Security Analysis of the Smart Lock Products against the Device Hijacking Attacks

Hiroki Kimura[1,a)] Jun Kurihara[1,b)] Toshiaki Tanaka[1,c)]

Abstract: In the digital age, the widespread adoption of IoT devices in smart homes introduces increased risks of cyber-attacks. This study focuses on the security of smart lock products, essential for controlling physical access to homes. We conducted a detailed analysis of five commercially available smart lock products and their mobile applications using debugging programs and static analysis to investigate their network communications and application source codes. Our research uncovered significant risks, including the theft of handshake keys used for device pairing, susceptibility to replay attacks, and potential unauthorized access by malicious entities. Based on these findings, we proposed a series of countermeasures designed to strengthen the security framework of these devices, thereby enhancing user safety in smart home environments. These recommendations aim to mitigate identified risks and provide a more secure operational context for smart locks.

1. Introduction

1.1 Background

The adoption of IoT platforms, particularly for smart homes, has surged recently, with devices for home automation like smart locks gaining popularity. These systems, which manage lighting and air conditioning through smartphones, enhance convenience but also introduce cybersecurity risks due to vulnerabilities in firmware and devices. Smart locks, controlling entry points like doors, are especially vulnerable to physical intrusion if hijacked. Attackers can exploit weak credentials, such as default or easily guessed passwords, or directly target the devices. Smart locks typically use the Bluetooth Low Energy (BLE) protocol, which, without strong security measures, is susceptible to man-in-the-middle or replay attacks, allowing unauthorized control of the locks.

1.2 Challenges

A study by Ye et al. [1] revealed that August's smart locks store handshake keys in plaintext on smartphones, exposing them to hijacking or denial-of-service attacks if the user operates a rooted smartphone. The study also highlighted risks of personal information leakage. Other research [2], [3] has examined the security of various smart locks, but many products still lack transparency in their security specifications, hindering users' ability to make informed choices.

1.3 Approach of the Research

This paper analyzes the security and black box specifications of smart lock products to identify risks and propose countermeasures. We examine the communication protocols and smartphone applications of five smart lock products, rooting the smartphones to access internal storage data and modifying APK files to enable debugging. This allows us to analyze network communications and program behavior in an unencrypted state. Our findings verify whether known risks persist in current products, leading to recommendations for standard security specifications and improvements to address identified risks.

[Research Ethics] This paper addresses the security concerns of commercially available smart locks. To ensure confidentiality, all product and company names are anonymized. The analysis is based on firmware and application versions current as of April 30, 2024. We conducted reverse engineering strictly within the scope of this study, ensuring that findings are used solely for research purposes. The study not only identifies security risks but also proposes countermeasures to improve the security of smart locks. And, we did not attempt unauthorized access to vendor cloud services, and the investigation was conducted anonymously, minimizing any potential harm to services.

1.4 Organization of the Thesis

The structure of this paper is organized as follows: Section 2 presents related research on the security of IoT devices and smart lock products. Section 3 provides an overview of the smart lock products and the service architecture utilized in this study. Section 4 details the target devices for our security analysis, outlines our research perspective, and describes the specific methodologies employed. Section 5 presents the results of our analysis. Section 6 discusses the results from several perspectives and proposes security enhancements based on the identified risks. Finally, Section 7 concludes the paper by summarizing our findings and

[1] Graduate School of Information Science, University of Hyogo
[a)] hiroki.wr@gmail.com
[b)] kurihara@ieee.org
[c)] toshi@gsis.u-hyogo.ac.jp

contributions.

2. Related Works

Khanam et al. [4], Meneghello et al. [5], and Annamalai [6] identified key security concerns in IoT devices, detailing common attack vectors and defenses across various IoT layers. Westerlund et al. [7] and Chen et al. [8] discussed vulnerabilities from using static device IDs, which could enable man-in-the-middle attacks through device impersonation. Ibrahim et al. [9] and Feng et al. [10] outlined procedures to prevent IoT devices from downgrading or improperly updating firmware, mitigating related risks.

Regarding smart locks, Ho et al. [3] identified security risks in products with automatic unlocking via geo-fencing and touch-to-unlock features, which could inadvertently allow unauthorized access. Caballero-Gil et al. [2] validated a threat model for the Nuki Smart Lock, confirming AES encryption for BLE communication. Jiliang et al. [11] and Bapat et al. [12] proposed secure BLE communication protocols to enhance security.

