Secure Hardware Accelerators for Post Quantum Cryptography

Timo Zijlstra (supervisors: Arnaud Tisserand, Karim Bigou)

1. Public Key Encryption in Post Quantum Setting

Alice:

1. Generate secret key
2. Generate public key
3. Send public key → (public key)
4. Decrypt ciphertext ← (ciphertext)

Bob:

1. Generate plaintext
2. Encrypt plaintext
3. Send ciphertext

- Quantum computer will break today's crypto → use Post Quantum Cryptography (PQC)

2. Learning With Errors (LWE) based Cryptography

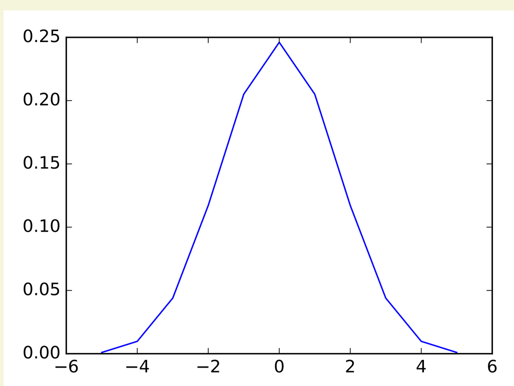
Ring-LWE (RLWE)

- Public keys: polynomials $\mathbf{a}(x), \mathbf{b}(x)$ of degree $< n$
  - $n = 1024$
  - 14-bit coefficients
- Encryption/decryption:
  1. Sample from binomial distribution
     - Generate random bits → use Pseudo Random Number Generator (PRNG)
     - Compute samples
  2. Polynomial arithmetic
     - Multiplication
     - Addition/subtraction

Module-LWE (MLWE)

- Public keys: $\begin{pmatrix} \mathbf{a}_{0,0}(x) & \mathbf{a}_{0,1}(x) \\ \mathbf{a}_{1,0}(x) & \mathbf{a}_{1,1}(x) \end{pmatrix}, \begin{pmatrix} \mathbf{b}_0(x) \\ \mathbf{b}_1(x) \end{pmatrix}$
  - $n = 256$
  - Vectors of size $k \in \{2, 3, 4\}$ with polynomial coefficients
- Encryption/decryption:
  - Similar to RLWE
  - Matrix/vector operations

