IoT Engineer C Test Project

Description

Design and **implement** a **secure** communication protocol by considering the following facts.

- We need to support up to **300** different **commands**
- Each **command** may have up to **200bytes** payload data
- Some commands may respond with a list of **thousands of items**, so we need to encrypt and send the response as we generate them, because we can not keep the whole response in memory.
- Memory is limited, try to keep the memory usage minimum
- No need to implement the client side, you can mock the client request if you want to.
- The project should be implemented using the C language
- No need to have a working project, but please make sure not to have any syntax errors.

- **Assume** there are **encrypt** and **decrypt** functions with the following signatures, note that they can only accept data up to **256 bytes** each time. (assume the output length is the same as the input length and they handle all security risks internally)
    - **void encrypt(uint8_t* input, uint8_t length, uint8_t* output);**
      void decrypt(uint8_t* input, uint8_t length, uint8_t* output);

- **Assume** there is a communication layer that provides the following methods:
    - **void Communication_onDataReceived(uint8_t* data, uint16_t length);**
    - **void Communication_sendData(uint8_t* data, uint16_t length);**

- You need to provide the following functions, so we can use them in the command handler logic:
    - Your logic will call this function when a request is received and decrypted
        - **void CommandHandler_handle(actionId, actionPayload);**
    - We will call this function to start the response
        - **void Communication_openResponse();**
    - We will call this function to append a list item to our response
        - **void Communication_appendResponse(uint8_t* data, uint8_t length);**
    - And once we are done with the response we will call this function to close the response and flush the buffers if needed.
        - **void Communication_closeResponse();**
- The function names in this document are just examples and you can use whatever you want.

Evaluation

It will be great if you can do all the requirements, however, if you don’t have that much time, don’t worry, it is ok not to **implement** all parts but please write down your solution for not implemented parts.

We will consider these while reviewing your project.

- Project structure and architecture
- Clean code
- Performance, memory usage
- Your solutions (problem-solving)
- Version control (commit messages, …)

Example of data flow:

- A client connects to the device
- The client builds a request based on your protocol and sends it to the device
- **Communication_onDataReceived** function will be called with the request data
- You need to write logic to parse this data and extract the action id and action payload then call the **CommandHandler_handle** function with the action data.
- Assume the **CommandHandler_handle** function is implemented in another parts of the application and will handle run the corresponding command
- The command handler will call the **Communication_openResponse** function to start a response
- The command handler may call **Communication_appendResponse** a few times (0 to 1000+ times) in order to append generate response chunks
- Once all response data are appended, the command handler will call **Communication_closeResponse** to notify the communication layer that the response is completed, so you can send your protocol footer signature if any, and flush your buffer.