However, secure storage of encryption keys within applications and devices, the critical endpoints of BLE communication, is vital to prevent device hijacking. Ye et al. [1] evaluated the August Smart Lock's threat model, highlighting risks from storing encryption keys in plaintext on rooted smartphones, which could lead to hijacking and denial-of-service attacks. This study examines the security specifications of applications used in the latest versions of several smart lock products. Based on our analysis, we propose a security framework to safeguard smart locks against advanced cyber threats.

3. Overview of Smart Lock Devices

In this chapter, we provide an overview of the main features of smart locks that are the subject of this research.

3.1 Overview

3.1.1 Functionality

A smart lock is an electronic system that replaces traditional keys with electronic mechanisms. In smart homes, these devices are typically installed on the thumb-turn of a door lock. The device typically features a front knob for manual operation from the inside and a rear indentation that attaches to the thumb turn. A control signal from a smartphone application activates an internal motor, which rotates the thumb-turn to lock or unlock the door.

Figure 1 illustrates a typical smart lock network configuration. Communication between the smartphone application and the lock primarily occurs via Bluetooth Low Energy (BLE), limiting operation to within the Bluetooth range. However, a Wi-Fi hub can extend control range by managing BLE communication with the lock and Wi-Fi communication with the smartphone. Some models also include external pads for numeric passcodes, card keys, or fingerprint authentication.

Users first install the manufacturer's app from the Google Play Store or Apple App Store. Once installed, the device can be paired with the app, requiring the app to be within range of the smart lock's Bluetooth Low Energy (BLE) signal. After pairing, the user adjusts the thumbturn position for installation. The app interface then displays a button to lock or unlock the door, and continuously monitors the smart lock's status.

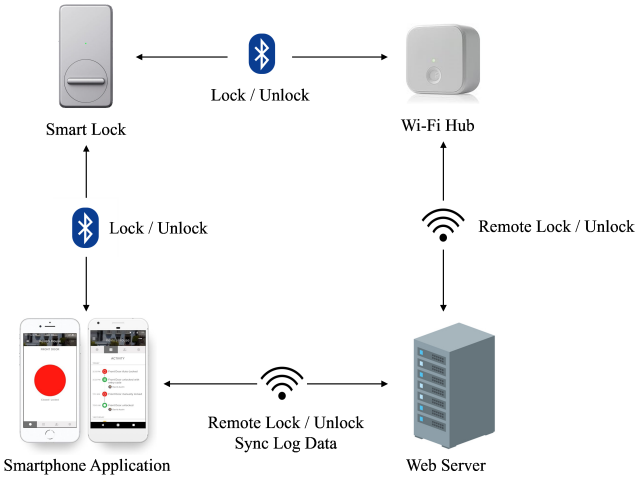

Fig. 1 Network diagram of smart lock

Smart locks retain the basic functions of traditional door locks while adding new methods of operation through applications, hubs, or pads. Although physical keys can still be used, smart locks introduce new cybersecurity risks through potential compromises of smartphones and lock devices, in addition to the traditional physical security risks.

3.1.2 Access Levels

Smart lock systems offer two main user privileges: Owner and Guest. Owners have full control, including locking/unlocking, managing accounts, and adding/removing devices and users. For example, parents might set up Owner accounts and issue Guest accounts to their children, limiting them to basic functions. Similarly, rental property owners can grant temporary Guest access to tenants and revoke it after the rental period. A major security concern is device hijacking, where attackers gain unauthorized access by compromising Owner privileges. This can happen if an Owner mistakenly grants the wrong privilege or if an attacker hacks the Owner's credentials, highlighting the need for strong security measures.

3.1.3 Access Logs

Smart lock applications typically record access logs whenever the device is used. For example, parents can verify when their child unlocked the door, and managers in workplaces can monitor access to specific areas.

However, in the case of device hijacking, access logs are limited in utility. While they record events, they do not prevent unauthorized access. Moreover, the logs cannot confirm the identity of the person who accessed the door. A malicious Owner could also reset the device, erasing the logs and preventing legitimate users from reviewing the history, indicating that access logs alone are insufficient to counteract hijacking attacks.

