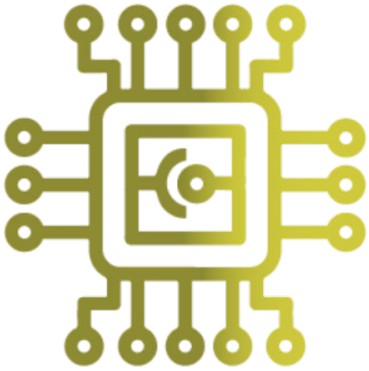


CONVOLVE

Seamless design of smart edge processors

GRANT AGREEMENT NUMBER: 101070374

Deliverable D3.4

Component-Based Security Evaluation and Assessment

Title of the deliverable	Component-Based Security Evaluation and Assessment
WP contributing to the deliverable	WP 3
Task contributing to the deliverable	Task 3.4
Dissemination level	PU – Public
Due submission date	31/07/2025
Actual submission date	31/07/2025
Author(s)	Mottaqiallah Taouil (CIC), Fouwad Mir (CIC), Sven Argo (RUB), Jan Richter-Brockmann (RUB), Tim Güneysu (RUB), Christian Larmann (TUD), Alejandro Garza (NXP), Adrian Marotzke (NXP), Jeffrey Smith (TUE), Marc Geilen (TUE), Henk Corporaal (TUE), Manil Dev Gomony (TUE).
Internal reviewers	Andre Guntoro, Tobias Grosser (GNA)

Document Version	Date	Change
V0.1	09/05/2025	Initial Document
V0.2	12/05/2025	Updated Layout
V0.3	10/06/2025	Content improvement in WP3 meeting
V0.4	07/07/2025	Finalizing contributions from WP partners
V0.5	11/07/2025	Final adjustments for internal review
V0.6	28/07/2025	Addressed feedback comments
V1.0	31/07/2025	Submission to the EU portal

Table of Contents

Executive Summary

This document presents the security evaluation and assessment of the CONVOLVE hardware security framework prepared for the Working Package 3 (WP3) project. This framework has a wide spectrum of security measures encompassing trusted execution of cryptographic cores within the SoC, generation and evaluation of PQC hardware accelerators, and physical side channel security of proprietary neural network accelerators against IP theft.
Concretely, the contributions are as follows:

- RUB: As the principal developer of the HADES tool (c.f. "D3.2 Security Architecture and Trusted Execution Environment (TEE) Implementation" and "D3.3 Update of Security Architecture and TEE Implementation"), RUB ensures the correctness, performance, and security of hardware accelerators generated with HADES. This includes comparisons with reference implementations for functional correctness and performance characteristics. Further, we conducted a practical *test vector leakage assessment* (TVLA) in our lab to complement the theoretical security analyses. All implementations are functionally correct, adhere strictly to standardized specifications, and demonstrate strong resistance to first-order side-channel attacks, with empirical TVLA results, confirming no detectable leakage up to 100 million traces under realistic conditions and validating the test setup through second-order leakage detection. Together with NXP, we investigated the best algorithms that could be used for secure boot in the TEE and signatures.
- NXP: NXP has further developed and optimized a post-quantum-enabled TEE, featuring hardware acceleration, Sphincs+ hash-based signatures for maximum security, as well as memory and code size optimizations. We integrated both pre-quantum (Ed25519) and post-quantum (Dilithium, Sphincs+) signature schemes, ensuring resilience against future threats. Hardware acceleration via Keccak dramatically boosts performance up to 343x for Sphincs+, compared to software implementations, while memory and boot-ROM optimizations significantly reduce resource demands, enabling secure and efficient operation even in constrained environments. Additionally, together with CIC we introduced TeeRTOS that achieves secure task isolation and dynamic memory protection with only 4% task switch overhead using the PMP Snapshot Unit, while supporting advanced features not present in other secure FreeRTOS variants.
- CIC: CIC has explored the power side-channel leakage assessment of the digital computation-in-memory (CIM) accelerators being developed together with TU Delft in WP2. This involves the conventional adder-tree-based accelerators and a novel, adder-tree-free design (Dream CIM) that replaces adder-trees with a multi-sub array and pipelined periphery to achieve area and energy optimization. Considering the observations made during the assessment, we proposed lightweight countermeasures and evaluated them using security metrics such as correlation power analysis (CPA), TVLA, and signal-to-noise ratio (SNR). Both designs showed no observable leakage in TVLA and CPA, even with up to 1 million traces, validating the effectiveness of the implemented countermeasures and the robustness of the secure CIM architectures. Furthermore, our countermeasures achieved substantial efficiency gains: the adder-free design delivered a 12.7x reduction in area and 13.3x reduction in energy, while the adder-tree-based design achieved 3x area and 3.65x energy savings, compared to masking schemes.
- TUE: We assessed methods to merge TEEs with real-time composable platforms to design a secure, predictable, and composable platform for low-power SoCs. Simply

encapsulating a composable framework in TEEs is not possible, as this affects the predictable execution of the applications. We investigate securing virtualised hardware within the composable framework as an alternative. As the real-time composable platforms also isolate virtualised hardware environments, we develop and test a composable security framework to provide an analogue to a TEE for real-time composable platforms. TUE and NXP collaborated in analysing the feasibility of protecting the composable framework with TEE and found that the two frameworks were not compatible. We then decided upon the design of securing a composable framework with security features that provided a similar solution to that of the TEE using standard security tools within the composable design.

1 Introduction

This document, "Component-based Security Evaluation and Results," is a deliverable of Working Package No. 3, "Composable Real-Time and Hardware Security", task T3.4, "Security Evaluation and Assessment under the task lead of CIC.

The CONVOLVE project aims to create secure, ultra-low-power processing hardware for edge devices. The project involved a list of deliverables at every stage, where this deliverable marks the final stage of the CONVOLVE project. This document presents the comprehensive security evaluation results for all components proposed and implemented in preceding deliverables.

1.1 WP3 Objectives

WP3 has three objectives:

- O3.1 Detection and prevention of physical/hardware attacks, including side-channel analysis and fault injection attacks, through ultra-low power protection mechanisms.
- O3.2 Design of ultra-low power (ULP), real-time, modular, and composable, long-term quantum-secure TEE for RISC-V processor architectures.
- O3.3 Extend TEE with secure hardware accelerators to achieve ULP long-term security using quantum-secure crypto cores and secure CIM-based neuromorphic computing.

To streamline the work in WP3, the above goals are addressed in four tasks:

- T3.1 Requirements, Threats, and Vulnerability Analysis (M01 to M06)
- T3.2 Security Architecture for Trusted Execution Environments (M03 to M30)
- T3.3 Composable Implementation of Secure Cores and Trusted Execution Environment (M13 to M33)
- T3.4 Security Evaluation and Assessment (M13 to M33)

This deliverable is based on the findings of task T3.4, i.e., security evaluation and assessments of all the components developed for the preceding deliverables. These components include the TEE, secure PQC cores, secure CIM accelerators, and the composable framework. The evaluation covers the experimental setup, results from vulnerability assessments of baseline designs, formal security proofs for the secure implementations, and analysis of associated overheads.

1.2 Deliverable Structure

The subsequent sections of this document are structured as follows: Section 2 first defines security requirements for each component and then highlights and elaborates the key performance indicators (KPIs) of the current work package as defined in WP1. Section 3 describes the verification methodologies, experimental setups, and assessment criteria for each component. Section 4 presents the results of the experiments conducted as proof of the security or functional integrity of each individual component. Section 5 outlines the Chimera SoC block diagram with integrated security features. Finally, Section 6 concludes this document with discussions on the achievements of WP3 and future work.

