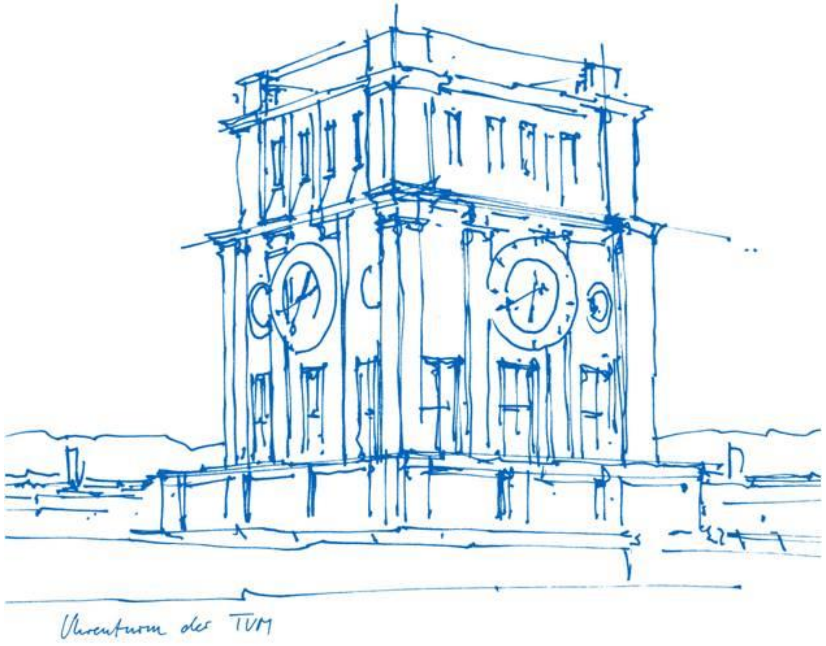
Chair of Network Architectures and Services
Department of Computer Engineering
TUM School of Computation, Information, and Technology
Technical University of Munich

Reproducible Experiments of Threshold Cryptography Functions and Trusted Execution Environments - METHODA



Authors:
Filip Rezabek, Kilian Glas, Richard von Seck,
Achraf Aroua, Tizian Leonhardt, Georg Carle
rezabek@net.in.tum.de

Refresher

Trusted Execution Environments

A TEE is an isolated region in a CPU

- Promises to be a secure location for code and data
- Offers confidentiality and integrity to varying degrees
- Additionally, (remote) attestation usually available
- Measure and attestate the state of a system to assess trustworthiness

Refresher
Trusted Execution Environments

Requires CPU support, different vendors offer different capabilities

- Intel – Intel SGXv1 and v2, TDX
- AMD – AMD SEV, SEV-SN, and SEV-SNP
- ARM – TrustZone, *CCA*
- *RISC-V – Keystone, …*

VM-based vs Process-based

We opt for a VM-based TEE for our solution

Specifically: **AMD SEV-SNP** [1]

[1] - https://github.com/AMDESE/AMDSEV

Refresher

Trusted Execution Environments

What did we select as a VM?

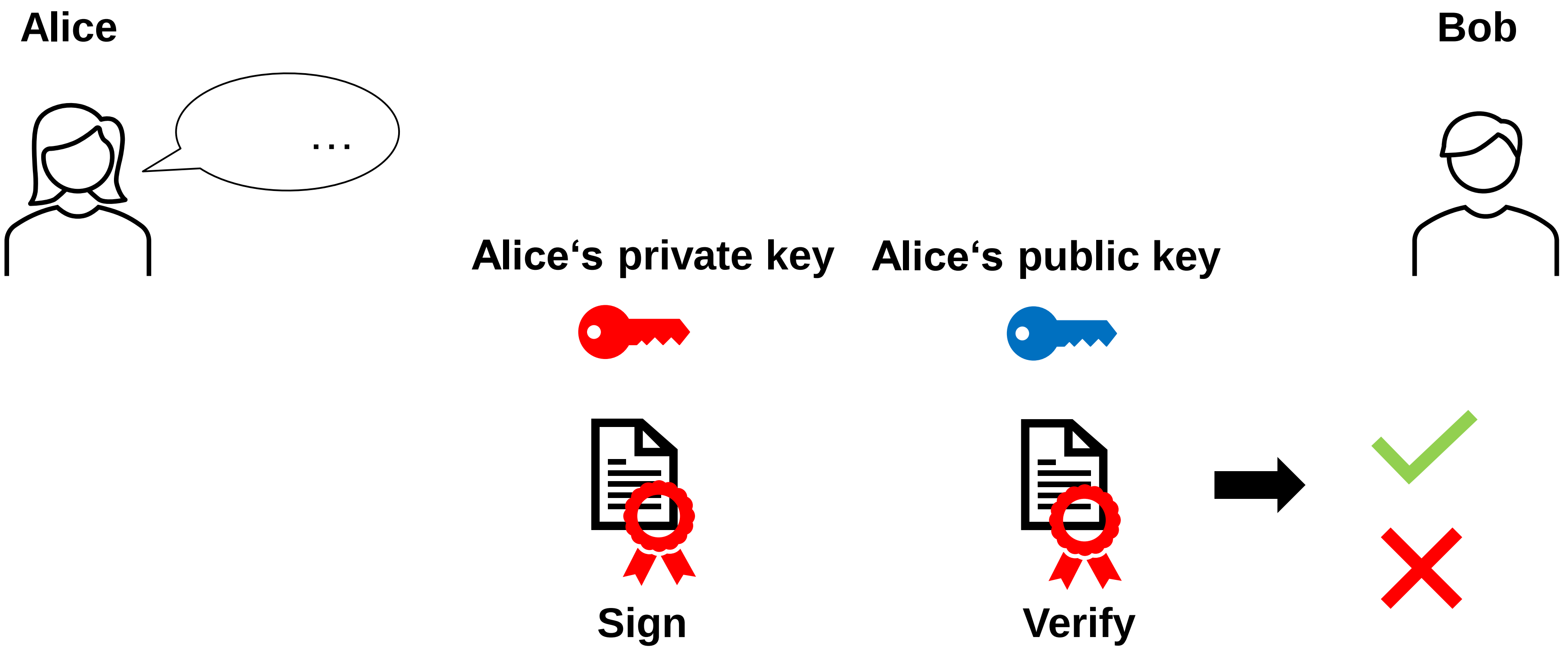
- We need a VM-based solution that offers good performance and is easy to deploy

→ Our choice: Kata Containers [1]

- Kata aims to be as performant as containers with the isolation of **VMs**
- Supports Intel TDX and **AMD SEV-SNP**
- Fully OCI-compliant, can be used as drop-in container runtime for **Docker** [2] and others

[1] - https://katacontainers.io/
[2] - https://www.docker.com/

Refresher
Public Key Cryptography



Refresher
Threshold Cryptography

Classical public key cryptography

- Single point of attack and failure on the private key

Possible solution – threshold cryptography

- Distributed private key into multiple shares
- Store the shares among *n* parties
- During signing or decryption require at least threshold *t* parties

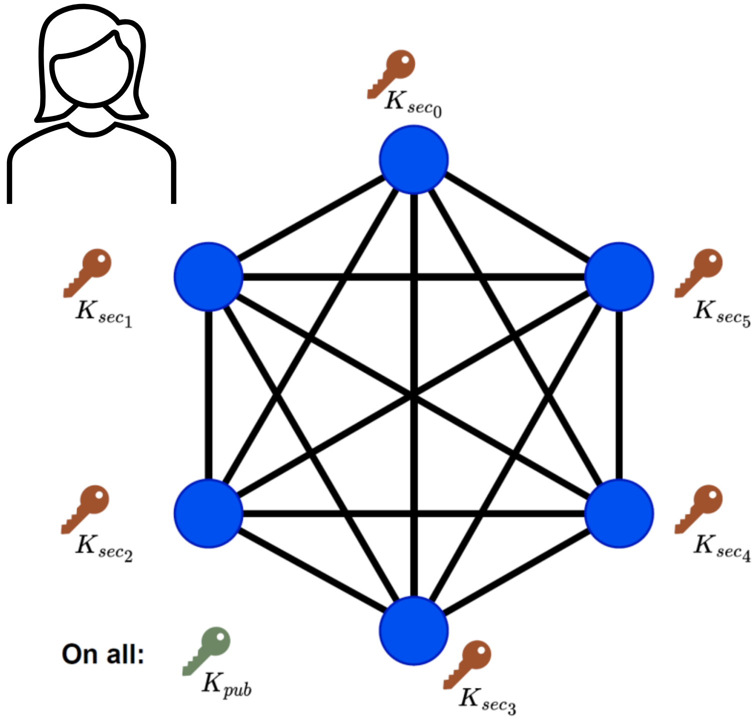
Available and secure as long as *t-1* parties online