3.2 Research Scope

Section 3.1.1 mentioned related devices like Wi-Fi hubs and pads, but the risks they pose are common to all electronic locks and not unique to smart locks. Therefore, this study focuses on the security features at the endpoints and the communication between

the application and the device. The analysis was conducted using only the application and the smart lock device, which is consistent across all smart lock products examined in the research.

4. Methodology

4.1 Research Objectives

In this chapter, we outline the objectives and methodology of our investigation into the security of smart lock products.

Our investigations aim to identify security risks associated with smart locks and recommend strategies for defending against cyber attacks. Existing studies, particularly those examining the August Smart Lock, have highlighted four primary risks:

Storing handshake keys in plaintext. Owner account leakage. Personal information leakage. Denial of Service (DoS) attacks.

In addition to addressing these established concerns, our study examines risks associated with sharing duplicate keys and the specifications for account logins to assess the risks of unauthorized access. We present results and discussions from several perspectives to provide a comprehensive analysis of the current security landscape for smart locks.

4.2 Objectives

4.2.1 Storing Handshake Key in Plaintext

A handshake key is used to encrypt Bluetooth Low Energy (BLE) communications during the pairing process between an application and a device. Secure key-sharing protocols, such as Elliptic Curve Diffie-Hellman (ECDH), help mitigate eavesdropping by generating a shared secret from public/private key pairs. With Android 11, storing application data in internal directories has become mandatory, restricting access to these directories to the respective applications. However, if a device is rooted, attackers can access these directories, potentially stealing plaintext handshake keys and hijacking the smart lock. The risk is heightened when acquiring used smartphones that may unknowingly be rooted.

4.2.2 Owner Account Leakage

When users log into a smart lock application, authentication with the server involves issuing a Refresh Token and an Access Token as per OAuth2.0 [17]. However, theft of these tokens alone does not lead to device hijacking, as they are used for server authentication, not for sending control signals via BLE. On the other hand, unauthorized access can be facilitated by duplicating profile data from the application ' s internal storage, allowing attackers to bypass the login process and replicate BLE requests with the legitimate account.

4.2.3 Personal Information Leakage

Users often enter personal details such as residence name, address, and full name when setting up a smart lock. While accurate information is not mandatory, entering such data increases the risk if accessed by attackers, especially on rooted smartphones where sensitive data stored in the app ' s directory can be exposed. Additionally, some smart locks require location permissions to be enabled, raising concerns about excessive data collection and potential personal information leakage, particularly through server breaches.

4.2.4 Denial of Services

Defending against Denial of Service (DoS) attacks is crucial for IoT devices like smart locks, which are vulnerable to multiple redundant requests. Because attackers can prevent legitimate users from controlling the lock by establishing a connection with it first. In addition, smart locks are battery-powered, and if the battery depletes, the lock becomes inoperable. Continuous operation can drain batteries quickly, and attackers can expedite this by initiating unnecessary pairings.

4.2.5 Unauthorized Access to Smart Lock Account

Some smart locks have insufficient security measures for account privileges, potentially allowing attackers unauthorized access. Standard authentication typically includes account ID and password verification, often with two-factor authentication (2FA). However, some smart locks have weaker protocols, allowing login with just an ID and a short security code, without requiring a password. Additionally, some smart locks are vulnerable to brute force attacks on invitation codes used to issue duplicate keys. While owners can remove unauthorized users, there remains a risk that an attacker could compromise the lock before the owner is aware.

4.3 Target Devices

In this paper, we analyze five smart lock products available on the market. We acquired both the smart lock devices and their respective smartphone applications from the manufacturers. To maintain confidentiality and avoid the risks associated with disclosing specific product names and specifications, we will not reveal the identities of these products. Instead, we refer to them anonymously and randomly as Devices A, B, C, D, and E.

4.4 Preparation

4.4.1 Rooting Smartphone

In this study, we prepare a rooted Android tablet to facilitate the investigation of various smart lock applications. Rooting refers to the process of granting a user elevated root permissions, which enables access to directories and the ability to execute commands that are typically restricted. This enhanced access allows for a detailed examination of the internal storage of smart lock applications to ascertain what types of information are stored there. The stored data within internal storage can contain sensitive information that may be susceptible to unauthorized access or personal data leakage.

