Lux: Enabling Ephemeral Authorization for Display-Limited IoT Devices

Logan Blue
University of Florida
bluel@ufl.edu

Samuel Marchal
F-Secure Corporation & Aalto University
samuel.marchal@aalto.fi

Patrick Traynor
University of Florida
traynor@ufl.edu

N. Asokan
University of Waterloo & Aalto University
asokan@acm.org

ABSTRACT

Smart speakers are increasingly appearing in homes, enterprises, and businesses including hotels. These systems serve as hubs for other IoT devices and deliver content from streaming media services. However, such an arrangement creates a number of security concerns. For instance, providing such devices with long-term secrets is problematic with regards to vulnerable devices and fails to capture the increasingly transient nature of the relationship between users and the devices (e.g., in hotel or airbnb settings, this device is not owned by the customer and may only be used for a single day). Moreover, the limited interfaces available to such speakers make entering such credentials in a safe manner difficult. We address these problems with Lux, a system to provide ephemeral, fine-grained authorization to smart speakers which can be automatically revoked when the user and hub are no longer in the same location. We develop protocols using the LED/light channel available to many smart speaker devices to help users properly identify the device with which they are communicating, and demonstrate through a formally validated protocol that such authorization takes only a few seconds in practice. Through this effort, we demonstrate that Lux can safely authorize devices to access user accounts while limiting any long-term exposure to compromise.

CCS CONCEPTS

• **Security and privacy → Authorization**; Access control.

ACM Reference Format:
Logan Blue, Samuel Marchal, Patrick Traynor, and N. Asokan. 2021. Lux: Enabling Ephemeral Authorization for Display-Limited IoT Devices. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '21), May 18–21, 2021, Charlottesvle, VA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3450268.3453530

1 INTRODUCTION

The popularity of smart speaker systems including the Amazon Echo and Google Home has recently exploded [34]. These devices allow users to not only interact intuitively via voice commands but can also serve as hubs to coordinate the activities of other co-located IoT devices. Additionally, users can link smart speakers to content services (e.g., Spotify, Netflix) and direct the smart speaker (or associated smart TV) to render media received from these services.

While the convenience of smart speakers is compelling, three important problems arise by sharing credentials with them. First, the security of many IoT devices is notoriously poor, significantly limiting the confidence with which users can entrust long-term secrets such as passwords to them. Second, a growing number of users are interacting with devices that they neither own or use for long periods of time yet wish to use them to gain access to content [33]. For example, hotels, enterprise meeting rooms, Airbnb properties, and other shared spaces increasingly come equipped with smart speakers [11, 51]. Even if users could trust such devices not to leak their long-term secrets, users typically have no option but to grant complete authorization to all connected services to the smart speakers (even during temporary visits). Constraining these devices to the principle of least-privilege is possible, but not without significant impact to usability and convenience (e.g., by the user having to remember to remove the credentials once temporary usage is over). Finally, because of limited interfaces (i.e., the general lack of screens and keyboards), identifying the correct device and entering credentials in a safe fashion (e.g., not reading them aloud via the voice interface) is difficult.

In this paper, we develop Lux, a system that addresses the need for *ephemeral authorization* of smart speakers to services. We consider the scenario in which a user is the temporary occupant of a shared physical space equipped with a smart speaker that allows the occupant to control connected IoT devices and access services on the Internet through voice commands. A user can ask the smart speaker to launch a movie on a connected smart TV using their Netflix or Hulu account, can place an order for food delivery using their Uber Eats account, or can ask to launch a video call to a friend using their Google Talk or Skype account. While the smart speaker has microphones, can play audio, supports several wireless communication technologies (e.g., 802.11, BLE), and has multiple LEDs, it lacks a traditional display interface. Our goal is to provide fine-grained authorization to specific accounts in this highly constrained yet potentially dynamic setting.

Lux avoids the over-provisioning of access rights to smart speakers during temporary visits using two main components. First, we create a subsystem for the initial temporary authorization of a smart speaker by making use of a *light channel* between the

IoTDI '21, May 18–21, 2021, Charlottesvle, VA, USA

ACM ISBN 978-1-4503-8354-7/21/05…$15.00
https://doi.org/10.1145/3450268.3453530

smart speaker and the user's personal device. This allows users to correctly associate with a specific speaker, which is critical in spaces where multiple devices potentially exist and names are not necessarily meaningful to the user (e.g., conference_room_1 vs. conference_room_2). Second, we develop a subsystem for automatic de-authorization based on determining the co-presence of the smart speaker and the personal device. Our co-presence technique is based on comparing WiFi access points visible to the two devices and is more efficient than previous co-presence detection techniques [54]. Acting in concert, the two subsystems enable Lux to temporarily authorize a smart speaker to act on the user's behalf and ensure that the authorization is suspended whenever the user leaves the speaker's proximity. While there has been previous work on context-assisted intuitive authentication, Lux is the first work that addresses the case of smart speakers.

We claim the following contributions:

- **Problem Identification:** Authorization is a long-studied space. However, the context in which we study it (i.e., users interacting for potentially short periods with devices that are neither owned by the user or have traditional interfaces) is emerging. To our knowledge, this work is the first to attempt to reason about authorization in this context (Section 2).
- **System Design:** We develop Lux, a system for intuitive and temporary authorization of smart speakers (Section 3 and 4) and automatic deauthorization based on co-presence determination (Section 5).
- **Systematic Evaluation:** We perform both a formal analysis of our protocols using ProVerif [5] and a comprehensive analysis of its performance in multiple settings (Section 6.1). Our results demonstrate that Lux is not only secure but also improves performance significantly over the most directly related work on co-presence detection (Section 6.2).

2 MODELS AND ASSUMPTIONS

2.1 System Model

Our system model consists of three parties: (a) a *smart speaker* (e.g., a Google Home) and an associated cloud service which we call the *root service*, (b) a user and their *personal device* (e.g., their smartphone) which is running an associated application for the smart speaker, and (c) one or more additional *online services* (e.g., Spotify music). Typically, the software running on the smart speaker, the personal device, and the root service are developed and distributed by the *same provider* (e.g., Google or Amazon). The user and the smart speaker occupy the same physical *shared space*, such as a room. Typical interactions among the parties are as follows:

The user issues a voice command (e.g., "play Game of Thrones on HBO") to the smart speaker, which, along with the root service and appropriate online service, processes and executes the command. This interaction requires the smart speaker to be authorized to access the user's online service account. The user typically provides their credentials to the root service, which grants access to the online service for the smart speaker and stores the credentials to enable authentications transparently later. This allows the online service to execute requests from the smart speaker and return the responses. Smart speaker are designed to execute any received voice command using the appropriate online service, however, this is not necessarily secure. The smart speaker should allow the owner of the account to access their online services but prevent others from do so in their absence.

Typically, users possess a personal device that they trust and use to run the smart speaker's companion application. Through this, the root service is granted long-term authorization to access various online service accounts of the user. Smart speakers are generally equipped with microphone(s), speaker(s), and personal assistant software. They also support several wireless communication technologies for connecting IoT devices. In addition, smart speakers are typically equipped with several LED lights to provide visual information about their current status.

2.2 Adversary Model

We consider as an adversary any device or person that has access to the shared space where the smart speaker is located. The goal of the adversary is to use the online services for which the smart speaker was granted access to by the user. An adversary could use these services by obtaining the user's credentials or directly using the smart speaker or another device that has been granted access to the user's accounts.

We assume that neither the personal device nor the online service provider is compromised. Additionally, we assume that the adversary cannot undetectably compromise the smart speaker prior to or during its use by the temporary user. However, the adversary may compromise the smart speaker once the user has left the shared space for an extended period. We allow the adversary to observe any communication within the shared space and to inject their own malicious messages.

2.3 Requirements

Our goal is to design and implement a protocol for secure delegation of temporary access rights to a user's online service for a smart speaker in close proximity. Our protocol was designed with the following goals in mind:

- **G1** The personal device and user need to be able to correctly identify which smart speaker they wish to connect to.
- **G2** The smart speaker does not need to authenticate the personal device. It should participate in this protocol with any party that is willing to grant it access to their accounts.
- **G3** The root service needs to be able to construct a binding between the smart speaker and personal device. This relationship is required for the root service to grant the smart speaker the permissions of the account associated with the personal device.
- **G4** The protocol should only rely on the root service that is associated with the built-in assistant on the smart speaker. Access to all other online services will be granted via the root service.

