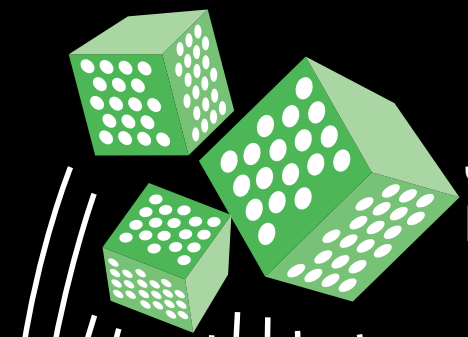
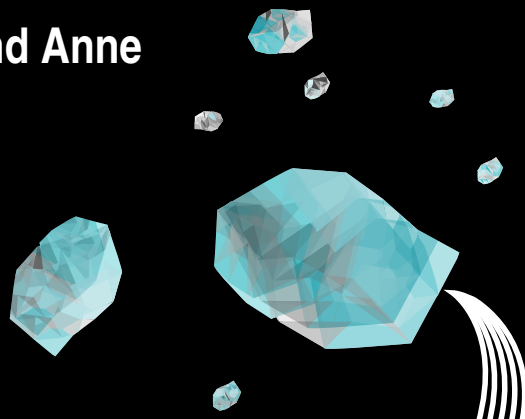
UNIVERSITY OF TWENTE.

Evaluating a process-aware IDS for smart grids on distributed hardware



Joint work with Kai Oliver Großhanten and Anne Remke



This research is conducted as part of the ISoLATE project (CS.016) funded by NWO.

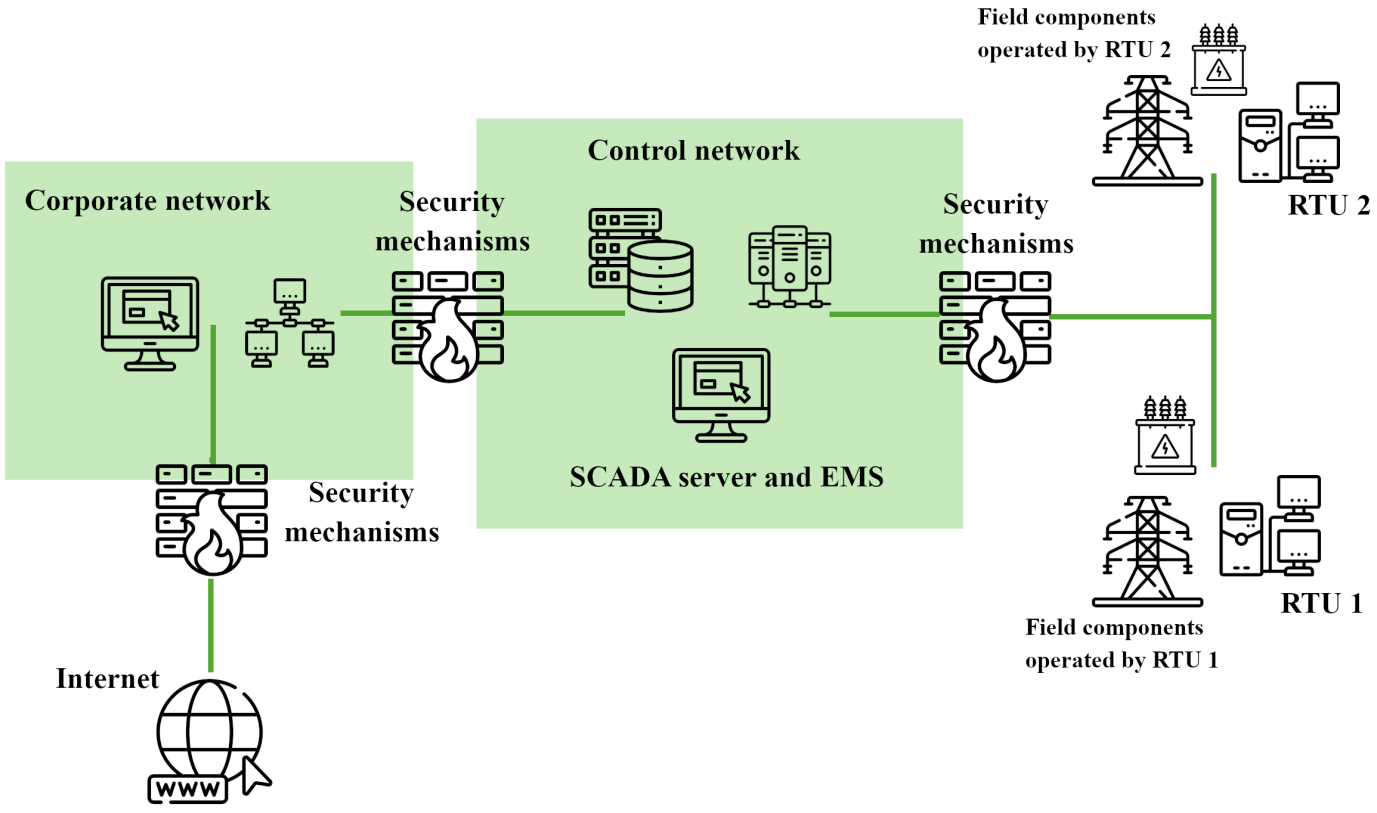
Overview

- Motivation
- Attack model
- Approach
- Truly distributed approach I
- Side Note I
- Side Note II
- Truly distributed approach II
- Properties for real-life test case
- Next steps

