Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale

WP2 New accelerator designs exploiting mixed precision

D2.5 eXtreme Secure Crypto IP, part 2

V1.0

http://textarossa.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme, EuroHPC JU, grant agreement No 956831

TEXTAROSSA

Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale

Project Start Date: 01/04/2021 **Duration**: 36 months

Coordinator: *AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE, L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE - ENEA, Italy.*

Deliverable No	D2.5
WP No:	WP2
WP Leader:	CINI-UNIPI
Due date:	M30
Delivery date:	30/11/2023

Dissemination Level:

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Grant Agreement No.: 956831

Deliverable: D2.5 eXtreme Secure Crypto IP, part 2

DOCUMENT SUMMARY INFORMATION

Project title:	Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale
Short project name:	TEXTAROSSA
Project No:	956831
Call Identifier:	H2020-JTI-EuroHPC-2019-1
Unit:	EuroHPC
Type of Action:	EuroHPC - Research and Innovation Action (RIA)
Start date of the project:	01/04/2021
Duration of the project:	36 months
Project website:	textarossa.eu

WP2 New accelerator designs exploiting mixed precision

Deliverable number:	D2.5					
Deliverable title:	eXtreme Secure Crypto IP, part 2					
Due date:	M30					
Actual submission date:	02/12/2023					
Editor:	Sergio Saponara					
Authors:	S. Saponara, S. Di Matteo					
Work package:	WP2					
Dissemination Level:	Public					
No. pages:	28					
Authorized (date):	30/11/2023					
Responsible person:	Sergio Saponara					
Status:	Plan	Draft	Working	Final	[✓ Submitted]	Approved

Revision history:

Version	Date	Author	Comment
0.1	2023-10-31	S. Saponara	Draft structure
0.2	2023-11-15	S. Di Matteo	First version completed
0.3	2023-11-20	S. Saponara, S. Di Matteo	Revision
1.0	2023-11-27	S. Saponara, S. Di Matteo	Including answer to internal reviewer

Quality Control:

Checking process	Who	Date
Checked by internal reviewer	Carlos Alvarez	25 November 2023
Checked by WP Leader	Sergio Saponara	27 November 2023
Checked by Project Coordinator	Massimo Celino	December 1st, 2023

COPYRIGHT

ACKNOWLEDGEMENTS

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement no 956831. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Italy, Germany, France, Spain, Poland.

Please see http://textarossa.eu for more information on the TEXTAROSSA project.

The partners in the project are AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE, L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE (ENEA), FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V. (FHG), CONSORZIO INTERUNIVERSITARIO NAZIONALE PER L'INFORMATICA (CINI), INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), BULL SAS (BULL), E4 COMPUTER ENGINEERING SPA (E4), BARCELONA SUPERCOMPUTING CENTER-CENTRO NACIONAL DE SUPERCOMPUTACION (BSC), INSTYTUT CHEMII BIOORGANICZNEJ POLSKIEJ AKADEMII NAUK (PSNC), ISTITUTO NAZIONALE DI FISICA NUCLEARE (INFN), CONSIGLIO NAZIONALE DELLE RICERCHE (CNR), IN QUATTRO SRL (in4). Linked third parties of CINI are POLITECNICO DI MILANO (CINI-POLIMI), Università di Torino (CINI-UNITO) and Università di Pisa (CINI-UNIPI); linked third party of INRIA is Université de Bordeaux; in-kind third party of ENEA is Consorzio CINECA (CINECA); in-kind third party of BSC is Universitat Politècnica de Catalunya (UPC).

The content of this document is the result of extensive discussions within the TEXTAROSSA © Consortium as a whole.

DISCLAIMER

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

The information contained in this document is provided by the copyright holders "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the members of the TEXTAROSSA collaboration, including the copyright holders, or the European Commission be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the information contained in this document, even if advised of the possibility of such damage.

Table of contents

List of Acronyms

Acronym	Definition
ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface (AXI)
CINI	Consorzio Interuniversitario Nazionale per l'Informatica
CKKS	Cheon-Kim-Kim-Song
CPU	Central Processing Unit
CRYSTALS	CRYptographic SuiTe for Algebraic LatticeS
DS	Digital Signature
FPGA	Field Programmable Gate Array
HE	Homomorphic Encryption
HW	Hardware
HPC	High-Performance-Computing
IP	Intellectual Property
IPR	Intellectual Property Rights
JTAG	Joint Test Action Group
KEM	Key Encapsulation Mechanism
XOF	eXtendable Output Function
MPSOC	Multi-Processor SoC
NIST	National Institute for Standard and Technology
PQC	Post Quantum Cryptography
RISC	Reduced Instruction Set Computer
RLWE	Ring Learning With Errors
SE	SEAL Embedded
SEAL	Simple Encrypted Arithmetic Library
SOC	System On Chip
SW	Software
UART	Universal Asynchronous Receiver Transmitter interface

VHDL	VHSIC Hardware Description Language

Executive Summary

This document D2.5 reports the activities done by TEXTAROSSA partners CINI (UNIPISA), with reference to design, verification, and synthesis of accelerator IPs in WP2 for cryptography.

Aiming at giving hardware acceleration to advanced secure services, not already covered by EPI1 activities, the work in TEXTAROSSA implements and verifies in FPGA technology:

- A hardware accelerator for Homomorphic Encryption, having as benchmark the SEAL-Embedded Library provided by Microsoft.

- A hardware accelerator for eXtendable Output Functions (XOFs) SHAKE-128/256, a new family of hashing functions used within algorithms like CRYSTALS-Dilithium, recently standardized by NIST for digital signature generation/verification compliant with a Post-Quantum Cryptography (PQC) scenario.

The Secure Crypto IPs are designed according to the specifications defined in the revised D2.1 [1].

With respect to D2.4 [2], the advancement in D2.5 is mainly in terms of improved and finalized verification of the IPs. To this aim, the reference provided by the SEAL Microsoft library and by the C reference code of the CRYSTALS -Dilithium algorithm have been used.

The IPs proposed in D2.5 have been provided with a standard memory-mapped AXI4 interface, so that they can be easily interfaced to any other digital IPs. To this aim, the implementation results on FPGA of the integration of the two proposed secure IPs with open-source RISC-V 32-bit and RISC-V 64-bit processors are also shown.

The repository of the IPs is also reported and for each of the two IPs a scientific submission to an IEEE journal has been made (one of them already published).

