require any hardware changes to wireless devices. We evaluated it through extensive simulations and validated it on real IoT-class hardware. With appropriate parameter values, our mechanism always correctly identified when to decommission and never falsely decommissioned. These parameters can be tuned to user needs. Our work presents a baseline for locally triggered autonomous decommissioning, providing researchers with a useful starting point for exploring and improving alternative designs.

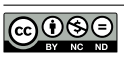
CCS Concepts

• **Hardware → Wireless devices**; • **Networks → Network experimentation**; • **Security and privacy → Mobile and wireless security**.

Keywords

Wi-Fi, Smart Homes, IoT, Security and Privacy

ACM Reference Format:
Ravindra Mangar, Jared Chandler, Jingyu Qian, Carl A. Gunter, Timothy J. Pierson, and David Kotz. 2025. A Trigger for the Autonomous Decommissioning of Smart Devices. In *The 15th International Conference on the Internet of Things (IOT 2025), November 18–21, 2025, Vienna, Austria.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3770501.3770522



IOT 2025, Vienna, Austria

ACM ISBN 979-8-4007-1595-2/25/11
https://doi.org/10.1145/3770501.3770522

1 Introduction

The lifecycle of Internet of Things (IoT) devices encompasses several critical phases: design, deployment, operation, and decommissioning. While significant attention has been paid to the initial stages, the decommissioning phase remains under-explored, posing substantial security and privacy risks [15]. As IoT devices become increasingly integrated into our daily lives, they collect and store vast amounts of sensitive data, including personal information and usage patterns, as well as network and cloud credentials. When these devices reach the end of their useful life or are transferred to new owners – such as when sold or discarded – the security and privacy implications become significant if devices are not cleansed of this sensitive information.

The problem. A discarded security camera, for example, might use non-volatile memory to store the passwords to both a home's Wi-Fi network and to a cloud-based server that stores the images captured by the camera. If an adversary recovers these credentials from a discarded device, the adversary would gain access to the home and the cloud.

Other sensitive information, beyond security credentials, could also be compromised. A health-related device might indicate a person's medical condition; a sex toy may expose their romantic partners or sexual preferences [27]; an entertainment device might contain their viewing or listening history [11]. Baggili et al. successfully recovered personal data, including user credentials and personal activity data, from second-hand smartwatches [4]. Similarly, Giese and Noubir recovered network credentials and sensitive user interactions from discarded Amazon Echo devices [9]. Such information could allow an adversary to learn a lot about the person and their habits [19].

We define *sensitive information* to include data that might compromise a device owner's security or privacy. It is necessary, then, to *securely decommission* any device that has been discarded, lost, stolen, gifted, or sold, to ensure that no sensitive information can be recovered by an adversary – even if they have physical access to the device.

In this paper, we assume the existence of a built-in API that can make sensitive information stored on a device unrecoverable; we thus focus on *when* to decommission rather than *how* to decommission. Given the prevalence of cloud-backup features in smart-home platforms [23], we also assume each device's critical data is already protected by a secure, incremental backup mechanism that can

be used to restore data in the event of device loss or erroneous decommissioning.

Relying on manual resets to remove sensitive information has been widely shown to be inadequate due to user negligence or lack of awareness of secure device disposal [17]. Indeed, many users feel their data is of no interest to potential adversaries and do not feel the need to decommission devices [14]. Regardless, if a device is stolen, the owner will not have an opportunity to cleanse the device. Thus, even when secure reset mechanisms exist, an automatic trigger is often missing. We supply that missing piece: an algorithm that decides when to securely decommission, ensuring that –even when data is encrypted at rest –the corresponding decryption keys are irrecoverably destroyed, preventing recovery by adversaries with physical access or UI bypasses.

An automated solution. Our solution provides a mechanism for a device to *autonomously* detect conditions that require decommissioning and to *automatically* initiate that process. In short, devices running our protocol will discover when they are no longer 'at home', and trigger an internal function to erase any sensitive data or credentials.

This paper makes two main **contributions**:

- A method for wireless devices to determine when to self-initiate the decommissioning action;
- An assessment of our approach using simulation and an implementation on IoT-class hardware (ESP32), demonstrating its practicality and efficiency.

We ran extensive simulations to find a plausible set of parameters for our autonomous decommissioning trigger, balancing the need for the system to accurately determine when to decommission, without unnecessarily wiping devices. The simulations suggested 8 candidate sets of parameters. In our experiments on real devices, 7 of the 8 parameter sets correctly decommissioned devices every time they should, with no erroneous decommissioning. These 7 parameter sets give users the flexibility to adjust the trigger sensitivity and reactivity to meet their particular needs.

Our solution does not require any hardware modification to wireless devices and adds limited network traffic. We focus on Wi-Fi devices, although our techniques could be expanded to work with any device with a wireless interface.

2 Background

Our primary focus in this paper is a smart home containing multiple wireless devices. We anticipate many devices in this home, such as a smart refrigerator or smart door lock, will be fixed in place and non-mobile. Other devices may be mobile, but are expected to always be inside the home. For example, a robot vacuum cleaner may move within the home, but is not expected to leave the home's boundaries while commissioned. Still other devices, such as smart phones and recreational equipment, may be mobile and will occasionally leave the home. Sometimes mobile devices may leave for an extended period of time, such as when the home resident goes on vacation and takes their smart toothbrush with them. Our method is designed for devices that ordinarily stay in the home. Because legitimate extended absences (e.g., loaning a device or taking it on a trip) can occur, we treat such departures as out of scope rather than assuming they signal a need for decommissioning.