2 Security Requirements and KPIs

The security requirements and KPIs for each component of WP3 are detailed in WP1 and are summarized in Table 1. This table is an updated version of Table 2 in D3.3, streamlined in accordance with each component in WP3.

TABLE 1 OVERVIEW OF SECURITY REQUIREMENTS ADDRESSED BY EACH COMPONENT OF OUR SECURITY ARCHITECTURE

Security Requirement	TEE		Secure PQC Cores and DSE	Secure CIM	Composable Framework
	Keystone + PQC	Free RTOS + PMP*			
Secure Boot	X				X
Memory Encryption	X	X			
Long-term PQC Security	X		X		
Hybrid Classic & PQC Security	X		X		
Timing Side-channel Resistance	X	X	X	X	
Power Side-channel Resistance			X	X	
Fault Injection Resistance		X			
Encr. & Auth. Communication	X		X		X
Low Power Cryptography				X	
Composable Security Framework					X

PMP = Physical Memory Protection

Detailed description of component-wise security requirements and KPIs is as follows.

2.1 Trusted Execution Environments

The TEE part of the architecture comprises two components: Keystone + PQC, and FreeRTOS with Physical Memory Protection (PMP). The following subsections outline the security requirements for each of them.

2.1.1 Keystone + PQC

The primary security requirement of a TEE is the ability of the SoC to isolate so-called enclaves from another, including any operating system. This also includes the ability to remotely attest the correct operation of the TEE. This involve a cryptographic proof of the systems hard and

software, such that a remote verifier can confirm the system is running in secure known state, and that the individual enclaves are indeed securely isolated from another.

For long-term security against quantum attackers, we also add that the cryptography must be secure against quantum attackers. In addition, the cryptography should also adhere to the security requirements in the next section, notable, the correctness and timing side-channel resistance.

2.1.2 FreeRTOS + PMP

The following are the security requirements for a FreeRTOS-based system with PMP. To enhance data privacy, strict isolation between tasks must be enforced, ensuring each task operates within its own memory space and interacts with the kernel only through a secure system call interface. The FreeRTOS kernel must be isolated from all task code and data to maintain system integrity. Inter-task communication must be secured such that neither the kernel nor other tasks can access the content of exchanged messages. PMP configurations must be hardened against fault injection (FI) attacks to prevent unauthorized memory access. Task switching must be optimized to reduce execution time and support real-time responsiveness. External memory must be encrypted using constant-time algorithms to resist timing side-channel attacks. Lastly, the system must support secure and efficient dynamic memory allocation to enable flexible resource management without compromising isolation or security. Our KPIs primarily address task switch time overhead and inter-task communication latency. Additionally, we aim to achieve a significantly reduced Trusted Computing Base (TCB), targeting less than 30% of the FreeRTOS kernel's code size.

2.2 Secure Cores and DSE

We distinguish between *security requirements* and *KPIs* for the evaluation of secure cryptographic cores and our HADES tool. The former are strict yes-no requirements that are indispensable. Any design not meeting these requirements is - by definition - not eligible. The latter KPIs indicate the actual performance, e.g., area or latency, on a continuous scale.

2.2.1 Security Requirements

For the hardware designs created with HADES we have three security requirements: cryptographic correctness and resistance against power-side channels and timing attacks. Note that we do not consider fault-injections or combined attacks, that is an attacker who, in addition to monitoring the device, can introduce targeted errors such as individual bit flips or bit resets.

Cryptographic Correctness. The cryptographic algorithm must obviously be implemented correctly and as specified. In case the algorithms are officially standardized, the specified implementation techniques and parameters must be used. This is, for instance, the case for CRYSTALS-Kyber and CRYSTALS-Dilithium, which have been selected as first-round winners of the NIST Post-Quantum Cryptography Call. Following official specifications enables seamless interactions with other devices, such as remote controllers.

Power Side-Channel Resistance. All implementations should be resistant against power-side channels. Abstractly speaking, a power side-channel means that an attacker can probe the value of one or more wires of the circuit during execution. In practice, the actual "probing" is done using advanced statistical methods and millions (or even billions) of power traces. We require that each design is at least first order secure, i.e., an attacker that probes one wire cannot learn anything about the key. As a bonus, HADES actually supports arbitrary security orders at the cost of significantly larger and slower designs, but we only consider first order security for CONVOLVE and in this deliverable.

Timing Side-Channel Resistance. The runtime/latency of all implementations should be independent of any secret. Otherwise, an attacker can infer information about secret data based on the execution time.

2.2.2 Key Performance Indicators

We consider two KPIs to assess the hardware accelerators obtained with HADES as well as the HADES tool itself.

Better Hardware Designs. First and foremost, the hardware accelerators must be competitive with state-of-the-art implementations from scientific literature. To this end, we compare the outputs of HADES with existing reference implementations regarding KPIs such as the latency, area and randomness requirements. We also consider combined performance metrics, e.g. area-latency-product (ALP), to identify potential sweet spots.

Faster Design Time. The HADES tool systematically explores a large number of possible hardware implementations for a given cryptographic algorithm. As our KPI, we compare the time to identify the most optimal design for a given performance metric to that of manual analysis. Further, we discuss that the implementer can focus on the algorithm **without** considering power side-channel security, as this is added mechanically. As a result, the design process is simplified and accelerated.

2.3 Secure CIM

The security requirements defined for Secure CIM in Table 1 are timing side-channel resistance, power side-channel resistance, and low power. These requirements are placed to make sure that the protected CIM design should be secure against power and timing side-channel attacks and should bear minimum area and energy overheads. For power side channel analysis, we assume CIM designs should be secure against first-order reverse attacks for a minimum of 1 million traces. For timing side-channel analysis, we require that the execution time and pattern should not give away sensitive information regarding the NN model being executed in the CIM accelerator. Finally, ultra-low-power requirement translates to achieving the above-mentioned requirements with minimum possible design overheads.

The KPIs considered for secure CIM design are minimum number to disclosure (MTD), and design overhead. MTD, as the name suggests, is the minimum number of traces required to extract the proprietary weights from the NN model implemented in CIM. In our scheme, we consider the design secure if the MTD remains below 25% for an MTD of 1 million traces. It is important to know

that at least 75% of the weights should be known to reverse engineer an NN model with minimal functionality. On the other hand, the design overhead is referred to as the area and energy overhead inferred by the secure design as compared to the baseline design. Therefore, we only focus on exploring countermeasure schemes that incur minimum overhead. The goal is to find an optimal trade-off between security and design overhead.

2.4 Composability Framework

Security requirements for real-time composable frameworks have been investigated with the aim of merging the real-time, composable guarantees of a composable platform with the security provided by a TEE. After establishing that simply wrapping the real-time composable framework with a TEE would disrupt predictable WCETs, investigating how to apply similar security to the virtualised components of a real-time Composable framework could be achieved.
A strictly predictable and composable framework such as CompSOC provides similar isolation of virtualised hardware to that of a TEE. While not previously researched, the remaining security elements required are security on boot, which is notionally handled with secure boot tools such as OpenSBI, and secure transfer of information between applications from the isolated virtualised hardware and the owner of the applications that may be remote. A good example of this is over-the-air software updates that need to be tested for authenticity and integrity.

