Artifact Evaluation (AE) Appendix

Angeliki Aktypi
University of Oxford
angeliki.aktypi@cs.ox.ac.uk

Kasper Rasmussen
University of Oxford
kasper.rasmussen@cs.ox.ac.uk

In this document we provide all the materials necessary for Artifact Evaluation (AE) of the paper "Iris: Dynamic Privacy Preserving Search in Authenticated Chord Peer-to-Peer Networks", accepted to Network and Distributed System Security (NDSS) Symposium 2025.

A. Artifact Structure

In the repository we provide the code and the data we use in the paper. As a fist step, users can run the code to execute the IRIS algorithm, and to collect execution data. Subsequently, the users can run the provided plot scripts to recreate the figures used in the paper. The network generation is randomized, so to fully reproduce the results we present in the paper, we share the data we generated and used for our plotting.

In Figure 1 we provide the schema of the coding files in the repository. The majority of the work is done in these four files:

- `Id_Space_Linear.m` creates a circular id space with a number of participating nodes placed uniformly at random. A fraction of them, *i.e.*, $0, \frac{1}{2}, \frac{1}{3}, \frac{1}{8}, \ldots$, are chosen as colluding adversaries.
- `Iris.m` implements the IRIS algorithm as described in Section VI of the paper. This function performs an iterative search for a target object O_p, initiated by a requester N_r keeping the privacy parameters α and δ constant during the search.
- `Privacy.m` calculates the loss of privacy at every node that is queried in a search. This loss is proportional to the ratio of the *posterior* and the *prior* knowledge about the target object, as specified in Section V of our paper.
- `Chord_Lookup.m` implements the Chord `lookup` protocol, as described in Section II of our paper. Given a specified target object, the code returns the address of the next node to be queried.

B. Set Up

1) Access: The complete artifacts and most recent version of the code can be accessed at https://github.com/angakt/iris. A snapshot of the release (based on which the artifacts evaluation has been performed) has also been uploaded at Zenodo and can be accessed at https://doi.org/10.5281/zenodo.14251874.

2) Hardware Dependencies: Any computer that can run Matlab or GNU Octave. For our implementation we use a computer with an Intel Core i7-4820K processor and 16GB installed RAM memory.

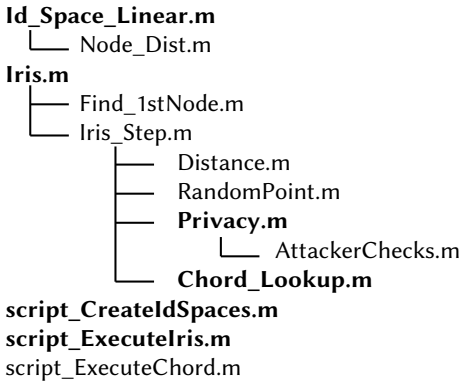

Fig. 1. *The schema of the Iris code base.*

3) Software Dependencies: The code can be executed on any operation system that supports Matlab or Octave, e.g., Windows, MacOS, or Ubuntu. We test our code on Windows 10, Education edition, version 22H2, running the Matlab version R2023a with an academic licence. However, as the code does not use any special libraries, GNU Octave, an open-source alternative to Matlab, is also sufficient to run the scripts. For convenience we link to a GNU Octave docker image available at https://github.com/gnu-octave/docker

C. Claims

The provided code is used to evaluate the proposed algorithm, IRIS. More specifically the experiments performed help us attest:

- C1 IRIS's correctness: the algorithm converges to the target address.
- C2 IRIS's privacy guarantees: the algorithm is (α, δ)-*private*, *i.e.*, the privacy ratio against any queried node and colluding adversary does not drop below α, and the algorithm does not provide an advantage to the attacker's guess.

D. Execution

In the paper we perform five different experiments. The first two experiments presented in Figure 6 of the paper, focus on the evaluation of the performance cost that is introduced by IRIS, *i.e.*, the extra hops that a query needs to perform. The third experiment presented in Figure 7, attests that IRIS is an (α, δ)-*private* algorithm, whereas the last two experiments

illustrated in Figure 8 and in Figure 9 examines the attacker's advantage when executing IRIS.

1) Network Generation: The preliminary step for all the experiments is the creation of a number of different networks. These networks are generated with the `script_CreateIdSpaces.m` script.

The script creates 500 different networks of 1000 nodes each with 2^{23} number of addresses. These three parameters are hardcoded in the script but can be changed based on user's needs by changing lines 7, 11 and 12, respectively. After executing this command 500 mat files, each one containing one initialised network, are created and saved under the folder `./experiments/networks/1000_nodes/`. After completing this preliminary step we can proceed with the execution of the experiments.

2) Experiments: For all the experiments we make use of the `script_ExecuteIris.m` script. This script allows us to run IRIS a specified number of times, specifying the α and δ parameters, and the fraction of colluding adversaries, across every set of experiments.

For every execution, the script uses a different network of 1000 nodes by loading a new network file from the `./experiments/networks/1000_nodes/` folder. For every execution the script selects the address of the target and the requester uniformly at random, checking that the requester is not among the colluding nodes.

The experiment parameters are embedded in the script thus for every new experiment a few lines needs to be changed to generate the required data for a particular experiment.