In addition, our protocol needs to meet several other requirements:

- **R1** **Minimalism**: The smart speaker must only be granted access to the subset of services that the user actually uses with the smart speaker. The burden imposed to the user for granting access should be the same regardless of the number of services used.

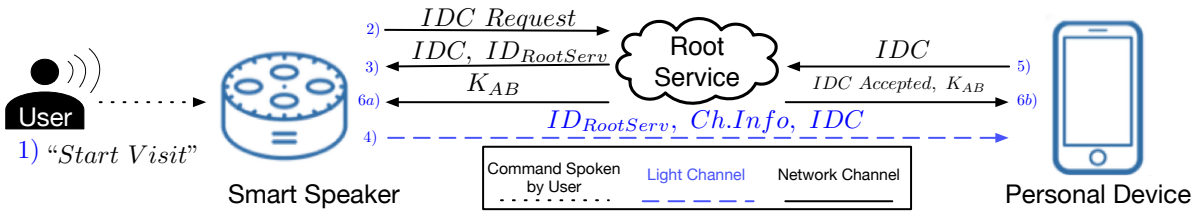

Figure 1: The initialization protocol that authorizes the smart speaker to access the root service on the user's behalf. At the end of the protocol, the root service will issue a shared secret to the smart speaker and the personal device for use in future communications.

- **R2 Deployability**: The changes stipulated by the protocol must be deployable by a single entity, the one that controls software across the smart speaker, personal device, and root service. It must be easy to implement considering the current capabilities of these devices/entities (e.g., no hardware modification should be required).
- **R3 Bounded authorization**: The granted access rights must be bound to the temporary usage in space and time. They should be suspended or expire when the user is not present in a space or when temporary usage ends.

2.4 Lux overview

Our solution, called Lux, to the secure delegation of temporary access rights to a user's online service for a smart speaker consists of two main components:

- **A protocol** defining *how* to securely grant and revoke access rights to online services for a smart speaker (Section 3 and 4)
- **An automated mechanism to detect the co-presence** of a user and a smart speaker, which in turn determines *when* access rights must be granted and revoked (Section 5).

Our protocol provides the user with a transparent way of selecting, authorizing, and de-authorizing a smart speaker for temporary use. Each service the users wishes to access is individually authorized, thus enforcing the principle of least privilege. The protocol is designed to require no third party (i.e., the hotel or airbnb deploying the smart speakers) support in order to function. This helps to simplify the protocol and prevent additional parties from having to known the user's credentials or actively support our protocol.

Our co-presence detection detects reliably and in a timely manner when the user joins and leaves the shared space. This way, access rights can be *dynamically* granted to the smart speaker when the user is present and *dynamically* revoked when it is absent. This approach provides better security guarantees than a pre-defined validity period for access rights specified by, for example, the check-out time for hotel room or the end of the booked time for a meeting room.

3 PROTOCOL DESIGN

In order to address our goals laid out in the previous section, we designed a three-phase protocol. The protocol allows root services to provide smart speakers with access tokens, which exists within one of three states. These three states are active, suspended, and expired which reflect the token's current validity. The states are automatic set by the de-authorization method discussed in Section 5.

The first phase of the protocol (Section 3.1) will be used to authorize the root service as well as share a secret between the user's personal device and the smart speaker. This phase is only required to be performed once during the initial setup protocol. The shared secret created during this phase is used in later phases of the protocol to authenticate the smart speaker to the personal device. Authorization to additional online services are granted to the smart speaker as the user requests them via the second phase protocol described in Section 3.2. This phase is only required when the user interacts with an online service for the first time. Finally, our third phase (Section 3.3) consists of the transitions that can occur between the various states a smart speaker's token can be in. This phase of the protocol is responsible for dynamically revoking and granting authorization for previously used online services based on the presences of the personal device.

3.1 First Authorization and Setup

The first phase is shown in Figure 1. At this stage, the user is attempting to connect to the smart speaker in the space they are temporarily visiting. The user likely does not know how to connect to a given smart speaker or even which device to connect to. If the user were to simply select a device from a list, they could authorize the wrong device (e.g., the device in the next room over). To mitigate both of these factors and satisfy **G1** and **R1**, we make use of a non-traditional communication channel, light. The light based channel helps to satisfy **G1** since a user can intuitively indicate which smart speaker they wish to authorize by directing their personal device's camera at it. Additionally, the light channel aids in meeting **R1** since many already deployed smart speakers are equipped with LED indicator lights (12 in the case of the Amazon Echo and Google Home). To begin, the user provides a voice command informing the smart speaker that they wish to begin the protocol, (1) in Figure 1. The smart speaker will then communicate to the root service that a new visit is about to begin and request an Inter-Device Communication identifier (*IDC*) from the root service over a TLS connection (2). The *IDC* is a unique identifier that the root service uses to link the two devices and is returned to the smart speaker via (3) over the same TLS connection.

Next, the smart speaker needs to communicate the *IDC* as well as other necessary information to the user's personal device. However, at this point the devices still do not have a communication channel between them and the user still has not indicated which device they wish to authorize. Theoretically all smart speakers within audible range have performed messages 1-3 with their respective

root services. Now the user will direct the personal device's camera at the indicator lights on the selected smart speaker efficiently indicating which smart speaker they wish to authorize. This removes the possibility of the user selecting the wrong smart speaker at this point. The smart speaker will use its lights to transmit an identifier of which online service's interface is being authorized ($ID_{RootServ}$), a list of all supported networking channels the smart speaker supports (e.g., a WiFi SSID and the device MAC Address), and the *IDC* value provided by the root service, (4) in Figure 1. This ensures the user has selected the correct device they wish to authorize (**G1**) and forces the personal device and smart speaker to be located within close proximity for several seconds. During this time, the personal device will configure our automatic de-authorization method outlined in Section 5.

The personal device does not need to send any additional information to the smart speaker, given that the smart speaker does not need to authenticate the personal device (**G2**). Once the personal device has received (4), it will decode the message and attempt to validate the *IDC* by presenting it to root service indicated in message (5). The root service will accept the first device to arrive and present the specific *IDC* identifier. Once the *IDC* has been presented to the root service, it is able to link the personal device with the smart speaker and allow it limited access to the user's account satisfying **G3**.

Next, the root service responds to the personal device with whether or not the *IDC* was accepted and if it was, a key K_{AB} will also be sent (6b). By informing the personal device of success or failure, we prevents a possible adversary from forcing the user to use their malicious account. This is possible if an adversary presents the *IDC* value before the user's personal device. If such an attack were to occur, the user would be alerted that their accounts failed to be connected to the smart speaker. The same K_{AB} is sent to the smart speaker in (6a) that was sent to the personal device in (6b). This new shared key K_{AB} will be used by the two devices for later communications.

Upon a successful *IDC* validation, the personal device will select a communication medium from the list provided in (4). It will then open a connection to the smart speaker for use in future authorization requests. This connection will be maintained whenever the personal device detects that it is co-located with the smart speaker. In addition, whenever the personal device is co-located with the smart speaker, both devices will need to maintain a second TLS connection to the root service as well. We will discuss these channels in depth in the next two subsections. By the end of this protocol, the root service will have created a meaningful connection between the personal device and smart speaker while authorizing the smart speaker as a device the user wishes to use. However, at this point the smart speaker only has permission to access the user's root service account, but not any of the other online service linked to that account. This prevents the temporary device from becoming over privileged and ensures that the temporary smart speaker has as few privileges as needed.