We investigate lightweight security methods for data transfer that fit within the constrained resources of a real-time composable platform for SoC. We demonstrate with the CompSOC platform that common secure data transfer tools such as hashing, digital signing, and encryption can be applied within the virtualised hardware at the application level. While the methods are well known, the ability to incorporate these onto a composable platform is not. We demonstrate our proposed methods operate on the CompSOC composable platform through measurements of the processing and memory requirements of each implemented security feature. Combining the secure transfer with secure boot and the inherent isolation of applications in a real-time composable framework, we can achieve similar security to the TEE while retaining the predictability of execution time.

3 Verification Methodology

To ensure robust protection, a comprehensive verification methodology is essential for evaluating the security of the design. The following are the security evaluation schemes of all the components involved in WP3.

3.1 Trusted Execution Environment

As TEEs are broad in scope, different implementations can be based on varying threat models and security requirements. The following sections outline the verification methodology applied to each of these components.

3.1.1 Keystone + PQC

We perform benchmarking on our demonstrator SoC, based on the Chipyard framework. We run both performance benchmarks to measure cycle counts of the cryptographic operations, as well

as known answer test to ensure the outputs (e.g. keys, signatures etc.) match with known software refence implementations.

3.1.2 FreeRTOS + PMP

The FreeRTOS TEE version has been tested in various ways to access or execute a task's data or code from other tasks or from the kernel, as well as for inter-task communication. The task-switching mechanism ensures that all previous information in the CPU registers is overwritten, preventing information leaks between tasks. The system call interface is designed to offer only a minimal set of services with as few parameters as possible, making it straightforward to assess. For fault injection (FI) protection, we developed a hardware unit that is able to speed up writing to PMP registers and, at the same time, can detect corruptions. The PMP configuration was randomly corrupted in simulation, which consistently resulted in an error. While there are theoretical ways to bypass the FI protection, they require the ability to precisely modify multiple specific bit positions simultaneously. That is something we do not consider realistic without expensive equipment and expert knowledge of the system and fault injection techniques. We integrated a memory encryption unit for the external memory. The encryption unit uses the PRINCE algorithm that performs encryption and decryption in constant time.

3.2 Secure Cores and DSE

To guarantee the security of the hardware implementations generated by HADES we rely on theoretical security proofs, composability rules as well as practical security evaluation.

Reference Implementations. To assert the functional correctness of our implementations we use known-answer tests from official standardization institutions such as NIST[1]. Additionally, the outputs can be compared with (unoptimized) reference implementations. Given the non-deterministic nature of cryptography and the avalanche effect, we obtain very strong correctness guarantees.

Provably Secure Gadgets (HPC). In order to ensure that the designs are resistant against power side-channel, we build on strong theoretical results. More precisely, we use so called gadget-based masking. A gadget is a logical unit which computes a function such as AND or OR *without* leaking information via side-channels. The inputs to a gadget are shared, that is instead of computing AND(a, b), we compute AND(a_1, a_2, …, a_d, b_1, b_2, …, b_d) where a = XOR(a_1, a_2, …, a_d) and analogously for b. Additionally, the gadget requires randomness which is used to "mask" the values. Several different masking gadgets such as HPC1, HPC2 and HPC3 have been presented in literature. They have fundamentally different performance trade-offs, like requiring more/less randomness or more/less cycles. Additionally, they have been formally proven to be secure and composable. By using these gadgets instead of constructing custom (potentially insecure) circuits we can leverage (some) of the security guarantees.

Figure 1 below shows an HPC2-AND gadget for a security order d=1. As explained earlier, the inputs are given as separate shares.

[1] https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/block-ciphers

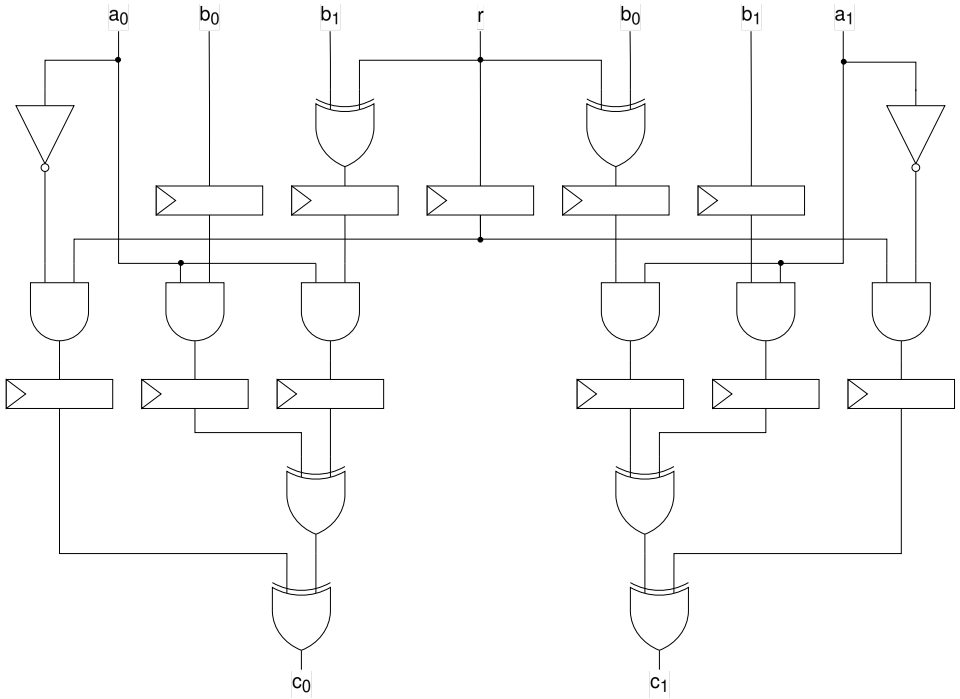


FIGURE 1 HPC2-AND GADGET FOR SECURITY ORDER D=1. THE INPUTS ARE GIVEN AS TWO SHARES EACH AND THE GADGET REQUIRES 1 BIT OF RANDOMNESS

Also, the gadget requires a random bit r and outputs its results as two shares. One can easily see the immense overhead introduced by masking countermeasures: a single AND gate is expanded to six AND gates, six XOR gates, two inverters, and two register stages.

These gadgets also have two more advantages. First, they are scalable to arbitrary masking orders. This is the key component that allows hardware implementers to be agnostic of any security order and later let HADES mechanically replace gates with d-th order masked gadgets. Second, the gadgets are modular, and their composition remains secure if simple "composability notions" are respected. In contrast, when "manually masking" an implementation, one has to ensure that "no combination of signals from up to d wires is statistically dependent on the secret" for the entire circuitry. To make sure we properly replaced and composed everything, we empirically verified our designs in our lab.

Practical Security Evaluation (TVLA). We also verify the side-channel resistance empirically using *test vector leakage assessment* (TVLA) in our lab. Concretely, we used a first-order secure, round-based AES128 with HPC3 gadgets on a SakuraX board with Kintex 7 FPGA. The HPC3-gadgets are implemented as Xilinx primitives (LUTs) to avoid synthesizer optimizations. To capture the power traces, we use a ZFL-2000GH+ low-noise amplifier with a gain of 25dB and a Spectrum M4 oscilloscope with 8-bit resolution and a sampling rate of 1.25GS/s. As a randomness source, we have a parallel Keccak with a 1600-bit state as a pseudo-random number generator (PRNG). We collected 100 million traces and then performed Welch's t-test based on the first statistical moment, as this corresponds to a first-order attacker. We expect a t-value <4.5 since this indicates with overwhelming probability that no information is leaked. As a sanity check, we repeat the t-test on the second statistical moment (corresponding to a second-order attacker). We expect clear indications of leakage for this case, as otherwise the testing setup is flawed.

