Smart Locks: Lessons for Securing Commodity Internet of Things Devices

Grant Ho
Derek Leung
Pratyush Mishra
Ashkan Hosseini
Dawn Song
David Wagner

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-11
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-11.html

March 12, 2016

Acknowledgement

We thank Linda Lee and Peter Bailis for assisting us with two of our smart lock case studies, and Kurt Thomas, Michael McCoyd, Frank Li, and John Chuang for insightful discussions and feedback. This work was supported by the Intel Science and Technology Center for Secure Computing (ISTC-SC), AFOSR under MURI award FA9550-12-1-0040, FORCES (Foundations Of Resilient CybEr-Physical Systems) under NSF award numbers CNS-1238959, CNS-1238962, CNS-1239054, CNS-1239166), and generous support from Cisco Systems.

Smart Locks: Lessons for Securing Commodity Internet of Things Devices

Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, David Wagner

UC Berkeley

ABSTRACT

We examine the security of home smart locks: cyber-physical devices that replace traditional door locks with deadbolts that can be electronically controlled by mobile devices or the lock manufacturer's remote servers. We present three categories of attacks against smart locks and analyze the security of five commercially-available locks with respect to these attacks. Our security analysis reveals that flaws in the design, implementation, and interaction models of existing locks can be exploited by several classes of adversaries, granting them capabilities that range from unauthorized home access to irrevocable control of the lock. To guide future development of smart locks and similar Internet of Things devices, we propose several defenses that mitigate the attacks we present. One of these defenses is a novel approach to securely and usably communicate a user's intended actions to smart locks, which we prototype and evaluate. Ultimately, our work takes a first step towards illuminating security challenges in the system design and novel functionality introduced by emerging IoT systems.

1. INTRODUCTION

Growing interest in the Internet of Things has spurred the commoditization of many cyber-physical devices for personal use, such as smart home appliances [29], wearables devices [39], and new car models [27]. These emerging "smart devices" extend their mechanical counterparts by integrating them with electronic components that allow external computer systems to control them. Although this integration enables new functionality, it greatly increases the system's attack surface.

While prior work on the Internet of Things (IoT) has focused on the cryptographic protocols used by these systems, limited work has studied the security implications of common network architectures used by these smart devices and the new modes of user interaction these devices enable [15, 33]. As a first step towards exploring these new security challenges, this paper studies one important class of smart devices: home smart lock systems. These locks replace traditional deadbolts with electronically controllable ones that communicate with a user's smart phone or the lock manufacturer's servers. Not only do these locks use network architectures prevalent in other IoT systems, but like other smart devices, they also offer a host of new features that facilitate new methods of interacting with the device. For example, smart locks have followed a trend seen in newer car models where the device will automatically unlock the door if it infers that a legitimate user intends to enter [27].

To help us understand the security problems posed by these new aspects of IoT systems, we propose a security model for smart locks and analyze five popular commercial systems. We present three categories of attacks on smart locks and show that all existing devices we studied are vulnerable to at least one of these attacks. Our analysis reveals that these attacks are possible primarily because of weaknesses in their system design, rather than implementation bugs specific to any particular smart lock.

Examining the network architectures and access control policies used by smart lock systems, we show that an attacker can evade the revocation mechanisms and access logging procedures used by most devices; this enables an adversary to maintain unauthorized, surreptitious access to a user's home. Next, we assess the security of the automatic unlocking protocols provided by several smart locks. We find that existing protocols can often undesirably unlock the door by accident or in the presence of an adversary. Fundamentally, these vulnerabilities arise because all current systems use insecure mechanisms to capture a user's intended actions.

To guide future development of smart locks and similar IoT devices, we propose several practical defenses against these attacks. We find that using an "eventual consistency" design provides robust revocation and access logging mechanisms, while minimizing the system's dependency on external entities (such as the lock's remote servers); this distributed systems design not only enables the lock to maintain a high level of availability, but it also helps reduce the system's vulnerability to remote compromise by allowing smart locks to forgo direct connection to the Internet. Finally, to eliminate weaknesses in existing automatic unlocking mechanisms, we explore three defenses that maintain the usability benefits of automatic unlocking while offering better security than existing smart locks. While two of these defenses draw on prior work, we propose one novel defense that leverages body area networks. We develop a physical prototype of this defense and present evaluation results that suggest this defense achieves our desired security and usability goals.

Ultimately, our work takes a first step towards illuminating security challenges in the system design and novel functionality introduced by emerging IoT systems, such as smart lock devices. We find that existing smart locks fall short of providing adequate security because of tensions between availability and security that emerge in common IoT network architectures; the new modes of interaction offered by smart devices; and the collection of data on users' daily

	Architecture	Interaction model	Devices	Admin interfaces
Kevo	DGC	touch-to-unlock	smartphone, keyfob	mobile app, website
August	DGC	press button in mobile app; automatic unlocking	smartphone	mobile app
Dana	DGC	press button in mobile app; automatic unlocking	smartphone	mobile app, website
Okidokeys	DGC	press button in mobile app	smartphone	mobile app, website
Lockitron	direct Internet connection	press button in mobile app or web interface	smartphone, website	mobile app, website

Table 1: Summary of the system design and properties for each smart lock. DGC stands for a Device-Gateway-Cloud architecture. The Interaction model column corresponds to the user process for unlocking/locking the door; automatic unlocking works via proximity. The Devices column specifies which kinds of electronic devices can be used to lock/unlock the smart lock. The Admin interfaces column indicates how homeowners can administer their lock.

lives to provide new functionality. Nonetheless, our work presents several practical defenses that enable these new devices to provide not only greater convenience, but also better security than their traditional counterparts.

2. BACKGROUND: SMART LOCK SYSTEMS

To ground our exploration of smart lock systems, we searched on Amazon, Google, eBay, and KickStarter for digital home door locks. We then eliminated all locks which were not shipping, not available for purchase, or did not have the ability to connect with mobile devices or the Internet. We deliberately excluded several digital locks that just replace a traditional deadbolt with a numerical PIN pad; because they lack integration with other computer systems, the security and system design of these PIN code locks differ distinctly from the IoT systems we seek to study. This search process yielded five locks: August [2], Danalock [10], Kevo [20], Okidokeys [30], and Lockitron [25].

2.1 Common Smart Lock Properties

Home smart locks consist of three components: an electronically-augmented deadbolt installed onto an exterior door (Figure 1), a mobile device that can electronically control the lock, and a remote web server. Users can use their mobile devices to control the lock by installing the lock's mobile app, creating an account on the manufacturer's servers, and then pairing their mobile device with the lock using a local wireless channel, such as Bluetooth Low Energy (BLE). Table 1 lists the common properties for each lock we studied.

Architecturally, we found that smart locks use one of two network designs. In the first architecture, shown in Figure 2, smart locks themselves do not have a direct connection to the Internet. Instead, these locks rely on users' mobile phones to act as an Internet "gateway" that relays information to and from the manufacturer's servers whenever the phone enters BLE range of the lock. We call this architecture the Device-Gateway-Cloud (DGC) model.

The second architecture, used only by Lockitron, has a direct Internet connection between the smart lock and the remote server. The lock contains a built-in Wifi modem, which allows homeowners to connect it to their home's Wifi network. In this architecture, users can still operate the smart lock with a mobile device; however, all of the smart lock's communications with the server occur through the lock's Wifi connection, and state updates (such as an updated user permission list) are transmitted through this Internet connection, rather than a local wireless channel, such as BLE.[1]

[1]We were unable to successfully use the BLE functionality on the currently-available "Beta" version of Lockitron. Searching their online forums, we found that other users have encountered this same problem; when we emailed Lockitron's support address and asked about this, we did not receive a response. As a result, we were only able to use Lockitron's Internet-based communication during our analysis.

2.2 Digital Keys

Smart locks allow home owners to grant other users access by issuing them a "digital key", providing greater convenience and fine-grained access control per user. The locks we studied allow home owners to issue digital keys that belong to one of four abstract access levels: owner, resident, recurring guest, or temporary guest. An "owner" key can lock and unlock the smart lock at any time, grant or revoke keys of any access level, and use any other administrative feature the lock provides (such as viewing the lock's access logs). "Resident" keys allow a user to access the home at any time, but these users do not have access to any administrative abilities. "Recurring guest" keys can only be used at fixed time windows set by an owner (e.g., only on weekdays from 3-6pm for a baby sitter). Finally, temporary guest keys provide short-term access (e.g., a 24-hour access window).

When the lock is initially installed, the first user who pairs her mobile device with the lock using BLE automatically receives "owner"-level access to the device. Because Lockitron uses Wifi instead of BLE, after the lock has been installed and connected to a user's home Wifi network, Lockitron allows a user to acquire ownership of the lock by pairing with the lock over this Wifi network. For all the locks we studied, once the lock has locally paired with a smart device, no other device can manually pair with the lock in this manner unless the lock is reset. Instead, the owner can grant access to other users' accounts by looking up their email address or phone number.

