



Mentor: Jen Cordaro, Org 05632, Manager: Tiawna Cayton, Org 056831

Problem

When programming cryptography, a mathematical proof of security does not guarantee a secure implementation.

A secure protocol's implementation may be broken by: bad randomness/keys, errors in rounding/conversion, leaked information, and more.

Objective

Develop and automate tests to find common vulnerabilities in arbitrary implementations of post quantum lattice cryptography protocols

System

Drop in any implementation



Impact

NIST is currently standardizing the next generation of post quantum cryptography algorithms, which need rigorous testing

Arthimo ...

- Is agnostic to implementation language, external libraries, compilers, compiler optimizations, hardware, source of randomness
- Incorporates existing research on attacks, side channels, and metamorphic testing
- Provides security estimates in classical and quantum settings

Next Steps

- Create more examples of implementation harnesses to evaluate a larger variety of existing, untested implementations
- Implement testing for PQC/classic hybrid protocols, the next logical step for real-world implementations

Tests

Do basic functionality tests correspond to the original reference implementation?

$$Dec(sk, Enc(pk, m)) == m?$$

Do badly formatted inputs cause unexpected behavior?

- Extra message length
- Changed ciphertexts without a proper decryption

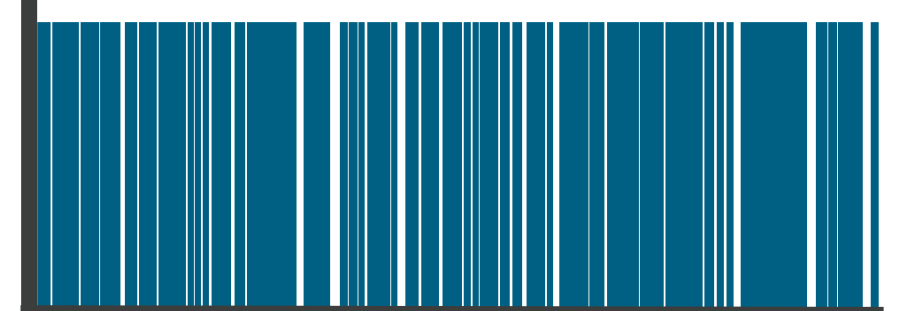
Are randomly generated elements well distributed?

Do intermediate values or outputs leak information?

Is this implementation vulnerable to established side channel attacks?

- BKZ lattice reduction, given hints
- Chosen ciphertext attacks

Distribution of shared secrets



Distribution of ciphertexts



Protocol: Kyber512 (CRYSTALS)
Implementation: KyberJCE (Steven K Fisher)