COMP4337/9337 - Did I Meet You (DIMY) Assignment

Divya Tyagi - z5514961
Haithm Ezzaddin - z5482376

Lab Session H11A
Term 2, 2025

1 Executive Summary

Our DIMY protocol implementation provides a decentralized contact tracing system using Ephemeral IDs (EphIDs), Shamir's Secret Sharing, and Bloom filters. Key features include privacy-preserving EphID distribution, secure Encounter ID (EncID) computation via Diffie-Hellman key exchange, and efficient storage with Daily Bloom Filters (DBFs). An attacker node (`Attacker.py`) demonstrates vulnerabilities by broadcasting fake shares, disrupting EphID reconstruction.

2 Compiling and Running the Code

During development, we encountered an issue with the `asyncio` library when handling large data chunks – `ValueError: separator is not found, and chunk exceed the limit`. This error occurred because `asyncio`'s default separator limit was too low for the data we were processing. To resolve this, we referred to StackOverflow [5] and increased the default separator limit by adjusting the `limit` parameter in our `asyncio` configuration to a higher value, allowing the code to handle larger data chunks without errors.
Please refer to README.md for the complete instructions.

3 Demonstration Video

The demonstration video may be accessed here.

4 Implementation of the DIMY Protocol

We implemented the DIMY protocol as specified in Tasks 1–11, with the following features:

- **Task 1**: EphID generation every t seconds using X25519 public keys (`secretSharingBroadcaster.py`).
- **Task 2**: Shamir's Secret Sharing with $k \geq 3$, $n \geq 5$
- **Task 3**: Staggered UDP broadcasting every 3 seconds.
- **Task 3a**: 50% message drop rate.
- **Task 4**: EphID reconstruction (hash verification enabled).

- **Task 5**: Diffie-Hellman key exchange for EncIDs.
- **Task 6**: Bloom filter storage with 800,000 bits, 3 hash functions.
- **Task 7**: DBF rotation/deletion.
- **Task 8**: QBF creation (`DBFmodule.py`).
- **Task 9**: CBF upload via TCP (`trigger_bloom.py`).
- **Task 10**: QBF queries with 10-bit matching threshold.
- **Task 11, Part A**: Security mechanisms documented (see Section 5).
- **Task 11, Part B**: Attacker node broadcasting fake shares (`Attacker.py`).
- **Task 11, Part C**: Attack on backend communication (see Section 7).

5 Task 11, Part A: Security Mechanisms

The DIMY protocol incorporates several security mechanisms to protect user privacy, ensure data integrity, and maintain system reliability. These mechanisms, implemented across Tasks 1–10, are listed below with their purposes.

Table 1: Security Mechanisms in the DIMY Protocol

Mechanism	**Purpose**
Ephemeral ID (EphID) Generation	Generates temporary 32-byte X25519 public keys every t seconds to prevent long-term tracking and enable secure EncID computation via Diffie-Hellman key exchange.
Shamir's Secret Sharing	Splits EphIDs into $n \geq 5$ shares, requiring $k \geq 3$ to reconstruct, ensuring resilience against eavesdropping and fault tolerance.
Staggered UDP Broadcasting	Broadcasts one share every 3 seconds, reducing the window for attackers to capture k shares and minimizing network congestion.
Message Drop Mechanism	Drops 50% of incoming UDP messages to simulate unreliable networks, forcing attackers to intercept more shares.
EphID Reconstruction and Hash Verification	Reconstructs EphIDs and verifies integrity using SHA256 hash, detecting errors or tampering (currently disabled).
Diffie-Hellman Key Exchange	Computes 32-byte EncIDs as shared secrets, ensuring encounter confidentiality and anonymity.
Bloom Filters for EncID Storage	Stores EncIDs in 800,000-bit Bloom filters, obfuscating data for privacy and enabling efficient storage.
Deletion of EncIDs	Deletes EncIDs after encoding, minimizing sensitive data exposure.
Daily Bloom Filter Rotation/Deletion	Rotates DBFs every $t \times 6$ seconds, deleting old ones to limit data retention.
Query Bloom Filter (QBF) Creation	Combines DBFs into QBFs for privacy-preserving risk queries.
Contact Bloom Filter (CBF) Upload	Uploads CBFs for positive cases via TCP, ensuring privacy and consent.
TCP Communication	Uses TCP for reliable CBF/QBF transfers with structured messaging.
Bloom Filter Matching Threshold	Requires 10 common bits for matches, balancing accuracy and privacy.
Parameter Validation	Validates t, k, n to ensure secure protocol operation.