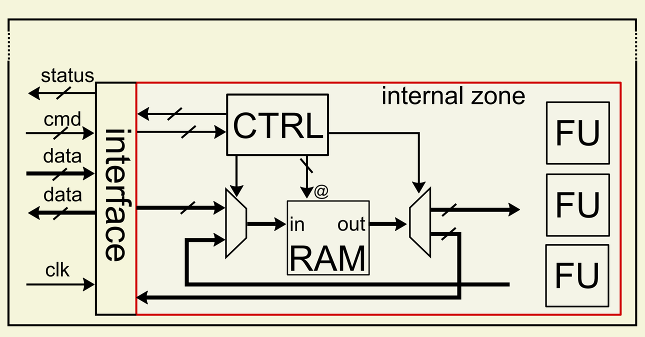
Figure: Binomial distribution



3. PKE Algorithm

- *Key generation.* Let $\mathbf{s} \xleftarrow{\$} \mathcal{B}_\lambda(\mathcal{R}_q^k)$, $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathcal{R}_q^{k\times k})$ and $\mathbf{e}_0 \xleftarrow{\$} \mathcal{B}_\lambda(\mathcal{R}_q^k)$ and compute $\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e}_0$.
  - Private key: $\mathbf{s}$
  - Public key: $(\mathbf{A}, \mathbf{b})$
- *Encryption.* Let the plaintext $\mathbf{m} \in \mathcal{R}_q$ be a polynomial with binary coefficients. Sample $\mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_\lambda(\mathcal{R}_q^k)$ and $\mathbf{e}_3 \xleftarrow{\$} \mathcal{B}_\lambda(\mathcal{R}_q)$. Compute $\mathbf{c}_1 \leftarrow \mathbf{A}^\intercal \mathbf{e}_1 + \mathbf{e}_2$ and $\mathbf{c}_2 \leftarrow \mathbf{b}^\intercal \mathbf{e}_1 + \mathbf{e}_3 + \left\lfloor \frac{q}{2} \right\rceil \cdot \mathbf{m}$.
  - Ciphertext: $(\mathbf{c}_1, \mathbf{c}_2)$
- *Decryption.* Let $\mathbf{d} \leftarrow \mathbf{c}_2 - \mathbf{c}_1^\intercal \mathbf{s}$. For each coefficient of $\mathbf{d}$, decode to 0 if it is closer to 0 than to $\left\lfloor \frac{q}{2} \right\rceil$, else decode to 1.

4. Cryptographic Accelerator on FPGA

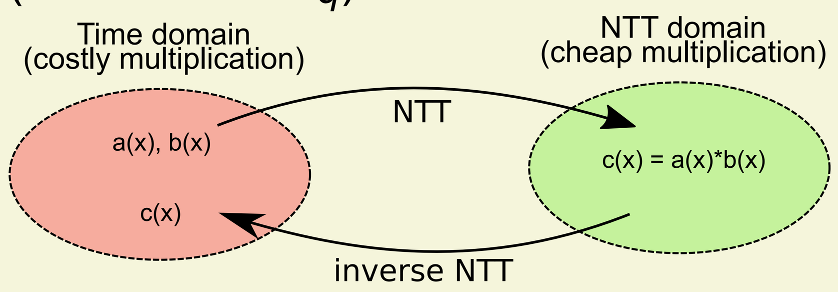


- Intensive computations → use hardware acceleration
- FPGA with HLS design

5. Computations in LWE Cryptography

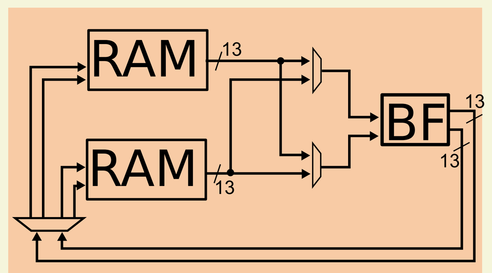
Arithmetic in:

- $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ (integers mod q)
  - Modular multiplication
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n + 1)$ (polynomials)
  - Number Theoretic Transform (FFT over $\mathbb{Z}_q$)



- $\mathcal{R}_q^k$
  - Matrix-vector multiplications
  - Vector additions/subtractions

Figure: NTT architecture



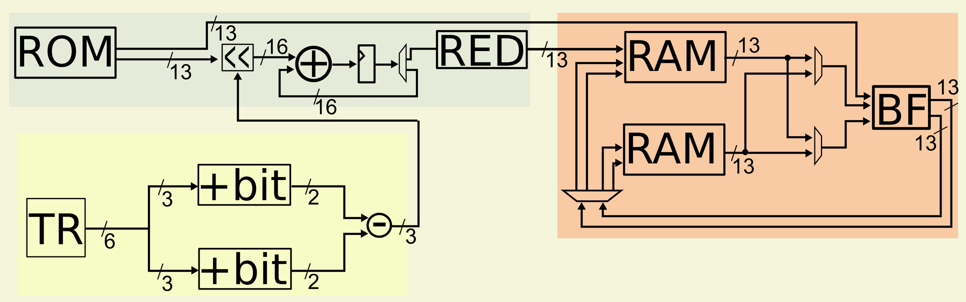
Parameters for Kyber, NewHope

- $q = 7681$
- $n = 256$ or 1024
- $k \in \{1, 2, 3, 4\}$

NTT complexity: $\frac{n}{2} \log n \approx 1000$ multiplications in $\mathbb{Z}_q$