1. Introduction

The adoption of dedicated hardware for cryptography functions and services has become common practice on platforms requiring low-latencies, high-throughput, and energy efficiency. Particularly in the context of High-Performance Computing, computational power and energy-efficiency are crucial aspects. Some of these needs have been already addressed in EPI SGA1, where a cryptography hardware IP has been developed under UNIPI, for accelerating standard cryptographic algorithms for symmetric-key cryptography (based on Advanced Encryption Standard), public-key cryptography (based on Elliptic curve Cryptography), hash functions (based on SHA2 and SHA3 algorithms) and the generation of random numbers.

TEXTAROSSA aims to provide hardware acceleration to advanced security services that have a crucial impact on the context of High-Performance Computing and that are not already covered in EPI1. The need of such security services arises from the need to address emerging challenges and threats in the field of information security, such as: the quantum computing threat, which has the potential to break many of the widely used cryptographic schemes that currently provide security for sensitive information. Traditional cryptographic algorithms, such as RSA and ECC, are vulnerable to attacks using quantum algorithms, rendering them ineffective in a post-quantum computing era; the need of data privacy and confidentiality related to the increasing popularity of cloud computing. The possibility to delegate complex computations to cloud services raises concerns about privacy and security, especially related to the trustness of the providers. The former (the quantum computing threat) requires research effort to provide software and hardware implementations of the new quantum-resistant algorithms. The latter (data privacy on cloud environments) can be overcome thanks to a new cryptographic technique: Homomorphic Encryption.

This document D2.5 reports the activities done by TEXTAROSSA partner CINI (UNIPISA) in WP2, related to the hardware implementation of cryptography services that address the identified new challenges and threats:

1. Development and validation on FPGA of a hardware accelerator for Homomorphic Encryption. The benchmark for this accelerator is the SEAL-Embedded Library provided by Microsoft [3, 4, 5], which is discussed in Section 2.

2. Development and validation on FPGA of a hardware accelerator for eXtendable Output Functions (XOFs) SHAKE-128/256. These functions belong to a new family of hashing functions used in algorithms like CRYSTALS-Dilithium [6]. They were standardized by NIST [7] in Summer 2023 for digital signature generation/verification in a Post-Quantum Cryptography (PQC) scenario. Section 3 provides detailed information on this accelerator.

The Deliverable D2.5 deals with the finalization of the HDL design, using SystemVerilog, verification and synthesis of accelerator IPs for cryptography. The Secure Crypto IPs are designed according to the specifications defined in the revised D2.1 [1] and finalized following the preliminary design in D2.4 [2].

Both Intellectual Properties (IPs) are equipped with a standard memory-mapped AXI4 interface, enabling easy integration with other digital IPs. The implementation results demonstrate the integration of these secure IPs with open-source RISC-V 32-bit [8] and RISC-V 64-bit [9] processors.

Sections 2 and 3 showcase the implementation results using FPGA technology. In the TEXTAROSSA project, FPGA serves as the target technology for hardware design and for characterizing and verifying the IP macrocells. The use of Application Specific Integrated Circuit (ASIC) design is not anticipated.

Section 4 presents the main differences between this document (D2.5), and the previous and preliminary deliverable (D2.4).

Conclusions and the repository of the IPs are reported in Section 5; for each of the two IPs, a scientific submission to an IEEE journal has been made. Section 5 also shows impact and dissemination strategy.

2. Accelerator IP for Homomorphic Encryption

This chapter highlights the hardware design and the implementation results on FPGA of the following cryptography function and service: Homomorphic Encryption (HE) in the context of communication between edge devices and a cloud server, where the goal is ensuring data privacy to the many users accessing the server from edge devices. As already discussed in Section 3.2 of D2.1, this is a new feature, complementary to the security features present in EPI1.

HE is a form of encryption that allows computations to be performed on encrypted data, without decrypting them first. This makes it possible to perform computations on sensitive data while keeping it encrypted, which can be useful in applications where privacy and security are paramount. Some of the main HE libraries and their application are:

- Microsoft SEAL [5]: an open source HE library developed by Microsoft Research. It is used for secure cloud computing, secure data sharing and secure machine learning.
- PALISADE [10]: an open source library developed by New Jersey Institute of Technology for secure computation of financial data and secure machine learning.
- HELib [11]: is a HE library developed by IBM research.

Among them, the SEAL library is considered as a reference for Homomorphic Encryption; it can be downloaded at [13]. In addition, SEAL is adopted by big players of the high-performance computing market like Intel and Nvidia, which are developing hardware optimizations for this library and for HE computations in general. For instance, the Intel Homomorphic Encryption Toolkit (Intel HE Toolkit) [14] is a software and hardware solution that boosts the performance of HE-based cloud solutions running on the latest Intel platforms. It exploits the Intel Advanced Vector Extensions 512 (Intel AVX-512) acceleration instructions to speedup HE computations. Nvidia included homomorphic encryption based on SEAL on the framework for federated learning in health applications called CLARA [15].

2.1. Homomorphic Encryption: SEAL-embedded library and benchmarks on RISC-V

Homomorphic Encryption (HE) is a specialized type of encryption that allows specific computations on the encrypted data and generates a cyphertext that, once decrypted, matches the result of operations performed on the plaintext data. HE is nowadays considered a strong privacy-preserving solution that allows users to share data with clouds or any non-secure server. However, HE requires high computational resources and memory consumption, which limits its use in resources constrained IoT devices. All the HE libraries presented before are not specifically designed for resources-constrained devices. The SEAL-Embedded (SE) library [3,4] is the first HE library targeted for embedded devices that employs several optimizations to perform the encoding and encryption of data, featuring the Cheon-Kim-Kim-Song (CKKS) HE scheme [16].

SE follows the lattice-based algorithm Ring Learning With Errors (RLWE), which states that given R_Q^n ring of integers modulo *Q* with degree less than *n* and given an error distribution χ, the ciphertext can be computed as a couple of polynomials *(a, b)* such that:

$$b = a \ \cdot s + e \ (mod \ Q)$$

where e is an error perturbation sampled independently from an error distribution over χ, a is a public polynomial and s is the secret polynomial (it assumes the meaning of secret key); both belong to R_Q. Retrieving the secret s is considered hard even for quantum computers. In SE polynomial degree *n* is chosen as a power of two. Following the RLWE algorithm, on SE all elements are polynomials represented as *n*−length vectors of their unsigned integer coefficients, whose values may vary in the range of $[0, Q-1]$. The polynomial degree *n* ranges from 1024 to 16384; it impacts the security strength, the size of the encrypted message and the performance of the encryption function. SE encrypts data following the CKKS scheme, allowing encryption over floating point values. The two main functions of the SE library are:

- Encoding: since encryption and decryption work on polynomial rings it is necessary to convert the floating-point message into unsigned integer polynomial without information loss.
- Encryption: follows the RLWE encryption.

