FFI Norwegian Defence Research Establishment

CRYSTALS-Kyber

Martin Strand

29 Nov 2023

Foto: AdobeStock

FFI Norwegian Defence Research Establishment

~~CRYSTALS-Kyber~~ FIPS-203 ML-KEM

Martin Strand

29 Nov 2023

Foto: AdobeStock

Cryptography oversimplified

World A

- The parties share strong secrets
- Encryption is cheap and efficient

World B

- The parties don't share [fresh] secrets
- Encryption is significantly less efficient
- Goal: Get to world A

Idea

Encrypt a symmetric key in world B, and use it in world A.

Cryptography oversimplified

World A

- The parties share strong secrets
- Encryption is cheap and efficient

World B

- The parties don't share [fresh] secrets
- Encryption is significantly less efficient
- Goal: Get to world A

Idea

Encrypt a symmetric key in world B, and use it in world A.
KEM: PKE only for random keys

Recap: IND-CPA security definition

Game $\mathrm{IND\text{-}CPA}_{\mathcal{E},\mathcal{A}}(\lambda)$	Oracle $\textsc{Dec}(c)$
$b \leftarrow\$ \{0,1\}$	**if** $c \neq c^*$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{KGen}(1^\lambda)$	**return** $\mathcal{E}.\mathsf{Dec}(c;\mathsf{sk})$
$(\mathsf{state}, m_0) \leftarrow \mathcal{A}(\mathsf{pk})$	
$m_1 \leftarrow\$ \{0,1\}$	
$c^* \leftarrow \mathcal{E}.\mathsf{Enc}(m_b;\mathsf{pk})$	
$b' \leftarrow \mathcal{A}(\mathsf{state},\mathsf{pk},c)$	
return $b = b'$	

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$$

Recap: IND-CCA security definition

Game $\mathrm{IND\text{-}CCA}_{\mathcal{E},\mathcal{A}}(\lambda)$	Oracle $\textsc{Dec}(c)$
$b \leftarrow\$ \{0,1\}$	**if** $c \neq c^*$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{KGen}(1^\lambda)$	**return** $\mathcal{E}.\mathsf{Dec}(c; \mathsf{sk})$
$(\mathsf{state}, m_0) \leftarrow \mathcal{A}^{\textsc{Dec}}(\mathsf{pk})$	
$m_1 \leftarrow\$ \{0,1\}$	
$c^* \leftarrow \mathcal{E}.\mathsf{Enc}(m_b; \mathsf{pk})$	
$b' \leftarrow \mathcal{A}^{\textsc{Dec}}(\mathsf{state}, \mathsf{pk}, c)$	
return $b = b'$	

$$\mathsf{Adv}^{\text{ind-cca}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left| \Pr[\text{IND-CCA}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

Textbook ElGamal

KGen Let $G = \langle g \rangle$ of order q. Choose a secret $\mathrm{sk} = s$ and compute $\mathrm{pk} = h = g^s$

Enc(pk, m) Select r at random. Compute $c = (c_1 = g^r, c_2 = mh^r)$

Dec(sk, c) Compute $m' = c_1^{-s} c_2$

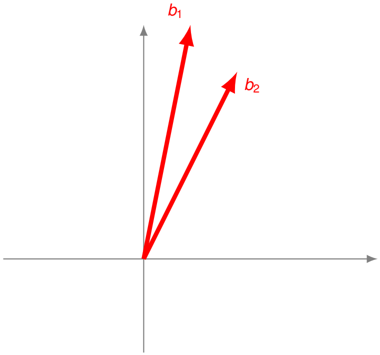
ElGamal: A sky-high perspective

KGen — In some structure, let s be some secret, and let pk embed the secret in the structure.

Enc(pk, m) — Select r at random. Bind r to the structure as well as to the public key. Bind the message to the latter.

Dec(sk, c) — Use the private key on the embedding of r, and compute its inverse. Use commutativity to remove both r and s from the message.

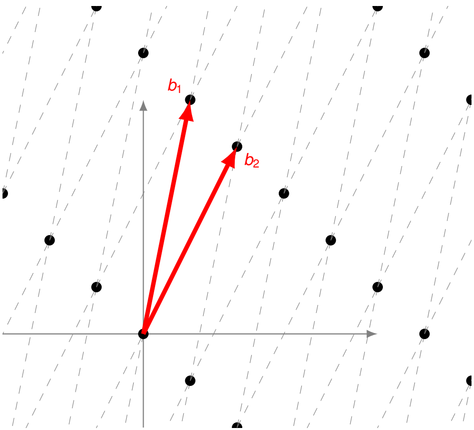
Lattice problems



Lattice problems

Shortest Vector Problem

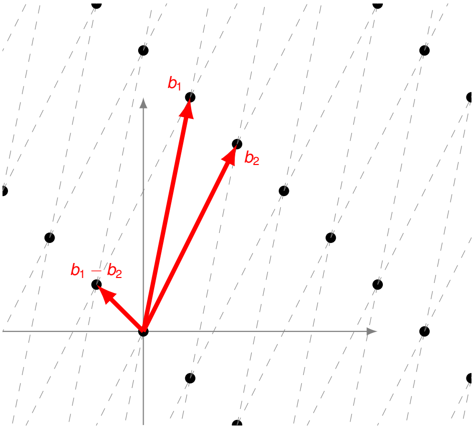
Given a basis for L, find the shortest vector in V that is also a point in L.



