Project #3: Support of New Encryption Algorithms

We have added some previously unsupported encryption algorithms to the t_cose library. This is an embedded focused library for working with COSE, a standard for encrypting CBOR messages, a binary serialisation format, somewhat like a binary version of JSON. We have selected a few AES-CCM and AES-GCM algorithms [1] and added support for these algorithms to the t_cose library [2].

Pitch

https://youtu.be/fCl_3q4QySc

Collaboration

Roles:

- George Rennie: Implementation Lead, Technical Documentation
- Avishka Ambagaha Hewage Don: Testing and Verification, Presentation
- Nicole Osazuwa: Programming, Presentation
- Edward Clewer: Project Management, Programming, Documentation
- Tsveta Todorova: Programming, Documentation, Photography

Planning

We began our collaboration with a planning meeting on Monday. Our goals for the meeting were to examine the suggested projects and choose one suitable for the set of skills and expertise that we have in the team. We decided that there were two projects which we might like to investigate: *Project #1: Conformance Test Suite*; and *Project #3: Support of new Encryption Algorithms*. By the end of the meeting, the team had collected some resources for our chosen topic, had agreed it would be best to meet in person, and had a go at setting up programming environments for the next day.

What went well? What didn't?

We made good progress throughout the day and managed to reach our objective of implementing some new encryption algorithms that had not yet been implemented in the t_cose library. We managed our time well and had the discipline to stop coding when we needed to focus on creating the report and video.

We had some trouble running the Python implementation of t_cose: PyCose, meaning that we could only test t_cose against itself.

Execution

Our initial goal was to have programming environments set up on all devices. However, we only managed to prepare a few, so all coding took place on a few computers. Edward took over the role of project manager and divided the goals for the day into subtasks with set timeframes. To ensure seamless integration, collaboration, and high productivity, we used pair-programming and pair-work practices. At key points during the day when tasks were being finished, we held small meetings in which we detailed what the next steps to take would be, and allocated tasks to members of the team.

We started the morning by setting up an environment to code in, and then we investigated the codebase which we had to extend. There was an existing pull request for the t_cose library, adding support for encrypted messages which we based our work off. We broke down and tried to understand the code that we needed to write using the whiteboard in the room that we had booked out for the hackathon.

In the afternoon, we specialised into smaller collaborative teams. George took the responsibility for most of the algorithms' implementations. Avishka was on the testing side, comparing the results George was producing to the expected values. Nicole took over preparing the presentation, and Tsveta and Edward worked on the documentation. Edward had the further task of monitoring the seamless collaboration between the presentation, documentation, and coding teams. This approach helped us stay on track and meet our deadlines.

The existing GitHub pull request [3] added basic 128-bit and 256-bit AES-GCM encryption to t_cose, so we built off of this to add 192-bit AES-GCM as well as four different AES-CCM algorithms, using the algorithm IDs from the IANA specification [1]. This consisted of adding support in the backend of t_cose to use different encryption modes in the MBedTLS PSA cryptography library. We tested these implementations by adapting the example demo code to encrypt and decrypt example CBOR messages, ensuring that the decrypted messages were the same as the plaintext. There are a few more encryption modes that could have been implemented, but we did not have time to handle them during the hackathon.

Other Challenges

We faced some challenges in determining whether what we had implemented was correct, especially when using the python libraries to check the outputs we received from our t_cose implementations. We ended our time in the hackathon still pondering whether all our implementations were correct, and how we might go about testing them more thoroughly.

Future work

In the future, we would like to be able to implement the rest of the algorithms that are listed on the IANA website, although we did not have time to do so on the day of the hackathon. We think that it would also be a good idea to test our implementations against existing implementations of the COSE algorithms so we can be more confident in the correctness of implementation.

References

[1] "COSE Algorithms," [Online]. Available: https://www.iana.org/assignments/cose/cose.xhtml#algorithms. [Accessed 26th July 2022].

[2] "laurencelundblade/t_cose: Commercial quality COSE_Sign1 implementation in C for constrained environment. Works with MbtedTLS and OpenSSL Crypto.," [Online]. Available: https://github.com/laurencelundblade/t_cose. [Accessed 26th July 2022].

[3] "Pull Request: COSE Encryption Functionality (with HPKE-based key distribution) #46," [Online]. Available: https://github.com/laurencelundblade/t_cose/pull/46. [Accessed 26th July 2022].

[4] "laurencelundblade/QCBOR: Comprehensive, powerful, commercial-quality CBOR encoder/ decoder that is still suited for small devices. (github.com)," [Online]. Available: https://github.com/laurencelundblade/QCBOR. [Accessed 26th July 2022].