The personal device and smart speaker must maintain K_{AB} and their TLS sessions with the root service. The TLS sessions do not only serve as secure communication channels, *they also act as authentication tokens for our protocol.* One TLS connection must be

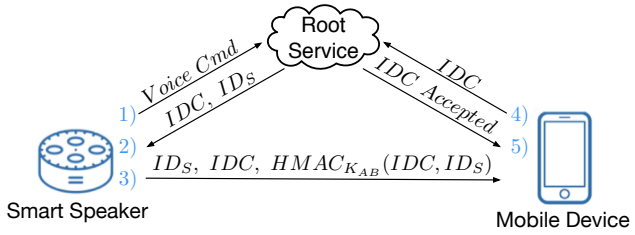

Figure 2: The Secondary Authorization protocol is used to grant access to the smart speaker for additional online services as the user makes requests.

maintained by the personal device per temporary device it has authorized because the user can maintain temporary ownership of multiple smart speakers. This prevents the user from accidentally granting additional authorization to the wrong smart speaker if two smart speaker's share the same root service. The smart speaker will also have to maintain a unique TLS connection for every personal device that connects to it. These sessions do not necessarily need to be active, but a resumption token does need to be maintained. The root service also needs to maintain knowledge of both TLS connection (personal device and smart speaker) and a list of online services the smart speaker has been authorized to access.

3.2 Secondary Authorization

Once the first authorization phase as successfully completed, the two devices now share a secret key and a network communication channel. Through this channel, our two devices can securely authorize additional online services the user requests, helping to satisfy **R1**. Unlike before, the user does not have to specifically request for authorization. Instead, the user simply makes a request which requires the smart speaker to access a new online service. When this request is made, the protocol shown in Figure 2 will begin. The recorded command from the user is sent to the root service over the pre-existing TLS connection the smart speaker has maintained (1). The root service will then process and check if the smart speaker has been granted authorization to access that online service. If it has, it will fulfill the request as normal. Otherwise, it will respond with an *IDC* and online service identifier (ID_S), shown in message (2) of Figure 2.

The smart speaker will then forward the *IDC* and ID_S to the personal device via their shared network connection (3). This message also includes a keyed HMAC of the sent *IDC* and ID_S. The personal device will recalculate this HMAC in order to verify the message's integrity and the sender's knowledge of the secret key K_{AB}. However, the network connection between the two devices may not currently exist. The connection between the smart speaker and the personal device, opened at the end of the First Authorization Protocol, is only maintained when the personal device detects that it is co-located with the smart speaker. Without this connection, the smart speaker cannot send (3) and thus no new access is granted. If, however, it is co-located and the network connection is therefore active, then (3) will be sent to the personal device. The co-location detection method is discussed later in Section 5.

After verifying the HMAC of (3), the personal device will present the *IDC* to the root service using the pre-existing TLS connection it has maintained (4). The root service will check if the *IDC* value

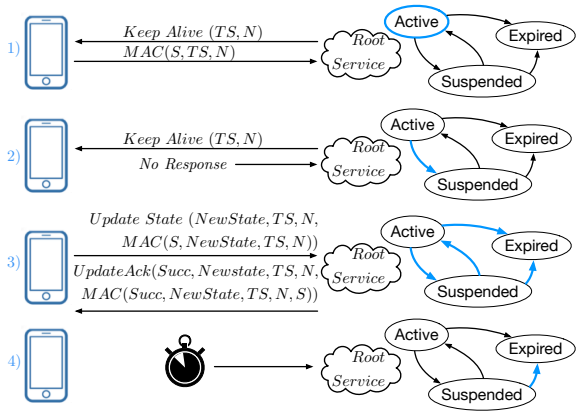

Figure 3: A state machine that represent the various phases a token can be in.

is correct and then alert the personal device to whether the *IDC* was accepted or rejected (5). If accepted, the root service will add the requested online service to the list of online services the smart speaker has access to. Following this, the smart speaker will receive a response to its earlier query. If rejected, the smart speaker will simply timeout on that request.

3.3 Permission States

As mentioned at the beginning of this section, the access tokens can exist within one of three states: active, suspended, and expired. The active state represents a token that currently has permission to make requests on the user's behalf. This is the state that the token will remain in during use and when the user is co-located with the smart speaker. The suspended state represents when a token is temporarily inactivated because the user is not co-located with the smart speaker. This state is for when the user has temporarily left the shared space (e.g., when the user has left the hotel or airbnb room for the day). Finally, the expired state is for when token has been determined to be no longer relevant to the user. This accounts for when the user relinquishes their temporary control over the smart speaker and the shared space (e.g., checking out of the hotel or airbnb).

The protocol will automatically move the access token into the different states depending on several factors. Figure 3 shows the various transitions that can be made between the different states. In transition 1, we see the root service maintaining a keep-alive with the personal device. This keep-alive maintains that the device is both active and still co-located with the smart speaker. The personal device will also need to maintain a local connection to the smart speaker in case additional authorization is needed.

As long as the personal device continues to respond to the keep-alive messages, the access token will remain in the active state. This will allow the user to continue to access to their accounts via the smart speaker. If the personal device does not respond to a keep-alive message or if the TLS connection is closed, then the root service will transition the access token into the suspended state. This is transition 2 in Figure 3 and it ensures that the default state for an access token is suspended. The frequency of the keep-alive can be altered in order to optimize the tradeoff between battery consumption and security. Less frequent keep-alive messages (e.g., the default in the Android Socket library is two hours [29]) can minimize the impact of these messages on the personal device's battery but allows a potentially larger window for an adversary to gain unwarranted access to the user's accounts.

The personal device can also automatically update the access token to any state. This is done through transition 3 and provides an interface for the personal device's Co-Presence Inference to update the token based on the user's location. This message contains the new state for the access token, a timestamp for freshness, and a nonce. The online service will in turn respond to the device with an additional message indicating the success of the transition. This is the only way to move the access token back to the active state once it has left. If moved to the active state, the personal device and root service will begin an active TLS connection with periodic keep-alive messages (transition 1). Transition 3 may also be used by the user in order to preemptively expire a token if need be (e.g., changing rooms at a hotel). The last possible transition, transition 4, comes from the lack of communication from the personal device after a long period of time. This transition will cause the root service to move the access token into the expired state. In reality, the expired state is the deletion of the access token. The root service performs this transition by deleting the TLS resumption ID for the smart speaker and the personal device. This forces the smart speaker to reinitialize the TLS connection and account authorization (which it cannot do without the personal device) in order to access the user's accounts. By having the token expire after a certain amount of time of inactivity, we prevent the smart speaker and root service from having to maintain overly large histories of access tokens. Additionally, it allows users to leave a temporary space without having to force the token into the expired state via a type 3 transition. The user can simply leave the location and know that in a predetermined amount of time (e.g., 1-3 days for a hotel or airbnb) the token will be useless.[1]

4 PROTOCOL IMPLEMENTATION

Our protocol involves three parties: the personal device, the smart speaker, and the root service. We present one of many possible implementation for each party here.

4.1 Personal Device

4.1.1 Main Device. The personal device in this paper was represented by a Nexus 6p running Android 8.1. It connected to both our root service and smart speaker implementations using the local WiFi and constructed messages over Protobuf [5]. We used the built-in camera application on the Nexus to adjust the white balance and record a short video at 29.97 frames per second of the lights on the smart speaker as they transmitted the light message in our protocol. We chose to outsource the decoding of the light message to a different machine to ease the construction of our prototype. However, popular apps like Snapchat and QR Code scanners show that doing on device video processing is possible and effective.

We took several other steps to simplify our prototype, such as 3D printing a jig to hold our phone during testing. The jig allowed

[1]The predetermined amount of time before expiration is variable, so the time could be increased to several months (or never) for more long term devices.

us to take consistent measurements and to run multiple tests automatically. Additionally, the protocol was initiated by an on screen button rather than by the speaker's voice command.

4.1.2 Light Decoder. The recorded video file was sent over WiFi from the Nexus to a nearby iMac running an Intel i7-6700k with 32 GB of ram. Our choice to process the video on a secondary machine was simply to facilitate easier and faster code development. The Light Decoder program was written in Python 3.7.0 using the scikit-video library. In a production system, this video processing step would be completed locally on the personal device.

The light message consisted of a series of flashing RGB LEDs from our smart speaker. A single LED was determined to be the clock, which would flash on or off in order to indicate that new data was available. The rest of the LEDs were used to transfer data. Each pixel consisted of three channels: red, green, and blue. A single bit of data was transmitted using a single channel of each pixel. In order to reliably decode our data, the decoder needed to adjust for the white balance in the video. To do this, the smart speaker would display every possible combination of an LED (eight in total) before the message began. The decoder would use these values to find the average intensity for each LED in an on or off state and define a threshold for each color channel. During transmission, the decoder would monitor the clock LED and collect the other LED's values every time it changed. Once the transmission ended, each LED value was decoded into three bits using the thresholds found earlier. To improve reliability, we added a (7, 4) Hamming Error Correcting Code (HECC) scheme [21] which needed to be decoded. Additionally, it needed to truncate the message to the appropriate length which was specified at the beginning of the message. Finally, the decoder would parse out the separate fields in the message before sending the extracted data back to the Nexus for use in the protocol.