Lattice problems

Shortest Vector Problem

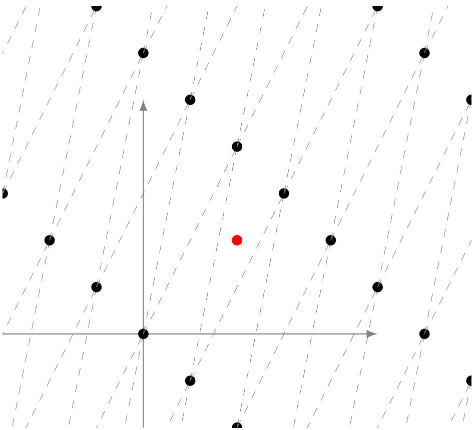
Given a basis for L, find the shortest vector in V that is also a point in L.



Lattice problems

Shortest Vector Problem
Given a basis for L, find the shortest vector in V that is also a point in L.

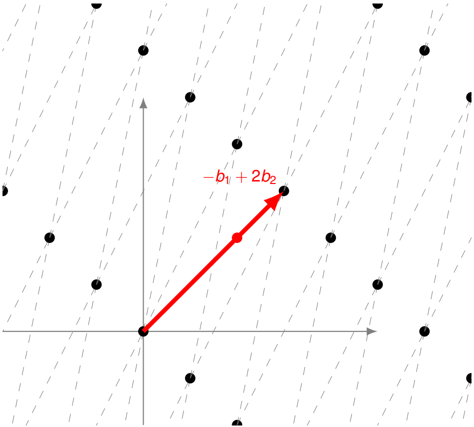
Closest Vector Problem
Given a basis for L and a point v in V, find closest lattice point to v in L.



Lattice problems

Shortest Vector Problem
: Given a basis for L, find the shortest vector in V that is also a point in L.

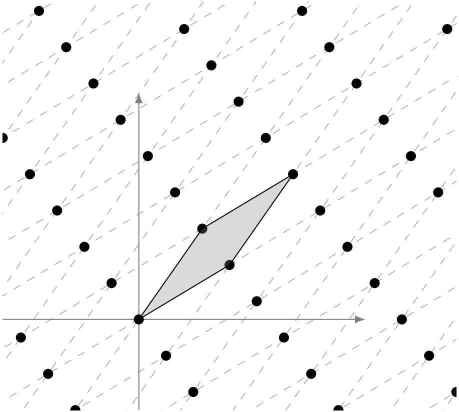
Closest Vector Problem
: Given a basis for L and a point v in V, find closest lattice point to v in L.



Learning with errors

$$a_{1,1}s_1 + \ldots a_{1,n}s_n + e_1 = b_1$$
$$a_{2,1}s_1 + \ldots a_{2,n}s_n + e_2 = b_2$$
$$a_{3,1}s_1 + \ldots a_{3,n}s_n + e_3 = b_3$$
$$a_{4,1}s_1 + \ldots a_{4,n}s_n + e_4 = b_4$$
$$a_{5,1}s_1 + \ldots a_{5,n}s_n + e_5 = b_5$$
$$\vdots$$

Given A, b, and if e_i are small, what is s?



Learning with errors

Let n, q be positive integers and let χ be a probability distribution over $\mathbb{Z}$.
Let a_i be a vector over $\mathbb{Z}_q$, and let $s \leftarrow \chi^n$, $e_i \leftarrow \chi$ be sampled independently according to χ.

Challenge

Distinguish between

- (a_i, b_i), b_i uniformly sampled from $\mathbb{Z}_q$, and
- $(a_i, a_i^T \cdot s + e_i)$

Ring-LWE

Let q be a prime, let $f(X)$ be a polynomial, and let $R_q = \mathbb{Z}_q[X]/\left(f(X)\right)$. Let χ be a distribution over R_q.
Let $a_i(X)$ be a polynomial from R_q, and $e_i(X)$ and $s(X)$ be small polynomials sampled independently according to χ.

Challenge

Distinguish between

- $(a_i(X), b_i(X))$, $b_i(X)$ uniformly sampled from R_q, and
- $(a_i(X), a_i(X) \cdot s(X) + e_i(X))$

Module-LWE

Let q be a prime, d a power of 2 and n an integer. Let $f(X) = X^d + 1$ be a polynomial, and let $R_q = \mathbb{Z}_q[X]/\left(f(X)\right)$. Let χ be a distribution over R_q.
Sample a_i from R_q^n and e_i, s similarly as before.