2.3 Administrative Interfaces

With the exception of August, the other four smart locks provide a standalone website interface that allow users with an owner key to perform key management operations, such as granting and revoking access to other users. Additionally, three of these locks allow owners to view the lock's access logs remotely through this web interface, as discussed below in Section 2.4.

2.4 Access Logging

All of the locks we studied contain a built-in access logging feature, which can be viewed by users with owner keys. Each time a user interacts with the lock or an owner distributes or revokes a key, the lock generates a record of the action, the user who performed it, and the time. None of

Figure 1: View of one smart lock (Okidokeys) from the interior of the home. All locks augment or replace the interior deadbolt's turn knob with the smart lock's electronically controllable component; however, all locks can be manually operated from the inside and they maintain a traditional keyhole on the exterior of the door.

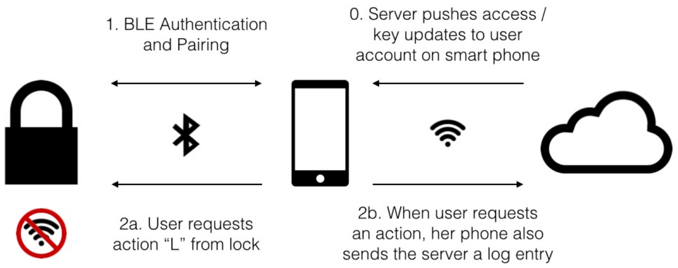
1. BLE Authentication and Pairing
0. Server pushes access / key updates to user account on smart phone
2a. User requests action "L" from lock
2b. When user requests an action, her phone also sends the server a log entry

Figure 2: Internet connectivity model for August, Danalock, Kevo, and Okidokeys. The smart locks themselves do not connect to the Internet. Rather, they connect to a user's phone via BLE and expect the smart phone to be connected to the Internet, where it will be able to push and receive relevant information and updates (such as updates to the lock's software or a new digital key). We call this the DGC model.

the locks provide users with a way to configure which actions get logged.

2.5 Locking and Unlocking Process

In all but the Kevo system, users can unlock their door by pressing a button in the lock's mobile app.

Additionally, several of the locks we studied provide automatic unlocking features for greater user convenience. Both August and Danalock will automatically unlock the door whenever an authorized user gets near the door, sparing the user the need to take out their phone or interact with the mobile app. Kevo has a similar feature, but it works slightly differently: to unlock, the user touches the exterior deadbolt face (which contains a built-in touch sensor); the lock unlocks if an authorized device is in BLE range. Unlike the other four locks, Kevo's mobile app does not have a button to explicitly unlock or lock the door; users must use the touch-to-unlock process. These automatic unlocking schemes raise some of the most interesting security challenges; we discuss them further in Section 3.4.1.

3. SECURITY ANALYSIS

In this section, we propose a security model for smart locks, present our analysis of five popular smart locks under this security model, and systematize the vulnerabilities we discovered by introducing three categories of attacks. Each category corresponds to a fundamental challenge in designing a secure smart lock system. Table 2 summarizes our findings. We have reached out to the vendors of all the systems we studied and are working with them to address the vulnerabilities we discovered.

3.1 Threat Models

In addition to canonical attackers such as network attackers, malware on client devices, and server-side adversaries, we articulate four additional threat models that smart locks might want to protect against. We refer to the legitimate owner as Alice and the attacker as Mallory.

1. A **physically-present attacker** can observe Alice's physical interactions with the smart lock (including accidental ones, such as Alice inadvertently leaving her door unlocked) and can also physically interact with the smart lock at any time. However, this type of attacker does not possess an authorized device and cannot physically alter the lock.
2. A **revoked attacker** possesses legitimate access that Alice gave her, which will be revoked in the near future. For example, consider an apartment or AirBnB tenant whose lease is expiring, or a household worker such as a baby sitter who is being relieved of duty.
3. **Thief:** Mallory steals Alice's authorized device.
4. In a **relay attacker** threat model, Mallory has an accomplice, Michael, one of whom is near Alice and the other is near the smart lock. They both possess a Bluetooth device that can communicate with other Bluetooth-enabled devices, as well as transmit data between the two of them over long distances. Neither of them are authorized to open Alice's lock.

Our security analysis primarily focuses on these four non-traditional adversaries; to a limited extent, we also explore privacy risks posed by server compromise. Because these locks augment conventional deadbolts, we do not analyze purely physical attacks, such as lock picking.

3.2 Security Goals

The primary security goal of smart lock systems is to **prevent unauthorized access**. Specifically, smart locks should only lock or unlock when an authorized user intends for the action to occur. In our analysis of existing systems, we also consider two additional goals: access log integrity (an adversary should not be able to tamper with the access logs or prevent his/her use of the lock from being recorded) and privacy from the lock manufacturer (the lock manufacturer should not learn anything about the lock's usage history).

3.3 State Consistency Attacks

Four out of the five locks we studied use a DGC architecture, where smart locks lack a direct Internet connection to the manufacturer's servers. As a result, these smart locks rely on the user's mobile device for connectivity to the Internet: the only way they can receive state updates from the server is through messages relayed by the user's mobile device.

State consistency attacks exploit this trust model and network design, allowing an attacker to evade revocation and access logging. While it is tempting to trust the user's device to faithfully relay messages between the lock and remote server, we can see that this trust is inappropriate when Alice

	State consistency (§ 3.3)	Unwanted unlocking (§ 3.4)	Privacy leakage (§ 3.5)
August	thief	physically-present attacker, relay attacker	server compromise
Danalock	thief, revoked attacker	physically-present attacker, relay attacker	server compromise
Kevo	thief, revoked attacker	physically-present attacker, relay attacker	server compromise
Lockitron		N/A (no auto-unlock feature)	server compromise
Okidokeys	thief*, revoked attacker*	N/A (no auto-unlock feature)	

Table 2: Summary of vulnerabilities discovered. The three columns correspond to our three classes of attacks. Non-empty cells correspond to an attack we discovered, and list the kinds of attackers that can successfully execute the attack. For state consistency attacks against Okidokeys, in some cases, the adversary only has a short time frame to execute the attack (see Appendix A for more details). Subsequent to our analysis, Okidokeys released a geo-fencing auto-unlock feature like the one used in August and Danalock.

revokes someone's digital key, as this corresponds to marking their device as untrusted. We show how an attacker, Mallory, can prevent Alice from revoking Mallory's access (**revocation evasion**). We also show how Mallory can prevent logging of her interactions with the lock (**access log evasion**) by blocking all packets to the remote server. In practice, these attacks are as simple as switching Mallory's phone into offline mode.

We focus on two types of attackers: a thief who steals Alice's phone[2] and a *revoked attacker* (e.g., a misbehaving tenant whom Alice evicts from an apartment or an ex-spouse).

3.3.1 Revocation Evasion

All smart locks we studied allow owners to revoke other users' access via the lock's mobile or website interface. To revoke a lost or stolen owner's device, they either enable owners to revoke other owners or they provide a "Lost Phone" feature that forcibly logs a particular user's account out on all associated devices. We found that these revocation mechanisms can be evaded by an attacker for every lock that uses a DGC architecture.

3.3.2 Access Log Evasion

As discussed in Section 2.1, the smart locks we studied have a built-in access logging feature. In theory, this allows vigilant owners to detect unauthorized access to their home. However, we found that in most instances where Mallory could evade revocation, she could also ensure that her interactions with the lock were never recorded. This evasion undermines the primary purpose of these logs and may instill a false sense of security in some users.

3.3.3 Case Studies: Danalock and Lockitron

Danalock: We use Danalock as a representative example of state consistency attacks against locks that have a DGC architecture; for details of these attacks against the other DGC architecture locks, see Appendix A.

Danalock allows users of any access level to freely interact with the lock, even when not Internet-connected; this ensures that users will be able to use the device if Danalock's servers are unreachable (e.g., Internet or cellular outage). However, this also means that both a thief and a *revoked attacker* (e.g., Airbnb tenant or ex-spouse who previously had legitimate access) can evade revocation by simply switching the malicious phone to airplane mode. In Danalock and other locks that use a DGC architecture, key revocation works by having the remote server push a revocation message to the revoked user's phone; however, if the phone is offline, then the server cannot push this information to phone and the lock remains unaware of the revocation. Furthermore, we also discovered that even if a legitimate user interacts with the lock with a different device after issuing the revocation, the offline phone maintains access to the smart lock: the server will not push revocation updates to the lock via other devices. Finally, we found that both the mobile app and website interface for Danalock display a confirmation message that indicated a successful revocation, even when a revoked phone maintained access via a state consistency attack. This insecure UI design might lead users to believe that the revocation succeeded, when in fact the thief or *revoked attacker* still has access.

