SITUATION-AWARENESS INTERACTIVE SELF-CHECK-IN SERVICES BY SMARTPHONES AND SMART LOCKS

John Yoon[1] and Jin Sup An[2]
[1]*Dept of Math & Computer Sciences, Cybersecurity Programs, Dobbs Ferry, New York, USA*
[2]*Dept of Information Technologies, SMA Group, New York, New York, USA*

ABSTRACT

One of the secure and intelligent functionalities in smart cities is a so-called smart lock and its operability. Smart door locks have been developed and deployed in home or hotel doors, building entrance doors, car doors, elevators, etc. It can be locked and unlocked by smartphone applications as well as it can be globally managed by a cloud server. However, the level of smartness in such door locks and its operability is not satisfied in part because the so-called smart lock is not inactive to the smartphone applications, and in part because the lock and the smartphone do not aware the situations surrounding one another. This paper surveys the current technologies and employ sensor and restful API server technologies to the smart lock development and the lock/unlock interoperability. The door locks will be smarter as the situations surrounding the smartphones is recognized or as the credentials of the human is verified. The most reliable requirement for smart self-check-in services includes 1) the appropriate services to guest's lock requests, and 2) the efficient and situation-awareness services in self-locking/-unlocking doors. To achieve the requirement, this paper proposes ways of resolving the errors by interoperating with sensors, and ways of improving the smartness of locks by interactive communications with a cloud server and smartphone applications. The contribution includes an improvement of the security and intelligence in smart self-check-in services.

KEYWORDS

WIFI, BLE, Smartphone Applications, PHP Servers, Smart Door Locks, Sensors and Actuators

1. INTRODUCTION

As sensors are widely used, the advances in the sensor technologies are profound and the power of microcontrollers gets stronger (J. Wilmsdorff, J. Kolf, A. Kuijper, 2022, T. Kalsoom, N. Ramzan, S. Ahmed, and M. Ur-Rehman, 2020, G. Baluyot, S. Chen, J. Villaverde, D. Chung, 2019, P. Poudel, B. Ray, A. Milenkovic, 2021), various devices can become connected one to another wirelessly and they behave intelligently in response to the change of (surrounding) data states (J. Xiao, H. Li, M. Wu, H. Jin, M. Deen, J. Cao, 2023, L. Xin, J. Becker, S. Gvozdenovic, D. Starobinski, 2019, T. Renzler, M. Spork, C. Boano, K. Romer, 2018). One of such devices is a smart lock. A smart lock is widely deployed in residential homes, hotels, and Airbnb (www.airbnb.com), automobiles, etc. (X. Zhang, M. Song, Y. Xu, Z. Dai, 2021). At a simple case, some of those smart locks are locked and unlocked by smart card, keypad, or smartphones. At another extreme end, as they are complicated, some of the smart locks are registered in a remote cloud system and a database. They are saved and managed in a server, and their profiles and accessibilities are controlled remotely from a cloud server. Lock access control is verified at the server side, and the decision is transferred to the smart lock and the guest's smartphones.

The simple smart locks are installed at residential homes and automobile cars, elevators, while in the complex case of multiple doors, those complicated smart locks are installed on hotel doors or Airbnb doors. The smart locks can be locked or unlocked by wireless communications with the web applications (or web App) from a server or the mobile applications (or mobile App) on a smartphone.

The smart door locks are equipped with a microcontroller which can run a program to switch the power supply in response to the request from a user (either through a web App or a mobile App). The smart door locks can communicate with other devices: in TCP protocol over WIFI with mobile applications (see 1 in Figure 1), in HTTP protocol with cloud server (see 1 and 2 in Figure1), and ad-hoc connection with mobile applications

(see 3 in Figure 1). The ad-hoc connection can include a smart key card thru near-field communication (NFC), Bluetooth Low-Energy (BLE), Z-wave or Zigbee for home-devices for example.

