V2: The Kyber PKE and KEM

Kyber and Dilithium

© *Alfred Menezes*

August 2024

Kyber

- Kyber is a quantum-safe Key Encapsulation Mechanism (KEM).
- Standardized by NIST in FIPS 203, where it is called ML-KEM (Module-Lattice-based KEM).
- Kyber-KEM was designed by applying the Fujisaki-Okamoto transform to a public-key encryption scheme (Kyber-PKE).

 © Alfred Menezes

V2 outline

- V2a: Kyber-PKE (simplified)
- V2b: Optimizations
- V2c: Kyber-PKE (full scheme)
- V2d: Kyber-KEM

 © Alfred Menezes

V2a: Kyber-PKE (simplified)

1. Rounding
2. Domain parameters and key generation
3. Encryption and decryption
4. Security
5. Decryption doesn’t always work

 © Alfred Menezes

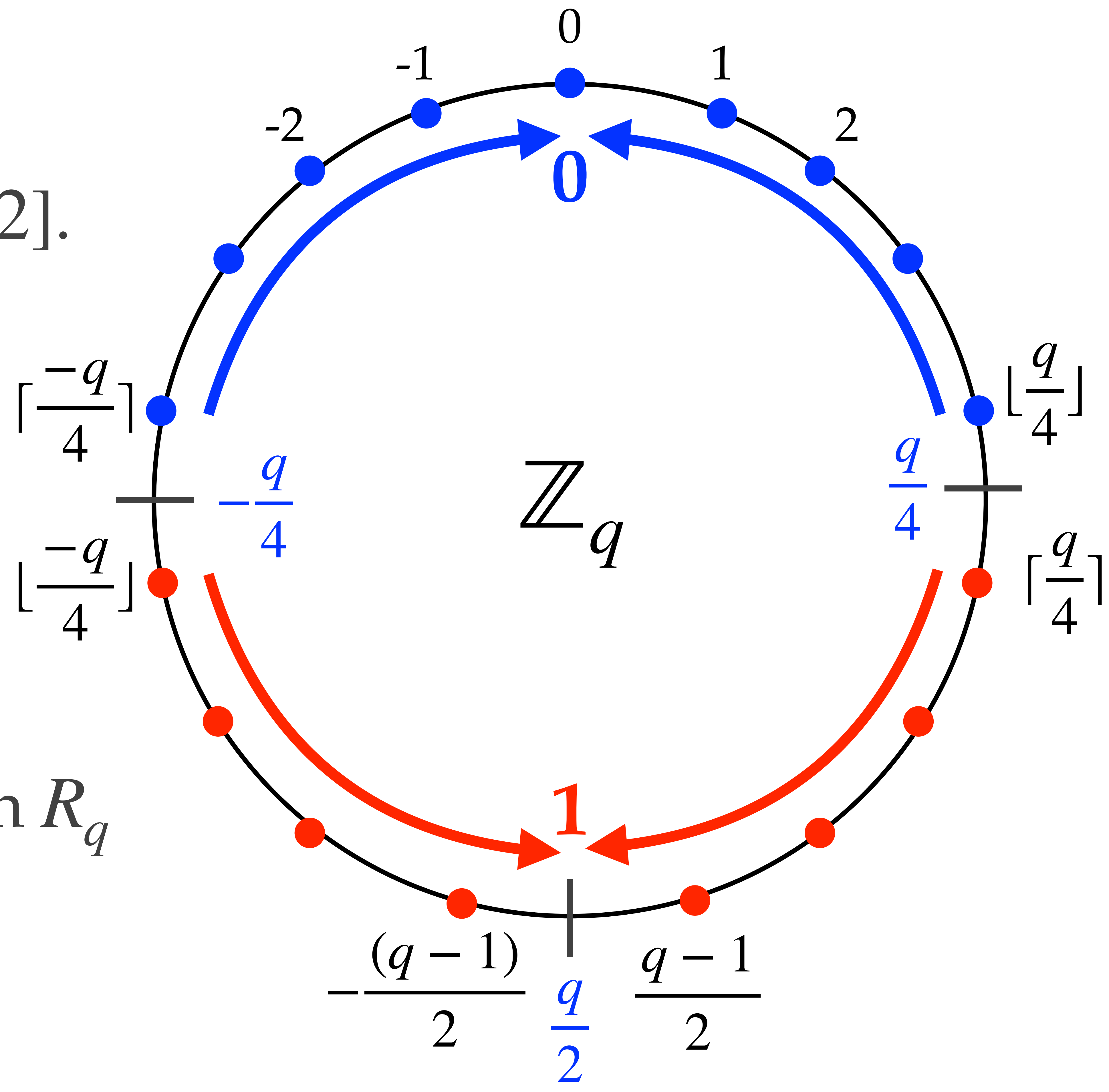
Notation

- Recall:
  - $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.
  - S_η = set of polynomials in R_q with (mods q) coefficients in $[-\eta, \eta]$.
  - R_q^k, S_η^k
- The plaintext space is $\{0,1\}^n$.
  A plaintext $m \in \{0,1\}^n$ is associated with a polynomial in R_q with 0-1 coefficients.
  - Example: If $n = 5$ and $m = 10110$, then $m \leftrightarrow m(x) = 1 + x^2 + x^3$.
- $\lfloor x \rfloor$ is the largest integer $\leq x$, and $\lceil x \rceil$ is the smallest integer $\geq x$.
  - Example: $\lfloor 5.25 \rfloor = 5$ and $\lfloor -5.25 \rfloor = -6$, whereas $\lceil 5.25 \rceil = 6$ and $\lceil -5.25 \rceil = -5$.
- $\lceil x \rfloor$ denotes the closest integer to x, with ties broken upwards.
  - Example: $\lceil 13.3 \rfloor = 13$, $\lceil 13.5 \rfloor = 14$, $\lceil 13.7 \rfloor = 14$, $\lceil -13.5 \rfloor = -13$, and $\lceil -13.7 \rfloor = -14$.

 © Alfred Menezes

Rounding

- Let q be an odd prime, and let $x \in [0, q-1]$.
- Let $x' = x \text{ mods } q$; recall that $x' \in [-(q-1)/2, (q-1)/2]$.
- Then $\text{Round}_q(x) = \begin{cases} 0, & \text{if } -q/4 < x' < q/4, \\ 1, & \text{otherwise}. \end{cases}$
  - Example: Let $q = 3329$.
    Then $\text{Round}_q(x) = \begin{cases} 0, & \text{if } -832 \le x' \le 832, \\ 1, & \text{otherwise}. \end{cases}$
- The Round_q operation can be extended to polynomials in R_q by applying it to each coefficient of the polynomial.
  - Example: Let $q = 3329$. Then
    $\text{Round}_q(3000 + 1500x + 2010x^2 + 37x^3) = x + x^2$.



© Alfred Menezes

Domain parameters and key generation

For concreteness, we'll use the ML-KEM-768 domain parameters:

- $q = 3329$
- $n = 256$
- $k = 3$
- $\eta_1 = 2$
- $\eta_2 = 2$

Kyber-PKE(s) key generation: Alice does:

1. Select $A \in_R R_q^{k \times k}$, $s \in_R S_{\eta_1}^k$, and $e \in_R S_{\eta_1}^k$.
2. Compute $t = As + e$.
3. Alice's encryption (public) key is (A, t); her decryption (private) key is s.