Challenge

Distinguish between

- $(a_i(X), b_i(X))$, $b_i(X)$ uniformly sampled from R_q, and
- $(a_i(X), \langle a_i(X), s(X)\rangle + e_i(X))$

Kyber as mathematics

Let $R_q = \mathbb{Z}_q[X]/(X^{256}+1)$ be a ring.

KGen
1. Choose matrix A from $R_q^{k\times k}$
2. Choose short vectors $\mathrm{sk} = s$ and e from R_q^k
3. Compute $t = As + e$, and set $\mathrm{pk} = (t, A)$

Enc(pk, m)
1. Choose short r, e_1 from R_q^k and e_2 from R_q^k
2. Set $u = A^T r + e_1$ and $v = t^T \cdot r + e_2 + m$
3. Return $c = (u, v)$

Dec(sk, c) Compute $w = v - s^T \cdot u$ and return w

Correctness

$$v - s^T \cdot u$$

Correctness

$$v - s^T \cdot u = t^T + e_2 + m - s^T \left(A^T r + e_1\right)$$

Correctness

$$
\begin{aligned}
v - s^T \cdot u &= t^T + e_2 + m - s^T \left(A^T r + e_1\right) \\
&= (As + e)^T r + e_2 + m - (As)^T r + s^T \cdot e_1
\end{aligned}
$$

Correctness

$$
\begin{aligned}
v - s^T \cdot u &= t^T + e_2 + m - s^T \left(A^T r + e_1\right) \\
&= (As + e)^T r + e_2 + m - (As)^T r + s^T \cdot e_1 \\
&= (As)^T r - (As)^T r + e^T \cdot r + e_2 + s^T e_1 + m
\end{aligned}
$$

Correctness

$$\begin{aligned} v - s^T \cdot u &= t^T + e_2 + m - s^T \left(A^T r + e_1\right) \\ &= (As + e)^T r + e_2 + m - (As)^T r + s^T \cdot e_1 \\ &= (As)^T r - (As)^T r + e^T \cdot r + e_2 + s^T e_1 + m \\ &= m + (small) \end{aligned}$$

Warning: Mathematics follows!

Tool: Fourier series

Let s be a periodic function.

$$A_0 = \frac{1}{P}\int_{-P/2}^{P/2} s(x)\,dx$$

$$A_n = \frac{2}{P}\int_{-P/2}^{P/2} s(x)\cos\left(\frac{2\pi nx}{P}\right)\,dx$$

$$B_n = \frac{2}{P}\int_{-P/2}^{P/2} s(x)\sin\left(\frac{2\pi nx}{P}\right)\,dx$$

$$s(x) \sim A_0 + \sum_{n=1}^{\infty}\left(A_n\cos\left(\frac{2\pi nx}{P}\right) + B_n\sin\left(\frac{2\pi nx}{P}\right)\right)$$

$(A_0, A_1, B_1, A_2, B_2, \ldots)$ and s are equivalent representations of the same function.

The Kyber polynomial

Let $q = 3329 = 2^8 \cdot 13 + 1$ and let $\zeta = 17$, a primitive 256-th root of unity modulo q.

Fact

$$X^{256} + 1 = \prod_{i=0}^{127} (X^2 - \zeta^{2i+1})$$

The Kyber polynomial

Let $q = 3329 = 2^8 \cdot 13 + 1$ and let $\zeta = 17$, a primitive 256-th root of unity modulo q.

Fact

$$X^{256} + 1 = \prod_{i=0}^{127} (X^2 - \zeta^{2i+1}) = \prod_{i=0}^{127} \left(X^2 - \zeta^{2\mathrm{BitReverse}_7(i)+1} \right)$$

FFI

The Number Theoretic Transform (NTT)

Fact

$$R_q = \mathbb{Z}[X]/(X^{256}+1) \simeq \bigoplus_{k=0}^{127} \mathbb{Z}_q[X]/\left(X^2 - \zeta^{2\mathrm{BitReverse}_7(i)+1}\right) = T_q$$

Let $f \in R_q$. Then $\mathrm{NTT} : R_q \to T_q$ is given by

$$\mathrm{NTT}(f) = \left(f \bmod \left(X^2 - \zeta^{2\mathrm{BitReverse}_7(0)+1}\right), \ldots, f \bmod \left(X^2 - \zeta^{2\mathrm{BitReverse}_7(127)+1}\right)\right)$$

and NTT^{-1} is also efficient.

Kyber in the NTT realm

Multiplication in R_q:
256×256 multiplications

Multiplication in T_q:
128×4 multiplications

Sampling algorithms

SampleNTT	Convert a stream of bytes into a polynomial in the NTT domain
SamplePolyCBD$_\eta$	Sample a coefficient array of a polynomial $f \in R_q$, according to a centered binomial distribution specified by η.