4.4.2 Analysis by Debugging

In addition to investigating internal storage, we analyze the sequence of network communications from installing the application to pairing with the smart lock and performing lock control. We obtain the APK files necessary for our analysis from an online platform, i.e., Google Play. Typically, applications in the development stage are analyzed with debugging enabled, but it is disabled at the stage of public release to prevent internal specification analysis. This is either specified in the application's Manifest file as

```
android:debuggable=false,
```

or by the absence of this entry. To enable debugging,

`android:debuggable=true`,

must be specified, a change that can be easily made by modifying the APK files. This enables us to monitor the network communications between the server and devices, providing deeper insight into the key exchange and lock control processes.

4.4.3 Analysis of Bluetooth HCI Snoop Log

Since Android 4.4, the ability to capture and log HCI[*1] packets for Bluetooth communications has been available. By activating the "Enable Bluetooth HCI snoop log" option in the developer settings on the targeted Android device, Bluetooth communication logs can be saved internally. These logs are stored in the data/misc/bluetooth/logs directory and can be analyzed using packet analysis tools such as Wireshark [20]. As smart lock applications utilize BLE communication, it is possible to determine what signals are sent to the device. However, the data in the body of packets exchanged via BLE communication is in binary format, making it difficult to directly understand the contents of the communications. Therefore, by combining this with the network analysis enabled by the application debugging described earlier, a more detailed protocol analysis can be performed.

5. Results

This chapter describes the findings relative to the investigative perspectives discussed in Section 4-2. Initially, the research findings are summarized in a table shown in Table 1. The existence of risks are indicated as either "Yes" or "No."

5.1 Storing Handshake key in Plaintext

We performed rooting on Android devices to inspect files in specific directories, examining how handshake keys are stored. To verify the presence of handshake keys used during the pairing process, we conducted a comparative analysis between the data from applications when paired with a device and when not paired.

Product A.

Upon examining the application's specific directory, we did not find encryption keys in XML files or databases. However, by analyzing log files that recorded network communications with the server and BLE communication histories with the device, we confirmed that the application receives encryption parameters from the server as shown in Figure 2. Storing sensitive information in plaintext within log files presents a risk of misuse.

Product B.

Upon inspecting the application's database, we identified values believed to be handshake keys used for communication with the device, as depicted in Figure 3. These values were found both in the database file and in XML files that store cache information for the device. Additionally, log files that record communications with the smart lock revealed that they also include timestamps of when the handshake occurred, along with the values of the used hash codes and random numbers.

We also found the risk with remote control using Wi-Fi hubs while examining network communications. We discovered that it was possible to execute replay attacks for locking and unlocking mechanisms. As illustrated in Figure 4, the parameters for HTTP requests were identified through debugging. By replicating these parameters and sending the HTTP request, we confirmed that the remote lock could be operated.

Product C.

Upon examining the application's database, we confirmed that a table managing device information contained a 'Secretkey', as shown in Figure 5. Additionally, static analysis of the source code revealed the logic used to compute the hash code values from the stored information. This suggests that a malicious attackers could easily replicate BLE communication with the device.

Product D.

Upon reviewing the request-response messages from server communication and the application's specific directory, we did not find any key information used for encrypting communications with the device. Additionally, no communication logs were found. The absence of key exchange information stored in the application suggests a low likelihood of key leakage.

Product E.

In the application's specific directory, XML files named key pairs and hashes were identified as depicted in Figure 6; however, it was not possible to confirm whether these are used for pairing. No key-like information was found in the communication logs either. Given that no key exchange information is stored in the application, the risk of key leakage is considered low.

5.2 Owner Account Leakage

We investigated whether malicious actors could obtain unauthorized account privileges by exploiting data leakage from di-

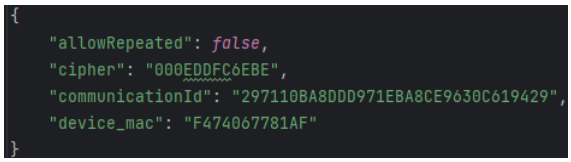

Fig. 2 Product A. Cipher code

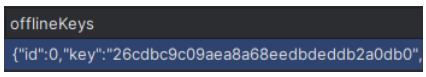

Fig. 3 Product B. Offline keys

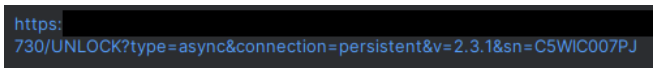

Fig. 4 HTTP remote unlock request

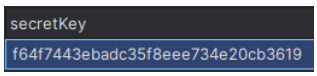

Fig. 5 Product C. Secret key

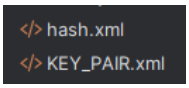

Fig. 6 Product E. Key pair and hash

[*1] HCI: Host Controller Interface.

