symmetry

MDPI

Article

AACS: Attribute-Based Access Control Mechanism for Smart Locks

Zhenghao Xin [1], Liang Liu [1,*] and Gerhard Hancke [2]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, No. 29 Yudao Street, Nanjing 210016, China; xinzhenghaox@nuaa.edu.cn

[2] College of Science and Engineering, City University of Hong Kong, Kowloon 999077, Hong Kong, China; gp.hancke@cityu.edu.hk

* Correspondence: liangliu@nuaa.edu.cn

Received: 1 June 2020; Accepted: 22 June 2020; Published: 23 June 2020

Abstract: This article researched the security and application of smart locks in Internet of Things environments in the domain of computer and engineer science and symmetry. Smart locks bring much convenience for users. However, most smart lock systems are cloud-based and it is problematic managing and enforcing the permissions of an authorized device if the device is offline. Moreover, most smart lock systems lack fine-grained access control and cascading removal of permissions. In this paper, we leverage attribute-based access control mechanisms to manage the access of visitors with different identities. We use identity-based encryption to verify the identity of the visitor. In our proposed system, the administrator uses the policy set in the smart lock to implement access control on the device side, which reduces the dependence of access control on the server. We set attributes such as role, time, date, and location to have fine-grained control over access to different permissions and roles that might appear in the house. And the scheme provides the cascading delete function while providing the group access function. Our solution considers multiple roles in the home as well as hierarchical management issues, and improves the applicability of the smart lock system in complex residential and commercial situations. In the experimental section, we show that our system can be applied to premises with many different inhabitant identities.

Keywords: attribute-based access control; smart locks

1. Introduction

1.1. State of the Art

In recent years, the Internet of Things is increasingly influencing our daily life. These intelligent devices, which brings many conveniences to our life, creating security, privacy, and other risks, e.g., in smart voice assistants [1], the Internet of vehicles [2], and smart homes [3–5].

Nowadays, smart locks have been accepted by more household and businesses, and they are gradually replacing traditional locks. New smart locks no longer need a tedious unlocking method. Smart locks are also installed in smart car systems, e.g., such as [6]. The user only needs to be close to the smart lock, and unlock the lock by holding up his/her phone. However, the convenience comes with a possible security issue, such as the risk of locks being opened illegally. As the most important safety protection component for smart home systems, smart locks have received more attention with respect to security and reliability. Many researchers have worked on the security of smart locks in recent years, e.g., [7–10] with authentication and granular access control methods for smart locks becoming a popular research direction.

To enhance the security and convenience of smart locks, some researchers have studied authentication schemes based on biometrics [11–13], for example Liu et al. [11] used three sensors to

simulate the user's hand movement in three-dimensional space as a signature. However, such smart locks are not as convenient and intelligent as those using digital keys.

When the location of a smart lock has a fixed gateway, the smart lock system can provide the user with the ability of remote access. In order to increase the security of remote access, Chao et al. [14] integrated blockchain and group signatures to provide anonymous authentication to family members, and used message authentication codes to authenticate the home gateway. But in many houses, the location of the lock does not have a fixed gateway and the lock requires the use of the user's device as a gateway to interact with the server when the user accesses it. The data for the smart lock resides in the cloud [15]. In this paper we consider this type of smart lock system consisting of three components: An electronically-augmented door bolt installed onto an exterior door, a mobile device that can connect and control the lock, and a remote web server. Users can install the mobile application on their mobile devices to control the smart lock. They can start by creating an account on the manufacturer's server. They can then pair their mobile devices with the lock via a local wireless channel such as Bluetooth Low Power (BLE).

When a user's phone enters the operational range of the lock, information from the manufacturer's server is transferred to the lock. Ho et al. call this architecture the Device-Gateway-Cloud (DGC) model [15]. We investigated several types of smart locks on the market based on the DGC architecture, such as August [16], Danalock [17], Okidokeys [18], and Kevo [19]. Detailed information of the DGC architecture is shown in Figure 1. We are aware that simpler locks exist, which are off-line and lack integration with other systems, using simple keypads, fingerprint locks, and smart card tokens. These types of locks simply replace a key with an alternate mechanism but remain unintelligent, relying on manual human configuration and management and having no means to dynamically update access conditions, e.g., physically pre-programmed with a common PIN that everyone uses or still having a user collect a configured card.

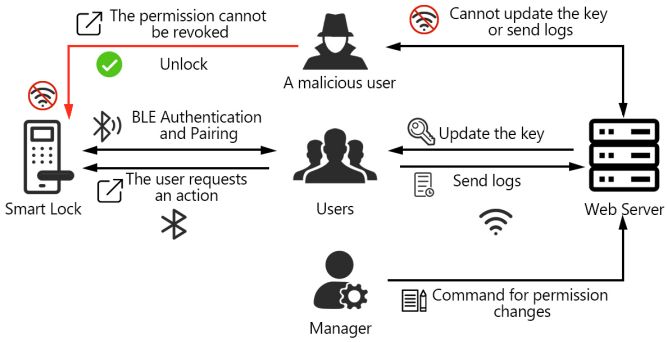

Figure 1. This is the Device-Gateway-Cloud (DGC) model, a common communication mode between intelligent locks and the cloud. The smart lock USES the user's smartphone as a mobile gateway to communicate with the cloud and connect with the user's phone via Bluetooth Low Power (BLE). Since most smart locks store data in the cloud rather than inside the lock, users access the locks using a digital key on their smartphone. As a result, malicious users can take their phones off-line to avoid having their digital keys withdrawn by administrators.

1.2. Motivation and Contribution

For smart locks that use the DGC model, the lack of connection between the smart lock and the Internet leads to the inability of the owner to directly control the lock, with access permissions of an off-line lock, when accessed with an off-line device, not changing as it cannot receive instructions in that moment. This could lead to a race condition, with users with revoked permissions gaining access if they disconnect their device and reach the lock before another user. Moreover, the complexity of social relations raise a challenge on permissions distribution for the smart lock system. For example, for premises with more occupants and complex identities, only having a single person as managing authority can be troublesome. However, the ability of multiple people to distribute permissions

will cause the problem of cascading the deletion of permissions: When the manager deletes the permissions of a member, other people that are granted permissions by this member should also lose the permissions they have. Otherwise, the operation of deleting permissions will become extremely complicated and have security risks.

Many smart locks have another shortcoming in that it lacks fine-grained access control. Some smart lock systems simply control access to the personas. They are not "smart" enough. There may be many different access roles for a household of business. These different roles may need to be controlled on a specific time or date. To make matters worse, today's smart lock systems cannot handle relay or replay attacks. As smart locks can not judge the visitor's position, it is easy to be deceived by relay and replay attacks. Considering the complexity of the environment and the layout of different buildings, we summarize the vulnerability of the smart lock models. We consider the possibility that someone might exceed their access rights due to coarse-grained access control.