Compression using seeds

$\rho \leftarrow \mathrm{ek}_{\mathrm{PKE}}[384k : 384k+32]$ ▷ extract 32-byte seed from $\mathrm{ek}_{\mathrm{PKE}}$
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ re-generate matrix $\hat{\mathbf{A}} \in (\mathbb{Z}_q^{256})^{k \times k}$
  for $(j \leftarrow 0;\ j < k;\ j{+}{+})$
    $\hat{\mathbf{A}}[i,j] \leftarrow \mathsf{SampleNTT}(\mathsf{XOF}(\rho, i, j))$
  end for
end for

Computer-friendly representation

Algorithm 4 ByteEncode$_d(F)$

Encodes an array of d-bit integers into a byte array, for $1 \le d \le 12$.

Input: integer array $F \in \mathbb{Z}_m^{256}$, where $m = 2^d$ if $d < 12$ and $m = q$ if $d = 12$.
Output: byte array $B \in \mathbb{B}^{32d}$.

for $(i \leftarrow 0;\ i < 256;\ i{+}{+})$
  $a \leftarrow F[i]$ $\quad\triangleright\ a \in \mathbb{Z}_{2^d}$
  for $(j \leftarrow 0;\ j < d;\ j{+}{+})$
    $b[i \cdot d + j] \leftarrow a \bmod 2$ $\quad\triangleright\ b \in \{0,1\}^{256 \cdot d}$
    $a \leftarrow (a - b[i \cdot d + j])/2$ $\quad\triangleright$ note $a - b[i \cdot d + j]$ is always even.
  end for
end for
$B \leftarrow$ BitsToBytes(b)
return B

Algorithm 5 ByteDecode$_d(B)$

Decodes a byte array into an array of d-bit integers, for $1 \le d \le 12$.

Input: byte array $B \in \mathbb{B}^{32d}$.
Output: integer array $F \in \mathbb{Z}_m^{256}$, where $m = 2^d$ if $d < 12$ and $m = q$ if $d = 12$.

$b \leftarrow$ BytesToBits(B)
for $(i \leftarrow 0;\ i < 256;\ i{+}{+})$
  $F[i] \leftarrow \sum_{j=0}^{d-1} b[i \cdot d + j] \cdot 2^j \bmod m$
end for
return F

Compression and decompression of numbers

$$
\begin{aligned}
\mathrm{Compress}_d : \mathbb{Z}_q &\to \mathbb{Z}_{2^d} \\
x &\mapsto \left\lfloor \left(2^d/q\right) \cdot x \right\rceil \\
\mathrm{Decompress}_d : \mathbb{Z}_{2^d} &\to \mathbb{Z}_q \\
y &\mapsto \left\lfloor \left(q/2^d\right) \cdot y \right\rceil
\end{aligned}
$$

Compression and decompression of numbers

$$\begin{aligned}
\mathsf{Compress}_d : \mathbb{Z}_q &\to \mathbb{Z}_{2^d} \\
x &\mapsto \left\lfloor (2^d/q) \cdot x \right\rceil \\
\mathsf{Decompress}_d : \mathbb{Z}_{2^d} &\to \mathbb{Z}_q \\
y &\mapsto \left\lfloor (q/2^d) \cdot y \right\rceil
\end{aligned}$$

$\mathsf{Decompress}_d \circ \mathsf{Compress}_d \approx 1$

$$[\mathsf{Decompress}_d(\mathsf{Compress}_d(x)) - x] \bmod {}^{\pm} q \leq \left\lfloor q/2^{d+1} \right\rceil$$

The finished K-PKE algorithm

Algorithm 12 K-PKE.KeyGen()

Generates an encryption key and a corresponding decryption key.

Output: encryption key $\mathsf{ek}_{\mathrm{PKE}} \in \mathbb{B}^{384k+32}$.
Output: decryption key $\mathsf{dk}_{\mathrm{PKE}} \in \mathbb{B}^{384k}$.