3.3 Secure CIM

To validate the security of our design, we perform side-channel analysis at various levels of abstraction (such as pre-silicon and post-silicon) using standard assessment criteria such as CPA, TVLA, and SNR. We also explored ML-based models first for extraction and analysis. In addition to that, we performed a statistical assessment of our Pseudo random number generator (PRNG) sequence using NIST statistical test suite[2].

Pre-silicon Test Setup. For the security evaluation of pre-silicon design, we create a test setup for trace generation and analysis. The RTL design is synthesized using technology libraries and Cadence IC design tools. A gate-level simulation is performed on the generated netlist to create a Value change dump (VCD) file. This VCD file, along with the netlist, is then provided to a power estimation tool such as Synopsys Splyglass to generate a trace. White Gaussian noise is then added to the trace to emulate real-world conditions. These traces are then evaluated for side-channel assessment against above mentioned assessment methodologies.

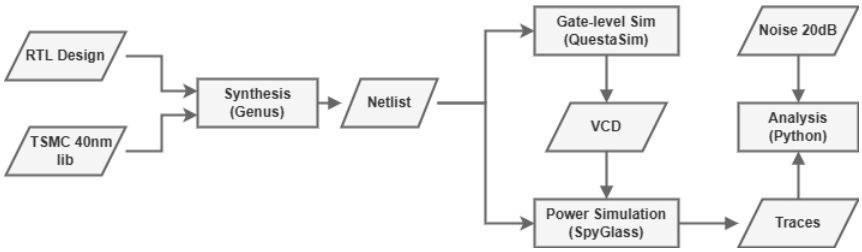


FIGURE 2 TEST SETUP FOR PRE-SILICON TRACE COLLECTION AND ANALYSIS

Post-silicon Test Setup. For post-silicon security evaluation of the design, we utilize a power side-channel analysis platform called ChipWhisperer, provided by NewAE technology. We implement the digital periphery of the CIM accelerators on Artix XC7A100T FPGA housed on CW305 development board to acquire power traces. The required number of traces are then generated in Python utilizing the toolchain provided by ChipWhisperer.

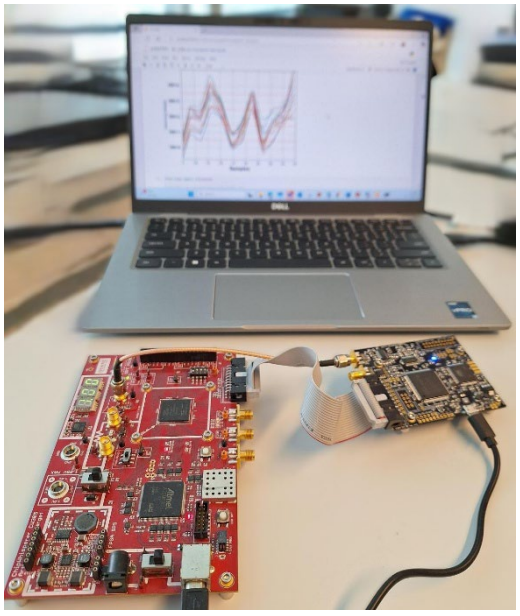


FIGURE 3 TEST SETUP FOR POST-SILICON TRACE COLLECTION AND ANALYSIS

[2] A. Rukhin et al., "A Statistical Test Suite for Random and Pseudo random Number Generators for Cryptographic Applications," NIST, Tech. Rep., 2010.

Security Assessment. There are numerous techniques in the literature that are utilized to perform security evaluations. Among those, TVLA and SNR are the leakage detection techniques, but they do not reveal the actual secret data. CPA is a widely used value extraction method. We perform our side-channel analysis on 1 million traces acquired from either of the two methods mentioned previously.

3.4 Composability Framework

We demonstrate a lightweight security framework that runs at the application level on stat-of-the-art real-time composable platform CompSOC, validated with experimentation on FPGA. The developed composable security framework for data transfer is a basic set of open source tools that provide all necessary functionality for confidentiality, integrity and authenticity. Key metrics for this investigation is to verify that a software solution running from the virtualised hardware at the application level can be implemented such that it functions correctly, fits within small memory constraints, and executes within a reasonable timeframe.
The developed methods are gathered as an open-source library that has no dependencies and will run with basic C interpreters. Each of the elements of the security framework are validated to perform their respective functions, and also analysed over a range of expected message lengths where relevant and assessed based on memory and clock cycle requirements. We have demonstrated that each of the proposed functions fit within the confined virtualised hardware requirements and describe the measured resource costs in terms of memory size and clock cycles.

4 Component-based Security Evaluation Results

The security evaluation results for each individual component against the verification methodology in Section 3.

4.1 Trusted Execution Environments

This section begins by presenting the results for the TEE components. The results for each component are reported separately, detailed in dedicated subsections.

4.1.1 Keystone + PQC

We are able to successfully boot Linux on our demo SoC. During boot, the security monitor indicates via debug console output that its initialization has succeeded and that the PMP registers have been set correctly to ensure subsequent isolation.

