# The University of New South Wales

# COMP4337/9337 Securing Fixed and Wireless Networks

# Assignment specifications for T2 2024 (24T2)

Version 1.1

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

## 1. Change Log

v1.1: released on 21 July 2024

- Changed the order of Task 9 and Task 10 in the implementation details section to match with the assessment tasks.

v1.0: Released on 17th June 2024

- Draft specifications

## 2. Due dates:

Final report/code/demo video submission: 1700 Hrs Friday 2nd August 2024

## 3. Goal and learning objectives

For this assignment, your task is to implement a hybrid 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 document listed under the section references to understand the scope and working of the DIMY protocol. You can use multiple processes/threads/virtual machines running on one laptop/desktop (with Linux OS) to setup the implementation environment.

### 3.1 Learning Objectives

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

1. Understanding and implementing several security mechanism for privacy-preserving, secret sharing, key exchange and confidentiality 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 a close contact (within 2m for at least 15 min) with an infected person is at a high risk of contracting the coronavirus.

Digital 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 the contact tracing process.

### 4.2 DIMY Digital Contact Tracing Protocol.

Download the reference paper [1] and read through it to understand various components of the DIMY protocol. 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 'node' refers to an instance of the DIMY protocol implementation (client) running on your laptop/desktop. Your main front-end program should be named **Dimy.py**. Note that you also need to implement the backend centralised server that should run on your laptop/desktop. Your backend server code should be named **DimyServer.py**.

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 marking guidelines appear at the end of the assignment specifications and are provided to indicate the distribution of the marks for each component of the assignment.

## Assignment 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 three major differences: 1) We will employ UDP/TCP socket-based message passing between the nodes instead of using BLE communication. 2) We use different parameters values described in detail later in this section. 3) You are required 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 subsection on the backend server.

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

**Task 1**: Generate a 32-Byte Ephemeral ID (EphID) after every 15 sec. Note that the reference paper proposed a 16-Byte EphID due to limitation on the size of a Bluetooth message broadcast.

**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 5 respectively.

**Task 3**: Broadcast these *n* shares @ 1 unique share per 3 seconds. For this implementation, you are not required to use Bluetooth message advertisement, rather you can use simple UDP broadcasting to advertise these shares. Also, you do not need to implement the simultaneous advertisement of EphIDs proposed in the reference paper [1].

**Task 3a:** Implement a message drop mechanism that drops a message which is ready to be transmitted with probability 0.5. This should be implemented at the sender. Hint: generate a random number between 0 and 1. If this number is less than 0.5, don't transmit that message (chunk).

**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 nodes have remained in contact for at least 9 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 node proceeds with applying Diffie-Hellman key exchange mechanism to arrive at the secret Encounter ID (EncID).

**Task 6**: A node, 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 90-second period. A new DBF is initiated after the 90-second period and each node stores at most 6 DBFs. DBF that is older than 9 min from the current time is deleted from the node's 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 90-second windows.

**Task 8**: Every 9 minutes, a node combines all the available DBFs into another Bloom Filter called Query Bloom Filter (QBF).

**Task 9**: 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 node uploads a CBF, it stops generating the QBFs. The node will receive a confirmation that the upload has been successful.

**Task 10**: Each node 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 node will receive the result of matching performed at the back-end server. The result is displayed to inform the user. You are required to use TCP for this communication between the node and the back-end server.

**Task 11**: This task performs simple security analysis of your implementation of the DIMY protocol.

A) List all the security mechanism proposed in the DIMY protocol and explain what purpose each of the mechanism serves.

B) There are two types of communications in the DIMY protocol: i) Nodes communicate with each other using UDP broadcasts. ii) Nodes communicate with the backend server using the TCP protocol. Create an attacker node by modifying your implementation of the DIMY frontend. This code is named Attacker.py. Assume that this node can receive all of the UDP broadcasts from other legitimate nodes. Think of one attack that can be launched by this attacker node. Implement this attack and show how this attack affects the DIMY nodes.

C) Now focus on the communication of nodes with the backend server. Again, think of one attack that can be launched by the attacker node assuming the communication is not encrypted and the attacker node can listen to any node communicating with the backend server. Explain how this attack affects the working of the DIMY protocol. Note that you do not need to implement this 2nd type of attack on communication with the backend server.

D) Finally, suggest measures (if possible) that can be implemented to prevent the attacks you identified in B and C above for both types of communications.

General:

- 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.
- DBF, QBF and CBF are all of size 100KB and use 3 hashes for encoding.
- You are required to run the assignment with three nodes running the DIMY frontend (plus the attacker node in Task 11) and one back-end server.

## Back-end Server

Your client implementation interacts with a backend-server to send CBF/QBF and receive the results for the risk analysis performed at the back-end. Note that, 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 is 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 client 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 nodes establish a new TCP connection with the *back-end server* to transfer CBF/QBF to the *server* and receive the results of the queries.
- The *back-end server* stores all the received CBFs and can perform matching for each QBF received from devices. It informs the node 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.

## 5. Additional Notes

- **Groups**: You are expected to work in groups composed of maximum two students. Use the same groups that you have formed for the labs.
- Use Python to implement this assignment.
- You are required to develop and test the implementation on your own laptop/desktop instead of using the CSE login servers.
- You are free to design your own format for messages exchanged between the nodes 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 on Ed 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

You need to submit a report, your source code and a demo video. Only one member of the Group is required to do the submission. Put the details of the group members in each document.

The report (*AssignmentReport.pdf* see details in Section 7) should include the group ID, members name and zIDs, and an assignment diary that details weekly tasks performed by each group members. Add a note about how to run your program detailing the steps required to compile/run your submitted code. Moreover, describe your method used for implementing the specified tasks, and issues faced along with their adopted solutions. For task 11, explain how the attacker node can launch your selected attacks on the DIMY protocol.