Table 1 Existence of risks identified through security analysis

Product	Plaintext Handshake Key	Owner Account Leakage	Personal Information Leakage	Denial-of-Services	Unauthorized Access
A	"Yes"	"Yes"	"No"	"Yes"	"Yes"
B	"Yes"	"Yes"	"Yes"	"No"	"No"
C	"Yes"	"Yes"	"Yes"	"No"	"Yes"
D	"No"	"No"	"Yes"	"Yes"	"Yes"
E	"No"	"Yes"	"Yes"	"Yes"	"Yes"

rectories due to smartphone rooting. Ideally, legitimate users and attackers would use different devices. In this analysis, given that malicious actors could potentially falsify device information, this study was conducted using only one tablet device. The steps of the procedure were as follows:The procedure was as follows:

1. Log into the application account
2. Extract internal data
3. Reinstall the application
4. Overwrite the extracted data
5. Restart the device

This process aimed to determine if an attacker could gain account privileges without knowing the ID and password. For clarity, the account initially logged into is referred to as Account A.

Product A.

After logging into Account A, we extracted all the files under the application directory. Following this, we deleted and reinstalled the application, allowed the location permissions necessary for pairing with the smart lock upon first launch, and then overwrote the extracted data in the specific directory before rebooting the device. Upon restarting and launching the application, it can be logged in as Account A without requiring authentication, indicating a potential risk of owner account leakage for Product A.

Product B.

Similar to Product A, by extracting and then overwriting Account A's information, we could log into the application as Account A without authentication.

Product C.

Same as Product A, by extracting and then overwriting Account A's information, we could log into the application as Account A without authentication. This product requires location permission to use the smart lock.

Product D.

After logging into Account A, extracting information, and overwriting it post-reinstallation of the application, the application launched in a logged-out state. Unlike other products, we could not confirm owner account leakage through profile information modification.

Product E.

After logging into Account A, extracting the specific directory, and then reinstalling and allowing both permissions for location and Bluetooth connection at the initial launch, we restarted the smartphone, and then overwrote the profile. This allowed the application to launch logged into Account A.

deviceID	houseID	latitude	longitude
フィルター	フィルター	フィルター	フィルター
8196FDCC70CA42B68BEDF8888C2D4730	NULL	3[redacted]	13[redacted]

Fig. 7 Product B. Location data stored in internal storage

5.3 Personal Information Leakage

Smart locks require the registration of different data on the application. However, some smart locks mandate and store personal information.

Product A.

While the registration of personal details such as address and full name is not required, it is necessary to set room name and lock name, and the account nickname can be optionally registered. In the internal storage of the application, no data related to personal information was found, except for the email address. Therefore, it can be said that Product A has a low risk of personal information leakage.

Product B.

As shown in Figure 7, we confirmed that the application database creates tables for latitude and longitude. These fields remain empty when the lock is used only with Bluetooth; however, when using the Wi-Fi hub to control the lock remotely, permission for location information is required, and latitude and longitude data are written to the table. Thus, using both Wi-Fi hub and rooted device could potentially expose the address where the lock is installed.

Product C.

The account's email address is mandatory, and the nickname can be optionally entered. Additionally, it was confirmed that no personally identifiable information is stored within the application. However, since location information permission is necessary for the use of the smart lock, there is a possibility that the user's location information could be transmitted to the service server.

Product D.

In this product, users can freely register the nickname of the account and the name of the device. The information requested from the server includes the email address and nickname, but no other personally identifying information was found. However, since location information permission is required for the use of the smart lock, there is a possibility that the user's location information could be sent to the service server.

Product E.

In this product, users can freely register the name of the device. The XML in the specific directory contains an 'IdToken' in JWT format [21], which includes an email address, but no other personally identifying information was found. Additionally, since location information permission is required for the use of the smart lock, there is a possibility that the user's location information could be transmitted to the service server.

