The University of New South Wales

COMP4337/9337 Securing Wired and Wireless Networks

Assignment specifications for T1 2021 (21T1)

Version 1.0

Updates to the assignment, including any corrections and clarifications, will be posted on the course website at Teams. Please make sure that you check the subject website regularly for updates.

1. Change Log

Draft V0.1: Released on 9th Feb, 2021

- Early draft specifications

Draft V0.2: Released on 18th Feb, 2021.

- Removed the requirement to implement BLE message passing between devices
- Added UDP broadcasting

Version 1.0: Released on 3rd March 2021

2. Due dates:

Mid-term report/recorded presentation submission: **1st April 2021**

Final report/code/demo video submission: **23rd April 2021**

3. Goal and learning objectives

For this assignment, your task is to implement a digital contact tracing protocol called "DIMY: Did I Meet You". You should implement various components of the protocol by following the specifications listed in this document, and reading the reference paper listed under the section references to understand the scope and working of the DIMY protocol. You are required to use the standard setup consisting of 1 Raspberry Pi4 module and one laptop/desktop running Linux OS (see the hardware requirements in Section 9). Your program should run on each device and exchange messages to register the contact between the Pi4 and the laptop/desktop. Optionally, you can select to implement the functionality listed under the extension section to gain additional bonus marks.

3.1 Learning Objectives

On completing this assignment, you will gain sufficient expertise in the following skills:

1. Understanding and implementing several privacy-preserving and confidentiality mechanisms such as Diffie-Hellman key exchange, Shamir Secret Sharing, Hashing and Bloom Filters.
2. Learning how UDP/TCP socket-based communications take place.
3. Integration of various technologies to achieve *Confidentiality*, *Integrity* and *Privacy*.
4. Experience in implementing a real protocol.

4. Assignment Specifications

This section gives detailed specifications of the assignment.

4.1 COVID-19 and Contact Tracing

The outbreak of the COVID-19 pandemic has changed many aspects of everyone's way of life. One of the characteristics of COVID-19 is its airborne transmission, which makes it highly contagious. Moreover, a person infected with COVID-19 can be asymptomatic, thus spreading the virus without showing any symptoms. Anyone who comes into close contact (within 2m for at least 15 min) with an infected person is at a high risk of contracting the coronavirus.

Contact tracing applications aim to establish the close contacts of an infected person so that they may be tested/isolated to break the chain of infection. The digital contact tracing app is typically composed of two main entities, the smartphones acting as clients and a back-end server. In this model, the smartphones of two individuals with tracing apps installed would exchange some random identification code (this identification code does not reveal any sensitive information about their actual identities) when they are in close proximity. The back-end is typically maintained by health organisations (or the government), and once a person is diagnosed with COVID-19, they can opt to share the local list of contacts stored on their smartphone with the back-end server to identify at-risk users. Digital contact tracing apps are not meant to replace the traditional manual contact tracing processes, rather, these have been designed to supplement it.

4.2 DIMY Digital Contact Tracing Protocol.

Download the reference paper [1] from the course website and read it to understand various components of the DIMY protocol. Note that we will be implementing a cut down version of the DIMY protocol with different parameters than those listed in the reference paper. For implementation requirements, see Section 4.3. In this section, we summarise the functionality of the DIMY protocol as it appears in the reference paper. Briefly, devices participating in DIMY periodically generate random ephemeral identifiers. These identifiers are used in the Diffie-Hellman key exchange to establish a secret key representing the encounter between two devices that come in contact with each other. After generating their ephemeral identifiers, devices employ the "*k*-out-of-*n*" secret sharing scheme to produce *n* secret shares of the ephemeral

identifiers. Devices now broadcast these secret shares, at the rate of one share per minute, through advertisement messages. A device can reconstruct the ephemeral identifiers advertised from another device, if it has stayed in this device's communication range for at least *k* minutes.