To solve the above problems, in this paper we propose a model based on Attribute-Based Access Control (ABAC) mechanisms. ABAC is an access control model that allows for dynamically changing properties, such as time of day and location, in access control decisions [20]. Some previous studies [21–25] showed that complex ABAC is better suited to the Internet of Things than Role-Based Access Control (RBAC).

Our proposed Attribute-Based Access Control Mechanism for Smart Locks (AACS), can make use of the attributes related to subject, object, and environment as the basis of authorization policy making, which makes it more flexible for access management. AACS describes the requester and the requested resource through subject attributes and objects attributes, and uses environment attributes to describe access constraints. The administrator's administrative objects are reduced to policies. Only one policy needs to be created or deleted to grant permissions or revoke permissions when members join or leave. In addition, the administrator directly manages the policy set in the smart lock. Therefore, the malicious user cannot retain their rights by disconnecting from the network because the administrator directly modifies the information in the lock. At the same time, we use multiple environment attributes to restrict access to different users to accommodate access control in complex family relationships. At the same time, we use the attribute sRole to associate the authorizers with their subordinates. When cascading deletions are needed, the administrator can remove the members associated with a member by disabling one of the sRole attributes. At the same time, we introduced identity-based encryption to verify the identity of the visitor. Compared with previous methods, the application of AACS can avoid the accidental unlocking in various house designs and effectively prevent attackers from off-line or changing the time of their mobile phones to retain permissions or access the door locks outside the permission time [20]. Finally, our experiment shows that AACS can successfully control access to multiple environment attributes for premises with multiple occupants and achieve controlled cascading deletion of permissions. We also proved through experiments that AACS still has high efficiency in large apartments with many users.

Overall, current smart lock systems lack consideration for possible situations within a complex home, such as multiple identities and group management of team members. We found these problems and designed AACS to deal with some of the problems in complex housing. Future research on smart lock systems should pay more attention to the applicability of smart locks in complex situations. Our contributions are summarized as follows:

- We investigate the limitation of existing smart lock systems, and design a fine-grained access control mechanism for multiple environment attributes;
- We introduced identity-based encryption to verify the identity of the visitor, which makes our scheme safe for different types of smart locks;
- Compared with the existing mechanism, our scheme is more suitable for smart lock systems. It enables group management and cascading deletion of permissions and also prevents possible attacks for current smart lock system;

- We perform a detailed evaluation of our system, which demonstrates that our scheme is practical and efficient for real-world use.

1.3. Organization

In Section 2, we summarize previous work and compare it to our system. We introduce some concepts and point out the problems with intelligent lock systems in Section 3. A brief introduction of our AACS system is given in Section 4. In Section 5, we describe the system framework of our AACS system in detail. Section 6 describes our experimental design and evaluation results. Section 7 gives the conclusion.

2. Related Work

In this section we briefly review the state-of-the-art research on smart locks and its security issues, and compare related work with ours.

2.1. Smart Locks with Biometric Authentication

Regular door locks are cumbersome to unlock and it is easy to lose keys. Smart locks improve usability, accessibility, and security of traditional physical locks, and especially, address the lost key aspect, e.g., no need to change lock and a new "key" can be issued immediately. A former widely-adopted authentication approach in smart locks is to use biometric identification. Specifically, Liu et al. [11] designed an intelligent lock authentication scheme based on behavioral biometrics. The scheme uses three sensors to model the movement of the user's hand in three-dimensional space to generate a signature.

Zhu et al. [12] proposed an efficient face recognition system based on OpenCV. Zhu et al. conducted extensive research on the OpenCV platform and its integration library, using modern and powerful hardware to generate code that correctly and reliably recognizes faces. Ahmad et al. [13] also designed an intelligent door lock system via face recognition. Users can unlock locks with their faces and monitor their status remotely.

Biometric locks are highly secure but require significantly more complex operations than some smart locks that use a digital key on a smartphone. While these schemes focus on authentication they do not consider fine-grained access control for the different roles that may arise in the Internet of Things environment.

2.2. Smart Lock with a Device-Gateway-Cloud (DGC) Structure

In order to adapt to the Internet of Things environment, many manufacturers of smart door locks use a DGC structure. The unlock mechanism of such smart locks is that when the user is authorized to enter the range of the lock, the user's mobile device unlocks the lock via a BLE connection. August and Danalock [16,17] are both automatically unlocked, so the user does not need to do anything else. The Kevo's smart locks need a user's touch to unlock [19]. After the user enters the range of smart locks, they need to touch the dead bolt to start the unlocking process.

As shown in Figure 1, for the smart lock of the DGC structure, a gateway is necessary for the lock to communicate with the cloud. But many homes cannot guarantee a fixed gateway at the location of the smart lock. Therefore, in the smart lock system, the lock usually takes the user's smart phone as the mobile gateway to communicate with the cloud. In this case, when a malicious user takes their phone offline, their digital key cannot be retrieved by the cloud and they can still access the lock.

The work of some researchers has also improved smart locks based on the DGC structure, e.g., Ho et al. [15] proposed a solution to the common problem of unconnected and reserved privileges in DGC structured door locks. This method updates the user list when a user who connected to the network interacts with the lock. In this design, once a reliable user interacts with the lock, the server can update the list of authorized users of the lock.

Patil et al. proposed a new smart lock framework, SecSmartLock [26], which includes an architecture and a secure communication protocol. This scheme is suitable for smart lock with the DGC structure and contributes to the development of intelligent lock security research. Patil et al. implemented a prototype smart lock and simulated an application installed on an Android smartphone. They demonstrate the practicality of the solution and demonstrate the security of the protocol. This framework can effectively prohibit attackers without access rights to gain privileges.

Han et al. [27] proposed an asynchronous communication scheme for malicious users to disconnect the network to avoid the revocation of permissions. The scheme requires users to connect to the cloud every 24 h to obtain a valid 24-hour authentication, which solves the problem of malicious circumvention. Han et al. also provide a lightweight, efficient tree-based access control solution for the smart lock network problem that supports cascading deletion. However, a disconnected attacker can still access the door lock within 24 h, even if someone revokes his privileges, which is clearly insecure.

2.3. Smart Locks That Use Special Communication Technology to Assist Authentication

Using Near-Field Communication (NFC) as the wireless communication channel between the smart lock and mobile device is a convenient way for providing short-range proximity interaction between the user device and lock. Divyashikha et al. [28] studied NFC-based authentication in Internet of Things environment. They proposed a mutual authentication and authentication framework based on NFC security elements in the Internet of Things environment, which adopts the Host Card simulation (HCE) mode based on NFC to conduct Internet of Things access for mobile devices and other user devices. This method of using NFC can be very effective in preventing unintentional unlocking, such as accidentally opening the wrong door, due to the limitation of communication range. This approach is vulnerable to relay attacks and previous work has successfully demonstrated practical relay attacks on NFC [24,29].

To summarize, although the door lock based on biometric authentication enhances the security of the door lock, it lacks the classification control ability of multiple roles. Smart locks with a DGC structure are more suitable for the IoT (Internet of Things), but most of the designed systems rely on the server to manage the rights when the lock is online. Although they can solve the problem of user disconnection to a certain extent, there are still risks in some special cases. Some special technologies have their own disadvantages, such as NFC's vulnerability to relay attacks. In addition, most studies lack the discussion of permission cascade deletion.