4.2 Smart speaker

While several commercial smart speakers are available on the market, its is not easy to implement customized functionality on them. Consequently, we decided to build a custom smart speaker to simplify the implementation of our protocol. We believe the hardware configuration we selected for our smart speaker is an effective proxy for actual commercially available products. Many of these smart speakers are equipped with LED indicator lights which we leverage to enable our light communication. For example, the Google Home and Amazon Echo both come equipped with 12 built-in LED indicator lights.

4.2.1 Main Device. We used a Raspberry Pi 2 B+ with a Matrix Voice [42] development board as our smart speaker. The Matrix Voice is equipped with 18 RBG LEDs and connects to the Pi via its 40 GPIO pins. The Pi connected to both the personal device and the root service via its Ethernet port. The code for our smart speaker was written in Python 3.5.3 running on Rasbian 9.

4.2.2 Light Encoder. The Pi created the light message by first encoding the necessary data into binary. Next, the data was encoded using the (7, 4) HECC scheme mentioned earlier. The length of the coded message was then prepended to the message (also encoded using the (7, 4) HECC scheme) before the message was sent to the Matrix Voice development board via its build in APIs. The data was divided into section of 51 bits that made up a single frame. A frame consisted of 1 LED either being on or off to service as the clock and the other 17 LEDs serving for data transfer. The data transferring LEDs each represented three bits of data, one bit for each of three channels red, green, and blue. The LEDs were changed 15 time per second to ensure that the camera on the personal device was able to capture every frame of data. This gives our light communication channel an effective throughput of approximately 95 bytes per second of HECC encoded data.

4.2.3 Light Diffusion Filter. The LEDs on the Matrix Voice development board are not covered by anything, and thus the light emitted by them blew out the light balance on the Nexus's camera. This resulted in the LEDs appearing white regardless of RGB value (provided at least one channel was on) when recorded by the camera. To eliminate this issue, we construct a light diffusion filter which consisted of piece of card stock and sanded piece of AF4300 Overhead Projector Film. These two materials placed over the LEDs diffused enough light to allow our camera to detect the actual color displayed. This is consistent with smart speakers available on the market today such as the Google Home or Amazon Echo, both of which have built-in diffusion filters over their lights.

5 AUTOMATIC AUTHORIZATION VIA CO-PRESENCE DETECTION

In order to evaluate and test out ephemeral authorization protocol, we need to leverage a form of co-presence detection. The underlying technique used in this paper is one of many possible localization techniques (e.g., UWB) *that could have been selected as they become more widely available.* It is important to note that our protocol could be use with other or future localization techniques that perform better overall or in a specific scenario.

However, in this work we chose to use the technique discussed in this section to predominately satisfy **R3**. To do this, Lux needs to promptly detect when the user leaves the shared space, and suspend the authorizations granted to the smart speaker until the user re-enters the shared space. We take a simple approach for realizing such de- and re-authorization based on detecting co-presence of the personal device and smart speaker. We will refer to this process as "automatic de-authorization."

5.1 Our approach: comparing WiFi scans

A number of co-presence detection techniques are known from the literature. In principle, Lux could use any of them for automatic de-authorization. However, co-presence detection techniques relying solely on the availability of short range wireless communications are vulnerable to relay attacks which have been demonstrated for Bluetooth [37], RFID [14], and NFC [15]. A potentially secure alternative is to use a distance bounding protocol [7, 50]. However, distance bounding requires extra hardware [3] that is not typically available on personal devices and smart speakers, thereby limiting deployability (**R2**). Although distance bounding techniques are commercially available, they are not yet sufficiently inexpensive for cost-sensitive commodity devices. A third class of co-presence techniques is based on having the two devices sense their ambient environment via their available sensors [31, 43, 54]. Some of

these techniques were designed for zero-interaction authentication which is only a one-time operation [31, 43]. In contrast, Lux requires *continuous co-presence detection.* Consequently, Lux needs a co-presence detection mechanisms that has minimal resource requirements.

Truong et al. [54] showed that comparing available WiFi access points in the vicinity is an effective means to detect the co-presence of two devices. From the perspective of deployability (**R2**), this approach is particularly suitable for Lux because WiFi monitoring has low resource requirements and is straightforward to implement on any personal device such as smartphones, tablets, etc. (all equipped with WiFi chips and providing convenient APIs). We now describe a co-presence detection methodology that is as effective as the one proposed by Truong et al., but is simpler and more efficient. We assume the smart speaker is stationary, it remains in the same shared space overtime, to devise our technique where the personal device determines if it is co-present with the smart speaker. When the personal device initially associates with the smart speaker (during First Authorization), it records a reference *signature* Sig_{loc} for the shared space. This signature is based on the surrounding WiFi access points (APs) at the time of this association. Later, the personal device periodically computes a similar signature Sig_t for its location at time t. Sig_t is compared to Sig_{loc} to infer if the personal device is in the same location as the smart speaker.

5.2 Methodology

The personal device monitors beacon frames from WiFi access points and select the MAC addresses of access points which correspond to Received Signal Strength (*RSS*) higher than a pre-defined threshold dB_{lim}. The signature *Sig* of a location consists of the set of tuples $< mac_i, p_i >$ where p_i is a metric for the received signal strength RSS_i from an access point with the MAC address mac_i. To enable relevant comparison of signatures, we compute p_i as shown in Equation 1. We normalize the signal strengths such that the sum of all p_i from a signature *Sig* is equal to 1.

RSS_i (in decibels) is a negative value typically between -100 and -20. We subtract the threshold value dB_{lim} and add a constant number (10) to ensure that p_i is positive.

$$p_i = \frac{RSS_i - dB_{lim} + 10}{\sum_{j=1}^{n} RSS_j - dB_{lim} + 10} \tag{1}$$

Consequently, *Sig* can be seen as a probability distribution:

$$Sig = \{(mac_1, p_1), \cdots, (mac_n, p_n)\}, \text{where } \sum_{i=1}^{n} p_i = 1 \tag{2}$$

The personal device computes the initial signature for the location of the hub (Sig_{loc}) and subsequent periodic signatures of its own location at time t (Sig_t) in the same manner. The only differences are that Sig_{loc} is computed from n consecutive WiFi scans and that only MAC addresses which we received more than $n \times scan_{rate}$ beacons were included in Sig_{loc}. Additionally, the parameters $scan_{rate} \in [0, 1]$ and $dB_{lim} \in [-100, -20]$ are fixed.

The personal device computes the distance between the probabilistic distributions Sig_{loc} and Sig_t using the Hellinger distance H [8], a metric used to quantify the dissimilarity between two probabilistic distributions. H is bounded in $[0, 1]$ where 1 represents complete dissimilarity and 0 means equality.

Definition 1 (Co-presence detection). *Given a signature Sig_{loc} associated with a smart speaker and a signature Sig_t for the location of the personal device at time t. The personal device determines if it is co-present with the smart speaker (i.e., located in the same shared space) at time t if the distance between the two probabilistic distributions Sig_{loc} and Sig_t (as defined by the Hellinger distance $H(Sig_{loc}, Sig_t)$) is lower than the detection threshold δ*

$$H(Sig_{loc}, Sig_t) < \delta \tag{3}$$

5.3 Implementation

Our co-presence inference is implemented as an Android application running on the personal device. It uses the Android `WifiManager` API to scan for beacon frames.

The computation of Sig_{loc} for a specific smart speaker is triggered when the user launches Lux's First Authorization Protocol (Section 3). The personal device remains in close proximity to the smart speaker (required by the light channel communications) during this process. Our application records several scans while the personal device remains stationary. Whenever movement detected by the personal device's accelerometer (`SensorManager` API), the recording process is stopped. All scans recorded until then are used to compute Sig_{loc}.