Focus on the threshold signing

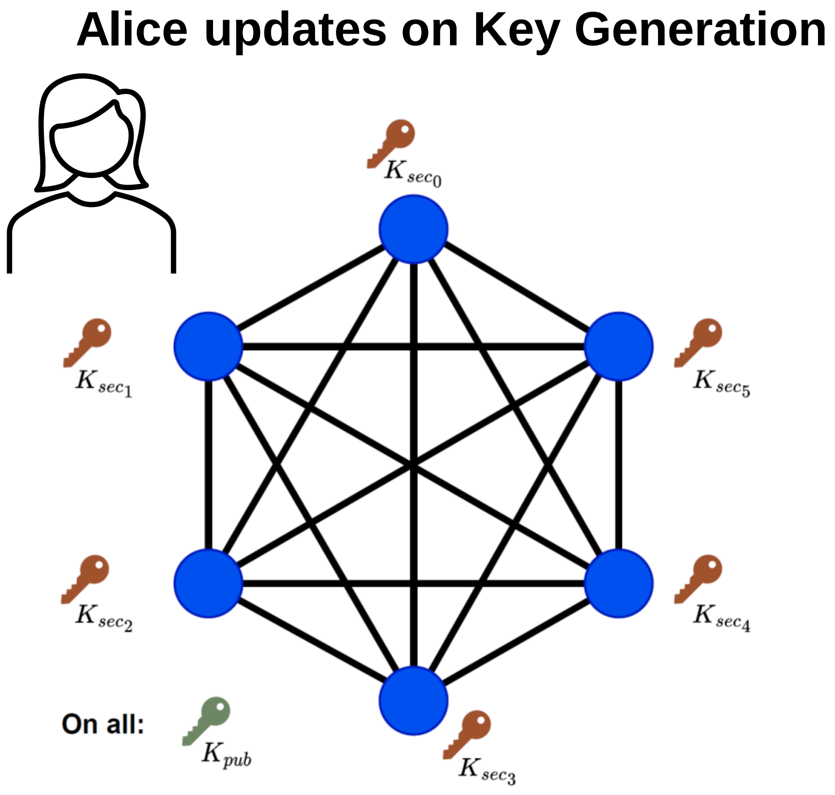
- Applications for servers as signing services (cryptographic wallets), IoT devices, …

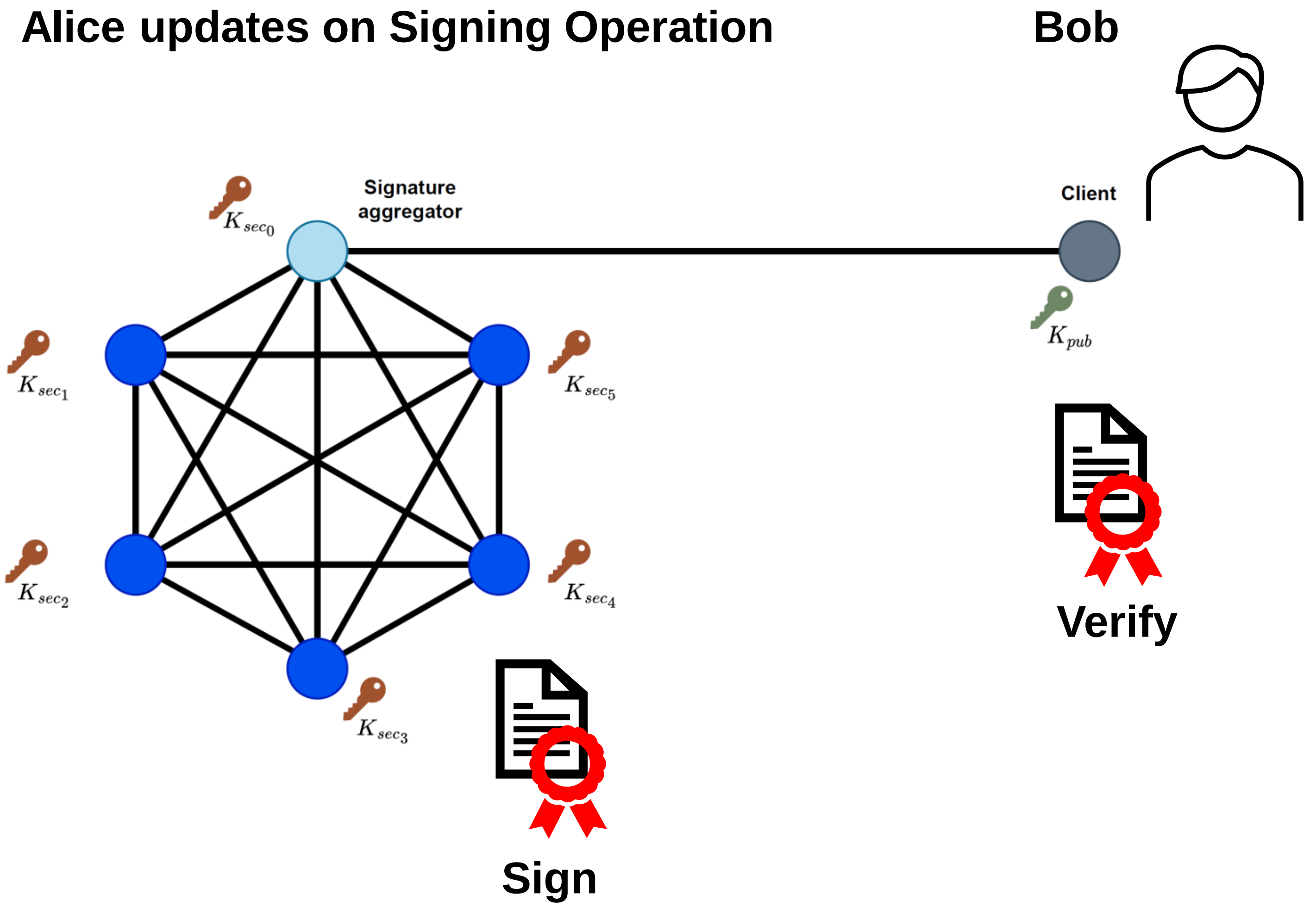
Refresher
Threshold Signing – Update on Operations