Motivation

- smart and decentralised (automated) energy management comes with risks
- we need more accurate data to work with flexible renewables
- legacy SCADA software is under more & more attacks

Motivation

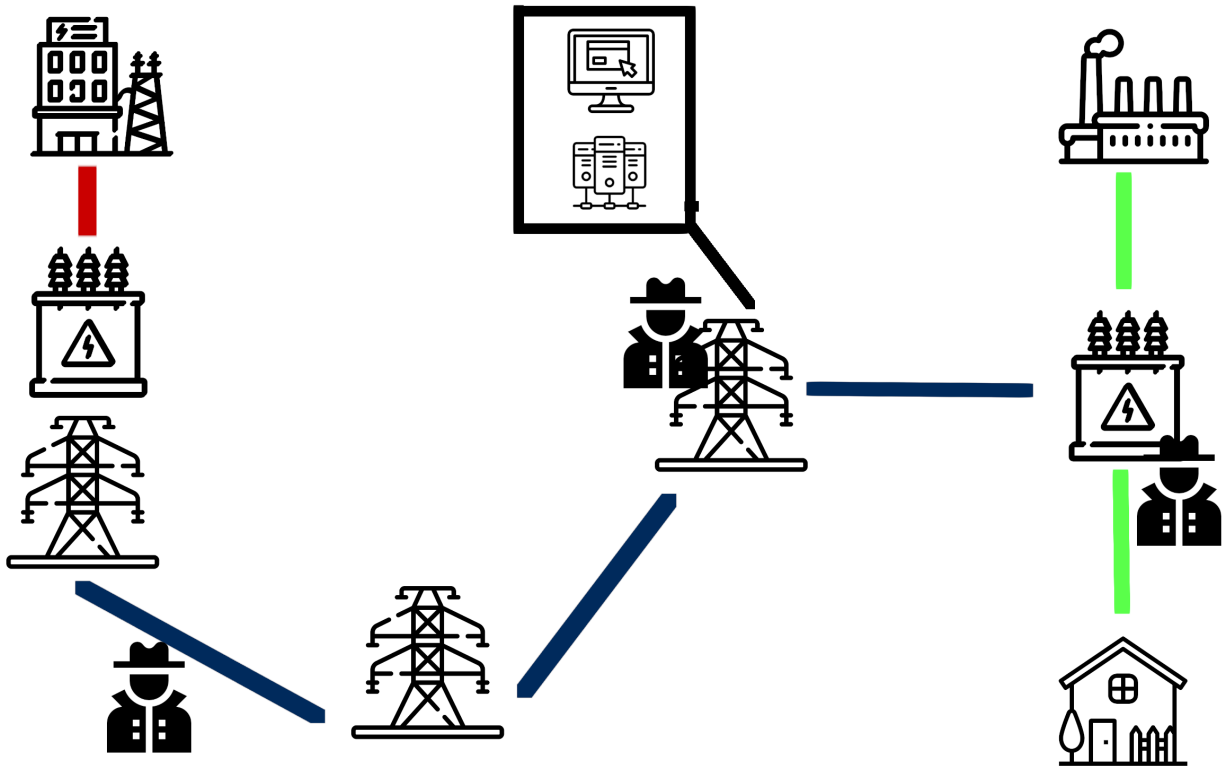


paradigm of *security through obscurity* and *air gap*

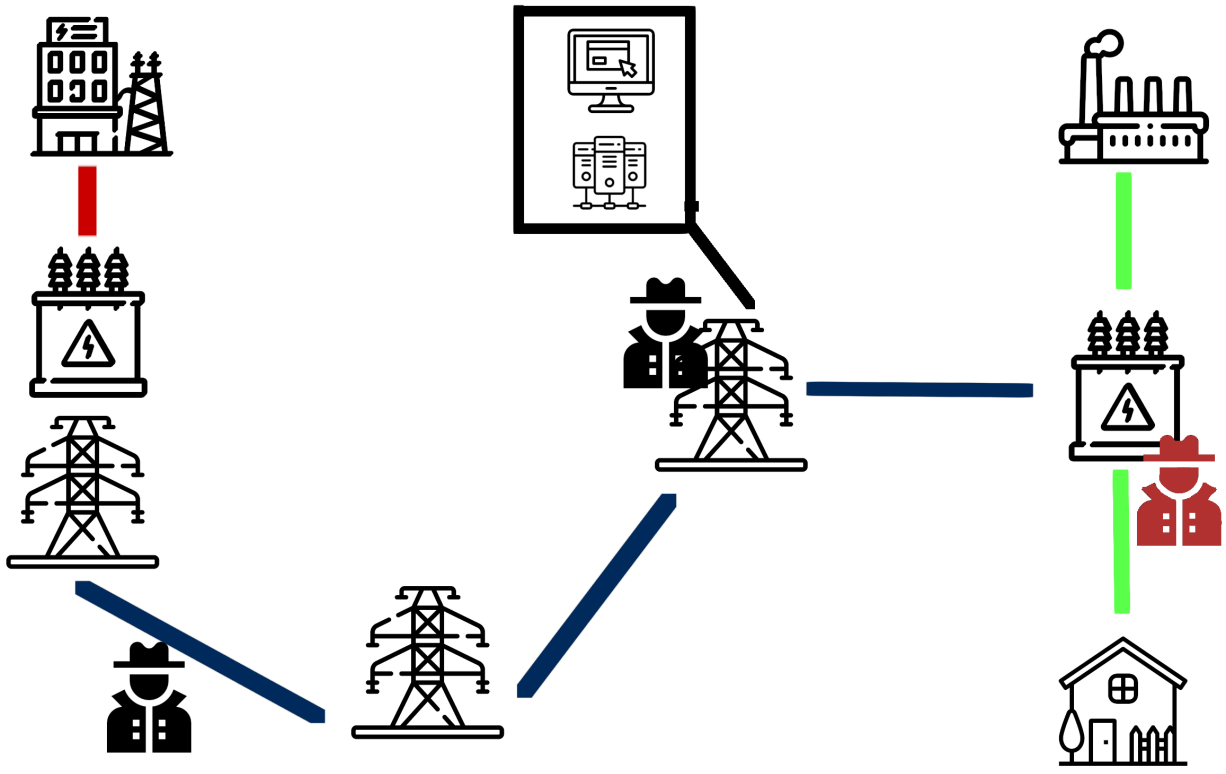
Motivation

- lack of (good) training data
- state estimation (SE):
  assumption communicated data is correct
- multiple security solutions are needed

Attack model



Attack model



Attack model

- able to get into SCADA communication
- eavesdrop, intercept, manipulate and exchange messages
- knowledge about protocols and common grid architecture

Approach

- process-awareness[1]

[1] J. J. Chromik, A. Remke, and B. Haverkort, “An integrated testbed for locally monitoring SCADA systems in smart grids,” Energy Informatics, vol. 1, pp. 1 29, 2018.

[2] V. Menzel, J. L. Hurink and A. Remke, "Securing SCADA networks for smart grids via a distributed evaluation of local sensor data," SmartGridComm, 2021, pp. 405-411.

Approach

- process-awareness[1]
- securing the last mile

[1] J. J. Chromik, A. Remke, and B. Haverkort, “An integrated testbed for locally monitoring SCADA systems in smart grids,” Energy Informatics, vol. 1, pp. 1 29, 2018.

[2] V. Menzel, J. L. Hurink and A. Remke, "Securing SCADA networks for smart grids via a distributed evaluation of local sensor data," SmartGridComm, 2021, pp. 405-411.

Approach

- process-awareness[1]
- securing the last mile

