APPENDIX A
ARTIFACT APPENDIX

This artifact appendix provides details on how to reproduce the results presented in the main body of the paper. We leverage two types of evaluation environments: (1) a theoretical setting and (2) an empirical setting.

In the theoretical setting, each proposed technique is evaluated under probabilistic assumptions derived in the methodology sections. In contrast, the empirical setting—implemented in `Python` using the `SimPy` library—involves either numerical estimations of DLM and CAM metrics or full simulations modeling end-to-end communication flows in mixnets. In the simulation environment, clients generate messages that are routed through a sequence of mixnodes according to the proposed routing strategies. Each mixnode performs mixing operations to anonymize and forward packets, with delays modeled using empirical latency measurements from Nym to capture realistic inter-node transmission characteristics.

The artifact includes approximately 6K lines of `Python` code and reproduces all results presented in the main body of the paper.[1]

A. Description & Requirements

1) How to access: The artifact repository is available via a persistent DOI: https://doi.org/10.5281/zenodo.17681892, and on GitHub: https://github.com/whenmixnetsfail/mix_adversary.

2) Hardware dependencies: Note that the artifact includes precisely the same configurations and settings presented in the paper. The only difference is that the number of iterations has been scaled down so that the artifact can run on standard systems with 16 GB RAM and 50 GB of disk space (you can also run the artifact on Google Colab).

The estimated runtimes are based on executing the artifact on a system with the following specifications:

- 64-bit CPU: Intel® Core™ i7-10850H @ 2.70 GHz (1 core used)
- Physical Memory (RAM): 16 GB

3) Software dependencies: Additionally, we tested the artifact on both `Google Colab` and `Ubuntu` 18.04. For users running the artifact on their own servers, we guarantee full functionality on `Ubuntu` 18.04 with `Python` 3.8.10 installed. Prior to running the experiments, it is essential to ensure that all dependencies listed in the `requirements.txt`[2] file are satisfied. We emphasize the importance of adhering to the specified hardware and software dependencies to ensure that the experiments complete within the expected time limits.

4) Benchmarks: Our evaluation relies on the latency and geographical datasets of the Nym network (https://nym.com), which are included in the artifact repository.

[1] At the time of artifact submission, the paper was accepted subject to minor revision; thus, the results provided here are nearly identical to those in the final version.

[2] If you are not using `Python` 3.8.10, you may encounter compatibility errors when running the artifact. In that case, please consider running `requirements_.txt`.

B. Artifact Installation & Configuration

After setting up a system with `Python` version 3.8.10, one can execute the artifact by first installing the required dependencies using the script specified in `requirements.txt`, which is provided in the repository bundled with the code. Upon installing the dependencies, running `main.py` with the arguments explained below will generate the results, which will be saved in the `Figures` directory.

C. Major Claims

The major claims of the paper are summarized as follows:

- (**C1**): The first claim concerns the trend illustrated in Figure 3. Across all settings and scenarios, we observe that as the parameters vary—specifically, increasing the size of $\mathcal{S}_h$ (Figure 3a), decreasing K (Figure 3b), and increasing α (Figure 3c)—the values of H_D and $P(D)$ on the Y-axis consistently decrease, while $P(R)$ increases. This claim is substantiated by Experiment E1, which generates Figure 3 and clearly demonstrates this trend.
- (**C2**): The second claim concerns the trend illustrated in Figure 4. Across all settings and scenarios, we observe that as the parameters vary—specifically, increasing the size of $\mathcal{S}_h$ (Figure 4a), decreasing K (Figure 4b), and increasing α (Figure 4c)—the values of DLM on the Y-axis consistently decrease. This claim is supported by Experiment E2, which produces Figure 4 and demonstrates this trend.
- (**C3**): The third claim concerns the trend illustrated in Figure 5. Across all settings and scenarios, we observe that as the parameters vary—specifically, increasing the size of $\mathcal{S}_h$ (Figure 5a), decreasing K (Figure 5b), and increasing α (Figure 5c)—the values of CAM on the Y-axis consistently decrease. This claim is supported by Experiment E3, which produces Figure 5 and demonstrates this trend.

