TYRCA: A RISC-V Tightly-coupled accelerator for Code-based Cryptography

Alessandra Dolmeta [1,2], Stefano Di Matteo [2,3], Emanuele Valea[3], Mikael Carmona[2], Antoine Loiseau[2], Maurizio Martina[1], Guido Masera[1]

1 Politecnico di Torino, DET - Dipartimento di Elettronica e Telecomunicazioni , Turin, Italy, 2 Université Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France, 3 Université Grenoble Alpes, CEA, List, F-38000 Grenoble, France

Finanziato dall'Unione europea

Introduction

- Quantum computers threaten classical cryptosystems like RSA and ECC, prompting the need for **Post-Quantum Cryptography** (PQC). Among PQC candidates, **HQC** has been selected for standardization (*March 2025*).
- It is a strong alternative but suffers from inefficient implementation. Indeed, its standardization process is not only about the development of the mathematical models but has also given rise to a broad spectrum of research, spanning to software and **hardware**.

Background

HQC is a Key Encapsulation Mechanism (KEM), based on the syndrome decoding problem on structured **codes**, using two linear codes.

- It consists of three functions:
  - KeyGeneration
  - Encapsulation
  - Decapsulation

  HQC has three security levels [1].

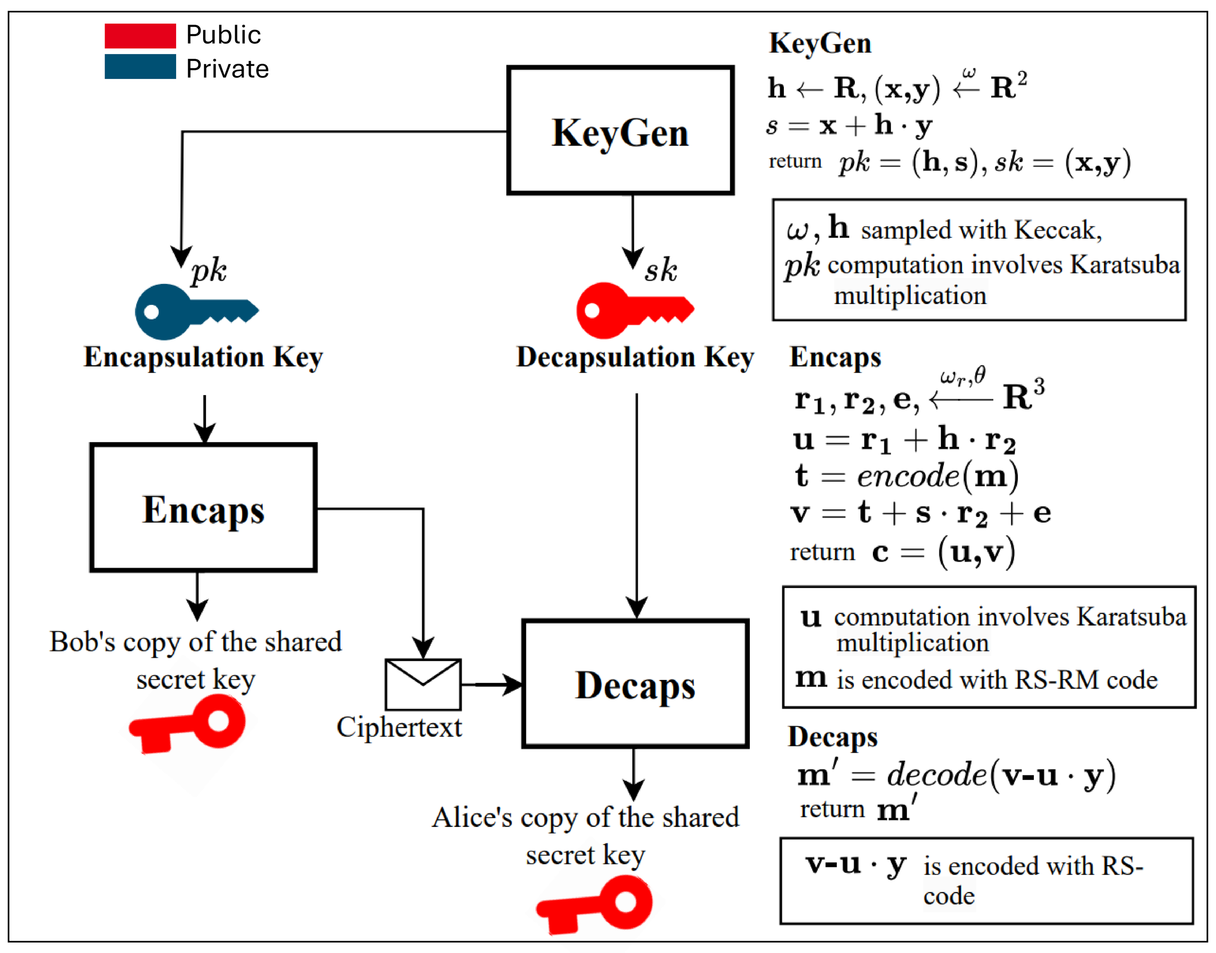

Figure 1. Key Encapsulation Mechanism.