5.4 Denial of Services

As part of replicating a Denial of Service (DoS) attack, we observed the behavior when multiple applications from the same device simultaneously performed lock control actions, using two smartphones logged into the same account for this study.

Product A.

While one application was performing a lock operation, executing an unlock operation from another application resulted in the device's action being overwritten, replacing the lock action with an unlock action. This indicates that the device does not discriminate between BLE communication sources and accepts all BLE communications directed at it. Thus, a malicious attacker could continuously send opposing lock/unlock commands, making a Denial of Service attack feasible.

Product B.

Simultaneous use of two smartphones was possible. When one application performed an unlock operation, its status was synchronized with the other application because the applications retrieved the device's lock status in real-time. Even considering a replay attack by a malicious attacker, where a lock operation is requested immediately after an unlock operation via Wi-Fi, the operations were processed in the order received, and the lock operation was executed after the unlock was completed.

Product C.

The behavior was similar to that observed with Product B, with the possibility of connection from two devices simultaneously and real-time synchronization of status.

Product D.

When connected to one device from two smartphones, only one of the applications would accept lock control, and the other application could not communicate with the device. The connection starts when the device detects that it is within BLE communication range, allowing a malicious attacker to connect to the device and take control, preventing legitimate user operations. Specifically, if a malicious attacker continuously sends lock signals, the legitimate user cannot connect to the device, thus enabling a Denial of Service attack.

Product E.

This behavior was similar to that observed with Product D, where only one application receives lock control when a device is connected from the same account.

5.5 Unauthorized Access to Smart Lock Account

We researched the necessary information for account login and the authentication requirements for inviting guest accounts. Additionally, we examine how to revoke access in the event of an account leakage.

Product A.

Users log in with an email address and password. If the password is forgotten, a reset requires entering a six-digit security code sent via email. When granting permissions to another account, a six-digit security code, combining uppercase letters and numbers, is generated on the Owner account. This code allows the guest account to control the lock and is valid for three days. The risk exists that attackers could obtain the code through leakage or brute-force attacks, potentially leading to Denial of Service (DoS) attacks by stealing multiple users' rights. In cases of account compromise, Owners can view login history and remotely log out suspicious devices. However, if attackers gain access to the Owner's account, they could also log out legitimate users. Password changes require a security code sent to the Owner's email, making account takeover difficult without email access, thus allowing potential recovery.

Product B.

Login requires a six-digit security code sent to either an email address or telephone number. Password resets require codes sent to both the email and phone. When granting permissions, entering the recipient's phone number automatically grants access to that account. This system prevents erroneous permissions unless the Owner inputs the wrong phone number.

Recovery options include viewing accounts with access to the smart lock and removing unauthorized ones. Unlike Product A, this system does not track logins by individual devices. Hijacking attempts are harder than Product A due to the need for dual-code entry during password resets.

Product C.

This product does not use a traditional password for login; instead, it requires a four-digit code sent to the user's email. This method may increase the likelihood of unauthorized logins. Permissions are granted by entering the user's email or scanning a QR code. Account recovery involves managing logout operations similar to Product B. Although the four-digit code is vulnerable to brute-force attacks, the product ' s unique hardware reset feature allows device initialization, providing an extra security layer in case of severe breaches.

Product D.

Users log in with an email and password. Password resets require a six-digit security code sent via email. After pairing, users cannot log out without uninstalling the app. Permissions are granted via a static URL with the device's serial number, which can be sent to a specified email. The invited user must have the same email address as specified during the invitation. For account leakage recovery, immediate device initialization and password reset are necessary, as logout is not possible once paired. If an

attacker initializes the device first, they can rebind it to another account, rendering the lock unusable.

Product E.

Account login is done through email and password. Password resets are handled via a URL link sent to the email. Permissions are granted by sharing a URL with a dynamically generated 36-digit invitation ID. The invitation is valid for 24 hours, reducing the risk compared to Product A but still vulnerable to URL leakage or brute-force attacks.

Recovery is similar to Product B, where logout is managed on an account-by-account basis. Account takeover requires a password reset through an email link, so without email compromise, takeover is not possible.

