CORENEXT

D4.3

Trustworthy Computation and Orchestration

Funded by
the European Union

Revision v1.0

Work package	WP4
Task	T4.3, T4.4
Dissemination level	PU – Public, fully open. e.g., website
Deliverable type	R – Document, report (excluding periodic and final reports)
Due date	30-09-2024
Submission date	30-08-2024
Deliverable lead	Barkhausen Institut (BI)
Version	v1.0
Authors	Mohand Achouche (III-V), Nils Asmussen (BI), Romain Beurdouche (EUR), Andreas Georgakopoulos (WINGS), Anastasia Grebenyuk (EAB), Julien Lallet (NNF), Michael Roitzsch (BI)
Contributors	Task partners (see below)
Reviewers	Andreas Georgakopoulos (WINGS), Markus Ulbricht (IHP)

Abstract

Upcoming 6G mobile communication networks will process sensitive data and will be a driver for industrial automation, critical infrastructure, as well as personal robotics. Therefore, 6G networks must offer fundamental trustworthiness from terminal to base station to edge cloud. This trustworthiness must be an afterthought but must be built into fundamental components by design. In this deliverable, we address such trustworthiness foundations for key compute and orchestration building blocks. To create a heterogeneous compute architecture usable for 6G signal processing, the costs of added trustworthiness features must be balanced against cost factors such as added latency or added hardware costs. We will survey these trade-offs to prepare their evaluation in later deliverables by work package 6.

Keywords

FPGA, DSP, heterogeneity, virtualization, isolation, attestation

Funded by
the European Union

Document Revision History

Version	Date	Description of change	Contributor(s)
v0.1	19-04-2024	initial table of contents and outline	Michael Roitzsch (BI)
v0.2	24-06-2024	first complete version	Mohand Achouche (III-V), Nils Asmussen (BI), Romain Beurdouche (EUR), Andreas Georgakopoulos (WINGS), Anastasia Grebenyuk (EAB), Julien Lallet (NNF), Michael Roitzsch (BI)
v1.0	10-07-2024	changes after internal review	Michael Roitzsch (BI)

Contributing Partners

Abbreviation	Company name
BI	BARKHAUSEN INSTITUT
EAB	ERICSSON
CYB	CYBERUS TECHNOLOGY
EUR	EURECOM
WINGS	WINGS ICT SOLUTIONS
ETHZ	EIDGENOESSISCHE TECHNISCHE HOCHSCHULE ZUERICH
IHP	IHP MICROELECTRONICS
NNF	NOKIA NETWORKS FRANCE
IIIV	NNF/IIIV LABS
IFAT	INFINEON TECHNOLOGIES
KAL	KALRAY
EF	ERICSSON FRANCE

Disclaimer

Copyright Notice

Executive Summary

This deliverable explains how COREnext advances the state-of-the-art in trustworthiness for key components of a 6G network infrastructure. Specifically, we report on the prototype status of the following components:

- FPGA multi-tenancy,
- Digital signal processor virtualization,
- M^3 microkernel-based system,
- IoT management, and
- Radio link authentication.

For each component described, we include details on partner collaborations as well as performance indicators to be evaluated.

While this deliverable reports on trustworthiness advances, the sibling deliverables D4.2 and D5.1 cover acceleration to improve efficiency and the analogue part of radio link authentication respectively. Taken together, the components described in these three deliverables will be fitted into the overall COREnext architecture in D3.2. The combination delivers significant improvements to the trustworthiness and efficiency of future mobile networks.

Table of Contents

List of Figures

List of Tables

Funded by the European Union

Acronyms and Definitions

CP	Cloud Provider
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
IoT	Internet-of-Things
IP	Intellectual Property
KPI	Key Performance Indicator
MAC	Media Access Control
ML	Machine Learning
OAuth	Open Authorization
OFDM	Orthogonal Frequency-Division Multiplexing
OPS	Operations per Second
OSR	Oversampling Ratio
PA	Power Amplifier
PBO	Power Back-Off
PHY	Physical Layer
QAM	Quadrature Amplitude Modulation
RAN	Radio Access Network
TA	Trusted Authority
TCU	Trusted Communication Unit
TEE	Trusted Execution Environment

1 Introduction

Trustworthiness is a fundamental need for future mobile communication networks. Already today, these networks process data from diverse clients simultaneously and must withstand cyberattacks to offer high service availability. With upcoming use cases such as autonomous driving or networked critical infrastructure, this requirement becomes ever more relevant. At the same time, the growing data volumes require novel acceleration in the signal processing path, thus increasing the complexity of the network. In such an environment, increasing trustworthiness with strong hardware-based isolation measures is a key technique to reduce system complexity. Isolation helps to separate functional concerns in the network and to protect confidentiality and integrity of data flows through the network. We discuss our fundamental design principles in Section 2, before we discuss component advancements in Section 3.

Input to this deliverable is provided by D3.1, which converged to four fundamental component advancements, structured along the dimensions of trustworthiness/efficiency as well as digital/analogue. In this deliverable, we present our component status for the digital trustworthiness quadrant (see Table 1).

	Digital	Analogue
Efficiency	Power-efficient signal processing	Power-efficient high-throughput interconnect
Trustworthiness	Heterogeneous compute platform with TEEs	Radio link authentication and infrastructure attestation

Table 1: COREnext component advancements

The accompanying sibling deliverables D4.2 and D5.1 cover the digital efficiency and analogue trustworthiness parts respectively (see Figure 1). In Section 4, we summarize the component costs and benefits to distill performance indicators that can be measured for evaluation. Together with similar results from D4.2 and D5.1, the component advancements and their performance indicators become input to deliverable D3.2, where their fit for the overall COREnext architecture is discussed. Ultimately, validation of component systems will happen in work package 6.