3. Problem Definition

In this section, we will briefly introduce the architecture of the DGC systems, and analyze the problems in the architecture of most current smart locks. We also list possible types of attacks against existing smart locks and issues to consider.

3.1. System Architecture

The focus of our research is on the smart lock of the DGC structure. Smart locks of the DGC structure on the market can be regarded as three parts: Electronic door bolts installed on house doors, mobile devices and applications that can connect locks, and remote web servers. As shown in Figure 1, the door lock has no direct connection to the Internet. The door lock connects with the user's device through short range wireless during the process of user unlocking, in our reference design is Bluetooth Low Energy (BLE), and uses the user's device as a gateway to interact with the remote server to push and receive relevant information and updates. The administrator sends permission change instructions to the remote server and the server updates or revokes the key on the user's device current network connection, e.g., WiFi or mobile network.

BLE is a personal Local Area Network (LAN) technology designed and managed by the Bluetooth Technology Alliance. It is designed for emerging applications in health care, sports, location-based services, security, and entertainment. Compared to classic Bluetooth, BLE aims to maintain the same

communication range while significantly reducing power consumption and cost. BLE uses the same 2.4 GHz radio frequency as classic Bluetooth, so dual-mode devices can share the same antenna. Physical proximity can be estimated using the Received Signal Strength Indicator (RSSI)value of the radio receiver, although this is not an absolutely calibrated distance. A typical design is to alert devices when the distance between them exceeds a set threshold.

3.2. Adversarial Model

State Consistency Attacks: Most of the smart locks we studied use DGC structures, in which the lock lack a direct connection to the cloud platform, as shown in Figure 1. Therefore, the only way a smart lock can update its own information is to use the user's device as a gateway. It is easy to believe that a trusted user's device can faithfully send messages between the lock and the remote server. However, we can see that this trust assumption could be inappropriate, as the device of someone whose digital key has been revoked should be considered potentially untrustworthy, e.g., Ho et al. showed how attackers can prevent administrators from revoking their privileges [15]. When an attacker switches his phone to airplane mode or disconnects from the Internet, its permissions cannot be revoked. This behavior blocks the connection with the remote server, preventing the administrator from interacting with the lock.

We focus on theses types of attackers: An attacker who stole 'Alice's' phone, and attackers who were previously held legitimate access rights, e.g., a previous tenant. In order to be applicable to more situations, we have to allow user devices to access the door locks offline. As a result, someone with an ulterior motive cannot disconnect from the server to avoid revoking permission or leaving access logs.

Unauthorized unlock: People using the lock may have many different roles and their access to locks should be limited by time or date. For example, the relatives who come to stay in this home, hourly workers who clean at a fixed time, and so on. However, many smart locks do not provide fine-grained access control for these roles. In addition, if we no longer consider user devices as trusted components of the DGC structure, we should mitigate for cases where the attackers, who already have access rights, can modify system parameters to gain unauthorized access.

For example, an hourly worker is assigned to work between 12:00 and 13:00, but if access control is enforced by the phone and he manages modifying the time information on his device he could unlock the door outside of that time frame.

Cascading deletion of permissions: Due to the complex membership and social relations of occupants, the authorization should not be granted by only one person in the smart door lock system. People with high authority should be able to delegate the ability to grant access to others. Thus, the problem of cascading deletes of permissions arises. For example, Alice previously had granted access to her friend Bob, but now Alice has moved out of the property and her access is revoked. If the system does not have cascading deletion, the permissions of the users authorized by Alice remains unless Alice's housemates check the permissions and finds out whether any were authorized by Alice and manually revokes them. This is undoubtedly complex and unsafe.

3.3. Diversity of Identity

In the home an IoT environment, smart locks should be able to adapt to some complex scenarios of the IoT in addition to security concerns, e.g., for a family, different identities of users require different restrictions.

For different users, smart locks require access control under different conditions depending on their identities. The distinction of these identities is mainly reflected in the limitation of time and date.

The possible roles of smart door locks in homes [15]:

- **Owner**: The head of the household is this identity. He can manage the rights of other users and his rights can only be revoked by the manufacturer of the smart lock. He can view access logs or use administrative functions provided by other manufacturers;

- **Resident**: This status is held by the other occupants of the house. They can access the smart lock at any time just like the owner. However, they cannot view access records or manage permissions for others;
- **Recurring Guest**: People with this status are frequent visitors (such as tutors or hourly workers). They are able to access the door lock at fixed times of the day. These times is set by the owner;
- **Temporary Guest**: This type of user is a temporary guest in the home. They can access the door lock at certain times. For example, out-of-town relatives can access the door lock within three days.

4. System Overview

The core concept of AACS is attributes. In AACS, we divide attributes into four types: S, O, P, and E. S refers to the attributes of the subject attributes of the user who initiated the access and wants to unlock the smart lock, such as sID, sRole, and so on. O represents attributes of the object being accessed (such as a smart door lock or access log), such as oID. P represents the user's permission attributes and represents the various actions that the user can perform, such as unlocking the smart lock or accessing the log. E represents the environment attribute, that is, the environment information when the access control process occurs, such as the time when the user initiates the access and the geographic network location. Users with different identities can be classified and controlled from multiple levels. It is independent of the accessor and the resource to be accessed.

Our system can be divided into two phases according to the type of operation performed: The preparation phase and the implementation phase. The preparation phase is responsible for collecting the set of attributes needed to build the access control system and describing the access control policy. System collects, stores, and manages all attributes required to build security access control in advance and the corresponding relationship between attributes and permissions to form an AA list for system inspection at the execution stage. The execution phase is responsible for responding to access requests and updating access policies.

Compared with smart locks which use RBAC, AACS can solve problems better in complex scenarios. First, attributes are easy to manage. We can easily design attributes for different users. In a home or office, gender, age, position, and rank are all natural attributes. Moreover, attributes are of the K-V type, and a one-to-many table can control the correspondence between subject, object, and attribute. Secondly, the management cost of access control is low. In our system, the administrator's administrative objects are reduced to policy, which deals only with access control. For example, if a new member enters a house, the system administrator can simply create a new policy and write to allow access. After the user leaves, the policy can be deleted or invalidated. In the RBAC's system, it is not that simple.

In the system of AACS, in order to avoid the attack of state consistency, the administrator is required to be within the connection range of the door lock when deleting the permission operation. Figure 2 shows the communication structure of AACS, which is improved on the basis of the DGC structure. The door lock still has no direct connection to the remote server. The difference in AACS is that the administrator needs to make changes to the policy set in the lock instead of sending instructions to the remote server. Such a design requires the administrator to be within the lock range when changing the user's permissions, but this is not a harsh condition for a family. As the policy set inside the door lock is modified directly, the smart door lock no longer relies on the visitor's mobile device as the gateway to update its information. Therefore, an attacker cannot evade permission deletion by taking the mobile device offline. At the same time, in order to ensure the security of AACS, we added encryption authentication before processing the access request, and we will detail the authentication and unlocking process of the user in Section 5.