In this work, the focus for the hardware acceleration is symmetric encryption, which has been evaluated as the main bottleneck of the SE library.
In SE, the ciphertext is evaluated as a couple of vectors of 32-bits unsigned integers such that:

$$c_0 = -a \cdot s + m + e$$
$$c_1 = a$$

where *a* is randomly sampled from a uniform distribution, *e* is sampled from a centered binomial distribution, *s* is the secret key and *m* is the message to be encrypted (already encoded from floating-point to unsigned integer polynomial).

The following chapters will show first the results of the benchmark campaign carried out using RISC-V processors on FPGA technology of the symmetric encryption function of the SE library, and next the design strategy and the implementation results of the hardware accelerator.

Benchmark on RISC-V CPUs

The source code of the SEAL-Embedded library can be found in [4]. Two different RISC-V processors have been selected for the benchmark campaign, and two different environments have been implemented on the FPGA Board Zynq UltraScale+ MPSoC ZCU106 equipped with the System-on-Chip (SoC) XCZU7EV-2FFVC1156. Figure 2-1 shows the proposed hardware systems running the benchmark.
The selected RISC-V processors are:

- The 32-bit RISC-V RI5CY, whose HDL code can be downloaded in [8]. The left side of Figure 2-1 shows the complete system implemented in the target FPGA which encompasses the RI5CY CPU, 256KB of on-chip memory, and AXI4 peripherals (i.e., JTAG and serial UART interface).
- The 64-bit RISC-V CVA6, whose HDL code can be downloaded in [9]. The right side of Figure 2-1 shows the complete system implemented in the target FPGA which includes the CVA6 CPU, 512MB of memory (i.e., onboard DDR4), and AXI4 peripherals (i.e., JTAG and serial UART interface).

Both systems run at 100 MHz of frequency on the target FPGA.

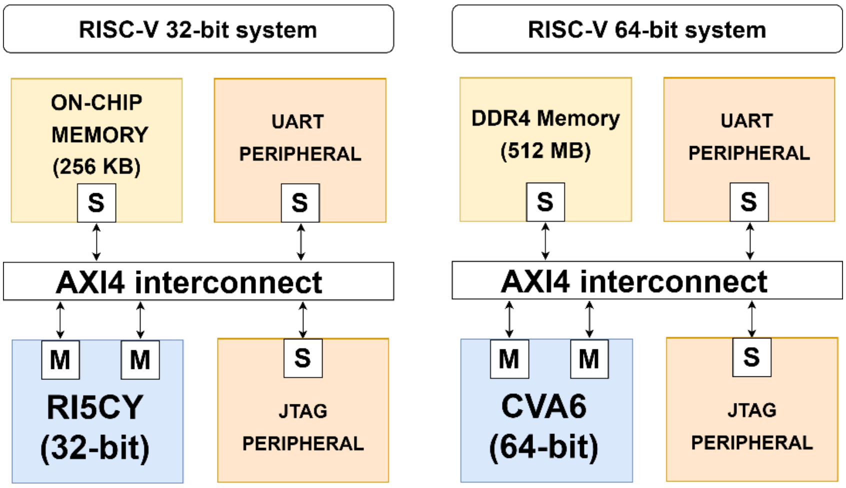


Figure 2-1: RISCV-based systems for benchmarking the SEAL-Embedded library.

Table 2-1 shows the benchmark results of the symmetric encryption function of the SEAL Embedded library on the selected CPUs.

Poly-Degree	Msg size	CVA6 (64-bit)	RI5CY (32-bit)
1024	2048 B	17.19 ms	207.10 ms
2048	4096 B	37.09 ms	444.22 ms
4096	8192 B	273.80 ms	2806.43 ms
8192	16384 B	1184.19 ms	--
16384	32768 B	5861.02 ms	--

Table 2-1: Benchmark results for the encryption function of the SEAL-Embedded Library. Column 1 indicates the selected polynomial degree for the RLWE encryption, column 2 indicates the message size in Bytes, column 3 shows the results for the CVA6 processor and column 4 the results for the RI5CY processor. Both CPUs run at 100 MHz.

Despite the SEAL-Embedded being targeted for resource-constrained devices, it cannot be successfully executed on the RI5CY CPU for polynomial degrees (Poly-Degree) higher than 4096 (256KB of memory are not enough).
In addition, the latency for the encryption process is extremely high: around 3 seconds are required to encrypt 8 KB with 4096 Poly-Degree. Hence, a hardware accelerator has been designed and implemented in FPGA technology.

2.2. Hardware accelerator IP design, interfacing to host processor and implementation results on FPGAs

The following are the main specifications of the hardware accelerator for homomorphic encryption:

- Hardware acceleration of the symmetric encryption function of the SEAL-Embedded library supporting all the security levels and polynomial degrees of the library.
- Encryption latency around hundreds of milliseconds;
- Standard AXI4 memory-mapped interface; desiderably a DMA interface.

The target for the hardware acceleration is the RLWE encryption function:

$$c_0 = -a \cdot s + m + e$$

Considering the overhead caused by polynomial multiplication ($a \cdot s$), in SE this operation is optimized utilizing the Number Theoretic Transform (NTT), computed following the Harvey Butterfly operations. In a polynomial multiplication between two (*n*−*1*) degree polynomials, evaluating the NTT of both allow to multiply their coefficient in a point-wise manner, reducing the complexity of the polynomial multiplication from $O(n^2)$ to $O(\mathrm{nlogn})$. NTT is a specialized form of the Discrete Fourier Transform except that operates over a prime field instead of a complex field. Given a polynomial $a = (a_0, a_1, \dots, a_{n-1}) \in R_q$ and the primitive root of unity ω , the NTT outputs a vector $A = (A_0, A_1, \dots, A_{n-1})$ following the equation:

$$A_i = \sum_{j=0}^{n-1} a_j \omega^{ij}$$

where $0 \le j \le n$. The multiplication between two polynomials *a* and *b* becomes:

$$a \cdot b = \left(1, \omega^{-1}, \dots, \omega^{-(n-1)}\right) \bullet NTT^{-1}(\, NTT(a) \bullet NTT(b)\,)$$

where • indicates the coefficient-wise multiplication. Operatively, the NTT is executed through butterfly-operations among polynomial coefficients and $(n-1)$ powers of a primitive root of unity ω (denoted as twiddle factors). The Harvey Butterfly configuration (reported in Figure 2-2) is used.

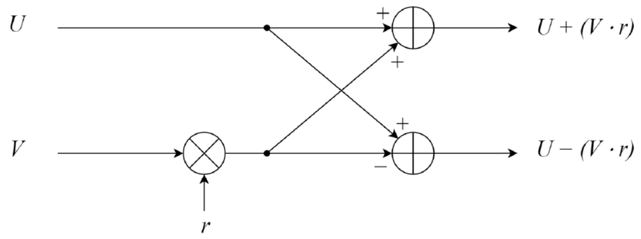


Figure 2-2: Harvey Butterfly configuration. The input *r* indicates the twiddle factors, U and V are the polynomial coefficients.

