# Exercise Sheet 2

Introduction to IT-Security

Submit your solutions via Gitlab
Deadline: *Wednesday, November 26th 2025, 09:00 a.m. CET*

## Diffie-Hellman

1. Alice, Bob and Carol want to use a shared secret key for symmetric encryption.

   (a) (4 points) Modify the original Diffie-Hellman key exchange to support three parties. Use as few steps as possible for negotiating the key. You can assume that the generator $g$ and the modulus $n$ are already known for each participant.

   (b) (2 points) Compare the Diffie-Hellman key exchange with an RSA key exchange for $n$ parties in terms of public parameters, message count (complexity) and forward secrecy. For RSA assume that the public keys are known and don't require additional messages.

2. Alice and Bob use the Diffie-Hellman key exchange to negotiate a common shared key. They agree on using generator 10 with modulus 1789. An attacker might eavesdrop the negotiation and, for instance, see that Alice sends 919 and Bob sends 648 over the public transport.

   (a) (7 points) Implement a tool `crack_dh.py` to automatically derive the secrect *shared* key from the publically transferred integers, which are exchanged by Alice and Bob.

   ```
   usage:  crack_dh.py [-h] -g INT -n INT --alice INT --bob INT
   ```

   With `-g` specifying the generator, `-n` the modulus, and `--alice` and `--bob` the publically transferred integers. The derived key should be written to standard output (`stdout`) with no additional comments.

   (b) (1 point) Which problem you need to solve and why do you succeed for the given example?

# RSA

3. (3 points) Messages corresponding to the numbers 0, 1 and $N - 1$ have a special property when encrypted using the RSA algorithm. What is this property? Give a proof for each number to that this property holds true.

4. **Master:** Alice uses the RSA algorithm for encrypting a message $m$. Her public key (exponent) is 211 with modulus 67063.

   (a) (7 points) Implement a tool `crack_rsa.py` to automatically reveil the plaintext from a ciphertext $c = 19307$.

   ```
   usage:  crack_rsa.py [-h] -e INT -n INT --ciphertext INT
   ```

   With `-e` specifying the exponent, `-n` the modulus, and `--ciphertext` the ciphertext to break. All values are specified as integers to simplify the task. The result (the reveiled plaintext message) should be written to standard output (`stdout`) with no additional comments.

   (b) (1 point) Which problem you need to solve and why do you succeed for the given example?