Similarly for access log evasion, because Danalock relies on users' devices to faithfully communicate their actions to the remote server, an attacker who blocks the app's packets from reaching the server (e.g., by taking the phone offline) can prevent her interactions with the lock from being recorded. Additionally, we found that even if other users subsequently use the lock, the attacker's interactions will not be updated in any log viewable by a legitimate user because the lock itself does not store and push log entries.

Lockitron: Lockitron devices use an embedded Wifi modem to connect directly to Lockitron's servers. Each time a user interacts with the lock, the user's mobile app contacts Lockitron's servers, which check whether the user is authorized to perform the request. If the user is authorized, Lockitron's servers push the request directly to the lock via a long-lived TCP connection that the lock established during its initial installation and setup. This design means that legitimate users might be locked out if the Lockitron servers are unavailable or unreachable. However, it has the advantage that neither a revoked attacker nor a thief can evade revocation since the server (which contains the authoritative access control list) will process all interaction requests before the smart lock receives them. Additionally, this direct connectivity architecture enables Lockitron to prevent access log evasion because the lock can independently transmit all interactions to Lockitron's servers. We discuss several of the trade-offs between using a DGC architecture versus direct connectivity in Section 4.1.

[2]Transportation and navigation apps may store Alice's home address, social media apps may contain Alice's address in her user profile, messaging/email services may contain invitations or directions to Alice's home, and finally, some smart lock apps display the location of the user's home within the app. Any of this information can be used by a thief to identify the location of Alice's home.

3.4 Unwanted Unlocking

August, Danalock, and Kevo provide some form of automatic door unlocking. In these interaction models, whenever a device with the correct access permissions enters BLE communication range of the smart lock, the door will unlock automatically. While this interaction model greatly improves the usability of lock systems, we found that all existing locks that provide this functionality can undesirably unlock the door by accident, allowing a physically-present attacker to gain unauthorized access.[3] Furthermore, prior work has shown that relay attackers can exploit similar auto-unlock mechanisms in car systems to gain unauthorized access [16].

3.4.1 Unintentional Unlocking

August and Danalock: August and Danalock both use location services on the user's phone to determine when to auto-unlock the door. For this feature, the homeowner enters the location of her home/smart lock in the lock's mobile app; the app then uses geo-fencing to establish a 50 meter radius around the smart lock. If Alice exits and then subsequently re-enters this boundary, the lock will automatically unlock her door once she gets close enough to establish BLE communication with the lock. Once the door has been automatically unlocked in this way, the auto-unlock feature remains dormant until Alice exits this radius again.

This geo-fencing design implicitly assumes that when Alice leaves and returns home, she will re-enter her home through her smart lock door; however, this will not always be the case, as many homes have multiple entrances. Suppose Alice installs a smart lock on her front door, but she also enters and exits her home from the garage where her car is parked. If Alice leaves through the garage and drives to work in the morning, she will exit the geo-fence boundary. When Alice returns home from work and parks in the garage, her front door will automatically unlock when she enters BLE range (up to 10 meters away) because she has re-entered the geo-fence boundary. Searching online, we found several reviews and user complaints that reported instances of this vulnerability and expressed a desire for manufacturers to securely provide this auto-unlocking functionality [26]. Likewise, if Alice lives in a large home with many roommates and several entrances with smart locks, entering through one entrance will automatically unlock all the other doors as Alice moves around her house (causing her smart phone to enter BLE range of the other locks). This leaves Alice's home open to theft and unwanted intrusion.

We verified the practicality of these attacks by installing August on a door in the back of one author's apartment. The author exited the geo-fence boundary and re-entered the apartment through the front door. We repeated this procedure for ten trials and August automatically unlocked the back door each time.

Kevo: Kevo uses a touch-to-unlock model for its automatic unlocking mechanism. To unlock her lock, Alice taps the deadbolt face when she wants to open her door from the outside. The Kevo deadbolt face is augmented with a capacitive touch sensor. If the lock registers a touch and detects that an authorized device is within BLE range, it toggles the lock from locked to unlocked or vice versa.

Since Bluetooth has a communication range of 10 meters, this interaction model might leave Kevo locks vulnerable to "side-of-the-door attacks", where a physically-present attacker attempts to gain unauthorized access by touching the lock while an authorized device is located inside the home. For instance, consider a neighborhood burglar who gains access because Alice or a housemate left her smart phone or key fob at home. Or, Mallory might be a disgruntled acquaintance or unwelcome visitor who gains access while Alice is at home. In order to prevent this kind of attack, Kevo uses a proprietary algorithm based on Bluetooth directional sensing to detect if the authorized device is inside or outside of the home and will only unlock if an authorized device is detected to be outside the house. We found that Kevo's side-of-the-door detection algorithm seems to work well in practice for most home layouts. For our experiment, we installed Kevo on a standard (steel) apartment door. We then placed an authorized device at six different locations inside the apartment: five feet and ten feet directly behind the lock, and five feet and ten feet behind the lock at forty-five degree angles to the left and right. For each location, we tried ten times to unlock the door from outside; across all sixty trials, Kevo correctly rejected access.

However, we found that Kevo did not effectively prevent side-of-the-door attacks in homes with a "concave" door layout, where part of the home extends past the Kevo-protected door; see Figure 3 for an illustration. We tested this layout by placing an authorized device in a room that was approximately fifteen feet to the side of the door and ten feet in front of it. The attack succeeded in all ten of our trials. Thus, if a home has a concave layout and an authorized user has left her smartphone or keyfob in the room that Kevo thinks is "outside" the house, a burglar would be able to enter the house simply by tapping on the lock's exterior to unlock the door. This kind of attack requires no sophistication, and it would be easy for burglars or intruders to recognize houses that might be vulnerable (given Kevo's distinctive exterior) and try to gain access.

Discussion: For all three locks, these vulnerabilities represent important violations of home security. Each might enable theft, physical intrusion, or dangerous physical confrontations with an intruder. Even if the worst never happens, door locks serve not only as a physical protection mechanism but also help users feel comfortable and safe: the presence of a locked door provides an emotional sense of security. Thus, even if a homeowner discovers an unintentionally unlocked door before an attacker can physically capitalize on it, many homeowners might feel a violation of their sense of security. Because the unwanted unlocking arises due to shortcomings of the technology rather than user mistakes (such as forgetting to lock the door), these vulnerabilities could lead to loss of confidence and trust in the system. Therefore, in addition to enabling physically-present attackers to gain unauthorized access, unintended unlocking vulnerabilities are significant because they violate users' trust in the system and their sense of physical safety.

3.4.2 Relay Attacks

Relay attacks also pose a risk to these auto-unlock mechanisms. While we did not acquire attack hardware to physically test whether these smart locks are vulnerable to these

[3]This attack does not apply to Lockitron and Okidokeys because they did not have an auto-unlock feature at the time of our analysis. Subsequent to our analysis, Okidokeys released a geo-fencing auto-unlock feature like the one used in August and Danalock.

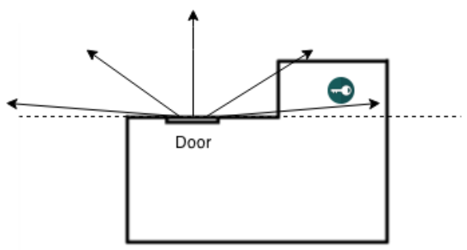

Figure 3: A concave home layout, where Kevo's side-of-the-door defense does not work as desired. The arrows emanating from the door represent regions that Kevo believes to be "outside of the home".

attacks, numerous prior papers have demonstrated the practicality of relay attacks against analogous auto-unlock protocols in cars [12, 16, 17, 21] and Bluetooth authentication protocols [24]. However, as we discuss later in Section 5.3, known defenses against relay attacks require new hardware, which existing smart phones do not possess [7]. Thus, unlike all the other attacks we discuss in this paper, we did not physically verify the relay attacks discussed below, but rather rely on prior work that demonstrates their feasibility and the lack of defensive hardware in current mobile devices. We discuss relay attacks because like the *Unintentional Unlocking* vulnerabilities discussed above, the root cause of relay attacks is the lack of a robust and secure mechanism for capturing a user's intended actions; we discuss ways of addressing this underlying problem in Section 5.

Based on prior work, we believe a relay attacker could gain unauthorized access against a Kevo lock as follows: When Alice is away from her home, Mallory could follow Alice from a distance of several meters while her accomplice, Michael, taps Kevo's deadbolt face to begin the touch-to-unlock procedure. Using his Bluetooth relay device, Michael captures the lock's Bluetooth authentication challenge message and relays it to Mallory (e.g., over Wifi). Upon receiving the relayed challenge from Michael, Mallory's device broadcasts it, captures the response from Alice's device, and relays the response back to Michael's device, which broadcasts this response to the smart lock. As with standard relay attacks, because the lock communicates with a truly authorized device, Michael can gain unauthorized access to Alice's home.

