Kyber terminates

Manuel Barbosa[1] and Peter Schwabe[2]

[1]University of Porto & INESC TEC
mbb@fc.up.pt
[2]MPI-SP & Radboud University
peter@cryptojedi.org

2023-05-17

Abstract

The key generation of the lattice-based key-encapsulation mechanism CRYSTALS-Kyber (or short, just Kyber) involves a rejection-sampling routine to produce coefficients modulo $q = 3329$ that look uniformly random. The input to this rejection sampling is output of the SHAKE-128 extendable output function (XOF). If this XOF is modelled as a random oracle with infinite output length, it is easy to see that Kyber terminates with probability 1; also, in this model, for any upper bound on the running time, the probability of termination is strictly smaller than 1.

In this short note we show that an (unconditional) upper bound for the running time for Kyber exists. Computing a tight upper bound, however, is (likely to be) infeasible. We remark that the result has no real practical value, except that it may be useful for computer-assisted reasoning about Kyber using tools that require a simple proof of termination.

1 Introduction

The lattice-based key-encapsulation mechanism Kyber [BDK+18, ABD+21], which has recently been selected by NIST for standardization [AAC+22], works over the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $q = 3329$ and $n = 256$. Kyber's public keys are of the form $\mathbf{As} + \mathbf{e}$, where $\mathbf{A} \in \mathcal{R}_q^{k \times k}$ "looks uniform", and $\mathbf{s}, \mathbf{e} \in \mathcal{R}_q^k$ are sampled from a noise distribution. The value $k \in \{2, 3, 4\}$ depends on the parameter set of Kyber. More concretely, the entry at position (i, j) of the matrix $\mathbf{A}$ is sampled from a 32-byte public random seed $\rho = (\rho_0, \ldots, \rho_{31})$ as $\mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$, where XOF is instantiated with SHAKE-128 [BDPA11, Nat15].

Intuitively, the Parse routine interprets the output of XOF as a (possibly infinitely long) sequence of 12-bit unsigned integers. The first 256 integers in that sequence that are smaller than q are taken as the coefficients of the output polynomial in $\mathcal{R}_q$; integers larger or equal than q are discarded. A high-level pseudocode description of Parse is given in Algorithm 1; for a more implementation-oriented specification that takes a byte sequence as input, see [ABD+21, Alg. 1].

Note that in Kyber, the output of Parse is assumed to be in NTT domain, but this detail does not matter for the discussion here, so we ignore it in Algorithm 1 for better readability. Also note that the approach of generating **A** by rejection-sampling the output of SHAKE-128 is not unique to Kyber. It was introduced in NewHope [ADPS16, Sec. 3] and is used, for example, also in Dilithium [DKL+18, BDK+21]. The reasoning in this short note can be adapted straightforwardly to those schemes and possibly other lattice-based primitives that use the same approach to generate a matrix **A**.

Algorithm 1 Parse: $\{0,\dots,4095\}^* \to R_q$

```
Require: Sequence of 12-bit unsigned integers $d_0, d_1, d_2 \cdots \in \{0,\dots,4095\}^*$
Ensure: Polynomial $\mathbf{a} \in R_q$
 1: $i \leftarrow 0$
 2: $j \leftarrow 0$
 3: while $j < n$ do
 4:     if $d_i < q$ then
 5:         $a_j \leftarrow d_i$
 6:         $j \leftarrow j + 1$
 7:     end if
 8:     $i \leftarrow i + 1$
 9: end while
10: return $\mathbf{a} = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$
```

In practice, the matrix-generation step is fast, but it is not obvious to see that there exists an upper bound on the number of XOF-output 12-bit integers that Parse needs to look at before finding 256 coefficients smaller than q to produce the output polynomial in $\mathcal{R}_q$. In fact, as we will briefly discuss in Section 2, for Parse on uniformly random input no such bound exists. However, the input to Parse is not uniform, but output of SHAKE-128, on input (ρ, i, j). This allows us to prove, in Section 3, that such an upper bound *does* exist.

We remark that the generation of the matrix **A** is the only part of Kyber that makes termination somewhat intricate to prove. All other parts execute in a fixed, input-independent, number of steps.