The proposed hardware accelerator aims to execute the symmetric encryption function, with the CKKS message encoding deferred to the software. The hardware-software partitioning can be seen in Figure 2-3.

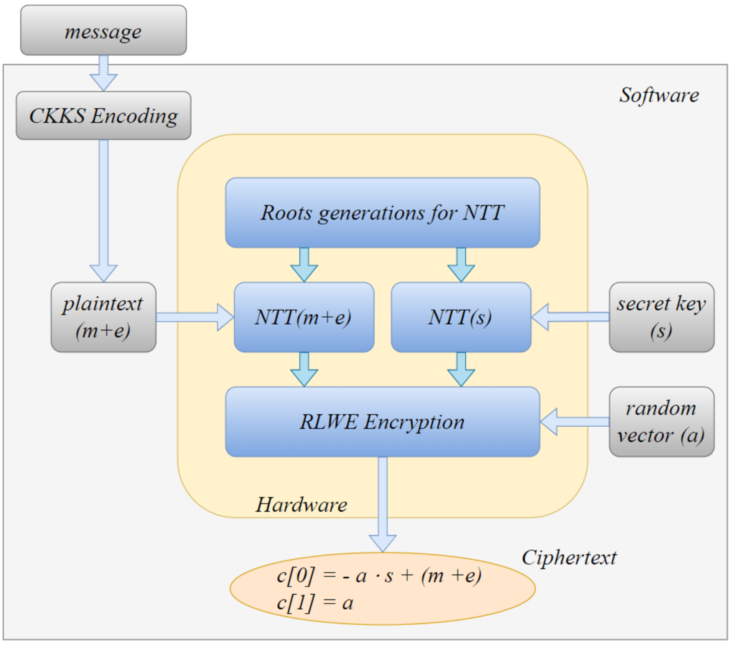


Figure 2-3: Hardware-software partitioning for the acceleration of symmetric encryption function of the SE library.

The hardware accelerator acquires the vectors *a, s, (m + e)* from the software and proceeds with the encryption, evaluating *NTT(s), NTT(m+e)* and performing all the polynomial operations to obtain the ciphertext. The architecture of the proposed hardware accelerator, that can be seen in Figure 2.4, features the following main blocks:

- Two dual-port RAMs (DPRAM1 and DPRAM2 in Figure 2-4): they are used to store the polynomial coefficients up to the maximum supported polynomial degree (i.e., *n* =16384) for SE. The size of these memories range from 8KB to 128KB depending on the maximum polynomial degree the hardware accelerator is able to support (from 1024 to 16384), which can be configured at synthesis time.
- A dual-port RAM (shared DPRAM in Figure 2-4): it is used as shared memory between the hardware accelerator and the main processor or DMA. Its size ranges from 4KB to 64KB (can be configured at synthesis time depending on the supported polynomial degree).
- An Arithmetic Logic Unit (ALU Butterfly in Figure 2-4): it performs the butterfly operations stage by stage and the polynomial operations needed to obtain the ciphertext.
- The Roots Generator module (Roots Generator in Figure 2-4): it computes and stores the roots of unity for the NTT.
- A finite state machine (ALU NTT Fsm in Figure 2-4) that manages the RLWE algorithm and the memory accesses.
- An AXI4 Slave interface (indicated as AXI Slave in Figure 2-4) that can be used to communicated with both the CPU and the DMA. The data size is 32-bit and the address size is 18-bit.

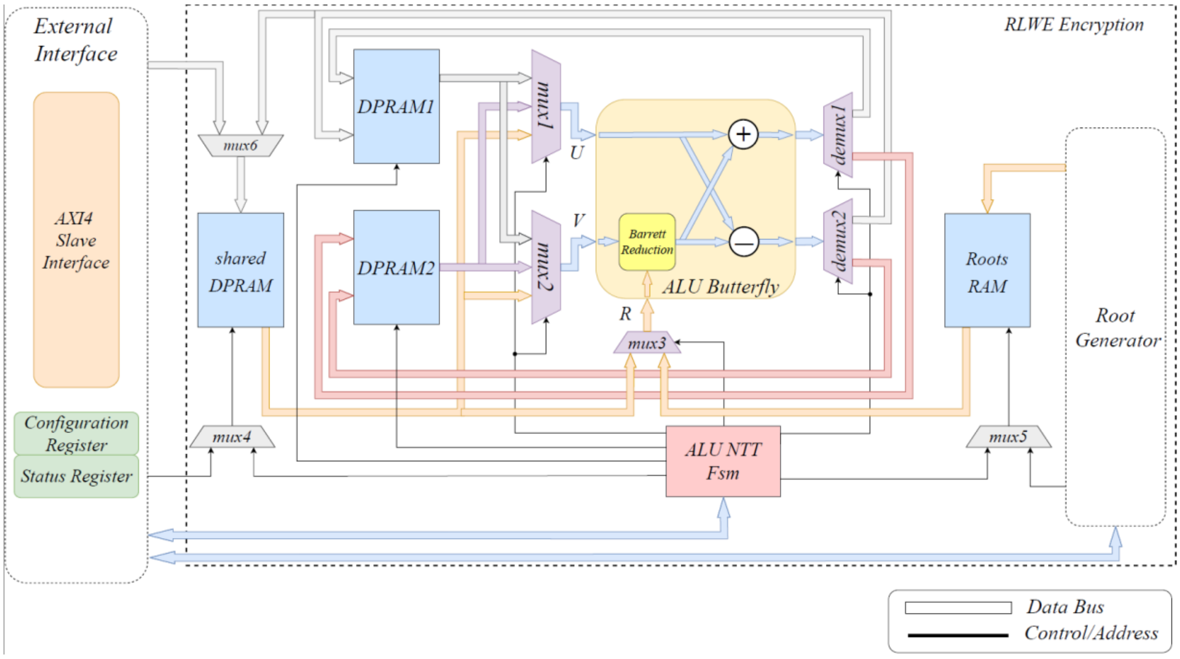


Figure 2-4: Hardware architecture of the accelerator for the SEAL-Embedded symmetric encryption function.

The hardware accelerator for SE has been implemented on the FPGA Board Zynq UltraScale+ MPSoC ZCU106 equipped with the System-on-Chip (SoC) XCZU7EV-2FFVC1156.