There are two types of user interfaces (aka applications) to communicate to smart door locks: (1) Web App, where a web browser on a thin client enables a hotel guest to activate the door unlock request; (2) Mobile App, which runs on a smart phone to enable a hotel guest to click the door unlock button.

- A Mobile App presents an unlock request to the smart door lock. The request is transmitted to a TCP socket server in the smart door lock. The TCP socket server can receive lock requests from multiple socket clients (which means multiple guests or keys in the same room). This is illustrated in 1 in Figure 1.
- A Web App requests first unlock the lock to a cloud server (see 2 in Figure 1). Then, the server determines the door access to the smart door lock. The web server runs in a cloud server where a database contains the guests' profiles.

The former explains how the smart door lock functions directly by a Mobile App, and how it functions by a Web server which is requested by a Web App. Note that 3 in Figure 1 will be explained later.

This paper is motivated by the following motivating examples.

MOTIVATING **EXAMPLE 1** (*WIFI Shutdown*): Consider Figure 1, where a hotel guest tries to open the hotel door lock. When the guest approaches the cell phone to the door lock, if the cloud database does not respond (see 2 in Figure 1 has an issue), the request cannot be progressed, and the smart door will not respond (see 1 in Figure 1 where the router is down). In this case, rather than leaving the failure unfixed, the other route of data communication (possibly 3 in Figure 1). How can this be possible?

MOTIVATING **EXAMPLE 2** (*Intrusion Attack*): An attacker tries to open a hotel door lock as if the attacker presents as a legal hotel guest. The attacker requests to hotel door unlock through 1 to 1 in Figure 1 using a Web App or a Mobile App. If the attacker is not present in front of the hotel door, the server should not open the door for the attack guest. In this case, how can the server discern the attacker from the guest who must be in front of the hotel door?

The contribution described in this paper includes a TCP socket server that is equipped in a smart door lock as well as a cloud database server to cover the complexity of multiple doors, and an ad-hoc connections for emergency cases. We also employ sensor technologies and cloud database servers to improve the security and intelligence of self-check-in services.

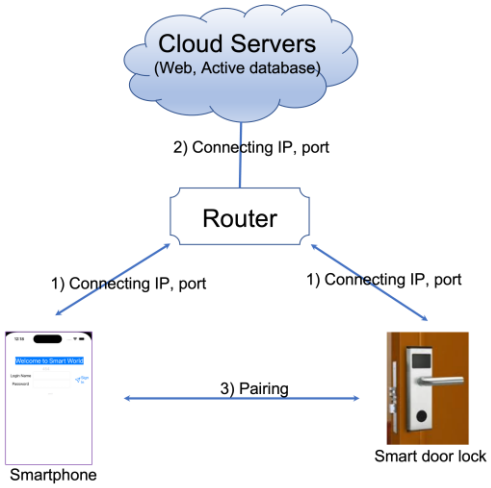


Figure 1. A self-check-in process in protocol view

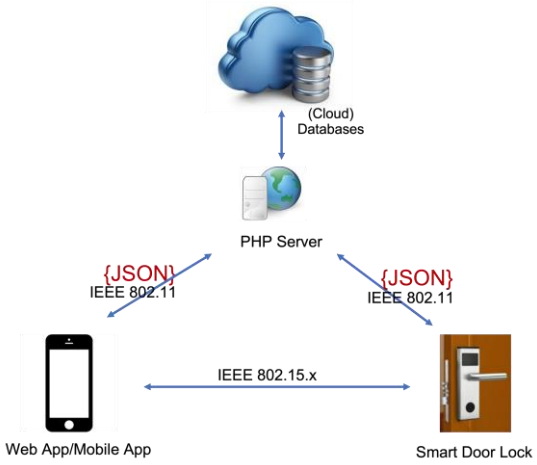


Figure 2. Global view of the architecture

2. PRELIMINARIES & RELATED WORKS

2.1 Smart Locks

Door locks become smart when they have both computing power and wireless connectivity. It is likely that a so-called smart door lock has a microcontroller embedded in a hardware lock fixture that has computing and

communicating power as illustrated in Figure 1. An example of the microcontroller may be a Raspberry Pi (X. Zhang, M. Song, Y. Xu, Z. Dai, 2021) or a Pico W version, which can connect other devices thru WIFI.