The geo-fencing design used by Danalock and August makes relay attacks more difficult, but does not entirely prevent them. In particular, Mallory and Michael will need to both conduct a Bluetooth relay attack against Alice and the smart lock, and spoof a false geo-location onto Alice's phone. Based on Android source code analysis of the Danalock and August apps, we found that these apps query passive geo-location (such as WLAN-based positioning, based on nearby Wifi network id's) once every few minutes and occasionally use GPS to confirm this location if passive geo-location is unreliable. Unfortunately, several prior papers have shown that a relay attacker (with no software on Alice's phone) can successfully spoof the location on Alice's phone, regardless of either of these two geo-location methods [28, 35, 36]. Thus, a relay attacker can still gain unauthorized access by first conducting a location spoofing attack on Alice's phone and then proceeding with a regular relay attack on Alice's phone and the smart lock.

3.5 Privacy Leakage

Although access logs help owners identify if unauthorized access has occurred, they raise a number of privacy and social relationship questions. While recent work by Ur et al. [38] explores complex privacy and trust issues in teen-parent relationships that result from home surveillance systems, our analysis of smart lock access logs specifically considers privacy against the lock manufacturer's servers. In particular, we consider a smart lock to provide privacy from the server if the servers cannot view a home's access logs without shipping updates to users' mobile apps that exfiltrate the logs and without compromising a homeowner's login credentials. This threat model corresponds to a number of practical scenarios: first, this notion of privacy prevents an insider (e.g., employee of the lock company) from abusing her access to the servers to learn private information about other users (e.g., to stalk an ex-partner). Second, this level of privacy helps insulate the lock vendor from supplying the access logs of all its users in response to government issued requests; for example, recently the New York Attorney General's office issued a subpoena for all of AirBnB's hosts in New York [19]. Finally, defending against this threat model provides some protection against an attacker who has compromised the remote server. If the attacker remains in the system for a limited amount of time and does not push malicious code to the clients (a more overt and risky action that can later be reversed with a benign patch), then the attacker will not be able to arbitrarily view the access logs for all users who have purchased the lock.

To understand what information lock servers see, we inspected data sent from their apps to the server by conducting active man-in-the-middle (MITM) attacks on interactions with our own locks. No lock's app, except for August, pins their server's certificate or public key. Because we could not view August's network traffic, we also decompiled the Android APKs for all five locks and inspected the access logging source code of these apps.

We found that only Okidokeys provides privacy against the server. The servers for all other locks receive access logs in plaintext form. Although these systems use TLS to encrypt access logs while in transit to the server, the server receives the access logs in unencrypted form.

The Okidokeys lock provides privacy against the server because it only stores access logs locally on the lock. To view the logs, users must be within BLE range of the lock; there is no way to remotely view the logs through the mobile app or website. However, this has a cost: for instance, parents cannot check the audit logs from work to see whether their child has returned home.[4] In Section 4.2 we show how to resolve this tension by providing privacy against the server while allowing users to remotely view their home's access logs. Overall, our analysis shows that like many IoT devices, most smart locks have followed the trend of accruing formerly private information about users' physical behaviors and social dynamics. This underscores the need for a dis-

[4]Subsequent to our analysis, Okidokeys released a Push notification feature that allows owners to receive a real-time notification whenever a user interacts with her lock. This mechanism could be securely implemented to maintain privacy if the lock and owner's phone share a symmetric key during their initial pairing (the server must not know this key); push notifications could then be encrypted using this symmetric key.

cussion on the nature of privacy in an Internet of Things world and the development of techniques to meet both the privacy and functionality demands of users.

4. DEFENSES

In this section we propose defenses against two of the three attack categories listed earlier. We defer discussion of the remaining category, Unwanted Unlocking, to Section 5.

4.1 Mitigating State Consistency Attacks

Comparing the security of smart locks against traditional mechanical locks, it might seem like even when smart locks are vulnerable to state consistency attacks, they offer security on par with their traditional counterparts. For most homes, keys do not have variable access levels, so the only way to revoke a malicious key is to re-key or replace the lock. Additionally, traditional door locks do not have any access logging at all. However, because smart locks provide features like access logs and digital revocation of users, the average user might have different expectations for her lock's security and functionality. For example, when a user revokes an attacker's access and the lock displays a confirmation to the user, an average person would expect that the lock successfully revoked the attacker. Similarly, when a smart lock's access logs do not show any signs that an unauthorized user has interacted with the smart lock, a homeowner might rationally believe that a revoked or unauthorized device hasn't been used to access the user's home, providing a false sense of security and potentially incorrect belief of who has accessed the user's home.

State consistency attacks allow attackers to violate these (rational) expectations of security that users have, based on the functionality and confirmation messages provided by smart locks. Should such an attack happen, even though an analogous physical attack might have occurred with a traditional lock, these expectations may cause users to feel as if their trust and expectations in the system have been violated and that the smart lock ultimately failed to provide adequate security. As such, it is important that smart locks provide robust defenses to these kinds of attacks, particularly since several of the defenses we discuss do not require any hardware changes to the lock and do not require direct Internet connectivity on the lock.

For smart locks that follow a DGC architecture, state consistency attacks fundamentally arise because they are distributed systems, and their design does not provide consistency in the face of network partitions. Recall the CAP Theorem for distributed systems: it states that if network partitions can occur, it is impossible to provide full availability for the system's service, while simultaneously maintaining the latest, consistent state across all nodes in the system [8]. Thus, no distributed system can provide perfect consistency and availability in the face of partitions.

In the context of smart lock systems, we can consider the smart lock, the lock server, and each user's mobile device to constitute the nodes of a distributed system; the lock's access control list and access logs constitute the important, security-related state that the system seeks to keep consistent. For most normal smart lock usage in a DGC architecture, the user's mobile device will establish an ephemeral edge in the network that connects the lock and server; in this case no partitioning happens and the lock and server can synchronize state (consistency) and allow all authorized lock actions (availability). However, when a user's device does not connect to the lock's servers and tries to interact with the smart lock, the system suffers from a partitioning between the lock and server, and the smart lock must choose between allowing interactions (availability) and rejecting requests from the user until the phone can connect to the server and receive updates (consistency). Because partitioning can happen for a number of benign reasons (cellular outage, lock server outage, etc.), the correct choice between availability and consistency is not always clear.

Eventual Consistency: To mitigate state consistency attacks, we advocate for an eventual consistency model for updating security-critical state, such as access control lists, and for deciding when to allow access to users in the presence of server unavailability.

The DGC architecture can support eventual consistency. We assume that the lock and server share a long-term symmetric key, set up when the lock is initially installed, which can be used to provide end-to-end secure communication between the two. The server stores the authoritative copy of the access control state. Each time a user interacts with the lock, the user's mobile app fetches a signed, updated access list from the server and sends it the lock; to prevent replay of old access lists, the signed update message should include an incrementing version number or timestamp. Upon verifying that the updated list came from the server and is fresh, the lock updates its access list accordingly. If the lock does not receive a valid update from the server, it does not modify its current access control list; however, it does allow the current user to perform smart lock actions consistent with the user's permissions on the lock's current access control list.

Against revocation evasion attacks, we see that once an owner revokes an attacker's access, this design only allows a thief or revoked attacker to maintain unauthorized access so long as no legitimate user uses the lock. As soon as any honest user (not just the owner) interacts with the lock, the server will be able to update the lock with the new access control list that revokes the attacker's access. Thus, Alice or a housemate can immediately return home to ensure timely revocation (giving the attacker a very small window to physically beat Alice to her home); or, if she does nothing, the thief will lose access once Alice or any other housemate uses the lock again in their normal routine (in most cases, giving the attacker only a few hours until end of day to gain unauthorized access).

Okidokeys follows some elements of this model, but falls short in key respects. Crucially, Okidokeys updates the lock's state only if an owner requests a manual sync. In contrast, our design automatically updates the lock's state during each interaction with any Internet-connected user (regardless of the user's access level), which we expect will be more effective.

Eventual consistency can also mitigate access log evasion. The lock will have the authoritative copy of the latest log entries, and it will eagerly push them to the server whenever possible. Every time the lock executes an action, it increments a monotonically increasing sequence number, appends the action and sequence number to its local queue of unacknowledged entries, and attempts to push a signed copy of all entries in its queue to the server via the mobile device currently interacting with it. The server responds with a signed acknowledgment of the highest sequence number it has received. When the lock receives a valid acknowledgment, it clears all entries up to the last acknowledged entry, retaining all unacknowledged entries. In this scheme, every user that interacts with the lock will attempt to push new events to the lock's global access logs on the server. As soon as *any* honest user interacts with the lock, all access events will reach the server and be available for the lock's owners to view. Thus, an attacker can only hide her lock interactions for a limited amount of time. Furthermore, the amount of state the lock needs to store and transmit per usage will be very small since the lock only needs to queue events when a malicious access event occurs, or when a legitimate user does not have Internet connectivity.

In conclusion, eventual consistency provides a good balance between robust availability and quickly ensuring the smart lock upholds an average user's security expectations. In this model, the lock and server are guaranteed to receive new security-relevant state as soon as any Internet-connected, honest user interacts with the lock—allowing smart locks to offer more convenient and more secure key management and intrusion detection than traditional locks.