Sig_t distributions are computed at regular time intervals (e.g., 30 seconds) in a background process and compared to Sig_{loc} for co-presence detection. The application uses the result of this process to keep track of the state of the personal device (co-present or not). Whenever the state changes, an Android intent containing the new state is sent to the Lux protocol application which uses this information to manage its permission state as described in Section 3.

6 EVALUATION

6.1 Protocol Evaluation

The evaluation of our protocol consists of three parts. Parts one and two are a simple timing analysis of the First Authorization and Secondary Authorization protocol respectfully. The last part is a more formal analysis of the both the previously mentioned protocols using ProVerif [5].

6.1.1 Timing Analysis of the First Authorization. The First Authorization protocol takes on average $4,036 \pm 320$ ms to perform over 10 recorded runs. While this may seem slow, it is much faster than the standard procedure for setting up, for example, a Google Home. According to Google's own documentation [18], initial setup is a 19 step process that takes several minutes to complete.

The First Authorization protocol has four phases: the initial *IDC Request* made by the smart speaker (messages (2) and (3) in Figure 1), the encoding and transmission of the light message (message (4)), the decoding of the light message via the secondary device (between messages (4) and (5)), and lastly the *IDC* presentation made to the root service by the mobile device (messages (5), (6a), and (6b)). Only the initial *IDC Request* and the encoding and transmitting of the light message will be witnessed by our user.

The initial *IDC Request* only takes 36 ms or 0.90% of the total time to perform, which is acceptable.

The encoding and transmission of the light message is a segment of the protocol observed by the user and takes 1,853 ms or 46.1% of the total protocol time to perform. The time needed to transmit this message is related to two factors. The first factor is the length of the supported channel list. Shorter channel names, using a fix port, or only supporting a limited set of communication channels could shrink the message length significantly, but it would hurt the overall connectivity of the device. The second factor is the throughput of the light channel and is a more promising prospect to decrease the time of this segment. Currently, we use a (7, 4) HECC scheme that results in only 57.1% of the transmitted bits being actual data. We can nearly double our throughput by using a more efficient scheme like (127, 120) HECC or (255, 247) HECC. Additionally, we only encode 3 bits of data per LED, 15 times a second. RGB LEDS can support 2^{24} different values, so finding an encoding scheme that achieves more than 2^3 bits per LED is very possible with more advanced video processing. Additionally, many cameras found on personal device are capable of recording video at higher rate than 29.97 FPS. By leveraging that, the throughput of the light channel could increase by up to an order of magnitude. Consequently the 1,853 ms our system takes to transmit the light message should be viewed as an upper bound of such communications.

The third phase, decoding of the video message, took on average 2,016 ms or 49.7% of the total protocol time. This section of the protocol is after the user interaction has ended and will only affect how long until the user is informed on the status of their authorization. This greatly reduces the impact of this phase, making 2,016 ms an acceptable amount of delay.

The final phase took 129 ms which is the time the user has to wait before using the smart speaker after recording the video message.

6.1.2 Time Analysis for the Secondary Authorization. On average, the Secondary Authorization protocol takes 155 ms to perform with a standard deviation of 6.2 ms over 10 recorded runs. The main side effect of this protocol's run time is the small delay the user will experience while attempting to use an online service for the first time on a new smart speaker. We believe that the 155 ms is a reasonable amount of time for the user to wait for a one time operation. In this phase, we believed this protocol is bounded by the network latency caused by the five network messages. We found that 135 ms or 79.66% of the total protocol time was used by network communications.

6.1.3 ProVerif Analysis. We used ProVerif [5] to analyze our First Authorization and Secondary Authorization protocols. Our analysis reveals that of both protocols never compromised the TLS connections between any involved parties. Also, it determined the Secondary Authorization protocol does not leak the key K_{AB}. By ensuring that our protocol does not leak K_{AB}, we can assert that an adversary cannot forge $HMAC_{K_{AB}}(IDC)$.

6.2 Automatic De-authorization

We evaluate our automatic de-authorization method along two criteria: 1) accuracy of co-presence detection when the personal device and the smart speaker are co-present in the shared space, and 2) delay in de-authorization after the user has left the shared space. High accuracy in co-presence detection implies better usability as it ensures that the smart speaker will promptly carry out commands by an authorized user (owner of the personal device). Low delays in de-authorization implies better security as they guarantee that an adversary cannot make use of a user's online service authorizations after that user left the shared space.

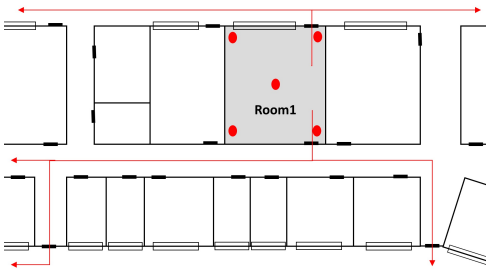

Figure 4: Example room used for collecting WiFi scans to evaluate our co-presence detection method. Red dots represent data collection points for the presence dataset. Red arrows represent paths used for leaving the room during collection of the exit dataset.

6.2.1 Data collection. We collected IEEE 802.11 beacon frames using an Android application on a Google Nexus 6 smartphone running Android 7.1.1. The application recorded beacon frames received by the smartphone at pre-defined time intervals. These frames are stored in a *scan* along with the timestamp and the location where the scan was performed (e.g., *room1-corner2*). Data was collected from three different locations, *room1* and *room2* are located in a university building while *room3* is a studio apartment. We gathered two datasets, the presence dataset and exit dataset:

The *presence dataset* consists of scans performed in five different locations of each room: the center and the four corners as depicted in Figure 4. We scanned for beacon frames every 30 seconds for one hour at each monitoring location gathering 600 scans per room.

The *exit dataset* consists of scans performed by a person leaving a room while carrying the smartphone, starting from the center of the room and walking out. Each room had multiple different exit paths (example depicted in Figure 4). The action of leaving was repeated 10 times for each of the three rooms, choosing a random exit path each time. Scans were recorded every five seconds during an 80-second period, resulting in 510 scans (17×30).

6.2.2 Evaluation setup. Before evaluating our co-presence detection method, we must define its hyper-parameters dB_{lim} and $scan_{rate}$ (cf. Section 5). We use the presence dataset to build a random signature Sig_{loc} by selecting n consecutive scans for a given scan location, for example *room1-corner2*. The remaining scans from the same room (not used to compute Sig_{loc}) are used to compute different Sig_t. We then compute the mean distance $H(Sig_{loc}, Sig_t)$ for different hyper-parameters values. Our goal is to find optimal values for dB_{lim} and $scan_{rate}$ that minimize $mean(H(Sig_{loc}, Sig_t))$ for all signatures from the same room. This ensures that a signature Sig_{loc} has a maximum similarity with any other scan Sig_t performed in the same room.

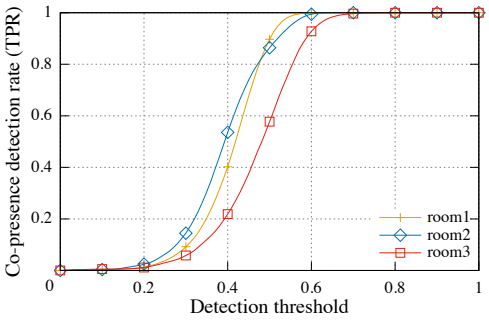

Figure 5: Increase in co-presence detection (TPR) in three different scenarios (room) as we increase the detection threshold δ. A threshold $\delta = 0.7$ ensures 100% co-presence detection in all tested scenarios.

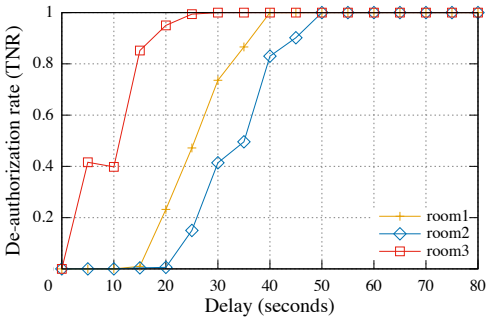

Figure 6: Increase in de-authorization rate as the delay after leaving a location increases ($\delta = 0.7$). The rate can have a non-monotonic evolution (e.g. room3) as the path for leaving a room may require to first go further from and then come closer to the room (by following corridors).