A door lock will be smarter if the lock guides interactively the usage of devices which are held by the guest, or if the lock collaborates with one or more sensors (G. Bai, Y. He, C. Zhao, 2021) that can identify the ambient conditions to the door and the guest.

2.2 Wireless Communications

Theoretically, all types of radio waves in various frequency ranges can be used to communicate between the guest's smartphone and the door locks. IEEE 802.11-based WIFI is a typical wireless protocol to smart door locks (see 1 in Figure 1). While a network router assigns an IP address to each of the smart door locks and each of the Mobile Apps, TCP/IP network connections are possible. We can consider an ad-hoc network connection. IEEE 802.15-based wireless signals, e.g., BLE, ZigBee, etc., are also possible protocols (see 3 in Figure 1). A smart door lock and a Mobile App can be connected each other by pairing the key.

2.3 Web Apps and Mobile Apps

An interface to smart door locks can be available on a web browser and on a smartphone device, denoted as Web App and Mobile App, respectively.Web Apps are thin applications running on a web browser at a client, where the main logics are running at a server. The languages used to implement Web Apps are HTML, CSS, JavaScript including PHP. If user profiles are stored in a database, as a user requests to unlock a door lock, the user's access is verified based on the database. Then, the PHP server converts the verification result into a JSON data (D. Gope, D. Schlais, M. Lipasti, 2017). The JSON data will then be forwarded to a Web App and a smart door lock.

Mobile Apps are thick applications that run in their own implemented logics to request and respond to a door lock and a remote server. Mobile Apps are implemented in XCode Swift for iPhone devices or in Java for Android devices. They can communicate with a database through a PHP server by sharing data in a JSON format (I. Hrubaru, G. Talaba, M. Fotache, 2019) or communicate directly with a door lock through a socket coding. Of course, for the former communications, a PHP server at the cloud server should be implemented to receive the request from or send out the reply to a smartphone, while for the later communications, socket codes should be implemented on a smart door lock and open a port to receive TCP or UDP packets from or send encoded packets to a smartphone.

2.4 Sensors and Actuators

For security and intelligence improvement, sensors and actuators are employed (V. Modekurthy, M. Rahman, A. Saifullah, 2023, X. Cheng, M. Sha, 2023). The sensors that can be used for self-check-in services include thermal cameras, distance sensors, geo-code from GPS, motion detection sensor, etc. A thermal camera indicates the temperature of an object. Using it, the presence a human can be detected. A distance and motion sensor can detect the proximity of an object in front of the door lock. These sensors can be equipped and embedded in a smart door lock.

Since smartphones have a few of sensors available. The GPS sensor equipped in a smartphone can detect a geo-code. The geo-code can be used to determine the closeness of the smart door lock and a smartphone.

An actuator is a device that can react in response to the result of sensing. Monitoring the environment by one or more sensors may lead to take an action of an actuator. Sensing and actuating is coded at the smart door locks.

3. ARCHITECTURE

Our proposed approach consists of 4 major services: database service, web service, application services and smart lock service, which appear to be connected in Figure 2. The self-check-in services discussed in this paper relies on IEEE 802.11, WIFI signals to carry requests and responses among those four services. In addition to WIFI, cell signals are available to carry data for smartphones. Since smartphones have additional radio

frequencies and sensors equipped, the ad-hoc signals in IEEE 802.15.x such as Bluetooth, GPS, NFC, etc. can be utilized.

The PHP services running on web servers 1) provide API's to relay the data between Web and Mobile Apps and Smart door locks, and 2) connect the backend databases to pose queries and receive answers to convert into JSON. The database queries are formed at the PHP services for the requests from Web/Mobile Apps.