Alice updates on Key Generation



Refresher
Threshold Signing – Update on Operations



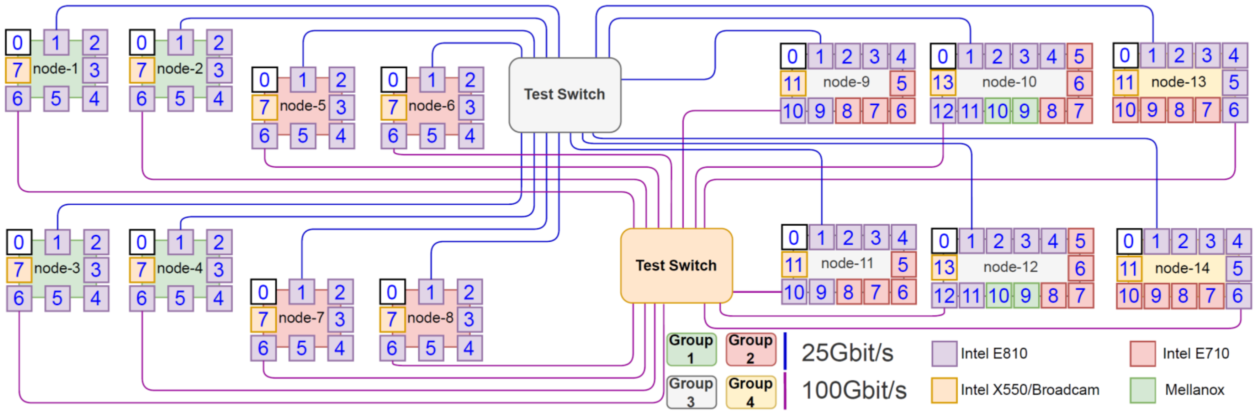


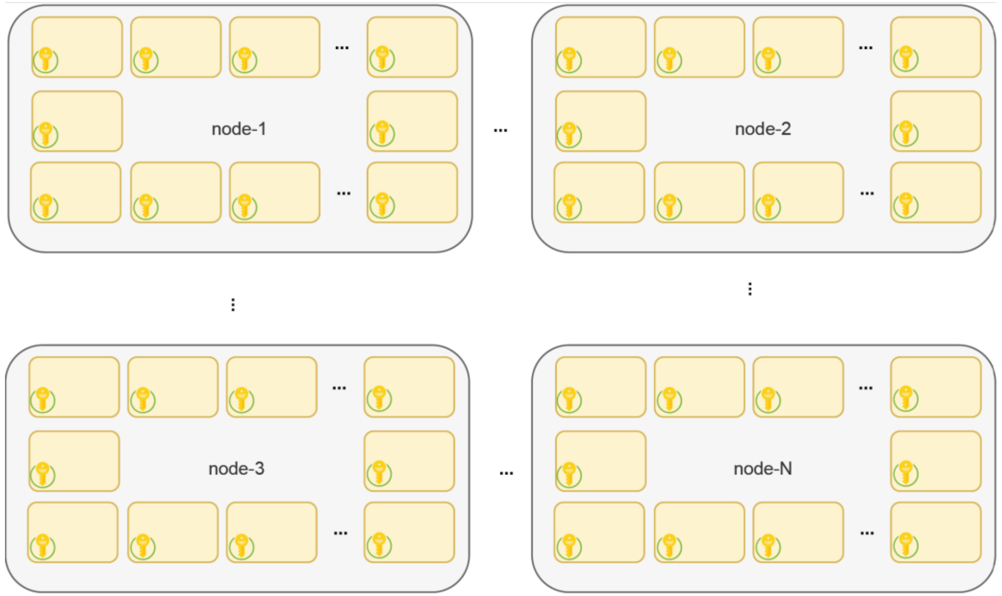
Introduction

Motivation

Structured approach to assessing the capabilities of various distributed systems e.g., **cryptographic protocols**, peer-to-peer systems, and privacy preserving systems …

- In a **reproducible** manner
- Evaluate of such deployments in **scale**
- Handle **heterogenous** setups without loosing granularity





Introduction

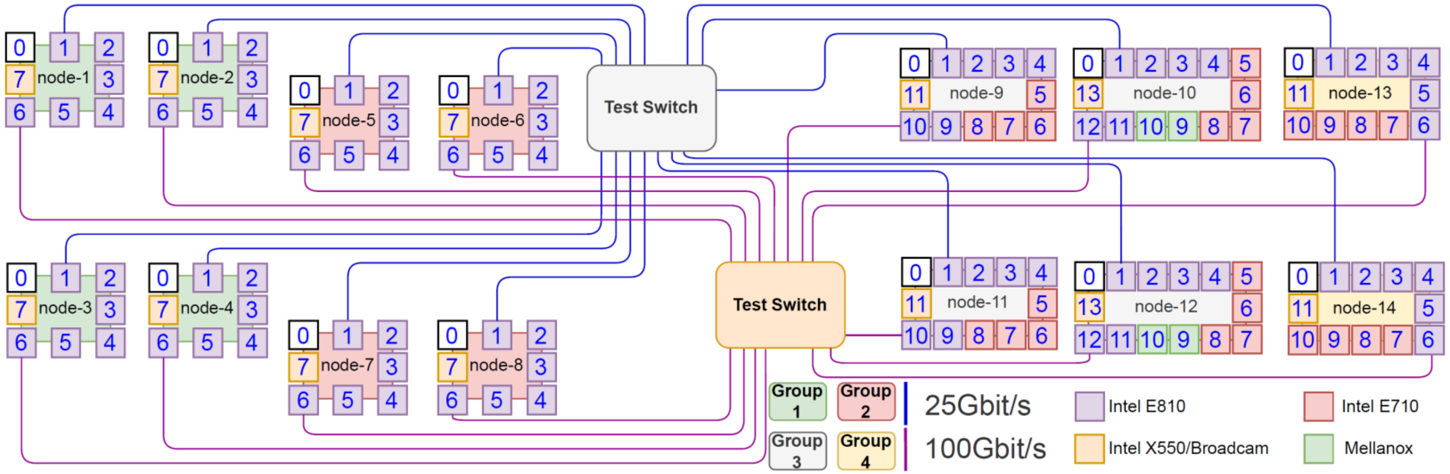
Motivation

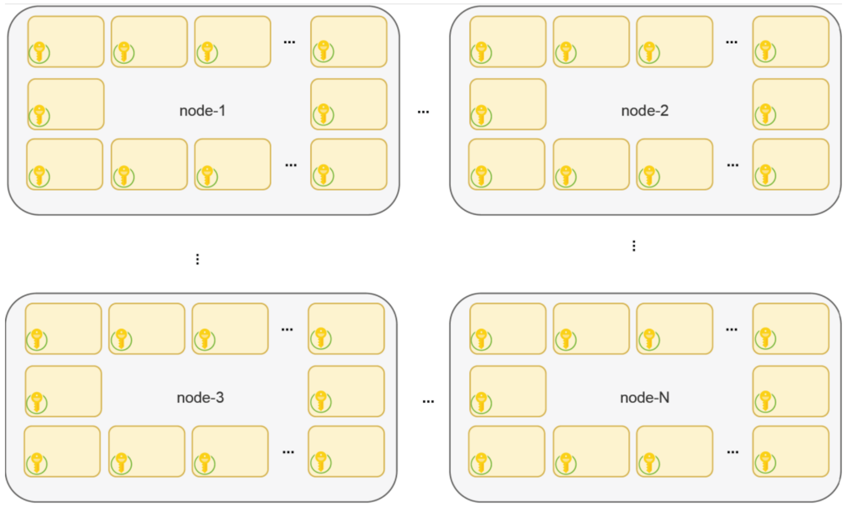
Structured approach to assessing the capabilities of various distributed systems e.g., **cryptographic protocols**, peer-to-peer systems, and privacy preserving systems …

- In a **reproducible** manner
- Evaluate of such deployments in **scale**
- Handle **heterogenous** setups without loosing granularity

Serves as a base that can be used for improvements and optimizations

- Trusted Executed Environments impact on performance
- Optimization of the performance of such systems