Note: Computing s from (A, t) is an instance of MLWE.

© Alfred Menezes

Encryption and decryption

Kyber-PKE(s) encryption: To encrypt a message $m \in \{0,1\}^n$ for Alice, Bob does:

1. Obtain an authentic copy of Alice's encryption key (A, t).
2. Select $r \in_R S_{\eta_1}^k$, $e_1 \in_R S_{\eta_2}^k$, and $e_2 \in_R S_{\eta_2}$.
3. Compute $u = A^T r + e_1$ and $v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$.
4. Output $c = (u, v)$.

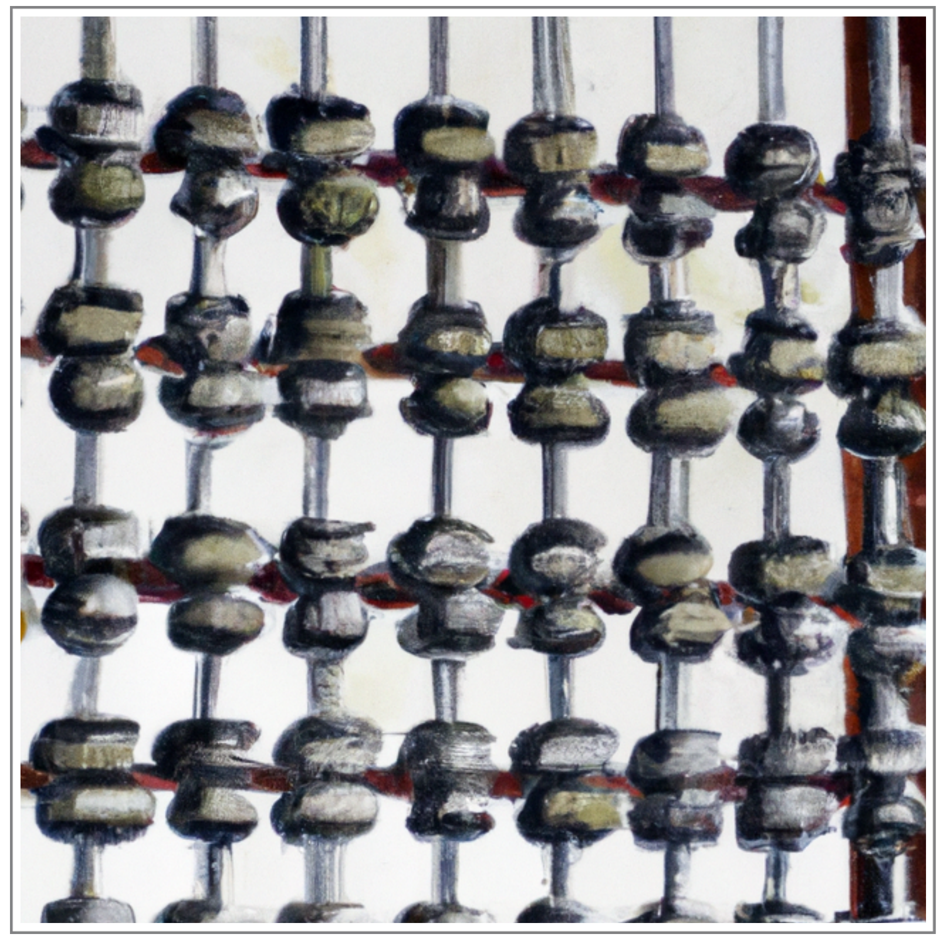
Note: $c \in R_q^k \times R_q$.

Kyber-PKE(s) decryption: To decrypt $c = (u, v)$, Alice does:

1. Compute $m = \text{Round}_q(v - s^T u)$.

Note: Alice uses her decryption key s.

Toy example: Kyber-PKE(s) (1)

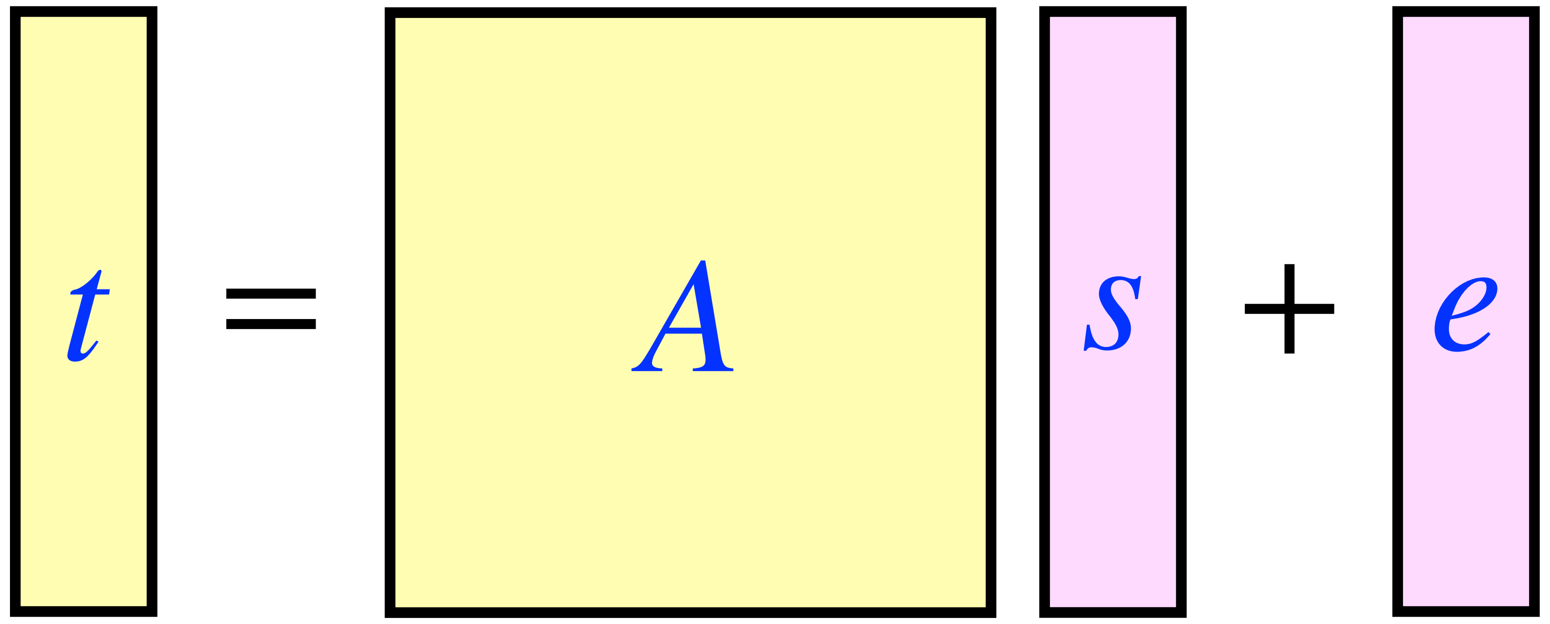


- **Domain parameters**: $q = 137,\ n = 4,\ k = 2,\ \eta_1 = 2,\ \eta_2 = 2.$
- **Key generation**: Alice selects:

$$A = \begin{bmatrix} 21 + 57x + 78x^2 + 43x^3 & 126 + 122x + 19x^2 + 125x^3 \\ 111 + 9x + 63x^2 + 33x^3 & 105 + 61x + 71x^2 + 64x^3 \end{bmatrix},$$

$$s = \begin{bmatrix} 1 + 2x - x^2 + 2x^3 \\ -x + 2x^3 \end{bmatrix},\quad e = \begin{bmatrix} 1 - x^2 + x^3 \\ -x + x^2 \end{bmatrix},$$ and computes

$$t = As + e = \begin{bmatrix} 55 + 96x + 123x^2 + 7x^3 \\ 32 + 27x + 127x^2 + 100x^3 \end{bmatrix}.$$