In the normal situation, a smart door lock opens a port to receive a request packet from a guest. The Mobile App at the guest side has an IP assigned. In this situation, a guest activates the Mobile App to transmit the unlock request to the TCP socket server (to a specific IP and the port opened) at the smart door lock.

When it comes to multiple locks required, e.g., in a big hotel or in a large complex village, the security, safety and accuracy becomes the additional requirement for self-check-in services. Both the smart door locks and Mobile Apps connect to the PHP server to request additional data from the cloud database. Since the master data is available and maintained in the database, any services that require interrelationships among the guests and the smart locks need to communicate with the database through the PHP server. Note that the PHP server provides all Restful API's and connects to the database.

In an emergency, e.g., power outrage, network disconnection, etc., an ad-hoc network service, e.g., IEEE 802.15.x can be available. To cope with an emergent case, a smartphone needs to activate a special module to communicate with the smart door lock through Zwave, ZigBee, Bluetooth, etc. The communication in this protocol does not need a router.

4. SELF-CHECK-IN SERVICES

Our This section discusses three approaches of self-check-in services where smartphones and smart door locks are communicating through IEEE 802.11 WIFI signals or IEEE 802.15.x ad-hoc signals (IEEE 802.11 Standard, IEEE 802.15.x Standard). Depending on combination of services illustrated in Figure 1, we consider three approaches for smart door lock services: (1) Access control based on WIFI communications, (2) Centralized access control, and (3) Access control based on ad-hoc wireless communications.

4.1 Access Control Over WIFI – At Normal Situation

This approach does not refer to the authentication data in a remote database but makes it possible to communicate between the smart door lock and a Web or Mobile App. A network router enables them to IP addresses and the programs on the smart door lock and a Mobile App should open the IP with a port to communicate. This is illustrated in Figure 3(a).

4.2 Centralized Access Control – For Scaling Up

Consider a database that contains the information about all the guests and smart door locks. Data stored in a database can be accessed by Web and Mobile Apps and the smart door locks through a PHP server. The PHP server can function as a Restful server, which provides API's for the Web and Mobile Apps and for smart door locks.

In this approach, as a guest needs to open a door lock, a request for door opening is initiated on either Web or Mobile Apps. See Figure 3(b). The request can be transmitted to a web server through WIFI or a cell tower wirelessly. The web server includes the PHP server, which takes the request and relay it to the backend database. Since the database contains user profile table and authentication table, the answer to the request is returned to the PHP server. Then, the PHP server converts the result into the JSON data. The JSON data consists of the authentication for a given request as well as more data which will be used for self-check-in service improvement.

This approach is useful and necessary for a large scale of smart door locks, for example hotel room door management. Using this approach, a hotel with over hundred rooms can be managed automatically without having to much human interventions or manual services.