The core idea behind our approach is that a device should continue to operate normally as long as it can (i) maintain a Wi-Fi association with its home access point or (ii) detect the familiar constellation of other devices normally present in the household. When both cues disappear for an extended interval – no link to the home AP and scant presence of familiar household devices – the device concludes it has left its trusted setting and triggers the decommissioning routine. Decommissioning erases local secrets, preventing unauthorized data recovery after loss, theft, or disposal.

3 Security Model

We consider an adversary who has obtained a household IoT device and whose primary goal is to extract sensitive data from the device. Consequently, the system aims to determine if the conditions for decommissioning have been met in a timely manner, then to initiate the decommissioning process, making data unavailable to the adversary. We consider two critical properties of our approach: *correctness*, ensuring the autonomous decommissioning trigger activates when (and only when) the device permanently leaves its trusted environment, and *timeliness*, ensuring prompt activation of the trigger to minimize the risk of data compromise.

We assume the adversary has physical access to the target device – that is, the device from which the adversary wants to extract sensitive information – but not the AP. (The owner may have gifted or sold the device to the adversary; or lost the device and the adversary found it; or disposed it and the adversary recovered it from the trash or recycling stream; or, the adversary stole the device from the owner; regardless, the adversary now has physical control over the device.) We thus assume the adversary may extract the target's memory contents, reverse-engineer its firmware, and inspect its internal state. If the device has not been decommissioned, sensitive data may remain available to such an adversary.

To prevent our protocol from succeeding, the adversary may attempt network-based manipulations, such as Sybil attacks [5], replay attacks, message spoofing, or synchronization interference, to mislead the device's internal-decision making process.

We assume devices are initially provisioned to a trusted AP, which the adversary cannot compromise, impersonate, or extract keys from. Devices trust the AP as a certificate authority that securely distributes cryptographic keys and scheduling information.

All parties trust that standard cryptographic protocols and primitives (e.g., WPA3 and cryptographic signatures) are secure.

4 System Design

To protect sensitive information, in our approach, household devices will self-decommission if they decide they are no longer at home. How do they make this determination? During normal operation, each device maintains a mutually authenticated connection to the home's AP (in our prototype we use WPA3); to a device, this ongoing authenticated association is confirmation it remains within its home environment. When a device loses contact with its home AP, however, it needs a means to decide, for itself, whether it has left home; after all, it is possible the device is still at home but the AP is down due to a power failure in the home. When the AP is out of contact, a device decides it is still 'at home' as long as it can still hear Wi-Fi signals from a sufficient number of other

household devices. If not, it concludes it has left home, and triggers decommissioning.

We adapt a change-detection method in which the goal is to monitor a time-series signal and detect sudden changes in the value of the signal, while ignoring noise and other short-term variations. Specifically, each device tracks the number of familiar Wi-Fi devices – those it has previously observed and authenticated – over both short and long time intervals, and computes the ratio of the Short-Term Average to the Long-Term Average (STA/LTA). Originally developed for signal processing in seismic event detection [25], STA/LTA compares short-term variations to a longer-term baseline to identify sudden changes in signal behavior.

In addition to facilitating secure connections, the AP coordinates periodic communications among participating devices by distributing a common broadcast schedule, assigning each device a unique, periodic time slot within the schedule.

Each device transmits encrypted broadcasts during its assigned time slot to announce its continued presence to neighboring devices.

Other devices decode the encrypted broadcasts and update their neighbor count to calculate a stable baseline – the Long-Term Average (LTA) – reflecting the number of neighboring devices within radio range under normal conditions. (Mobile devices set a flag in their broadcasts to indicate their intermittent presence in the home; we explore the implications of mobile devices on the decommissioning process in later sections.)

The AP organizes the communication schedule into *rounds*. During each round, all devices participating in this protocol transmit, one at a time, in an AP-assigned time slot, similar to Time Division Multiple Access (TDMA). For a home with N devices and slot duration S seconds, the cumulative time all devices spend broadcasting in a round is $T_r = N \times S$ seconds. To reduce energy use and channel load, we insert a quiet interval after each round. Thus, the elapsed duration of a round is $T_{rs} = N \times S + I_s$ in STA mode and $T_{rl} = N \times S + I_l$ in LTA mode. Typically, the interval duration $I \gg N \times S$.

4.1 Overview of the Algorithms

Each IoT device runs our method independently, with support from the AP. At initialization, each device associates with the AP via mutual authentication, then receives a schedule and authentication keys so it can coordinate with other devices. After initialization, it operates in two alternating phases; see Figure 1 for an overview.

4.1.1 Algorithm LTA – normal operation. During normal operation, the device is at home, connected to the home AP, and runs the LTA block of Algorithm 1 to maintain a running long-term average number of neighbor devices, LTA. In each *round*, each of N devices broadcasts F frames in its assigned slot and listens during the other $N-1$ slots. It counts the number of unique neighbors detected during that round (ignoring mobile devices), and appends that number to the sliding buffer. Because the buffer has capacity R_l, the oldest value (from R_l rounds earlier) is discarded. LTA is then computed as the average of the R_l numbers in that buffer and represents the "usual" number of neighbors. After each round, devices are idle for I_l seconds, allowing them to sleep and save energy.