The block design implemented in the EDA tool Vivado 2020.2 is reported in Figure 2-5. The system includes the RI5CY processor (running at 100 MHz), the Xilinx Central DMA, an AXI4 interconnect logic and standard peripherals (UART, JTAG). The memory in this case is a 512MB DDR4 memory in order to be able to successfully run the SE encryption function for alle the polynomial degrees. Figure 2-6 pictures the prototyping environment with the host PC running Vivado 2020.2 tool and the FPGA board Xilinx ZCU106.

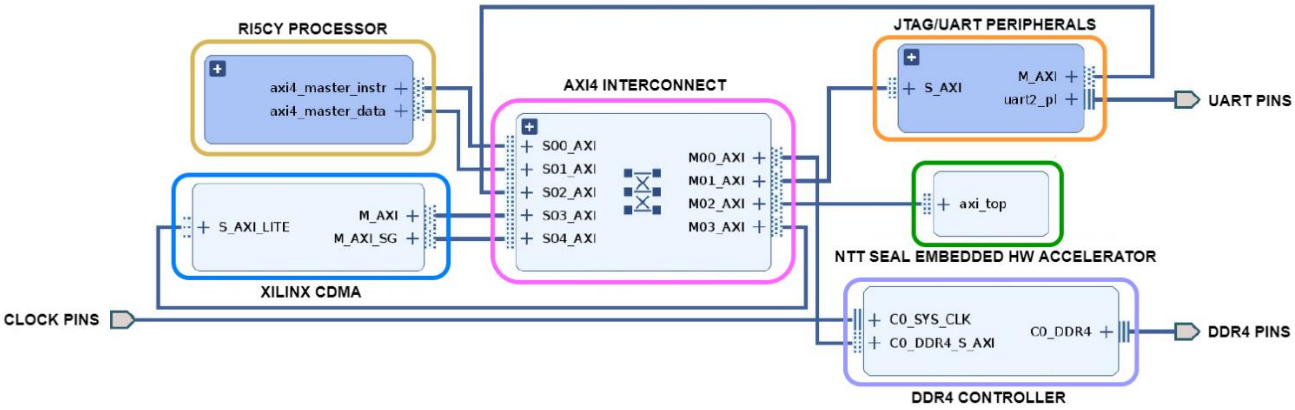


Figure 2-5: Block design implemented in Xilinx Vivado 2020.2 of the proposed system.

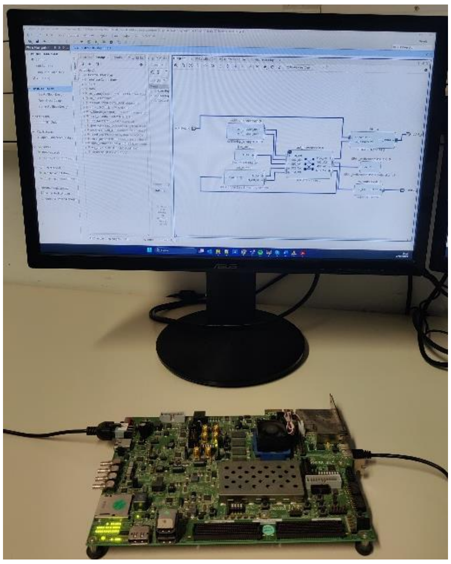


Figure 2-6: Prototyping setup: FPGA board Xilinx ZCU106 and host PC with Vivado 2020.2 (block diagram view of the system integrating the hardware accelerator for SEAL-Embedded library).

Table 2-2 reports a comparison between the software performance (third column) and the software plus hardware acceleration of the symmetric encryption function. In particular, the fourth column includes the results using the CPU as Master in the AXI4 communication instead of the fifth column that considers the Xilinx Central DMA as Master.

Table 2-3 shows the resources consumption on the target FPGA. The software results on the RI5CY processor are slightly different with respect to the benchmark previously presented since the interconnection network is different.

Polynomial Degree	Msg Size	SW (ms)	HW (ms)	HW DMA (ms)
1024	2048 B	168.35	7.84	0.142
2048	4096 B	364.53	15.68	0.297
4096	8192 B	2352.74	93.72	1.866
8192	16384 B	10032.13	374.83	7.79
16384	32768 B	45856.85	1624.12	35.19

Table 2-2: Performance results of the hardware accelerator for the symmetric encryption function of the SE library. The first column indicates the polynomial degree, the second the message size, the third indicates the execution time for a full encryption using the RI5CY processor, the fourth shows the result of the software plus hardware acceleration and the fifth software plus hardware acceleration with data transfers executed using the Xilinx Central DMA.

Module	Max. Frequency	CLB LUTs	CLB REGs	BRAMs	DSPs
Hardware accelerator	180 MHz	3742	1524	87	42
RI5CY	100 MHz	4487	2181	0	5
AXI4 Interconnect	100 MHz	24058	17882	80	0
AXI Central DMA	100 MHz	1221	2168	0	0

Table 2-3: Resources consumption of the proposed system on the (SoC) XCZU7EV-2FFVC1156.

As reported in Table 2-2, the performance improvement using the hardware accelerator ranges from around a 25x using the RI5CY processor for the data exchange, to around a 1200x using the DMA. Considering the case of polynomial degree 4096 (which is the typical use case), the encryption time with the software implementation is of the order of seconds, which is impractical for a real scenario; the adoption of hardware acceleration reduces this time to milliseconds.

3. Accelerator IP for XOFs SHAKE-128/256

3.1. eXtendable Output Functions (XOFs): SHAKE-128/256 and benchmarking

An eXtendable Output Function (XOF) is a variable-length HASH function in which the length of the output can be chosen to meet the requirements of individual applications.
The XOFs can be specialized to hash functions or used in a variety of other applications. The reference standard for the XOF is the NIST FIPS 202 [6], where two XOFs are specified: SHAKE-128 and SHAKE-256. NIST Post-Quantum algorithms for both Key Encapsulation Mechanism (KEM) and Digital Signature (DS) adopt the SHAKE algorithm: CRYSTALS-Kyber for KEM and CRYSTALS-Dilithium, FALCON and SPHINCS+ for DS. In particular, in DS algorithms the hardware acceleration of XOFs becomes crucial since XOFs are employed to generate the digest of the message to be signed/verified.