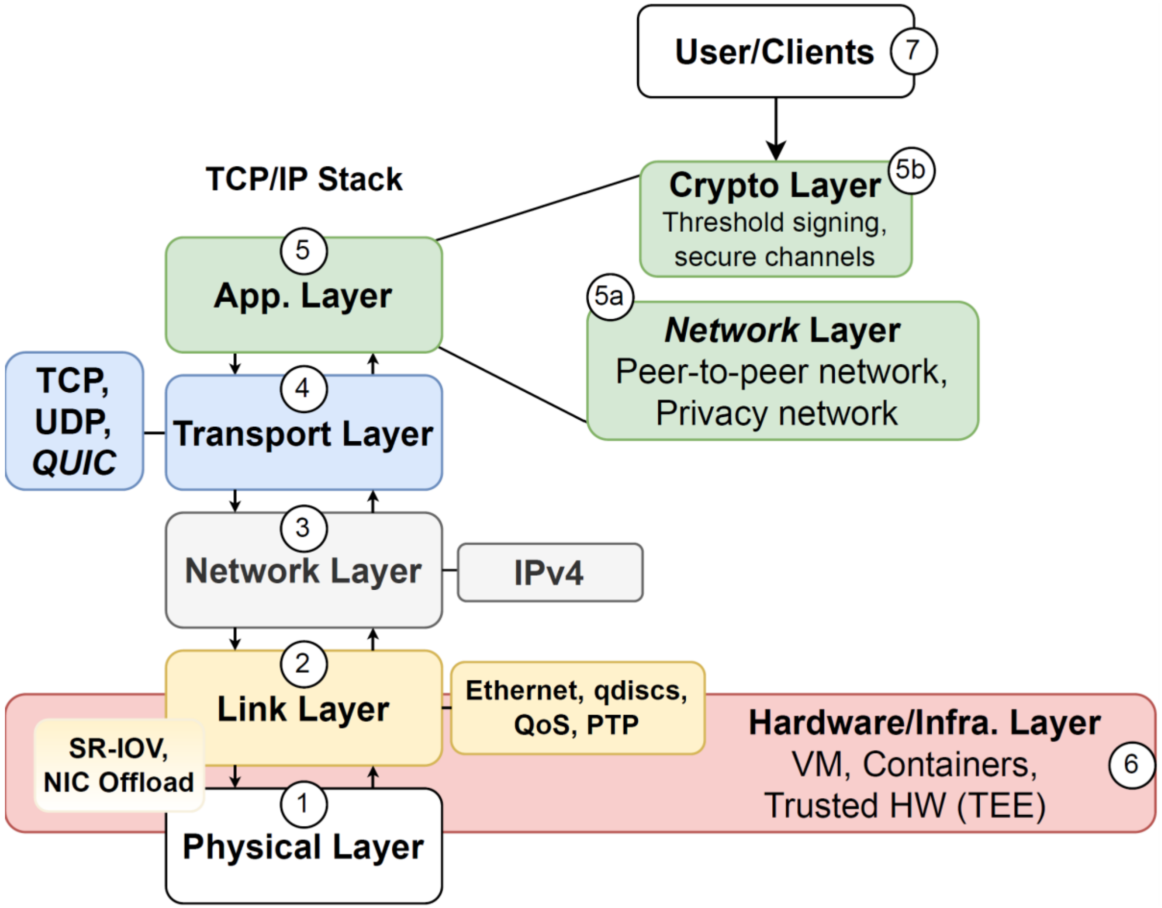


Analysis

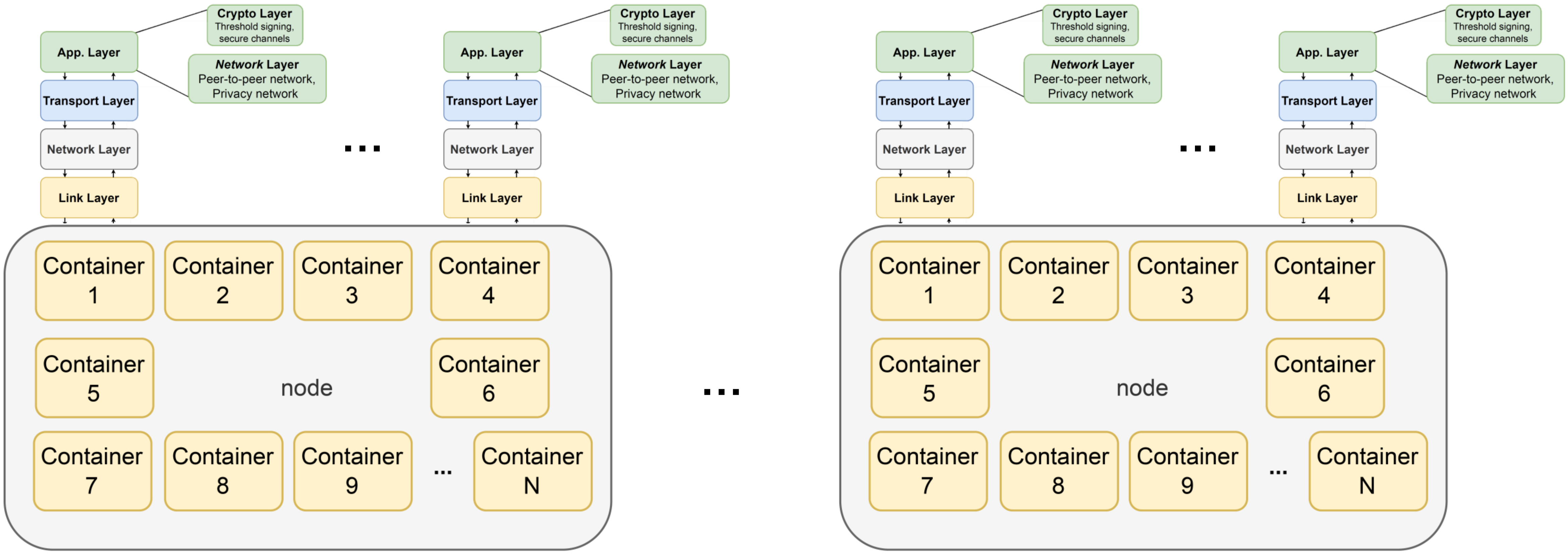
Overview of Deployment Stack

Generalization step towards various distributed systems

1) Handle various speeds, ports
2) Ethernet with e.g., TSN shapers or other *qdiscs*
3) Mainly IPv4
4) Transport Layer – UDP or UDP
5) Application layer
    a) Peer-to-Peer
    b) Crypto operations – DKG, Signing
6) Deployment on Host OS or Containers
7) Interact with the systems for load generation



Analysis
Overview of the Deployment



Design

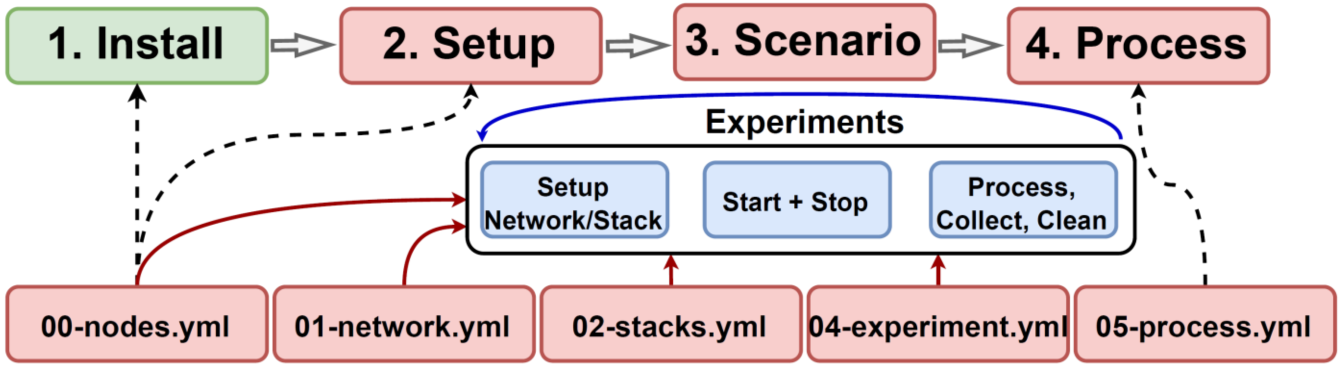
Extension of EnGINE - METHODA

METHODA – **M**ultilayer **E**nvironment and **T**oolchain for **H**olistic Netw**O**rk **D**esign and **A**nalysis