Alice's encryption key is (A, t); her decryption key is s.

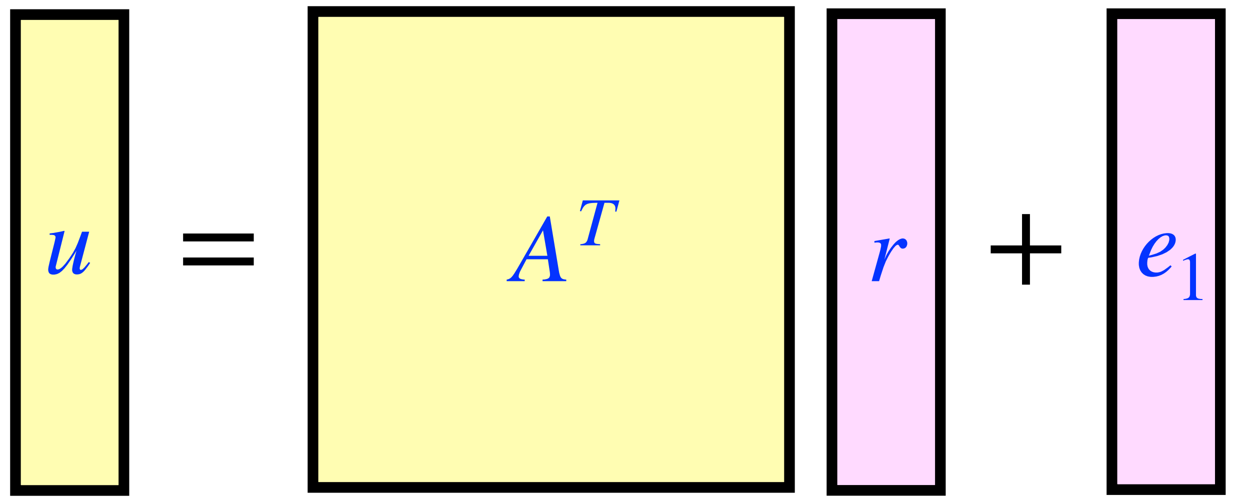
© Alfred Menezes

Toy example: Kyber-PKE(s) (2)

- Encryption: To encrypt the plaintext message $m = 0111 \leftrightarrow x + x^2 + x^3$, Bob selects

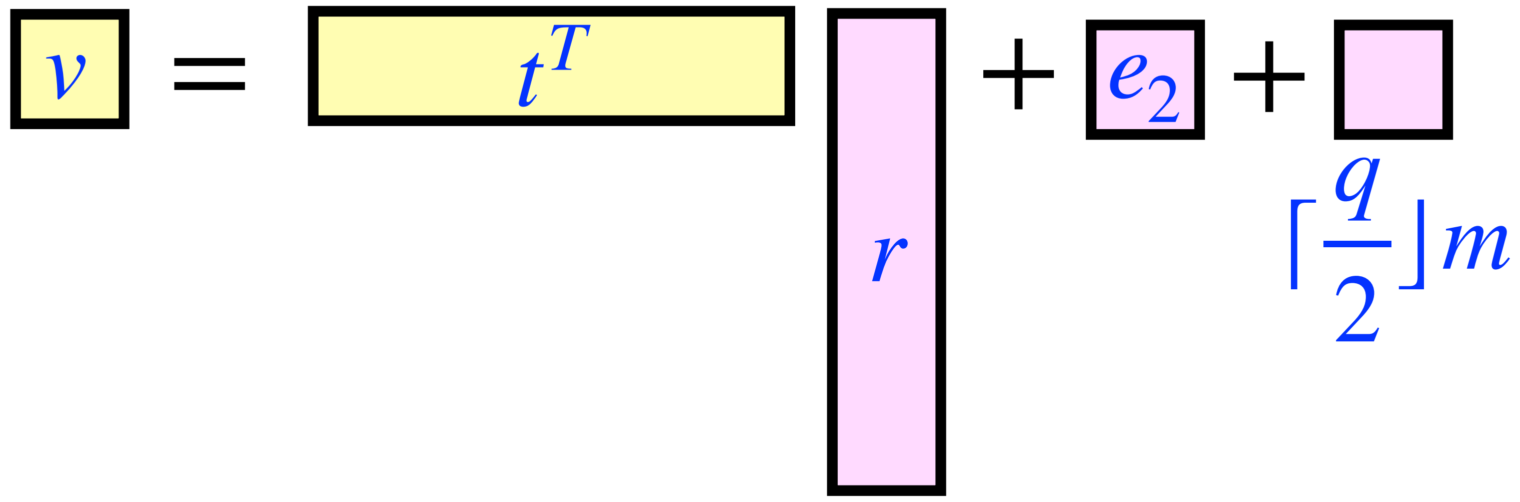
$$r = \begin{bmatrix} -2 + 2x + x^2 - x^3 \\ -1 + x + x^2 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 1 - 2x^2 + x^3 \\ -1 + 2x - 2x^2 + x^3 \end{bmatrix}, \quad e_2 = 2 + 2x - x^2 + x^3,$$

and computes $u = A^T r + e_1 = \begin{bmatrix} 56 + 32x + 77x^2 + 9x^3 \\ 45 + 21x + 2x^2 + 127x^3 \end{bmatrix}$



and $v = t^T r + e_2 + 69m = 3 + 10x + 8x^2 + 123x^3$.
The ciphertext is $c = (u, v)$.

- Decryption: To decrypt $c = (u, v)$, Alice uses her decryption key s to compute $v - s^T u = 4 + 60x + 79x^2 + 66x^3$, and then rounds its coefficients to obtain $x + x^2 + x^3$, thereby recovering the plaintext $m = 0111$.