The eXtendable Output Functions (XOFs) SHAKE128/256 are described in the SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions [6]. Unlike SHA-3 functions, the output length of the SHAKE functions can be chosen arbitrarily to meet the requirements of individual applications. The SHAKE functions are based on the Keccak algorithm, which is a family of sponge functions; such functions are produced from an iterative approach called sponge construction described in [12], which employs three main components: a fixed-length permutation (or transformation) *f* that operates on a state of fixed number of bits *b* (width); a parameter *r* called rate; a padding rule named *pad*. The state is composed of *b = r + c* bits, where the value *c* is the capacity. The analogy to a sponge is due to the fact this class of functions “absorb” an arbitrary number of input bits into the state, and “squeeze” an arbitrary number of output bits out of the state. Figure 3-1 explains this procedure: initially, the input string of *N* bits is padded with a reversible padding rule *pad* and cut into blocks of *r* bits. Then, the *state* of the sponge function is initialized to zero, and the sponge construction proceeds in two phases, the absorbing and the squeezing phase. In the absorbing phase, each *r*-bit block of the input string (after padding and cutting) is first XORed with the first *r* bits of the *state* and interleaved with applications of the *f* function. When all input blocks are processed, the sponge construction switches to the squeezing phase. In this phase, the first *r* bits of the *state* are returned as output blocks, interleaved with applications of the function *f*. The number of output blocks is chosen by the user. The first *r* bits of the *state* corresponding to the rate are directly affected by the input blocks in the absorbing phase and are output during the squeezing phase. The last *c* bits of the state corresponding to the capacity are never directly affected by the input blocks and are never output during the squeezing phase.

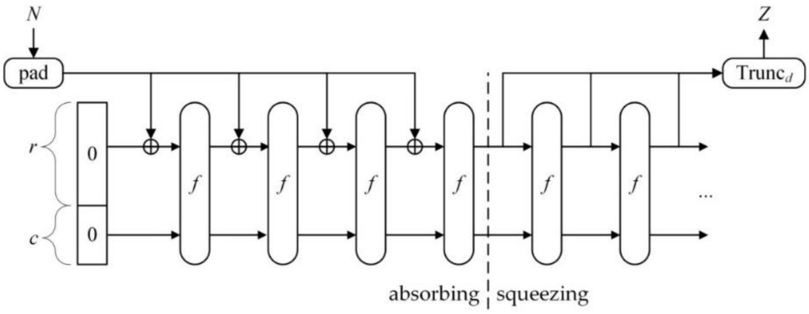


Figure 3-1: Sponge construction for the SHA3 family functions.

The parameters of the SHAKE algorithms and of the underlying Keccak sponge function are reported in Table 1.

Algorithm	*b*	*r*	*c*	Output length
SHAKE-128	1600	1344	256	Unlimited
SHAKE-256	1600	1088	512	Unlimited

Table 3-1: Parameters of the SHAKE functions

In Keccak, the underlying function f is a permutation function of 24 rounds, named Keccak-f[1600,24].

As reported before, SHAKE functions are employed as hash functions in Post-Quantum digital signature algorithms to generate the digest of the message to be signed/verified. Some IoT applications, for instance Firmware-Over-The-Air (FOTA) update, require verifying the DS of large messages (e.g., up to Gigabytes) with low latency. Next section will show the benchmark results of the SHAKE256 function running on both RISC-V processors and ARM-A53 CPUs, and the implementation results of a hardware accelerator for SHAKE128/256 functions.

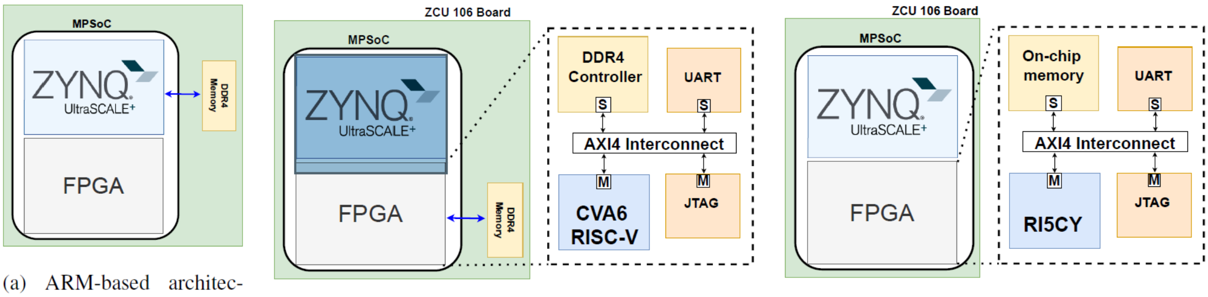
Performance evaluation of SHAKE256 function in Post-Quantum Digital Signature Algorithms

The Deliverable D2.4 reported the performance results of the verification function for the DS Post-Quantum algorithms CRYSTALS-Dilithium [17] and FALCON [18].

The results reported in the deliverable D2.4 showed that the bottlenecks of these algorithms rely on the SHAKE256 function. For this reason, a benchmark campaign to assess the performance of this function has been carried out on three different hardware architectures reported in Figure 3-2 and implemented on the Xilinx ZCU106 board with an MPSoc ZynQ Ultrascale+, which includes a multicore ARM Cortex-A53 processor and an FPGA:

1. The ARM-based architecture reported in Figure 3-2(a), which mounts an ARM Cortex-A53 equipped with 32KB L1 cache and 1MB L2 cache, which is included inside the MPSoc ZynQ Ultrascale+. The ZynQ Ultrascale+ equips the interface for DDR4 memory (2 GiB of memory), and the UART peripheral to communicate with the host PC. The clock frequency of the ARM Cortex-A53 is 1.2 GHz.
2. The CVA6-based architecture reported in Figure 3-2(b), which mounts a CVA6 RISC-V [9] processor implemented on the FPGA, equipped with 32 KiB L1 cache, and connected to 512 MiB of DDR4 memory and to standard peripherals (i.e. UART and JTAG) provided as Vivado hardware IPs. The clock frequency of the CPU and peripherals/interconnect is 100 MHz.
3. RI5CY-based architecture reported in Figure 3-2(c), again implemented through the ZCU106 board. Such an architecture consists in a 32-bit RISC-V RI5CY [8] processor (also named CV32E40P) implemented on the FPGA and running at 100 MHz.

We measured the execution time of the SHAKE256 function used as HASH function (the same adoption is done in Post-Quantum algorithms for DS), for different message sizes ranging from 100KiB to 1GiB.



(a) ARM-based architecture.

(b) CVA6-based architecture.

(c) RI5CY-based architecture.

Figure 3-2: Hardware Architectures used for the performance evaluation.

Table 3-2 reports the average computation time w.r.t. the message size on the three considered architectures.

Message length	SW-only SHAKE256 computation time [ms]		
	ARM-based arch.	CVA6-based arch.	RI5CY-based arch.
100KiB	1.43	86.47	3106.74
500KiB	7.13	432.13	16548.07
1MiB	14.60	884.96	34174.09
5MiB	73.10	4142.28	174287.85
10MiB	146.18	8848.53	--
100MiB	1460.44	88483.49	--
1GiB	14954.28	--	--

Table 3-2: Computation time for the SHAKE256 function on the three different hardware architectures (SW-only performance).

