Assessment Specification Document

Lightweight Authentication System for Micro:Bit IoT Device

Module Assessment

Table 1 summarizes the module assessment component and their respective ratios.

Table 1: Module Assessment

Weighting percentage between components		**Comp A: 60**	**Comp B: 40**
Comp A: E-portfolio	**Element Weight**		
A.1 Individual E-portfolio	24% (40% of the component, i.e., 24 out of 60)		
A.2 Group Work (Demo and E-portfolio)	36% (60% of the component)		
Comp B:	**Element Weight**		
Group Presentation	40 (100% of the component)		

Specification

As shown in Table 1, this is related to component **A.2** of the module assessment and contributes to **36%** of the module final mark. The aim is to implement and demonstrate a lightweight IoT device authentication system using a pair of Micro:bits. Those devices have been provided for each student by the University. **This is a group assessment, in which you are going to work in groups of 3. However, each student will need to provide an individual submission on Blackboard (BB) that illustrates the whole work and the individual contribution.** Further details on group arrangements will be discussed during the teaching sessions. Working on this assignment will help you to understand concepts of IoT security, programming for embedded systems and IoT devices using the C programming language. It will also help you develop communication and team working skills, as invaluable transferable skills in your future careers.

For this task, you need to implement an authentication system and encrypted data communication between two Micro:bits **using either** the simplified radio communication (**uBit.radio**), **or** the more advanced Bluetooth Low Energy (**BLE**). This includes the following steps:

1. Define 3 'commands', and a single PIN code that should be **shared** by the two devices (PIN code here just refer to a shared secret code, not necessary a code to be typed). You may simply define the PIN inside your code and you do not have to require typing them (hard coded PIN). You are free to choose the format and the size of the PIN. A 'command' here is defined as any action that can be performed using a Micro:bit device, or simply launched by it and executed by another device (e.g., a computer connected to it). Using the first Micro:bit (sender), the user can select the command using the Micro:bit buttons and your system should encrypt the command with AES (`aes_enc`) and transmit the it to the other Micro:bit.

2. The second Micro:bit (receiver), should be able to receive the encrypted message, decrypt it (`aes_dec`), and then **authentically** identify the "command" and execute it.

More precisely, you should implement the following simple protocol:

Sender

- Generate a random **salt.**
- Generate a data protection key, using the shared pin and salt as follows: **dpk**=**sha256**(**pin**+**salt**). pin+salt here refer to any combination function of your choice of pin and salt such as arithmetic addition, string appending (concatenation), etc.

- Use AES to encrypted the command **cipher**=**aes_enc**(**command**, **dpk**). The command here is the message that identify the command. It is up to you to define its format if needed, and any parameter may also be included if needed.
- Send the (**cipher, salt)** to the receiver Micro:bit via radio or BLE.

Receiver

- Receive **cipher and salt**.
- Generate a data protection key, using the shared pin and salt, **dpk=sha256** (**pin**+**salt**).
- Decrypt the cipher, **command=aes_dec**(**cipher**, **dpk**).
- Run the **command.**

The commands to be implemented are open to your choice, based on the functionality offered by the Micro:bit. You may use the Micro:bit LED display, or display on a computer screen using the device serial port. However, you are strongly encouraged to design more elaborate commands. It is important that any commands you implement can be demonstrated in the demo. Be creative.

NB: A single AES block should be sufficient to fit your commands. In all cases, you are not asked to use any AES inter block dependency mode (AES-ECB is enough).

Deliverables

The deliverable of this part of the assessment is the group portfolio, which includes.

- A 5-minute video that illustrates all the functionality of your system. Please use a common cross-platform video format for this (e.g., MP4).
- All source code files should be provided, clearly organised and commented.

- A README document that briefly describes your system, the commands, format of messages, etc. *(maximum 500 words).*
- ***"My-contribution"*** document*:* A document including a very short paragraph (max 200 word), describing your individual contribution. This is the only component of the submission that differs between group members. Otherwise, all members of a group should submit exactly the same documents for to this part of the assessment. **Please use standard formats (.doc, text, or .pdf)**.

Please **submit** your documents **as a single zip file on BB**, and **name it using the following format**: your-name_your-group-number.zip, where your-name and group-number refer to your surname and assessment group number respectively.

NB: Although this is a group work, please notice you need to **submit individually** on BB. Please coordinate, be synchronized, and **make sure all the group members submit exactly the same documents** for this part (except the "my-contribution" document). **In case of any problem in a group,** please let us know ASAP, don't wait until the submission deadline. Please also notice it is very important you **accurately submit the version of the code that you use in the demo**.

Marking Criteria

Table 2 describes the different marking criteria. By functionality, it is meant to clearly show that the authentication system enables the sender to send the commands and the receiver to recognize and run them, no matter how simple the commands are.

Your demo should demonstrate that upon launching the command from the sender, it will be received on the other side and the appropriate command is executed. It is obvious that the command should be encrypted and decrypted using the scheme given above. You may not be able to show all this encryption details in the demo, but your docs, source code, and presentation later on will reflect this. If you decide to go for more elaborated commands, you may associate it with a very simple display (at the receiver) upon recognizing the command, e.g., displaying the command number. This will allow you to maximize the points related to functionality independently from the progress on the elaborated commands.

You will need to submit your work via Blackboard by Thursday 6th May 2021.

Table 2: Marking Criteria

	0-4	5-8	9-12	13-18
Functionality: out 18 points.	Hash code maybe successfully calculated at the sender but the message not encrypted.	Both hash function and encryption successful, but not the communication.	All done from the sender. Communication implemented but the receiver cannot decrypt the message.	All the authentication process completed and the commands are executed.
	0-3	4-6	7-9	10-12
Commands design: out of 12 points.	Very elementary commands (e.g., simple display).	One of the commands is more advanced.	two of the commands are more advanced.	All commands are advanced.
	0-2	3-4		5-6
Internal Documentation: out of 6 points.	Little or Inconsistent documents.	Internal documentation consistent and properly describing the solution.		Outstanding Internal documentation.