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

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]

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

Refresher
Public Key Cryptography

Alice

Bob

Alice’s private key
Alice’s public key

Sign
Verify

Bob's signature is valid.
Bob's signature is invalid.
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

On all:

$K_{sec}$

$K_{pub}$
Refresher
Threshold Signing – Update on Operations

Alice updates on Key Generation

- $K_{sec0}$
- $K_{sec1}$
- $K_{sec2}$
- $K_{sec3}$
- $K_{sec4}$

On all:
- $K_{pub}$

Bob

Sign

Verify

Alice updates on Signing Operation

- $K_{sec0}$
- $K_{sec1}$
- $K_{sec2}$
- $K_{sec3}$
- $K_{sec4}$

Signature aggregator

Client

$K_{pub}$

Bob

Sign

Verify
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 losing 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 losing 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 \textit{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 – Multilayer Environment and Toolchain for Holistic NetwOrk Design and Analysis

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

<table>
<thead>
<tr>
<th>Group</th>
<th>CPU (cores/threads)</th>
<th>RAM</th>
<th>NICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>24C/48T Intel® Xeon Gold 6312U</td>
<td>512 GB DDR4</td>
<td>4x 25 GbE E810-C, 2x 100 GbE E810-XXV, 2x 10 GbE X552</td>
</tr>
<tr>
<td>Group 2</td>
<td>32C/64T AMD EPYC 7543</td>
<td>512 GB DDR4</td>
<td>4x 25 GbE E810-C, 2x 100 GbE E810-XXV, 2x 10 GbE BCM574</td>
</tr>
<tr>
<td>Group 3</td>
<td>32C/64T AMD EPYC 9354</td>
<td>768 GB DDR5</td>
<td>4x 25 GbE E810-C, 2x 100 GbE E810-C, 2x 10 GbE X710</td>
</tr>
<tr>
<td>Group 4</td>
<td>32C/64T Intel® Xeon Gold 6421N</td>
<td>512 GB DDR5</td>
<td>2x 100 GbE E810-XXV, 2x 100 GbE MT28908, 2x 10 GbE X552, 4x 10 GbE X710, 4x 25 GbE E810-C</td>
</tr>
</tbody>
</table>
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

**[1]** - Chelsea Komlo, Ian Goldberg, FROST: Flexible Round-Optimized Schnorr Threshold Signatures, 2020
**[3]** - [https://github.com/isislovecruff/frost-dalek](https://github.com/isislovecruff/frost-dalek)
Preliminary Evaluation
TEE Overhead – Compute Bound

Matrix multiplications

\[ a_{i,j} = \sum_{k=0}^{n} b_{i,k} \cdot c_{k,j} \]

The value \( b_{i,k} \) stored in the cache → compute bound
Preliminary Evaluation
TEE Overhead – Memory-Bound Triad

Data set:

\[ \vec{a} = \vec{b} + \vec{c} \times \vec{d} \]

Asses how fast the vectors loaded to CPU
Preliminary Evaluation
Threshold Schnorr – no TEE vs TEE

- $msg_{len} = 8092B, t = 2$
- $msg_{len} = 256B, t = 2$
- $msg_{len} = 128B, t = 2$

Time [ms]

Number of Nodes $n$

Time [ms]

Threshold $t$
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!

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

<table>
<thead>
<tr>
<th>Type</th>
<th>[57]</th>
<th>[38]</th>
<th>[52]</th>
<th>[72]</th>
<th>[74]</th>
<th>[49]</th>
<th>[65]</th>
<th>[56]</th>
<th>Us</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: Repeat</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>R2: Reproduce</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>R3: Replicate</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>R4: Openness</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>R6: Autonomy</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>R7: MalScenario</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>R9: Gran.Con.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>R12: Scalable</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>R14: Diversity</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>R15: Standard</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
METHODA Design

Overview

Orchestrated from the management host
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