After the ephemeral identifier is re-constructed, DIMY adopts Bloom filters to store the relevant contact information. Each device maintains a Daily Bloom Filter (DBF) and inserts all the constructed encounter identifiers in the DBF created for that day. The encounter identifier is deleted as soon as it has been inserted in the Bloom filter. Devices maintain DBF on a 21 days rotation basis, identified as the incubation period for COVID-19. DBFs older than 21 days automatically get deleted.

For the back-end, DIMY utilises blockchain to satisfy the immutable and decentralised storage requirement. Once a user is diagnosed with COVID-19, they can volunteer to upload their encounter information to the blockchain. Health Authorities (HA) then generate an authorisation access token from the blockchain that is passed on to the device owner. The user's device combines 21 DBFs into one Contact Bloom Filter (CBF) and uploads this filter to the blockchain. The blockchain stores the uploaded CBF as a transaction inside a block (in-chain storage) and appends the block to the chain.

Daily, the app will query the blockchain to perform risk-analysis, checking whether the user has come in close contact with any person diagnosed positive. A device combines all of the locally stored DBFs (the maximum number is limited to 21) in a single Bloom filter called the Query Bloom Filter (QBF). The QBF is part of the query that gets uploaded to the blockchain. The blockchain matches the QBF with CBF stored as a transaction in the blockchain and returns "matched" or "not matched" as a response. If the response from the blockchain is negative, the device deletes its QBF. Conversely, if the user is found to be at-risk, the user is notified, and the QBF is stored separately for further verification by HA in the follow up manual contact tracing process.

4.3 Implementation Details

In this assignment, you will implement the DIMY protocol with a few modified parameters.

Note that in this specification, the term device refers to either Raspberry Pi4 or your laptop/desktop running an instance of the DIMY protocol implementation. You are free to choose any programming language out of Python, JAVA or C for your implementation. Your main front-end program should be named **Dimy.py** (or **Dimy.java** or **Dimy.c**). If you choose to implement the backend centralised server as listed in the extension section, your backend server code should be named **DimyServer.py** (**DimyServer.java** or **DimyServer.c**).

This assignment specification has been modified to use TCP/IP protocol stack-based message passing instead of BLE communication. It also uses different parameters as compared with the original specifications listed in reference paper [1]. This is to cut down the development, testing and demo time for the assignment. The remainder of the specification is divided into two parts, beginning with the base specification as the first part and the subsequent extension part adding new functionality to the base specification. Students can choose to only attempt the base specifications or optionally can also attempt

the extension part of this assignment for bonus marks. The marking guidelines appear at the end of the specification indicating the distribution of marks for both the base as well as the extension case.

Part 1: Base Specification

We will follow most of the original specifications from the reference paper [1] except the changes that are listed in this section. There are two major differences: 1) We will employ UDP/TCP socket-based message passing between the devices instead of using BLE communication. 2) We use different parameters values described in detail later in this section. For the base specifications, you are not required to implement the backend functionality. Rather, your front-end will interact with a running implementation of Blockchain-based back-end by using the specified API calls. For the extension part, you may opt to implement a simple centralised server acting as the back-end server instead of the Blockchain proposed in the reference paper. For details, please go through the section on assignment extension. We use the term *back-end server* to refer to default Blockchain-based implementation for base specifications and your own private centralised server in the extension part of this assignment.

In DIMY protocol, each device performs the following steps to broadcast and register a shared secret key representing an encounter with other another device in close proximity. We have listed these steps in the form of tasks you will be assessed on.

Task 1: Generate a 16-Byte Ephemeral ID (EphID) after every 1 minute.

Task 2: Prepare *n* chunks of the EphID by using *k*-out-of-*n* Shamir Secret Sharing mechanism. For this implementation, we use the values of *k* and *n* to be 3 and 6 respectively.

Task 3: Broadcast these *n* shares @ 1 unique share per 10 seconds. For this implementation, you do not need to implement the simultaneous advertisement of EphIDs proposed in the reference paper [1].