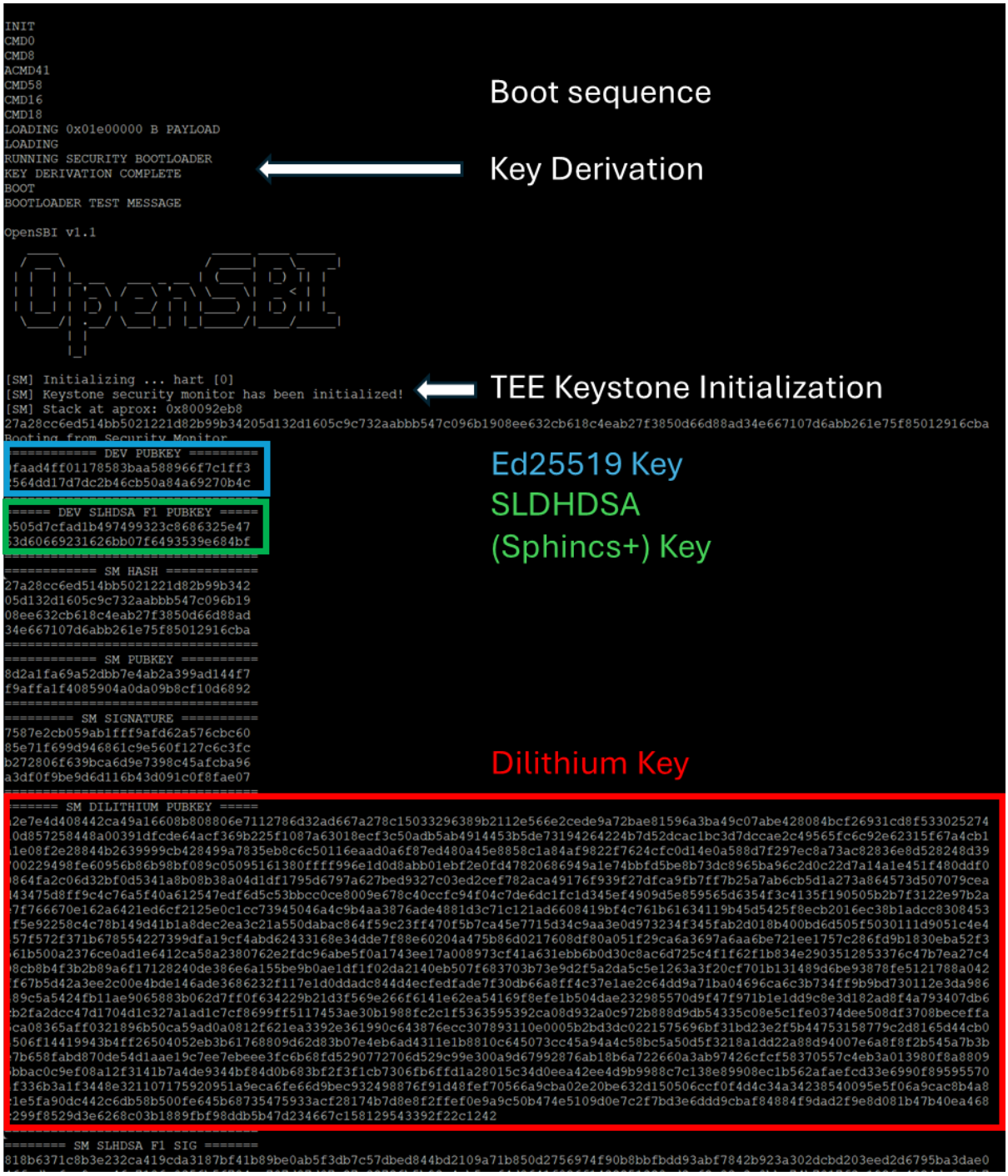


FIGURE 4 INITIALIZING KEYSTONE SECURITY MONITOR. WE CAN SEE THE HASH OF THE SM, THE UNIQUE DEVICE PUBLIC KEY, AS WELL AS THE GENERATED KEYS AND SIGNATURES.

Once the Linux OS has booted, we run an enclave that is isolated from the system. The enclave performs a remote attestation operation, and the resulting attestation report can be verified. A modification to the enclave or the security monitor, as would be performed by an attacker, immediately leads to a failed attestation report verification.

FIGURE 5 ATTESTATION OPERATION AND REPORT GENERATION. WE CAN SEE THE HASH OF THE ENCLAVE AS WELL AS THE SIGNATURES OF THE ENCLAVE AND SECURITY MONITOR, AS WELL AS THE VERIFICATION OF THE REPORT.

An attempt to access memory outside if the enclaves designated region also immediately results in a hardware fault, as does an attempt of the Linux OS to access the memory region of the enclave or security monitor.

The cryptography used in the attestation report is the prequantum Ed25519, as well as the post-quantum Dilithium and Sphincs+ (also known as ML-DSA and SLH-DSA). This ensures optimal long-term security.

The integration of the Keccak accelerator has massively sped up all the signing operations of Sphincs+ during boot, by a factor of between 282 and 343, depending on the operation.

Dilithium is also sped up, with an acceleration of the Dilithium signing algorithm by 37%.

General purpose hashing, such as SHA3 and SHAKE that use the Keccak accelerator are also sped, with the hashing of one block seeing an acceleration of 9,5.

In addition, by using the modified parameter set for Sphincs+ with a reduced maximum signature count, the signature size is reduced from 7856 bytes to 3904 bytes. This is very close to the combined signature and public key size of Dilithium, which together are 3936 bytes. The Sphincs+ public key is very small, at 32 bytes.

Thanks to the addition of Dilithium and Sphincs+, the signatures generated by the TEE are now post-quantum secure and offer maximum long-term security.

For Dilithium, we also measure the memory consumption of the optimized low memory. For the most memory intensive operation, signing, the memory was reduced from 52KB to 6176 bytes. This did incur a performance penalty of 1.9. However, the lower memory allows us to reduce the per core scratch memory of the SM to 24KB, from 128 KB initially.

Our optimization of the boot-ROM reduces the size from 60KB to just 15.2 KB. This is even smaller than Keystone default boot-ROM size, which is 50 KB in size, even without PQC algorithms.

4.1.2 FreeRTOS

We evaluated the execution time for task switching, inter-task communication using queues and stream buffers and dynamic memory allocation (malloc/free). We compared the secure FreeRTOS version, here called TeeRTOS to the baseline version of FreeRTOS, two FreeRTOS versions with less security requirements (only task isolation, kernel is trusted) called FreeRTOS-PMP and FreeRTOS-MPU. The FreeRTOS-PMP was also developed by us as an intermediate step to the TeeRTOS implementation. The FreeRTOS-MPU cycle numbers are from [DSD+22] and normalized to our numbers. For accelerating the task switches we developed a novel hardware unit, called the PMP Snapshot unit (PSE), that can reconfigure the PMP configurations during a task switch in only one cycle.

The results in Figure 6 show that the FreeRTOS-PMP cycle numbers are usually higher than the FreeRTOS-MPU numbers. This is because, in their work, a simpler and less secure system call architecture is used, as well as only 7 MPU regions, which is less than the 16 PMP regions that we used. However, they do not support dynamic memory allocation. For the task switch, TeeRTOS is 28% higher than FreeRTOS-PMP. When the PSE is enabled, it is effectively reduced to 4%. For sending and receiving messages via the queue and stream buffer, the increase is significantly higher. The reason for that during these operations, the FreeRTOS kernel is invoked multiple times. Because the FreeRTOS kernel is not trusted, more context switches and PMP reconfigurations are needed. The PSE is able to reduce the overhead. We also implemented the send and receive in a way that the FreeRTOS kernel cannot access the messages (purple), but this requires additional context switches. Lastly, the overhead for dynamic memory allocation with malloc and free is significant. This is because the actual malloc and free implementations are very simple and can be executed with only a few cycles. When a memory region is allocated for the task, the PMP region needs to be added. This requires the intervention of the secure TeeRTOS kernel, because neither the FreeRTOS kernel nor the task itself is able to change the PMP rules. This feature is not available in other secure FreeRTOS implementations yet.

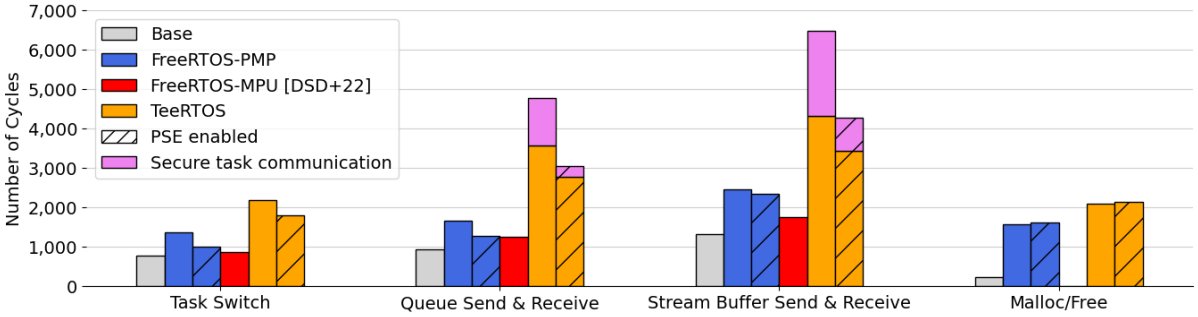


FIGURE 6 RESULTS COMPARISON WITH STATE-OF-THE-ART [3]

4.2 Secure Cores and DSE

Correctness and Constant-Time Implementations. All of our designs are correct in so far, as their outputs pass known answer tests and behave, to the best of our knowledge, equivalent to the reference implementation. Further, we exclusively support standardized parameter sets and follow the specifications. The latency of implementations is independent of any secrets by algorithmic design; consequently, no timing differences can be exploited to obtain information about secret values.

Empirical TVLA. The practical TVLA results are shown in Figure 7 and Figure 8. The red lines indicate the threshold of 4.5. As expected, the t-value is constantly between -4.5 and 4.5 for the first-order t-test experiments. This indicates that no data is leaked with overwhelming likelihood even with access to 100 million traces. A practical attack scenario is therefore not plausible for a first order attacker.