6 Task 11, Part B: Attacker Node

The attacker node (`Attacker.py`) launches a **forged share attack** by broadcasting fake shares with valid `eph_id_hash` but invalid share data. This disrupts reconstruction by flooding fake shares as legitimate shares.

Attack Mechanism:

- Listens for UDP broadcasts on port 12345, capturing `node_id`, `index`, `eph_id_hash`, and `share`.
- Upon capturing a share, immediately floods 10 random 32-byte shares per index (1 to $n = 5$) to `255.255.255.255`, reusing the captured `EphID`.
- Legitimate nodes, with a 50% message drop rate, may accept fake shares, leading to incorrect EphID reconstruction.

Impact on DIMY Protocol:

- Incorrect EphIDs result in invalid EncIDs, corrupting DBFs, CBFs, and QBFs.

- Contact tracing fails, as CBFs and QBFs do not match (e.g., "not matched" with 2 common bits vs. "matched" with 12 before the attack).
- `Dimy.py` nodes fail to reconstruct EphIDs, producing secrets that are not 32 bytes long (e.g., 3 bytes), causing errors in EncID generation.

7 Task 11, Part C: Attack on Node-to-Backend Communication

The DIMY protocol uses unencrypted TCP communication on port 55000 for nodes to upload Contact Bloom Filters (CBFs) and query Query Bloom Filters (QBFs) to the backend server. Assuming an attacker can eavesdrop on this communication, we propose a **CBF/QBF Interception and Replay Attack** and analyze its impact on the protocol.

Attack Mechanism

– **Mechanism**: The attacker intercepts TCP packets containing CBF uploads ({`'type': 'upload', 'bloom': <BloomFilter>`}) and QBF queries ({`'type': 'query', 'bloom': <BloomFilter>`}) sent to the server. Using network sniffing tools (e.g., Wireshark), the attacker extracts the pickled dictionaries after the 4-byte length header. The attacker replays captured CBFs as new uploads, falsely indicating additional positive cases, or replays QBFs to probe for matches or overwhelm the server.

– **Execution**: The attacker captures a CBF from a node diagnosed with COVID-19 and sends it to the server as a new upload. Alternatively, the attacker replays QBFs multiple times to extract match information or disrupt server operations.

Impact on the DIMY Protocol

- **False Positives**: Replayed CBFs are stored as new positive cases, causing the server to return "matched" for unrelated QBFs, leading to incorrect exposure notifications and unnecessary quarantines.
- **Privacy Breach**: Analyzing CBFs and QBFs may reveal encounter patterns (e.g., shared EncIDs via common bits), compromising user anonymity despite Bloom filter obfuscation.
- **Denial-of-Service (DoS)**: Repeated QBF replays consume server resources, delaying legitimate queries and uploads, hindering timely contact tracing.
- **Trust Erosion**: False notifications reduce user confidence, decreasing participation in the contact tracing system.

8 Task 11, Part D: Preventive Measures for Attacks in Parts B and C

To mitigate the **Forged Share Attack** (Part B) on UDP-based share broadcasting and the **CBF/QBF Interception and Replay Attack** (Part C) on TCP-based node-to-server communication, we propose the following measures.

Preventing the Forged Share Attack (Part B)

The forged share attack involves an attacker broadcasting fake UDP shares with valid `eph_id_hash` but random share data, causing incorrect Ephemeral ID (EphID) reconstruction and failed contact tracing.