D. Evaluation

This section details the set of experiments conducted to support the main claims presented in the artifact appendix. Executing these experiments produces results stored in the `Figures` directory. Additionally, we explain how to run scripts to regenerate specific results corresponding to figures or tables presented in the main body of the paper.

1) Experiment (E_1) [Figure 3] ($\leq$ 5 min): This experiment supports claim **C1**.

[Configuration Parameters] The configuration parameters match those used in Fig. 3, specifically: $L = 3$, $W = 300$, $\beta = 0.1$, and $m = 500$ (corresponding to 1 MB of data).

[Preparation and Execution] Run the following command with the `Input` argument set to `100`:

```
python3 main.py
```

[Results] The results will be saved in the `Figures` folder as: Fig. 3a–3c.

TABLE I
MAPPING OF INPUT ARGUMENTS TO SPECIFIC FIGURES AND TABLES.

Experiment	Input	Experiment	Input
Fig. 1	1	Fig. 3	3
Fig. 4	4	Fig. 5	5
Fig. 6	6	Fig. 7	7
Fig. 8a & 9a	891	Fig. 8b & 9b	892
Fig. 8c & 9c	893	Fig. 10a	101
Fig. 10b	102	Fig. 10c	103
Fig. 11a	111	Fig. 11b	112
Fig. 11c	113	Tab. 1	1000

2) Experiment (E_2) [Figure 4] ($\leq$ 40 min): This experiment supports claim **C2**.

[Configuration Parameters] The configuration parameters match those of Fig. 4, specifically: $L = 3$, $W = 300$, and $m = 500$ (1 MB of data).

[Preparation and Execution] Run the following command with the `Input` argument set to `200`:

```
python3 main.py
```

[Results] The results will be saved in the `Figures` folder as: Fig. 4a–4c.

3) Experiment (E_3) [Figure 5] ($\leq$ 15 min): This experiment supports claim **C3**.

[Configuration Parameters] The configuration parameters match those of Fig. 5, specifically: $L = 3$, $W = 300$, and $\mu = 30{,}000$ packets/s.

[Preparation and Execution] Run the following command with the `Input` argument set to `300`:

```
python3 main.py
```

[Results] The results will be saved in the `Figures` folder as: Fig. 5a–5c.

4) Experiment (E^*) [All Others] ($\leq$ 1 h): This experiment allows the user to generate any figure or table shown in the main body of the paper, regardless of whether it is linked to a claim.

[Preparation and Execution] Refer to Table I for the corresponding `Input` argument value and run:

```
python3 main.py
```

[Results] Figures will be saved in the `Figures` folder. Table data should be printed directly in the terminal.

E. Parameter Settings and Execution Time

The network parameter settings are consistent across all experiments and reflect the configurations described in the evaluation section of the paper. These parameters are initialized in `Main_Functions.py` and do not require manual changes to reproduce the core results.

However, for users interested in exploring additional experiments or tuning performance, a subset of parameters can be safely adjusted. Note that mixnets are highly interdependent systems, and improper configuration may lead to invalid results or unexpected runtime behavior. In particular, some values are derived from prior studies to ensure consistency and fairness across experimental settings.

Safely Adjustable Parameters in Main_Functions.py:

- `Iterations` $\rightarrow$ Controls the number of simulation iterations. This can be increased (e.g., up to 30) to improve statistical accuracy; however, note that the computational cost grows approximately linearly.
- `num_targets` $\rightarrow$ Specifies the number of target messages in the simulation. Acceptable values range from 20 to 200.
- `run` $\rightarrow$ Sets the duration of each simulation time slot. Can be tuned between 0.3 and 1.0 (real values).
- `delay1` $\rightarrow$ Represents the average delay introduced at each mixnode. This can be modified within the interval $[0.01, 0.08]$ to simulate different latency conditions.

Other parameters in `Main_Functions.py` should **not** be modified unless the user has deep familiarity with the internal logic and dependencies of mixnet protocols. Inappropriate changes may compromise result validity or break simulation behavior. Users seeking to explore such changes are encouraged to contact the authors for further guidance.