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", submitted to Network and Distributed System Security Symposium (NDSS) 2025. The paper has received a *major revision* decision. The reviewers ask the authors to discuss further the selection of IRIS parameters, limitations of the proposed work and its applicability. These changes do not alter the submitted artifacts.

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 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 repository with the complete artifacts can be accessed at https://github.com/angakt/iris.

2) Hardware Dependencies: Any computer that can run Matlab.

```
Id_Space_Linear.m
 └── Node_Dist.m
Iris.m
 ├── Find_1stNode.m
 └── Iris_Step.m
         ├── Distance.m
         ├── RandomPoint.m
         ├── Privacy.m
         │     └── AttackerChecks.m
         └── Chord_Lookup.m
script_CreateIdSpaces.m
script_ExecuteIris.m
script_ExecuteChord.m
```

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

3) Software Dependencies: The provided code is written in Matlab. For our implementation we used an academic Matlab 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.

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.