© Alfred Menezes

Security

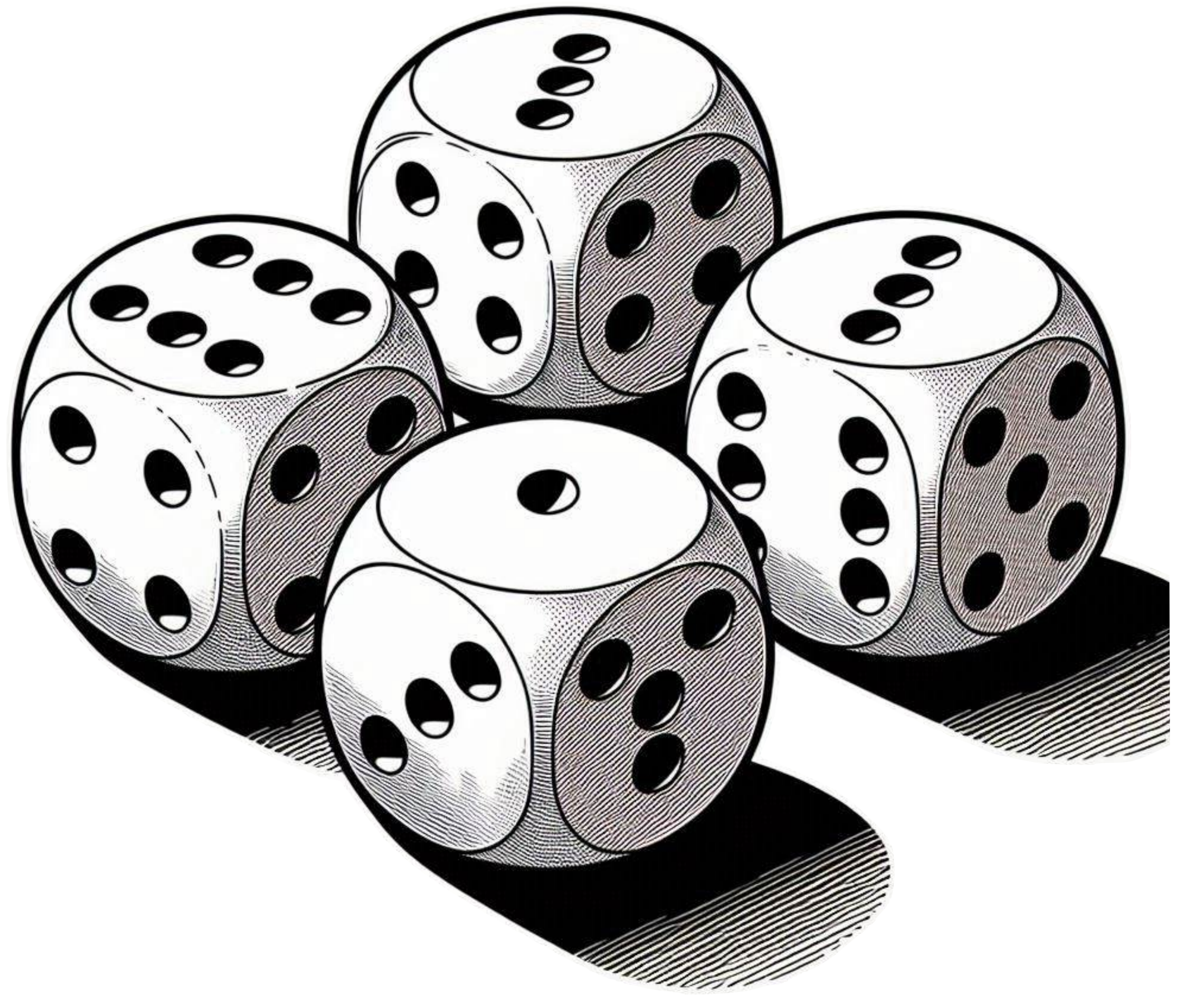
- Claim: Simplified Kyber-PKE(s) is indistinguishable against chosen-plaintext attack assuming that D-MLWE is intractable.
- Proof: The encryption operation can be written as: $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} A^T \\ t^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor m \end{bmatrix}$.

  By the D-MLWE assumption, $\begin{bmatrix} A^T \\ t^T \end{bmatrix}$ is indistinguishable from random. Again by the D-MLWE assumption, $\begin{bmatrix} A^T \\ t^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} A^T r + e_1 \\ t^T r + e_2 \end{bmatrix}$ is indistinguishable from random.

  Thus, from the adversary's perspective, v appears to be the sum of the random element $(t^T r + e_2)$ in R_q and the message polynomial $\lceil \frac{q}{2} \rfloor m$, so the adversary can learn nothing about m. $\square$

© Alfred Menezes

Decryption doesn't always work



- Question: Does decryption work? i.e., does $m = \text{Round}_q(v - s^T u)$?

- We have $v - s^T u = (t^T r + e_2 + \lceil q/2 \rfloor m) - s^T u$

$$= (s^T A^T + e^T) r + e_2 + \lceil q/2 \rfloor m - s^T (A^T r + e_1)$$
$$= s^T A^T r + e^T r + e_2 + \lceil q/2 \rfloor m - s^T A^T r - s^T e_1$$
$$= e^T r + e_2 - s^T e_1 + \lceil q/2 \rfloor m\,.$$

- Thus, $\text{Round}_q(v - s^T u) = m$ if each coefficient E_i of the error polynomial $E(x) = e^T r + e_2 - s^T e_1$ satisfies $-q/4 < E_i \text{ mods } q < q/4$, i.e., $\|E\|_\infty < q/4$.
- Now, $\|E_i\|_\infty \le kn\eta_1^2 + \eta_2 + kn\eta_1\eta_2$.
- For the ML-KEM-768 parameters ($q = 3329$, $n = 256$, $k = 3$, $\eta_1 = \eta_2 = 2$), we have $\|E_i\|_\infty \le 6146 \not< q/4$. Hence, *decryption is not guaranteed to succeed.*
- However, it can be shown that $\|E\|_\infty < q/4$ with probability extremely close to 1. Consequently, *decryption will almost certainly succeed.*

© Alfred Menezes

V2b: Optimizations

1. Smaller public keys
2. Ciphertext compression
3. Central binomial distribution
4. Fast polynomial multiplication

 © Alfred Menezes

Encryption key and ciphertext sizes

- For concreteness, we'll consider the ML-KEM-768 parameters ($q = 3329, n = 256, k = 3, \eta_1 = 2, \eta_2 = 2$).
- The bitlength of an integer in $\mathbb{Z}_q$ is $\lceil \log_2 3329 \rceil = 12$ bits.
- Encryption key: The size of an encryption key (A, t) is $(9 \times 256 \times 12) + (3 \times 256 \times 12)$ bits, or 4,608 bytes.
- Ciphertext: The size of a ciphertext $c = (u, v)$ is $(3 \times 256 \times 12) + (256 \times 12)$ bits, or 1,536 bytes.

© Alfred Menezes

Smaller encryption keys

- Idea: Generate A from a random (and public) 256-bit seed ρ.
- The polynomials in A can be generated by first selecting $\rho \in_R \{0,1\}^{256}$, and then generating the coefficients of the polynomials by hashing ρ with a counter.
- The encryption key is (ρ, t) instead of (A, t).
- Anyone who knows ρ can generate A.
- The encryption key size is now $256 + (3 \times 256 \times 12)$ bits, or 1,184 bytes (a substantial reduction from 4,608 bytes).

