Artifact Evaluation for “Network Scheduling for Secure Cyber-Physical Systems”

Vuk Lesi, Ilija Jovanov and Miroslav Pajic

Duke University

In this document, we provide a summary of functions required to obtain the results presented in this paper, and information on how they can be modified. For testing convenience, as one set of figures requires installation of additional package, we have split this overview into two sections.

In Section 1, **Reachability Analysis Framework**, we provide the description of codes needed to obtain **Figure 5, Figure 8, Figure 10, and Figure 11**. A quick reference table is given at the beginning of the section (Table 1), after which the full script description and script calls are provided. These figures are related to analysis of values that estimation error can take in presence of an attack (reachable regions), and analysis of Quality of Control in presence of different integrity enforcement policies ( curves). In order to run these scripts no additional libraries are required, and they can be executed using a standard Matlab distribution (Note: Lines number 23, 40, 57, 76, 92, 108 in PlotAllFigures.m have been commented out, since yticks() was added in Matlab version R2016b. If running on recent version, these lines can be uncommented to obtain the same y axis markers as in paper).

In Section 2, **Scheduling Framework**, we provide the description of codes needed to obtain **Figure 7** and **Figure 9**. These figures are related to network scheduling of messages with periodic and intermittent data authentication, as discussed in the paper. Scheduling framework codes were run using Gurobi 7.0.2 with Matlab 2015b.

***In order to successfully run all codes, Matlab needs to be equipped with the Statistics and Machine Learning Toolbox and Gurobi MILP optimizer. More information on Gurobi can be found at*** <http://www.gurobi.com/products/modeling-languages/matlab>***.***

***Once Gurobi is installed, instructions on how to setup Gurobi for Matlab can be found at*** <https://www.gurobi.com/documentation/7.0/quickstart_mac/matlab_setting_up_gurobi_f.html>***.***

|  |  |  |
| --- | --- | --- |
| Figure # | Matlab script that generates the figure | Short Description |
| 5 | CruiseControlPlot3DRegions.m | Reachable regions for a system with 3 states |
| 7 | L\_allowableRegion.m | Allowable region of inter-enforcement distances for a given utilization bound |
| 8a | CaseStudy4.m | Cruise control curve |
| 8b | CaseStudy3.m | Differential braking curve |
| 8c | CaseStudy2.m | Steering control for lane tracking curve |
| 9 | NPR\_synthesis\_eval.m | Statistics of execution times for Gurobi solver |
| 10 | PlotAllFigures.m | Cruise control – simulated execution |
| 11 | Steering control – simulated execution |

**Table 1.** List of all figures related to reachable regions, Quality of Control, and scheduling and Matlab scripts that need to be executed to obtain them

1. Reachability Analysis Framework

**Figure 5: Reachable State Estimation Error Regions:** To obtain evolution of the state estimation error regions for automotive cruise control in presence of attacks on distance sensor (Figure 5), one needs to execute **CruiseControlPlot3DRegions.m**

This function relies on following data and subroutines:

* CruiseControlGA.mat – This file provides state space matrices (, describing the dynamics, and , describing the noise) needed for the analysis.
* FindErrorRegionsOurs.m – This file finds the error regions using method described in “*Relaxing Integrity Requirements for Attack-Resilient Cyber-Physical Systems*” available on <https://arxiv.org/abs/1707.02950>
  + findK.m – Finds Kalman gain for given state space system.
  + IsPerfectlyAttackable.m – Finds whether the attacker can cause unbounded state estimation error.
  + FindAlpha.m – Finds robustness condition corresponding to detector threshold.
* Ellipse\_plot.m – Plots the obtained 3D ellipsoids.

This script can be easily modified to provide the results for any 3-state system with 3 sensors, simply by replacing system matrices () with desired problem matrices. For any further modifications (e.g., more or less states, different sensor configuration, more steps, different enforcement policy), there is a thorough explanation for arguments forwarded to FindErrorRegionsOurs.m inside of the script.

**Figure 8: l-e Curves:** To obtain l-e curves (Inter-enforcement distance versus Quality of Control) presented in Figure 8 (a, b, and c), one needs to execute **CaseStudy4.m**, **CaseStudy3.m**, and **CaseStudy2.m**, respectively. These scripts rely on following scripts:

* PlotErrorCurve.m – Takes information about the system architecture and attack (matrices related to system and estimator dynamics, compromised sensors, and risk taken by the attacker), and provides the l-e curve information.
  + FormGlobalPeriodicPolicy.m – Forms matrices that describe a global integrity enforcement policy and returns them in the form expected by PlotErrorCurve.m.
  + FindErrorRegionsCurve.m – Similar to FindErrorRegionsOur.m, but instead of full regions just returns the extreme point of the region.
    - findK.m, IsPerfectlyAttackable.m, FindAlpha.m

These scripts can be easily modified to present l-e curves for any state-space system by changing the system matrices, attack matrices, and/or detector conditions. Detailed information about parameters that are being sent to the key script FindErrorRegionsCurve.m can be found inside of the script.