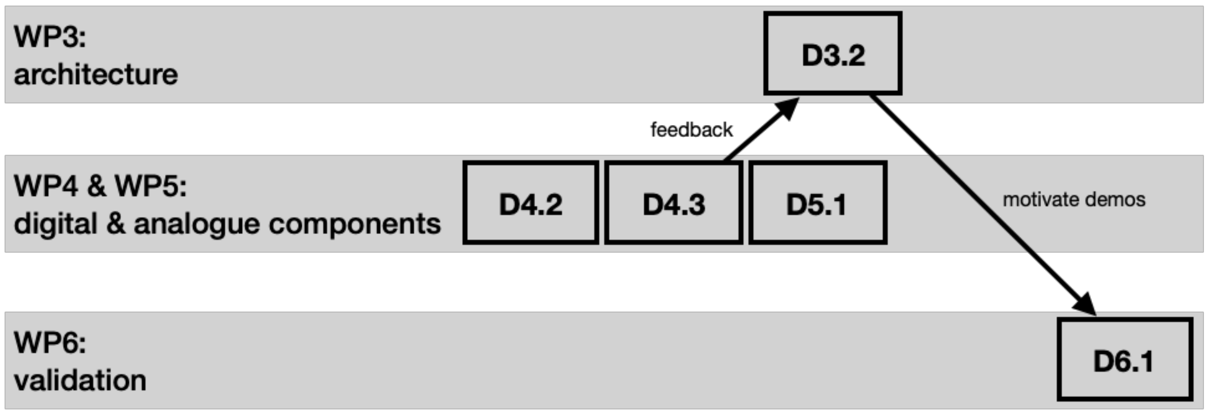

Figure 1: Overview of deliverables connected to D4.3 and flow between them

2 Trustworthy Orchestration of Heterogeneous Compute Resources

When providing compute resources for 6G workloads, accelerators are key building blocks to improve the energy efficiency of the system. However, running computations on a mix of general-purpose cores and accelerators requires data movement between these heterogeneous compute resources. While deliverable D4.2 explains the accelerator advancements themselves in more detail, here we discuss how to **orchestrate** these accelerators to form a trustworthy fabric of compute resources.

A key technique for trustworthiness is **isolation.** When processing data from different clients or running applications from different tenants, the computations must be strictly isolated from each other. This means they should be prevented by fundamental mechanisms from reading or manipulating each other's data and code, because read-access would constitute a breach of confidentiality and write-access would constitute a breach of integrity. As the isolation mechanism itself must be robust against attacks, it must be implemented at an architectural level below the user data and applications. Operating systems are one typical layer that provides isolation for software. But isolation against hardware-based attacks must itself be located in hardware.

However, a system where everything is isolated from everything else cannot perform meaningful work. Therefore, controlled breakage of the isolation is necessary so that components can cooperate to process a workload together. However, such breakage must be governed by an **access control mechanism.** An example mechanism are capabilities, where a component owns access rights to other components by holding a capability: Comparable to an entry ticket in real life, a capability is an unforgeable token combining a reference to the target component with associated access permissions.

With the question of access control comes the question of **resource allocation.** Systems require policy-making authorities that maintain an overview of the set of resources available and make decisions which workload to run on which hardware resources. Consequently, these policy components also disseminate corresponding access rights so that the cores and accelerators selected to run a given workload can cooperate. At the same time, the policy component can enforce high-level platform security properties, such as workloads sharing certain hardware or not.

Resource allocation is not necessarily a binary decision. Depending on the type of compute hardware, it may be possible to partition the hardware and assign shares of it to different clients. Partitioning may be possible in space (like assigning a portion of a processing array to one workload) or in time (like running one workload for a given time slice and then reassigning the resource to another workload). **Virtualisation** is when the partitioning is implemented without requiring application cooperation. Instead, applications are oblivious to virtualisation and instead have the illusion of exclusive access to resources.

With multiple applications sharing common resources, it is important to implement **budgeting.** Applications should be restricted from overstepping their allocated share and should thus be prevented from withholding resources from their co-running neighbours.

Given virtualized resources, applications become detached from the underlying hardware. This decoupling bears the danger of applications falling victim to manipulated infrastructure that does claims to provide properties such as secure data isolation but does not actually implement the necessary protections. **Attestation** is a cryptographic mechanism, by which applications can check hardware and software resources for their authenticity. Applications can decide, to use specific accelerators or remotely hosted compute resources only after a successful attestation check.

3 Orchestration Component Prototypes

Within work package 4, the prior deliverable D4.1 has structured the development of digital components as illustrated in Figure 2. Here, we cover the lower half of this component progression.