 © Alfred Menezes

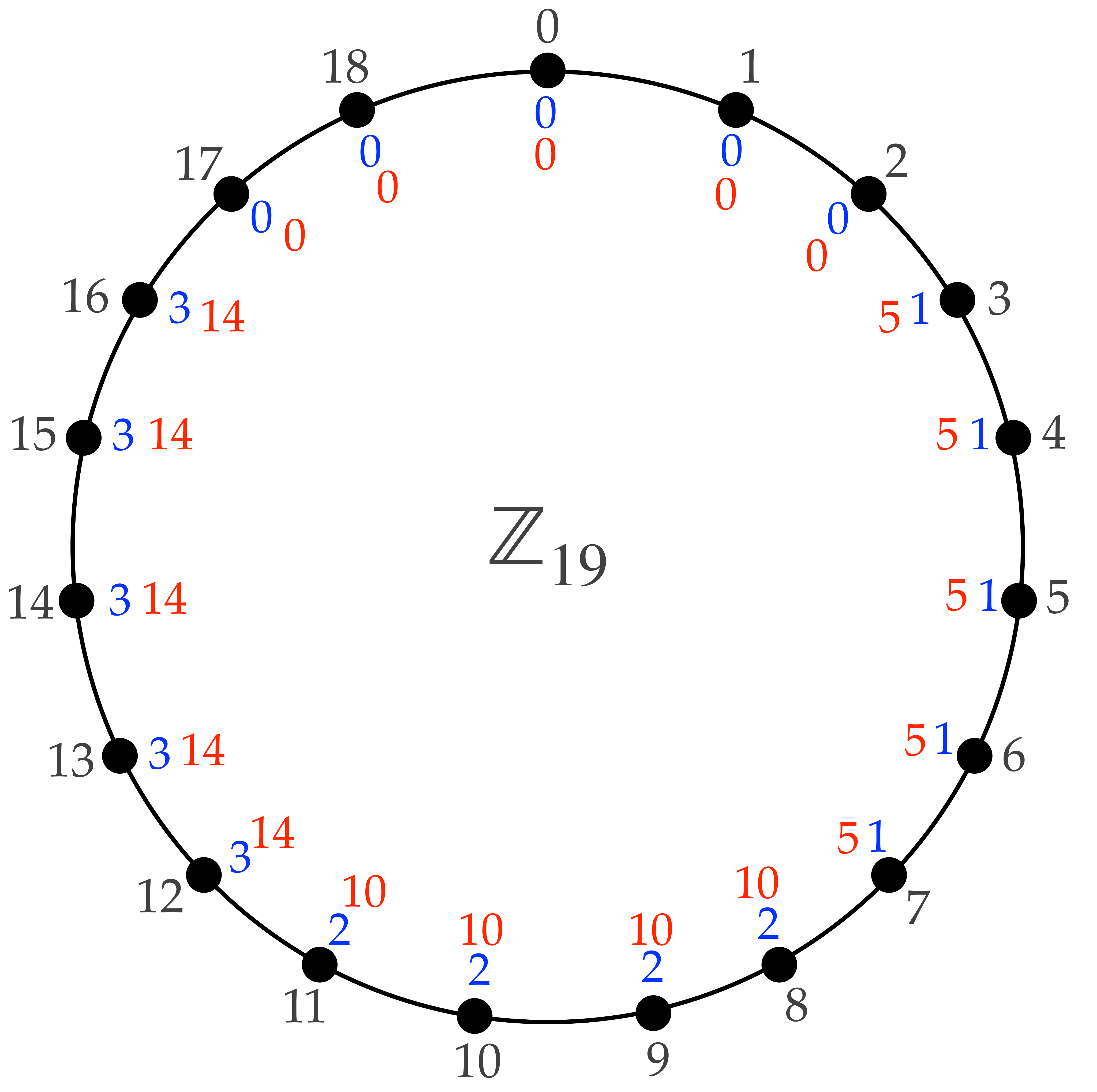
Compression

- Idea: Discard the "low order" bits of the coefficients of all polynomials in the ciphertext $c = (u, v)$.
- Let $1 \leq d \leq \lfloor \log_2 q \rfloor$, and define:
  - For $x \in [0, q-1]$, $\text{Compress}_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \bmod 2^d$.
  - For $y \in [0, 2^d - 1]$, $\text{Decompress}_q(y, d) = \lceil (q/2^d) \cdot y \rfloor \bmod q$.
- Fact: Let $x \in [0, q-1]$ and $x' = \text{Decompress}_q(\text{Compress}_q(x, d), d)$. Then $\|x' - x\|_\infty \leq \lceil q/2^{d+1} \rfloor$.
- The functions Compress and Decompress extend in the natural way to polynomials in R_q and polynomial vectors in R_q^k.

 © Alfred Menezes

Examples: compression and decompression (1)

- Let $q = 19$ and $d = 2$.
- Let $x \in [0, 18]$.
- Let $y = \text{Compress}_{19}(x, 2)$.
- Let $x' = \text{Decompress}(y, 2)$.
- Then $\|x' - x\|_\infty \le 2$.



 © Alfred Menezes

Examples: compression and decompression (2)

- Let $q = 3329$, $d = 10$, $x \in [0, q-1]$.
- Let $x' = \text{Decompress}_q(\text{Compress}_q(x, d), d)$.
- Then $|(x - x') \text{ mods } q| \leq 2$.
- Example:
  - $\text{Compress}_q(223 + 1438x + 3280x^2 + 798x^3, 10) = 69 + 442x + 1009x^2 + 245x^3$.
  - $\text{Decompress}_q(69 + 442x + 1009x^2 + 245x^3, 10) = 224 + 1437x + 3280x^2 + 796x^3$.
  - The error polynomial is $-1 + x + 2x^3$.

© Alfred Menezes

Examples: compression and decompression (3)

- Let $q = 3329$, $d = 4$, $x \in [0, q-1]$.
- Let $x' = \text{Decompress}_q(\text{Compress}_q(x, d), d)$.
- Then $|(x - x') \text{ mods } q| \leq 104$.
- Example:
  - $\text{Compress}_q(223 + 1438x + 3280x^2 + 798x^3, 4) = 1 + 7x + 4x^3$.
  - $\text{Decompress}_q(1 + 7x + 4x^3, 4) = 208 + 1456x + 832x^3$.
  - The error polynomial is $15 - 18x - 49x^2 - 34x^3$.