- HQC-128 profiled on a 32-bit SoC (CV32E40PX core, using –O2 optimization flag). Our analysis shows that polynomial multiplication dominates execution time (> 95%), while RS-RM encoding/decoding creates bottlenecks in encapsulation and decapsulation. Keccak, though minimal, is consistently used throughout the algorithm.
- **Integrating Accelerators into RISC-V.** There are three possible integration methods: loosely-coupled, tightly-coupled and coprocessors. **Coprocessors** are usually connected via a dedicated interface, enabling higher flexibility and access to external registers.

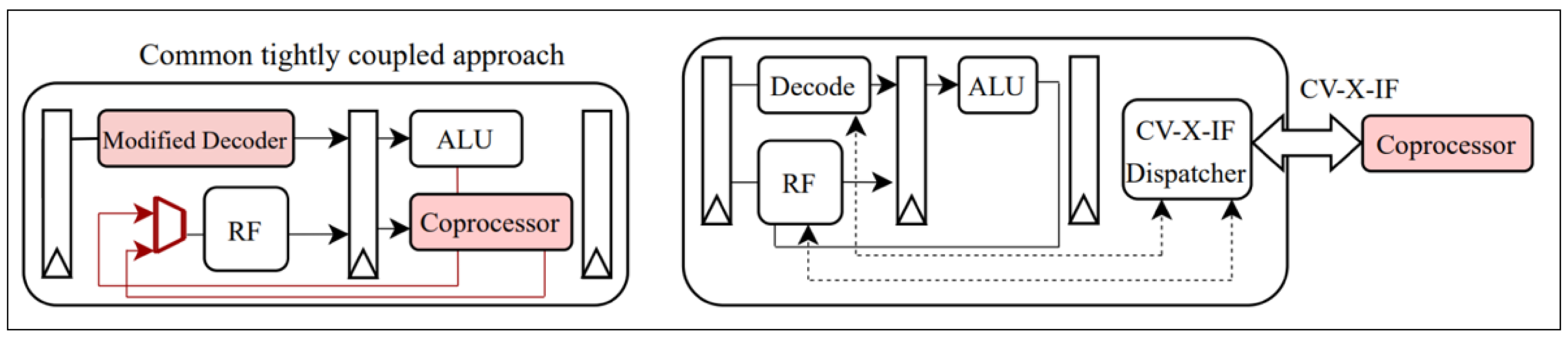

Figure 2. Common tightly approach vs. CV-X-IF.

- CV-X-IF [2] **adds custom instructions** without modifying the CPU decode unit, exploiting unused opcodes to trigger TYRCA. It ensures:
  - **low-latency register access**
  - **external extension support**
  - **synchronous execution**.

TYRCA

- We chose an adaptive platform that includes a **CV32E40PX** [3] core, allowing us to leverage the CV-X-IF. The main elements are: TYRCA, the RISC-V core, instruction and data memories, the JTAG module, the UART, an OBI bus.
- **TYRCA** architecture is mainly composed of:
  - ❑ the CV-X-IF controller
  - ❑ various accelerators ($\mathcal{R}$-Unit, RS-decoder, Keccak).
- Each of these accelerators is customized to execute one or more instructions.