Algorithm 1: STA & LTA Operations

```
Initialise sliding buffer B, set LTA ← 0, C ← 0
C: consecutive STA/LTA < Θ
while true do
    if connected to the AP then
        // LTA phase
        E ← ∅
        for slot i = 1 to N do
            if I am device i then
                ts ← current timestamp
                Broadcast ⟨i, ts, sig_i, cert_i⟩, repeatedly F
                  times
            else
                For each valid frame received
                from non-mobile device: verify cert, add ID
                  to E
        Append |E| to B
        if |B| ≥ R_l then LTA ← avg(B)
        Sleep for I_l
    else
        // STA phase
        for slot i = 1 to N do
            if I am device i then
                ts ← current timestamp
                Broadcast ⟨i, ts, sig_i, cert_i⟩, repeatedly F
                  times
            else
                For each valid frame received:
                verify cert, add ID to E
        Append |E| to B
        if |B| ≥ R_s then
            STA ← avg(B)
            C ← (LTA > 0 ∧ STA/LTA < Θ) ? C + 1 : 0
            if C ≥ R_d then Trigger decommission; exit
        Sleep for I_s
```

4.1.2 Algorithm STA – disconnected operation. If a device becomes disconnected from the AP, perhaps because it has roamed out of range, or because the AP has gone down, it freezes the LTA and begins computing the short-term average (STA). Algorithm 1 uses a sliding buffer that stores the number of unique devices detected in each of the R_s recent rounds. This operation is much the same as when computing LTA, except that authenticated mobile devices *are* counted. After each round, all devices are idle for I_s seconds, allowing them to sleep and save energy. After each round, Algorithm 1 computes the STA/LTA ratio to determine whether the ratio has dropped below Θ; if so, and if that condition persists for R_d consecutive rounds, the device decommissions. This persistence check avoids false triggers due to temporary network fluctuations.

We summarize the algorithm's parameters in Table 1. Clearly, the system's sensitivity to change depends on several of these parameters. For example, with threshold $\Theta = 0.5$, losing three peers

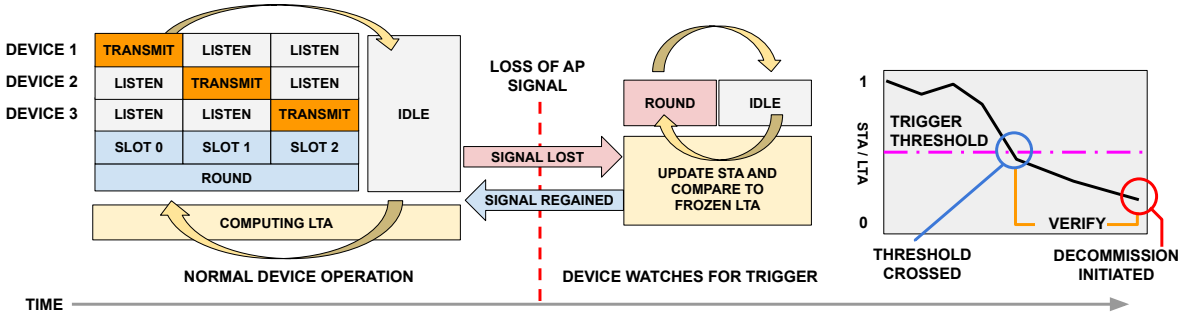


Figure 1: Illustration of device operation.

in a 5-node deployment pushes the STA/LTA ratio to 0.40 and triggers the algorithm, whereas the same loss in a 20-node deployment leaves the ratio at 0.85 and no trigger fires. Likewise, lengthening the short-term window R_s or requiring more consecutive confirmation rounds R_d smooths out brief packet losses – reducing false positives but delaying activation of the trigger. Because different scenarios may have different requirements, or different users may have different preferences, we envision a practical system would include a UI to configure these parameters – for example, a slider from 'less sensitive' to 'more sensitive' that would translate internally into adjustments to these parameters. Indeed, each device may be configured with separate Θ or R_d values to suit the device usage and its information sensitivity.

In practice, our technique must handle an important complication: mobile devices. Their absence or intermittent presence introduces variability in neighbor counts, requiring either specialized provisions or adjusted threshold and other parameters to avoid unwarranted decommissioning triggers. Our system does not count authenticated mobile devices (such as a recreational device that may often leave the home for outdoor recreation) in the LTA, but does count them in the STA. With this approach, the presence of a known mobile device can nudge the STA/LTA ratio upwards and potentially prevent an incorrect decommissioning. In fact, if mobile devices are present during STA/LTA calculation, the STA/LTA ratio can potentially go above 1.0 because the mobile device will be counted in the STA, but not the LTA.

4.2 Access Point Coordination

The AP maintains a registry of connected devices, tracking their MAC addresses and computing scheduling information. The AP also acts as a certificate authority (CA) to sign device's public keys, enabling devices to authenticate other devices' broadcast messages. By tracking only authenticated broadcasts, devices can reliably count the number of other legitimate devices within radio range and can avoid adversaries attempting to spoof the presence of neighbors.

When it first authenticates and associates with the AP, a new device uploads its public key P_i to the AP. The AP signs P_i with its private CA key, producing the certificate $\text{cert}_i = \langle P_i,\ \text{sig}_{\text{CA}} \rangle$, and returns cert_i to the device along with the CA's public key.