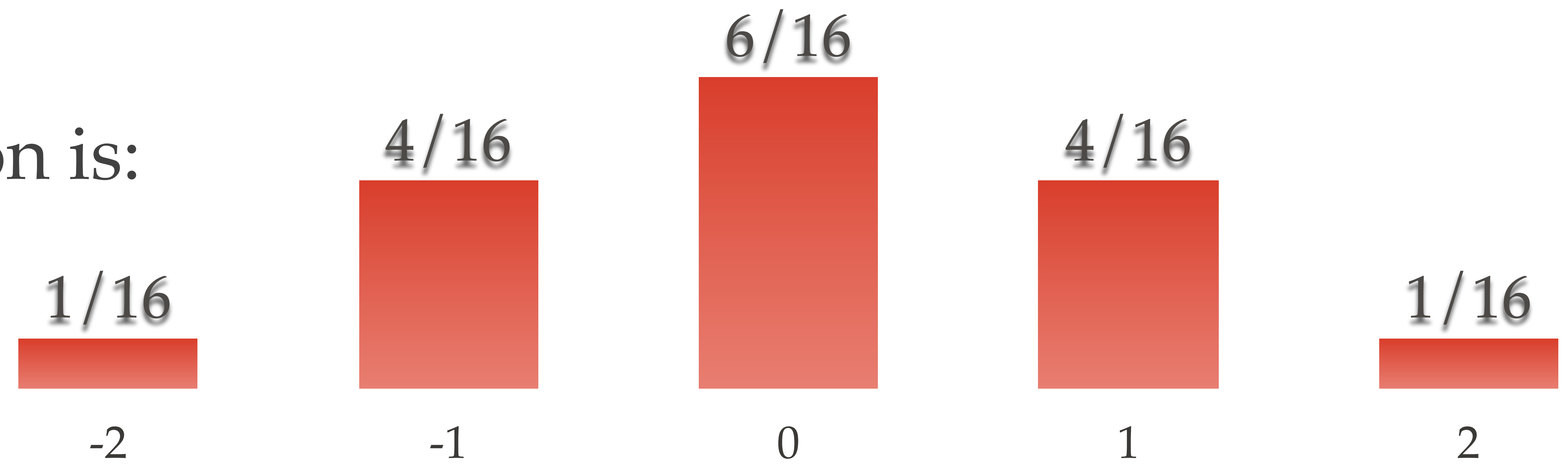
Ciphertext compression

- The ciphertext components u and v are replaced by $c_1 = \text{Compress}_q(u, d_u)$ and $c_2 = \text{Compress}_q(v, d_v)$.
- The ML-KEM-768 parameters ($q = 3329$, $n = 256$, $k = 3$, $\eta_1 = 2$, $\eta_2 = 2$) have $d_u = 10$ and $d_v = 4$.
- So, the size of the compressed ciphertext is $3 \times 256 \times 10 + 256 \times 4$ bits, or 1,088 bytes (a significant reduction from 1,536 bytes).

 © Alfred Menezes

Central binomial distribution

- Idea: A polynomial can be selected uniformly at random from S_η by selecting each of its coefficients uniformly at random from $[-\eta, \eta]$. To simplify this, the coefficients c are drawn instead according to a central binomial distribution (CBD) as follows.
- Select η pairs of bits (a_i, b_i) (with $1 \leq i \leq \eta$) uniformly at random, and output $c = \sum_{i=1}^{\eta} (a_i - b_i)$. Note that $c \in [-\eta, \eta]$.
- In fact, for each $j \in [-\eta, \eta]$, $\Pr(c = j) = \binom{2\eta}{\eta + j} / 2^{2\eta}$; this is the CBD.
- Example: For $\eta = 2$, the central binomial distribution is:



© Alfred Menezes

Fast polynomial multiplication

- The computation times for encryption and decryption is dominated by the time to multiply polynomials in $R_q = \mathbb{Z}_{3329}[x]/(x^{256}+1)$.
- The multiplication can be sped up considerably by using the Number-Theoretic Transform (NTT), which will be covered in V4.

 © Alfred Menezes

V2c: Kyber-PKE (full)

1. Domain parameters and key generation
2. Encryption and decryption
3. Decryption doesn’t always work
4. Security

 © Alfred Menezes

Domain parameters and key generation

For concreteness, we'll use the ML-KEM-768 domain parameters:

- $q = 3329$
- $n = 256$
- $k = 3$
- $\eta_1 = 2$ and $\eta_2 = 2$
- $d_u = 10$ and $d_v = 4$

Kyber-PKE key generation: Alice does:

1. Select $\rho \in_R \{0,1\}^{256}$ and compute $A = \text{Expand}(\rho)$, where $A \in R_q^{k\times k}$.
2. Select $s \in_{CBD} S_{\eta_1}^k$ and $e \in_{CBD} S_{\eta_1}^k$.
3. Compute $t = As + e$.
4. Alice's encryption (public) key is (ρ, t); her decryption (private) key is s.

© Alfred Menezes

Encryption and decryption

Kyber-PKE encryption: To encrypt a message $m \in \{0,1\}^n$ for Alice, Bob does:

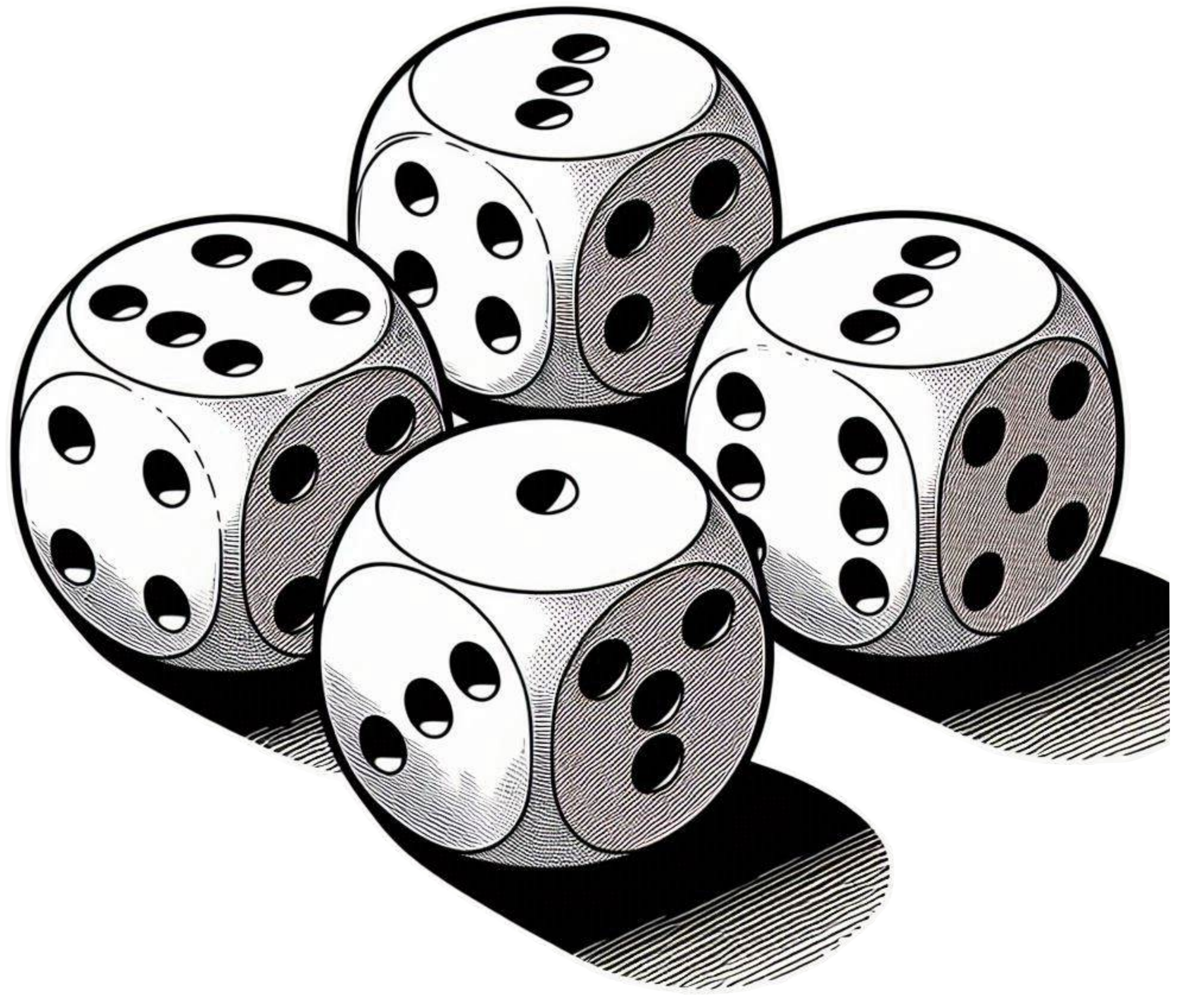
1. Obtain an authentic copy of Alice's encryption key (ρ, t) and compute $A = \text{Expand}(\rho)$.
2. Select $r \in_{CBD} S_{\eta_1}^k$, $e_1 \in_{CBD} S_{\eta_2}^k$, and $e_2 \in_{CBD} S_{\eta_2}$.
3. Compute $u = A^T r + e_1$ and $v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$.
4. Compute $c_1 = \text{Compress}_q(u, d_u)$ and
   $c_2 = \text{Compress}_q(v, d_v)$.
5. Output $c = (c_1, c_2)$.