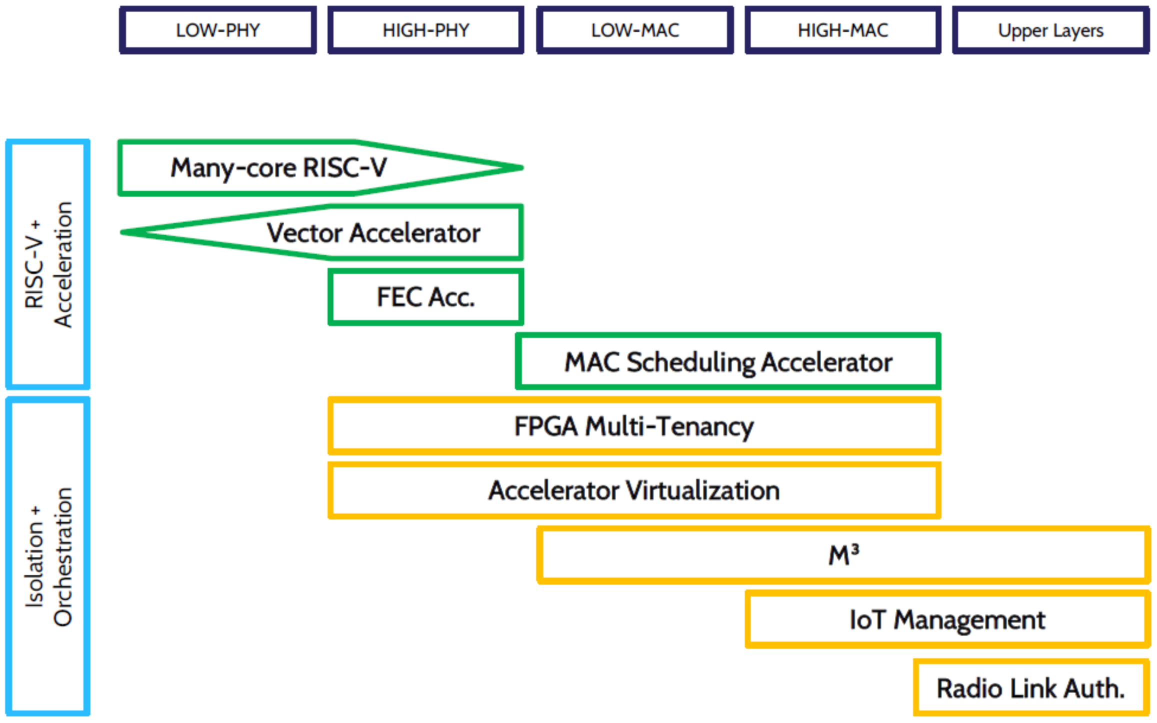

Figure 2: Overview of digital components.

Table 2 shows, which component utilizes which techniques of the trustworthy orchestration toolbox outlined in the previous section. In the following, we explain for each component the development and prototype status, interactions between partners, and planned performance measurements.

Component	Isolation	Access Control	Resource Allocation	Virtualisation	Budgeting	Attestation
FPGA multi-tenancy	✓	✓	✓			
Digital signal processor virtualization	✓	✓	✓	✓		
M³ microkernel-based system	✓	✓	✓	✓	✓	✓
IoT management		✓	✓			
Ratio link authentication		✓				✓

Table 2: Trustworthy orchestration techniques by component

Funded by the European Union

3.1 FPGA Multi-Tenancy

FPGAs in the cloud (Amazon, Microsoft, Alibaba etc.) are big systems available to accelerate specific processing. These systems are oversized to allow the implementation of almost any kind of algorithm. The cost of FPGA renting is slightly higher than other cloud resources, and most of the time, FPGA resources may be shared to increase their use efficiency. Unfortunately, this is nowadays not offered by cloud providers most probably for security reasons. In this work package, we aim to provide a secure way of sharing FPGA resources among multiple users, and this is called multi-tenancy.

Cloud security is critical for a client when choosing a commercial Cloud Provider (CP). Commercial cloud users expect secure remote computation and access to FPGA accelerators with minimal impact on their design performance. Security mechanisms need to be adapted for appropriate cloud usage. First, the client needs to ensure that its data is kept private. The client does not want to disclose sensitive Intellectual Property (IP) and data to the cloud provider. To ensure that, the client needs an encrypted channel with the FPGA isolated from the CP. Authentication is another important security aspect to establish a secure remote connection between a client and the hardware acceleration material. The client needs to ensure that the correct FPGA is used and that no other users may access the allocated resources. Authentication is necessary to manage FPGAs and different cloud service accesses to mitigate client impersonations and data breaches. CP security methods are not transparent concerning data encryption methods, bitstream protection and IP theft. To remove this drawback, it is necessary to use methods and protocols which respect user privacy and intellectual property. To reinforce security, as presented in Figure 3, an intermediate authority between the client and the CP must be introduced. The entity, called Trusted Authority (TA) instance, enables isolation between end-users and CP. Often, the TA would implicitly be the chip manufacturer for practical and security reasons. It can safely implement security mechanisms (e.g. secret keys, Physical Unclonable Functions). From a client's perspective, the TA achieves device authentication and isolation from the CP. From the CP's perspective, the TA implements tasks like FPGA access management and authentication.

Access control is an important security tool to enforce user authorization rules set by the access manager. The user must be authenticated and identified to obtain authorization from CP. Additionally, user identification is also necessary to enforce access control rules. To the best of our knowledge, any CP proposes multi-tenant FPGA usage. Regarding OpEx and CapEx, multi-tenant is highly attractive. Security limitations exist from literature proposals. Our objective is to enhance existing solutions and propose a new security scheme for multi-tenant FPGA cloud solutions. The security framework must support: FPGA and user authentication, multi-tenancy, scalability, user-CP isolation, access control, and access sharing.

We propose TokSek, a token-based multi-tenant FPGA cloud security framework. To the best of our knowledge, it is the first framework to introduce the concept of access tokens to the FPGA cloud architecture. TokSek is an adaptation of OAuth 2 to share access to FPGA devices instead of credentials or applications. TokSek enables a scenario, where an FPGA deployed in the cloud can be accessed by multiple tenants. As illustrated in Figure 3, the TA enables isolation between CP and end-user. TA is responsible for FPGA access creation and security enforcement. The CP manages FPGA resources and co-operates with TA. The TokSek principle is presented in Figure 4 and Figure

5. The protocol is detailed in Figure 6. The implementation is in progress, targeting a Xilinx FPGA. Our objective is to evaluate the security framework performance.