**Figures 10-11: Simulation results:** To obtain simulation of the system execution that shows effects of different integrity enforcement policies (presented in Figure 10 and Figure 11) we wrote script **PlotAllFigures.m**. This script relies on following data and subroutines:

* CruiseControlLis13.mat – Contains the data obtained via SimulateCase4.m for .
* CruiseControlLis5.mat – Contains the data obtained via SimulateCase4.m for .
* CruiseControlChangingLs.mat – Contains the data obtained via SimulateCase4ChangingLs.m.
* LanePositionLis6.mat – Contains the data obtained via SimulateCase2.m for .
* LanePositionLis4.mat – Contains the data obtained via SimulateCase2.m for .
* LanePositionChangingLs.mat – Contains the data obtained via SimulateCase2ChangingLs.m.
* SimulateCase4.m – performs simulation of Cruise control case study
  + SimulateClosedLoopSystemV2GAasEA – simulates closed loop system for architecture given in paper. Accepts system matrices, and plans and performs an attack that attempts to cause high error given the limitations halfway through the simulation.
    - findK.m, IsPerfectlyAttackable.m
    - PlanAttackGAasEA.m – Attacks are being computed using algorithm described in <https://cpsl.pratt.duke.edu/sites/cpsl.pratt.duke.edu/files/u42/RelaxingIntegrityRequirements_CDC17.pdf>. Given that for multistep planning computation time can get high, we implement stepwise planning in this script, using properties of multistep attacker observed in our previous work.
      * PlanAttack.m – Plans single or multistep attack
      * ComputeZ.m – Computes residue change caused by attack
* SimulateCase4ChangingLs.m – Performs simulation of Cruise control case study with varying inter-enforcement distance l.
  + CCListOfLs.m – Contains the list of inter-enforcement distances used throughout the simulation.
  + SimulateClosedLoopSystemChangingLs.m – Performs functionality of SimulateClosedLoopSystemV2GAasEA, adapted to changing l values.
    - PlanAttackGAasEAchangingL.m – Using data obtained for different values of l via script PlanAttackGAasEA.m, forms the attack based on the list of ls provided as an input (essentially by concatenating different attacks). To save time, attacks are precomputed and stored in following files:
      * CCaLis1.mat, CCaLis2.mat, CCaLis3.mat, CCaLis4.mat, CCaLis5.mat, CCaLis6.mat, LPaLis1.mat, LPaLis2.mat, LPaLis3.mat, LPaLis4.mat.
* SimulateCase2.m – Performs simulation of Steering control case study
  + SimulateClosedLoopSystemV2GAasEA.m
    - findK.m, IsPerfectlyAttackable.m, PlanAttackGAasEA.m
      * PlanAttack.m, ComputeZ.m
* SimulateCase2ChangingLs.m – Performs simulation of Steering control case study with varying inter-enforcement distance l.
  + LPListOfLs.m – Contains the list of inter-enforcement distances used throughout the simulation.
  + SimulateClosedLoopSystemChangingLs.m
    - PlanAttackGAasEAchangingL.m
      * CCaLis1.mat, CCaLis2.mat, CCaLis3.mat, CCaLis4.mat, CCaLis5.mat, CCaLis6.mat, LPaLis1.mat, LPaLis2.mat, LPaLis3.mat, LPaLis4.mat.

