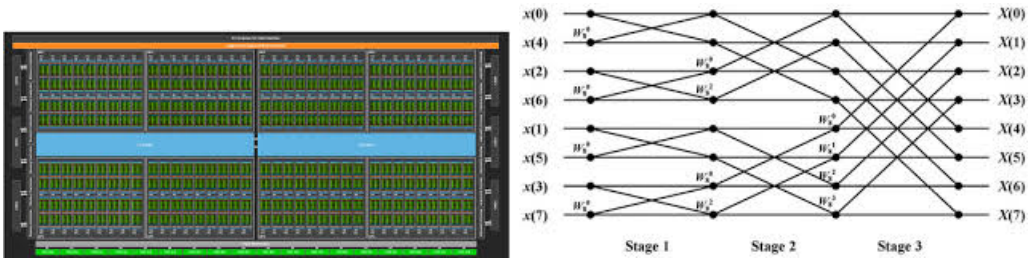
GPU Acceleration of Cryptographic Primitives in Encrypted Compute



Imagine a hospital that wants to use a cloud-based machine learning service to diagnose medical images. For patient privacy, the hospital cannot simply upload raw patient scans. Ideally, the hospital would encrypt the images, send the encrypted data to the cloud, have the cloud run the ML model directly on the ciphertext, and receive a result that only the hospital can decrypt.

This is the promise of Fully Homomorphic Encryption (FHE). It allows a third party to perform arbitrary computations on encrypted data without ever accessing the decryption key. Given an encryption function Enc, a decryption function Dec, and some computation f, a fully homomorphic scheme guarantees that $\mathrm{Dec}\big(\mathrm{Eval}(f,\ \mathrm{Enc}(x))\big) \ = \ f(x)$.

However, FHE in practice is extremely slow. A single homomorphic multiplication can be four to six orders of magnitude slower than its plaintext equivalent. FHE operates on very large polynomials, typically of degree $N = 2^{15}$ or 2^{16}, with coefficients defined modulo a large integer Q that can be hundreds of bits wide. Every high-level operation decomposes into low-level primitives on these polynomials: transforms, modular arithmetic, basis conversions, and key-switching steps. These primitives are massively parallel in nature, making GPUs a natural fit.

One example of a critical primitive is the Number Theoretic Transform (NTT), closely related to a Discrete Fourier Transform (DFT). FHE operates in the polynomial ring $R_Q = \mathbb{Z}_Q[X] \,/\, (X^N + 1)$, where elements are polynomials of the form $a(x) = \sum_{j=0}^{N-1} a_j X^j$. Naïvely, multiplying two degree-$(N-1)$ polynomials costs $O(N^2)$ operations. The NTT reduces this to $O(N \log N)$ by evaluating polynomials at roots of unity ψ^k in $\mathbb{Z}_q$ (integers modulo a prime q):

$$\hat{a}_k \ = \ \sum_{j=0}^{N-1} a_j \cdot \psi^{jk} \pmod{q}.$$

The fast algorithm follows the same butterfly decomposition as the FFT. Next to NTT, FHE relies on other primitives in need of optimization like key switching, basis conversion, or decomposition.

The student will begin with a literature study of FHE primitives and existing GPU implementations. Based on this survey, one or more primitives will be selected and a targeted optimization will be designed, implemented in CUDA, and evaluated against state-of-the-art baselines.

Practicalities

Promotor:	Prof. Dr. Ir. Ingrid Verbauwhede		
Daily supervisor:	Henri Vermeersch	henri.vermeersch@kuleuven.be	office 01.23
	Thomas de Ruijter	thomas.deruijter@kuleuven.be	office 01.17
Nature of the work:	40% literature, 10% theoretical work, 50% software		
Number of students:	1		

 © 2026 – 2027 COSIC