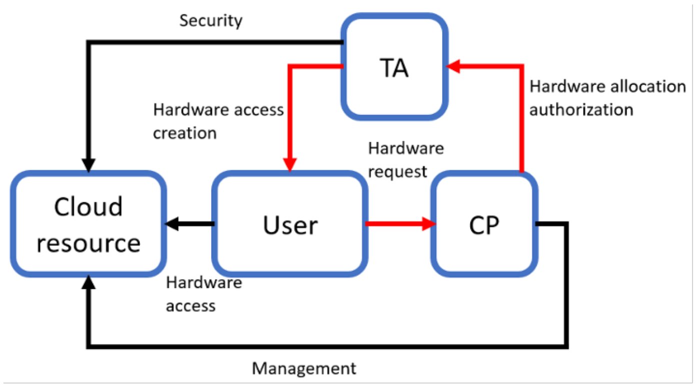

Figure 3: Multi-tenant FPGA cloud scenario

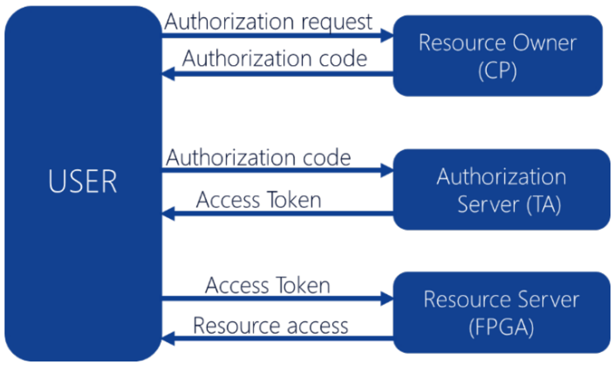

Figure 4: TokSek adaptation from OAuth2.0-based protocol

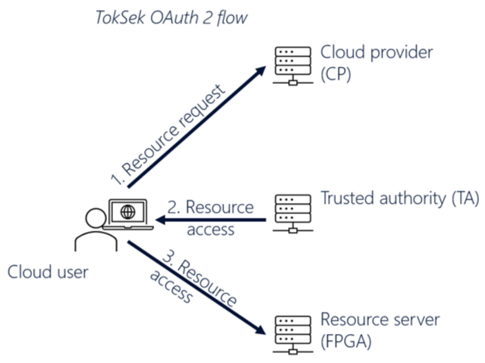

Figure 5: TokSek adaptation from OAuth2.0-based protocol

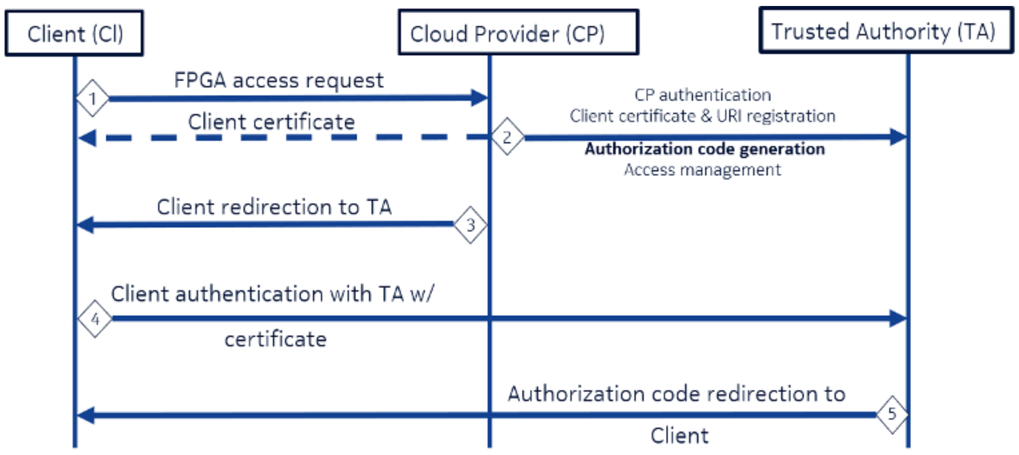

Figure 6: TokSek protocol scheme

3.1.1 Prototype Status

The prototype development is ongoing. First, we validate some concepts on an embedded platform with some functions done in software. TokSek implementation is in progress in Xilinx FPGA target. The objective is to evaluate the performances of the proposed security scheme. In parallel, we also started to implement some hardware functions of TokSek on the Carfield platform developed by ETH Zürich.

3.1.2 Interactions with Partners

Due to the interest of NNF and III-V Lab in RISC-V architecture, some collaborations were started in Q3/2023 with ETH Zürich. In this context, we have ported the Carfield platform from ETH to the Xilinx VCU118 board used at NNF.

3.1.3 Planned Performance Measurements

Following the TokSek prototype building, a validation phase will follow considering some use-cases scenarios. This step will be key to further develop cooperation with other project partners.

3.2 Digital Signal Processor Virtualization

Modern RAN implementations, including some mostly relying on software, rely on a variety of Digital Signal Processors (DSPs) within compute nodes. While the infrastructures are turning to the cloud and thus to virtualization, DSPs remain difficult to use for workload containers, sharing this resource. Since a DSP does not share the memory and tasks of its host device, a proper and efficient sharing and isolation cannot be ensured by the host system hypervisor using the usual resource model. An adapted way for the hypervisor to use a DSP and an abstraction for DSPs within containers have to be enabled.

3.2.1 Prototype Status

This contribution aims to embed in a container a mobile network software relying on a DSP for parity check coding. The software is currently being improved on bare metal to achieve the best efficiency possible in the use of the DSP. The mobile network software is already able to be embedded in an application container – with Docker or OpenShift -. The next steps will be to figure out how to use the DSP within a container and how to share the DSP between multiple containers.

3.2.2 Interactions with Partners

This contribution is done by EUR. A collaboration with the FPGA Multi-Tenancy contribution will occur. This other contribution is also about sharing resources with different tasks and memory than the host system. This collaboration is necessary to solve the complex problem of sharing this kind of resources.

3.2.3 Planned Performance Measurements

This contribution will have a particular interest in the workload density criterion which is the number of workload instances that can be fitted on a single compute node with one DSP while warrantying a given quality of service. In practice, it is the number of software instances with a given configuration that can be fitted on the compute node which will be measured and optimised.