Devices periodically request the schedule from the AP via a secure TLS [20] connection. The AP releases a schedule only once a minimum of n devices are connected, so that a proper number of devices actively participate in the protocol. If too few (or an unrepresentative subset of) devices participate, a device may be falsely labeled as isolated, leading to an erroneous decommissioning.

The schedule defines when each device should broadcast or listen. The schedule parameters include the values shown in Table 1.

Once provisioned by the AP and ready to broadcast, the device derives its identifier $ID_i = \text{SHA-256}(P_i)[0{:}7]$, that is, the first eight bytes of a hash of the public key. During its designated timeslot, the device broadcasts a frame that includes M, a message containing a timestamp of the current time and a flag to denote whether the sender is a mobile device. The frame contains $ID_i \| \text{cert}_i \| \text{sig} \| M$, where $\text{sig} = \text{ECDSA}_{S_i}(M)$. The device sends F copies of this frame, within its slot, to increase the chance it will be received.

Upon reception of such a frame, the receiving node verifies the CA signature in cert_i with the CA public key P_{CA}, extracts P_i from cert_i, confirms that the advertised identifier matches $ID_i = \text{SHA-256}(P_i)[0{:}7]$, verifies the timestamp is recent, and verifies sig over the message M. The receiving node checks the freshness of a broadcast by comparing the timestamp embedded in the decrypted frame against its own locally recorded receive time. If the difference falls within an acceptable threshold, the frame is treated as recent; otherwise, it may be considered stale or replayed. These checks defeat attackers without the CA's private key who create bogus devices, defeat attackers who spoof a valid device, and defeat attackers who replay older broadcasts.

5 Implementation and Evaluation

To evaluate the proposed method, we implemented the system both in a simulated environment and in firmware on computationally constrained IoT boards. The simulation provides a controlled environment to test various network configurations and scenarios, and the implementation validates the feasibility of the approach on resource-constrained IoT devices.

5.1 Evaluation

Our evaluation metrics address *correctness* and *timeliness*. We use three metrics of correctness, namely Balanced Accuracy, False Positive Rate and False Negative Rate. When computing these metrics,

Table 1: Parameters for STA/LTA Algorithm.

Symbol	Parameter	What it controls	Units
n	Min. devices to start	Do not start schedules until at least n devices are present	count
N	Participating devices	Total devices included in the schedule	count
S	Slot duration	On-air time per device in each round	seconds
R_s	STA window	Num. rounds used to compute the short-term average (STA)	rounds
R_l	LTA window	Num. rounds used to compute the long-term baseline (LTA)	rounds
R_d	Persistence	Num. consecutive STA/LTA violations required to trigger	rounds
I_s	STA inter-round interval	Idle/sleep between STA rounds (AP down)	seconds
I_l	LTA inter-round interval	Idle/sleep between LTA rounds (AP up)	seconds
Θ	Trigger threshold	Fire if STA/LTA $< \Theta$ for R_d consecutive rounds	ratio
F	Frames per slot	Redundant broadcast frames sent per device per slot	count
T_r	Cumulative broadcast time per round	Sum of all device slots in one round $T_r = N \times S$	seconds
T_{rs}	STA round duration	Elapsed time per STA round; *calculated as* $T_{rs} = N \times S + I_s$	seconds
T_{rl}	LTA round duration	Elapsed time per LTA round; *calculated as* $T_{rl} = N \times S + I_l$	seconds

each device's decision (to decommission, or not, by the end of the experiment) represents one positive outcome (the device decommissioned) or negative outcome (the device did not decommission). A true positive (TP) occurs when a device that was intended to decommission indeed chose to decommission, indicating the system correctly identified and triggered the decommissioning operation on that device. A true negative (TN) represents a device that was not intended to decommission and correctly remained operational, showing that the system avoided unnecessary decommissioning. Conversely, a false positive (FP) happens when a device that was not intended to decommission was incorrectly decommissioned, and a false negative (FN) occurs when a device that was supposed to decommission failed to do so. Devices that were intended to decommission were chosen at random before the experiment was carried out.

Clearly, our approach aims to minimize false positives and false negatives, each of which have different consequences. A false negative may leave a device's sensitive data in the hands of an unauthorized person. Therefore, we report both the False Positive Rate (FPR) calculated as FP / (FP + TN) and the False Negative Rate (FNR) calculated as FN / (FN + TP). In these equations, FP, TP, FN, TN represent the number of cases with each of those outcomes.

The third metric, Balanced Accuracy (BA), provides a single-number summary metric: $\frac{1}{2}\left(\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}\right)$.

We measure the duration between the departure of a device from its home environment and the decommissioning decision. The expected decommission-trigger delay is the product of the consecutive rounds below threshold and the duration of one round: $(R_s + R_d)(NS + I_s)$. Then we define *timeliness* as the difference between the *actual* time to decommissioning and the *expected* time to decommissioning.

5.2 Simulation

To assess the algorithm's performance, we developed a simulation using the `ns-3` network simulator [21]. The simulation models a network consisting of one AP and multiple Wi-Fi devices and leverages the `ns-3` Wi-Fi module to simulate IEEE 802.11 features, including promiscuous mode, wireless signal propagation models, and broadcast operations. The simulated devices implement the STA/LTA algorithm and transition through different operational modes as dictated by the AP's schedule.

