Document title

Author 1

July 20, 2024

1 Kyber and ML-KEM

CRYSTALS-Kyber [BDK+18] is an `IND-CCA` secure key encapsulation mechanism whose security is based on the hardness of the Module Learning with Error (MLWE) problem. It is submitted to NIST's "Pots-Quantum Cryptography" contest, where it advanced to the third round [ABD+19].

A modified version was standardized by NIST [KE23] and renamed to "ML-KEM".

Algorithm 1 `Kyber.CPAPKE.KeyGen`()

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) \leftarrow G(d)$ ▷ G is instantiated with `SHA3-512`
$N \leftarrow 0$
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    for $j \in \{0, 1, \ldots, k-1\}$ **do**
        $\hat{A}_{i,j} \leftarrow \texttt{Parse}(\texttt{XOF}(\rho, j, i))$ ▷ `XOF` instantiated with `Shake128`
    end for
end for
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    $\mathbf{s}_i \leftarrow \texttt{CBD}_{\eta_1}(\texttt{PRF}(\sigma, N))$
    $N \leftarrow N + 1$
end for
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    $\mathbf{e}_i \leftarrow \texttt{CBD}_{\eta_1}(\texttt{PRF}(\sigma, N))$
    $N \leftarrow N + 1$
end for
$\hat{\mathbf{s}} \leftarrow \texttt{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} \leftarrow \texttt{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} \leftarrow \hat{A} \cdot \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$\texttt{pk} \leftarrow (\hat{\mathbf{t}}, \rho)$
$\texttt{sk} \leftarrow \hat{\mathbf{s}}$
return $(\texttt{pk}, \texttt{sk})$

Algorithm 2 $\texttt{Kyber.CPAPKE.Enc}(\texttt{pk}, m, r)$

Require: Public key $\texttt{pk} = (\rho, \hat{\mathbf{t}})$
Require: Message $m \in \mathcal{B}^{32}$
Require: Random coin $r \in \mathcal{B}^{32}$
$N \leftarrow 0$
$(\hat{\mathbf{t}}, \rho) \leftarrow \texttt{pk}$ ▷ Unpack and decode the public key
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    for $j \in \{0, 1, \ldots, k-1\}$ **do**
        $\hat{A}^{\intercal}[i][j] \leftarrow \texttt{Parse}(\texttt{XOF}(\rho, i, j))$
    end for
end for
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    $\mathbf{r}[i] \leftarrow \texttt{CBD}_{\eta_1}(\texttt{PRF}(r, N))$
    $N \leftarrow N + 1$
end for
for $i \in \{0, 1, \ldots, k-1\}$ **do**
    $\mathbf{e}_1[i] \leftarrow \texttt{CBD}_{\eta_2}(\texttt{PRF}(r, N))$
    $N \leftarrow N + 1$
end for
$e_2 \leftarrow \texttt{CBD}_{\eta_2}(\texttt{PRF}(r, N))$
$\hat{\mathbf{r}} \leftarrow \texttt{NTT}(\mathbf{r})$
$\mathbf{c}_1 \leftarrow \texttt{NTT}^{-1}(\hat{A}^{\intercal} \cdot \hat{\mathbf{r}}) + \mathbf{e}_1$
$c_2 \leftarrow \texttt{NTT}^{-1}(\hat{\mathbf{t}} \cdot \hat{\mathbf{r}}) + e_2 + m$
return $c = (\mathbf{c}_1, c_2)$

Algorithm 3 $\texttt{Kyber.CPAPKE.Dec}(\texttt{sk}, c)$

Require: Secret key $\texttt{sk} = \hat{\mathbf{s}}$
Require: Ciphertext $c = (\mathbf{c}_1, c_2)$
$\hat{\mathbf{s}} \leftarrow \texttt{sk}$
$(\mathbf{c}_1, c_2) \leftarrow c$
$m \leftarrow c_2 - \texttt{NTT}^{-1}(\hat{\mathbf{s}} \cdot \texttt{NTT}(\mathbf{c}_1))$
return $\texttt{Round}(m)$

Algorithm 4 `Kyber.CCAKEM.KeyGen()`

$z \xleftarrow{\$} \mathcal{B}^{32}$ ▷ Randomly sample 32 bytes (256 bits)
$(\texttt{pk}, \texttt{sk}') \xleftarrow{\$}$ `Kyber.CPAPKE.KeyGen()`
$\texttt{sk} = (\texttt{sk}', \texttt{pk}, H(\texttt{pk}), z)$ ▷ H is instantiated with SHA3-256
return $(\texttt{pk}, \texttt{sk})$

Algorithm 5 `Kyber.CCAKEM.Encap(pk)`

$m \xleftarrow{\$} \mathcal{B}^{32}$
$m \leftarrow H(m)$ ▷ Do not output system RNG directly
$(\bar{K}, r) \leftarrow G(m \| H(\texttt{pk}))$ ▷ G is instantiated with SHA3-512
$c \leftarrow$ `Kyber.CPAPKE.Enc(pk, m, r)` ▷ Because r is set, `CPAPKE` is deterministic
$K \leftarrow \texttt{KDF}(\bar{K} \| H(c))$ ▷ `KDF` is instantiated with Shake256
return (c, K)

Algorithm 6 `Kyber.CCAKEM.Decap(sk, c)`

Require: Secret key $\mathtt{sk} = (\mathtt{sk}', \mathtt{pk}, H(\mathtt{pk}), z)$
Require: `Kyber.CPAPKE` Ciphertext c

$(\mathtt{sk}', \mathtt{pk}, h, z) \leftarrow \mathtt{sk}$ ▷ Unpack the secret key; h is the hash of `pk`
$m' \leftarrow \mathtt{Kyber.CPAPKE.Dec}(\mathtt{sk}', c)$
$(\overline{K}', r') \leftarrow G(m' \| h)$
$c' = \mathtt{Kyber.CPAPKE.Enc}(\mathtt{pk}, m', r')$
if $c = c'$ **then**
    $K \leftarrow \mathtt{KDF}(\bar{K}' \| H(c))$
else
    $K \leftarrow \mathtt{KDF}(z \| H(c))$
end if
return K

References

[ABD$^+$19] 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. *NIST PQC Round*, 2(4):1–43, 2019.

[BDK$^+$18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[KE23] NIST Module-Lattice-Based Key-Encapsulation. Mechanism standard. *NIST Post-Quantum Cryptography Standardization Process; NIST: Gaithersburg, MD, USA*, 2023.