We performed a "grid search" over all pairs of hyper-parameter values $< dB_{lim}, scan_{rate} >$ drawn from the space $dB_{lim} = \{-90, -85, -80, -75, -70, -65, -60\}$ and $scan_{rate} = \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ to minimize $mean(H(Sig_{loc}, Sig_t))$. We selected $dB_{lim} = -85$ and $scan_{rate} = 0.5$ as hyper-parameters values. These values did not provide the minimal $mean(H(Sig_{loc}, Sig_t))$ but they were a good tradeoff between low average distance H and a long signature length. The length of the signature is linked to its security and the effort required by an adversary to spoof it. $dB_{lim} = -85$ and $scan_{rate} = 0.5$ are used for the remaining of the experiments.

We computed the average length (number of APs) for 50 randomly generated signatures Sig_{loc} using these hyper-parameters. We obtained *room1*: 13.17, *room2*: 12.72 and *room3*: 7.45. Such a signature length provides a satisfactory security level (cf. Sect. 6.2.5).

6.2.3 Accuracy of co-presence detection. We select hyper-parameter values similarly to before. Then, we randomly select $n = 6$ consecutive scans from the same location in the presence dataset, e.g., *room1-corner2*, to compute Sig_{loc}. The m remaining scans from the same room (e.g., room1) that were not used to compute Sig_{loc}, are used to compute m different Sig_t values. We apply our co-presence detection criteria (Definition 1) for each Sig_t: $H(Sig_{loc}, Sig_t) < \delta$ and compute the ratio of Sig_t meeting the criteria. This ratio is the rate of co-presence detection or the true positive rate (TPR). The TPR is maximized for signatures computed from scans performed in the same room. We compute this ratio for different values of the detection threshold δ.

Figure 5 depicts the increase in co-presence detection rate for our rooms as a function of the detection threshold δ. Each data point represents an average detection rate computed over 50 runs of the experiment using randomly selected Sig_{loc} signatures. We see that a δ value greater than 0.6 provides reliable co-presence detection with a TPR reaching 0.998 in both *room1* and *room2*. However, *room3* requires a higher δ to reach the same detection rate. Selecting a value for δ is a tradeoff. A high δ value increases the co-presence detection rate while a lower value reduces the delay in inferring that the user has left the shared space. Figure 5 and Figure 6 graphically display both of these findings. Consequently, we selected $\delta = 0.7$ as the optimal value, producing TPRs of: $TPR_{room1} = TPR_{room2} = 1$ and $TPR_{room3} = 0.997$.

6.2.4 De-authorization delay. We evaluate the delay for detecting non-co-presence after the personal device leaves the shared space with the user. First, we generate a random signature Sig_{loc} from the presence dataset using the same process as during our accuracy evaluation. Second, we compute the signatures Sig_t for each scan of the exit dataset. All Sig_t are tested for co-presence detection against the Sig_{loc} of their respective room. We compute the ratio of non-co-presence detection ($H(Sig_{loc}, Sig_t) \geq \delta$) according to the delay (in seconds) after leaving each room. This provides the ratio for de-authorization or true negative rate (TNR) as a function of the delay after leaving a room. Each computation is repeated 50 times using randomly selected Sig_{loc} signatures. Figure 6 presents the results of this experiment.

We observe that leaving a shared space is detected within a maximum of 50 seconds, while the average delay for detection is around 20–25 seconds. This corresponds to reaching a distance of 25 meters away from the center of the room assuming a walking speed of 4km/h. It is worth noting that the delay for detecting non-co-presence is much shorter for *room3* (apartment building) than for *room1* and *room2* (university building). Our interpretation of this phenomenon is two-fold. First, the apartment building has a configuration with smaller rooms and more walls than the university building, which results in WiFi signal strength fading faster. Second, the university building is likely equipped with high-end wireless access points, while the apartment building contains mostly low-end personal access points owned by tenants. Both factors make the WiFi signals strength in the university building stronger and more stable than in the apartment building.

6.2.5 Adversarial resilience. An adversary can try to fool the personal device into inferring that it is co-present with the smart speaker while it is actually not. To do so, the adversary must modify the signature Sig_t computed by the personal device such that $H(Sig_{loc}, Sig_t) < \delta$. We assume the adversary cannot remove WiFi APs from the neighborhood of the personal device but can only add tuples to Sig_t by sending WiFi beacon advertisements. To effectively modify Sig_t, the adversary must 1) know the exact signature Sig_{loc} of the smart speaker location as computed by the personal device, 2) predict the location of the personal device at the time of the attack and 3) deploy several adversarial APs at chosen distances from this predicted location.

To estimate the likelihood of attacker success, we assume a strong adversary endowed with these three capabilities and we want to estimate the number of adversarial APs it must deploy to fool our co-presence detection mechanism. We take a location signature Sig_{loc} computed from $room_i$ and a Sig_t computed from $room_{j\neq i}$. Then, we sequentially select (mac_i, p_i) from Sig_{loc} with maximum p_i and add (mac_i, p'_i) to Sig_t. We compute an optimal RSS_i to achieve $p'_i = p_i$. The *RSS* depends on the distance from the adversarial AP to the predicted personal device location. However, the *RSS* value is not a deterministic function of the distance to the AP. *RSS* values actually follow a left skewed Gaussian distribution for a given distance to an AP [30]. Consequently, we add Gaussian noise (left skewed) to the computed optimal *RSS* value before computing p'_i. We repeat the addition process to Sig_t until we achieve $H(Sig_{loc}, Sig_t) < \delta$.

We repeated this simulation for every scan in $room_j$ (600 repetitions) and for every room. It requires 6 APs on average to fool the co-location detection mechanism into inferring it is in a room of the university building while it is in the studio apartment ($room3 \rightarrow room1 : 5.95 \pm 0.95$ / $room3 \rightarrow room2 : 6.02 \pm 1.21$). The reverse process is even more costly as signatures from the university building are longer (cf. Sect. 6.2.2) and it requires 8 APs on average for the attack to succeed ($room1 \rightarrow room3 : 7.93 \pm 1.42$ / $room2 \rightarrow room3 : 7.73 \pm 1.43$). Deploying so many APs is costly and the prediction of the personal device location is a tedious task. Thus, we consider our mechanism resilient to practical adversaries.

We can consider an even stronger, if less realistic, adversary who can plant "bugs" ahead of time in the shared space in the form of adversarial APs close to the smart speaker. During first authorization, the computation of the reference signature Sig_{loc} by the user's personal device will be influenced by these attacker controlled APs since they will have higher *RSS* than legitimate APs. Later, when the user has left the shared space, the attacker can place the adversarial APs close to the personal device thereby fooling it into concluding that it is still co-present with the smart speaker. We assuming that a legitimate AP results in *RSS* of ~-70 dB (~25 m distance [9]) while an adversarial AP can result in a strong *RSS* of ~-35 dB (~1-3 m distance [9]), and that an average Sig_{loc} contains 11 APs (cf. Section 6.2.3). A single adversarial AP would result in $p_i = 0.18$ while 4 APs could result in a $\sum_{i=adv} p_i$ of 0.47. In other words, although such a strong adversary can fool Lux's co-presence at half the cost of a realistic adversary, the logistics of handling four adversarial APs is still likely to be too high to justify the rewards of stealing access to online services for a limited period.

6.2.6 Comparison to prior work. We compare the accuracy of our co-presence detection method to existing techniques that also uses contextual information. Truong et al. [54] studied the efficiency of four sensor modalities (WiFi, Bluetooth, GPS and Audio) for inferring the co-presence of two devices. Their approach relies on extracting features from several sensor measurements collected by two devices. They feed these features to a supervised machine learning technique that decides if two devices are co-located. We use the dataset they collected to compare their technique with ours. The dataset contains 2,303 samples consisting of several records for four sensor modalities (WiFi, Bluetooth, GPS and Audio) recorded by two devices. 49.5% of samples represent co-located devices and 50.5% represent non co-located devices.

The approach from Truong et al. for inferring co-presence using WiFi is slightly different from ours. They use 10 consecutive WiFi scans recorded during 30 seconds by each of the two devices. In contrast, we compare a signature (extracted from a few consecutive WiFi scans at a point in time) to a single current scan recorded by the same device. To perform a fair comparison, we adapt our technique to extract a signature from the 10 WiFi scans of each device: Sig_{dev1} and Sig_{dev2}. Then, we apply our co-presence detection (Definition 1) on these two signatures. We adjusted the detection threshold $\delta = 0.9$ to optimize the performance in this scenario.

