Appendix D

Maintenance Manual

D.1 Installing the code

Each implementation can be found on Github at `https://github.com/Litolo/informatics-project`. Clone the repo and use at your leisure.

D.2 Requirements & Dependencies

To run any of the following implementations you will need to install the following language versions:

- Python[1] version 3.12+.
- Java Development Kit[2] (JDK) version 21+.
- Maven[3] version 3.8+.
- Rust[4] version 1.86+.

Note, source programs may run on lower version but were developed using the versions listed, and so there are no guarantees that they will run than other on those listed above.

It is recommended that when running the Python programs that a separate virtual environment is created for each use-case, so that isolation can be maintained, and installed packages are not installed globally to avoid conflicts. If you are unfamiliar with how to work with virtual environments in Python, visit `https://realpython.com/python-virtual-environments-a-primer/` for more information and support.

D.2.1 Use-case specific Requirements - OPTIONAL

The first and second use-case introduce extra requirements if you are looking to run the code with minimal configuration changes. These are:

- For use case 1: End-to-End Email Encryption - Docker[5], and Docker Compose[6].
- For use case 2: Authentication and Authorisation Blog Posts - MySQL[7].

[1]`https://www.python.org/`
[2]`https://www.oracle.com/java/technologies/downloads/`
[3]`https://maven.apache.org/what-is-maven.html`
[4]`https://www.rust-lang.org/`
[5]`https://www.docker.com/`
[6]`https://docs.docker.com/compose/install/`
[7]`https://www.mysql.com/`

```
ALICE_KEY_PASSWORD=AliceKey123
BOB_KEY_PASSWORD=BobKey123
BOB_EMAIL_PASSWORD=BobEmail123
ALICE_EMAIL_PASSWORD=AliceEmail123
```

Listing 10: .env Secrets File

Listing 11: Gmail SMTP Server Configuration File

D.3 Configuration

Implementations of use-case 1 and use-case 2 allow for the changing of configuration details to allow for flexibility when connection to both the SMTP server (to send emails) and MySQL server (for managing user credential storage). The files specifying these configuration details are highlighted below. These configuration files should NOT be publicly viewable (like in a public GitHub repository) and SHOULD be kept secret in production settings since they contain sensitive information like passwords, and server connection details.

D.3.1 Secrets

Use case 1 requires encryption of emails. So, to ensure that key files are stored securely they are encrypted using a password. These passwords are stored in a .env file to simulate how passwords are traditionally handled in production settings, and so that the code is easily runnable out-of-the-box (configuring environment variables for each runtime and machine individually can be a bit of a headache). This the same for SMTP server authentication passwords. Below in listing 10 details an example of the .env containing email authentication passwords and private key file passwords for Alice and Bob.

D.3.2 SMTP Server Configuration

SMTP server configuration settings are modifiable in the masters_project/encryption/settings.json file. This file allows you to connect to any SMTP server so long as you specify the configuration values correctly, and does not have to be a local SMTP server. Listing 11 shows the SMTP server configuration file modified to connect to Gmail's SMTP server.

If you do not wish to change the configuration to point to another SMTP server, a docker-compose configuration file is provided within the project specifying the options to start a local test SMTP server, using smtp4dev[8]. This file is located at the masters_project/encryption/docker-compose.yml path, and the server can be started with the command:

```
$ docker compose up
```

Alternatively, a local server can be created via the command line, which can be configured with different arguments, with the command:

```
$ docker run --rm -it -p 5000:80 -p 2525:25 rnwood/smtp4dev
```

[8]https://github.com/rnwood/smtp4dev

D.3.3 MySQL Server Configuration

MySQL server configuration settings are modifiable in the masters_project/authentication/settings.json file. The structure is consistent with the server configuration settings shown in listing 11. You will still need to modify the password key in the json, even if you choose to install the MySQL server locally. Listing 12 shows an example configuration for the settings.json file. Keep in mind that connecting with the root user is discouraged in production settings as it extremely damaging if compromised, so if you are looking to use any of the code in this project you should create a new user for the SQL server to manage permissions to certain databases and operations within the server (if you control it).

```json
{
    "mysql": {
        "user": "root",
        "password": "super_secure_pAssw0rd123",
        "host": "127.0.0.1",
        "port": 3306
    }
}
```

Listing 12: Local MySQL Server Configuration File

D.4 Space & Memory Requirements

The only space requirement for the system is to have enough storage to hold the source code files of the project and install the required dependencies. Including the optional dependencies, the storage requirements are around 330 MB (may vary depending on system). If you wish to run the test demonstration of each implementation, you should ensure there is enough space of your system to store the resulting key files, and blog text files, which should be no more than a few MegaBytes (MBs) at most.