Extends **EnGINE** capabilities

Define **once**, use **multiple** times

- *01-network.yml*
- *02-stacks.yml*
- *04-experiment.yml*



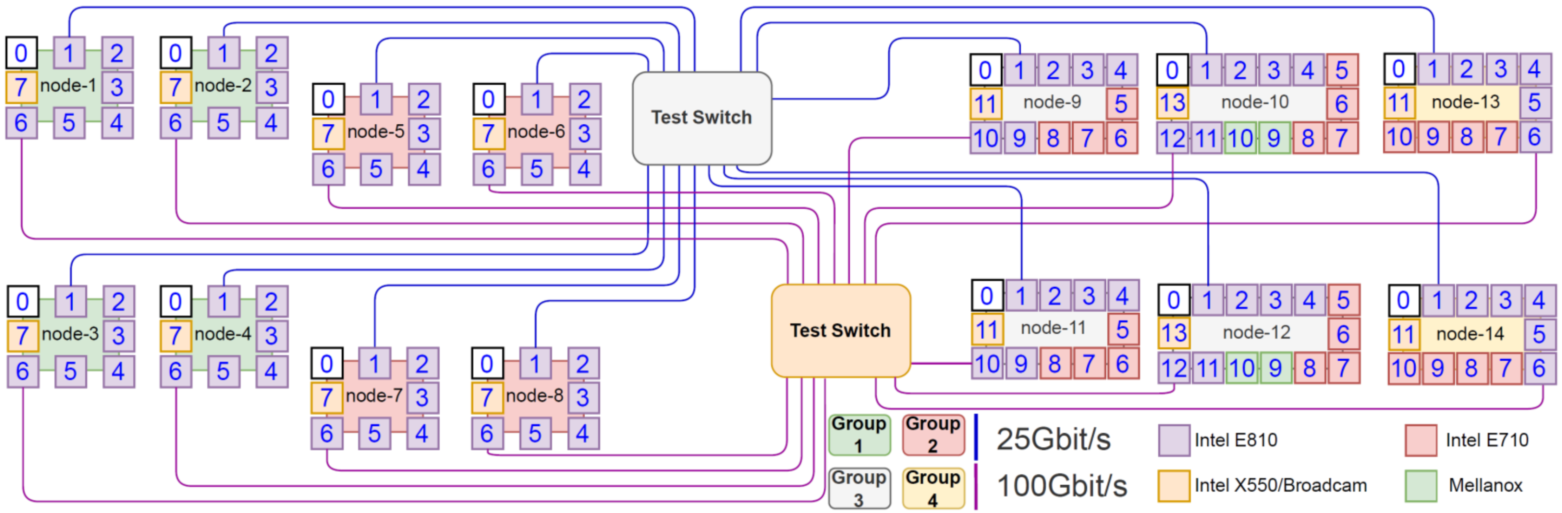
Change at *00-nodes.yml* the definition of containers or physical peers

- Kata, Docker, and Linux Containers (LXC)
  - Granular HW specs definition
  - Usage of CPU Isolation and Affinity for the LXC containers to lower the noise

→ Identified a “standardized” definition of experiments that is *easy to read* and can be comparable among different deployments

Design
Handle Heterogenous HW Configurations

	CPU (cores/threads)	RAM	NICs
Group 1 (4x)	24C/48T Intel® Xeon Gold 6312U	512 GB DDR4	4× 25 GbE E810-C†, 2× 100 GbE E810-XXV† 2× 10 GbE X552†
Group 2 (4x)	32C/64T AMD EPYC 7543	512 GB DDR4	4× 25 GbE E810-C†, 2× 100 GbE E810-XXV† 2× 10 GbE BCM574†
Group 3 (4x)	32C/64T AMD EPYC 9354	768 GB DDR5	4× 25 GbE E810-C†, 2× 100 GbE E810-C† 2× 10 GbE BCM574†, 4× 10 GbE X710†
Group 4 (2x)	32C/64T Intel® Xeon Gold 6421N	512 GB DDR5	2× 100 GbE E810-XXV†, 2× 100 GbE MT28908$^{\dagger\ \ddagger}$ 2× 10 GbE X552†, 4× 10 GbE X710†, 4× 25 GbE E810-C†



Design of Experiments
Metrics and Parameters

Evaluate the TEE overhead and Threshold Scheme called FROST [1,2]

Kata in the **AMD SEV-SNP**:
- CPU-bound matrix multiplication benchmark
- Memory-bound triad benchmark

FROST [3]:
- Whitebox testing – individual operations
- Blackbox testing – end-to-end latency
- Number of nodes and threshold values, and message sizes

Setup:
- 4 nodes, each with up to 8 containers, each with 2 CPU cores
- 1 node with up to 32 containers, each with 2 CPU cores

AMD EPYC 7543, 32C/64T
AMD EPYC 9354, 32C/64T

[1] - Chelsea Komlo, Ian Goldberg, FROST: Flexible Round-Optimized Schnorr Threshold Signatures, 2020
[2] - https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/
[3] - https://github.com/isislovecruft/frost-dalek