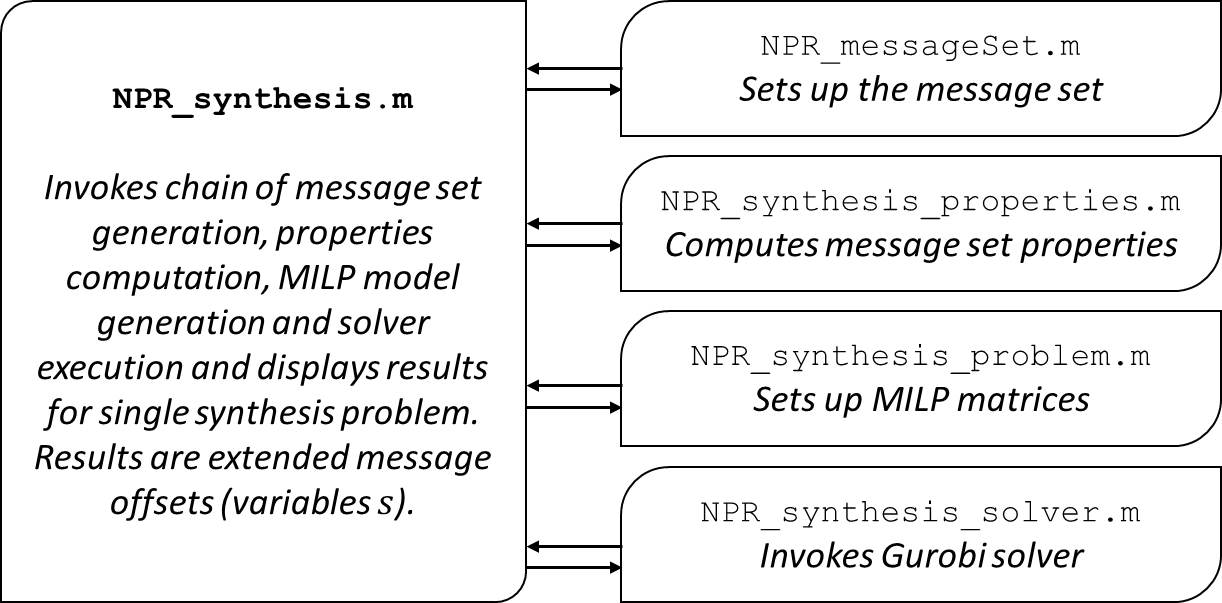
These codes can be easily modified to execute simulation for any system, given its state-space representation, and attacker model. The most important scripts for simulation of the execution are SimulateClosedLoopSystemV2GAasEA.m, SimulateClosedLoopSystemChangingLs.m, and they contain full directions on how they can be used to run modified systems.

1. Scheduling framework

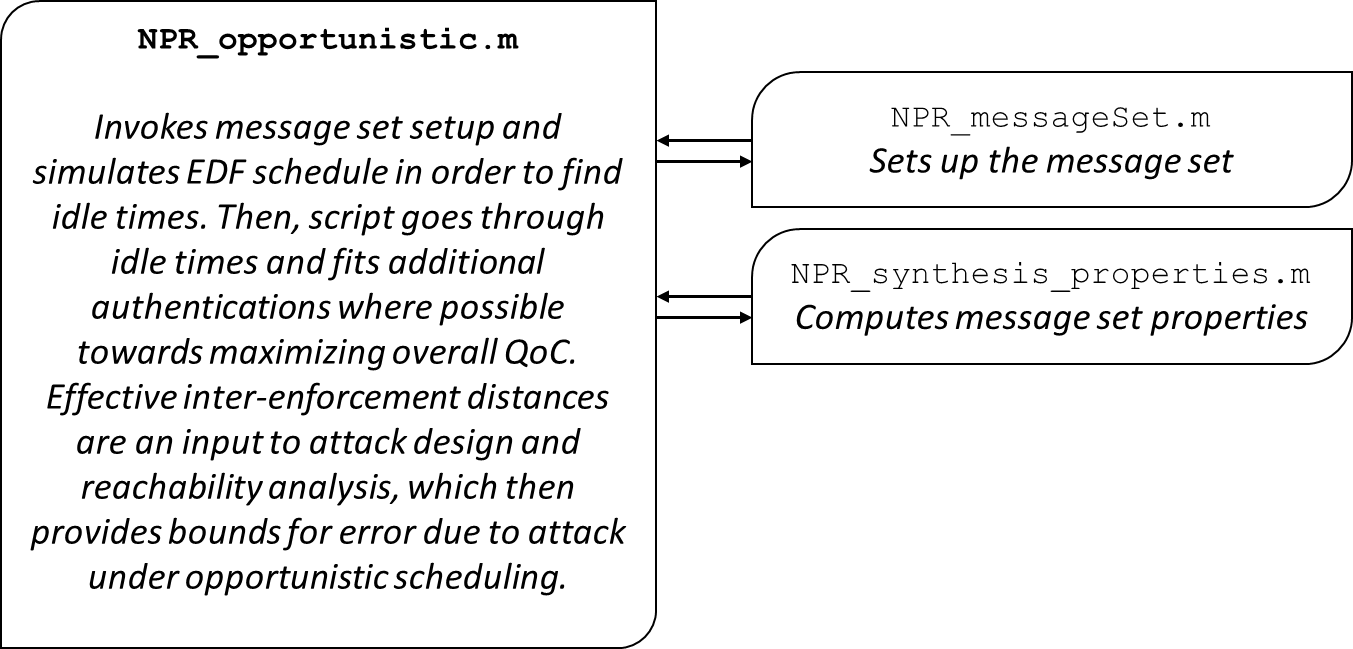
**Figure 7: Allowable region of inter-enforcement distances:** To obtain allowable points in the space of inter-enforcement distances, run script L\_allowableRegion.m. This script calls NPR\_messageSet.m that sets up the message set (see the paper Appendix). The results of this script (lower bounds on inter-enforcement distances) are then used in NPR\_optimization.m for obtaining a QoC-optima schedule.