$d \xleftarrow{\$} \mathbb{B}^{32}$ ▷ d is 32 random bytes (see Section 3.3)
$(\rho, \sigma) \leftarrow G(d)$ ▷ expand to two pseudorandom 32-byte seeds
$N \leftarrow 0$
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ generate matrix $\hat{\mathbf{A}} \in (\mathbb{Z}_q^{256})^{k \times k}$
    for $(j \leftarrow 0;\ j < k;\ j{+}{+})$
        $\hat{\mathbf{A}}[i,j] \leftarrow \mathsf{SampleNTT}(\mathsf{XOF}(\rho, i, j))$ ▷ each entry of $\hat{\mathbf{A}}$ uniform in NTT domain
    end for
end for
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ generate $\mathbf{s} \in (\mathbb{Z}_q^{256})^k$
    $\mathbf{s}[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(\sigma, N))$ ▷ $\mathbf{s}[i] \in \mathbb{Z}_q^{256}$ sampled from CBD
    $N \leftarrow N + 1$
end for
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ generate $\mathbf{e} \in (\mathbb{Z}_q^{256})^k$
    $\mathbf{e}[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(\sigma, N))$ ▷ $\mathbf{e}[i] \in \mathbb{Z}_q^{256}$ sampled from CBD
    $N \leftarrow N + 1$
end for
$\hat{\mathbf{s}} \leftarrow \mathsf{NTT}(\mathbf{s})$ ▷ NTT is run k times (once for each coordinate of $\mathbf{s}$)
$\hat{\mathbf{e}} \leftarrow \mathsf{NTT}(\mathbf{e})$ ▷ NTT is run k times
$\hat{\mathbf{t}} \leftarrow \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$ ▷ noisy linear system in NTT domain
$\mathsf{ek}_{\mathrm{PKE}} \leftarrow \mathsf{ByteEncode}_{12}(\hat{\mathbf{t}}) \| \rho$ ▷ $\mathsf{ByteEncode}_{12}$ is run k times; include seed for $\hat{\mathbf{A}}$
$\mathsf{dk}_{\mathrm{PKE}} \leftarrow \mathsf{ByteEncode}_{12}(\hat{\mathbf{s}})$ ▷ $\mathsf{ByteEncode}_{12}$ is run k times
return $(\mathsf{ek}_{\mathrm{PKE}}, \mathsf{dk}_{\mathrm{PKE}})$

The finished K-PKE algorithm

Algorithm 13 K-PKE.Encrypt$(\mathsf{ek}_{\mathrm{PKE}}, m, r)$

Uses the encryption key to encrypt a plaintext message using the randomness r.

Input: encryption key $\mathsf{ek}_{\mathrm{PKE}} \in \mathbb{B}^{384k+32}$.
Input: message $m \in \mathbb{B}^{32}$.
Input: encryption randomness $r \in \mathbb{B}^{32}$.
Output: ciphertext $c \in \mathbb{B}^{32(d_u k + d_v)}$.

$N \leftarrow 0$
$\hat{\mathbf{t}} \leftarrow \mathsf{ByteDecode}_{12}(\mathsf{ek}_{\mathrm{PKE}}[0 : 384k])$
$\rho \leftarrow \mathsf{ek}_{\mathrm{PKE}}[384k : 384k+32]$ ▷ extract 32-byte seed from $\mathsf{ek}_{\mathrm{PKE}}$
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ re-generate matrix $\hat{\mathbf{A}} \in (\mathbb{Z}_q^{256})^{k \times k}$
  for $(j \leftarrow 0;\ j < k;\ j{+}{+})$
    $\hat{\mathbf{A}}[i, j] \leftarrow \mathsf{SampleNTT}(\mathsf{XOF}(\rho, i, j))$
  end for
end for
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ generate $\mathbf{r} \in (\mathbb{Z}_q^{256})^k$
  $\mathbf{r}[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(r, N))$ ▷ $\mathbf{r}[i] \in \mathbb{Z}_q^{256}$ sampled from CBD
  $N \leftarrow N + 1$
end for
for $(i \leftarrow 0;\ i < k;\ i{+}{+})$ ▷ generate $\mathbf{e_1} \in (\mathbb{Z}_q^{256})^k$
  $\mathbf{e}_1[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$ ▷ $\mathbf{e}_1[i] \in \mathbb{Z}_q^{256}$ sampled from CBD
  $N \leftarrow N + 1$
end for
$e_2 \leftarrow \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$ ▷ sample $e_2 \in \mathbb{Z}_q^{256}$ from CBD
$\hat{\mathbf{r}} \leftarrow \mathsf{NTT}(\mathbf{r})$ ▷ NTT is run k times
$\mathbf{u} \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^{\intercal} \circ \hat{\mathbf{r}}) + \mathbf{e_1}$ ▷ NTT^{-1} is run k times
$\mu \leftarrow \mathsf{Decompress}_1(\mathsf{ByteDecode}_1(m))$
$v \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^{\intercal} \circ \hat{\mathbf{r}}) + e_2 + \mu$ ▷ encode plaintext m into polynomial v.
$c_1 \leftarrow \mathsf{ByteEncode}_{d_u}(\mathsf{Compress}_{d_u}(\mathbf{u}))$ ▷ $\mathsf{ByteEncode}_{d_u}$ is run k times
$c_2 \leftarrow \mathsf{ByteEncode}_{d_v}(\mathsf{Compress}_{d_v}(v))$
return $c \leftarrow (c_1 \| c_2)$

The finished K-PKE algorithm

Algorithm 14 K-PKE.Decrypt($\mathsf{dk}_{\mathrm{PKE}}, c$)

Uses the decryption key to decrypt a ciphertext.

Input: decryption key $\mathsf{dk}_{\mathrm{PKE}} \in \mathbb{B}^{384k}$.
Input: ciphertext $c \in \mathbb{B}^{32(d_u k + d_v)}$.
Output: message $m \in \mathbb{B}^{32}$.