3.2. Hardware accelerator IP design, interfacing to host processor and implementation results on FPGAs

Figure 3-2 depicts the architecture of the hardware accelerator for SHAKE128/256 functions. The hardware accelerator includes an AXI4 Memory Mapped Slave interface with a 32/64-bit data bus, a 1344-bit shift register for input/output from/to the AXI Master interface, and dedicated registers for configuration and status. The SHAKE core (reported in the blue box of Figure 3-2) contains the hardware logic that actually executes the SHAKE algorithm following the sponge construction. In the absorbing phase, the hardware padder processes the input message that is held in the Data register. This message is XORed with either the all-zero initial state (when it is the first block of the message) or the output of the Keccak function. The Keccak function is implemented using a 1600-bit state register and combinational logic to perform the processing on the state register. This processing involves a permutation function that is composed of a sequence of five transformations and reported in figure 3-2 as θ, ρ, π, Χ, and ι. The subsequent squeezing phase involves truncating the state register and executing the Keccak function until the required number of output blocks has been produced, if multiple output blocks are needed. The Data register is reused to send data to the AXI4 Master interface.

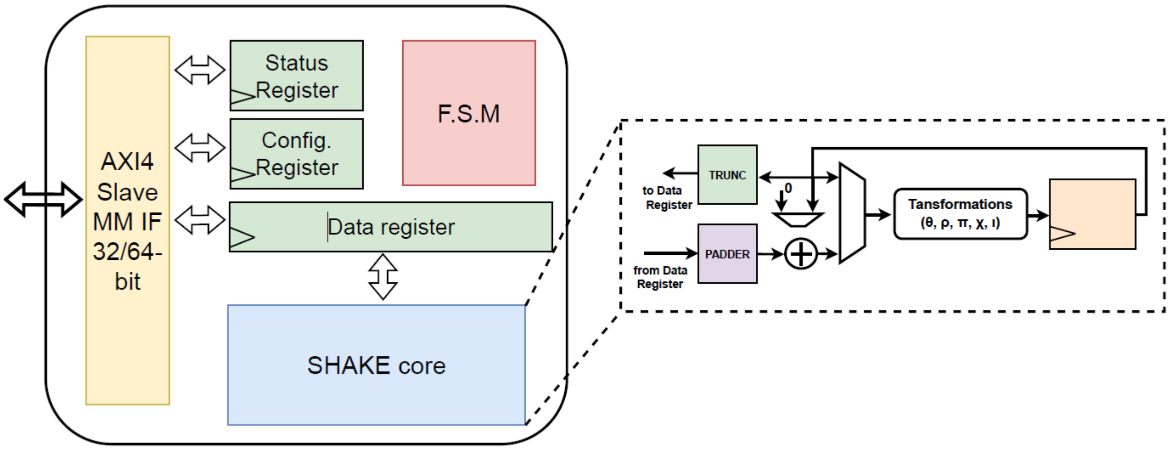


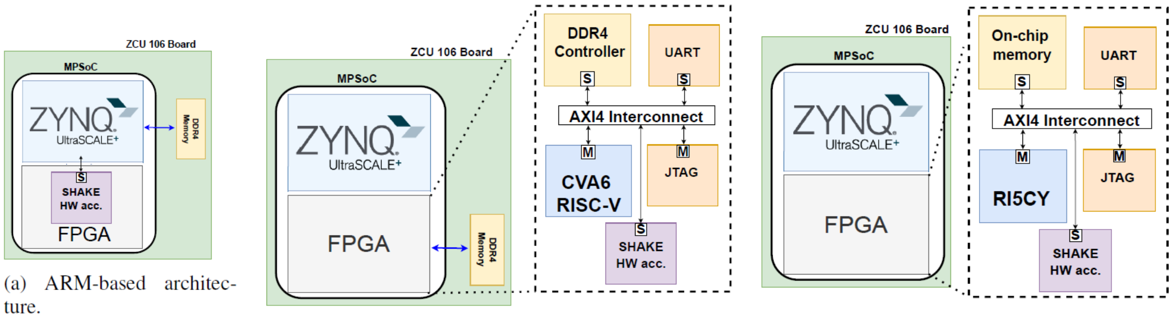
Figure 3-2: Hardware architecture of the accelerator for SHAKE128/256 functions.

Table 3-3 shows the synthesis results for the proposed hardware accelerators on the FPGA of the Xilinx ZCU106 board.

Max. frequency	CLB LUTs	CLB REGs	BLOCKs RAM	DSPs
333 MHz	6172	3327	0	0

Table 3-3: Resource consumption of the hardware accelerator for SHAKE128/256 functions.

The hardware accelerator for SHAKE functions has been integrated into the three architectures considered for the benchmark campaign, as reported in Figure 3-3. In the case of the ARM-based architecture, the accelerator is connected to the ARM-A53 processor through a 64-bit AXI4 Master Full Power Domain (FPD) interface that is implemented on the FPGA. Both the interface and the accelerator run at 300 MHz, which is the maximum frequency of the AXI4 Master interface. In both the CVA6-based and RI5CY-based systems the hardware accelerator has been connected to the CPUs through the AXI4 interconnect, in the former case using a 64-bit interface and in the latter with a 32-bit one. Figure 3-4 pictures the prototyping environment with the host PC running Vivado 2020.2 tool and the FPGA board Xilinx ZCU106.



(a) ARM-based architecture.

(b) CVA6-based architecture.

(c) RI5CY-based architecture.

Figure 3-3: Hardware Architectures used for the performance evaluation.

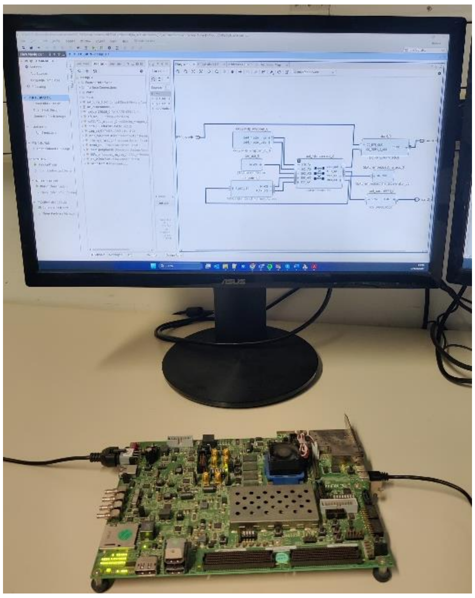


Figure 3-4: Prototyping setup: FPGA board Xilinx ZCU106 and host PC with Vivado 2020.2 (block diagram view of the system integrating the hardware accelerator for SHAKE128/256 algorithms).