Table 2: Evaluation parameter values.

	Parameter	Simulation
N	Number of Nodes	5, 10, 20, 50, 80
S	Slot duration	0.1, 0.3, 1.0 seconds
R_s	Rounds to build STA	1, 3, 5, 10
R_l	Rounds to build LTA	1, 3, 5, 20
R_d	Rounds to decision	1, 3, 5
I_s	STA Round Interval	1, 3, 5 seconds
I_l	LTA Round Interval	3, 5, 10, 30 seconds
Θ	STA/LTA Threshold	0.3, 0.5, 0.8
F	Frames per slot	As many as possible
N_m	Mobile Nodes	1, 2
J	Node to Decommission	Random
P	Packet Drop Rate	0.3, 0.5, 0.8

We used the values in Table 2 to find parameter combinations that should produce low FPR and FNR. In total we tried 1,976 different combinations of parameters. Based on those results, we selected 8 plausible sets of parameter values for use in our experiments on a real-world prototype.

5.3 Experiments

To validate the approach on real hardware, we implemented the system on 10 Espressif ESP32 development boards. These devices are equipped with Wi-Fi and support secure communication protocols, making them suitable for our use case. The Espressif boards are one of the leading IoT boards on the market [7] and are used in many commercial products such as the Sonoff Energy Monitor [26] and the SimpliSafe home monitoring system [16]. By demonstrating that our solution operates effectively on these resource-constrained devices, we provide strong evidence of its potential applicability in production devices.

In our implementation, each device follows the protocol outlined in Section 4, including the security protocols. After initialization, it cycles between broadcast mode, listen mode, and idle mode, according to the AP-assigned schedule.

We deployed our experimental prototype in a university computer science building, chosen for its high device density and complex signal environment. As a more demanding setting than a typical residential setting, it provides a conservative evaluation. Two boards were equipped with external antennas to introduce additional variation in wireless range and signal quality. We also varied the transmission power of the Wi-Fi interface to allow for disjoint groups of devices. We used a Linux computer equipped with a Qualcomm Atheros QCNFA765 Wi-Fi chipset and hostapd [12] as the AP. The ESPs were programmed using a real-time operating system [6].

We ran our experiments using parameter sets that achieved 100% accuracy in simulation.

To trigger the need for decommissioning, we moved a randomly chosen device out of Wi-Fi range and logged the time the device was moved.

After each run we checked whether each device decommissioned. We then replaced the selected device to its original location, reset all the devices, and began the next run. In total, we performed 24 runs: 21 *open-space* tests and 3 *isolated-set* trials.

A *isolated-set trial* divides the ten devices into two physically isolated groups – four devices in one room and six in another – placed beyond Wi-Fi range so that intra-group broadcasts cannot be heard.

For each run, we also needed to disable the access point and thus force the devices into disconnected operation. We ensured that the AP remained enabled for at least $3R_l$ rounds before shutting it off. For example, if the time for R_l rounds was 10 seconds, we disconnected the AP any time after 30 seconds. Similarly, when deciding when to move the decommissioning device, we waited at least $R_s + R_d + 1$ rounds before initiating its relocation. Similarly, the devices that were not meant to be decommissioned were also left on for at least enough time for the STA to be updated three times before concluding the experiment so as to give enough time to record a false decommissioning event if one was ever to occur. As such, experiments lasted between forty-five minutes to one hour.

6 Results

In 21 open-space tests (all devices mutually in range; $N \in \{5, 6, 10\}$), the system detected and decommissioned the single removed device in every run with zero false positives. (Here, larger N smooths counts but lengthens each round T_r=NS.)

In the three isolated-set trials, removal was still correctly identified in all runs, but we observed two false positives in one parameter set (one in the second trial, and another in the third). The results are summarized in Table 3.

Analysis of the situation that led to the devices falsely decommissioning saw a combination of a small slot duration S (in the aforementioned case 0.1 s) and a low number of frames sent per slot (F) during a time when the environmental Wi-Fi traffic was high. To verify that a low S and F were the cause of the false positive, we increased the values and re-ran the experiment.[1] Subsequently, there were no false positives – while still consuming only 0.5% of network bandwidth (vs. 0.1% originally). We define bandwidth consumption here as the fraction of total channel capacity occupied by messages transmitted by our protocol. Such results demonstrate that even under high-traffic conditions, where some transmissions may not be received, our protocol remains both reliable and lightweight.

As for timeliness, the experimental results show it is always positive – that is, devices decommission later than expected, never prematurely. The delay grows with STA and LTA duration: long windows (175 s, 300 s) add roughly 22–53 s, to the expected decommission time. While short windows ($\approx$ 26–30 s) stay within 7–15 s. Nonetheless, in our real-world experiments, devices typically decommissioned within a few seconds of their expected time, but always decommissioned within 1.5 minutes of their expected time, despite sizable network traffic from other devices operating in the building.

In terms of longevity, we carried out an experiment for 3 days in an apartment building and another for 3 days in a building on a university campus. Each experiment consisted of 10 devices, and the AP was turned off after 4 hours. Thereafter, the nodes relied on peer-to-peer communication to not falsely decommission.