$c_1 \leftarrow c[0 : 32 d_u k]$
$c_2 \leftarrow c[32 d_u k : 32(d_u k + d_v)]$
$\mathbf{u} \leftarrow \mathsf{Decompress}_{d_u}(\mathsf{ByteDecode}_{d_u}(c_1))$ ▷ $\mathsf{ByteDecode}_{d_u}$ invoked k times
$v \leftarrow \mathsf{Decompress}_{d_v}(\mathsf{ByteDecode}_{d_v}(c_2))$
$\hat{\mathbf{s}} \leftarrow \mathsf{ByteDecode}_{12}(\mathsf{dk}_{\mathrm{PKE}})$
$w \leftarrow v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^{\intercal} \circ \mathsf{NTT}(\mathbf{u}))$ ▷ NTT^{-1} and NTT invoked k times
$m \leftarrow \mathsf{ByteEncode}_1(\mathsf{Compress}_1(w))$ ▷ decode plaintext m from polynomial v
return m

Security amplification: The Fujisaki-Okamoto transformation

Theorem (Fujisaki-Okamoto (informal))

If $\mathcal{E}$ is an IND-CPA secure public-key cryptosystem, then $\mathsf{FO}(\mathcal{E})$ is an IND-CCA secure key encapsulation mechanism.

Security amplification: The Fujisaki-Okamoto transformation

Theorem (Fujisaki-Okamoto (informal))

If $\mathcal{E}$ is an IND-CPA secure public-key cryptosystem, then $\mathsf{FO}(\mathcal{E})$ *is an IND-CCA secure key encapsulation mechanism.*

Theorem 3.5 (PKE_1 **det.,** $\mathsf{OW\text{-}VA} \overset{\mathrm{ROM}}{\Rightarrow} \mathsf{KEM}_m^{\perp}$ $\mathsf{IND\text{-}CCA}$**).** *If* PKE_1 *is* δ_1*-correct, then so is* $\mathsf{KEM}_m^{\perp}$*. Furthermore, assume* PKE_1 *to be rigid. Let* G *denote the random oracle that* PKE_1 *uses (if any), and let* $q_{\mathsf{Enc}_1,\mathsf{G}}$ *and* $q_{\mathsf{Dec}_1,\mathsf{G}}$ *denote an upper bound on the number of* G*-queries that* Enc_1*, resp.* Dec_1 *makes upon a single invocation. If* Enc_1 *is deterministic then, for any* $\mathsf{IND\text{-}CCA}$ *adversary* B *against* $\mathsf{KEM}_m^{\perp}$*, issuing at most* q_D *queries to the decapsulation oracle* $\textsc{Decaps}_m^{\perp}$ *and at most* q_{G}*, resp.* q_{H} *queries to its random oracles* G *and* H*, there exists an* $\mathsf{OW\text{-}VA}$ *adversary* A *against* PKE_1 *that makes at most* q_D *queries to the* $\textsc{Cvo}$ *oracle such that*

$$\mathrm{Adv}_{\mathsf{KEM}_m^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathsf{B}) \leq \mathrm{Adv}_{\mathsf{PKE}_1}^{\mathsf{OW\text{-}VA}}(\mathsf{A}) + \delta_1(q_{\mathsf{G}} + (q_{\mathsf{H}} + q_D)(q_{\mathsf{Enc}_1,\mathsf{G}} + q_{\mathsf{Dec}_1,\mathsf{G}}))$$

and the running time of A *is about that of* B*.*

(Hofheinz, Hövelmanns, Kiltz: “A Modular Analysis of the Fujisaki-Okamoto Transformation” (2017))

ML-KEM

Algorithm 15 ML-KEM.KeyGen()

Generates an encapsulation key and a corresponding decapsulation key.

Output: Encapsulation key $\mathsf{ek} \in \mathbb{B}^{384k+32}$.
Output: Decapsulation key $\mathsf{dk} \in \mathbb{B}^{768k+96}$.

$z \xleftarrow{\$} \mathbb{B}^{32}$ ▷ z is 32 random bytes (see Section 3.3)
$(\mathsf{ek}_{\text{PKE}}, \mathsf{dk}_{\text{PKE}}) \leftarrow$ K-PKE.KeyGen() ▷ run key generation for K-PKE
$\mathsf{ek} \leftarrow \mathsf{ek}_{\text{PKE}}$ ▷ KEM encaps key is just the PKE encryption key
$\mathsf{dk} \leftarrow (\mathsf{dk}_{\text{PKE}} \| \mathsf{ek} \| H(\mathsf{ek}) \| z)$ ▷ KEM decaps key includes PKE decryption key
return $(\mathsf{ek}, \mathsf{dk})$

ML-KEM

Algorithm 16 ML-KEM.Encaps(ek)

Uses the encapsulation key to generate a shared key and an associated ciphertext.