2 Parse on random input

If we consider the Parse routine operating on uniformly random input, i.e., model SHAKE-128 as a random oracle with infinite output length, we compute the probability that Parse does not terminate after $\ell \geq 256$ iterations of the main loop (Line 3) as

$$P_{\mathsf{fail}}(\ell) = \sum_{i=0}^{255} \binom{\ell}{i} p^i (1-p)^{\ell-i},$$

where $p = 3329/4096$ is the probability that a 12-bit unsigned integer is smaller than $q = 3329$. The probability P_{fail} quickly converges towards zero as ℓ grows; for example, for $\ell = 560$ (corresponding to 5 blocks of SHAKE-128 output), we have $P_{\mathsf{fail}}(560) < 2.9 \cdot 10^{-79}$. This means that in practice, we are very unlikely to ever need more than 5 blocks of SHAKE-128 output per entry of the matrix

A. However, we also see that for any value ℓ, we have $P_{\mathsf{fail}}(\ell) > 0$, which means that on uniformly random input to Parse we cannot prove any upper bound on the execution time of Kyber's key-generation.

3 Parse on SHAKE-128 output

In reality, the input of Parse is the output of SHAKE-128. SHAKE-128 is built from the Keccak-$p[1600, 24]$ permutation [Nat15, Sec. 3.3]. Our reasoning does not require any particular properties of this permutation, other than the fact that it is a permutation over a 1600-bit state. In the following we will thus denote the Keccak-$p[1600, 24]$ permutation simply as π. SHAKE-128 consists of two phases: the Absorb phase which "soaks up" the input message into the state and the Squeeze phase, which generates a stream of output bytes from the state. Both Absorb and Squeeze use the same split of the state into 1344 *rate* bits (168 rate bytes) and 256 *capacity* bits.

In the specific use case in Kyber, SHAKE-128 takes as input the concatenation of the 32-byte random seed ρ and indices i and j (each encoded as a single byte). The Absorb phase on this input is fairly simple: It first produces the 200-byte state

$$S_{\text{-}1} = (\underbrace{\rho_0, \dots, \rho_{31}, i, j, 31, 0, \dots, 0, 128}_{168\text{-byte rate}}, \underbrace{0, \dots, 0}_{32\text{-byte capacity}}),$$

and then computes $S_0 = \pi(S_{\text{-}1})$. The Squeeze phase produces output in blocks of 168 bytes by copying the rate part of S_i to the output and then computing $S_{i+1} = \pi(S_i)$ for $i = 0, 1, \dots$

We are now ready to prove the existence of an upper bound on the running time of $\mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$ and, as a consequence, of Kyber. Our reasoning proceeds through three steps.

1. We use that π is a permutation and thus, a product of disjoint cycles.[1] This means that starting from state S_0, SHAKE-128 will permute the state in a cycle of some length m as follows:

$$S_0 \xrightarrow{\pi} S_1 \xrightarrow{\pi} S_2 \xrightarrow{\pi} \dots \xrightarrow{\pi} S_{m-1} \xrightarrow{\pi} S_0.$$

   The output of SHAKE-128 is the concatenation of the rate parts (i.e., the first 168 bytes) of the states S_i. After $168m$ bytes the output repeats.

2. In order for $\mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$ to *not* terminate, the rate parts of the states $S_0, \dots, S_{m-1}$, each interpreted as a sequence of 12-bit integers, must not contain a single integer in $\{0, \dots, 3328\}$. If any state contained such a 12-bit integer in its rate part, this integer would be part of the output of SHAKE-128 and 256 iterations through the permutation cycle (i.e., $256m$ invocations of π) would output this integer 256 times, which is sufficient for Parse to terminate.

[1] This fact alone shows that the intuition that the output of SHAKE-128 can be modelled as a random oracle of infinite output length is incorrect.