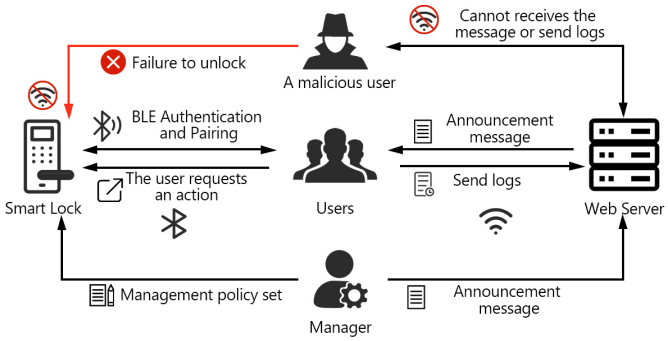

Figure 2. Attribute-Based Access Control Mechanism for Smart Locks (AACS) communication mode. Based on the DGC model, AACS takes the lock as the center and stores information in the lock, while the administrator directly manages the policy set within the lock. After authentication, the user sends access requests containing various attributes for authentication within the smart lock. In this case, malicious users can no longer retain access by disconnecting.

5. System Architecture

5.1. Unlock Process

5.1.1. Identity-Based Cryptographic Authentication

There are many manufacturers of smart locks, most of them small factories, which cannot guarantee the security of the access process because of the cost. Among them, the identity authentication without the application layer will often lead to malicious users impersonating other users by modifying their ID field to unlock. Unfortunately, due to the large number of types, it is obviously unrealistic to confirm the security mechanism for each lock. We hope that the proposed scheme can solve the spoofing problem in the process of door lock authentication without relying on the security of application layer design. Therefore, we introduce identity-based encryption in the authentication process of the scheme [30]. The Identity-Based Encryption (IBE) scheme supports both a single private key generator PKG and multiple users. IBE scheme is composed of IBE algorithms. Setup, IBE. KeyGen, IBE. Extract, IBE. Enc, IBE. Dec.

IBE. Setup(1^k): Inputs security parameter k and outputs common parameter $params$;

IBE. KeyGen$(params)$: Takes common parameters as input, and then outputs master secret/public key pair (msk, mpk);

IBE. Extract(msk, id): Takes the master secret key msk and id as input, and then outputs private key d_{id} for identity id;

IBE. Enc(d_{id}, m): Encryption algorithm takes d_{id} and message m as inputs and an encryption c as outputs;

IBE. Dec(mpk, id, c): On input a ciphertext c, public key msk and id outputs a message m encrypted in c.

Specifically, after the device of user A is connected to the smart door lock through BLE, the APP of user A uses its private key d_A to sign the access request m, obtain $c_A(m)$. Then the APP sends $c_A(m)$ to the smart lock through BLE. Finally, after receiving $c_A(m)$, the smart lock verifies the authenticity of the signature with id_A. If the authentication is successful and the access request comes from user A, the smart lock continues to process the access request m. Otherwise, the device aborts this operation and outputs an error symbol $\perp$.

Through the above scheme, we can ensure that malicious users cannot forge or change their ID information to illegally access the door lock.

5.1.2. Processing of Access Requests

As shown in Figure 3, after the visitor is authenticated, the smart lock records the resulting request m as a Natural Access Request (NAR). Upon receipt of this request, the Policy Enforcement Point (PEP) requests subjects attributes and accesses the AA list to request object attributes and related environment attributes based on the list. Then, PEP builds the Attribute Access Request (AAR) and passes the AAR to the Policy Decision Point (PDP). The PDP determines the user's identity information based on the subject attribute, object attribute, and related environment attribute provided by the AA list. By interacting with the Policy Administration Point (PAP), PDP compares the AAR with the policy set. The PDP determines whether the access request is authorized based on the results of the comparison and transfers the decision body to the PEP. The results of the access determination are finally performed by the PEP. When the environment attributes and permissions of the user's access exist in the policy set, the user can normally unlock the smart door lock otherwise the unlocking operation cannot be completed.

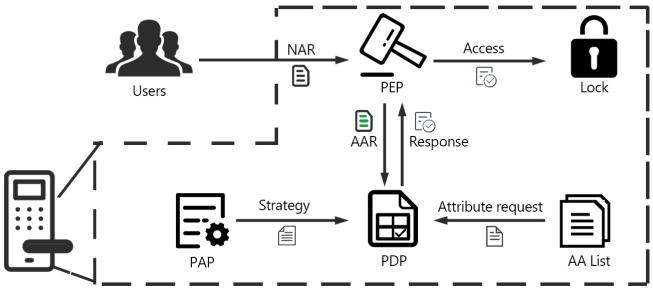

Figure 3. System structure of AACS. NAR: Natural access request, AA: Attribute authority, AAR: Attribute access request, PEP: Policy enforcement point, PDP: Policy decision point, PAP: Policy administer point.

5.2. Attribute Settings

Subject Attribute 1 (sID): In the AACS's system, we use the subject attribute of sID to distinguish between different users. We assign different permissions to different users based on the sID.

One of the most important things about access control is the control of visitors. However, in the existing system, permissions are indistinguishable, and the control of permissions depends on the connection between the user's mobile device and the Internet. Once a device is disconnected from the network or adjusted to flight mode, the information of its permission change cannot be transmitted to the smart lock.

As shown in Figure 2, in the AACS's system, the administrator needs to be in the sensor range of the door lock when changing the permissions of other users. The administrator directly modifies the policy set when changing the permissions of the target user.

Subject Attribute 2 (Position): The smart lock system does not have the ability to identify the user's location. As a result, it is vulnerable to physical attackers. AACS uses the subject property position to pass the user's location information. The location information is used to determine whether the user's location is within the set range of the door lock, thus reducing the possibility of physical attack.

Subject Attribute 3 (sRole): For a truly smart lock, the allocation and recall of permissions requires more consideration. In order to facilitate the application of the family, other members of the family may be required to have the right to assign permissions. In this way, cascading deletion of multiple levels of permissions becomes a necessary function.

AACS associates the dispatcher and the assigned person through the principal attribute sRole. They will have the same sRole attribute. The owner of the home can also revoke their unlock privileges by inserting this sRole's prohibit policy into the policy set.

Environment Attribute 1 (Time/date): This is often the case with traditional smart lock systems, where a visit from a family member requires permission to enter the room within a week or an hourly worker needs to visit at a set time each day. Their access is limited by time and date. In a traditional system, they can change the time of their device to access the door lock at an unauthorized time, which also poses a threat to the family.

By setting the environment properties of time and date, AACS can effectively control the access time and date of members that need to be controlled. Not only that, but also prevent replay attack when the family is traveling.

