CAA 24-25

Mini-project

02.12.2024

- Submit **your code** and **your report** on Cyberlearn.
- The **quality** of the cryptographic implementation will be graded.
- Describe the **key sizes, parameters, IVs, ...** in your report.
- Fix a **security level** and stick to it (e.g. 256-bit security).
- The grade 4.5 is obtained by a correct (and clear) modelling of the cryptography in a report.
- The remaining 1.5 points are obtained from a good implementation.
- The programming language is free (preferably among Rust, C/C++, Java, Python/Sage). If you would like to use another language, please ask first.
- Do not hesitate to ask questions.

1 Sending Messages into the Future

The goal of this project is to design an application allowing users to send messages that can be decrypted only after a specific date in the future.

- Users should be able to login with a simple username and password.
- Users should be able to connect to their account from any device.
- Users should be able to change their password.
- Users should be able to send messages to other users. These messages are confidential and only the receiver should be able to read them.
- Senders have to set a date (day-month-year) when they send a message. Messages should be readable by the receiver only after this date (and at any point after this date). The date can be in the past.
- The receiver should know at reception and at any time when the message will be unlocked.
- The sender of the message should be authenticated and should not be able to repudiate the message.
- We consider **active adversaries**.
- Since messages can be huge files (a video game, a video, ...), receivers can download them **before** the date without of course being able to decrypt them. A final short interaction with the server is fine.
- The server is **honest-but-curious**. It will follow the protocol but will try to break the confidentiality of the messages if it can.

- The number of users in the system can be huge (millions of users) and your solution should scale. In particular, the server should not spend tons of resources encrypting files.
- A typical interface, once logged-in is:

  1. Send message
  2. Read messages
  3. Change password

  with Read message, listing all received messages, their unlock date and the sender. If you want, you can modify this interface at will.

2 Deliverable

You have to deliver the following:

- A report describing your cryptographic architecture and explaining your choices (3.5/5). In particular, provide a figure describing how the keys are managed.
- Your code (1.5/5). Note that we do **not** ask you to implement any networking. If you want, you can simulate everything locally. Only the cryptographic part will be evaluated.
- You do not have to do a GUI. Command line is fine.
- Bonus points will be given for any cool (mostly crypto-related) additional functionality. **Describe them in your report so that I know what you did!**