Direct Connectivity: An alternative approach is to ensure that the smart lock is directly connected to the Internet, as Lockitron does. The server can then hold the authoritative copy of the state and communicate it directly to the lock. However, this approach has a number of trade-offs that have led many locks and other IoT devices to adopt a DGC architecture. Economically, the direct connectivity model requires adding additional hardware, which increases the integration and manufacturing costs of the device. Furthermore, directly connecting these devices to the Internet increases their vulnerability to large-scale remote compromise. Whereas a direct connectivity architecture could allow remote adversaries to directly access and exploit vulnerabilities in IoT devices, a DGC architecture requires an attacker to compromise and control a user's gateway device before being able to infect IoT devices like smart locks. Additionally, incorporating and using an embedded Wifi modem consumes more power from IoT devices. Lockitron attempts to address this problem by switching the lock into hibernation after it has been idle for a short time. When a user wants to subsequently use Lockitron, she must knock on the door to active a sensor in the lock that will wake the device from hibernation. While this conserves battery, it leads to longer wait times for the smart lock to unlock the door and therefore worse usability. Finally, even for locks that use a direct connectivity model, we recommend that they adopt an eventual consistency policy: whenever a user interacts with the lock, the lock should poll the server for updates; but if the lock cannot establish a connection, it should use its current access control list to make decisions about whether to grant the user's request or not. This will ensure that users will not get locked out of their home in the event of a variety of common, but benign instances where the lock cannot contact the server: the user's home Internet connection fails (e.g., a power or Internet outage) or if the smart lock's servers are temporarily unavailable (e.g., server crash) or permanently down (e.g., the manufacturer goes out of business).

4.2 Limiting Privacy Leakage

As Section 3.5 explains, Okidokeys provides access log privacy against the lock's servers but provides limited capabilities to remotely view the lock's access logs; the other locks allow remote viewing but do not provide privacy against their servers. We show how to provide privacy against the lock's servers while allowing remote access log viewing.

Our threat model considers an adversary at the remote server who has occasional access to content seen by the server, but cannot push updates or modify application code sent to the user's mobile app or web browser. This corresponds to several practical, server-side adversaries discussed earlier (§ 3.5).

To provide both this level of security and our desired functionality, we propose the following design:

1. During installation/reset, the lock generates a long-lived symmetric encryption key, which we call the "privacy key."
2. Every time the lock records a log entry, it encrypts the entry with its privacy key using a standard authenticated encryption scheme; thus, the server only sees encrypted log entries but not the decryption key.
3. When the lock interacts with an owner's mobile device, the lock checks to see if the app has a copy of the privacy key; if not, the lock transmits the privacy key to the mobile device, which the app then stores, encrypted using a fresh key derived from the user's password (using a suitable slow hash). Note that since this interaction occurs over a local, secure wireless channel such as BLE, the server never sees the privacy key when it's transferred from the lock to an owner's mobile device.
4. Finally, to support access log viewing from the lock's website interface, the mobile app uploads the encrypted privacy key to the server, which stores it with the other account information for that user.
5. Whenever Alice wants to view her lock's access logs from her mobile app, the app can decrypt using her account's password to recover the privacy key and then decrypt each log entry. The web interface can similarly use client-side Javascript to perform the same operations locally on Alice's machine, letting Alice view the access logs in her browser.

This simple scheme provides privacy against the server because the server only sees encrypted log entries and never learns the decryption key. While a server compromise could enable the adversary to brute-force Alice's account password and subsequently obtain her lock's privacy key, any adversary who learns Alice's password can simply login to her account and view the access logs anyways. Thus, this design provides full functionality while still providing relatively strong privacy against the server.

5. SECURE AND USABLE INTENT COMMUNICATION

So far, we have proposed defenses against two of the three attack categories. Defending against Unwanted Unlocking is more challenging. The Unwanted Unlocking attacks we found result from a tension between usability and security. Automatic unlocking seeks to make user interactions as fast and effortless as possible. Unfortunately, as both our work studying smart locks and prior work studying auto-unlocking in cars shows [18], existing auto-unlock mechanisms can often be exploited by an attacker to obtain unauthorized access; in our study, all existing smart locks that provide auto-unlocking functionality are vulnerable to at least one form of Unwanted Unlocking attacks.

Fundamentally, the vulnerabilities in existing auto-unlock mechanisms arise because they try to measure the user's proximity but they do not try to verify the user's intentions: just because a user is near the lock does not imply the user intends to unlock the door. This section explores three approaches for securely and usably conveying a user's intention to a smart lock system.

5.1 Goals and Threat Models

We focus on defending against physically-present attackers and relay attacks. We consider an unlocking intent protocol to be secure if it prevents these two classes of attackers from successfully unlocking the door.

The primary benefit of an auto-unlock feature is a fast, simple interaction model that only requires a user to approach or touch her door to unlock it. Thus, ideally a good mechanism should provide a fast unlock time (small amount of idle time spent waiting for a door to unlock) and a natural interaction model (the process for unlocking the smart lock should not require additional steps beyond approaching and touching the door, which the user needs to do anyway).

5.2 Combining Geo-fencing and Touch-to-unlock

Geo-fencing treats the process of a user leaving and returning home as an indication that the user intends to unlock her door. Unfortunately, as discussed earlier (§ 3.4), this approach can unintentionally unlock the owner's door without her realizing it or intending to.

To make this process more robust, one could combine geo-fencing with Kevo's touch-to-unlock model: the door should unlock only if the user not only exits and re-enters the geo-fence boundary, but also touches the exterior lock face within a set time period after re-entering the boundary.

This protocol has some limitations. It still grants unauthorized access to a physically-present attacker who touches the lock as Alice returns home. If an adversary touches the front door lock as Alice enters through the garage, this protocol will allow the attacker to unlock the door because it does not have a mechanism for verifying whether it was actually Alice that touched the door. Additionally, as discussed in Section 3.4, even though geo-location raises the bar against relay attacks, prior work has shown that an attacker can use location spoofing to trick Alice's phone into believing it has re-entered her home's geo-fence boundary and then conduct a standard relay attack on the door's auto-unlock protocol. Fundamentally, the combination of geo-fencing and touch-to-unlock does not protect against relay attackers because the mechanisms it uses to compute the location of a user/distance from her lock are not designed to be secure against adversarial spoofing.

5.3 Location-Limiting Defenses

Another possibility is to use protocols specifically designed to verify the location of the user more accurately. Securely verifying that a user is only a short distance away from the exterior of her door would stop many attacks. We consider two possible instantiations of this approach.

NFC: One natural approach for verifying that the smart lock and key are within a small distance of each other is to use NFC as the wireless communication channel between the lock and key. NFC is specifically designed for very short-range communication, typically no more than 10 cm [14].

This approach mitigates threats posed by a physically-present attacker, such as unintentionally unlocking the wrong door and side-of-the-door attacks. However, prior literature has shown that NFC is vulnerable to relay attacks [17, 21]. In particular, the small communication distance imposed by NFC results from the short-range signal/field strength generated by NFC devices; however, beyond this practical limitation, NFC protocols do not perform any computation to securely verify that the two devices are actually bounded by a short distance. As a result, prior work has built several prototypes that conduct successful relay attacks against NFC (including some where one attacker can remain several meters away from the victim by using a simple antenna to boost the signal between the attacker's device and the victim's device) [17, 21].

Distance-Bounding Protocols: A more secure approach is to use distance-bounding protocols to verify that the user is very near the lock. In these protocols, the two communicating devices engage in a series of challenge-response steps designed to upper-bound the distance between them. The protocols compute this bound by calculating the round-trip time (RTT) of each challenge-response step [7]. The speed of light makes implementing distance bounding extremely challenging, but existing work has demonstrated the ability to verify distance to within 12cm accuracy using specialized hardware and custom protocols [32].

A lock could use these distance-bounding primitives for secure auto-unlocking. Concretely, a user's device would authenticate to the lock over BLE as normal when it comes within BLE range, then repeatedly run the distance-bounding protocol. Once the lock can verify that the user is within a short distance (e.g., one foot of the door), it runs the side-of-the-door detection algorithm to check that the user is on the outside; if all these checks pass, it automatically unlocks. Unfortunately, BLE currently does not support distance bounding and mobile devices do not currently have the special hardware needed to support distance bounding, but if this support became available in the future, this could be an appealing solution to the attacks we found.

This protocol is secure against relay attacks because its distance-bounding computation will prevent a remote attacker from spoofing the location of an authorized key. It also prevents unintentional unlocking. The short distance bound requires Alice to be very close to her door before it unlocks, and the side-of-the-door detection ensures that the door will only unlock if Alice is on the outside, preventing a scenario where an adversary convinces Alice to come near the inside of her door (e.g., to look out of the peephole to see who's knocking on her door). The protocol provides a natural and convenient interaction model, as the door automatically unlocks as soon as Alice comes close enough. We found that Kevo's side-of-the-door detection algorithm completed instantly in 50 of 50 trials, so this protocol should have extremely fast response times. However, despite achieving all of our security and usability goals, this scheme requires hardware additions to the lock and users' mobile devices to enable distance-bounding, so it cannot be deployed today.