5.3. System Model

The system of AACS contains four properties: Subject, Object, Permission, and Environment. The Subject represents the properties of the principal. Object stands for the property that accesses object. Permission represents the operation that the subject wishes to perform. Environment represents environment properties at the time the operation is performed. As shown in Table 1, we list the various attributes within the access control system.

Table 1. List of attributes.

Subject: In our system User are the subjects.	
Login Name	P1, P2, P3, P4, P5, P6, P7
Role	owner, resident1, resident2, resident3
Object: Smart door locks and access logs as objects.	
Device as a object	
Dexice	Smart Lock
Information as an Object	
Content-type	Text
Environment	Time, Date, Position
Permission	unlock, read, write

For different types of people, their access policies are different. For example, some of them can open the door lock at any time while some of them are the hourly workers, who are only allowed to enter at certain times of day. All of them have access to the door lock, but they have their own restrictions. According to the different permissions, legitimate parties are divided into four categories: Owner, resident, recurring guest, and temporary guests. Their respective access policies are given in Table 2:

Table 2. Access strategy.

Owner/Resident = "can unlock at all times"	
Subject	Device Identifier, Device Location
Object	Device Type, Device Identifier
Environment	Position
Action	unlock
Recurring guest = "unlock at multiple fixed times"	
Subject	Device Identifier, Device Location
Object	Device Type, Device Identifier
Environment	Device Type, Device Identifier
Action	unlock
Temporary guests = "once-off unlock at fixed time"	
Subject	Device Identifier, Device Location
Object	Device Type, Device Identifier
Environment	Position, Date
Action	unlock

AACS associates the user ID with the role through the AA list, and control the identity permissions of each user. Meanwhile, the AA list connects the identity attribute of the subject with different environment attributes, objects, and actions. The AA list is used to interpret the original access request, and the new access request is composed of environment attributes. Then the policy set determines whether the access is successful or not. The AA list is a list of permissions for different identities.

5.4. Strategy Model

In AACS, the subject and object environment can be constrained by relevant attribute predicates respectively, denoted as SAP, OAP, EAP, with $SAP = sap_1 \wedge sap_2 \wedge sap_3 \wedge sap_n$, $OAP = oap_1 \wedge oap_2 \wedge oap_3 \wedge oap_n$, and $EAP = eap_1 \wedge eap_2 \wedge eap_3 \wedge eap_n$. A policy set can be described with attribute predicates for the subject-object environment.

According to the environmental requirements described above, we give the experimental strategy set, which models a single family household. The strategy set takes into account senior family (owners), other family members (residents), regular visitors like a babysitter or tutor (recurring guests), and other visitors (temporary guests) over a period of time. AACS control the access of people with different identities through environmental attributes such as time, date, and location. We tested the policy set through multiple access requests in the experiment. The contents of the policy table are shown in the experiment section.

5.5. Strategy Evaluation

5.5.1. User Request and Access Control Policy Evaluation

In the policy set, the results of each policy check are collated as follows:

The user's access request, Req, is abstractly defined as a quad $\langle S, O, E, act \rangle$, One $S = \{savp_1, \}$ $\{savp_2, ..., savp_r\}$ is a collection of subject attribute name-value pairs; and $O = \{oavp_1, oavp_2, ..., oavp_r\}$ is a collection of object attribute name-value pairs; $E = \{eavp_1, eavp_2, ..., eavp_r\}$ is a collection of environment attribute name-value pairs. Given access control policy $p = sign(SAP, OAP, EAP, ACT)$ and the user requests the $Req = \langle S, O, E, act \rangle$, when $\|SAP\|, \|OAP\|, \|EAP\|$ are all true and $act \in ACT$, strategies for the user request, $\|p\|$ strategy evaluation results is the authorized logo sign, otherwise, it is not-applicable.

5.5.2. Evaluation of Policy Sets

Given user request Req and policy set $p = \{p_1, p_2, ..., p_n\}$, P the evaluation results of the Req is still a collection, namely $\|P\| = \{\|p_1\|, \|p_2\|, ..., \|p_n\|\}$, with the element after the merger, there are 7 types of values:

$\{permit\}$: All policies in P are positive authorization policy and all applicable to the *Req*; the result is *permit*;

$\{deny\}$: All policies in P are negative authorization policies and apply to *Req*; the final result is *deny*;

$\{not\text{-}applicable\}$: All policies in P are not applicable to the user request *Req*; the system will provide the final decision according to the default authorization. For security reasons, the result is *deny*;

$\{permit, not\text{-}applicable\}$: P has positive policies applicable to *Req*; the remaining policies are not applicable to this request; the result is *permit*;

$\{deny, not\text{-}applicable\}$: P has negative policies applicable to *Req*; the remaining policies are not applicable to this request; the final result is *deny*;

$\{permit, deny\}$: P has positive and negative policies applicable to *Req*, and due to the need of our system, we set the *deny* in the result;

$\{permit, deny, not\text{-}applicable\}$: P has positive and negative policies applicable to *Req*, and some policies are not applicable to this request; the final result is *deny*.

6. Experimental Results and Analysis

In this section we present the implementation of the lightweight AACS designed for the smart lock. In order to prevent the fraud of time and date, we set the time tool in the door lock to conduct the time comparison when retrieving the policy set. We developed the user client application based on Android and used a Raspberry Pi 3 B+ as the door lock.

6.1. Experimental User Settings

We set up a home model with 8 users to test the functionality of AACS. These 8 users included four identities: Owner, resident, recurring guest, and temporary guests. Their relationship is shown in Figure 4. In the family model shown in Figure 4, Alice is the owner of the family, and P1, P2, and P3 are the family members. The four have no time or date limit because they play the occupants of their home. P4 and P6 represent hourly workers invited by the family. We set them as follows: They come to clean the house at a fixed time every day within a certain period of time. P5, P7 represents the visiting relatives or friends in the family: Free access within a period of time. In Section 6.2, we test AACS 'ability to control access to these 8 users.

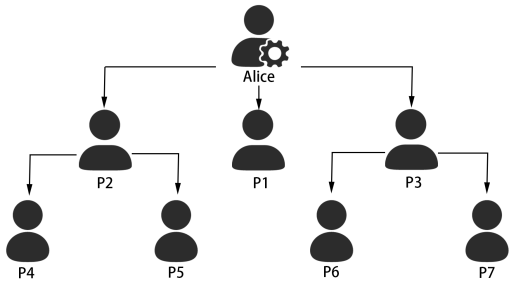

Figure 4. Permission allocation diagram. Alice is the owner of the house. P1, P2, and P3 are regular members of the family (wife, boyfriend and girlfriend, children, etc.), P4 and P5 represent users with rights granted by P2, and P6 and P7 represent users with rights granted by P3.

As shown in Figure 4, the permissions of P1, P2, and P3 are directly assigned by Alice. The permissions of P4 and P5 are assigned by P2, while the permissions of P6 and P7 are assigned by P3. We will discuss the cascading deletion of these roles in the Section 6.3 of this article.