- **Digital Signatures**: Nodes sign each share with a private key (e.g., using `pynacl` signing keys). Receivers verify signatures with the sender's public key, discarding fake shares from unauthorized nodes.

- **Increase k Threshold**: Raise the minimum number of shares (k) required for EphID reconstruction, reducing the likelihood of incorporating fake shares.
- **Randomized Share Timing**: Vary the 3-second broadcast interval randomly to disrupt the attacker's ability to synchronize fake shares with legitimate ones.

Preventing the CBF/QBF Interception and Replay Attack (Part C)

The interception and replay attack exploits unencrypted TCP communication to replay CBFs and QBFs, causing false positives, privacy leaks, and potential DoS.

- **TLS Encryption**: Implement Transport Layer Security (TLS) for TCP communications to encrypt CBF and QBF messages, preventing eavesdropping and tampering by attackers.
- **Message Authentication**: Include a Message Authentication Code (MAC) or digital signature in each {`type, bloom`} message, verified by the server to ensure authenticity and prevent unauthorized replays.
- **Nonce and Timestamp**: Add a unique nonce and timestamp to each message, checked by the server to reject replayed messages that are outdated or duplicated.
- **Rate Limiting**: Enforce server-side rate limiting for QBF queries per IP address or node ID to mitigate DoS attacks from excessive replays, ensuring fair resource allocation.

9 Design Trade-offs and Special Features

Trade-offs:

- **Privacy vs. Performance**: Bloom filters (800,000 bits, 3 hash functions) balance privacy through obfuscation with storage efficiency but risk false positives.
- **Reliability vs. Overhead**: TCP for CBF/QBF ensures reliable delivery but increases latency compared to UDP.

Appendix A: Project Diary

The following table logs weekly tasks and contributions by group members from 03rd March to 22nd April.

A.1 Divya's Diary

Date	Event
04/03/2025	Reviewed assignment specification and requirements
15/03/2025	Studied DIMY paper and design principles
30/03/2025	Reviewed Haithm's initial git code commit
04/04/2025	Drafted Diffie-Hellman key exchange setup
06/04/2025	Built draft version of DIMY server
10/04/2025	Added initial Bloom Filter, DBF Module logic
19/04/2025	Implemented Attacker.py
19/04/2025	Drafted first revision of report, added references and bibliography
20/04/2025	Added security discussion to report
21/04/2025	Drafted theoretical attack for task 11C
22/04/2025	Updated README.md
22/04/2025	Reviewed submission checklist and made final report edits
22/04/2025	Drafted demo video and editted it
22/04/2025	Reviewed submission checklist and made final report edits

A.2 Haithm's Diary

Date	Event
04/03/2025	Analysed the DIMY research paper
30/03/2025	Added initial git commit with basic code setup
04/04/2025	Worked on task 1-5
06/04/2025	Created DIMY server and client
15/04/2025	Integrated Diffie-Hellman and refined on it
19/04/2025	Reviewed and refined on Bloom Filter and DBF Modules
20/04/2025	Reviewed Attacker.py implementation
21/04/2025	Did final code integrations and reviewed submission checklist
22/04/2025	Drafted demo video

References

[1] Shamir, A., "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[2] Diffie, W. and Hellman, M., "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[3] Bloom, B. H., "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[4] Murugan, N. and Mondal, A., *DIMY: A privacy-preserving decentralized framework for digital contact tracing*, Journal of Network and Computer Applications, Volume 202, 2022, 103375. Available at: `https://www.sciencedirect.com/science/article/pii/S108480452200025X`

[5] StackOverflow, "How to avoid ValueError: separator is not found, and chunk exceed the limit," `https://stackoverflow.com/questions/55457370/how-to-avoid-valueerror-separator-is-not-found-and-chunk-exceed-the-limit`, 2019, accessed on 22 April 2025.