Kyber-PKE decryption: To decrypt $c = (c_1, c_2)$, Alice does:

1. Compute
   $u' = \text{Decompress}_q(c_1, d_u)$ and
   $v' = \text{Decompress}_q(c_2, d_v)$.
2. Compute
   $m = \text{Round}_q(v' - s^T u')$.

 © Alfred Menezes

Decryption doesn't always work



- Question: Does decryption work? i.e., does $m = \text{Round}_q(v' - s^T u')$?
- Let $u' = u + e_u$ and $v' = v + e_v$.

- We have $v' - s^T u' = (v + e_v) - s^T(u + e_u)$

$$= v - s^T u + e_v - s^T e_u$$

$$= e^T r + e_2 - s^T e_1 + e_v - s^T e_u + \lceil q/2 \rfloor m\,.$$

- Thus, $\text{Round}_q(v' - s^T u') = m$ if each coefficient E_i of the error polynomial $E(x) = e^T r + e_2 - s^T e_1 + e_v - s^T e_u$ satisfies $-q/4 < E_i \text{ mods } q < q/4$, i.e., $\|E\|_\infty < q/4$.
- For the ML-KEM-768 parameters $|E_i| \not< q/4$ and hence *decryption is not guaranteed to succeed*.
- However, it can be shown that $\|E\|_\infty < q/4$ with probability extremely close to 1. Consequently, *decryption will almost certainly succeed*.

 © Alfred Menezes

Security

- Ciphertext compression doesn't affect the security of Kyber-PKE. Consequently, the following claim holds:

- Claim: Kyber-PKE is indistinguishable against chosen-plaintext attack assuming that D-MLWE is intractable.

- Note: Kyber-PKE is *not* intended for stand-alone use.

© Alfred Menezes

V2d: Kyber-KEM

1. Key encapsulation mechanisms
2. Kyber-KEM
3. Parameter sets
4. Omitted details

© Alfred Menezes

Key encapsulation mechanisms

- A key encapsulation mechanism (KEM) allows two parties to establish a shared secret key.
- A KEM is comprised of three algorithms:
  1. Key generation: Each user, say Alice, uses this algorithm to generate an encapsulation key *ek* (public key) and a decapsulation key *dk* (the private key).
  2. Encapsulation: Bob uses Alice's encapsulation key *ek* to generate a secret key K and ciphertext c, and sends c to Alice.
  3. Decapsulation: Alice uses her decapsulation key *dk* to recover K from the ciphertext c.

© Alfred Menezes

Kyber-KEM

- Kyber-KEM is derived by applying (a slight modification) of the "Fujisaki-Okamoto" (FO) transform to Kyber-PKE.
- The FO transform is a generic method for converting a public-key encryption scheme that is secure against chosen-plaintext attacks to one that is secure against chosen-ciphertext attacks.
- The transform uses three hash functions: $G : \{0,1\}^* \longrightarrow \{0,1\}^{512}$, $H : \{0,1\}^* \longrightarrow \{0,1\}^{256}$, and $J : \{0,1\}^* \longrightarrow \{0,1\}^{256}$.

© Alfred Menezes

Fujisaki-Okamoto transform

- Encapsulation: Kyber-PKE is used to encrypt a randomly selected $m \in \{0,1\}^{256}$.
  - Derandomization: m and the encapsulation key ek are hashed to produce a random seed R and the secret key K. The random polynomials r, e_1 and e_2 needed for encryption are derived from R.
- Decapsulation: The intended recipient decrypts the Kyber-PKE ciphertext c to recover m', and then hashes m' and ek to obtain R' and K'. She then re-encrypts m' (using R') and compares the resulting ciphertext c' with c.
  - If $c = c'$, she accepts K'; otherwise, she outputs a random key $\overline{K}$ (which is independent of K) obtained by hashing c and a secret z.
- Kyber-KEM has plaintext awareness, i.e., decapsulation will produce K (and not $\overline{K}$) only if the entity who performed the encapsulation already knows K.
  - This provides resistance to chosen-ciphertext attacks.

 © Alfred Menezes

Domain parameters and key generation

For concreteness, we'll use the ML-KEM-768 domain parameters:

- $q = 3329$
- $n = 256$
- $k = 3$
- $\eta_1 = 2$ and $\eta_2 = 2$
- $d_u = 10$ and $d_v = 4$

Kyber-KEM key generation: Alice does:

1. Use the Kyber-PKE key generation algorithm to select a Kyber-PKE encryption key (ρ, t) and decryption key s.
2. Select $z \in_R \{0,1\}^{256}$.
3. Alice's encapsulation key is $ek = (\rho, t)$; her decapsulation key is $dk = (s, ek, H(ek), z)$.

© Alfred Menezes

Encapsulation and decapsulation

Kyber-KEM encapsulation: To establish a shared secret key with Alice, Bob does:

1. Obtain an authentic copy of Alice's encapsulation key ek.
2. Select $m \in_R \{0,1\}^{256}$.
3. Compute $h = H(ek)$ and $(K, R) = G(m, h)$, where $K, R \in \{0,1\}^{256}$.
4. Use the Kyber-PKE encryption algorithm to encrypt m with encryption key ek, and using R to generate the random quantities needed; call the resulting ciphertext c.
5. Output the secret key K and ciphertext c.

