Layer 2: Homomorphic Lattice-Based Transformation (MLWE)

Overview

Layer 2 implements a post-quantum secure MLWE-based encryption and homomorphic processing layer using Kyber-style parameters. This report covers:

- MLWE hardness and lattice setting
- Centered-binomial sampling (χₑₜₐ)
- Polynomial ring arithmetic (convolution + NTT skeleton)
- Message encoding/decoding with repetition and checksum
- Key generation, encapsulation, decapsulation protocols
- Additive homomorphism and multiplication placeholder
- Security, performance, and integration insights

1. MLWE and Kyber Parameters

We work in the ring $R_q = ?\mathbb{Z}_q[X]/(X^n + 1)$
with:

$$n = 256,\ q = 3329,\ \chi_\eta:\ \text{centered-binomial}\ (\eta = 2),\ \text{module dimension}\ k = 2.$$

1.1 Hardness

Approximate MLWE problem:

$$\text{Given}\ (A, t = As + e)\ \text{mod}\ q,\ extrecovers, e$$

is as hard as worst-case lattice problems (SIVP, SVP) in dimension nk[110].

2. Centered-Binomial Sampler (χₑₜₐ)

For each coefficient:

1. Draw 2η bits uniformly.
2. Compute Hamming weights:

$$u = \sum_{i=1}^{\eta} b_i,\ v = \sum_{i=\eta+1}^{2\eta} b_i,$$

3. Output $u - v \in [-\eta, \eta]$.

This approximates a discrete Gaussian with variance η/2.

3. Polynomial Arithmetic

3.1 Convolution-based Multiplication

For polynomials $a, b \in R_q$:

1. Compute convolution $c = a * b$ in $extlength2n - 1$.
2. Pad to length $2n$ and fold via $X^n = -1$:

$$c_0, , c_{n-1}, c_n, , c_{2n-1}$$

→

$$f_i = c_i - c_{i+n}, \;\; i = 0 \ldots n - 1.$$

3. Reduce mod q.

3.2 NTT Skeleton

Kyber uses in-place Cooley–Tukey NTT with precomputed zeta roots. A correct NTT multiplies in $O(n\, extlogn)$ using Montgomery/barrett reductions.

Pseudo-code:

```
for len=1; len&lt;n; len*=2:
  for start in 0..n-1 step 2*len:
    for j in 0..len-1:
      w = zetas[idx++];
      u = a[start+j];
      v = montgomery_reduce(w * a[start+j+len]);
      a[start+j] = (u+v) mod q;
      a[start+j+len] = (u-v) mod q;
```

Inverse NTT uses $inv\, extzetas$.

4. Message Encoding / Decoding

4.1 Encoding

To encode up to $L = \lfloor n/REP \rfloor - CHECKSUM$ bytes:

1. Append truncated checksum (4 bytes): $CS = \mathrm{SHA3\text{-}256}(M)[: 4]$.
2. Form $M' = M || CS$.
3. Map each byte $b \in [0, 255]$ to $c = \lfloor b\, q/256 \rfloor$.
4. Repeat each c REP times in consecutive slots.

4.2 Decoding

1. For each REP-sized block, compute centered coefficients and average:

$$?arc = \frac{1}{REP} \sum_i ((c_i + q/2) \bmod q - q/2),$$

2. Recover byte: $b = \mathrm{round}(\bar{c} * 256/q)$.
3. Split (M, CS) and verify $CS == \mathrm{SHA3\text{-}256}(M)[:4]$.

5. Protocols

5.1 Key Generation

1. Sample .
2. Generate public matrix $A \in R_q^{k \times k}$.
3. Compute $t = As + e \in R_q^k$.

Outputs: Public key (A, t), secret key s.

5.2 Encapsulation

1. Sample .
2. Compute

$$u_i = A_{\cdot,i} \cdot r + e1_i, v = t \cdot r + e2 + \mathrm{Encode}(M).$$

Outputs ciphertext (u, v) and payload M.

5.3 Decapsulation

1. Compute $v - u \cdot s = Encode(M) + noise$.
2. Decode M and verify checksum.

6. Homomorphic Operations

- **Addition**:

$$(u1 + u2, v1 + v2)$$

- **Multiplication**: Requires digit decomposition and relinearization keys to collapse degree-2 terms.

7. Security & Performance

- **Post-Quantum**: Based on MLWE hardness (SVP, SIVP).
- **Noise Management**: Convolution path correct but slow; NTT path needed for efficiency.
- **Integrity**: Checksum detects tampering.
- **Batching**: CRT packing can increase throughput.

- **Side-Channel**: Constant-time NTT and sampling protect against leakage.

References

1. Kyber specification (ref/ntt.c) for zeta constants.
2. NIST Post-Quantum Cryptography: Kyber round-final parameters.[50]
3. Learning With Errors foundational hardness proofs.[69]
4. Fully Homomorphic Encryption survey for multiplexing and relinearization techniques.