3.3 M^3 Microkernel-Based System

The use cases considered in COREnext such as automotive infrastructure or smart city require trustworthy devices on both the terminal and base-station side. In particular, both devices need to be able to integrate third-party components, which are not necessarily trustworthy and therefore have to be properly isolated from the rest of the system. Furthermore, the support of heterogeneous compute units is required to deliver the expected performance while being as energy efficient as possible. To achieve these goals, we use the M^3 hardware/software platform.

M^3 proposes a new system architecture based on a hardware/software co-design. On the hardware side, M^3 builds upon a tiled architecture, as shown in Figure 7. M^3 extends its tiles by adding a new hardware component called trusted communication unit (TCU) to them. Each tile contains a TCU and either a core, an accelerator, or memory (e.g., a memory interface to off-chip DRAM) and the tiles are connected via a network-on-chip. As the TCU is the only way to access tile-external resources, the TCU controls the tile's access permissions. By default, all tiles are isolated from each other. To perform message-passing between tiles or access memory, a corresponding communication channel (thick black lines in the figure) needs to be established. These communication channels are represented as endpoints in the TCU (orange dots).

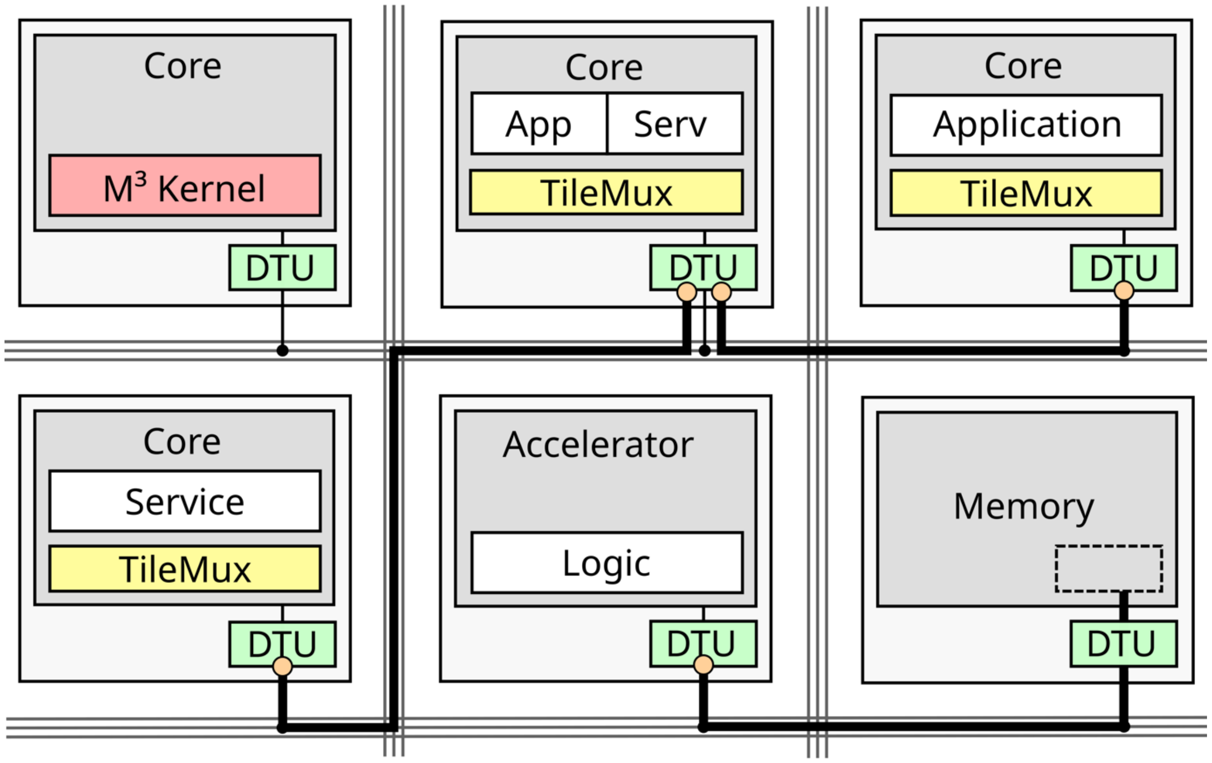

Figure 7: System architecture of M³

On the software side, M³ runs a microkernel (red) on a dedicated kernel tile, and applications and OS services on the remaining user tiles. Applications and OS services on user tiles are represented as activities, comparable to processes. An activity on a general-purpose tile executes code, whereas an activity on an accelerator tile uses the accelerator's logic. Activities can use existing communication channels, but only the M³ kernel is allowed to establish such channels. Applications are placed on different tiles by default, but as shown by the M³v work, tiles with general-purpose cores can also be shared efficiently and securely among multiple applications. For that reason, every core-based user tile runs a multiplexer called TileMux (yellow), which is responsible for isolating and scheduling the applications on its own tile, like a traditional microkernel. However, in contrast to a kernel, each TileMux instance has no permissions beyond its own tile. Instead, only the M³ microkernel can make system-wide decisions, hence its name.

In summary, M³ is specifically designed for heterogeneous systems and can integrate untrusted hardware components such as accelerators or complex processors without risking a compromise of the rest of the system. This is because processors and accelerators isolated by default via the TCU and need to receive permission to access other components of the system by the M³ kernel.

3.3.1 Prototype Status

The basic hardware/software platform is working and usable. However, we are working on multiple fronts to further improve its security properties and efficiency.

At first, we are working on the improvement of real-time guarantees that the M³ platform can provide. Real-time guarantees are important for cyber-physical systems to, for example, ensure that no harm is done to the physical world. One key advantage of the M³ platform we identified is

Funded by the European Union

the ability of tile-local reasoning. That is, the real-time analysis can concentrate on a single tile instead of the whole system. Furthermore, the TCU offers faster cross-tile communication than existing systems, which also benefits real-time workloads. This work has been finished and was published on the 30th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS).