5.6 Summary

By analyzing five products, the specification differences in smart locks depending on the product became evident. A security risk identified in existing research, where handshake keys used for BLE communication are stored in plaintext within the application's internal directory, still exists in some products. This implementation allows malicious attackers to impersonate legitimate BLE communications and hijack smart lock operations. Moreover, it was confirmed that malicious attackers could extract data from the directory and replicate it on another smartphone, enabling unauthorized account logins or rendering devices unusable through DoS attacks. In cases of account compromise, it was confirmed that malicious attackers could log out legitimate users or initialize smart lock devices. This attack demonstrates that attackers could take over a device by re-pairing it with another account, and rendering smart locks without a hardware reset feature completely unusable.

Additionally, in Product B, when using the remote lock feature via a Wi-Fi hub, it was confirmed that location information is stored in the internal directory, posing a risk of personal information leakage. Furthermore, if remote lock communications are intercepted by a proxy, malicious attackers could resend URL requests, enabling replay attacks that remotely control lock operations.

6. Discussion

In this chapter, we provide suggestions for security measures to reduce the risks for smart locks identified in this analysis.

6.1 Secure Storing of Handshake Key

Concerning the plaintext storage of handshake keys for BLE communication observed in some products, it was verified that although a key exchange protocol encrypts the communication path, the risk of compromise remains if the device is compromised due to the keys being stored in plaintext within the smartphone software. A viable countermeasure is the utilization of secure elements of smartphone devices. Secure elements, which are tamper-resistant hardware domains, are incorporated into many contemporary Android devices. Information in the secure elements is unreadable once it has been written. Therefore, by entrusting the encryption of BLE communication messages to a secure element, the handshake key cannot be readable even on the rooted device. This approach significantly reduces the risk of cryptographic key leakage on rooted devices.

6.2 Owner Account Profile Protection

We investigated whether malicious attackers could gain unauthorized account privileges through data leakage from specific directories facilitated by smartphone rooting. In four of the five analyzed products, excluding Product D, it was possible to achieve a logged-in state on other smartphones by copying extracted account profiles, bypassing the need for traditional login credentials. This highlights a significant security flaw on rooted smartphones: it is easier for attackers to duplicate legitimate application data than to intercept and misuse tokens (Refresh Tokens or Access Tokens) intended for server communications.

Further investigation focused on Product D's mechanisms that prevent login from copied profiles was inconclusive. The product's resistance could not be definitively assessed due to encrypted or obfuscated source code and secure network communications. Therefore, a thorough specification analysis of Product D's security measures is designated as future work.

As a proactive security measure, implementing a root detection mechanism within the application is recommended to restrict access when the application is launched on a rooted smartphone. This strategy aims to reduce the risk of account profile theft by malicious attackers, thereby preventing unauthorized use.

6.3 Appropriate Use of Personal Information

It has been discovered that several products require users to grant permission for location data usage even when only using BLE communication. Therefore, it is recommended that products designed to operate solely on BLE communication should not grant location permission for users and reserve location data usage solely for instances involving Wi-Fi hub operation.

Moreover, it has been revealed that Product B stores latitude and longitude information within the application's internal storage, increasing the risk of address identification. Companies providing smart locks must disclose what personal information is stored on devices or web servers to ensure users fully understand their data privacy. Additionally, it is strongly recommended that no personally identifiable information be retained within the application to prevent any breach of personal data.

6.4 Defence against Denial-of-Service Attacks

Concerning DoS attacks, it has been determined that malicious attackers can render the lock inoperable by holding its connection and can perform attacks that involve re-pairing it with a different account by resetting the compromised account's lock.

Regarding lock possession, similar to the semaphore concept in computer science, products that implement exclusive control—where only one application can connect at a time—are vulnerable to attacks. If a malicious attacker connects to the device after the legitimate user leaves the connection range, the legitimate user cannot operate the lock. To prevent device takeover through lock possession, it is advisable to avoid exclusive control features. Allowing simultaneous connections from multiple devices can

prevent attackers from monopolizing the lock. Moreover, prioritizing connections from a specified smartphone during pairing can help avoid unauthorized control.

6.5 Prevention of Unauthorized Access

The analysis revealed that some products have inadequate measures against unauthorized account access. The lack of a password requirement at login and the absence of two-factor authentication allow for the potential of unauthorized logins through brute-force or credential-stuffing attacks. Therefore, it is crucial to mandate two-factor authentication via SMS or email in addition to ID and password verification.

