Deliverable

- This exercise should be completed in groups. Please form groups of up to five members.
- Randomly selected groups will present their implementations and results during the exercise session. Be prepared for discussion and questions.
- Keep a record of challenges encountered during the task, and reflect on the lessons learned from this exercise.

Exercise 2.1 - Mbed TLS Setup

You will now set up Mbed TLS, an open source implementation of the TLS/DTLS protocols. Mbed TLS is used in embedded devices where code size and performance matter. When you run the example programs, as explained below, a basic TLS handshake will be executed. Look at the debug output to determine whether it was successful.

1. Clone the Mbed TLS repository and compile the project:
   - `git clone https://github.com/Mbed-TLS/mbedtls.git`
   - `cd mbedtls/`
   - `git checkout mbedtls-3.5.2`
   - `mkdir build && cd build`
   - `cmake ..`
   - `make`
   - `make test`
2. If Python dependencies are missing, install them:
   - `python3 -m pip install -user -r scripts/basic.requirements.txt`
3. Generate a CA certificate (still in `/mbedtls/build/`):
   - `openssl ecparam -genkey -name secp256r1 -out ca.key`
   - `openssl req -x509 -new -SHA256 -nodes -key ca.key -days 3650 -out ca.crt`
4. Create a server certificate:
   - `openssl ecparam -genkey -name secp256r1 -out server.key`
   - Note: Specify `server` as CN-Name after executing this command: `openssl req -new -SHA256 -key server.key -nodes -out server.csr`

- `openssl x509 -req -SHA256 -days 3650 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt`

5. Verify the generated certificates:

   - `openssl req -in server.csr -noout -text`
   - `openssl x509 -in ca.crt -text -noout`
   - `openssl x509 -in server.crt -text -noout`

6. Start the TLS server:

   - `./programs/ssl/ssl_server2 debug_level=5 ca_file=ca.crt crt_file=server.crt key_file=server.key`

7. Start the TLS client:

   - `./programs/ssl/ssl_client2 debug_level=1 server_name=server server_addr=127.0.0.1 ca_file=ca.crt`

8. The Mbed TLS configuration file can be found at:

   - `include/mbedtls/mbedtls_config.h`

Aufgabe 2.2 - Super Jumbo Record Limit Extension

In this exercise, you will extend the open-source Mbed TLS implementation to support a new feature called the Super Jumbo Record Limit extension. The specification for this extension is available at: `https://datatracker.ietf.org/doc/draft-ietf-tls-super-jumbo-record-limit/`

The goal of this extension is to enhance the TLS/DTLS protocols to allow the transmission of records larger than 2^{14} bytes. Like most TLS extensions, this feature is negotiated dynamically during the handshake via the ClientHello and ServerHello messages. It is only enabled if both parties indicate support for it.

Note: This specification is still a work in progress within the IETF TLS working group, and may contain errors or be subject to change.

1. **About This Exercise:**
    - A minimal goal is to enhance the feature negotiation mechanism in the Mbed TLS implementation.
    - A secondary goal is to extend the TLS application data protocol to support the larger limit size.

Code Files to Modify:

- Start by looking at the files, which implement the main logic of client and server functionality, respectively:
- `library/ssl_tls13_client.c`
- `library/ssl_tls13_server.c`
- Debug output is shown when running client and server.

Sample Programs:

- `programs/ssl/ssl_client2.c`
- `programs/ssl/ssl_server2.c`
- These example programs are started, as illustrated in the previous section.

Rebuilding After Changes:

- After modifying the source code, recompile with `make` in the build directory.
- If you modify preprocessor directives or Makefiles, you will need to re-run `cmake`.

Configuration File:

- `include/mbedtls/mbedtls_config.h` controls the build process.
- Reasonable defaults are provided; for simplicity, you should not need to make any modifications.

Final Recommendations:

- Use a virtual machine and your preferred text editor (e.g., VS Code, Eclipse) for development.
- Navigating large open-source projects takes practice. Don't give up early!

Expected Learning Outcomes:

- Gain experience working with larger open source projects, including the use of build tools and quickly understanding existing source code.
- Learn to read and comprehend (simple) technical specifications.
- Collaborate effectively in teams to solve problems.