CRYSTALS

Cryptographic Suite for Algebraic Lattices

Introduction

Dilithium is a digital signature scheme that is strongly secure under chosen message attacks based on the hardness of lattice problems over module lattices. The security notion means that an adversary having access to a signing oracle cannot produce a signature of a message whose signature he hasn't yet seen, nor produce a different signature of a message that he already saw signed. Dilithium is one of the candidate algorithms submitted to the NIST post-quantum cryptography project.

For users who are interested in *using* Dilithium, we recommend the following:

- Use Dilithium in a so-called *hybrid mode* in combination with an established "pre-quantum" signature scheme.
- We recommend using the Dilithium3 parameter set, which —according to a very conservative analysis—achieves more than 128 bits of security against all known classical and quantum attacks.

Scientific Background

The design of Dilithium is based on the "Fiat-Shamir with Aborts" technique of Lyubashevsky which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure. The scheme with the smallest signature sizes using this approach is the one of Ducas, Durmus, Lepoint, and Lyubashevsky which is based on the NTRU assumption and crucially uses Gaussian sampling for creating signatures. Because Gaussian sampling is hard to implement securely and efficiently, we opted to only use the uniform distribution. Dilithium improves on the most efficient scheme that only uses the uniform distribution, due to Bai and Galbraith, by using a new technique that shrinks the public key by more than a factor of 2. To the best of our knowledge, Dilithium has the smallest public key + signature size of any lattice-based signature scheme that only uses uniform sampling.

Performance Overview

The table below gives an indication of the performance of the Dilithium with all the updates we applied to the parameter sets for round-3 of the NIST PQC project. All benchmarks were obtained on one core of an Intel Core-i7 6600U (Skylake) CPU. We report benchmarks of two different implementations: a C reference implementation and an optimized implementation using AVX2 vector instructions.

Dilithium2					
Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
		gen:	300751	gen:	124031
pk:	1312	sign:	1355434	sign:	333013
sig:	2420	verify:	327362	verify:	118412
Dilithium3					
Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
sk:		gen:	544232	gen:	256403
pk:	1952	sign:	2348703	sign:	529106
sig:	3293	verify:	522267	verify:	179424
Dilithium5					
Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
sk:		gen:	819475	gen:	298050
pk:	2592	sign:	2856803	sign:	642192
sig:	4595	verify:	871609	verify:	279936