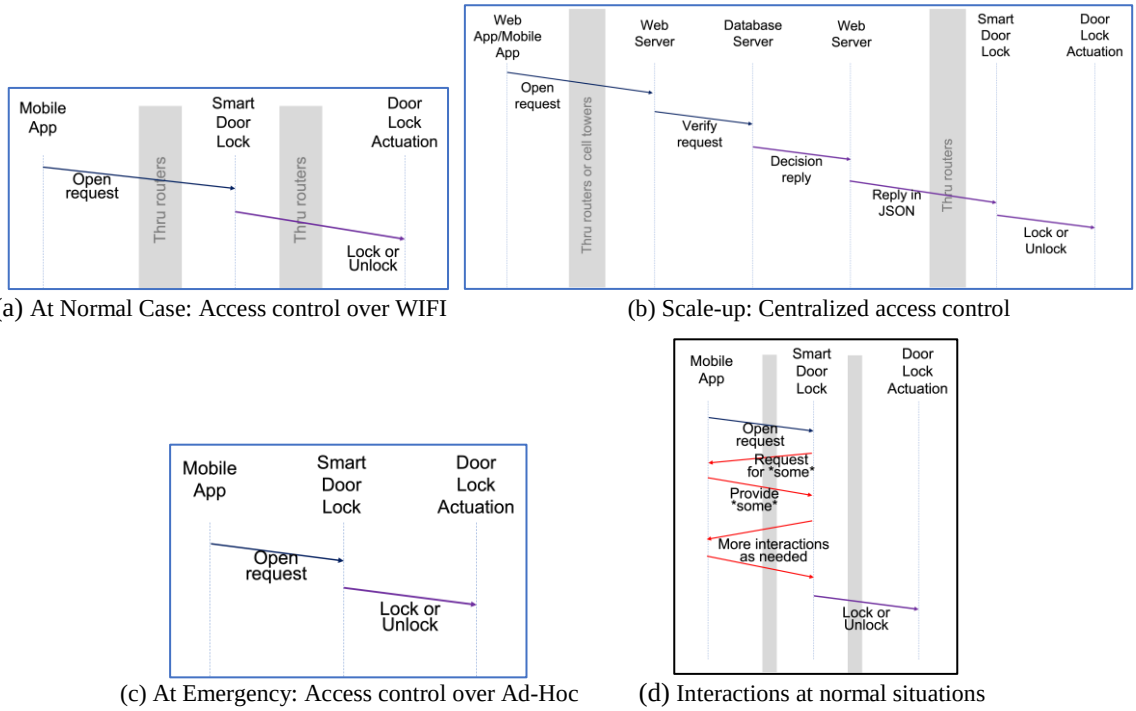
4.3 Access Control Over Ad-Hoc – At Emergency

Another approach does not use the network router WIFI protocol but uses ad-hoc communications. It means that this approach is appropriate for the case of emergency, e.g., power outrage or no WIFI available. Relying on IEEE 802.15.1, IEEE 802.15.2, or IEEE 802.15.4, ZigBee or BLE signal carries the lock or unlock request from a smartphone App. The program development and deployment in this approach is similar to the other approaches above. This approach is illustrated in Figure 3(c).

4.4 Sensor-Driven Improvement

Recall the sensors briefly introduced in Section 2.4. When a Web or Mobile App requests to unlock or lock the smart door lock, not only an authentication of the guest's credential but also one or more sensors can be installed at the smart door lock. For example, consider a distance sensor that measures the distance from an object. Only if the distance is in an expected range, additional authentication can be further processed.

The following is part of the TCP socket coding running on the smart door lock, which is the second approach as illustrated in Figure 3(d).



(a) At Normal Case: Access control over WIFI

(b) Scale-up: Centralized access control

(c) At Emergency: Access control over Ad-Hoc

(d) Interactions at normal situations

Figure 3. Approaches for self-check-in services

```
    # some segments omitted
    # the following computation depends on the specification of a distance sensor
    pulse_duration = pulse_end_time - pulse_start_time
    distance = round(pulse_duration * 17150, 2)

    if distance < 50:    # if the distance is shorter than 50Cm
       good2To = False
       return True
    else:
       reply5 = 'None appears in front of the door.'
       conn.send("From Lock: {}".format(reply5).encode())
       # interactively reply to guest's Web or Mobile App
       time.sleep(15)
       reply6 = 'Please approach to the hotel door lock'
          conn.send("From Lock: {}".format(reply6).encode())
       # this interaction can be iteratively provided for each different sensor data
```

As illustrated above, when the distance is beyond one half meter (or approximately 2 feet), the smart door lock can ask the guest to proceed closer to the door lock. This interaction can take place in response to various sensor's data.

5. CONCLUSION

Smart door locks presented in this paper become smarter and more intelligent than typical (commercial) locks. We have proposed to a WIFI-based TCP socket server that can be equipped in a door lock. For the management of multiple complicated door locks, a cloud server and PHP server can be involved. For an emergency case, e.g., WIFI is unavailable, or power supply is down, the ad-hoc network service can be activated to provide a seamless self-check-in service.