- fixed set of rules to decide:
  does this data/command make sense?

[1] J. J. Chromik, A. Remke, and B. Haverkort, “An integrated testbed for locally monitoring SCADA systems in smart grids,” Energy Informatics, vol. 1, pp. 1 29, 2018.

[2] V. Menzel, J. L. Hurink and A. Remke, "Securing SCADA networks for smart grids via a distributed evaluation of local sensor data," SmartGridComm, 2021, pp. 405-411.

Approach

- process-awareness[1]
- securing the last mile
- fixed set of rules to decide:
  does this data/command make sense?
- local evaluation: attacks within a field station

[1] J. J. Chromik, A. Remke, and B. Haverkort, “An integrated testbed for locally monitoring SCADA systems in smart grids,” Energy Informatics, vol. 1, pp. 1 29, 2018.

[2] V. Menzel, J. L. Hurink and A. Remke, "Securing SCADA networks for smart grids via a distributed evaluation of local sensor data," SmartGridComm, 2021, pp. 405-411.

Approach

- process-awareness[1]
- securing the last mile

- fixed set of rules to decide:
  does this data/command make sense?

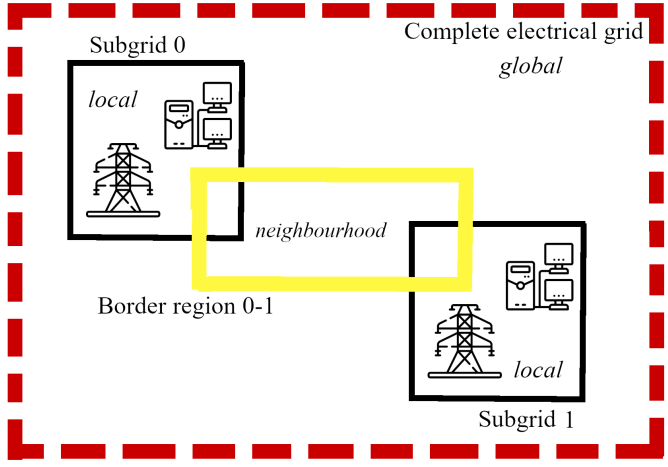
- local evaluation: attacks within a field station
- neighbourhood evaluation[2]: attacks against a complete field station

[1] J. J. Chromik, A. Remke, and B. Haverkort, “An integrated testbed for locally monitoring SCADA systems in smart grids,” Energy Informatics, vol. 1, pp. 1 29, 2018.

[2] V. Menzel, J. L. Hurink and A. Remke, "Securing SCADA networks for smart grids via a distributed evaluation of local sensor data," SmartGridComm, 2021, pp. 405-411.