Validated input: encapsulation key $\mathsf{ek} \in \mathbb{B}^{384k+32}$.
Output: shared key $K \in \mathbb{B}^{32}$.
Output: ciphertext $c \in \mathbb{B}^{32(d_u k+d_v)}$.

$m \xleftarrow{\$} \mathbb{B}^{32}$ ▷ m is 32 random bytes (see Section 3.3)
$(K, r) \leftarrow G(m \| H(\mathsf{ek}))$ ▷ derive shared secret key K and randomness r
$c \leftarrow$ K-PKE.Encrypt(ek, m, r) ▷ encrypt m using K-PKE with randomness r
return (K, c)

ML-KEM

Algorithm 17 ML-KEM.Decaps(c, dk)

Uses the decapsulation key to produce a shared key from a ciphertext.

Validated input: ciphertext $c \in \mathbb{B}^{32(d_u k + d_v)}$.
Validated input: decapsulation key $\mathsf{dk} \in \mathbb{B}^{768k+96}$.
Output: shared key $K \in \mathbb{B}^{32}$.

1: $\mathsf{dk}_{\mathrm{PKE}} \leftarrow \mathsf{dk}[0 : 384k]$ ▷ extract (from KEM decaps key) the PKE decryption key
2: $\mathsf{ek}_{\mathrm{PKE}} \leftarrow \mathsf{dk}[384k : 768k + 32]$ ▷ extract PKE encryption key
3: $h \leftarrow \mathsf{dk}[768k + 32 : 768k + 64]$ ▷ extract hash of PKE encryption key
4: $z \leftarrow \mathsf{dk}[768k + 64 : 768k + 96]$ ▷ extract implicit rejection value
5: $m' \leftarrow$ K-PKE.Decrypt$(\mathsf{dk}_{\mathrm{PKE}}, c)$ ▷ decrypt ciphertext
6: $(K', r') \leftarrow G(m' \| h)$
7: $\bar{K} \leftarrow J(z \| c, 32)$
8: $c' \leftarrow$ K-PKE.Encrypt$(\mathsf{ek}_{\mathrm{PKE}}, m', r')$ ▷ re-encrypt using the derived randomness r'
9: **if** $c \neq c'$ **then**
10: $\quad K' \leftarrow \bar{K}$ ▷ if ciphertexts do not match, “implicitly reject”
11: **end if**
12: **return** K'

Parameter sets and key sizes

	n	q	k	η_1	η_2	d_u	d_v	required RBG strength (bits)
ML-KEM-512	256	3329	2	3	2	10	4	128
ML-KEM-768	256	3329	3	2	2	10	4	192
ML-KEM-1024	256	3329	4	2	2	11	5	256

Table 2. Approved parameter sets for ML-KEM

	encapsulation key	decapsulation key	ciphertext	shared secret key
ML-KEM-512	800	1632	768	32
ML-KEM-768	1184	2400	1088	32
ML-KEM-1024	1568	3168	1568	32

Table 3. Sizes (in bytes) of keys and ciphertexts of ML-KEM

NIST security levels

NIST cat.	As strong as	Kyber
I	AES-128	ML-KEM-512
II	SHA-256	
III	AES-192	ML-KEM-768
IV	SHA-384	
V	AES-256	ML-KEM-1024

Is ML-KEM-512 really cat. I?

Daniel J. Bernstein: You can't change the metrics halfway in. What's going on in the background here?

NIST, and many others: Hey, it's fine. Also, we want the algorithms to be usable and efficient in practice.

Is ML-KEM-512 really cat. I?

Daniel J. Bernstein: You can't change the metrics halfway in. What's going on in the background here?

NIST, and many others: Hey, it's fine. Also, we want the algorithms to be usable and efficient in practice.

Bernstein: Also, your counting is wrong, and you are exaggerating your numbers. Memory access is considerably cheaper than you calculate. You'd better abandon Kyber altogether.

NIST, and many others: Dan, you're wrong. We're right to count memory access as we do.

Is ML-KEM-512 really cat. I?

Daniel J. Bernstein:	You can't change the metrics halfway in. What's going on in the background here?
NIST, and many others:	Hey, it's fine. Also, we want the algorithms to be usable and efficient in practice.
Bernstein:	Also, your counting is wrong, and you are exaggerating your numbers. Memory access is considerably cheaper than you calculate. You'd better abandon Kyber altogether.
NIST, and many others:	Dan, you're wrong. We're right to count memory access as we do.
Bernstein:	Also, these S-unit attacks are devastating for lattices.
Our MS student:	Well, they ought to work, but we can't verify the speed.

