Engineering Cryptographic Software 2026/27

November 3, 2026

Submission Deadline: January 15, 2027, 23:59 (Amsterdam time)
Submission via Brightspace only!
Submission in **groups of two**!

Assignment

The assignments are in a public git repository at https://github.com/cryptojedi/ecsw-assignments-public

Code requirements

In order to get a passing grade (i.e., ≥ 6) for the course, the code you submit must fulfill **all the following minimal requirements**:

1. All code must be implemented in Jasmin targeting the Cortex-M4 microcontroller.
2. It must not have any secret-dependent branches or access to memory at secret-dependent locations. It has to use the Jasmin information-flow typesystem to enforce this and pass tests by **jasmin-ct**.
3. The submitted software must pass the tests provided in the repository.
4. Both ChaCha20 and ECDH25519 must be substantially optimized, i.e., take at most
   - 25 000 cycles for generating 1024 bytes of ChaCha20 `crypto_stream_chacha20_ietf` output, and
   - 8 000 000 cycles for ECDH `scalarmult_base` and also 8 000 000 for `scalarmult`.

Submission requirements

Each submission must consist of just one file, specifically a `tar.bz2` archive containing your code. This should be simply the original code package with additional files and modified Makefiles as required for your optimized versions. In addition, include two files in the root directory of the `tar.bz2` archive:

1. One file called `submitters.txt` in the root directory containing names and student numbers of both group members.
2. One file called `llmusage.txt` in which you explain briefly if you used any LLMs (AI tooling) for the assignemnts and if you did, what tools and how you used them.

Oral exam

We will check that submitted code fulfills all requirements and if this is the case, follow up with an oral exam of about 10–15 minutes per group, where we will ask you to explain specific aspects of the code. **Both group members have to be able to clearly explain the code. Students who are not able to answer questions about the code will fail the course.**

Hints for optimizing the three primitives

Optimizing ChaCha20

The natural way to reimplement the core of ChaCha20 in assembly are the following steps:

- Write `quarterround` function in assembly.

- Merge 4 `quarterround` functions into a full round.
- Implement loop over 20 rounds in assembly.

For further optimization you might want to also write the loop over the message length in assembly.

Optimizing ECDH

Before optimizing the ECDH sofware, it makes sense to investigate for possible timing leaks (secret-dependent branches and/or memory addresses) and eliminate those.

There are many ways how the ECDH software can be sped up, for example:

- more efficient scalar multiplication,
- specialized base-point scalar multiplication,
- optimized group arithmetic, and
- optimized field arithmtic (using larger radix).