The functions that make the smart door locks smarter can be still improved by including sensors and machine learning technologies. Depending on sensors to monitor, interactions between the smart door locks and Mobile Apps can be made. Depending on the complexity of door locks, machine learning techniques can be applied to massive transactions of numerous guests' movements to determine the verification of guests' credentials and movements and then the result can be downloaded to the smart door locks. Then, the smart door locks interact with the guest's smartphone for additional guidance. The contribution includes an improvement of the security and intelligence in smart self-check-in services.

REFERENCES

Bai, G., He, Y., Zhao, C. (2021). "Research on task collaboration framework of mobile intelligent sensor cluster based on edge computing architecture," Proc. of the 1st Int'l Conf. on Control and Intelligent Robotics, pp. 86-91.

Baluyot, G., Chen, S., Villaverde, J., Chung, D. (2019). "Android application-based electrocardiogram design using microcontroller and bluetooth technology," Proc. of the 19th Int'l Conference on Biomedical Engineering and Technology, pp 27-30.

Cheng, X., Sha, M. (2023). "Autonomous traffic-aware scheduling for industrial wireless sensor-actuator networks," ACM Trans. On Sensor Networks (TOSN), vol. 19, pp. 1-25.

Gope, D., Schlais, D., Lipasti, M. (2017). "Architectural support for server-side PHP processing," Proc. of the 44th Annual Int'l Symposium on Computer Architecture, pp. 507-520.

Hrubaru, I., Talaba, G., Fotache, M. (2019). "A basic testbed for JSON data processing in SQL data servers," Proc. of the 20th Int'l Conf. on Computer Systems and Technologies, pp. 278-283.

IEEE 802.11 Standard, https://standards.ieee.org/ieee/802.11/7028/

IEEE 802.15.x Standard, https://standards.ieee.org/ieee/802.15.1/3513/, https://standards.ieee.org/ieee/802.15.3/6211/ and https://standards.ieee.org/ieee/802.15.4/7029/

Kalsoom, T., Ramzan, N., Ahmed, S. and Ur-Rehman, M. (2020). "Advances in Sensor Technologies in the Era of Smart Factory and Industry 4.0," Sensor, MDPI, vol. 20, pp. 1–22.

Modekurthy, V., Rahman, M., Saifullah, A. (2023). "Towards mixed criticality industrial wireless sensor-actuator network," Proc. of the 24th Int'l Conf. on Distributed Computing and Networking, pp. 425-430.

Poudel, P., Ray, B., Milenkovic, A. (2021). "Microcontroller fingerprinting using partially erased NOR flash memory cells," in ACM Transactions on Embedded Computing Systems (TECS), vol. 20, pp 1-23.

Renzler, T., Spork, M., Boano, C., Romer, K. (2018). "Improving the efficiency and responsiveness of smart objects using adaptive BLE device discovery," in Proc. of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects, pp 1-10.

Wilmsdorff, J., Kolf, J., Kuijper, A. (2022). "Reducing deployment cost for passive electric field sensors," Proc. of the 7th Int'l Workshop on Sensor-based Activity Recognition and Artificial Intelligence, pp 1-8.

Xiao, J., Li, H., Wu, M., Jin, H., Deen, M., Cao, J. (2023). "A survey on wireless device-free human sensing: application scenarios, current solutions and open issues," ACM Computing Surveys (CSUR), vol. 55, pp 1-35.

Xin, L., Becker, J., Gvozdenovic, S., Starobinski, D. (2019). "Benchmarking the physical layer of wireless cards using software-defined radios," in Proc. of the 22nd Int'l ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp 271-278.

Zhang, X., Song, M., Xu, Y., Dai, Z. (2021). "Intelligent door lock system based on Raspberry Pi", Proc. of the 2nd Int'l Conf. on Artificial Intelligence and Information Systems, pp. 1-7.