... and then it went downhill

It seems like Dan changes his point of view – inconsistently with himself – and never "retracts" misinformation that he has published.
What could be the reason for this? I've come up with three thoughts:

1. *Daniel J. Bernstein is so incredibly egotistic, that scientific truth doesn't matter to him.*
2. *Daniel J. Bernstein is acting on behalf of a foreign (to the U.S.) intelligence agency, with the aim to undermine the work of NIST in the long run.*
3. *Daniel J. Bernstein is a shill for the NSA*

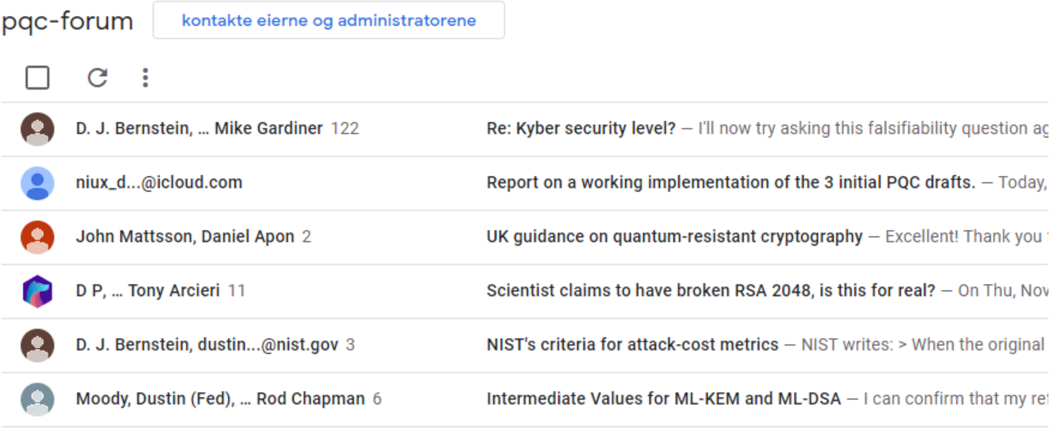
Well – to me – (2) seems not so likely. In such a scenario, surely the NSA would throw its weight around to stave off such a threat.
So, it seems to me, there are two likely cases: Either DJB is an independent actor with no regard for the truth (only to satisfy his own ego), or DJB is acting on behalf of the NSA to subvert public cryptographic standards.

Well, that escalated quickly

pqc-forum | kontakte eierne og administratorene

D. J. Bernstein, ... Mike Gard[redacted]	**Re: Kyber security level?** — I'll now try asking this falsifiability question ag
niux_d...@icloud.com	**Report on a working implementation of the 3 initial PQC drafts.** — Today,
John Mattsson, Daniel Apon 2	**UK guidance on quantum-resistant cryptography** — Excellent! Thank you
D P, ... Tony Arcieri 11	**Scientist claims to have broken RSA 2048, is this for real?** — On Thu, Nov
D. J. Bernstein, dustin...@nist.gov 3	**NIST's criteria for attack-cost metrics** — NIST writes: > When the original
Moody, Dustin (Fed), ... Rod Chapman 6	**Intermediate Values for ML-KEM and ML-DSA** — I can confirm that my re

Well, that escalated quickly



Expected adoption

Protecting Chrome Traffic with Hybrid Kyber KEM

Thursday, August 10, 2023

Teams across Google are working hard to prepare the web for the migration to quantum-resistant cryptography. Continuing with our strategy for handling this major transition, we are updating technical standards, testing and deploying new quantum-resistant algorithms, and working with the broader ecosystem to help ensure this effort is a success.

As a step down this path, Chrome will begin supporting X25519Kyber768 for establishing symmetric secrets in TLS, starting in Chrome 116, and available behind a flag in Chrome 115. This hybrid mechanism combines the output of two cryptographic algorithms to create the session key used to encrypt the bulk of the TLS connection:

- X25519 – an elliptic curve algorithm widely used for key agreement in TLS today
- Kyber-768 – a quantum-resistant Key Encapsulation Method, and NIST's PQC winner for general encryption

Expected adoption

Cloudflare now uses post-quantum cryptography to talk to your origin server

09/29/2023

 Suleman Ahmad  Luke Valenta  Bas Westerbaan

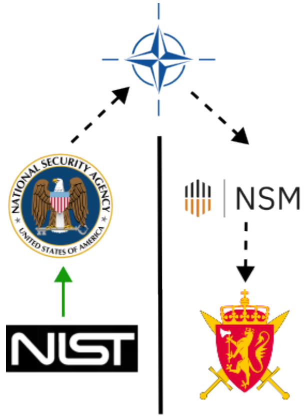
13 min read

Expected adoption

Table III: CNSA 2.0 quantum-resistant public-key algorithms

Algorithm	Function	Specification	Parameters
CRYSTALS-Kyber	Asymmetric algorithm for key establishment	TBD	Use Level V parameters for all classification levels.
CRYSTALS-Dilithium	Asymmetric algorithm for digital signatures	TBD	Use Level V parameters for all classification levels.

Expected adoption



FFI
Norwegian Defence
Research Establishment

FFI turns knowledge and ideas into an effective defence