**Figure 9: MILP solver execution times statistics:** To obtain average execution time and corresponding confidence interval, this script invokes the predefined-QoC scheduling nIter number of iterations. *Reviewer is advised to reduce this variable (line 13 in script NPR\_synthesis\_eval.m) due to slow matrix generation process. Resulting execution times will not significantly difer from the ones shown in the paper.*

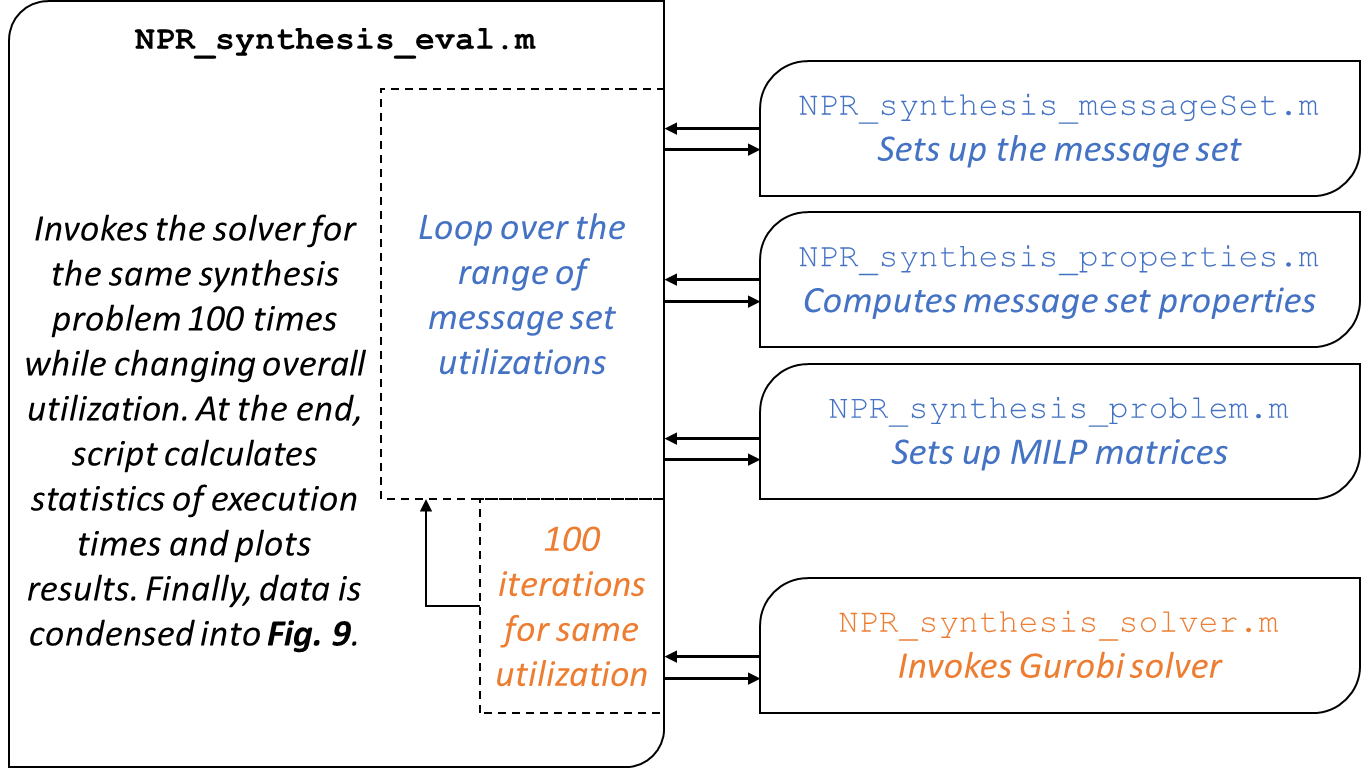
The codes can be easily modified to perform schedulability analysis for any message set, specified via the message model discussed in the paper. All scripts are fully annotated.