Table 3-4 reports the performance results of the hardware accelerator for the computation of the SHAKE256 function w.r.t. different message size, on the three different architectures.

Message length	HW-accelerated SHAKE256 computation time [ms]		
	ARM-based arch.	CVA6-based arch.	RI5CY-based arch.
100KiB	0.25	8.12	20.96
500KiB	1.24	40.49	104.80
1MiB	2.52	82.92	214.63
5MiB	12.78	414.28	1073.15
10MiB	25.57	828.56	--
100MiB	253.16	8289.25	--
1GiB	2617.80	--	--

Table 3-4: Computation time for the SHAKE256 function (executed in HW) on the three different hardware architectures.

The speedup on the ARM-based architecture is around 5.7x, on the CVA6-based architecture of around 10.5x and of around 160x on the RI5CY-based architecture.

4. Differences with previous Deliverable

With respect to the Deliverable D2.4, this document reports the following advancement:

- The description of the symmetric encryption function of the SEAL-Embedded library (Section 2.1) has been improved and expanded.
- The description of the hardware accelerator for the SEAL-Embedded library (in Section 2.2) has been extended and improved. The hardware architecure of the SoC implemented on FPGA has been modified and simplified. The results in terms of resource consumption and maxinum frequency of the hardware accelerator have been updated. Overall, the level of verification and validation of the developed IP has been improved.
- Section 3.1 has been heavely modified. The description of the SHAKE functions has been improved and expanded. The benchmark campaign has been specialized to the SHAKE256 algorithm. The hardware platform based on the RI5CY processor has been added for the benchmark campaign. The description of the benchmark platforms has been improved and the results have been updated.
- Section 3.2 has been updated according to Section 3.1: benchmark campaign specialized to SHAKE256 algorithm, new hardware platform, updated benchmark results. Overall, the level of verification and validation of the developed IP has been improved.

5. Conclusions and IP repository

This document has reported the activities done by TEXTAROSSA partner CINI (UNIPISA) for HDL design, verification, and synthesis results in FPGA technology for cryptographic IPs. Consolidated specifications of such IPs are reported in Deliverable D2.1, and preliminary results in Deliverable D2.4.

In Section 2 it has been addressed the design, interfacing, FPGA implementation and verification of a hardware accelerator for Homomorphic Encryption. The repository for this IP (HE_SEAL_Embedded_HW_IP) is:

https://drive.google.com/drive/folders/1UxmG3t8YLPV6oRAi0B9zwMZK13cAjAET?usp=sharing.

In Section 3 it has been addressed the design, interfacing, FPGA implementation and verification of a hardware accelerator for eXtendable Output Functions (XOF) SHAKE-128/256, a new hashing functions used within Post-Quantum algorithms like CRYSTALS -Dilithium, recently standardized by NIST [7] for digital signature generation/verification. The repository for this IP (SHAKE_HW_Accelerator_IP) is:

https://drive.google.com/drive/folders/1V07qDMeCX8IdoJrVOXxuQuj82D_gl_a-?usp=sharing.

The two IPs have been provided with a standard memory-mapped AXI4 interface, so that they can be easily interfaced to any other digital IPs. To this aim, results of the integration of the 2 proposed secure IPs with open-source RISC-V 32bits [8] and RISC-V 64bits [9] processors have been also presented.

Implementation results in Xilinx FPGA technology (Zynq UltraScale+ MPSoC ZCU106 equipped with the System-on-Chip (SoC) XCZU7EV-2FFVC1156) are also shown in both Sections 2 and 3.

Impact and dissemination

The proposed IPs are interesting, also in view of synergies between TEXTAROSSA and the other initiatives like EPI SGA2 and the European Pilot, since all the proposed accelerators can be integrated with RISC-V computing cores like the RISC-V in the EPAC (European Processor Accelerator).

Moreover, the two IPs can be also used as starting point to enrich in EPI SGA2 the hardware security module, called Crypto-Tile, initially developed in EPI SGA1 and missing PQC and HE capability.

This synergy is useful also since in TEXTAROSSA grant agreement is not foreseen a specific system-level application using PQC, which instead is part of the UNIPI work in EPI SGA2.

Moreover, TEXTAROSSA is just considering FPGA target with the aim of having FPGA acceleration while in projects like EPI2 an ASIC realization is foreseen.

The results about the homomorphic accelerator have been published with the paper on the journal IEEE ACCESS entitled:

S. D. Matteo, M. L. Gerfo and S. Saponara, "VLSI Design and FPGA Implementation of an NTT Hardware Accelerator for Homomorphic SEAL-Embedded Library," in IEEE Access, vol. 11, pp. 72498-72508, 2023, doi: 10.1109/ACCESS.2023.3295245.

Available as Gold Open Access at this link https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10182260

The results about the SHAKE accelerator have been submitted to the journal IEEE Access:

P. Perazzo, S. Di Matteo, G. Dini and S. Saponara, “On Hardware Acceleration of Quantum-Resistant FOTA Systems in Automotive”.

6. References

[1] D2.1, “Consolidated specs of accelerators IPs”, TEXTAROSSA project, revised, May 2023.
[2] D2.4, “eXtreme Secure Crypto IP”, TEXTAROSSA project, revised, May 2023.
Chen, H., Laine, K., & Player, R. (2017). Simple Encrypted Arithmetic Library - SEAL v2.1. Financial Cryptography Workshops.
[3] Natarajan, D., & Dai, W. (2021). SEAL-Embedded: A Homomorphic Encryption Library for the Internet of Things. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(3), 756–779. https://doi.org/10.46586/tches.v2021.i3.756-779.
[4] https://github.com/microsoft/SEAL-Embedded.
[5] https://github.com/Microsoft/SEAL
[6] M. Dworkin, “Sha-3 standard: Permutation-based hash and extendable output functions,” 2015-08-04 2015.
[7] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization
[8] https://github.com/pulp-platform/pulpino.
[9] https://github.com/openhwgroup/cva6
[10] PALISADE. https://gitlab.com/palisade. New Jersey Institute of Technology (NJIT).
[11] Halevi, S., & Shoup, V. (2020). HElib design principles. Tech. Rep.
[12] B. Guido, D. Joan, and P. Michaël, “Cryptographic sponge functions,” 2011.
[13] https://github.com/Microsoft/SEAL.
[14]https://www.intel.com/content/www/us/en/developer/tools/homomorphic-encryption/overview.html.
[15] https://developer.nvidia.com/blog/federated-learning-with-homomorphic-encryption
[16] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23 (pp. 409-437). Springer International Publishing.
[17] https://pq-crystals.org/
[18] https://falcon-sign.info/