Regarding the creation and sharing of duplicate keys, the method adopted by Product D (sending a duplicate key to the entered email address and verifying it matches the recipient's account email) can prevent unauthorized access due to user errors. A malicious attacker cannot successfully execute a duplicate key theft unless they compromise both the recipient's email address and the smart lock account.

Regarding the re-pairing of compromised accounts, measures such as those in Product C, which allow lock owners to reset the hardware, or enabling SMS authentication during lock initialization, are effective. In particular, the hardware reset feature of the device should be provided as it enables recovery from any software-related risks and obviates the need for complex programming designs.

7. Conclusion

In this paper, we investigated the security implementations and black-box specifications of smart lock products to identify potential security risks. Our research encompassed five smart lock products and their associated APK files available on the Internet, utilizing debugging programs and static analysis. We analyzed network communications of these devices and their applications' source code. From this analysis, we confirmed that there still remained the risks previously highlighted by existing research. Furthermore, we identified potential risks of replay attacks and unauthorized access by malicious attackers. Based on these analyses, we proposed countermeasures to mitigate the uncovered risks and enhance the security of smart lock products.

Acknowledgments This work was supported in part by NICT Grant No. NICT22401, JSPS KAKENHI Grant No. JP22K11994, and the KDDI Foundation Research Grant.

References

[1] M. Ye, N. Jiang, H. Yang, and Q. Yan, "Security analysis of internet-of-things: A case study of august smart lock," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 499–504.

[2] C. Caballero-Gil, R. Álvarez, C. Hernández-Goya, and J. Molina-Gil, "Research on smart-locks cybersecurity and vulnerabilities," *Wireless Networks*, May 2023.

[3] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 461–472.

[4] S. Khanam, I. Ahmedy, M. Idna Idris, M. Jaward, and A. Bin Md Sabri, "A survey of security challenges, attacks taxonomy and advanced countermeasures in the internet of things," *IEEE Access*, vol. 8, pp. 219 709–219 743, 2020.

[5] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? a survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[6] A. Lakshmanan, "Literature review on the latest security & the vulnerability of the internet of things (IoT) & a proposal to overcome," Apr. 2020.

[7] O. Westerlund and R. Asif, "Drone hacking with raspberry-pi 3 and wifi pineapple: Security and privacy threats for the internet-of-things," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, 2019, pp. 1–10.

[8] J. Chen, C. Zuo, W. Diao, S. Dong, Q. Zhao, M. Sun, Z. Lin, Y. Zhang, and K. Zhang, "Your IoTs are (not) mine: On the remote binding between IoT devices and users," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 222–233.

[9] M. Ibrahim, A. Continella, and A. Bianchi, "AoT - attack on things: A security analysis of IoT firmware updates," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS & P)*, 2023, pp. 1047–1064.

[10] X. Feng, X. Zhu, Q.-L. Han, W. Zhou, S. Wen, and Y. Xiang, "Detecting vulnerability on IoT device firmware: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 1, pp. 25–41, 2023.

[11] J. Wang, F. Hu, Y. Zhou, Y. Liu, H. Zhang, and Z. Liu, "Bluedoor: breaking the secure information flow via ble vulnerability," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 286–298.

[12] C. Bapat, G. Baleri, S. Inamdar, and A. Nimkar, "Smart-lock security re-engineered using cryptography and steganography," Nov. 2017, pp. 325–336.

[13] SwitchBot, "Switchbot_smartlock," 2024, https://www.switchbot.jp/products/switchbot-lock.

[14] August, "August_home," 2024, https://august.com/.

[15] CANDY-HOUSE, "Sesamesdk," 2024, https://github.com/CANDY-HOUSE.

[16] Google, "Storage updates in android 11," 2024, https://developer.android.com/about/versions/11/privacy/storage.

[17] D. Hardt, "The OAuth 2.0 authorization framework," RFC6749, Oct. 2012, https://www.rfc-editor.org/rfc/rfc6749.

[18] C. Gan, "Gjoy dex analyzer," 2024, http://www.gda.wiki:9090/index.php?language=en.

[19] iBotPeaches, "Apktool: A tool for reverse engineering android apk files," 2024, https://apktool.org/.

[20] W. Foundation, "Wireshark," 2024, https://www.wireshark.org/.

[21] M. Jones, J. Bradley, and N. Sakimura, "JSON web token (JWT)," RFC7519, May 2015, https://www.rfc-editor.org/rfc/rfc7519.