3dTrust | Distributed Manufacturing

# Warm-up exercise

The goal of this task is to have you get a deeper understanding of some of the technologies we are using. Further, this is the place to show your practical development skills.

A client-server application should be developed for secure streaming of (encrypted) data files. The application will conduct a TLSv1.2 handshake on the client's request which leads to authentication. We will implement a client authentication (i.e. mutual authentication) and do not implement a cipher negotiation - these are given by the server (optionally the cipher negotiation can be implemented).

Finally, a simple `GET filename` operation will be sent from the client to the server after the handshake. The server sends the requested file, optionally encrypted using different keys for encryption and MAC protection. Once the server has stopped sending, the client stores the (encrypted) file and terminates without further signaling.

## High-level implementation instructions

- The client and server will be developed using a programming language of your choice.
- The attached zip file contains the certificates for the client, server and the certification authority.
- Your implementation must allow the protocol to run between client and server on different hosts, with different IP addresses. Thus, you must use TCP/IP sockets.
- Your application must handle mutual authentication by default. As we will not implement cipher negotiation, you can use AES128-GCM-SH256 for simplicity. Of course you are free to select much stronger suits.
- Your implementation must be multi-session capable, i.e. it must be possible to have two clients communicating with the server simultaneously.
- After a successful TLS handshake the client will download a file from the server. You are responsible of generating the payload file.
- Optionally the payload will be encrypted. Using SHA you can also prove that the client and server have identical copies of the payload.

## Useful links as example for Python

- **Sockets** Python https://docs.python.org/2.7/howto/sockets.html
- some of these Python libraries might be useful: `pycrypto` for simple RSA, AES and HMAC operations, `pyopenssl` for reading X509 fields.

While implementing the task you should consider the correctness and functionality and to have a well-commented code. There is no hard deadline for submitting your solution. You have a certain degree of flexibility in interpreting the requirements.