6. Architecture of the Accelerator

Figure: BN unit

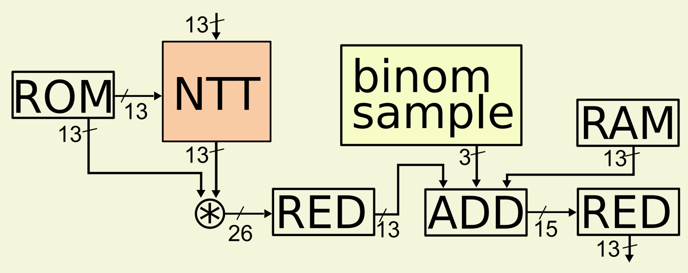


- Yellow: binomial samples
- Green: NTT pre-processing
- Red: NTT

Notations

- TR: Trivium PRNG
- BF: Butterfly operator
- RED: Mod function

Figure: EM unit

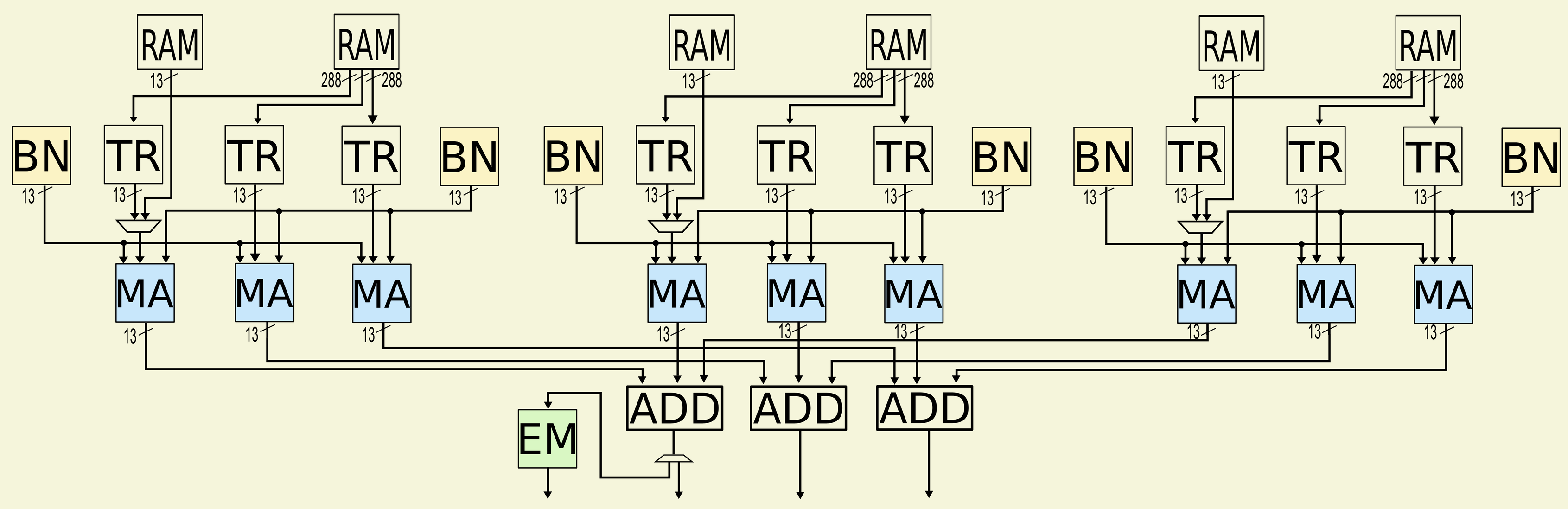


7. Architecture of MLWE Encryption

- Computation of $\begin{pmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{pmatrix} \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \end{pmatrix}$
- 4 polynomial multiplications (PM)

1. Sequential: time $= 4\times$ time$_{PM}$
2. Parallel: time $= 1\times$ time$_{PM}$

Figure: Parallel MLWE architecture ($k = 3$)



8. FPGA Implementation Results on Artix-7 200t