Our results are comparable to those of Truong et al. [54], achieving identical co-presence detection rate (TPR) and a slightly lower (95.7% vs. 98.2%) non-co-presence detection rate (TNR). Truong et al. did not report on the delays incurred while switching detection from co-presence to non-co-presence. As we showed in Section 6.2.4 that our technique eventually reaches 100% TNR in this de-authorization scenario.

On the other hand, our approach is much lighter than Truong et al.'s approach and most prior proposals for co-presence detection using WiFi [35, 41, 57]. It only relies on one WiFi scan collected by a single device to infer co-presence of personal device and smart speaker. The scan and computation takes only 23 ± 5 milliseconds on average in contrast to the 30 seconds of monitoring required each time for the computation of the Truong et al. technique. Consequently, our technique is more suited to the continuous process of detecting co-presence for de-authorization.

7 DISCUSSION

7.1 Potential Alternatives

Lux relies on a custom protocol in order to provide ephemeral authorization. We selected this approach because we wanted to be able to deal with all of the peculiarities of this context (specifically, the lack of a traditional interface) in the protocol itself. A commercial implementation may decide to potentially substitute portions of our approach with other underlying mechanisms. For instance, a smart speaker manufacturer may attempt to extend OAuth 2.0 to achieve the same ends. OAuth 2.0 is a framework, and as such no current implementation does exactly what Lux achieves. However, because OAuth 2.0 is flexible and extensible, it may be possible to provide the same mechanisms through this framework.

Such a decision, like the programming language or message contents ultimately selected to implement a commercial version of Lux, would necessarily be in the hands of the developer.

7.2 Additional Deployment Scenarios

In addition to the scenario presented here, Lux can easily be applied to other contexts. For instance, smart speakers are being built directly into vehicles [19]. This means that travelers could connect their online services for things like navigation and media playback to their rental cars. To apply Lux to this scenario, we only need to relax the requirement for the smart speaker to remain stationary. The vehicle's ability to move would require a different co-location technique than our co-presence detector presented in Section 5. However, unlike hotel rooms that have cleaning and hotel staff entering the shared space routinely, a rental car is more private. This would allow us to use a coarser co-location scheme, such as

cell phone towers since a user is unlikely to remain in the same city after returning their rental car.

Lux could also be useful for individuals who live in multiple dwellings over the course of the year (e.g., students, seasonal workers, or snow birds). These users could potentially have a smart speaker at multiple locations, but not interact with one or more of them for extended periods of time (e.g., all of summer). In this case, Lux could provide additional security for devices while their owners are away. Lux could automatically de-authorize all online services connected to a device except for the root service. This effectively prevents the device from having authorization for any of the user's online services while they are away. However, when the owner returns the device can easily re-authorize all of their online services. This prevents the user from having to re-setup their smart speakers every time they return to one of their dwellings.

We expect that there exist many other scenarios beyond those mentioned in the paper in which Lux would be appropriate to protect users and their credentials.

8 RELATED WORK

Authentication and authorization has been a cornerstone of computer security research since its inception. This has led to numerous different authentication techniques for a wide variety of environments. Techniques such as password [36, 39, 46, 47] and password managers [17, 38, 40] allow users to authenticate to services on the web as well as local devices. Public Key Infrastructure (PKI) allows entities to authenticate to one another via shared trust in a centralized third party [4, 12, 16, 24, 55]. Unfortunately, neither passwords or PKI's are suitable for devices in contested spaces. Both passwords and public keys are intended to be valid for long periods of time and are over privileged for the needs of the device. PKI deployments also requires additional complexity that is not suitable for many IoT devices [12, 24, 25, 48]. Furthermore, all the authentication technique mentioned thus far do not compensate for most IoT device's lack of traditional user interfaces.

To compensate for this, alternative forms of authentication, such as voice biometrics, could be used. Unfortunately, research has shown that it is trivial to compromise voice biometrics via machine learning [1, 2, 13, 49, 56] or replay attacks [27, 45]. Another alternative is zero effort authentication techniques. These techniques often make use of secondary device carried on the user in order to perform authentication. Zero Interaction Authentication (ZIA) [10] or wireless entry present on vehicles are good examples of zero effort systems. Generally, these kinds of systems rely on the secondary device to present an authentication token to the terminal device (i.e., the smart speaker) [6, 31]. These techniques rely on the limited range of their communication medium in order to limit locality of the user. However, zero efforts systems have been shown to be vulnerable to both replay [27] and relay [14, 20, 32] attacks. These kinds of attacks are trivial in nature since an adversary needs to only relay or replay a single signal. In contrast, our technique incorporates all access points around the device, increase the complexity and amount of hardware necessary for an adversary to attack our system.

OAuth 2.0 is an authorization framework that is capable of fulfilling several of the goals and requirements laid out in Section 2.3 [23]. However, it fails to achieve **G1** and **R3** which are both fundamental to Lux. Without these two properties any authorization system for temporary IoT devices will be over-privileged while the user is away and will be vulnerable during the initial setup of the user's visit. Without extensive modification, OAuth 2.0 would not be able to provide either of these two properties. We believe that the problem of temporary device authorization for smart speaker's is different enough from temporary authorization on traditional devices to warrant its own protocol.

For an authentication technique to be effective in a shared location, it must be aware of the user's comings and goings. In order to satisfy this, we need to have some form of location or distance bounding on the user. As discussed previously with zero effort system, simply using radio signal strength as a metric of distance is easily compromised by relay [15, 32], degrading [26], or spoofing [52, 53] the sent signal. Other distance bounding protocols use specialized radio (e.g., Ultra-Wide Band, GPS) or high frequency messaging in order to build an estimate of the distance between two devices [22, 28, 44]. However, both these techniques require either additional hardware [2] or high signaling rates that require high power usage. Both of these requirements make more traditional distance bounding technique ill-suited for deployment and use in smart speaker devices. In this paper we have presented our alternative technique to both authentication and location bounding with the limitations of modern smart speaker devices in mind.

9 CONCLUSION

Smart speaks serve as a natural hub for IoT-enabled homes – they provide a simple and intuitive interface and a wealth of communication interfaces to ensure that nearly any device can connect to them. Such utility has not gone unnoticed, and smart devices are increasingly being deployed in settings where users may no longer reasonably be expected to own them but secure interaction is still necessary. We address this emerging challenge with our system Lux. Our system provides ephemeral authorization protocols that allow users to grant fine-grained access to services such as Netflix and Spotify without requiring them to use their long-term secrets. Key in addressing this challenge is recognizing the limitation inherent to such devices - their lack of displays. Accordingly, Lux develops a novel LED/light-based protocol to both allow users to correctly identify the device to which they are connecting and prevent the user's accounts from being exposed once they leave that location. We demonstrate the advantages of Lux from both a formal and experimental perspective, showing that our proposed techniques represent a practical method for allowing users to connect to such devices.

10 ACKNOWLEDGMENTS

This work was supported by the Nation Science Foundation and the Academy of Finland through the SELIoT project (CNS-1702879/grant 309994). We would like to acknowledge partial support of this work by Business Finland through the 5G-FORCE project (6514/31/2018).

[2]No smart speaker device that we are aware of contains GPS or Ultra-Wide Band radios.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Academy of Finland, or Business Finland.

REFERENCES

[1] 2017. Adobe demos "photoshop for audio," lets you edit speech as easily as text. https://arstechnica.com/information-technology/2016/11/adobe-voco-photoshop-for-audio-speech-editing/.

[2] 2017. LyreBird. https://github.com/logant/Lyrebird.

[3] 3DB Technologies. [n.d.]. Proximity based access control. http://www.3db-technologies.com/en/Home.1.html. Online; accessed 7 August 2018.

[4] Carlisle Adams, Stephen Farrell, Tomi Kause, and Tero Mononen. 2005. *Internet X. 509 public key infrastructure certificate management protocol (CMP).* Technical Report.

[5] Bruno Blanchet. 2018. ProVerif: Cryptographic protocol verifier in the formal model. http://www.proverif.ens.fr/.

