# Mandatory assignment 2

Created by: Andreas Nicolaj Tietgen (anti@itu.dk)

## Running the dice game

In order for this project to run then you have to use docker. Before using docker then we need to generate the certificates first. We do so by using the terminal.

- Go to the root of the project
- Type in the following `cd ./src/assets/certificates && sh gen_cert.sh`
- Then we build and run the docker image by typing the following: `docker compose up --build`

### Output

When running the application through docker, the output will look similar to this(of course with a different result in terms of who wins):

```
sec-mandatory-exercise-02-bob-1    | Starting server...
sec-mandatory-exercise-02-bob-1    | Server listening on [::]:5001
sec-mandatory-exercise-02-alice-1  | Server endpoint address from flag: bob.dk:5001
sec-mandatory-exercise-02-alice-1  | Setting up client...
sec-mandatory-exercise-02-bob-1    | Alice Won!
sec-mandatory-exercise-02-alice-1  | Alice Won!
sec-mandatory-exercise-02-bob-1    | Alice Won!
sec-mandatory-exercise-02-alice-1  | Alice Won!
sec-mandatory-exercise-02-bob-1    | Alice Won!
sec-mandatory-exercise-02-alice-1  | Alice Won!
sec-mandatory-exercise-02-bob-1    | Bob Won!
sec-mandatory-exercise-02-alice-1  | Bob Won!
sec-mandatory-exercise-02-bob-1    | Alice Won!
sec-mandatory-exercise-02-alice-1  | Alice Won!
sec-mandatory-exercise-02-bob-1    | Bob Won!
sec-mandatory-exercise-02-alice-1  | Bob Won!
```

## Answer to the questions

Since this is an insecure network and Alice and Bob are paranoid who do not trust anybody, then we want the following security properties:

- **Confidentiality:** We don't want anybody who is not a part of the game to read our messages
- **Authenticity:** We need to be sure that the message actually comes from Alice or Bob

- **Integrity:** We need to be sure that the dice roll is not altered and that the message is the actual message being sent

In order to solve this we use Coin Tossing with hash based commitments, and TLS.

#### Why do we use TLS?

We use TLS in order to provide *Confidentiality* and *Authenticity*. It creates confidentiality by doing asymmetric encryption to perform key exchange such that a dolev-yao attacker cannot snif the keys latter being used in a symmetric encryption scheme.

TLS also creates authenticity by signing the messages that is being sent over the network. This ensures that we know that the message is in fact from Bob or Alice and that a Man-in-the-middle attack is not happening(unless the attacker has somehow created a signed certificate impersonating Bob or Alice).

#### Why do we use Coin Tossing with commitments?

The coin tossing scheme provides us with the ability to generate random values where we know that the other part cannot cheat by only roll a 6 in a 6 sided dice.

One roll is created by having two partial rolls. One from Bob and one from Alice. This create a roll by performing the XOR operation between the two partial rolls.

How can we be sure that the partial roll that they have sent is the actual roll? We do so by using *commitments*. By using commitments we ensure that message being sent is the actual roll. We verify this by using comparing the commitment hash with the opening message.

## How do Alice and Bob play the Dice game?

I have created a client/server solution. Alice is the client and Bob is the server. Alice is the one who initiates the game every time.

For the Dice game we have two operations that Alice and Bob communicates with: - Commit - Reveal

When Alice sends a Commit message to Bob, then Bob replies with a Commit. The same goes for the reveal.

In order for them both to have a dice roll then the above happens twice. After that, they compare the rolls and decides who wins(Which you can see in the output).

The TLS is create with a self-signed CA Certificate which have signed the server certificate.