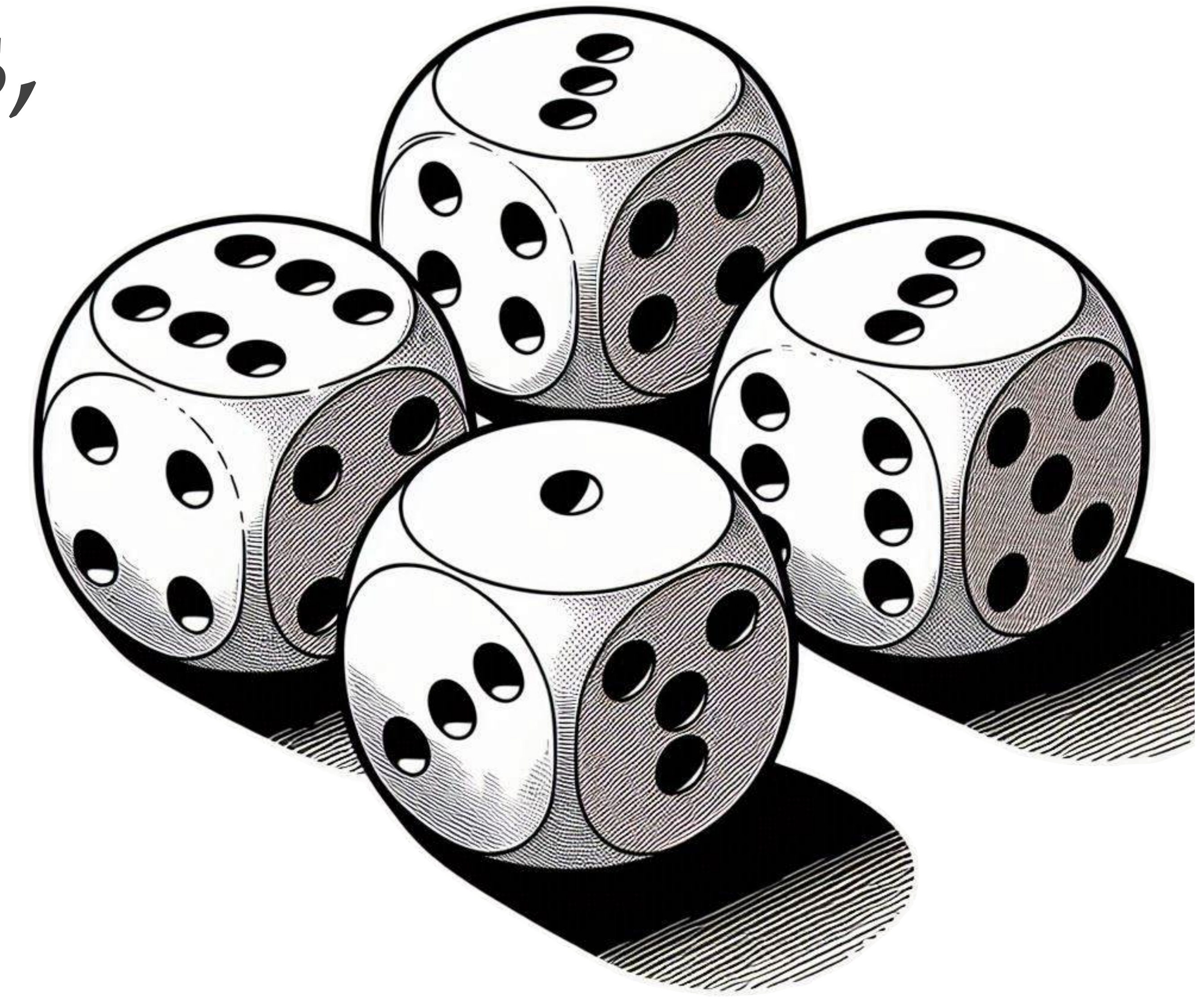
Kyber-KEM decapsulation: To recover the secret key K from c using $dk = (s, ek, H(ek), z)$, Alice does:

1. Use the Kyber-PKE decryption algorithm to decrypt c using decryption key s; call the resulting plaintext m'.
2. Compute $(K', R') = G(m', H(ek))$.
3. Compute $\overline{K} = J(z, c)$.
4. Use the Kyber-PKE encryption algorithm to encrypt m' with encryption key ek, and using R' to generate the random quantities needed; call the resulting ciphertext c'.
5. If $c \neq c'$ then return($\overline{K}$).
6. Return(K').

 © Alfred Menezes

Decapsulation failure

- Decapsulation fails when $c \neq c'$, whereby the key $\overline{K}$ that is outputted is (almost certainly) different from the key K that was encapsulated.
- This can occur even if the communicating parties, Alice and Bob, behave honestly since there is a (very small) probability that there is a failure in the underlying Kyber-PKE (whereby $m' \neq m$).



- It can be shown that the decapsulation failure rate is negligible.

© Alfred Menezes

Security

- Kyber-KEM is indistinguishable against chosen-ciphertext attacks assuming that D-MLWE is intractable, and G, H, J are random functions.
- Kyber-KEM is also indistinguishable against a chosen-ciphertext attack by a quantum adversary who is able to make both classical queries and quantum queries (in superposition) to G, H and J.

© Alfred Menezes

Parameter sets

	Security category	q	n	k	η_1	η_2	d_u	d_v	encaps. key size (bytes)	ciphertext size (bytes)	decapsulation failure rate
ML-KEM-512	1	3329	256	2	3	2	10	4	800	768	$< 2^{-139}$
ML-KEM-768	3	3329	256	3	2	2	10	4	1184	1088	$< 2^{-164}$
ML-KEM-1024	5	3329	256	4	2	2	11	5	1568	1568	$< 2^{-174}$

Categories 1, 3, 5: Fastest known attacks require at least as much resources as needed for exhaustive key search on, respectively, a 128-bit, 192-bit, and 256-bit block cipher.

© Alfred Menezes

Omitted details

FIPS PUB 202

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

- Formatting for bit strings and byte strings.
- Hash functions:
  - G is SHA3-512, H is SHA3-256, J is SHAKE256 (see FIPS 202).
- eXtendable Output Function (XOF) used to generate A.
  - SHAKE128 is used.
- PseudoRandom Function (PRF) used to generate s, e, r, e_1, e_2.
  - SHAKE256 is used.
- Number-Theoretic Transform (NTT)
  - for fast polynomial multiplication in $R_q = \mathbb{Z}_{3329}[x]/(x^{256}+1)$ (see V4b).

References

CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM

Joppe Bos[*], Léo Ducas[†], Eike Kiltz[‡], Tancrède Lepoint[§], Vadim Lyubashevsky[¶],
John M. Schanck[‖], Peter Schwabe[**], Gregor Seiler[††], Damien Stehlé[‡‡],

2018 IEEE European Symposium on Security and Privacy, pp. 353-367.

CRYSTALS-KYBER

Algorithm Specifications And Supporting Documentation
(version 3.0)

Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint,
Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé

October 1, 2020

pq-crystals.org/kyber

csrc.nist.gov/pubs/fips/203/final

NIST
Information Technology Laboratory
COMPUTER SECURITY RESOURCE CENTER

FIPS 203

Federal Information Processing Standards Publication

Module-Lattice-Based Key-Encapsulation Mechanism Standard