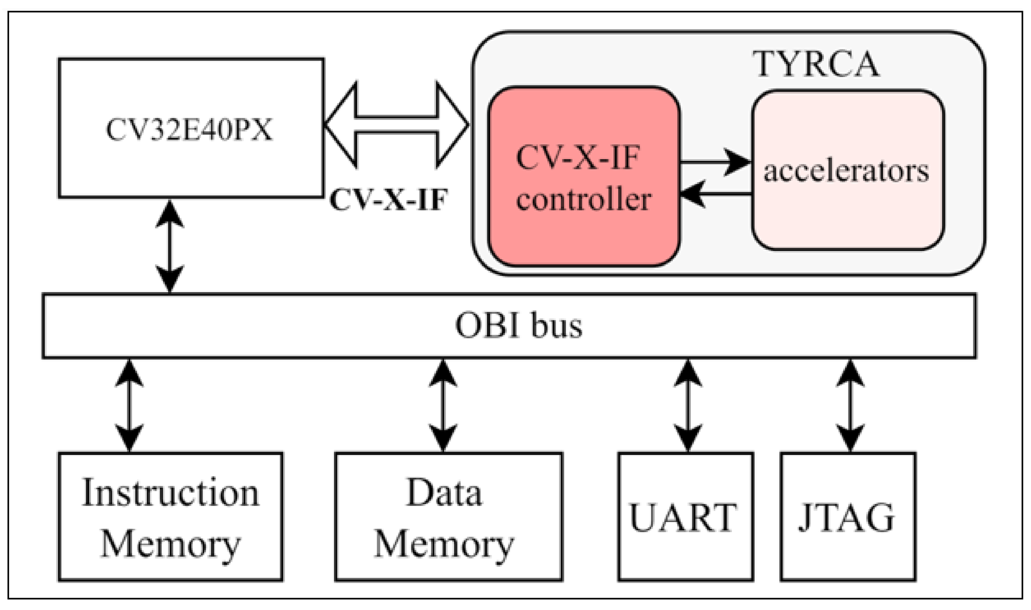

Figure 3. System on Chip architecture.

- The CV-X-IF is connected directly to the register file of the CPU. **Inline assembly** is used; by specifying function codes and operands, these instructions execute specialized operations on TYRCA.

- **$\mathcal{R}$-Unit.**
  - From 64-bit to 32-bit integer **Karatsuba multiplication** implementation.
  - *1 custom instruction* → break each 64-bit in two 32-bit chunks.
  - One carry-less multiplier, few registers, XOR-operations logic, and the logic to manage inputs, store intermediate results.

Karatsuba (A, B):

karats res_0, A_0, B_0
karats res_3, A_1, B_1
karats res_2, x_0, x_0
karats res_1, x_0, x_0

Code. Karatsuba structure.