Second, we work on a general data-streaming framework that is suitable for base band computations on M^3. The goal is to distribute the computation over multiple and potentially heterogeneous tiles and pipe the mobile communication traffic through this pipeline. The current state is that the general framework is working and in the next step we will use it to run baseband computations.

Finally, we strive to provide trusted execution environments (TEEs) on M^3. Although M^3 already provides strong isolation between different hardware and software components, currently all TCUs and the software infrastructure to load applications needs to be trusted. Our goal is therefore to enable the execution of applications in a TEE where after the initial loading the applications state is frozen and validated. If the validation succeeds, no further changes are possible, so that we no longer have to trust the loading infrastructure. We are currently still in the design phase and will soon start with the implementation.

3.3.2 Interactions with Partners

The work on the M^3 platform and the TEE extension is performed by BI. We are working together with TUD on the baseband computations and will use real workloads and data from EUR for the evaluation. Furthermore, we are collaborating with EAB to perform machine learning for the hardware fingerprinting on M^3.

3.3.3 Planned Performance Measurements

We plan to perform latency measurements for both the TEE extension and the baseband computations. Additionally, we will measure the costs of the security advantages that the M^3 platform provides in terms of silicon overhead or gate counts and measure the size of the trusted software in terms of lines of code.

3.4 IoT Management

As mentioned also in D4.1, the IoT has revolutionized the way we interact with technology, transforming everyday objects into smart devices interconnected through the internet. IoT management and devices are now pervasive in our homes, workplaces, and public spaces, providing us with convenience and efficiency. As the IoT landscape continues to expand, so does the importance of ensuring the trustworthiness of these devices (e.g., are they reliable; secure; ensure privacy of IoT devices and the data they collect and transmit). With these devices becoming increasingly integrated into critical systems and handling sensitive data, their trustworthiness has a direct impact on user safety, data security, and overall system integrity.

As a result, a Trust Manager is proposed and implemented. Edge operational devices, such as drones and collaborative robots (cobots), communicate with the Trust Manager Orchestrator to

Funded by the European Union

ensure efficient and secure operations. These devices generate a plethora of data related to network Key Performance Indicators (KPIs) and device-specific metrics. This data is essential for understanding the operational status and capabilities of each device within the network. Machine Learning techniques, particularly K-means clustering, are employed to analyze this data. By doing so, devices with similar characteristics are grouped into clusters, facilitating more streamlined and effective management.

Once the devices are grouped into clusters, each cluster is subjected to a detailed evaluation using various trust metrics. These metrics are tailored to assess the trustworthiness of each device within the group. The Trust Evaluation Function plays a critical role in this process. It calculates a trust index for each device, providing a quantifiable measure of trustworthiness. This index is then communicated back to the Trust Manager Orchestrator, ensuring that the orchestrator has up-to-date information on the trust levels of all devices in the network.

The use of computational offloading is an important strategy in this system. Depending on the specifications of each cluster—such as their compute resources, memory, and network capabilities—tasks can be offloaded to different devices to optimize performance. This approach not only enhances the efficiency of the network but also ensures that computational tasks are handled by the most capable devices. By leveraging the strengths of each cluster, the system can maintain high levels of performance and reliability.

Finally, the Trust Manager Orchestrator uses the trust index and the specific prerequisites of each task to determine the optimal input node for the task at hand. This decision-making process is crucial for ensuring that tasks are assigned to the most trustworthy and capable devices, thereby enhancing the overall security and efficiency of the network. By continuously evaluating and leveraging trust metrics, the orchestrator can dynamically adapt to changing conditions and maintain optimal performance across the network.

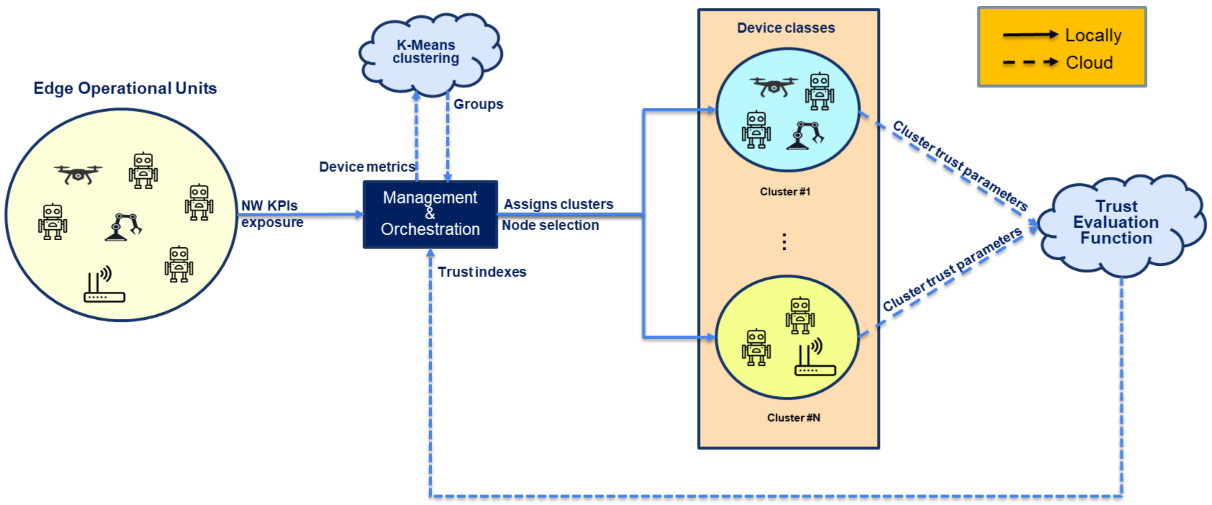

Figure 8: Trust Manager Flowchart

3.4.1 Prototype Status

A preliminary version of the Trust Manager is currently under evaluation. It should be noted that the Trust Manager is composed of various components which are responsible for the clustering, the management and orchestration, classification in device classes, and trust evaluation function. All these components interact with each other to calculate the trust index.