To reproduce our experiments and results, the following lines need to be changed:

- line 12 specifies the number of performed experiments of every set.
- lines 28, 29 and 34 specify the number of participating nodes in the network, the number of the id space addresses and the folder under which the mat file with the initialised network is saved, respectively.
- line 42 defines the α parameter.
- line 47 defines the δ parameter.
- line 53 defines the fraction of colluding nodes. (Recall that the colluding nodes are specified in the attackers variable in the mat file of the address space.)
- line 131 defines the name of the file to be saved with the experiments data.

To avoid an error-prone reproducibility of the performed experiments, we provide the parametrized `script_ExecuteIris.m` scripts that are to be used for every experiment. In the next section, we report their use together with further details regarding the execution of the experiments.

3) Graphs: Finally, to reproduce the graphs used in the paper, the data from the experiments can be plotted with the scripts found in the `./experiments/results/` folder. These are standard plots in either Matlab or R and we do not consider these part of our contribution, but we include them for completeness.

E. Evaluation

[Preparation] All the experiments in the paper were executed using networks with 1000 nodes placed on an address space with 2^{23} addresses.

1) Experiment (E1): [Figure_6a] [1 human-minute + 1 compute-minute]: In this experiment we examine IRIS's performance for different values of the α parameter, to support our first claim.

[Execution]

1) Run the script `results\fig_DistancesPerAlpha\script_ExecuteIris.m`, this will execute IRIS setting α equal to 0.25, 0.35, 0.5 and 0.75, producing 4 csv and 4 mat files. For each value of the α parameter we execute 100 experiments. Apart from α the rest parameters remain stable, $\delta = 1/16 * address\_space$ and $f = 0$.

2) Run the script `results\fig_DistancesPerAlpha\script_ExecuteChord.m`, this will produce 1 csv file (data_a1.csv) that contains the results when executing Chord using for comparison reasons the targets and the requesters of one of the other mat files.

[Graph]

Run the `results\fig_DistancesPerAlpha\script_PlotDistancesPerAlpha.m` to plot the average distances to the target for each α value. The plot needs to be executed in the same folder with the data produced above.

2) Experiment (E2): [Figure_6b] [1 human-minute + 1 compute-minute]:

This experiment is also related to our first claim examining IRIS convergence for different values of the δ parameter.

[Execution]

1) Run the script `results\fig_DistancesPerDelta\script_ExecuteIris.m`, this will execute IRIS setting δ equal to 1/4, 1/8, 1/16 and 1/32 of the address space, producing 4 csv and 4 mat files. For each value of the δ parameter we execute 100 experiments. Apart from δ the rest parameters remain stable, $\alpha = 0.35$ and $f = 0$.

2) Run the script `results\fig_DistancesPerDelta\script_ExecuteChord.m`, this will produce 1 csv file (data_a1.csv) that contains the results when executing Chord using for comparison reasons the targets and the requesters of one of the other mat files.

[Graph]

Run the `results\fig_DistancesPerDelta\script_PlotDistancesPerDelta.m` to plot the average distances to the target for each δ value. The plot needs to be executed in the same folder with the data produced above.

[Preparation] For the experiments in Figures 7, 8 and 9 we alter IRIS so as to focus solely on the nodes that have a correct estimation regarding the target. Thus, we need to comment lines 27-31 and uncomment lines 35-40 in Iris.m file. The next three experiments support our second major claim.

3) Experiment (E3): [Figure_7] [1 human-minute + 5 compute-minutes]:

[Execution]

1) Run the script `results\fig_PrivacyPerAttackers\script_ExecuteIris.m`, this will execute IRIS setting the f value equal to 0, 1/2, 1/3, 1/6 and 1/8, producing 5 mat files. For each f value we execute 500 experiments. Apart from f the rest parameters remain stable, $\alpha = 0.25$ and $\delta = 1/4 * address\_space$.

2) Run the script `results\fig_PrivacyPerAttackers\script_FindMinPrivacyRatio.m`, the script loads the privacy data of the 500 experiments of each f value and finds the min privacy ratio of every experiment saving the data to 5 csv files.

[Graph]

Run the script `results\fig_PrivacyPerAttackers\script_PlotMinPrivacyRatioPerAttackers` to plot the minimum acquired privacy ratios as histograms.

4) Experiment (E4): [Figure_8] [1 human-minute + 1 compute-minute]:

[Execution] 1) Run the script `results\fig_Probabilities\fig_DistancesNormalizedByDelta\script_ExecuteIris.m`, this will execute IRIS 500 times with $\alpha = 0.75$, $\delta = 1/128 * address\_space$ and $f = 0$, producing 1 mat file.

[Graph] 2) Run the `results\fig_Probabilities\fig_DistancesNormalizedByDelta\script_PlotDistancesNormalizedByDelta.m` to plot the histogram of the results.

5) Experiment (E5): [Figure_9] [1 human-minute + 1 compute-minute]:

[Execution]

1) The script `results\fig_Probabilities\fig_DistancesNormalizedByDelta\script_PlotDistancesNormalizedByDelta.m` from the previous step, produces 2 csv files with the distances the queried node has to the target and to the randomly picked address. If we do not want the plotting but only to extract the two csv files that are necessary for the fifth experiment, we have to comment lines 28-45.

[Graphs] 2) To plot the probabilities we execute the scripts in the `results\fig_Probabilities\fig_ConditionalProbabilities` folder. Each script corresponds to one Subfigure. We can alter the examined x value by changing line 9.