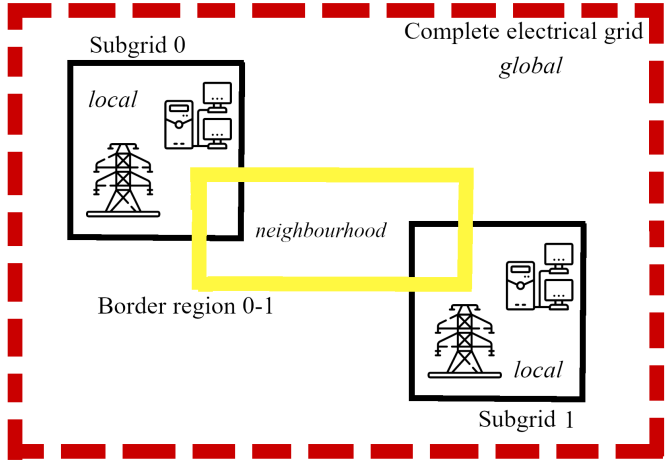
Approach

- two field stations supervising each other



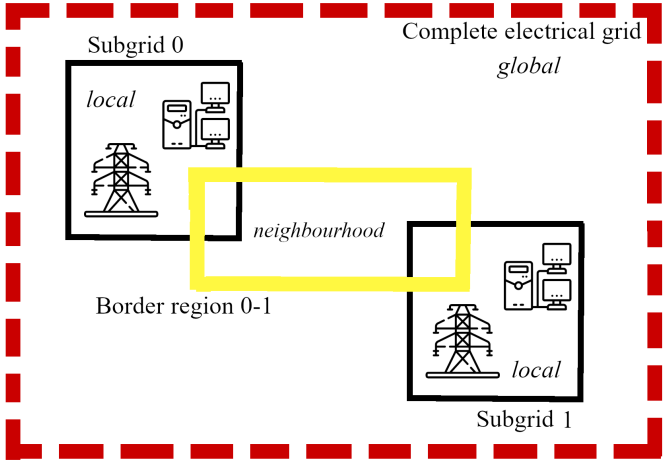
Approach

- two field stations supervising each other
- check done with every neighbour



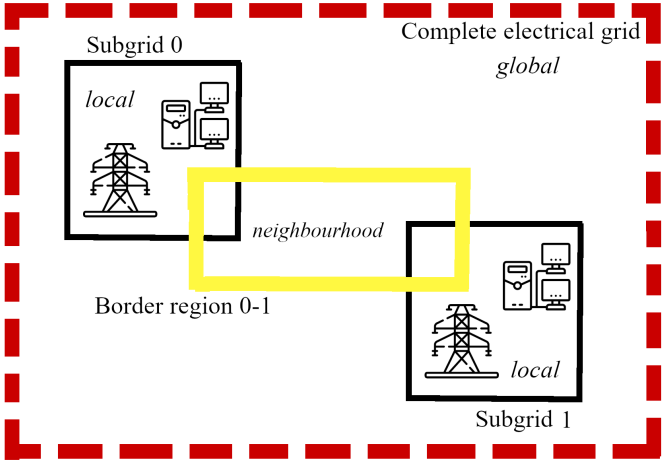
Approach

- two field stations supervising each other
- check done with every neighbour
- additional communication channels



Approach

- two field stations supervising each other
- check done with every neighbour
- additional communication channels
- OPC-UA



Approach

- formal model for splitting the grid & set of requirements

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region
- attack tool for eavesdropping, replay attacks and manipulations

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region
- attack tool for eavesdropping, replay attacks and manipulations
- prototype implementation of an IDS detectable:

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region
- attack tool for eavesdropping, replay attacks and manipulations
- prototype implementation of an IDS detectable:
  - attack within a subgrid

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region
- attack tool for eavesdropping, replay attacks and manipulations
- prototype implementation of an IDS detectable:
  - attack within a subgrid
  - attack within a border region

Approach

- formal model for splitting the grid & set of requirements
- testbed with the co-simulation framework MOSAIK featuring two subgrids and one border region
- attack tool for eavesdropping, replay attacks and manipulations
- prototype implementation of an IDS detectable:
  - attack within a subgrid
  - attack within a border region
  - attack within a subgrid, without causing local alerts

Truly distributed approach I

- before: docker is great!

Truly distributed approach I

- before: docker is great!
- but still one piece of hardware ...

Truly distributed approach I

- before: docker is great!
- but still one piece of hardware ...

- now: testbed, monitors and control and command server each on their own Raspberry Pi

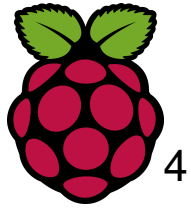
Side note I: docker

[3]

- OS-level virtualization to deliver software in containers
- free (and premium) platform as a service
- makes it easy to start multiple containers in light-weight environments, functioning every time

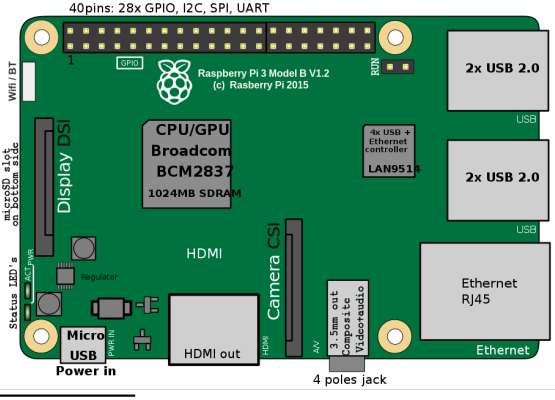
[3] https://en.wikipedia.org/wiki/Docker_(software)#/media/File:Docker_logo.svg

Side note II: Raspberry Pi



[4]

- popular single-board computer for education and DIY projects
- modular system to equip with all different sorts of additions
- here: Raspberry Pi 3 Model B V1.2



[5]

[4] https://en.wikipedia.org/wiki/File:Raspberry_Pi_Logo.svg

[5] https://commons.wikimedia.org/wiki/File:RaspberryPi3B.svg

Truly distributed approach II

Adaptions:

- need to configure wheels for ARM architecture

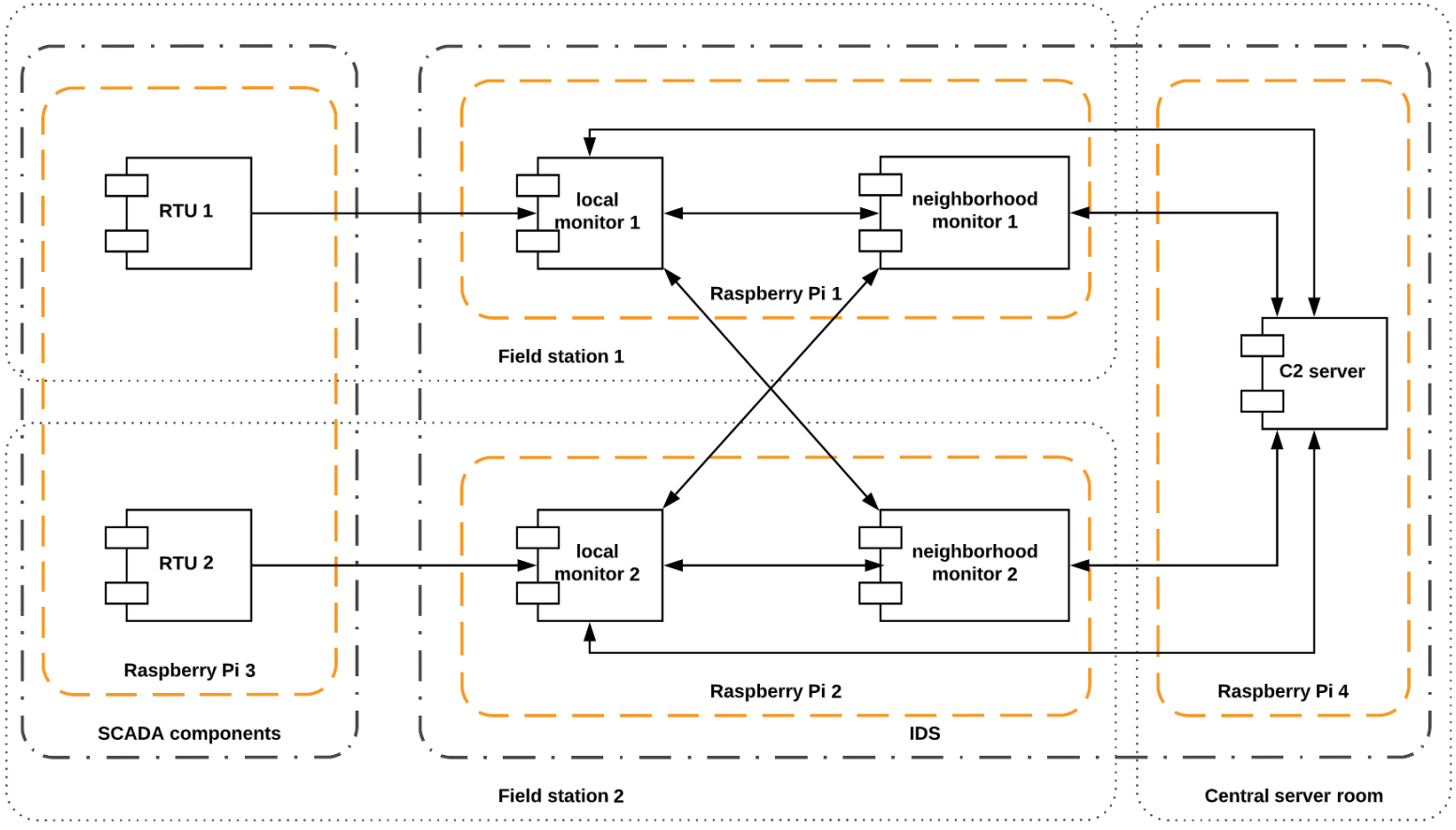
Truly distributed approach II

Adaptions:

- need to configure wheels for ARM architecture
- need to make IP addresses and ports externally available from outside the containers

Truly distributed approach II

Adaptions:



Truly distributed approach II

Results:

- in replay mode: same alerts are triggered as in centralized execution
  (... after bug fix)

Truly distributed approach II

Results:

- in replay mode: same alerts are triggered as in centralized execution
  (... after bug fix)
- all Raspberry Pis are bored

Truly distributed approach II

No value smaller than 85% idle was observed.

Truly distributed approach II

Results:

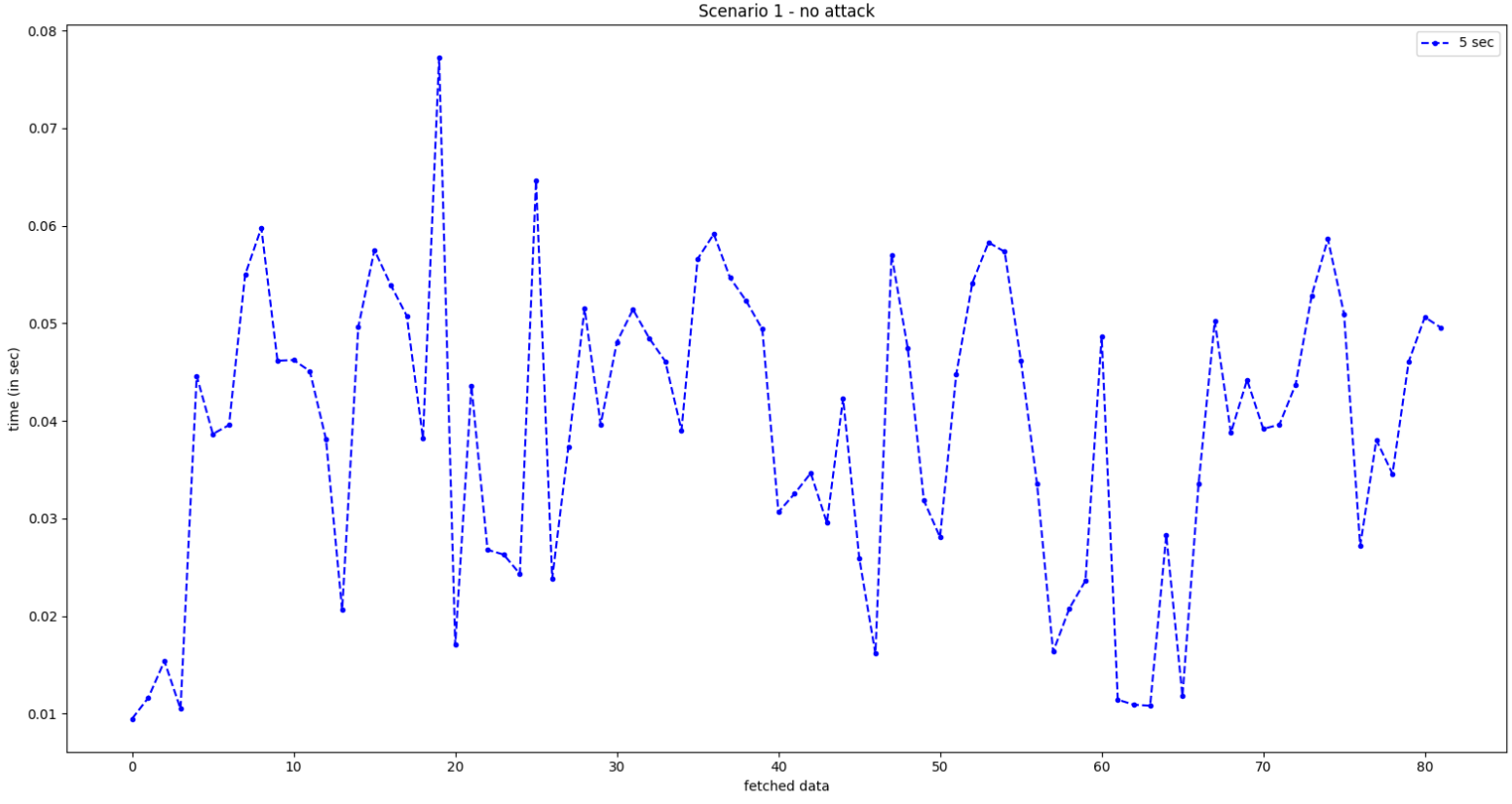
- in replay mode: same alerts are triggered as in centralized execution
  (... after bug fix)
- all Raspberry Pis are bored
- increased speed of new measurement is no problem

Truly distributed approach II

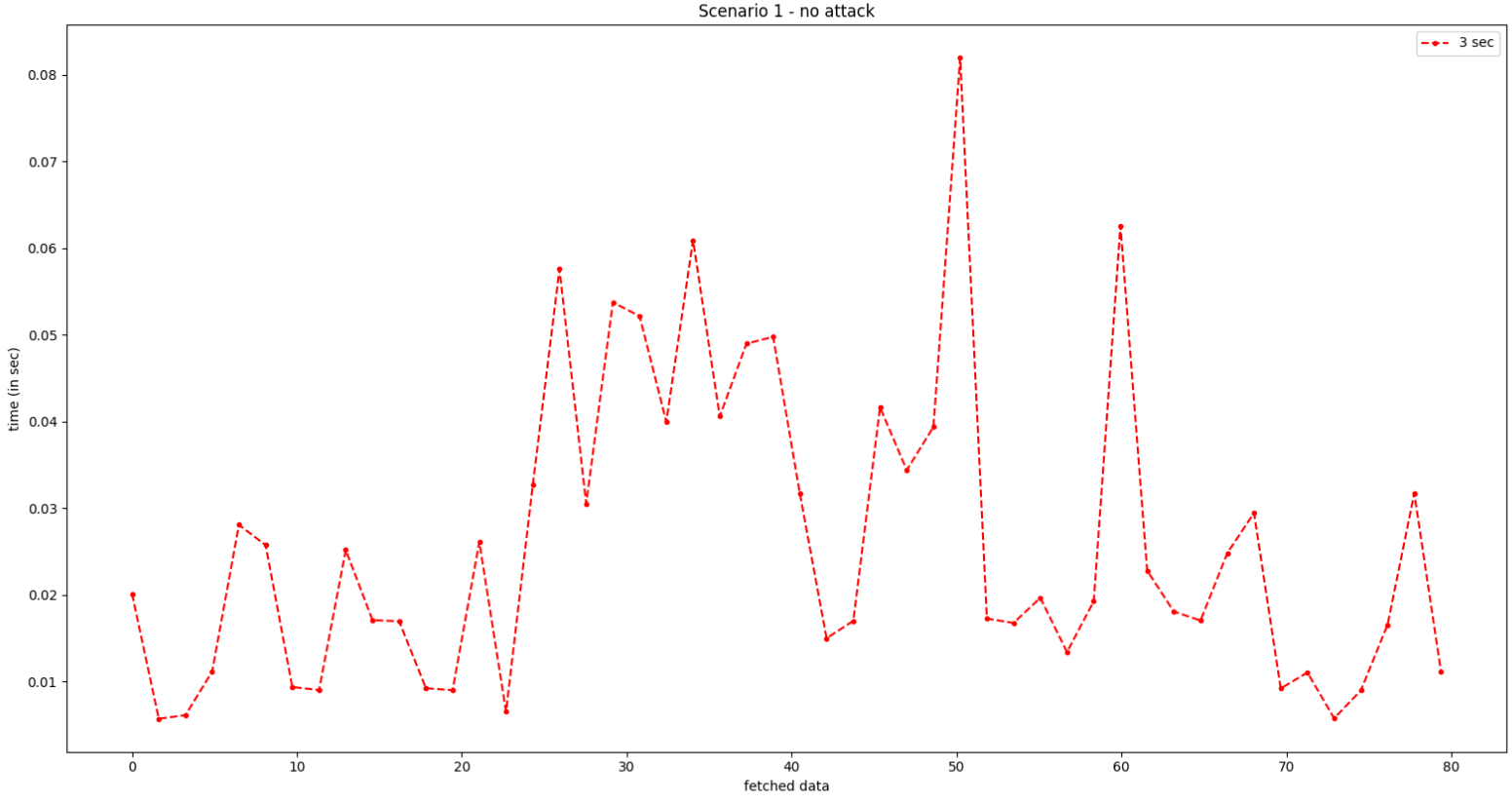
Results:

- in replay mode: same alerts are triggered as in centralized execution
  (... after bug fix)
- all Raspberry Pis are bored
- increased speed of new measurement is no problem
- evaluation is quick

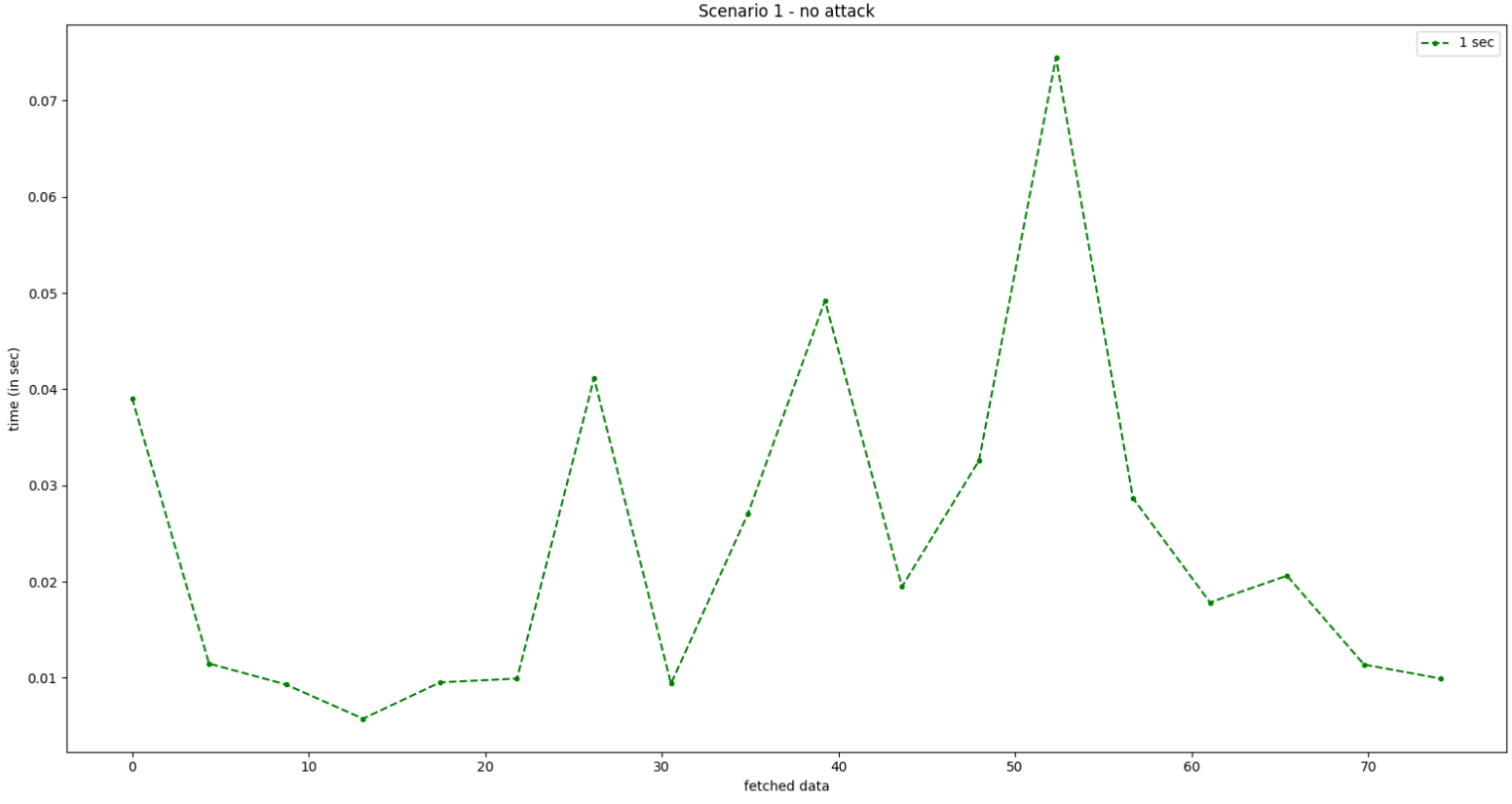
Truly distributed approach II



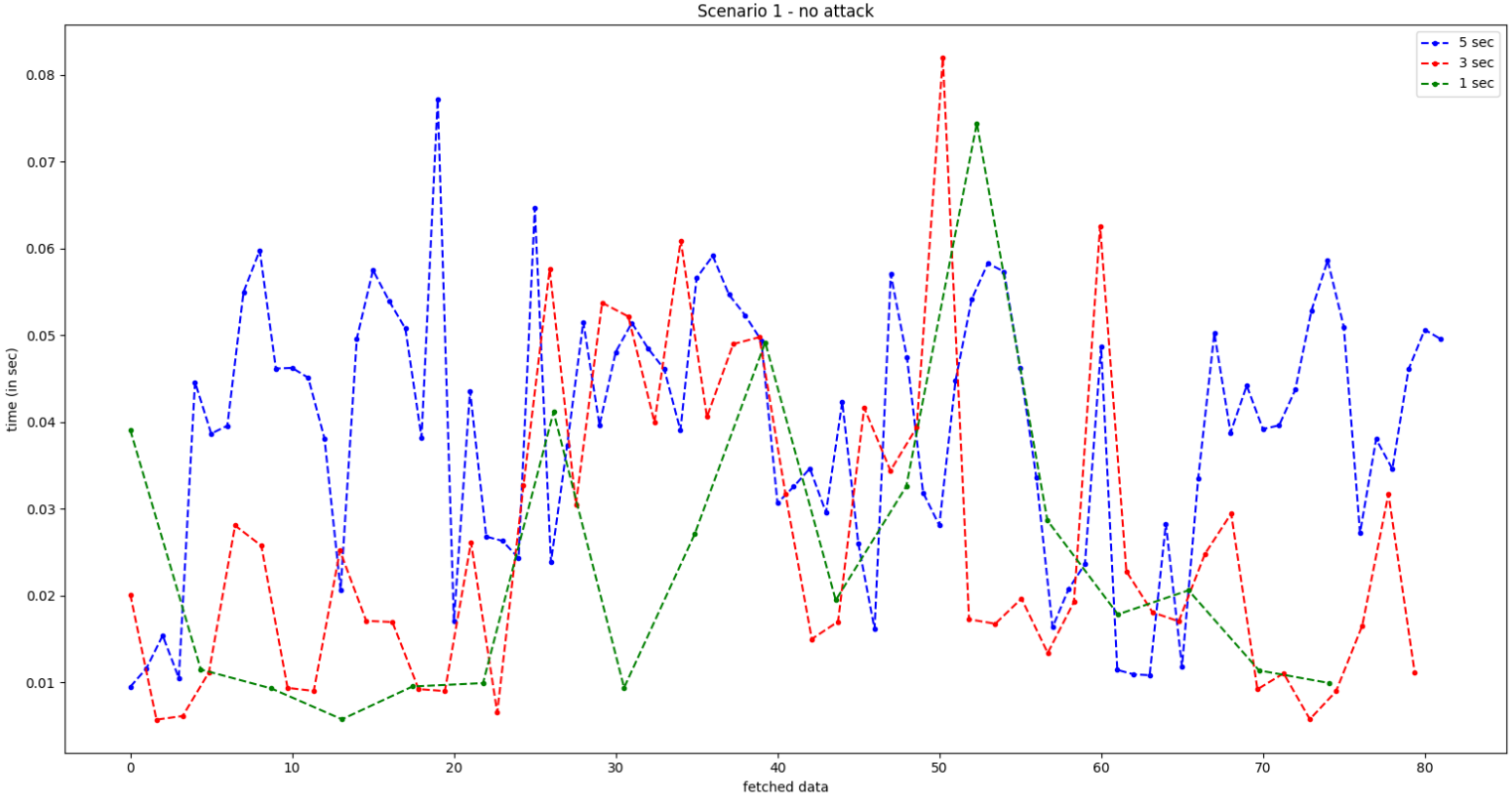
Truly distributed approach II



Truly distributed approach II



Truly distributed approach II



Properties for real-life test case

- Confidentiality
- Integrity
- Availability

Properties for real-life test case

- central command and control sever:
  traditional data center protection

Properties for real-life test case

- central command and control sever:
  traditional data center protection
- monitors:

Properties for real-life test case

- central command and control sever:
  traditional data center protection
- monitors:
  - Raspberry Pis were too strong/strong enough

Properties for real-life test case

- central command and control sever:
  traditional data center protection

- monitors:
  - Raspberry Pis were too strong/strong enough
  - lacking TPM & real-time clock

Properties for real-life test case

- central command and control sever:
  traditional data center protection

- monitors:
  - Raspberry Pis were too strong/strong enough
  - lacking TPM & real-time clock
  - hard to protect against DDOS

Properties for real-life test case

- central command and control sever:
  traditional data center protection

- monitors:
  - Raspberry Pis were too strong/strong enough
  - lacking TPM & real-time clock
  - hard to protect against DDOS
  - BUT we can detect it quickly :-)

Properties for real-life test case

- central command and control sever:
  traditional data center protection
- monitors:
  - Raspberry Pis were too strong/strong enough
  - lacking TPM & real-time clock
  - hard to protect against DDOS
  - BUT we can detect it quickly :-)
- in general: OPC-UA is a good choice (for C, I and A)

Properties for real-life test case

- central command and control sever:
  traditional data center protection
- monitors:
  - Raspberry Pis were too strong/strong enough
  - lacking TPM & real-time clock
  - hard to protect against DDOS
  - BUT we can detect it quickly :-)
- in general: OPC-UA is a good choice (for C, I and A)
- VPN could be an addition

Next steps

- real-world data
- thresholds
- more scenarios, more Raspberry Pis
- other HW
- experimenting with different monitor positions

- questions?