[6] Logan Blue, Hadi Abdullah, Luis Vargas, and Patrick Traynor. 2018. 2MA: Verifying Voice Commands via Two Microphone Authentication. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 89–100.

[7] Stefan Brands and David Chaum. 1993. Distance-bounding protocols. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 344–359.

[8] L.L. Cam and G.L. Yang. 2000. *Asymptotics in Statistics: Some Basic Concepts*. Springer.

[9] Cisco. 2014. Best Practices: Location-Aware WLAN Design Considerations. In *Wi-Fi Location-Based Services 4.1 Design Guide*, Cisco (Ed.). Cisco, Chapter 5, 5–1–5–70.

[10] Mark D. Corner and Brain D. Noble. 2002. Zero-Interaction Authentication. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom '02)*. ACM, New York, NY, USA, 11.

[11] Dani Deahl. 2018. Hilton is adding smart home features to its hotel rooms. https://www.theverge.com/2017/12/7/16748588/hilton-honors-smart-home-features-hotel-rooms.

[12] Carl Ellison and Bruce Schneier. 2000. Ten risks of PKI: What you're not being told about public key infrastructure. *Comput Secur J* 16, 1 (2000), 1–7.

[13] Marcos Faundez-Zanuy. 2004. On the vulnerability of biometric security systems. *IEEE Aerospace and Electronic Systems Magazine* 19, 6 (2004), 3–8.

[14] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2011. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Eidgenössische Technische Hochschule Zürich, Department of Computer Science.

[15] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 2010. Practical NFC peer-to-peer relay attack using mobile phones. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 35–49.

[16] Slava Galperin, Stefan Santesson, Michael Myers, Ambarish Malpani, and Carlisle Adams. 2013. X. 509 Internet public key infrastructure online certificate status protocol-OCSP. (2013).

[17] Paolo Gasti and Kasper B Rasmussen. 2012. On the security of password manager database formats. In *European Symposium on Research in Computer Security*. Springer, 770–787.

[18] Google. 2018. Set up your Google Home device. https://support.google.com/googlehome/answer/7029485.

[19] Volvo Car Group. 2018. Volvo Cars to embed Google Assistant, Google Play Store and Google Maps in next-generation infotainment system. https://www.media.volvocars.com/global/en-gb/media/pressreleases/228639.

[20] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. 2012. *Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data.*

[21] Richard W Hamming. 1950. Error detecting and error correcting codes. *Bell System technical journal* 29, 2 (1950), 147–160.

[22] Gerhard P Hancke and Markus G Kuhn. 2005. An RFID distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*. IEEE, 67–73.

[23] Dick Hardt. 2012. *The OAuth 2.0 authorization framework.* Technical Report.

[24] Russell Housley, Warwick Ford, William Polk, and David Solo. 1998. *Internet X. 509 public key infrastructure certificate and CRL profile.* Technical Report.

[25] Russell Housley, William Polk, Warwick Ford, and David Solo. 2002. *Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile.* Technical Report.

[26] Hui Hu and Na Wei. 2009. A study of GPS jamming and anti-jamming. In *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on*, Vol. 1. IEEE, 388–391.

[27] Otto Huhta, Prakash Shrestha, Swapnil Udar, Mika Juuti, Nitesh Saxena, and N Asokan. 2015. Pitfalls in Designing Zero-Effort Deauthentication: Opportunistic Human Observation Attacks. *arXiv preprint arXiv:1505.05779* (2015).

[28] Laymon Scott Humphries and Huey-Jiun Ngo. 2007. Method and system for tracked device location and route adherence via geofencing. US Patent 7,164,986.

[29] Google Inc. 2018. Android Socket Documentation. https://developer.android.com/reference/java/net/SocketOptions.html.

[30] Kamol Kaemarungsi and Prashant Krishnamurthy. 2004. Properties of indoor received signal strength for WLAN location fingerprinting. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*. IEEE, 14–23.

[31] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound.. In *USENIX Security Symposium*. 483–498.

[32] Z. Kfir and A. Wool. 2005. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*.

[33] Md Sakib Nizam Khan, Samuel Marchal, Sonja Buchegger, and N Asokan. 2018. chownIoT: enhancing IoT privacy by automated handling of ownership change. In *IFIP International Summer School on Privacy and Identity Management*. 205–221.

[34] John Koetsier. 2018. Smart Speaker Penetration Just Exploded 50% In 3 Short Months; Amazon & Google Are Winning. Forbes online.

[35] John Krumm and Ken Hinckley. 2004. The nearme wireless proximity server. In *International Conference on Ubiquitous Computing*. Springer, 283–300.

[36] Leslie Lamport. 1981. Password authentication with insecure communication. *Commun. ACM* 24, 11 (1981), 770–772.

[37] Albert Levi, Erhan Çetintaş, Murat Aydos, Cetin Kaya Koç, and M Ufuk Çağlayan. 2004. Relay attacks on bluetooth authentication and solutions. In *International Symposium on Computer and Information Sciences*. Springer, 278–288.

[38] Zhiwei Li, Warren He, Devdatta Akhawe, and Dawn Song. 2014. The Emperor's New Password Manager: Security Analysis of Web-based Password Managers.. In *USENIX Security Symposium*. 465–479.

[39] Chun-Li Lin, Hung-Min Sun, and Tzonelih Hwang. 2001. Attacks and solutions on strong-password authentication. *IEICE transactions on communications* 84, 9 (2001), 2622–2627.

[40] LogMeIn. 2018. LastPass***. https://www.lastpass.com/business-password-manager.

[41] Suhas Mathur, Robert Miller, Alexander Varshavsky, Wade Trappe, and Narayan Mandayam. 2011. Proximate: proximity-based secure pairing using ambient wireless signals. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 211–224.

[42] MATRIX. 2018. Matrix Voice: Voice Development Board For Everyone. https://www.matrix.one/products/voice.

[43] Markus Miettinen, N Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. 2014. Context-based zero-interaction pairing and key evolution for advanced personal devices. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 880–891.

[44] Pratap Misra and Per Enge. 2006. Global Positioning System: signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press* (2006).

[45] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. 2015. *All Your Voices are Belong to Us: Stealing Voices to Fool Humans and Machines.* 20th European Symposium on Research in Computer Security.

[46] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C Mitchell. 2005. Stronger Password Authentication Using Browser Extensions.. In *USENIX Security Symposium*. Baltimore, MD, USA, 17–32.

[47] Manjula Sandirigama, Akihiro Shimizu, and Matu-Tarow Noda. 2000. Simple and secure password authentication protocol (SAS). *IEICE Transactions on Communications* 83, 6 (2000), 1363–1365.

[48] Jim Schaad, Burt Kaliski, and Russell Housley. 2005. *Additional algorithms and identifiers for RSA cryptography for use in the internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile.* Technical Report.

[49] Babins Shrestha, Maliheh Shirvanian, Prakash Shrestha, and Nitesh Saxena. 2016. The Sounds of the Phones: Dangers of Zero-Effort Second Factor Login based on Ambient Audio. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

[50] Dave Singelee and Bart Preneel. 2005. Location verification using secure distance bounding protocols. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*. IEEE, 1–7.

[51] Mitra Sorrells. 2018. MSC Cruises puts voice-enabled assistants in every cabin of new ship. https://www.phocuswire.com/MSC-Cruises-voice-assistants.

[52] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. 2011. On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 75–86.

[53] Nils O Tippenhauer, Kasper B Rasmussen, Christina Pöpper, and Srdjan Capkun. 2012. iPhone and iPod location spoofing: Attacks on public WLAN-based positioning systems. *Technical report/ETH Zürich, Department of Computer Science* 599 (2012).

[54] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. 2014. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*. IEEE, 163–171.

[55] Steven Tuecke, Von Welch, Doug Engert, Laura Pearlman, and Mary Thompson. 2004. *Internet X. 509 public key infrastructure (PKI) proxy certificate profile*. Technical Report.

[56] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine Noodles: Exploiting the Gap Between Human and Machine Speech Recognition. *11th USENIX Workshop on Offensive Technologies* (2015).

[57] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal De Lara. 2007. Amigo: Proximity-based authentication of mobile devices. In *International Conference on Ubiquitous Computing*. Springer, 253–270.