3.4.2 Interactions with Partners

The key partner of this work is WINGS. This work can be combined with the platforms provided by other partners in the project to provide a trust index which can become part of these platforms and evaluate the connected devices or the connected hardware components to them.

3.4.3 Planned Performance Measurements

One of the planned performance metrics that will be taken into consideration is the time for execution for providing the trust index. Other performance metrics are also under investigation.

3.5 Radio Link Authentication

This section explores the use of machine learning (ML) technique for RF fingerprinting. The technique has recently emerged as a promising technique for Physical Layer Security for 5G and beyond. The basic premise of RF fingerprinting is that each transmitting device has minor manufacturing imperfections and operation impairments that result in unique, subtle characteristics or discrepancies in the radio signals it emits. These discrepancies, although often very limited, can be detected, measured, and processed allowing to create a 'fingerprint' of the device. The hardware impairments can manifest in imperfections such as quadrature imbalance, phase noise, frequency jitter, power amplifier (PA) in-band distortion, intermodulation distortion and reference spurs.

The task 4.3 within WP4 aims to develop acceleration solution(s) based on the algorithm-hardware co-design for RF fingerprinting to establish the trustworthiness of a device identity before authorizing any data exchange over a radio link.

3.5.1 Prototype Status

Data description

The data for RF fingerprinting is generated using MATLAB scripts. An Orthogonal Frequency-Division Multiplexing (OFDM) input signal is created using MATLAB's waveform generator application. This involves generating random input bits with amplitudes normalized between 0 and 1. These bits are then modulated using Quadrature Amplitude Modulation (QAM) schemes ranging from 4-QAM to 256-QAM to evaluate the effect of modulation order on RF Fingerprint identification methods.

For modeling non-linear behavior, four different polynomial power amplifier (PA) models are utilized. The model's coefficients are extracted from real PA measurements and post-layout simulations and are characterized by their non-linear responses which are independent of frequency.

Data is generated at various output power back-off (PBO) levels: -9 dB, -3 dB, and 0 dB. These PBO values are chosen to evaluate their impact on the non-linearity of the output signal, which is crucial for enhancing the discriminating power of RF fingerprinting across conditions ranging from moderate to high saturation in the amplifiers. Data is also generated for different Oversampling Ratio (OSR).

For ML purpose separated data for training and testing were produced, corresponding to the output generated by the waveform generator in which each amplitude of I and Q is random.

The ML framework has been built to perform the classification between four PAs models. To feed the IQ data to the ML algorithm, the In-phase (I) and Quadrature (Q) components are stacked to form a 1-dimensional (1D) dataset with 2 channels. Sliding window technique divides a continuous stream of IQ samples into smaller, fixed-size segments (windows) for processing.

We explored different values of the windows, IQ trace length, as well as stride on structuring input data for model training and testing. A small stride, such as 1, leads to significant overlap between sliding windows, enhancing learning of data dependencies but increasing computational load and risk of overfitting. In contrast, a larger stride creates windows with little to no overlap, reducing redundancy and computational needs but possibly missing finer data details. To enhance the ML performance different values for the IQ trace length (L) and stride (s) were tested, including L=144, 288, 576 and s=1, 4, 8, 16, 32, 64, 128, L as well as random values.

The ML architecture includes 1D convolutional layers and a total of about 4.7K trainable parameters, making it a lightweight model. Accuracy is the performance metric used to evaluate the model ability to classify fingerprints. Accuracy is the proportion of sample correctly classified. An accuracy near 0.25 mean a random guess since the classification has 4 classes/PAs. On the other hand, when ML model recognize each fingerprint, the accuracy is close to 1. At the beginning of model training (small number of epoch), the accuracy is poor, and start to increase as the model learn (with respect to epoch). Figure 1 shows the ML accuracy vs epochs for L=576 and stride of 1 (left), 16(middle) and 288 (right). With a stride of 1, the model overfits the training data, resulting in poor accuracy during testing. With a higher stride, the model generalizes better on the testing data, achieving an accuracy of about 0.78.

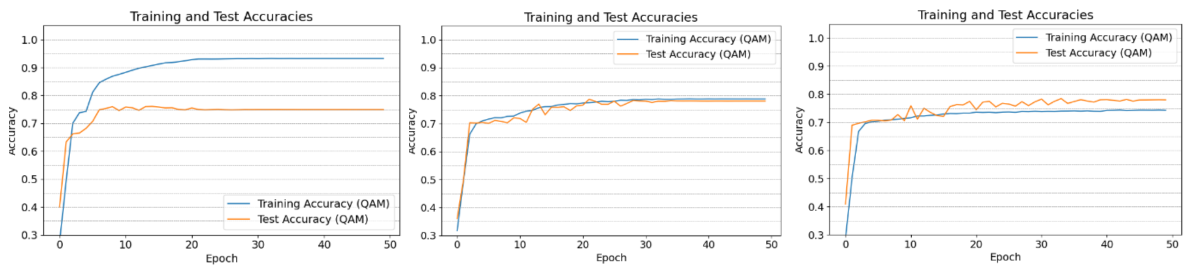

Figure 9: ML accuracy vs epochs for L=576 and stride of 1 (left), 16 (middle) and 288 (right).

We have also tested different algorithms, including ResNet, VGG, and attentional mechanisms. In general, we do not see a significant increase of an accuracy, suggesting that the current data has a discriminative power limit, with an achieved accuracy.