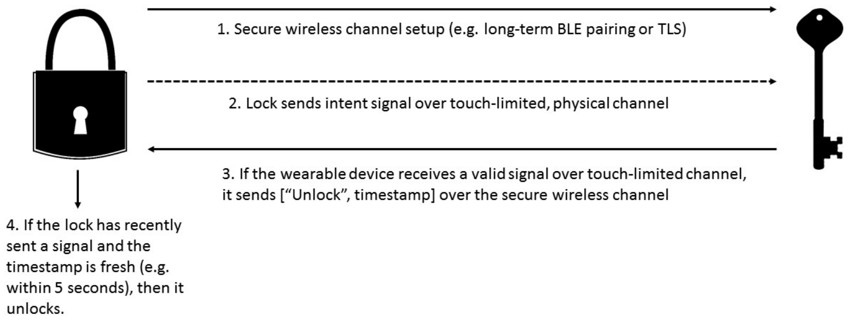
1. Secure wireless channel setup (e.g. long-term BLE pairing or TLS)
2. Lock sends intent signal over touch-limited, physical channel
3. If the wearable device receives a valid signal over touch-limited channel, it sends ["Unlock", timestamp] over the secure wireless channel
4. If the lock has recently sent a signal and the timestamp is fresh (e.g. within 5 seconds), then it unlocks.

Figure 4: The Touch-Based Intent Communication (TBIC) protocol. Solid lines represent communications sent over a secure wireless channel, such as TLS or BLE's secure pairing channel. Dashed lines represent communications sent over a body/touch-limited channel such as bone conduction or capacitive coupling. By withholding the unlock request until the wearable device receives a signal over the touch-limited channel, we require that an attacker engage in physical contact with the user in order to successfully execute an attack.

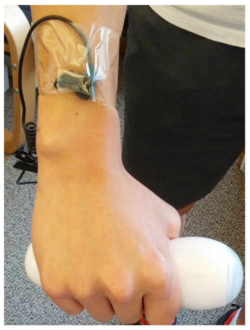

Figure 5: The setup and placement of our bone conduction microphone on a researcher's wrist and the hand-held vibrator we used to transmit the intent signal.

5.4 Touch-Based Intent Communication

A third approach is to use touch to convey intent. When an authorized user touches the exterior deadbolt or grabs the doorknob, this provides a robust indication that the user intends for the door to unlock. We implement this idea by transferring an "intent signal" between the lock and key over a physically-limited data channel, using body-area networking (BAN) [34]. BAN enables multiple devices to transfer data using the human body as the wireless communication medium, creating a touch-limited channel where data only propagates to devices touching the user's body. In particular, we propose the Touch-Based Intent Communication (TBIC) protocol, shown in Figure 4. The core idea is that an authorized device will issue a request to unlock the door if and only if it receives an intent signal from the smart lock over the touch-limited channel.

Our scheme assumes that the user wears a wearable device such as a smart watch, bracelet, or ring. The wearable device needs a wireless radio (e.g., BLE or Wifi) so it can communicate securely with the lock. Additionally, we assume that the wearable device and lock contain hardware that allows the lock to send a message to the wearable device via a physical BAN channel. The TBIC protocol works as follows:

1. When Alice enters wireless communication range with the lock (e.g., BLE communication range), her wearable device and the smart lock establish a secure channel over the wireless link using a long-term key that was exchanged during the initial smart lock installation. For instance, they might use BLE's pairing-based channel or DTLS for the secure channel.

2. To lock or unlock her door, Alice touches the exterior face of the lock.

3. When the lock's capacitive sensor registers a touch, the smart lock will transmit a one-bit intent signal to Alice's wearable device over the physical body-area channel.

4. Upon receiving a valid intent signal over the physical channel, the wearable device will send a timestamped "Unlock" message over the secure wireless channel.

5. When the smart lock receives an "Unlock" message, it checks whether it has recently sent an intent signal (e.g., within the past 5 seconds) and whether the timestamp also falls within that short time window. If all these conditions are met, the smart lock will unlock the door.

The TBIC protocol could be instantiated in a number of different ways, depending upon the body-area network technology used. We are aware of three possible technologies: capacitive coupling, galvanic coupling, and bone conduction. Capacitive and galvanic coupling have been explored extensively in the electrical engineering literature [9, 23, 34] and appear to be plausible candidates for deployment; for instance, they have been standardized in the IEEE 802.15.6 WBAN standards [1]. These two techniques allow data to be transmitted through the human body using electric signals sent between two electrodes on a user's body: capacitive coupling induces differences in electric potential between the two electrodes to transmit data, while galvanic coupling varies the electric current through the body [34]. While work in the early 2000's showed that capacitive and galvanic coupling can achieve data transfer rates of 5–10 Kb/second, more recent work suggests that capacitive coupling can reach speeds of up to 10 Mb/second [34]. Several companies have even presented new devices that use capacitive coupling: for instance, Ericsson's "Connected Me" demo at CES 2012 showed a smartphone playing music to a set of speakers in real time through the human body [13].

In contrast to the established body of literature exploring capacitive and galvanic coupling, there has been little research on bone conduction for BAN [40]. Bone conduction is widely used in hearing aids and specialized headsets to transmit audio to the wearer (as an earphone) and record audio from the wearer (as a microphone) in real time [3], but not for body-area networking. We show below that bone conduction could provide a viable alternative to capacitive and galvanic coupling and present *Vibrato*, a concrete instantiation of TBIC using bone conduction. This lends additional reason to believe that TBIC could be feasible to deploy for securing smart locks and other Internet of Thing devices.

5.4.1 *Vibrato: Touch-Based Intent Communication via Bone Conduction*

Scenario	μ	σ	Unlock Percent
Benign user	498	123	100%
Silence	5.97	4.91	0%
Talking	4.03	2.85	0%
Walking	16.47	13.91	0%
Typing	28.18	13.92	0%
Computer tone	6.39	2.56	0%
Vibrator tone	5.19	2.98	0%
Phone vibration	6.42	0.97	0%
Table surface	83.93	74.47	6%

Table 3: Summary of signal energies at 80 Hz for each of our experiments. The first five scenarios corresponds to normal usage. The last four rows corresponds to several attack scenarios discussed in Section 5.4.1. Signal energies have been scaled down by a factor of 10^8. The last column shows the fraction of trials where *Vibrato* would have unlocked the door, using an energy threshold of 200×10^8.

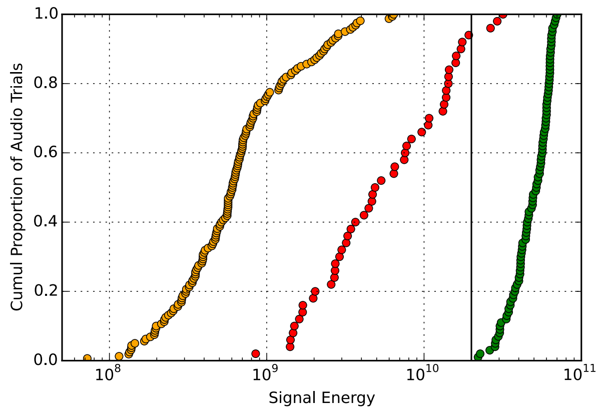

Figure 6: CDF of total signal energy, from our experimental evaluation of *Vibrato*; the x-axis is log-scaled. Green dots (right curve) denote trials for benign usage (the first row of Table 3). Orange dots (left curve) are a composite of the next seven rows of Table 3 (trials for benign non-use and all attacks other than the table surface attack). Red dots (middle curve) denote trials for an attacker who places the vibrator on a table surface that the user is touching. The vertical black line denotes our empirical threshold at 2×10^{10}. Only three of the table-surface attack attempts succeeded; all other attack attempts failed. *Vibrato* correctly unlocked for a benign user in all 100 attempts.

The *Vibrato* protocol is an instantiation of TBIC: the lock transmits an intent signal by vibrating the door handle or lock face, and the user's wearable device uses a bone conduction microphone (touching the user's skin) as a vibration sensor. When the user touches the door handle or lock face, her wearable device will detect the vibration and send an unlock command.

We developed a physical prototype to evaluate *Vibrato* using a $40 earbone conduction microphone and a hand-held, battery-powered massage vibrator. Current wearable devices do not contain a bone conduction microphone, so we modeled the wearable device by taping the bone conduction microphone to one of the researcher's wrists at the location where a watch or bracelet would be positioned; we then used the low-powered vibrator to model the door knob of a smart lock implementing *Vibrato* (see Figure 5). The wearable device would use its bone conduction microphone to record audio and analyze it in real time to detect the intent signal.