Task 4: A receiver can reconstruct the advertised EphID, after it has successfully received at least *k* shares out of the *n* shares being advertised. This means that if the devices have remained in contact for at least 30 seconds and received >= 3 shares of the same EphID, it can reconstruct the EphID. Verify the re-constructed EphID by taking hash and comparing with the hash advertised in the chunks.

Task 5: The device proceeds with applying Diffie-Hellman key exchange mechanism to arrive at the secret Encounter ID (EncID).

Task 6: A device, after successfully constructing the EncID, will encode EncID into a Bloom filter called Daily Bloom Filter (DBF), and delete the EncID.

Task 7: A DBF will store all EncIDs representing encounters faced during a 10-minute period. A new DBF is initiated after the 10-minute period and each device stores at most 6 DBFs. DBF that is older than

1 hour from the current time is deleted from the device storage. Note that in original specifications DBF stores a day worth of EncIDs, but for this demo we will use DBF to store EncIDs received in 10-minutes windows.

Task 8: Every 60 minutes, a device combines all of the available DBFs into another Bloom Filter called Query Bloom Filter (QBF).

Task 9: Each device sends this QBF to the *backend server*, to check whether it has come in close contact with someone who has been diagnosed positive with COVID-19. The device will receive the result of matching performed at the back-end server. The result is displayed to inform the user.

Task 10: A user who is diagnosed positive with COVID-19, can choose to upload their close contacts to the *backend server*. It combines all available DBF's into a single Contact Bloom Filter (CBF) and uploads the CBF to the *backend server*. Once a device uploads a CBF, it stops generating the QBFs. The device will receive a confirmation that the upload has been successful.

- Your front-end implementation should work in the debugging mode displaying messages sent and received, operations performed and state of Bloom filters in the terminal to illustrate that it is working correctly.
- Use UDP message broadcasting to implement send and receive functionality for inter device communications.
- DBF, QBF and CBF are all of size 100KB and use 3 hashes for encoding.

Part 2: Extension

Your implementation for the base specifications interacts with a pre-deployed Blockchain based-backend to send CBF/QBF and receive the results for the risk analysis performed at the back-end. For this extension part, you may opt to do your own implementation of the functionality for the *back-end server* to obtain 2 bonus marks**. However, you are not required to use a blockchain-based implementation, rather, you can use a simple centralised server to interact with the front-end.

- The *backend server* program can be deployed in your laptop or desktop machine using TCP port No 55000.
- You can provide the information regarding IP address and port No of the *backend server* to your front-end program through command line arguments. For example, Dimy.py 192.168.1.100 55000, where server is running on IP 192.168.1.100 and port No 55000 Or you can opt to hard code this information at the front-end.
- The devices establish a new TCP connection with the *back-end server* to transfer CBF/QBF to the *server*.

Task 11 (only for the extension part): The *back-end server* stores all the received CBFs and can perform matching for each QBF received from devices. It informs the device that has uploaded the QBF about the result of matching, matched or not matched. If there is no CBF available, the back-end returns "not matched".

****Condition for Bonus Marks: Please note that you can only receive bonus marks (up to 2 marks) if you do not already have full marks in the practical component of this course (i.e. your bonus marks + your total mark for the practical component of the course cannot exceed 70). If you have already achieved full marks for the practical component, your bonus marks will not be added to your quiz marks.**

5. Additional Notes

- **Groups**: You are expected to work in groups composed of either two or three students. ALL groups (whether same as for the labs or changed) must follow the below policy:
    - Post your group membership information at Teams under Notices -> Assignment group.
    - Deadline for posting your assignment group is 10th March, 2021. Any group information not completed by this deadline will incur a 2% penalty per day in the assignment mark for all group members.

- **Language and Platform**: You are free to use C, JAVA or Python to implement this assignment. Please choose a language that you are comfortable with.

- You are required to develop and test the implementation on your own laptops/provided Pi4 instead of using the CSE login servers.