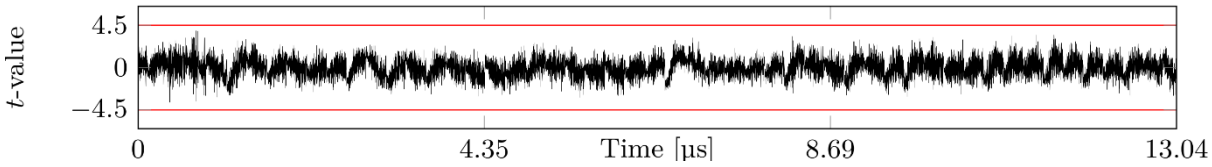


FIGURE 7 FIRST-ORDER T-TEST RESULTS FOR HPC3 PROTECTED AES-128 CORE

In Figure 8, we show that leakage is indeed detected (threshold of 4.5 is clearly exceeded) if we assume a second-order attacker. This is a vital sanity check and assures that the test setup is working properly. Note that this does not mean that our designs are not secure against second-order attackers if we were to use HPC3-gadgets with masking order d=2.

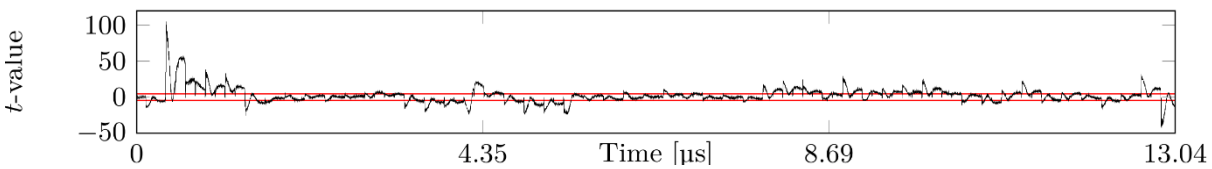


FIGURE 8 SECOND-ORDER T-TEST RESULTS FOR HPC3 PROTECTED AES-128 CORE

[3] [DSD+22] Du, Yufei, et al. "Holistic {Control-Flow} protection on {Real-Time} embedded systems with kage." _31st USENIX Security Symposium (USENIX Security 22)_. 2022.

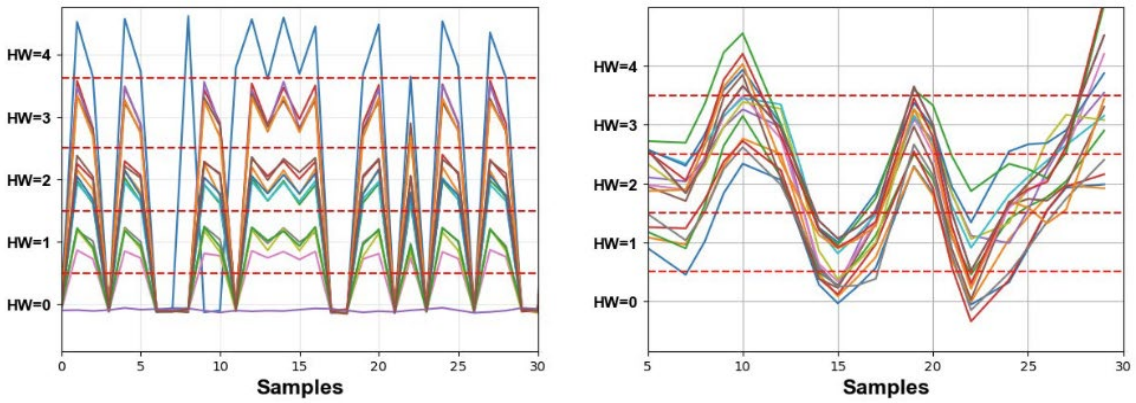
4.3 Secure CIM

We performed our analysis on two digital CIM designs and hence present the results for both designs.

4.3.1 Adder-tree based Design

We conducted power side-channel vulnerability analysis on a secure adder-tree-based CIM design and evaluated its resistance against TVLA, CPA, and K-means clustering attacks. To validate the effectiveness of these attack methodologies, we first applied them to a baseline (unprotected) version of the design.

Figure 9 illustrates the results of an ML-based attack where K-means clustering was applied to power traces. This technique categorizes NN weights based on their HW. In the unprotected design, a clear separation of HW categories is visible, indicating significant information leakage. In contrast, the protected design shows no discernible clustering, demonstrating effective protection against such leakage.



(a) Unprotected Implementation (b) Balanced Obfuscated-path Implementation

FIGURE 9 K-MEANS CLUSTERING FOR NN WEIGHT CATEGORIZATION ON ADDER-TREE BASED CIM

TVLA tests on unprotected and protected designs are shown in Figure 10 and Figure 11 respectively, further highlighting the effectiveness of the protection. The t-values exceeding the threshold region of ±4.5 (shown in red dotted lines) in Figure 10 indicate the presence of statistical similarity between the secret (weight values) and power patterns. While the t-values in Figure 11 remain within the threshold region, indicating no statistical similarity between the weight values and the power switching pattern.

