Course Project

A smart lock is an electromechanical lock that is designed to perform locking and unlocking operations on a door when it receives such instructions from an authorized device using a wireless protocol and a cryptographic key to execute the authorization process. It also monitors access and sends alerts for the different events it monitors and some other critical events related to the status of the device. Smart locks can be considered part of smart home systems[1].

There have been industrial standards of smart home protocols[2], but what we need is a much simpler version. Assume that we are designing a smart home system that contains 1) a smart lock, 2) a home gateway, and 3) a client. The client communicates with the home gateway, while the gateway communicates with the lock. The smart lock system should provide the following functionalities:

1. lock sends heartbeats and operation logs to the gateway, to be reviewed by the user on the client.
2. Lock & unlock if given the correct password (permanent or temporary).
3. Activate or deactivate the temporary password (once used to unlock the door, the temporary password should be disabled automatically).
4. Send notifications if someone breaks the lock.
5. For user operations like 2 and 3, return operation results, and error messages to the client.

This project requires to implement network protocols and set up message exchange between multiple devices. The functional requirements mentioned above are **NOT** the focus point, as this is not a software engineering course. Instead, focus on designing the communication protocols to satisfy the non-functional requirements. For example, regular heartbeat should be sent in an energy-efficient way, while the requirement on reliability is low (some packet losses are acceptable), so that the simplest way to implement the heartbeat would be sending UDP packets. The basic non-functional requirements to consider include latency, reliability, security, and energy efficiency.

The coursework and the checkpoints will build a solid foundation for the successful completion of a smart lock. This course project will be carried out in groups of 3-4 people. Each group will

1) Design and implement an application protocol for smart locks. The implementation can be based on any existing protocols (socket, HTTP, XML, SOAP, RESTFUL, MQTT, CoAP, etc.), as long as it works and you can *justify your design choices*.

2) Review the design and implementation of other groups.

The course project is divided into checkpoints, that require the implementation of different protocols in a client-server architecture. The details of each checkpoint would be provided during the semester.

The following is a tentative schedule for the checkpoints, which is subject to change under extraordinary circumstances.

Due Week	Particular	Brief description	Type
Week 2	Checkpoint 1	• Set up virtual machine • Install HTTP server • Wireshark installation and analysis.	Individual
Week 5	Checkpoint 2	• Implement a custom HTTP server • Find your group members	Individual
Week 8	Checkpoint 3	• setup port forwarding • explore MQTT/coAP/WebSocket • measure their performance using Wireshark	Group
Week 10	Checkpoint 4	• design your smart lock system and protocol • Consider the security aspects • Discuss with Professor	Group
Week 13	Checkpoint 5	• Implement your design • Final presentation • Final project report	Group

References:

[1] "Your Door Is About to Get Clever: 5 Smart Locks Compared" (Links to an external site.) *Wired* (Links to an external site.) 2013-06-19.
[2] https://www.electropages.com/blog/2019/02/smart-homes-explained-smart-home-protocol (Links to an external site.)
[3] https://www.einfochips.com/blog/a-quick-guide-to-understanding-iot-application-messaging-protocols/ (Links to an external site.)

[4] polling, SSE, and websocket, https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9 (Links to an external site.)

[5] Application Messaging Protocols for IoT https://www.einfochips.com/blog/a-quick-guide-to-understanding-iot-application-messaging-protocols/