Jacob Williams Individual Reflective Statement

Group Consisting of:

1. Jacob Williams – 15008632
2. Luke Murray – 19041865

Project Gitlab: https://gitlab.uwe.ac.uk/jj6-williams/iots_task3

Group Review

Working in this group was relatively straight forward, as soon as we formed the group and read through the specification for Task 3 we both immediately knew what the division of work would look like. This might be due to us already knowing each others strengths and weaknesses in programming due to us essentially peer reviewing each other not only in the other Internet of Things Security tasks but also other modules coursework.

Whilst still at University the coordination was fairly simplistic, we saw each other twice a week for lectures and this allowed us to have group meetings at these points too. I thought that as we left for the Christmas period that work cohesion may decrease due to the fact that our members live a considerable distance away from each other, but this was not the case, instead the coordination took place through messaging via WhatsApp instead.

With this group the work was formulated and adapted quickly and painlessly, a method of salt generation was conceived by my fellow member nearly as soon as we'd finished reading the specification, and I'd formulated two methods of pin entry by the time the salt generation development had started.

A major part of development, the use of AES encryption ended up being a difficult task due to varying technical and personal factors, but when I broached this there was no backlash, but instead an understanding and acknowledgment of the alternative I had formulated instead, what could have easily been a breaking point in the group was instead a relatively easy transition.

There is only one minor gripe among the group, but it is more common than not in most projects. The issue is programming standards, every single programmer has a different standard of programming to the others, and this was the case in this group also, it wasn't the cause of any internal disputes however, instead the group was either happy or simply didn't acknowledge my refactoring of most of the code.

Personal Review

There is no need to debate on the topic, the lack of AES encryption was in this implementation was entirely my fault. During development I feel perhaps I had grown ignorant of the possibility that implementing AES on the Micro-bit may be more difficult than adapting it a typical program due to the amount of implementations and documentation online surrounding AES.

Given the opportunity I would have started this coursework much sooner, there was much focus on the other pieces of coursework from other modules which had sooner deadlines. Ultimately this led to starting this coursework much later than what I now retrospectively should have.

I enjoyed experimenting with the Micro-bit, it was a new experience developing on a relatively un-established IoT device. But if I had to do this project again I would be highly convinced to a different device instead, such as the Raspberry Pi. In the specification it says that implementations using the Micro-bit radio are “easier”, I would debate this point. In short the Raspberry Pi has the ability to utilize heavy weight libraries that the Micro:bit does not, most libraries cant even compile onto the Micro:bit. Even if a library like OpenSSL could compile with Yotta onto the Micro:bit, it would be too heavy for the device to use efficiently anyways.

I do believe despite the obvious hurdle of AES, my coding has been to my usual standard of practice. Errors were quickly resolved in most cases, this being even more difficult due to having to test exclusively using the Micro:bits LED display as an output. The issue that consumed the most time other than the AES implementation was the ManagedString formatting issue. This issue is explored in more detail in the System Specification but the short of it is that two exactly similar strings, one parsed through the ManagedString format required for Micro:bit radio communications into a char array and the other just being a char array, when parsed through SHA256 produce entirely different strings. The characters still however equate equal value in a comparison, so the solution in the end was a simple character lookup table before storing the ManagedString in the char array.