- For the extension part, you are free to design your own format for messages exchanged between the devices and the back-end server. Just make sure your front-end and back-end programs can handle these messages appropriately.

- You are encouraged to use the course discussion forum to ask questions and to discuss different approaches to solve any issues faced during the implementation. However, **you should not post any code fragments on the forum.**

6. Assignment Submission

There are two deliverable stages:

1. Midterm deliverables: Midterm report including assignment diary and a recorded presentation
2. Final deliverables: Final report including assignment diary, source code and demo video

The midterm report should include the group number, members name and zIDs, assignment diary that details the overall plan with assigned task for each group member, progress on the implementation and issues faced. You are also to submit a recorded presentation, maximum of 6 slides, that explains your understanding of the DIMY protocol and what each component achieves in the protocol. You should divide the presentation equally among the group members with each member presenting their part. These mid-term deliverables carry 5 marks (2 marks for report and 3 marks for the presentation).

For the final deliverable, you will demonstrate your assignment with a video, and describe your method used for implementing the specified tasks, and issues faced along with their adopted solutions, in a detailed report (*AssignmentReport.pdf* see details in Section 7) to be submitted along with the video. You are also required to submit your source code used in the demonstration. The demonstration video carries 15 marks, while the report and code will be marked out of 10, for a total of 25 marks. The total marks for this assignment are 30.

The video should be a screen recording showing running of each step of the assignment. We recommend you ssh in the Pi4 from a terminal, so that you can capture the interaction between your laptop/desktop and the Pi4 running in different terminals on the same screen. You must include each of the following segments against Tasks 1 – 10.

Note that in the following table "show" means "A screen recording of the terminal windows (both laptop and ssh for the raspberry Pi4)".

Task	Segment	Description	Marks
Task 1	Segment 1	Show the generation of the EphID at both the devices.	0.5
Task 2	Segment 2	Show that 6 shares of the EphIDs are generated at each device.	1
Task 3	Segment 3-A	Show the sending of the shares @ 1 share per 10 seconds over UDP.	1.5
	Segment 3-B	Show the receiving of shares broadcasted by the other device.	0.5
	Segment 3-C	Show that you are keeping track of number of shares received for each EphID.	0.5
Task 4	Segment 4-A	Show the devices attempting re-construction of EphID when these have received at least 3 shares.	1
	Segment 4-B	Show the devices verifying the re-constructed EphID by taking the hash of re-constructed EphID and comparing with the hash value received in the advertisement.	1
Task 5	Segment 5-A	Show the devices computing the shared secret EncID by using Diffie-Hellman key exchange mechanism.	1
	Segment 5-B	Show that the devices have arrived at the same EncID value.	1
Task 6	Segment 6	Show that the devices are encoding EncID into the DBF and deleting the EncID.	1
Task 7	Segment 7-A	Show that the devices are encoding multiple EncIDs into the same DBF and show the state of the DBF after each addition.	1

	Segment 7-B	Show that a new DBF gets created for the devices after every 10 minutes. A device can only store maximum of 6 DBFs.	1
Task 8	Segment 8	Show that after every 60 minutes, the devices combine all the available DBFs into a single QBF.	1
Task 9	Segment 9-A	Show that the devices send the QBF to the *back-end server*. **For extension, the back-end server is your own centralised server.**	1
	Segment 9-B	Show that the devices are able to receive the result of risk analysis back from the *back-end server*. Show the result for a successful as well as an unsuccessful match. **For extension, the back-end server is your own centralised server.**	1
Task 10	Segment 10	Show that a device can combine the available DBF into a CBF and upload the CBF to the *back-end server*. **For extension, the back-end server is your own centralised server.**	1
Task 11 **(only for students attempting extension)**	Segment 11-A	Show that the device is able to establish a TCP connection with the centralised server and perform Tasks 9 and 10 successfully.	2 bonus marks**
	Segment 11-B	Show the terminal for the back-end server performing the QBF-CBF matching operation for risk analysis.	