In our prototype, the hand-held massager vibrates at 80 Hz. To recognize the intent signal, we bandpass-filtered the audio signal; for each 100 millisecond window, we computed the total signal energy in the range [70 Hz, 90 Hz] by summing the squared amplitudes of the filtered signal. We then compared the window's total energy to a fixed threshold to detect whether an intent signal is present.

Empirical Assessment: We conducted a series of nine experiments to assess the feasibility, usability, and security of our prototype of *Vibrato*. Table 3 summarizes the results of our experiments.

Because *Vibrato* only uses a single bit to indicate a user's intent to unlock (whether the energy exceeds the threshold or not), we need to ensure that everyday activities within BLE range of the lock do not cause *Vibrato* to unintentionally unlock the door. To measure the total energy when an authorized user unlocks the door, we ran 100 trials where a researcher held the vibrator in his hand for three seconds. Then, we ran 20 trials (each three seconds in duration) to measure the total energy for four scenarios corresponding to everyday, non-unlocking usage: standing in silence, talking, walking and waving one's arm, and typing on a keyboard. As Table 3 shows, there is more than an order-of-magnitude difference between the signal strength when touching the vibrator than when not (row 1 vs. rows 2–5); setting the detection threshold at 2×10^{10}, we found that *Vibrato* correctly unlocks the door in 100% of trials where the user touched the mock-lock and correctly does nothing in 100% of the remaining trials.

We also conducted experiments to understand how much security *Vibrato* provides against motivated relay attackers who might try to spoof the intent signal. For instance, during a relay attack, the attacker might try playing a loud tone near Alice, turning on a vibrator near Alice, calling Alice on her phone (causing her phone to vibrate in her hand), or tricking Alice into touching an object or surface that the attacker is vibrating. We conducted four experiments to simulate these attacks: an author stood one foot away from a laptop playing an 80 Hz audio tone at max volume; an author stood one foot away from the hand-held massager turned on (creating a vibrating noise at the target frequency), without touching it; an author held a phone that vibrated at the phone's default vibration speed; and an author touched a table one foot away from the vibrating massager on top of it. We ran 20 trials for the first three scenario and 50 trials for the last scenario since we expected it might be the most powerful attack.

As Table 3 indicates, none of these attacks is effective. There is an order-of-magnitude difference in received total energy between the first three attack scenarios and benign unlocking, and these attacks failed in every case (none of the 60 attack trials exceeded the 2×10^{10} threshold). The table-surface attack was slightly more successful, successfully spoofing the intent signal in 6% of trials. We consider this low success rate acceptable, as this attack can only be executed by an attacker who successfully convinces the user to touch an object vibrating at a specifically targeted frequency while simultaneously conducting a relay attack with a confederate present at the user's home.

Thus, we believe that in addition to capacitive and galvanic coupling techniques, bone conduction can serve as a secure implementation for a TBIC protocol. To attack *Vibrato*, a relay attacker must convince Alice to touch a malicious object that transmits a physical signal identical to her lock, while simultaneously conducting the relay attack. Our experiments suggest that even if an attacker can do this, the success rate will be low: 6%. Furthermore, physically-present attackers cannot gain unauthorized access because Alice's wearable device must receive a physical, touch-limited signal (while in BLE range of the lock) before it unlocks the door.

Our experiments also suggest that *Vibrato* meets our two usability goals for auto-unlock protocols: a natural interaction model and a fast unlock time. Users only need to grab the doorknob or touch the lock face and wait a few hundred milliseconds for the door to unlock. As users already need to touch the doorknob to open the door, this seems like a natural and intuitive procedure. Additionally, the latency is modest: about 100ms for the signal recording, plus 10ms for computation (in our experiments) and a few milliseconds for the final wireless transmission. Finally, the scheme achieved 100% accuracy for benign usage in our experiments.

Limitations: *Vibrato* and touch-based intent protocols in general do have a few limitations. First, like distance-bounding protocols, *Vibrato* requires the use of new hardware in both the user's device and the smart lock—though this hardware is already mature and available on the market. Vibration generators are commonplace in phones and could be incorporated into locks, and existing bone conduction microphones could be added to wearable devices. Second, because *Vibrato* relies on sensing vibrations in the human body, the bone conduction sensor must be touching the user's body, which might not happen with a loose watch or wristband/bracelet. Finally, users will need to touch the door with the hand they wear their wearable device on. For smart watches, this means touching the door with the watch-wearing hand, which is not the dominant hand for most users.

While these limitations restrict the near-term deployment of TBIC protocols, our experiments show they can successfully achieve all of our security and usability goals. More broadly, we suggest that using touch as an indicator of a user's intent (rather than inferred proximity) might be a powerful primitive for securing users' interactions with Internet of Things devices.

6. RELATED WORK

Digital Locks: Previously, the Grey project at CMU built and deployed a digital lock system for office doors in their department [5]. They considered a challenging setting where access credentials are scattered across different administrative entities in a non-trivial distributed system, and they designed several ways that a device can prove to a lock that it is authorized. Additionally, they presented a technique for automatically inferring and resolving misconfigurations in the system's access control policy that can correctly predict the intended access policy 58% of the time [6]. Finally, their work identified several lessons about usability of smart lock systems. Of particular note, they found that a major factor in the appeal and usability of a smart lock system is the reduction in the time spent idly waiting for a door to unlock; delays of just a few seconds lead to significant user dissatisfaction and complaints [4]. While their findings shaped the two usability goals in our exploration of secure intent communication protocols, Grey was built before the emergence of modern smart phones and designed for office doors. As such, their system design, security goals, usage models, and attack vectors differ significantly from modern consumer-oriented smart locks and other smart home devices in the Internet of Things.

Security and Privacy in Smart Homes: Apart from this early work on door locks, several researchers have studied the security and privacy of emerging smart home technology. Kim et al. conducted a user study to identify intuitive access control policies for future smart homes; based on their user study, they proposed four access control groups: full control, restricted control, partial control, and minimal control [22]. These levels largely resemble the four access levels provided by modern smart locks. Ur et al. examine three home-automation devices and find that all of them have different access control policies and mechanisms, and none of them provide access control functionality that fully maps to intuitive user expectations and desires in the context of their home [37]. More broadly, Denning et al. [11] present a general taxonomy of attacks and security goals for protecting a user's privacy and physical assets in smart homes. They find that smart homes have a larger and more complex attack surface than existing systems because of the broad range of heterogeneous home devices, the lack of a professional administrator to oversee and maintain these devices, a diverse set of variable and personalized security goals that each home resident might want, and potentially new attack scenarios enabled by cyber-physical, sensor-rich devices. Recently, Oren et al. have explored new attacks enabled by smart TVs; rather than demonstrating attacks on the TV itself, they show that communication protocols used by smart TVs allow attackers to launch attacks through smart TVs against traditional computer networks and systems [31]. Finally, Ur et al. examined differences between teen and parent privacy and security perspectives on home surveillance systems and smart lock access logs [38]. Their study illustrates how emerging smart devices create complicated and conflicting notions of privacy and security within households. In contrast, our work studies the security and privacy risks posed by external adversaries.

7. CONCLUSION

In this paper we studied the security of commodity home smart locks with the goal of informing the design of future Internet of Things devices. We presented three classes of attacks and showed that existing smart locks are vulnerable to many of these attacks, enabling adversaries to gain unauthorized home access and learn private information about the user's household.

For two of these attack categories, we present defenses that can be implemented today without any hardware changes to existing devices. The third class of attacks is more challenging to stop without sacrificing usability, and no existing system provides an adequate defense. We explore three approaches to defend against this final class of attacks. One of these builds upon a novel mechanism we introduce, a bone conduction channel, which we implement and evaluate, demonstrating its ability to achieve our security and usability goals. Ultimately, we believe that if smart locks were to adopt the defenses we suggest, they could provide both better convenience and better security than their mechanical counterparts. More broadly, the design vulnerabilities we discovered and the defenses we proposed can help enhance the security of similar Internet of Things devices, while maintaining the new functionality they provide.

8. ACKNOWLEDGEMENTS

We thank Linda Lee and Peter Bailis for assisting us with two of our smart lock case studies, and Kurt Thomas, Michael McCoyd, Frank Li, and John Chuang for insightful discussions and feedback. This work was supported by the Intel Science and Technology Center for Secure Computing (ISTC-SC), AFOSR under MURI award FA9550-12-1-0040, FORCES (Foundations Of Resilient CybEr-Physical Systems) under NSF award numbers CNS-1238959, CNS-1238962, CNS-1239054, CNS-1239166), and generous support from Cisco Systems.

9. REFERENCES

[1] *IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks*, 2012. http://standards.ieee.org/findstds/standard/802.15.6-2012.html.

[2] August. http://august.com/.

[3] Lindsey Banks. Best bone conduction headphones of 2015. http://www.everydayhearing.com/hearing-technology/articles/bone-conduction-headphones/, July 2015.