There are no memory requirements for the system. Systems with lower memory may find that computation takes longer.

D.5 Compiling & Running the Use-Case Implementations

For Python, navigate to the appropriate folder for the specific use case. E.g. masters_project/authentication/python/ for use case 1, masters_project/encryption/python/ for use case 2, etc. Then, download the dependencies included in the requirements.txt file. This may be done through pip with the command:

```
$ pip install -r requirements.txt
```

Lastly, for use-cases 1 and 2 you can run the main.py file which contains a demonstration and test of each of the functionalities listed for each use-case. Ensure that for use-case 1 that the SMTP server is running, and able to receive connections. This is the same for use-case 2 for your MySQL server. For use case 3, run the server.py file and client.py file in separate terminals.

For Java, Navigate to the appropriate root Maven directory for the use case. E.g. masters_project/encryption/java/email_encryption for use case 1, etc. For use-cases 1 and 2 run the following command to execute the Main class, and test the implementations:

```
$ mvn exec:java
```

For use case 3 in Java, in separate terminals run the following two commands to start the server, and client respectively:

```
$ mvn exec:java -Dexec.mainClass="com.github.Litolo.RPS.Server"
$ mvn exec:java -Dexec.mainClass="com.github.Litolo.RPS.Client"
```

D.6 Annotated File paths

Below is an incomplete directory hierarchy of the project, highlighting the important file paths. Each use case has three folders, rust/, java/, and python/. These folders contain the source files containing the methods which provide functionalities to meet each use-case requirements.

```
masters_project/
├── encryption/ - use case 1 - Email Encryption
│   ├── java/
│   ├── keys/ - private keys and digital certificate .pem files are stored here
│   ├── python/
│   ├── rust/
│   ├── .env
│   ├── docker-compose.yml - config settings to start local SMTP server using
│   │   smtp4dev
│   ├── settings.json - config settings for SMTP server
│   ├── Alice_email.txt - test email content
│   └── Bob_email.txt - test email content
├── authentication/ - use case 2 - Blog Posts Authentication
│   ├── java/
│   ├── posts/ - created blog posts are stored here
│   ├── python/
│   ├── rust/
│   └── settings.json - config settings for MySQL server
└── concurrency/ - use case 3 - RPS Game
    ├── java/
    ├── python/
    └── rust/
```

D.7 Future Improvements

D.7.1 Incorporate Additional Languages

The current implementation focuses on Python, Java, and Rust due to their contrast in paradigms and secure software engineering capabilities. Future iterations of the project can expand support to additional languages such as:

- Go, for its simplicity and built-in concurrency model.
- C#, due to its enterprise-grade tooling and support.
- C/C++, for a more thorough comparison with lower-level, memory-unsafe languages.

This expansion would facilitate a broader and more inclusive understanding of secure software engineering across programming languages.

D.7.2 In Depth Unit Testing

Future updates should incorporate more extensive automated testing frameworks. This includes:

- Coverage reports to identify untested code paths.
- Fuzz testing, particularly for the Rust implementation, to discover obscure bugs or edge-case vulnerabilities.

D.7.3 Stronger Authentication Protocols

The blog authentication system presently uses single-factor authentication. Future revisions should include support for multi-factor authentication (MFA), such as:

- TOTP-based codes.
- Email-based verification.
- Biometric or hardware token support.

D.8 Bug Reports

Known Bugs:

1. RPS game server throws an error when client sends a move request when a second player has connected, but not sent a move.
2. Rust RPS game server sometimes evaluates winner incorrectly.

Steps to Reproduce:

1. Start running the server instance in Rust. Connect to the server with two clients (i.e. run the client) and send a move.
2. Start running the server instance in Rust. Connect to the server with two clients, and send two different moves. Repeat, until result is incorrect.

Workarounds:

1. Allow only one client connection, until that player has submitted their move, then allow a second connection.
2. None, needs deeper investigation.

Severity:

1. Medium: Impacts users connecting potentially at the same time. Potential user experience issues with waiting for player to move.
2. High: Integrity issue with the game results. Telling someone they have won when they have not undermines the very rules of the game.