For code submission, please ensure that you use the mandated file name. Your main program should be named **Dimy.py** (or **Dimy.java or Dimy.c**). You may of course have additional header files and/or helper files. If you are using C, then you MUST submit a Makefile/script along with your code (not necessary with Java or Python). This is because we need to know how to resolve the dependencies among all the files that you have provided.

Important notes

- Midterm and final assignment documents are to be submitted on Teams. Instructions will be provided later on.
- Late submission penalty will be applied as follows:
    - 1 day after deadline: 20% reduction
    - 2 days after deadline: 40% reduction
    - 3 or more days after deadline: NOT accepted

NOTE: The above penalty is applied to your obtained marks. For example, if you submit your final assignment deliverables 1 day late and your score on the final component of the assignment is 10/25, then your final mark will be 10 – 2.0 (20% penalty) = 8.0.

7. Report

For the final deliverable, you have to submit a small report, **AssignmentReport.pdf** (no more than 4 pages) that must contain the following:

1. Assignment name, group number and names/IDs for all group members.
2. Executive summary that provides a brief introduction to the salient features in the assignment implementation.
3. A brief discussion of how you have implemented the DIMY protocol. Provide a list of features that you have successfully implemented. In case you have not been able to get certain features of DIMY working, you should also mention that in your report.
4. Discuss any design trade-offs considered and made. List what you consider is special about your implementation. Describe possible improvements and extensions to your program and indicate how you could realise them.
5. Indicate any segments of code that you have borrowed from the Web or other books.
6. Assignment Diary: Each group is also required to attach a 1-page assignment diary to the report. This diary should maintain a weekly log of activities conducted by each group and should explicitly indicate the part played by each team member in these activities. You may use any format (Gantt chart, table, etc.) for maintaining the diary. The diary is not marked. However, if the diary is not submitted, a penalty of 2 marks will be applied. Please attach the diary at the end of the report. Do not submit it as a separate file. Unless specified otherwise, contribution from all members will be considered equal. Any difficulty in working with team members must be reported to the tutor-in-charge at the earliest.

8. Plagiarism

You are to write all of the code for this assignment implementation yourself. All source codes are subject to strict checks for plagiarism, via highly sophisticated plagiarism detection software for code as well as the submitted report. These checks may include comparison with available code from Internet sites and assignments from previous semesters. In addition, each submission will be checked against all other submissions of the current semester. Do not post this assignment on forums where you can pay programmers to write code for you. We will be monitoring such forums. Please note that we take this matter quite seriously. The LIC will decide on appropriate penalty for detected cases of plagiarism. The most likely penalty would be to reduce the assignment mark to **ZERO** and reported to the **school plagiarism register**.

Forum use.

We are aware that a lot of learning takes place in student conversations, and don't wish to discourage those. You are free to discuss (and are in fact strongly encouraged to do so) generic issues relevant to the assignment on the course forum. However, refrain from posting specific code-fragments or scripts on the forum. Students will be heavily penalized for doing so. It is important, for both those helping others and those being helped, not to provide/accept any programming language code in writing, as this is apt to be used exactly as is, and lead to plagiarism penalties for both the supplier and the copier of the codes. It is

OK to borrow bits and pieces of code (not complete modules/functions) from sample code out on the Web and in books. You MUST however acknowledge the source of any borrowed code. This means providing a reference to a book or a URL where the code appears (as comments). Also indicate in your report the portions of your code that were borrowed. Explain any modifications you have made (if any) to the borrowed code.

Hardware requirements:

The implementation for this assignment requires access to the following hardware:

1. Raspberry Pi4 kit.
2. A laptop/desktop running native Linux OS or a Linux VM.

Students (onshore only) will be issued one Raspberry Pi4 kit that they can use for labs and this assignment. You are required to return the complete kit once the course is over. It is your own responsibility to arrange an appropriate laptop/desktop for developing the application.

References:

[1] DIMY: Enabling Privacy-preserving Contact Tracing. Pdf will be made available from the course website.