[4] Lujo Bauer, Lorrie Faith Cranor, Michael K Reiter, and Kami Vaniea. Lessons learned from the deployment of a smartphone-based access-control system. In *Symposium on Usable Privacy and Security (SOUPS)*, 2007.

[5] Lujo Bauer, Scott Garriss, Jonathan M McCune, Michael K Reiter, Jason Rouse, and Peter Rutenbar. Device-enabled authorization in the grey system. In *International Conference on Information Security*, 2005.

[6] Lujo Bauer, Scott Garriss, and Michael K Reiter. Detecting and resolving policy misconfigurations in access-control systems. *ACM Transactions on Information and System Security (TISSEC)*, 2011.

[7] Ioana Boureanu and Serge Vaudenay. Challenges in distance bounding. *Security & Privacy, IEEE*, 2015.

[8] Eric Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 2012.

[9] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor C Leung. Body area networks: A survey. *Mobile networks and applications*, 2011.

[10] Danalock. http://www.danalock.com/.

[11] Tamara Denning and Tadayoshi Kohno. Empowering consumer electronic security and privacy choices: Navigating the modern home. In *Symposium on Usable Privacy and Security (SOUPS)*, 2013.

[12] Saar Drimer and Steven J Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX Security*, 2007.

[13] CES 2012: Ericsson. https://www.youtube.com/watch?v=pJ5fSWspBpo.

[14] NFC Forum. http://nfc-forum.org/what-is-nfc/about-the-technology/.

[15] Behrang Fouladi and Sahand Ghanoun. Security evaluation of the Z-Wave wireless protocol. *Black Hat USA*, 2013.

[16] Aurélien Francillon, Boris Danev, Srdjan Capkun, Srdjan Capkun, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.

[17] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *Radio Frequency Identification: Security and Privacy Issues*. 2010.

[18] Lishoy Francis, Gerhard P Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. In *Radio Frequency Identification: Security and Privacy Issues*, 2010.

[19] David Hantman. Fighting for you in New York. http://publicpolicy.airbnb.com/fighting-for-you/, October 2013.

[20] Kevo. http://www.kwikset.com/kevo/default.aspx.

[21] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, 2005.

[22] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Challenges in access right assignment for secure home networks. In *HotSec*, 2010.

[23] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wireless Networks*, 2011.

[24] Albert Levi, Erhan Çetintaş, Murat Aydos, Cetin Kaya Koç, and M Ufuk Çağlayan. Relay attacks on Bluetooth authentication and solutions. In *Computer and Information Sciences (ISCIS)*. 2004.

[25] Lockitron. https://lockitron.com/.

[26] Farhad Manjoo. The August Smart Lock Shows Why You Should Stick with Dumb Keys. http://bits.blogs.nytimes.com/2014/10/14/the-august-smartlock-shows-why-you-should-stick-with-dumb-keys/, Oct 2014.

[27] Mercedes-Benz. http://techcenter.mercedes-benz.com/en/keylessgo/detail.html.

[28] Elinor Mills. Drones can be hijacked via GPS spoofing attack. http://www.cnet.com/news/drones-can-be-hijacked-via-gps-spoofing-attack/, June 2012.

[29] Nest. https://nest.com/.

[30] Okidokeys. https://www.okidokeys.com/.

[31] Yossef Oren and Angelos D Keromytis. From the aether to the ethernet–attacking the internet using broadcast digital television. In *USENIX Security*, 2014.

[32] Kasper Bonne Rasmussen and Srdjan Capkun. Realization of RF distance bounding. In *USENIX Security*, 2010.

[33] Mike Ryan. Bluetooth: With low energy comes low security. In *WOOT*, 2013.

[34] M Seyedi, Behailu Kibret, Daniel TH Lai, and Michael Faulkner. A survey on intrabody communications for

body area network applications. *IEEE Transactions on Biomedical Engineering*, 2013.

[35] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. On the requirements for successful GPS spoofing attacks. In *ACM Conference on Computer & Communications Security (CCS)*, 2011.

[36] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, Christina Pöpper, and Srdjan Čapkun. Attacks on public WLAN-based positioning systems. In *Proceedings of the 7th International Conference on Mobile systems, applications, and services*, 2009.

[37] Blase Ur, Jaeyeon Jung, and Stuart Schechter. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*, 2013.

[38] Blase Ur, Jaeyeon Jung, and Stuart Schechter. Intruders versus intrusiveness: teens' and parents' perspectives on home-entryway surveillance. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014.

[39] Android Wear. https://www.android.com/wear/.

[40] Lin Zhong, Dania El-Daye, Brett Kaufman, Nick Tobaoda, Tamer Mohamed, and Michael Liebschner. Osteoconduct: Wireless body-area communication based on bone conduction. In *Proceedings of the ICST 2nd International Conference on Body Area Networks*, 2007.

APPENDIX

A. STATE CONSISTENCY ATTACKS: REVOCATION EVASION

Kevo: We found that the same attacks that work against Danalock also work against Kevo. By conducting attacks analogous to the ones against Danalock, we found that a thief can successfully evade all of Kevo's revocation mechanisms (including revocation through its website) by blocking all packets from the lock's servers from reaching the stolen mobile device (e.g., by switching it to offline mode). Like Danalock, Kevo allows any user with a resident key or an owner key to use the lock without Internet connectivity. Thus, both thieves and revoked attackers (e.g., an ex-spouse or apartment tenant) can also evade all revocation mechanisms, despite misleading messages from Kevo indicating the revocation succeeded. Additionally, Kevo's servers will not push revocation updates via other devices, so the offline phone can maintain indefinite access.

August:
August allows an owner to revoke another user's access through its mobile app. The August website also provides a "Lost Phone" feature on its website, which logs the user out on all her devices.

While August requires users with resident, recurring guest, and temporary guests keys to have their phone connected to the Internet, phones that have owner keys can freely interact with the lock even when not Internet-connected. As a result, we found that if Mallory steals an owner's phone, she can simply switch the phone to airplane mode, and she will retain access to the lock. As with Kevo and Danalock, key revocation in August works by having the remote server push a revocation message to the lock when the user's phone unlocks the door; however, if the phone is offline, the server cannot push this information to the lock and the lock remains unaware of the revocation. Similarly, the "Lost Phone" feature on August's website has no effect if Alice's stolen phone has been put into airplane mode, as the server cannot push a message to the mobile app instructing it to log Alice out. Like Danalock and Kevo, we found the August's revocation mechanism is not designed to allow the server to push revocation messages to the lock via other devices; thus, even if Alice logs in to a new device and interacts with the lock, her (offline) stolen phone will maintain access to the smart lock.

Like Danalock and Kevo, these revocation evasion attacks are exacerbated by insecure UI design. We found that for both the revocation and "Lost Phone" mechanisms, August's mobile interface displays a confirmation message that indicated successful revocation, even when the stolen phone maintained access via a state consistency attack.

Okidokeys: Although Okidokeys follows a DGC architecture, we found that it is somewhat more resilient to revocation evasion than August, Danalock, and Kevo. First, while any user can interact with the lock even if their phone is offline, the Okidokeys app requires the user to enter in a four digit PIN before enabling any of the app's functionality; users set this PIN during account creation. Unfortunately, a large body of existing work has shown that users often choose weak, predictable PIN codes, which means a thief might be able to bypass this mechanism. Furthermore, against a revoked attacker, this PIN code mechanism offers no security because Mallory is using her own account to evade revocation.

Second, Okidokeys provides a “sync” operation that owners can invoke if their device is in BLE range of the lock. The sync operation causes the owner’s device to push an update from the Okidokeys servers to the lock (such as an updated access control list). As a result, Alice can ensure her lock receives new revocation information by explicitly performing this manual sync operation. Okidokeys displays a “Sync needed” message after the owner revokes a user’s access. However, we found that it simultaneously displays a confirmation message stating “Access successfully removed from this user”, and it immediately removes the revoked user from the list of users who are shown as having access to the lock. Thus, the Okidokeys UI implies revocation was successful even if the revocation failed because of an offline phone. Because Okidokeys requires users to manually perform a sync and displays potentially misleading success messages following revocation, its design may leave users vulnerable to revocation evasion attacks.

B. STATE CONSISTENCY ATTACKS: ACCESS LOG EVASION

August and Kevo:

Like Danalock, August and Kevo rely on users’ devices to faithfully communicate their actions to the remote server; thus, an attacker who blocks the app’s packets from reaching the remote server (e.g., by taking the phone offline) can prevent her interactions with the lock from being recorded. Additionally, for these two locks, even if other users subsequently use the lock, the attacker’s interactions will not be updated in any log viewable by a legitimate user because the lock itself does not store and push log entries.

Okidokeys: In contrast, Okidokeys stores its access log on the lock itself and will only allow an owner to view the logs if her device is within BLE range of the lock or via a push notification. While Mallory can bypass push notifications by blocking packets on her phone from reaching the remote server, if Alice stands within BLE range of the lock and views the locally stored activity logs, she will be able to see Mallory’s actions.