**Kyber:**

**inputs:**

ciphertext C1

re-encrypted ciphertext C2 = compress(U)

**output:**

C1 = C2?

**View of masking kyber:**

**inputs:**

ciphertext C1

uncompressed ciphertext shares u1, u2, u3 where U = u1 + u2 + u3 mod KYBER\_Q

**output:**

decompress(C1) = U?

**method:**

decompress(C1)->Upper bound E, Lower bound S

(if S<U<E, the output is true)

w = u1 – S

x = u1 – E

take the most significant bit of A2B(w, u2, u3) and A2B(x, u2, u3)

(The output is true if w+u2+u3 = u1+u2+u3-S > 0 and x+u2+u3 = u1 + u2 + u3 -E < 0) (1)

(After A2B, the sign of inputs should be preserved) (2)

**problems:**

1. These equations are only valid when u1+u2+u3<KYBER\_Q, but we cannot guarantee this due to the previous hashing process
2. The A2B method mentioned in “view of masking kyber” cannot handle negative numbers

**Another method: Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography**

“One option is to fall back to a generic comparison where all coefficients of the ciphertext are first converted into a Boolean sharing after which the comparison is performed using an appropriate masked Boolean circuit.”

That is to say, do A2B to each share in U, then compare the Boolean values.

**problem:**

The paper doesn’t mention what is “appropriate masked Boolean circuit” or how to implement it.

XOR(share i)1 to 3 =C2