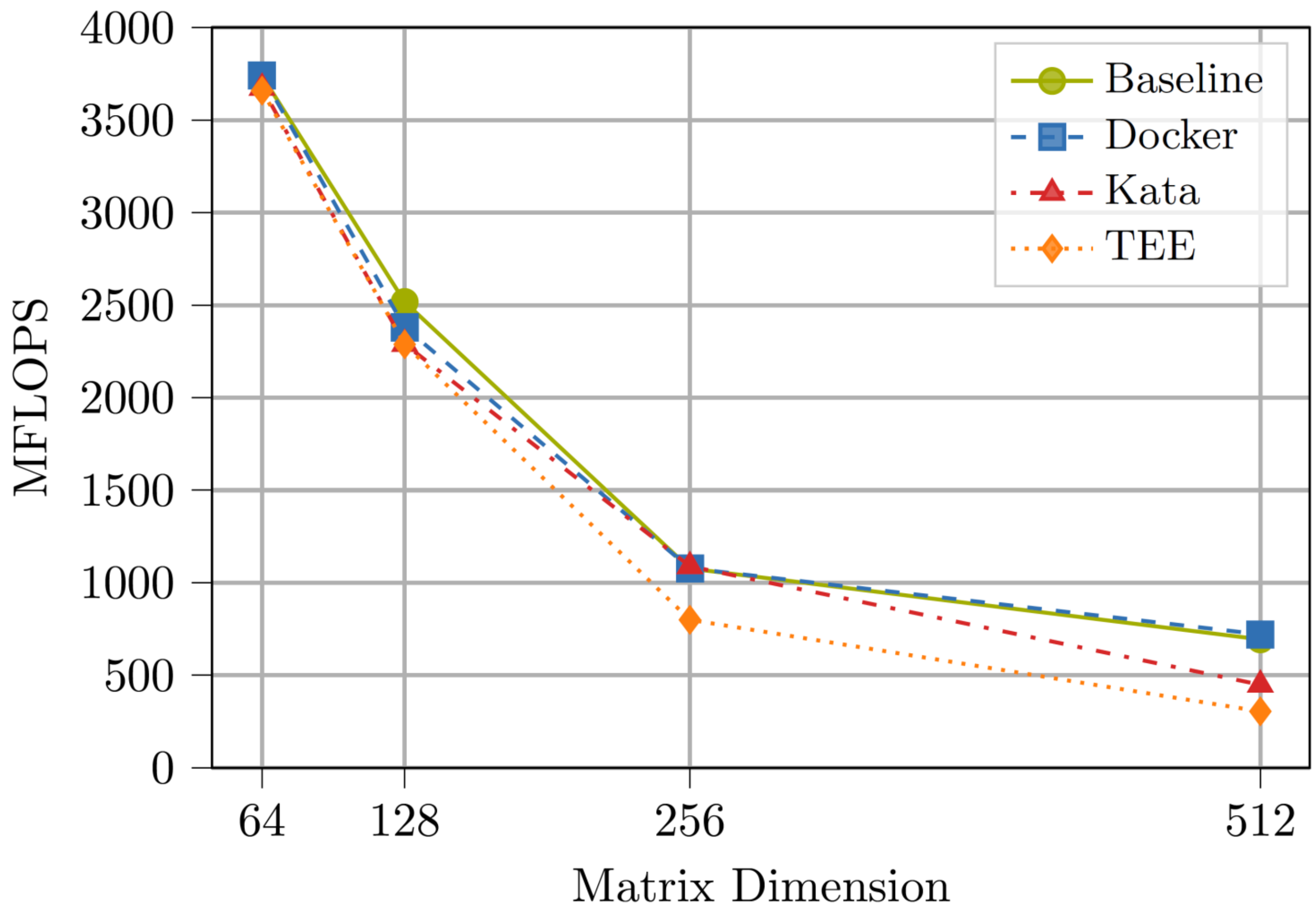
Preliminary Evaluation

TEE Overhead – Compute Bound

Matrix multiplications

$$a_{i,j} = \sum_{k=0}^{n} b_{i,k} * c_{k,j}$$

The value $b_{i,k}$ stored in the cache
→ compute bound

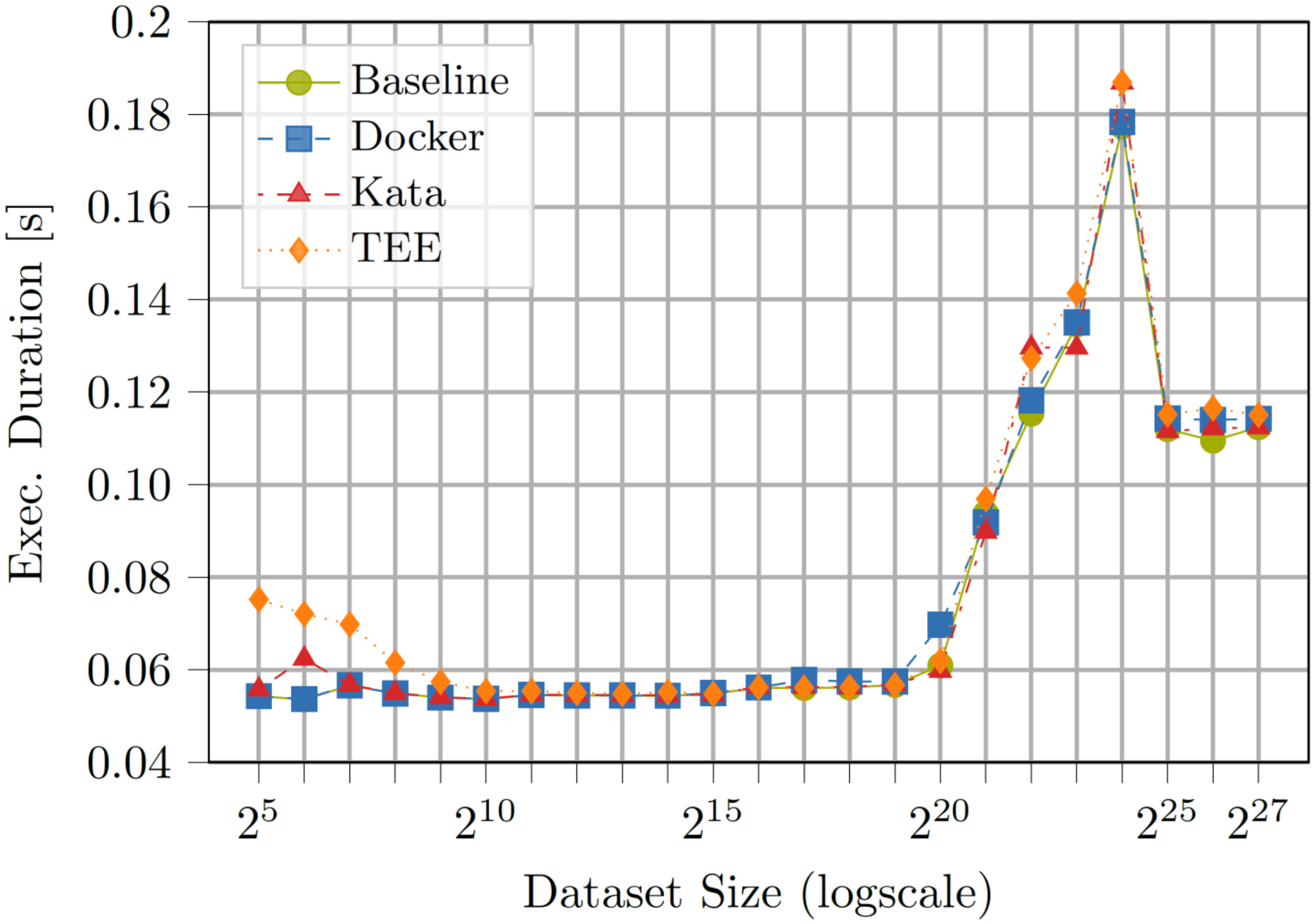


Preliminary Evaluation
TEE Overhead – Memory-Bound Triad

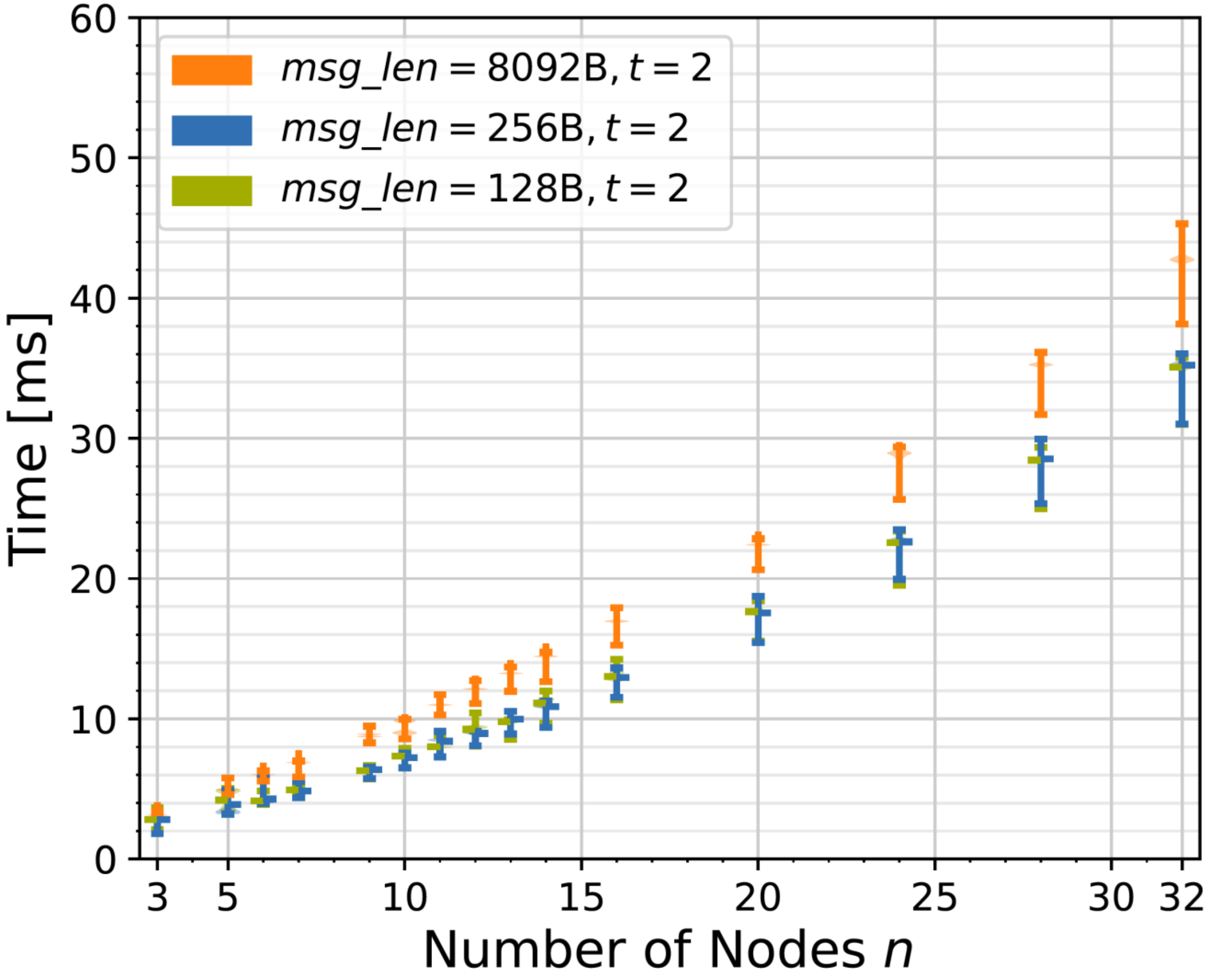
Data set:

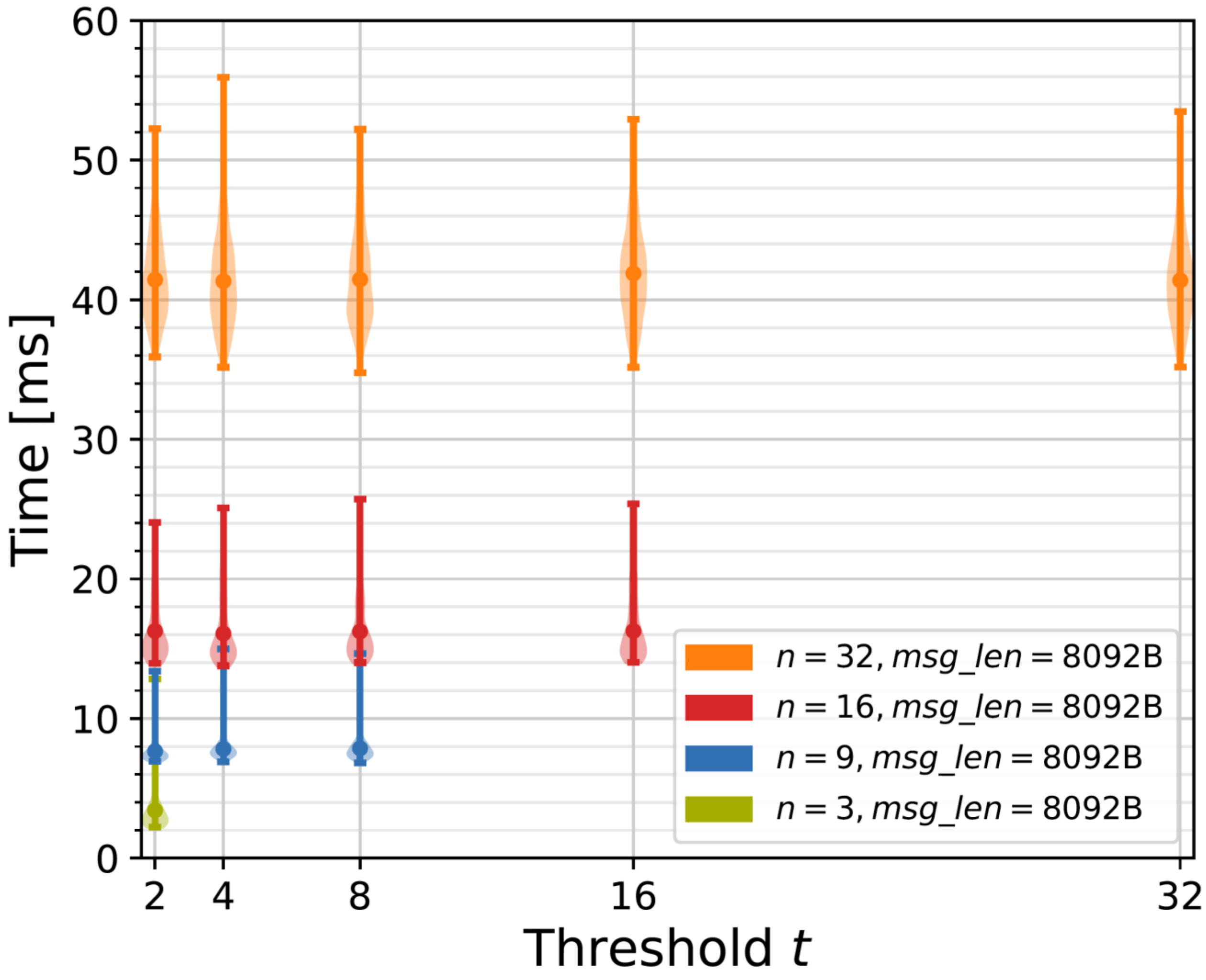
$$\vec{a} = \vec{b} + \vec{c} * \vec{d}$$

Asses how fast the vectors loaded to CPU



Preliminary Evaluation
Threshold Schnorr – no TEE vs TEE





Summary

Key takeaways and future work

Identified suitable solutions for scalable evaluation of various applications

Steppingstone to achieve combined view on interaction of individual building blocks

Current TEE solution introduces less overhead in comparison to previous versions

Extend METHODA setup

- TEE from various chip providers – heterogenous TEEs
- Additional cryptographic algorithms

Use-case evaluation

Optimizations using the features of EnGINE

Thank You!

Filip

@rezabfil

@rezabfil

rezabek@net.in.tum.de

Preprint: Filip Rezabek, Kilian Glas, Richard von Seck, Achraf Aroua, Tizian Leonhardt, and Georg Carle
Multilayer Environment and Toolchain for Holistic NetwOrk Design and Analysis, Nov., 2023

Preliminary Evaluation
Threshold Schnorr – local setup

DKG:

- Participation Creation
- Round One
- Round Two
- LT Verification Shares
- Finish

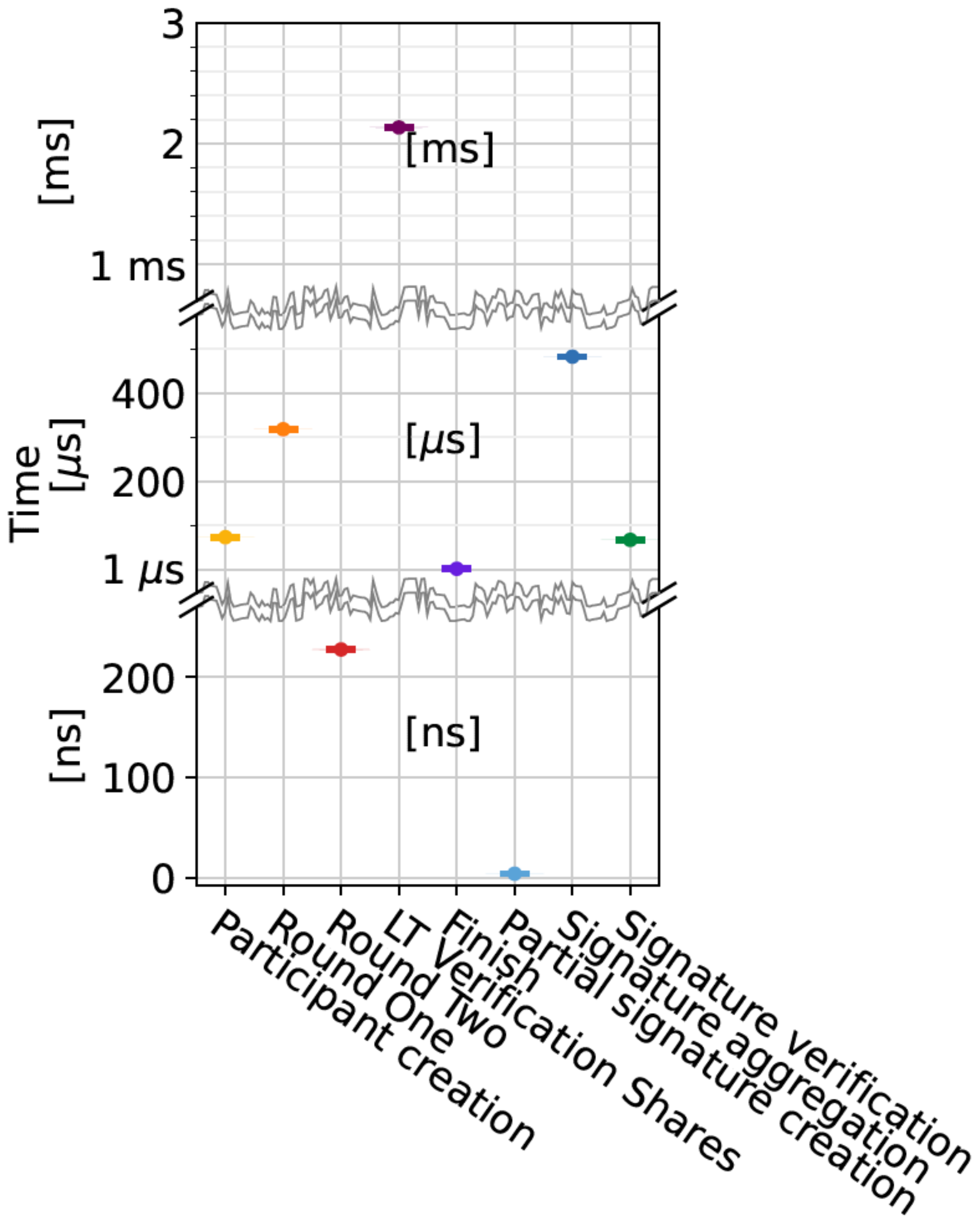
Signing:

- Partial Signature Creation
- Signature Aggregation

→ ~500us

Verification

- Signature Verification



Related Work

Evaluation methodology

- Experiment parameters
  - Number of peers
  - Threshold
  - Runtime config
  - Message size
  - HW specs
  - Fault injection

- Requirements definition

	[57]	[38]	[52]	[72]	[74]	[49]	[65]	[56]	Us
Type	?	†	†	ζ	ζ	‡	‡	‡	‡
R1: Repeat	✓	×	✓	✓	✓	✓	✓	×	✓
R2: Reproduce	×	×	○	×	✓	✓	○	×	✓
R3: Replicate	×	×	×	×	✓	✓	×	×	✓
R4: Openness	?	✓	✓	?	✓	✓	✓	✓	✓
R6: Autonomy	✓	○	✓	✓	✓	✓	✓	✓	✓
R7: MalScenario	✓	○	○	✓	✓	○	×	×	✓
R9: Gran.Con.	○	○	○	?	○	○	○	○	✓
R12: Scalable	?	○	○	✓	✓	○	✓	✓	✓
R14: Diversity	○	×	×	○	○	○	○	○	✓
R15: Standard	?	○	○	?	×	○	○	○	✓

METHODA Design
Overview

Orchestrated from the management host

Management Host Data Repository

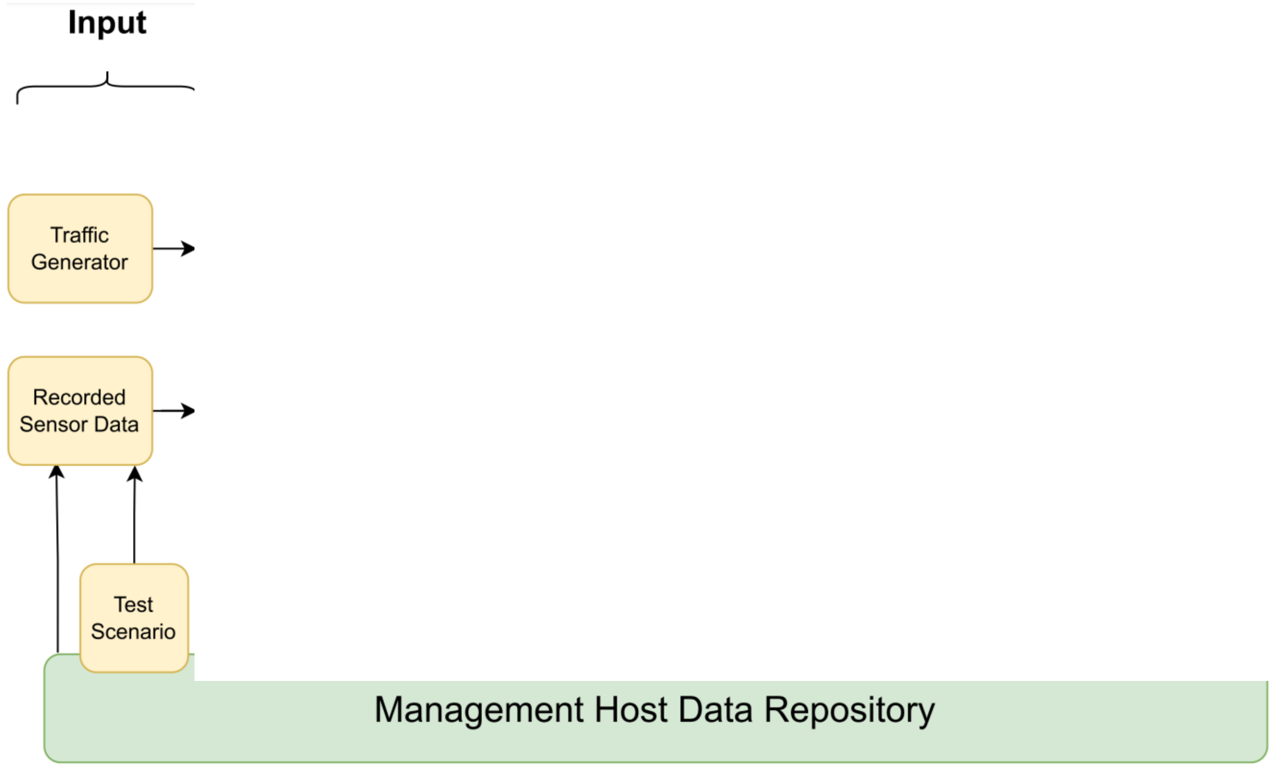
METHODA Design
Overview

Orchestrated from the management host

Three parts of each experiment

Input

- Defines the experiment
- Specifies data sources and network



METHODA Design

Overview

Orchestrated from the management host

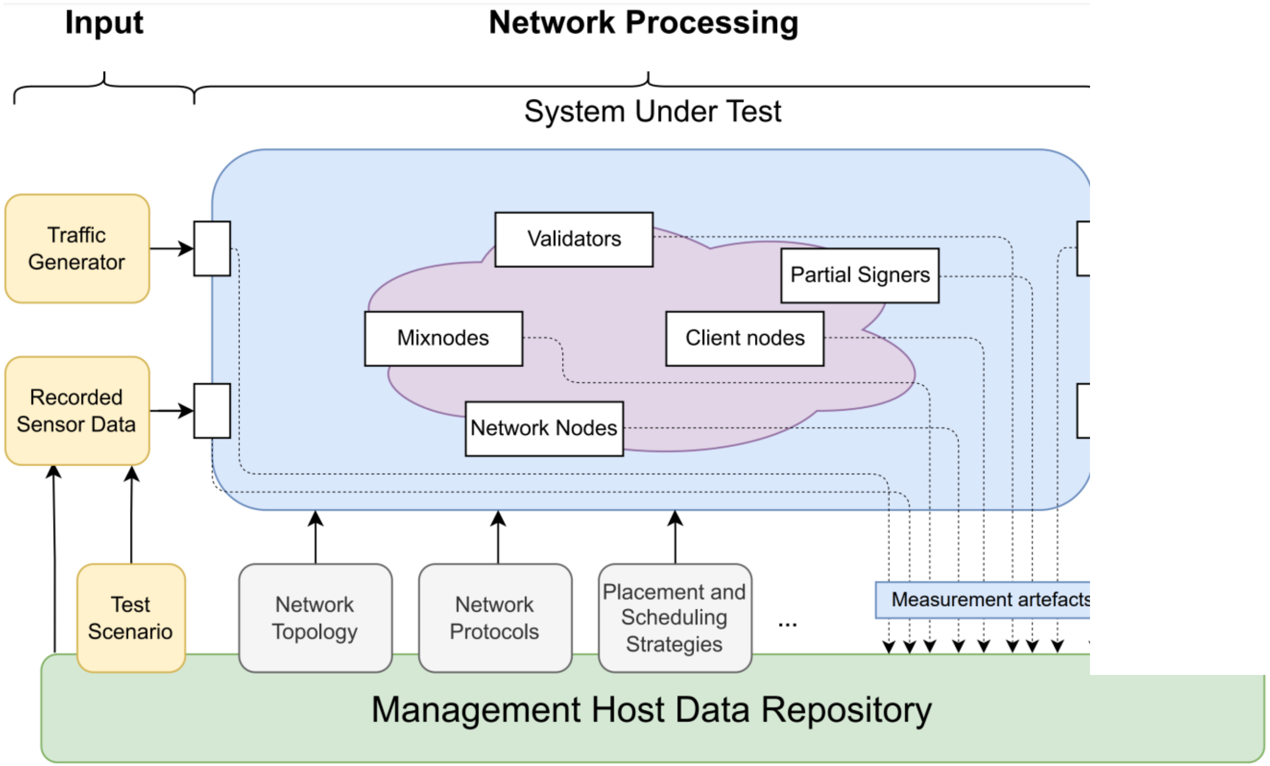
Three parts of each experiment

Input

- Defines the experiment
- Specifies data sources and network

Network Processing

- Encompasses the tested system
- Takes configuration from input
- Supports the experiment



METHODA Design
Overview

Orchestrated from the management host

Three parts of each experiment

Input
- Defines the experiment
- Specifies data sources and network

Network Processing
- Encompasses the tested system
- Takes configuration from input
- Supports the experiment

Output
- Records experiment results
- Can include physical actuation