You will demonstrate your assignment with a video. The video should be a screen recording showing running of each step of the assignment. We recommend you run each process in a separate terminal, so that you can capture the interaction between different terminals on the same screen. You must include each of the following segments against Tasks 1 – 11. You can store the video on a file sharing site (keep video private and unlisted) and share the link in the report.

You are also required to submit your source code (e.g., submit Dimy.py, DimyServer.py and Attacker.py) used in the demonstration. The demonstration video carries 15 marks, while the report and code will be marked out of 5, for a total of 20 marks.

For code submission, please ensure that you use the mandated file name. Your main program should be named **Dimy.py**. You may of course have additional helper files.

**Note that in the following table "show" means a screen recording of the terminal windows**.

| Task | Segment | Description | Marks |
|---|---|---|---|
| Task 1 | Segment 1 | Show the generation of the EphID at the client nodes. | 0.5 |
| Task 2 | Segment 2 | Show that 5 shares of the EphIDs are generated at each node. | 0.5 |
| Task 3/3a | Segment 3-A | Show the sending of the shares @ 1 share per 3 seconds over UDP while incorporating the drop mechanism. | 0.5 |
| | Segment 3-B | Show the receiving of shares broadcast by the other nodes. | 0.5 |
| | Segment 3-C | Show that you are keeping track of number of shares received for each EphID. Discard if you receive less than k shares. | 0.5 |
| Task 4 | Segment 4-A | Show the nodes attempting re-construction of EphID when these have received at least 3 shares. | 0.5 |
| | Segment 4-B | Show the nodes verifying the re-constructed EphID by taking the hash of re-constructed EphID and comparing with the hash value received in the advertisement. | 0.5 |
| Task 5 | Segment 5-A | Show the nodes computing the shared secret EncID by using Diffie-Hellman key exchange mechanism. | 0.5 |
| | Segment 5-B | Show that a pair of nodes have arrived at the same EncID value. | 0.5 |
| Task 6 | Segment 6 | Show that the nodes are encoding EncID into the DBF and deleting the EncID. | 0.5 |
| Task 7 | Segment 7-A | Show that the nodes are encoding multiple EncIDs into the same DBF and show the state of the DBF after each addition. | 0.5 |
| | Segment 7-B | Show that a new DBF gets created for the nodes after every 90 seconds. A node can only store maximum of 6 DBFs. | 0.5 |
| Task 8 | Segment 8 | Show that after every 9 minutes, the nodes combine all the available DBFs into a single QBF. | 0.5 |

|  |  |  |  |
|---|---|---|---|
| Task 9 | Segment 9 | Show that a node can combine the available DBF into a CBF and upload the CBF to the *back-end server*. | 0.5 |
| Task 10 | Segment 10-A | Show that the nodes send the QBF to the *back-end server*. | 0.5 |
|  | Segment 10-B | Show that the nodes 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. | 0.5 |
|  | Segment 10-C | Show the terminal for the back-end server performing the QBF-CBF matching operation for risk analysis. | 1 |
| Task 11 | Segment 11-A | Explain the purpose of each of the security mechanism employed in the DIMY protocol. | 2 |
|  | Segment 11-B | Show the attacker node launching your selected attack on the inter-node communication in the implementation setup. | 2 |
|  | Segment 11-C | Explain how the attacker node can possibly launch your selected attack on the communication between nodes and the backend server. | 1 |
|  | Segment 11-D | Discuss the countermeasures that can be taken to mitigate the effects of the attacks described in Segments 11-A and 11-B. | 1 |

## Important notes

- Assignment to be submitted by give.
- Late submission penalty will be applied as follows:
    - 5% reduction in obtained marks per day after the deadline
    - 6 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 in the assignment is 15/20, then your final mark will be 15 – 0.75 (5% penalty) = 14.25.

## 7. Report

For the final deliverable, you have to submit a small report, **AssignmentReport.pdf** (no more than 4 pages+ 1 page of diary) that must contain the following:

1. Assignment name, group ID and names/IDs for all group members.
2. A note on how to run your program detailing the steps required to compile /run your submitted code.
3. Executive summary that provides a brief introduction to the salient features in the assignment implementation.
4. 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.
5. 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.

6. Indicate any segments of code that you have borrowed from the Web or other books.
7. 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.

**References**:

[1] DIMY: Enabling Privacy-preserving Digital Contact Tracing, https://www.sciencedirect.com/science/article/pii/S108480452200025X

FAQs:

Implementation:

1. Can we use available cryptographic libraries and modules? Yes, you can use any library to help you with cryptographic primitives and you don’t need to implement algorithms from scratch.
2. Do we need to use libraries for Bloom filter implementation? No, you can design Bloom Filter by setting bits with bitwise operations in a byte array to 1 or 0 to represent an element.

Report and Video:

1. Do we need to include code or terminal window screenshots in the report? No, the video will be sufficient. Submit your code separately.
2. Can we shorten timer (Task 8) for the video presentation? No, but you can fast forward the recording.
3. Is there a time limit for the video? No, but show only terminal windows with the process and no code.
4. Can we reduce amount of the information printed to the terminal? Yes, ensure your terminal windows display the necessary information in a readable and neat manner. Some ideas to consider:
    a. Segment 4-A: EpID reconstruction DONE. EphID: 033f69 " (print only the first 6 char of the EphID)
    b. Segment 7-B: A new DBF has been created from 3 encounters.
    c. Segment 10-A: Sending the QBF to the back-end server...

Submission:

1. Are both team members required to submit the assignment? No, only one student from your team can submit, but remember to include zIDs of both members in your report.