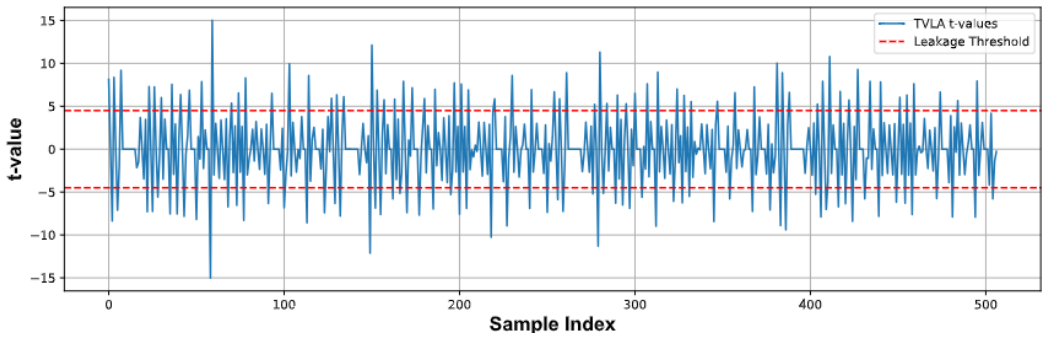


FIGURE 10 TVLA RESULTS OF UNPROTECTED ADDER-TREE BASED CIM

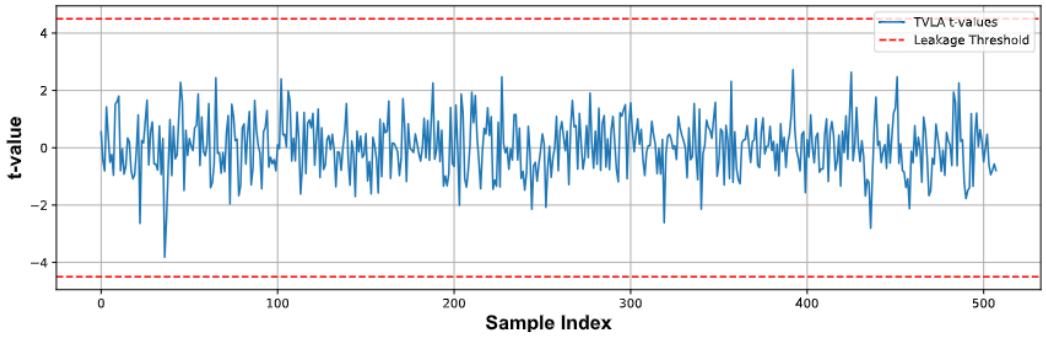


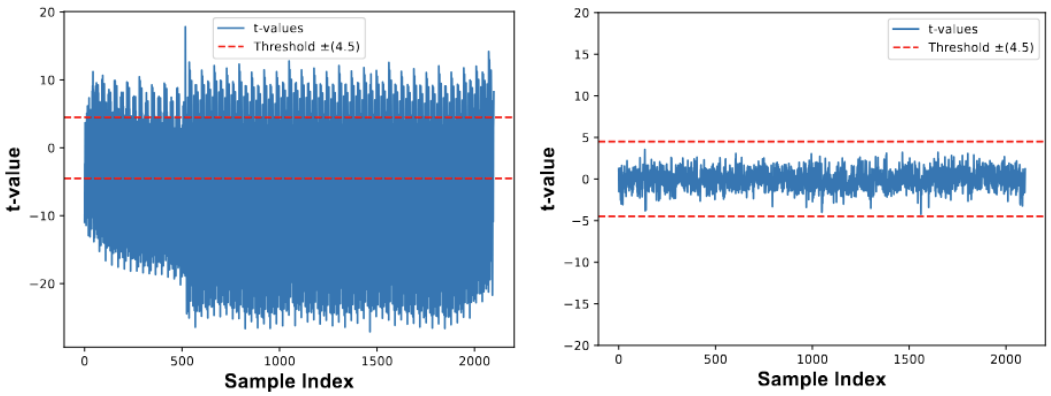
FIGURE 11 TVLA RESULTS OF PROTECTED ADDER-TREE BASED CIM

To assess the overhead incurred by the proposed security enhancements, our designs was evaluated against an unprotected baseline in terms of area, energy, and performance overheads. The design incurs moderate area (44%) and energy (89.8%) overheads but maintains latency due to a pipelined register design. An advanced, higher-order secure approach, which integrates glitch-resistant C-elements, results in significantly higher area (214%) and energy (356%) overheads. This overhead is 3x area and 3.65x energy efficient as compared to state-of-the-art masking schemes, highlighting our conformance with KPIs.

4.3.2 Adder-free Design

We evaluated post-silicon power side-channel analysis of the digital periphery of the adder-free digital CIM design using TVLA, CPA, and SNR. At first, TVLA is applied in a semi-fixed configuration to localize the specific time samples associated with input-dependent leakage. In this configuration, the test divides power traces into two sets: one with fixed inputs and weights, and the other with partially varying inputs while keeping weights constant. By applying the TVLA on these sets, we identify time intervals where the power consumption correlates with the toggling of specific input bits, pinpointing the temporal location of potential leakage. Figure 12(a) illustrates the effect of semi-fixed TVLA configuration. The information derived from this localization is then used to guide a CPA, targeting those time instances to recover the trained NN weights. As shown in Figure 12(b), all t-values remain within the ±4.5 threshold, confirming that the secure design exhibits no observable leakage. For a more quantitative assessment, we applied CPA on the power traces to extract the NN weight values. Figure 13 shows the weight guessing success rate. For the unprotected design, 100% of the weights could be correctly recovered using just 4.5K traces. However, for the protected design, the success rate remained consistently low even with up to 1 million traces, confirming the effectiveness of the implemented countermeasure.

The proposed countermeasure effectively protects against power leakage with significantly lower overhead compared to traditional masking techniques. Due to the lack of prior FPGA/ASIC implementations of shuffling-based defences for BNN accelerators, direct comparisons are limited. However, when benchmarked against a state-of-the-art masking scheme, the proposed design demonstrates a substantial reduction in area and power overhead (only 37% and 51.2% on ASIC, and 33.6% and 36.8% on FPGA), while maintaining zero latency impact. This highlights its conformance with KPIs and suitability for secure, resource-constrained environments.



(a) TVLA Without Countermeasure (b) TVLA With Countermeasure

FIGURE 12 TVLA RESULTS INDICATE NO LEAKAGE BEYOND THE THRESHOLD IN THE PROTECTED DESIGN

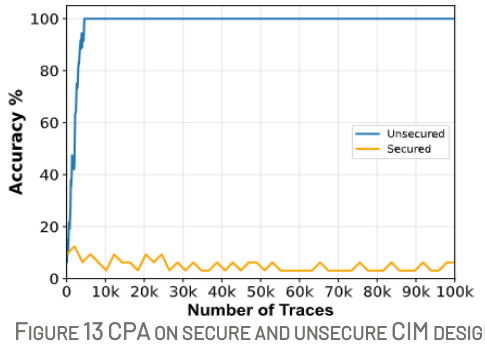


FIGURE 13 CPA ON SECURE AND UNSECURE CIM DESIGN

4.4 Composability

To design a real-time, composable framework that provides the security and isolation of a TEE, we develop a composable security framework that functions within the CompSOC[4] composable framework and provides typical light weight security features at the application level. When combined with the inherent hardware isolation of the CompSOC composable framework[5], this provides a similar solution to the TEE that does not interfere with predictable execution of tasks. Design considerations for the composable security framework are to provide typical confidentiality, integrity and authenticity security features while keeping the processing and memory requirements as low as possible. Additionally, we aim to have no dependencies and provide an open source solution that requires only a C interpreter so that these tools may be used on other SoC implementations, such as the native PULP / Cheshire[6] host domain target for the Chimera SoC.

As a proof of concept, we implement security features in the composable security framework and check that these can be executed within a virtualised partition on the CompSOC platform. We conduct all experiments on a Xilinx-7020 FPGA SOC PYNQ FPGA running the CompSOC composable platform configured for a maximum of 96KB of memory and a MicroBlaze soft

[4] Goossens, Kees ; Azevedo, Arnaldo ; Chandrasekar, Karthik et al. / Virtual execution platforms for mixed-time-criticality systems : the CompSOC architecture and design flow. In: ACM SIGBED Review. 2013.
[5] Goossens, K. *et al.* (2017). NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications. In: Ha, S., Teich, J. (eds) Handbook of Hardware/Software Codesign. Springer, Dordrecht.
[6] PULP / Cheshire architecture - https://pulp-platform.github.io/cheshire/um/arch/

processor running a RTOS micro kernel per Virtual Execution Platform (VEP). These VEPs consist of memory and processor resources and are isolated spatially and temporally so that it is impossible for applications on separate VEPs to affect each other. This is by design to protect the predictable execution time, however applies isolation in a security context as well. All composable security methods in the framework are validated on FPGA and clock cycle measurements taken. Clock cycle costs where they are message size dependant are calculated by linear best fit on data measurements as the message length is varied. See Figure 14 for measurements of the SHA-256 Execution time.

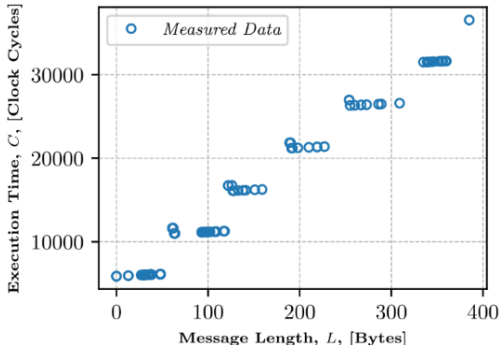


FIGURE 14 SHA-256 EXECUTION TIME VS MESSAGE LENGTH. NOTE THE LINEAR GROWTH IN EXECUTION TIME WITH MESSAGE LENGTH, WITH THRESHOLDING AT REGULAR INTERVALS WHERE NEW BLOCKS OF DATA ARE ALLOCATED.