Both experiments did not register any false negatives or false positives. Earlier, we had evaluated a preliminary version of our algorithm for 7 days and saw similar results. As an example, Figure 2 illustrates the progression of decommissioning for a run with 6 devices, 5 of which were fixed and 1 that was mobile. We can see how the values of the STA and LTA changed for devices as a node was being decommissioned as well as the influence mobile nodes had on the STA.

7 Related Work

Automated methods exist for identifying when a smart device has left a sanctioned environment, but such methods require additional sensors (such as GPS) and are generally used for smartphones [2, 10]. Such sensors do not exist in many IoT devices; our approach requires only a Wi-Fi interface and a firmware update.

Many authors explore using decentralized ledgers as a record of device ownership [1, 13, 18, 22]. These approaches focus primarily on ownership transfer of a functioning device as opposed to IoT end of life. While an immutable ledger could be used to signal to an IoT device that it should decommission, this channel is no different than existing remote-wipe approaches, where the device periodically polls for a signal and requires an Internet connection. Conceptually this is similar to the way Apple Find My [3], or enterprise mobile-device-management (MDM) suites, let a phone or laptop poll a cloud service and, if instructed by the registered owner, trigger a secure-wipe. All of these schemes share one critical assumption: at the moment of decommissioning the device is still able to contact (or to be contacted by) some cloud service. Our design targets the opposite case: we handle instances where a device might be off-grid or intentionally denied Internet access.

[1] To recreate the conditions that led to the false positive, we induced traffic into the channel by injecting 802.11 frames.

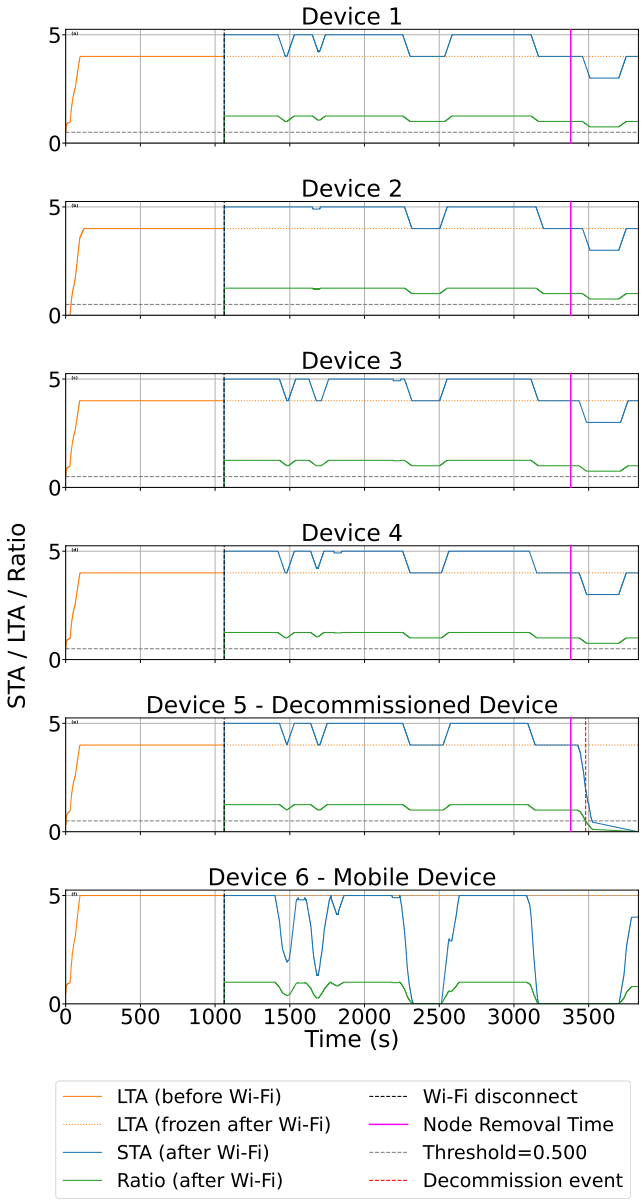


Figure 2: STA and LTA counts, and STA/LTA ratios, for devices belonging to a cluster of 5 devices with 1 mobile device. The Y-axis represents both the raw neighbor counts used to compute STA and LTA (typically integer values between 0 and the number of in-range neighbors) and the computed STA/LTA ratio (green line). Vertical lines represent the moment when the Wi-Fi AP became unavailable and the moment when Device 3 was moved out of range, leading it to decommission. A vertical red line indicates a decommission event. Device 6 was designated a 'mobile' device and was not meant to decommission.

Table 3: Results from hardware implementation *Timeliness* is the difference between actual and expected decommission time (in seconds).