To understand the effect of signal modulation on ML performance, we trained and tested algorithms on various QAM orders. Figure 2 shows training with all QAM orders, from 4 to 256, and testing with 4-QAM (left), 32-QAM (middle), or 128-QAM (right). We also tested scenarios with higher-order QAM training and lower-order QAM testing. Results indicate modulation order impacts classification accuracy, highlighting the need for robust methods. We explored a mixture-of-experts model, where each expert specializes in specific data, and a multi-tasking approach treating modulation as a separate task. Initial results show minimal improvement in generalization when training and testing QAM orders differ. Further studies will include PSK modulation data.

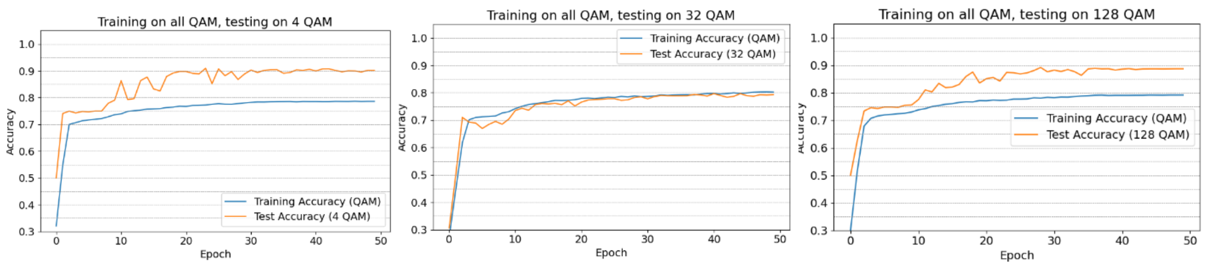

Figure 10: ML accuracy vs. epochs for training with all QAM orders and testing with 4-QAM (left), 32-QAM (middle), and 128-QAM (right) .

Additional studies have been also performed with the different power back-off (PBO) values. As expected, we observed high ML accuracy and better convergence at 0 dB, where the amplifier operates at full power, pushing it into the non-linear regime. In contrast, the -9 dB setting, which correlates with linear operation, shows lower accuracy that does not improve with additional training epochs. For the different oversampling ratio (OSR), we observed that OSR of 8 gives worse ML performance compared to OSR of 1.

3.5.2 Interactions with Partners

We are exploring how hardware can complement the ML-based RF Fingerprinting task through an algorithm-hardware co-design approach, involving our partners BI and IHP. BI employs the M3 platform, a scalable and efficient operating system designed for hardware/software co-design with 64-bit RISC-V cores. We need to address memory requirements for inference, including storage for model parameters, input data, and intermediate outputs.

IHP is leveraging the NVDLA architecture as a reference for their work, which promises scalable and parametrizable hardware solutions. The collaboration focuses on integrating advanced hardware capabilities with our ML models to enhance performance and efficiency in RF fingerprinting tasks.

3.5.3 Planned Performance Measurements

ML performance can be quantified in terms of model classification accuracy, false positive rate, and inference latency. Classification accuracy indicates the correct prediction rate, meaning the ratio between correctly classified items and all encountered ones; higher accuracy signifies better model capability in identifying different RF fingerprints. Another relevant metric is the false positive rate, which measures the number of devices wrongly classified out of the total number of actual impairments. Lower false positive rates mean fewer incorrectly labelled fingerprints, indicating the model has effectively learned to identify fingerprints.

Moreover, running ML algorithms requires specific computational resources and hardware. To choose the best hardware that suits the computational requirements of the ML model, a memory usage analysis will be performed. This analysis considers the inference latency and CPU/GPU utilization to measure the relevance and efficiency of the solution. We need the fastest possible inference without sacrificing accuracy to ensure the model's performance and usability.

KPIs emphasized in our collaborative efforts include peak performance (measured in operations per second, OPS) and energy efficiency (measured in J/OPS or OPS/W). Additionally, KPIs that combine hardware and application metrics, such as inferences per second for throughput or J/inference for energy efficiency, are important. These KPIs will be correlated with application specifics like accuracy and memory footprint, ensuring the hardware-accelerated ML models meet the required performance and efficiency standards in our collaborative research efforts.

4 Conclusion

The components for trustworthy orchestration we presented provide foundational layers within an overall trustworthy system. As such, these components can host other components and orchestrate them in a trustworthy way. Therefore, in Table 3, we show the interactions between the components presented here as well as with accelerator components delivering on efficiency targets described in deliverable D4.2. Component interactions leading to concrete collaborations are shown by listing the respective partners. Checkmarks highlight potential component interactions that are not (or not yet) explored within COREnext.

Orchestration Component	RISC-V Acceleration	Dedicated Acceleration	DSP Virtualization	Radio Link Authentication
FPGA multi-tenancy	NNF, IIIV, ETH		NNF, IIIV, EUR	
DSP virtualization		✓		
M³ microkernel-based system	BI, TUD, EUR	✓	✓	EAB, BI
IoT Management	✓			✓
Radio link authentication	✓	EAB, IHP		

Table 3: Component interactions

For each orchestration component, we have reported on these collaborations as well as the prototype status and targeted performance measurements. Together with sibling deliverables D4.2 and D5.1, the component status is reported to work package 3, where deliverable D3.2 will finalize the COREnext architecture and fit these components into the bigger picture, deriving integrated validation measurements from the individual component performance measurements presented here (see Table 4).

Orchestration Component	Cost Measure	Benefit Measure
FPGA multi-tenancy	FPGA area, latency added by authentication	Increase in FPGA utilization with secure authentication
DSP virtualization	Quality of service: latency added by virtualization	Number of workload instances sharing a DSP
M³ microkernel-based system	Latency and chip area added by TCU isolation component	Small size of trusted components
IoT Management	Execution time to provide trust index	Efficacy of trust classification for IoT devices
Radio link authentication	Latency and energy usage of classification model	Model classification accuracy and false positive rate

Table 4: Per-component cost and benefit measurements

Funded by the European Union