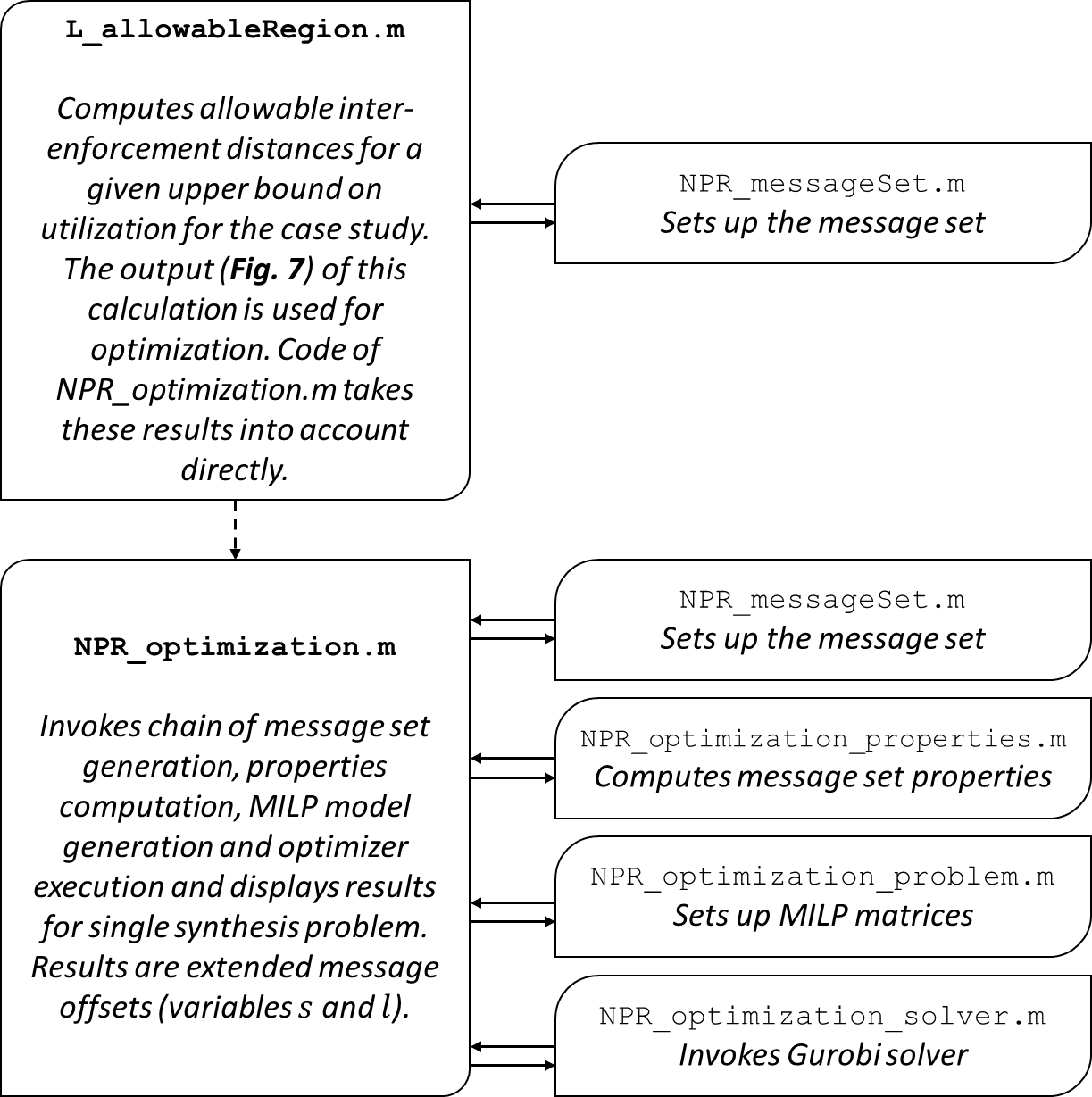
**Chart 1.** Scheduling with predefined QoC requirements; All related results are obtained by running NPR\_synthesis.m



**Chart 2.** Opportunistic scheduling; Effective inter-enforcement distances are the output of the script, which is then used for system simulation



**Chart 3.** Execution time evaluation; Script executes the synthesis problem times; Due to slow matrix generation, the reviewer is advised to reduce the number of iterations (variable nIter on line 13 of NPR\_synthesis\_eval.m)



**Chart 4.** Allowable inter-enforcement region calculation and QoC-optimal scheduling