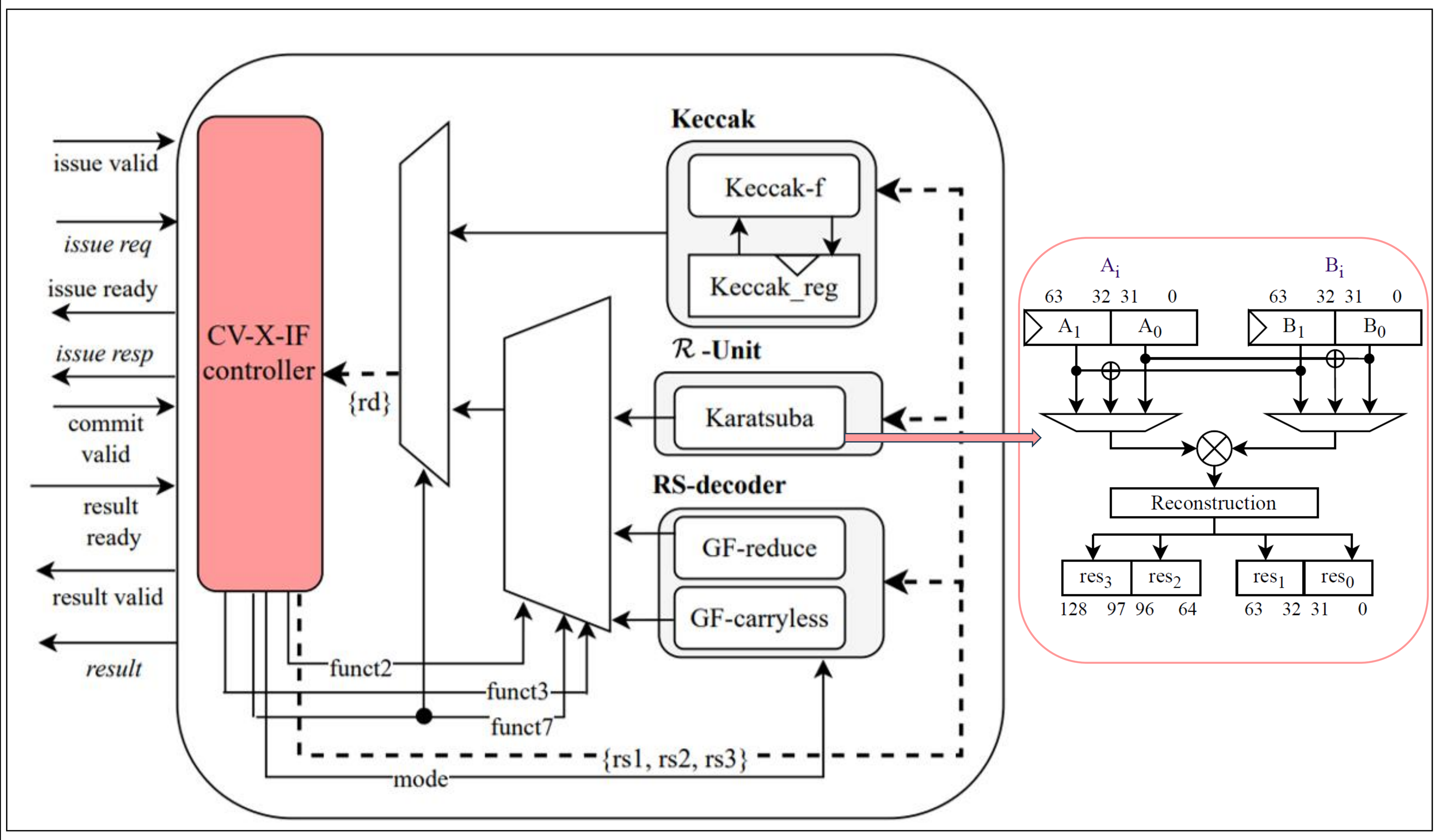

Figure 4. TYRCA architecture.

- **Reed-Solomon (RS) decoder.** To enhance performance, we accelerated critical Galois Field (GF) arithmetic operations:
  - ❑ *GF-reduce* - polynomial reduction via shift/XOR (*3 custom insn*).
  - ❑ *GF-carryless* - multiplication of 8-bit polynomials (*4 custom insn*).
- **Keccak.** Keccak processes 1600-bit data in 24 rounds. To reduce load/store overhead, it has a dedicated register and *3 custom insn*:
  - *store* – uploads state matrix (64-bit at a time).
  - *start* – initiates 24-round processing.
  - *load* – saves results (32-bit at a time).

Results & Conclusions

TYRCA is implemented at the RTL using SystemVerilog and deployed on the **Xilinx Kintex-7 FPGA**, specifically on the Digilent Genesys 2 board. Synthesis and Place&Route are performed using Xilinx Vivado.

Function	SW	SW + TYRCA	Calls	Speed-up (%)
Karatsuba	6,072	52	59,142	99.14
GF-carryless	300	56	4,998	81.33
GF-reduce	1,660	145	5,589	91.27
Keccak-f	26,825	2,538	143	90.54

Table 1. Performance Improvement Comparison [clock cycles]. Calls indicates the number of calls done to the different function in HQC-128, while speed-up is the ratio between SW and SW+TYRCA.

HQC	Version	KeyGen	Encaps	Decaps
hqc-128	SW	66,029,999	133,331,422	208,550,842
	TYRCA	3,108,737 [-95.29%]	6,302,661 [-95.27%]	11,095,034 [-94.68%]
hqc-192	SW	195,307,650	392,977,384	600,490,993
	TYRCA	10,847,737 [-95.22%]	22,578,534 [-95.27%]	34,895,339 [-94.97%]
hqc-256	SW	357,245,737	719,137,771	1,106,009,231
	TYRCA	20,153,688 [-95.06%]	41,469,541 [-95.13%]	64,999,457 [-94.82%]

Table 2. Clock cycles and improvement results (SW is HQC from version round 4 – 2024 [1])

	LUT	Registers
SoC Top-Level	33,701	21,806
TYRCA	8,894	3,710
• CV-X-IF Controller	75	163
- *Keccak*	5,418	1,628
- *$\mathcal{R}$-Unit*	905	255
- *RS-decoder*	222	0

Table 3. Resource Utilization on FPGA.

This underscores TYRCA's potential in robust PQC.

Reference

[1] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.- C. Deneuville, P. Gaborit, E. Persichetti, G. Zemor, and I. Bourges, "Hamming Quasi-Cyclic (HQC) Fourth Round Version." Online, 2024. Updated version 23/02/2024.
[2] https://github.com/openhwgroup/core-v-xif/tree/main
[3] https://github.com/esl-epfl/cv32e40px