N	R_s	R_l	T_{rs}	T_{rl}	Threshold	FN	FP	Bal. Acc.	FPR	Timeliness (Act. – Exp.)
						0	0	1.00	0.00	197 − 175 = 22
5	7	3	25	100	0.3	0	0	1.00	0.00	228 − 175 = 53
						0	0	1.00	0.00	201 − 175 = 26
						0	0	1.00	0.00	388 − 350 = 38
5	7	3	25	100	0.5	0	0	1.00	0.00	436 − 350 = 86
						0	0	1.00	0.00	375 − 350 = 25
						0	0	1.00	0.00	221 − 175 = 46
6	7	3	25	100	0.3	0	0	1.00	0.00	227 − 175 = 52
						0	0	1.00	0.00	220 − 175 = 45
						0	0	1.00	0.00	403 − 350 = 53
6	7	3	25	100	0.5	0	0	1.00	0.00	388 − 350 = 38
						0	0	1.00	0.00	380 − 350 = 30
						0	0	1.00	0.00	44 − 30 = 14
10	2	3	15	25	0.3	0	0	1.00	0.00	39 − 30 = 9
						0	0	1.00	0.00	40 − 30 = 10
						0	0	1.00	0.00	33 − 26 = 7
10	1	1	26	130	0.5	0	0	1.00	0.00	36 − 26 = 10
						0	0	1.00	0.00	34 − 26 = 8
						0	0	1.00	0.00	45 − 30 = 15
10	2	3	15	25	0.8	0	0	1.00	0.00	40 − 30 = 10
						0	0	1.00	0.00	41 − 30 = 11
						0	0	1.00	0.00	61 − 52 = 9
10	2	1	13	130	0.8	0	1	0.94	0.11	61 − 52 = 9
						0	1	0.94	0.11	60 − 52 = 8

Singh et al. propose a secure ownership transfer framework for IoT devices [24]. This framework leverages a trustworthy third party (akin to an escrow agent) to facilitate transfer between owners. Our approach does not require access to outside resources such as a ledger, or network access beyond the AP. Furthermore, our system does not rely on the AP after the devices obtain the scheduling information and cryptographic keys. Unlike ownership-transfer frameworks that wait for a user action, our devices continuously assess context via short-range peer broadcasts, reducing the amount of time data on the device is at risk and removing the need for users to notice a missing device.

8 Limitations and Future Work

Automatically decommissioning devices accurately and consistently is a challenging task. The correct device should decommission precisely when needed, while devices that remain in a familiar environment should continue to function. False negatives can lead to significant privacy and security issues. False positives can lead to loss of data or function. Here we discuss several limitations of our approach.

First, our approach requires direct communication with some set of participating neighbor devices. Our approach would not be applicable for scenarios with a lone device connected to a Wi-Fi AP, as there are no participating neighbors.

Ensuring accurate time synchronization between participating devices is critical to making efficient use of the channel, the computation resources of the devices themselves, and power consumption for battery powered devices. Low-cost systems such as the ESP32

often experience clock drift when run for extended periods of time. Although our longest experiment ran for 3 days with ten devices and reported no false positives, our system is sensitive to the ability of participating device clocks to stay in sync. While devices are connected to the AP, they all get NTP time synchronization, but if the AP were unavailable for an extended period of time, for example if the AP is physically destroyed and a new AP is not installed in a few days, device clocks could drift. We plan to explore the impact of (and remediation for) clock drift in future work, but we did not see issues with excessive drift in our experiments. We envision a solution where devices can use the captured timestamps from neighboring devices to compute the temporal offset between their own clocks and those of their peers.

We plan to build on existing algorithms for synchronizing clocks in distributed systems [8].

At present, our implementation and architecture is focused on Wi-Fi. We believe our approach could be extended to other radio protocols and architectures, including mesh networks. One interesting avenue of future work would be to explore using heterogeneous protocols, by observing devices using Wi-Fi, Bluetooth, and Bluetooth Low Energy simultaneously to build the STA/LTA ratio.

To be complete, an automatic decommissioning method should allow the owner of a device to confirm that the device has indeed decommissioned. At present, a removed device has no option to notify the owner that it has decommissioned (whether rightly or wrongly). We will explore methods for such notifications as future work.

We assume each device exposes an API that can render its data permanently inaccessible – for example by overwriting storage or, in encrypted designs, by deleting the local encryption key. When the key is escrowed in a trusted keystore, the owner can reverse an erroneous decommission by restoring the key. Formalizing these wipe and recovery workflows, and proving that they compose safely with our *when*-to-delete trigger, is a natural next step.

9 Summary

Many IoT devices collect and store sensitive credentials or user data; unfortunately, users often forget or are unable to remove this data when the device is decommissioned. Our proposed solution offers an effective autonomous mechanism for a device to make a decision *on its own* to make the sensitive data unavailable (for example, by erasing the data, or by deleting an encryption key). We evaluated our proposed approach using simulation and experiments. The simulation suggested 8 plausible parameter sets for our implementation, which our experiments showed 7 of the 8 parameter sets performed flawlessly. These results indicate our approach allows users to adjust parameters to suit their needs.

Acknowledgments

This research is supported by the National Science Foundation under award numbers CNS-1955805, and CNS-1955228. The authors also thank Chixiang Wang and Cesar Arguello for their assistance. The views and conclusions contained herein are those of the authors alone.

References

[1] Mansoor Alblooshi, Khaled Salah, and Yousof Alhammadi. 2018. Blockchain-based ownership management for medical IoT (MIoT) devices. In *International Conference on Innovations in Information Technology (IIT)*. IEEE, Al Ain, United Arab Emirates, 151–156.

[2] Apple. 2025. About Stolen Device Protection for iPhone. https://support.apple.com/en-in/120340. https://support.apple.com/en-in/120340

[3] Apple. 2025. Find My. https://www.apple.com/icloud/find-my/. [Online; accessed 14 July 2025].

[4] Ibrahim Baggili, Jeff Oduro, Kyle Anthony, Frank Breitinger, and Glenn McGee. 2015. Watch what you wear: preliminary forensic analysis of smart watches. In *International Conference on Availability, Reliability and Security (ARES)*. IEEE, Toulouse, France, 303–311.