6.2. Fine-Grained Access Control Experiments

We first test whether our system can provide fine-grained access control for different roles. Considering that a family may have visiting relatives, hourly workers, and other roles, in this experiment, we let users of different identities access the smart lock at different times and places to test the functionality of our system. The strategy set of this experiment is shown in Table 3.

In this experiment, we tested whether our system could complete fine-grained access control. Our tasks are shown as follows:

Req_1: srole = owner, sID = Alice, Time = 18:30, Position = near, date = Nov. 11th, unlock.
(Req_1: Alice asked to unlock the smart lock at 18:30 on Nov. 11th when she was near the smart lock.)
Req_2: srole = resident2, sID = P2, Time = 19:30, Position = near, date = Nov. 11th, unlock.
Req_3: srole = resident2, sID = P4, Time = 17:30, Position = near, date = Jun. 1st, unlock.
Req_4: srole = resident2, sID = P4, Time = 13:30, Position = near, date = Sep. 11th, unlock.
Req_5: srole = resident2, sID = P4, Time = 13:30, Position = near, date = Jun. 1st, unlock.
Req_6: srole = resident3, sID = P7, Time = 18:30, Position = near, date = Jun. 1st, unlock.
Req_7: srole = resident3, sID = P7, Time = 22:30, Position = near, date = Jan. 17th, unlock.
Req_8: srole = owner, sID = Alice, Time = 8:30, Position = far, date = Nov. 19th, unlock.

Req_9: srole = resident3, sID = P3, Time = 19:30, Position = far, date = Nov. 11th, unlock.
Req_{10}: srole = resident2, sID = P5, Time = 17:30, Position = far, date = Jun. 1st, unlock.

The result of policy evaluation is shown in the Table 4, "$\surd$" is permit, " $\times$ " is deny, and " $-$ "is not-applicable.

In the task list shown above, we have a total of 10 tests, which are accessed by users with different permissions under different environment properties. We test the feasibility of our system through the access of different users within and outside the permission.

Through the results of multiple access attempts we found that when the user was in a distant place the smart lock could correctly reject the access. Access requests from time-bound users were handled correctly: Access was only allowed for a specified period of time. From the results of this experiment, we can see that we successfully implemented fine-grained access control for users in terms of time, date, and location. And our system is centered around smart locks. The administrator changed the policy set and AA list of the door lock when revoking the permission to prevent the attacker from evading the revoking of the permission. At the same time, the time information was generated from inside the lock to ensure that the access restrictions were adhering to a trusted clock source.

Table 3. The policy set.

	SAP	OAP	EAP			Act	Sign
p_1	sID = Alice	smart lock	0:00 < Network Time < 24:00	Position = near	no limit	unlock	permit
p_2	sID = P1	smart lock	0:00 < Time < 24:00	Position = near	no limit	read	permit
p_3	sID = P2	smart lock	0:00 < Time < 24:00	Position = near	no limit	unlock	permit
p_4	sID = P3	smart lock	0:00 < Time < 24:00	Position = near	no limit	unlock	permit
p_5	sID = P4	smart lock	12:00 < Time < 14:00	Position = near	Jan. 1th < date < Jun. 30th	unlock	permit
p_6	sID = P5	smart lock	0:00 < Time < 24:00	Position = near	Aug. 15th < date < Nov. 15th	unlock	permit
p_7	sID = P6	smart lock	18:00 < Time < 20:00	Position = near	Jul. 1th < date < Dec. 30th	unlock	permit
p_8	sID = P7	smart lock	0:00 < Time < 24:00	Position = near	Jan. 15th < date < May. 15th	unlock	permit

Table 4. Results of strategic assessment.

Requests	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	Result
Req_1	$\surd$	-	-	-	-	-	-	-	$\surd$
Req_2	-	-	$\surd$	-	-	-	-	-	$\surd$
Req_3	-	-	-	-	-	-	-	-	$\times$
Req_4	-	-	-	-	-	-	-	-	$\times$
Req_5	-	-	-	-	$\surd$	-	-	-	$\surd$
Req_6	-	-	-	-	-	-	-	-	$\times$
Req_7	-	-	-	-	-	-	-	-	$\surd$
Req_8	-	-	-	-	-	-	-	-	$\times$
Req_9	-	-	-	-	-	-	-	-	$\times$
Req_{10}	-	-	-	-	-	-	-	-	$\times$

6.3. Cascading Deletion of Hierarchical Permissions

Cascading deletions of hierarchical permissions is another important part of our study. In our opinion, group management and cascading deletion are important components of more convenient access control for smart locks. In our scheme, both the owner and the resident user have the permission to issue permissions. In the experiment we conducted in this part, the sRole attribute is used to cascade delete permissions by associating the upper and lower levels.

In this experiment, we tested the feasibility of cascading deletions in our system. In Figure 4, we show a hierarchical relationship between eight users. We use the sRole attributes to associate the primary family member with the invited member in Figure 4. The sRole attribute of Alice is "manager". The sRole attribute of P1 is "resident1". P2 has the same sRole attribute as P4 and P5: "resident2".

P3 has the same sRole attribute as P6 and P7: "resident3". When we disable access to resident2 on the Role attribute, the three users will simultaneously lose the right to unlock the lock.

After the deletion, the representation in the policy table is shown in Table 5.

In order to clearly reflect the cascading deletion, we performed the following tasks in the policy sets shown in Tables 1 and 3 respectively. We compared the results of the two executions. We can see the results of cascading deletions clearly. The tasks of this experiment are as follows:

Req_{11}: srole = owner, sID = Alice, Time = 18:30, Position = near, date = Nov. 11th, unlock.
Req_{12}: srole = resident1, sID = P1, Time = 19:30, Position = near, date = Nov. 11th, unlock.
Req_{13}: srole = resident2, sID = P2, Time = 19:30, Position = near, date = Nov. 11th, unlock.
Req_{14}: srole = resident3, sID = P3, Time = 17:30, Position = near, date = Jun. 1st, unlock.
Req_{15}: srole = resident2, sID = P4, Time = 13:30, Position = near, date = Jun. 1st, unlock.
Req_{16}: srole = resident2, sID = P5, Time = 13:30, Position = near, date = Nov. 1st, unlock.
Req_{17}: srole = resident3, sID = P6, Time = 18:30, Position = near, date = Sep. 1st, unlock.
Req_{18}: srole = resident3, sID = P7, Time = 22:30, Position = near, date = Jan. 17th, unlock.

For the above tasks, same experiments are carried out on two policy sets, and the results are shown in Tables 6 and 7.

Table 5. The new policy set.