In Table 2, we list the composable security framework methods and their respective code size processing overhead in cycles. A simple 1 bit checksum is implemented as an absolute minimum integrity check. An improved integrity check using a SHA-256 algorithm is also tested and this is extended to HMAC digital signing by combining SHA-256 with a shared key that adds authentication and HMAC in turn is extended with a packet counter (HMAC#) that adds protection from replay attacks. AES tools for symmetric encryption / decryption scheme here the Electronic Code Book (ECB) is implemented with a maximum supported plaintext size of 512 bits per message for AES-128. Public-key cryptographic tools for Elliptical Curve Cryptography (ECC) uses Curve25519 which offers 128 bit security with a 256 bit key size. Curve25519 is widely adopted in real-world applications including in Apple's Airplay and OpenSSH.

TABLE 2 LIGHT WEIGHT SECURITY TOOLS AND CRYPTOGRAPHIC IMPLEMENTATIONS – LENGTH OF MESSAGE IN BYTES

Method	Confidentiality	Integrity	Authenticity	Code Size	Clock Cycles
Checksum		Low		512 B	764 + 6 x Length
SHA-256		Moderate		2.56 KB	3328 + 80 x Length
HMAC		High	X	2.56 KB	3328 + 80 x Length
HMAC#		High	X	2.56 KB	3328 + 80 x Length
AES-128	X	High	X	3.65 KB	6403 + 805 x Length
ECC	X	High	X	13.44 KB	26Million / 256 bit key

With this developed set of composable security tools, we provide the basic requirements for data transfer for confidentiality, integrity and authenticity in a composable framework that is executed at the application level. Simple integrity and authenticity measures are relatively cheap, where secure cryptographic protection provided by AES is an order of magnitude more computation, with a one off asymmetric key exchange required that is considerably more expensive to establish the AES session. It is assumed that asymmetric keys are pre-shared and

imbedded in the boot image. In the future, these tools may be integrated into the system application that governs the CompSOC platform, freeing up memory within the virtualised partition.

Clock cycle measurements confirm all security methods in the composable security framework exhibit predictable, input-length-dependent, i.e not content-dependent, execution time. This bounded behaviour is essential for real-time systems. These security tools are very small, however they may be shifted to the system application and requested by the user application with further investigation with the composable framework owners.
With the composable framework secured, multiple untrusted vendors can securely run applications on the same SoC without risk of interference. This eliminates the need for cross validation, which is a further benefit of composable real-time platforms, as it reduces development time for applications significantly when compared to alternative computation frameworks. When combined with some form of secure boot, the composable real-time platform with this developed composable security framework provides a secure composable domain that can be integrated into an SoC where real-time composable functionality is desirable for reducing the validation complexity or providing support for real-time and mixed criticality applications sharing hardware resources.

5 Integration of Security Features in Taped-out SoC

The comprehensive security framework of a fully integrated system is detailed in Section 5 of deliverable D3.3. It outlines a conceptual design that was not integrated into the final CONVOLVE SoC due to a fundamental trade-off between the significant overhead introduced by the security mechanisms and the project's primary goal of achieving a 100x improvement in energy efficiency for ultra-low-power (ULP) edge processing.
This section presents a conceptual blueprint of Chimera SoC (taken from WP7), enhanced with integrated security features. It explores how the security framework could have been incorporated in practice, given that the hardware requirements were not met in the final tape-out, and identifies the architectural changes that would have been required. While not implemented, the described TEE illustrates how such a security solution could be realized in the Chimera SoC, assuming the conditions and requirements outlined in Deliverable D3.2 were met. This serves as a valuable reference for future designs seeking to combine robust security with aggressive energy-efficiency targets.

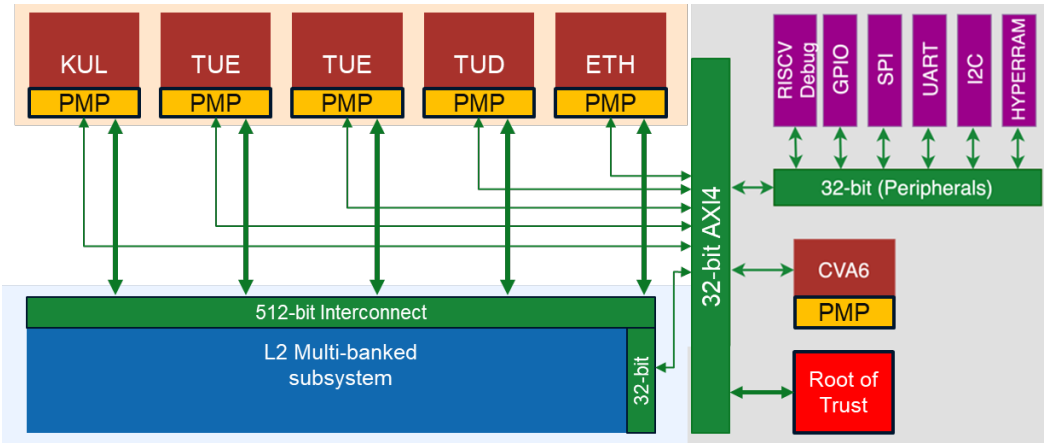


FIGURE 15 CONCEPTUAL TEE IN CHIMERA SOC. REGIONS OF THE MEMORY ISLAND ARE BLOCKED WITH THE PMP REGISTERS, SO THAT THE HOST CORE OR OTHER ACCELERATORS CANNOT ACCESS THEM (E.G. THE MODEL WEIGHTS).

The Security Monitor (e.g., Keystone) in the Host Core is running in M mode and controls the PMP registers upon context switch. The application code would have to run in U-mode. More details of this can be consulted in Deliverable 3.2, Section 4.

6 Conclusion

Deliverable D3.4 concludes the security evaluation and assessment activities within WP3 of the CONVOLVE project. Throughout this work package, we developed and evaluated a comprehensive, modular security framework tailored to the diverse requirements of edge computing platforms. These requirements were derived in collaboration with WP1 and use-case partners. Key achievements include the development of quantum-safe trusted execution environments, formally verified secure cores, side-channel-resistant CIM accelerators, and a lightweight composable security framework, as described in D3.3 "Update of Security Architecture and TEE Implementations". These components collectively address critical security challenges in edge computing, ranging from physical attack prevention to long-term cryptographic security.

The evaluation results demonstrate several notable successes across the project's components. The HADES tool passed both formal verification and practical TVLA tests, confirming its robustness against power analysis while meeting timing and correctness requirements. In TEE, the integration of post-quantum cryptographic algorithms and associated hardware optimizations showed significant improvements in performance and memory efficiency. The FreeRTOS and Keystone implementations achieved secure task isolation and remote attestation with minimal overhead. Secure CIM accelerators demonstrated strong resistance against power and timing side-channel attacks, validated through rigorous testing methodologies like TVLA and CPA. Finally, the composable security framework achieved its goal of providing essential security features while maintaining predictable execution times, crucial for real-time systems. All KPIs outlined in Section 2 were successfully met in the final demonstration.

The results confirm that the components developed in WP3 can serve as a secure foundation for future edge SoCs and highlight opportunities for further integration and optimization. Looking ahead, future work could extend the current security framework to address higher-order and combined physical attacks, refine PQC implementations, and support system-level integration. Overall, the outcomes of WP3 provide a strong basis for addressing emerging security challenges in next-generation edge computing systems.