[5] John R Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer, Berlin Heidelberg, 251–260.

[6] Espressif Systems. 2025. ESP-IDF FreeRTOS. Software package. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos.html

[7] Espressif Systems. 2025. Espressif Leads the IoT Chip Market with Over 1 Billion Shipments Worldwide. News Article. https://www.espressif.com/en/news/1_Billion_Chip_Sales

[8] Rui Fan and Nancy Lynch. 2004. Gradient clock synchronization. In *Proceedings of the Annual Symposium on Principles of Distributed Computing (PODC) (PODC '04)*. ACM, 320–327. https://doi.org/10.1145/1011767.1011815

[9] Dennis Giese and Guevara Noubir. 2021. Amazon Echo Dot or the reverberating secrets of IoT devices.

[10] Google. 2025. Android Theft Protection. https://blog.google/products/android/android-theft-protection/. https://blog.google/products/android/android-theft-protection

[11] Danny Yuxing Huang, Noah Apthorpe, Gunes Acar, Frank Li, and Nick Feamster. 2019. IoT Inspector: Crowdsourcing labeled network traffic from smart home devices at scale. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, Vol. 4. Princeton, ACM, 1–21. https://dl.acm.org/doi/10.1145/3397333https://iot-inspector.princeton.edu/iot-inspector-paper/

[12] Jouni Malinen. 2025. hostapd: Host Access Point Daemon. Software package. https://w1.fi/hostapd/

[13] Gulista Khan. 2021. A study of blockchain application for decentralized IoT device ownership. *ACADEMICIA: An International Multidisciplinary Research Journal* 11, 12 (2021), 27–33.

[14] Anastassija Kostan, Sara Olschar, Lucy Simko, and Yasemin Acar. 2024. Exploring digital security and privacy in relative poverty in Germany through qualitative interviews. In *USENIX Security Symposium (USENIX Security)*. Philadelphia, PA, 2029–2046.

[15] Ravindra Mangar, Timothy J. Pierson, and David Kotz. 2024. A Framework for Evaluating the Security and Privacy of Smart-Home Devices, and its Application to Common Platforms. *IEEE Pervasive Computing* 23 (2024), 7–19.

[16] Nicholas Miles and Chris Lyne. 2025. Inside SimpliSafe Alarm System. https://medium.com/tenable-techblog/inside-simplisafe-alarm-system-291a8c3e4d89 Tenable TechBlog on Medium.

[17] Taiwo Ojo, Hongmei Chi, Janei Elliston, and Kaushik Roy. 2022. Secondhand smart IoT devices data recovery and digital investigation. In *SoutheastCon 2022*. IEEE, Mobile, AL, USA, 640–648.

[18] Ahmad Sghaier Omar and Otman Basir. 2018. Identity management in IoT networks using blockchain and smart contracts. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, Halifax, NS, Canada, 994–1000.

[19] Souneil Park, Aleksandar Matic, Kamini Garg, and Nuria Oliver. 2018. When Simpler Data Does Not Imply Less Information: A Study of User Profiling Scenarios With Constrained View of Mobile HTTP(S) Traffic. *ACM Transactions on the Web* 12, 2, Article 9 (Jan. 2018), 23 pages.

[20] Eric Rescorla and Tim Dierks. 2025. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://www.rfc-editor.org/info/rfc5246

[21] George F Riley and Thomas R Henderson. 2010. The ns-3 network simulator. In *Modeling and tools for network simulation*. Springer, 15–34.

[22] Nazmus Sakib, Marzia Islam Mumu, and Nuran Mubashshira Momo. 2023. *Securing ownership management and transfer of consumer IoT devices with blockchain-based Self Sovereign Identity (SSI)*. Ph. D. Dissertation. Brac University.

[23] Samsung Electronics Co., Ltd. 2024. Samsung Cloud. https://www.samsung.com/us/support/owners/app/samsung-cloud/. [Online; accessed 17 July 2025].

[24] Kaptan Singh and Deepak Singh Tomar. 2023. Secure the ownership of WoT devices using secure ownership transfer framework. *International Journal of Information Technology* 15, 4 (2023), 2161–2171.

[25] Amadej Trnkoczy. 2009. Understanding and parameter setting of STA/LTA trigger algorithm. In *New manual of seismological observatory practice (NMSOP)*. Deutsches GeoForschungsZentrum GFZ, 1–20.

[26] Paisit Wongsongsarn. 2025. *SONOFF POW Ring Review – A WiFi CT Clamp power meter tested with eWeLink and Home Assistant.* https://www.cnx-software.com/2024/03/20/sonoff-pow-ring-review-a-wifi-ct-clamp-power-meter-tested-with-ewelink-and-home-assistant/ CNX Software.

[27] Matthew Wynn, Sandra Rueda, Kyle Tillotson, Ryan Kao, Andrea Calderon, Andres Murillo, Javier Camargo, Rafael Mantilla, Brahian Rangel, and Alvaro A Cardenas. 2017. Sexual Intimacy in the Age of Smart Devices: Are We Practicing Safe IoT?. In *Proceedings of the Workshop on Internet of Things Security and Privacy (IoTS&P)*. ACM, 25–30. http://dx.doi.org/10.1145/3139937.3139942