	SAP	OAP	EAP			Act	Sign
p_1	sID = Alice	smart lock	0:00 < Network Time < 24:00	Position = near	no limit	unlock	permit
p_2	sID = P1	smart lock	0:00 < Time < 24:00	Position = near	no limit	read	permit
p_3	sID = P2	smart lock	0:00 < Time < 24:00	Position = near	no limit	unlock	permit
p_4	sID = P3	smart lock	0:00 < Time < 24:00	Position = near	no limit	unlock	permit
p_5	sID = P4	smart lock	12:00 < Time < 14:00	Position = near	Jan. 1th < date < Jun. 30th	unlock	permit
p_6	sID = P5	smart lock	0:00 < Time < 24:00	Position = near	Aug. 15th < date < Nov. 15th	unlock	permit
p_7	sID = P6	smart lock	18:00 < Time < 20:00	Position = near	Jul. 1th < date < Dec. 30th	unlock	permit
p_8	sID = P7	smart lock	0:00 < Time < 24:00	Position = near	Jan. 15th < date < May. 15th	unlock	permit
p_9	sRole = resident2	smart lock	-	-	-	unlock	deny

Table 6. Results of strategic assessment before permission deletion.

Requests	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	Result
Req_{11}	√	-	-	-	-	-	-	-	√
Req_{12}	-	√	-	-	-	-	-	-	√
Req_{13}	-	-	√	-	-	-	-	-	√
Req_{14}	-	-	-	√	-	-	-	-	√
Req_{15}	-	-	-	-	√	-	-	-	√
Req_{16}	-	-	-	-	-	√	-	-	√
Req_{17}	-	-	-	-	-	-	√	-	√
Req_{18}	-	-	-	-	-	-	-	√	√

Table 7. Results of strategic assessment after permission deletion.

Requests	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	Result
Req_{11}	√	-	-	-	-	-	-	-	-	√
Req_{12}	-	√	-	-	-	-	-	-	-	√
Req_{13}	-	-	√	-	-	-	-	-	×	×
Req_{14}	-	-	-	√	-	-	-	-	-	√
Req_{15}	-	-	-	-	√	-	-	-	×	×
Req_{16}	-	-	-	-	-	√	-	-	×	×
Req_{17}	-	-	-	-	-	-	√	-	-	√
Req_{18}	-	-	-	-	-	-	-	√	-	√

In this experiment, 8 users legally accessed the door lock within their own authority. They successfully unlocked the door in the first experiment. However, after we added the no-access policy to sRole "resident2", P2, P4, and P5 failed to unlock while other users were still able to unlock normally. Of course, our experiment only correlated two layers of users. However, it is clear that we can control more layers of users just by adding attributes to correlate them.

6.4. Stress Tests in the Face of Malicious Attacks

In Sections 6.2 and 6.3, we demonstrated that AACS could handle houses with multiple identities and cascade delete users' permissions. In this section, we will show AACS stress tests under unauthorized access, evasion attack, and counterfeit attack. We separately tested the probability of AACS correctly denying malicious access under these three attacks.

The experimental results are shown in Table 8. Experimental results show that AACS can deal with these three attacks well.

Table 8. Probability of successful denial of malicious access.

Times of Attack	Unauthorized Access	Evasion Attack	Counterfeit Attack
100	100%	100%	100%

6.5. Management Stress Tests in Large Apartment Situations

In this section, we tested the applicability of AACS in large apartments with a large number of users. We tested the time required for AACS to process access requests with a user population of 100–2000. Our experimental results are shown in the Figure 5.

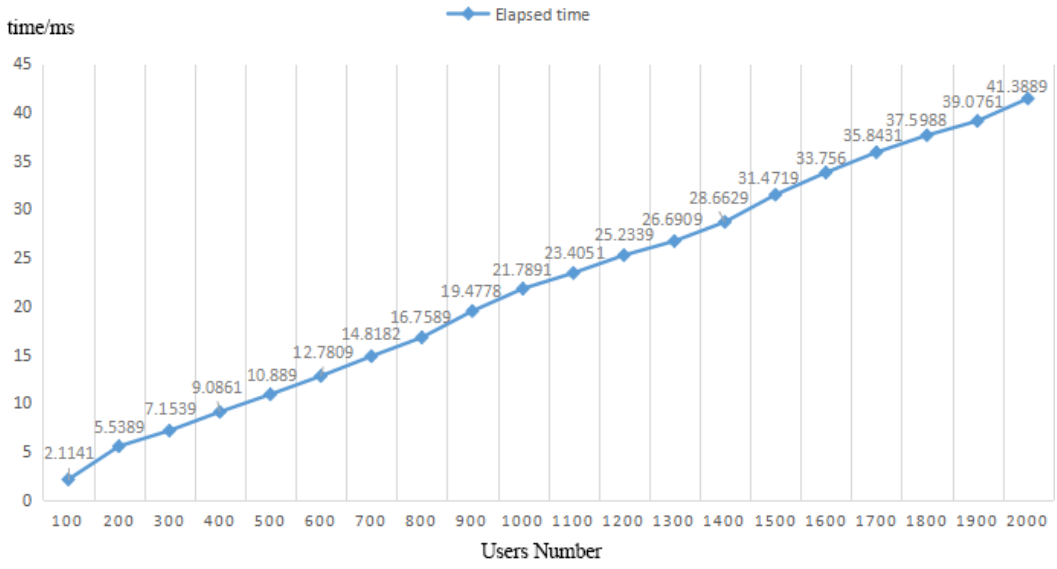

Figure 5. The time it takes AACS to process access requests under 100–2000 user conditions.

The experimental results show that as the number of users increases, AACS takes more and more time to process access requests. However, in the case of 2000 users, AACS took only 41.2889 ms to process access requests, which is enough for people to be unaware. Therefore, AACS also has good applicability in large apartments with thousands of people. Meanwhile, the experimental results show that Raspberry Pi 3 B+ could provide AACS with efficient and stable operating environment in large apartments with 2000 or more users.

6.6. Products Comparison

Finally, we compared AACS with some products on the market. As Table 9 shows, August and Kevo lack design for each of the three problems listed in the table. ENTR® Smart Lock supports multiple unlocking methods such as password, key, BLE, and can provide a time limit for each user. MIUI's smart lock supports multiple ways to unlock. Among them, the one-time password and nanny password can deal with some different identities of users, but is not flexible enough. For state consistency attacks and cascading deletion, MIUI and ENTR® has no solution. By contrast, from the results of the two experiments, it can be seen that our AACS centered on smart lock so that our system could prevent the avoidance of permission revocation and unauthorized access. Secondly, AACS could carry out fine-grained access control for users with different identities. At the same time, for multi-level permission allocation, our system could achieve cascading deletion, which enhances the "smart" of smart locks greatly.

Table 9. Product function comparison.

Products	State Consistency	Different Identities	Cascading Deletion
August	No	No	No
Kevo	No	No	No
ENTR®	No	Yes	No
MIUI	No	Yes	No
AACS	Yes	Yes	Yes

7. Conclusions

In this work, we described some of the challenges and shortcomings of current smart lock systems, and designed an attribute-based access control mechanism to solve these problems. The current problems with smart lock systems are that they cannot cope with malicious users being disconnected to avoid revoking privileges and that they lack fine-grained access control over multiple identities of users in complex homes. In addition, they lack the design of user group management and cascading deletion. In order to solve these problems, we designed an Attribute-Based Access Control Mechanism for Smart Locks (AACS). We restricted users with different identities through different environment attributes and cascade deletions through associating team members with sRole attributes. We also required administrators to directly modify the policy set within the lock to prevent users from evading permission revocation. AACS made better use of urban families with complex members and prevented malicious users from evading permission revocation. However, AACS requires administrators to be near the smart lock when managing member permissions, which can make permission changes less convenient. The smart lock system still has a lot to improve, we hope to continue to pursue in future work.