3. What remains is to show that at least one of the states $S_0, \dots, S_{m-1}$ necessarily contains at least one 12-bit integer in $\{0, \dots, 3328\}$ in its rate part. We do this by observing that $S_{m-1} = \pi^{-1}(S_0) = S_{\text{-}1}$ is part of the permutation cycle. Now we simply use the fact that we know large parts of $S_{\text{-}1}$ and write its rate part as a tuple of 12-bit integers as

$$(r_0, \dots, r_{20}, 16i + (\rho_{31} \mod 16), 3840 + j, 1, \underbrace{0, \dots, 0}_{87 \text{ times}}, 2048),$$

where the values $r_0, \dots, r_{20}$ are determined by ρ. As $i \in \{0, 1, 2, 3\}$ for all parameter sets of Kyber, we see that $S_{\text{-}1}$ contains not just one, but at least 90 values in $\{0, \dots, 3328\}$. This guarantees that, after at most 3 iterations through the permutation cycle (i.e., $3m$ invocations of π), SHAKE-128 will have produced sufficiently many 12-bit integers in $\{0, \dots, 3328\}$ for Parse to terminate.

Following this argumentation, we can easily derive an upper-bound on the maximum number of steps that Kyber's key-generation takes to terminate. As the state of Keccak has 1600 bits, clearly, $m \leq 2^{1600}$. Hence, each call to $\mathsf{Parse}(\mathsf{XOF}(\rho, i, j)$ takes at most $3 \cdot 2^{1600}$ invocations of π and, as there are k^2 such calls, Kyber's matrix generation is guaranteed to terminate after $3k^2 \cdot 2^{1600}$ invocations of π. This bound is highly non-tight for multiple reasons:

- As Keccak attempts to approximate the characteristics of a random permutation it is highly unlikely that it consists of only one cycle of length 2^{1600}.

- In order to reach $S_{\text{-}1}$, the permutation cycle starting from S_0 must not produce 256 output values in $\{0, \dots, 3328\}$ before reaching $S_{\text{-}1}$; in order to reach $S_{\text{-}1}$ twice, it must not produce 166 such output values, and in order to reach $S_{\text{-}1}$ three times it must not produce 76 such output values from all of the other states in the cycle. This means in particular that if we have all 2^{1600} possible states in one cycle, we will never reach $S_{\text{-}1}$ from S_0 before outputting 256 values in $\{0, \dots, 3328\}$.

- Most importantly though, for short outputs of SHAKE-128 the random-oracle heuristic is a good approximation, if the permutation underlying the sponge construction is modelled as being randomly sampled [BDPV08]. Consequently, we conjecture that, for all values of $\rho \in \{0, 1\}^{256}$ and $i, j \in \{0, 1, 2, 3\}$, the computation of $\mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$ will terminate after $c < 8 \ll 3 \cdot 2^{1600}$ invocations of Squeeze. Unfortunately, as far as we can tell, the only way to prove this conjecture would be to try all 2^{260} possible inputs, which is clearly infeasible.

  However, we are offering a dinner in Casa de Chá da Boa Nova[2] as a prize for reporting a combination of $\rho \in \{0, 1\}^{256}$, and $i, j \in \{0, 1, 2, 3\}$, such that more than 5 blocks (840 bytes, 5 invocations of Keccak-$p[1600, 24]$) of SHAKE-128(ρ, i, j) output are needed for Parse to terminate.

[2] https://www.casadechadaboanova.pt/en/

References

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status report on the third round of the NIST post-quantum cryptography standardization process. NISTIR 8413, 2022. https://csrc.nist.gov/publications/detail/nistir/8413/final. 1

[ABD+21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.02). Round-3 submission to the NIST PQC standardization project, 2021. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf. 1

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange – a new hope. In *Proceedings of the 25th USENIX Security Symposium*. USENIX Association, 2016. Document ID: 0462d84a3d34b12b75e8f5e4ca032869, http://cryptojedi.org/papers/#newhope. 2

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*, pages 353–367. IEEE, 2018. https://eprint.iacr.org/2017/634. 1

[BDK+21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm specifications and supporting documentation (version 3.1). Round-3 submission to the NIST PQC standardization project, 2021. https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf. 2

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference. Submission to the NIST SHA-3 competition, 2011. https://keccak.team/files/Keccak-reference-3.0.pdf. 1

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008. 4

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS – Dilithium: Digital signatures from module lattices. *Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018. http://cryptojedi.org/papers/#dilithium. 2

[Nat15] National Institute of Standards and Technology. FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions, 2015. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf. 1, 3