Author Contributions: Z.X. was responsible for the experiment and the writing of the article, and participated in the design of part of the experiment. L.L. is responsible for the design and supervision of the experiments and the examination of papers. G.H. was responsible for designing part of the experiment and correcting English grammar.

Funding: This research was funded by National Natural Science Foundation OF China: Grant No. 61802180.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yangyong, Z.; Lei, X.; Abner, M. Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 24–27 February 2019.
2. Ayesha, S.; Munam, Ali, S.; Hasan, A.K. Social Internet of Vehicles: Complexity, Adaptivity, Issues and Beyond. *IEEE Access* **2018**, *6*. doi:10.1109/ACCESS.2018.2872928. [CrossRef]

3. Yanbiao, L.; Zhiyi, Z.; Xin, W. A Secure Sign-On Protocol for Smart Homes over Named Data Networking. *IEEE Commun. Mag.* **2019**, *57*. doi:10.1109/MCOM.2019.1800789. [CrossRef]
4. Daemin, S.; Keon, Y.; Jiyoon, K. A Security Protocol for Route Optimization in DMM-Based Smart Home IoT Networks. *IEEE Access* **2019**, *7*. doi:10.1109/ACCESS.2019.2943929. [CrossRef]
5. Waheb, A.; Tee, K.K.; Roshahliza, M. Design and Fabrication of Smart Home With Internet of Things Enabled Automation System. *IEEE Access* **2019**, *7*. doi:10.1109/ACCESS.2019.2942846. [CrossRef]
6. Mercedes-Benz Design. Available online: https://www.mercedes-benz.com/en/mercedes-benz/design/ (accessed on 24 July 2018).
7. Yuan-Chih, Y. A Practical Digital Door Lock for Smart Home. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–15 January 2018.
8. Rui, Z.; Yunfei, T.; Zhang, H. Designing Cooperative User Experience for Smart Locks. In Proceedings of the International Conference on Cooperative Design, Visualization and Engineering, Berlin, Germany, 2–5 September 2018.
9. Sandeep, G.; Attaullah, B.; Bruno, C. SmartHandle: A Novel Behavioral Biometric-based Authentication Scheme for Smart Lock Systems. In Proceedings of the 3rd International Conference on Biometric Engineering and Applications, Stockholm, Sweden, 29–31 May 2019.
10. Kazuki, W.; Makoto, N.; Kentaro, A. Gait-Based Authentication for Smart Locks Using Accelerometers in Two Devices. In Proceedings of the International Conference on Network-Based Information Systems, Oita, Japan, 5–7 September 2019.
11. Chun-Yu, L.; Shanq-Jang, R.; Lai, Y.R. Finger-Vein as a Biometric-Based Authentication. *IEEE Consum. Electron. Mag.* **2019**, *8*, 29–34. doi:10.1109/MCE.2019.2941343. [CrossRef]
12. Zhiguo, Z.; Cheng, Y. Application of attitude tracking algorithm for face recognition based on OpenCV in the intelligent door lock. *Comput. Commun.* **2020**, 390–397. doi:10.1016/j.comcom.2020.02.003. [CrossRef]
13. Azlan Mohamed, A.S.; Mohd Nadhir Ab Wahab, S.R.K. Facial Recognition Adaptation as Biometric Authentication for Intelligent Door Locking System. In Proceedings of the International Visual Informatics Conference, Bangi, Malaysia, 19–21 November 2019; pp. 257–267.
14. Chao, L.; Debiao, H.; Neeraj, K. HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes. *IEEE Internet Things J.* **2020**, *7*, 818–829. doi:10.1109/JIOT.2019.2944400. [CrossRef]
15. Ho, G.; Leung, D.; Mishra, P. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIACCS), Xi'an, China, 30 May–3 June 2016; pp. 461–472.
16. August. Available online: https://www.august.com/ (accessed on 24 July 2018).
17. Danalock. Available online: https://www.danalock.com/ (accessed on 24 July 2018).
18. Okidokeys. Available online: https://www.okidokeys.com/ (accessed on 24 July 2018).
19. Kwikset Kevo Smart Lock. Available online: http://www.kwikset.com/kevo/default (accessed on 24 July 2018).
20. Coyne, E.; Weil, T.R. ABAC and RBAC: Scalable, Flexible, and Auditable Access Management. *IT Prof.* **2013**, *15*, 14–16. doi:10.1109/MITP.2013.37. [CrossRef]
21. Goyal, V.; Pandey, O.; Sahai. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006.
22. Antonio, L.N.; Yuri, L.; Artur, L. Demo Abstract: Attributed-Based Authentication and Access Control for IoT Home Devices. In Proceedings of the 17th ACM International Conference on Information Processing in Sensor Networks (IPSN), Porto, Portugal, 11–13 April 2018.
23. Gupta, M.; Benson, J.; Patwa, F. Secure Cloud Assisted Smart Cars Using Dynamic Groups and Attribute Based Access Control. *arXiv* **2019**, arXiv:1908.08112.
24. Francis, L.; Hancke, G.; Mayes, K. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In Proceedings of the 6th International Conference on Radio Frequency Identification: Security and Privacy Issues, Istanbul, Turkey, 8–9 June 2010; pp. 35–49.
25. Fernández, M.; Mackie, I.; Thuraisingham, B. Specification and Analysis of ABAC Policies via the Category-based Metamodel. In Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, Dallas, TX, USA, 25–27 March 2019; pp. 173–184.

26. Patil, B.; Vyas, P.; Shyamasundar, R.K. SecSmartLock: An Architecture and Protocol for Designing Secure Smart Locks. In Proceedings of the International Conference on Information Systems Security, Bangalore, India, 17–19 December 2018; pp. 24–43.
27. Zhaoyang, H.; Liang, L.; Zhe, L. An efficient access control scheme for smart lock based on asynchronous communication. In Proceedings of the ACM Turing Celebration Conference-China, Chengdu, China, 17–19 May 2019.
28. Sethia, D. NFC Secure Element-Based Mutual Authentication and Attestation for IoT Access. *IEEE Trans. Consum. Electron.* **2018**. doi:10.1109/TCE.2018.2873181. [CrossRef]
29. Francis, L.; Hancke, G.P.; Mayes, K. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *Cryptol. Inf. Secur.* **2011**, 618. doi:10.3233/978-1-61499-143-4-21. [CrossRef]
30. Siad, A.; Amara, M. A new framework for implementing identity-based cryptosystems. *J. Syst. Softw.* **2016**, *118*, 36–48. doi:10.1016/j.jss.2016.04.059. [CrossRef]