KYBER

By Pakin Panawattanakul

15/01/2026

PRESENTATION OUTLINE

INTRODUCTION

Presentation Objective

- Definition of Post-quantum Cryptography
- Explain basic of Kyber PKE and Kyber KEM
- Report our current progress

Post-quantum cryptography

Definition from Caltech

"**Post-quantum cryptography**, also known as quantum-proof cryptography, aims to create encryption methods that cannot be broken by algorithms, or calculations, that run on future quantum computers."

My definition

A very complex cryptographic algorithms

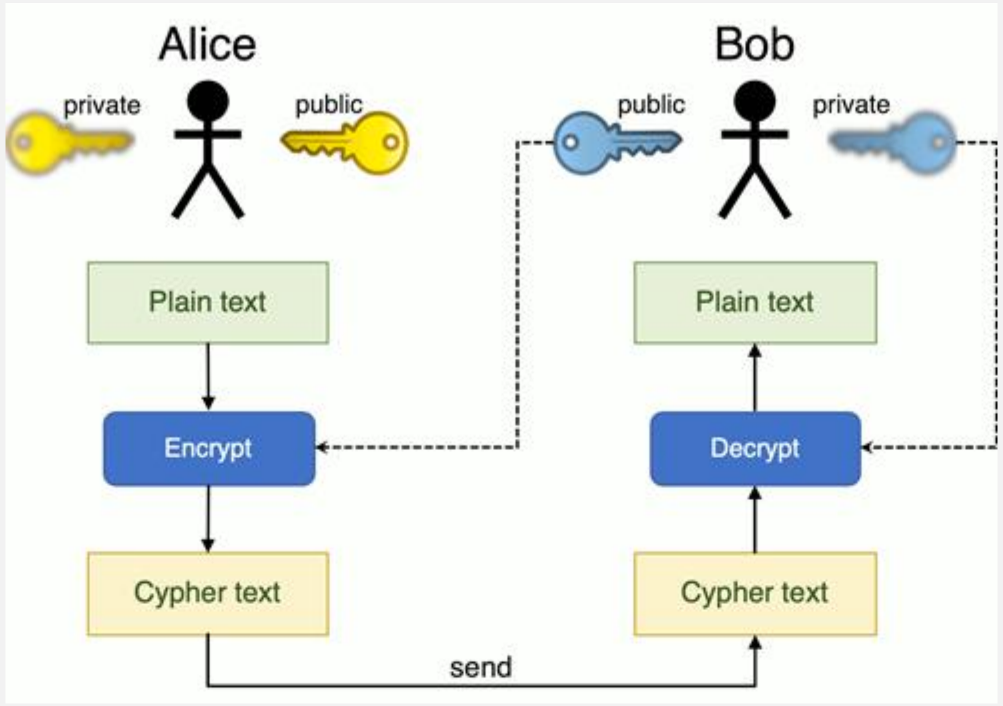
KYBER-PKE & KYBER-KEM

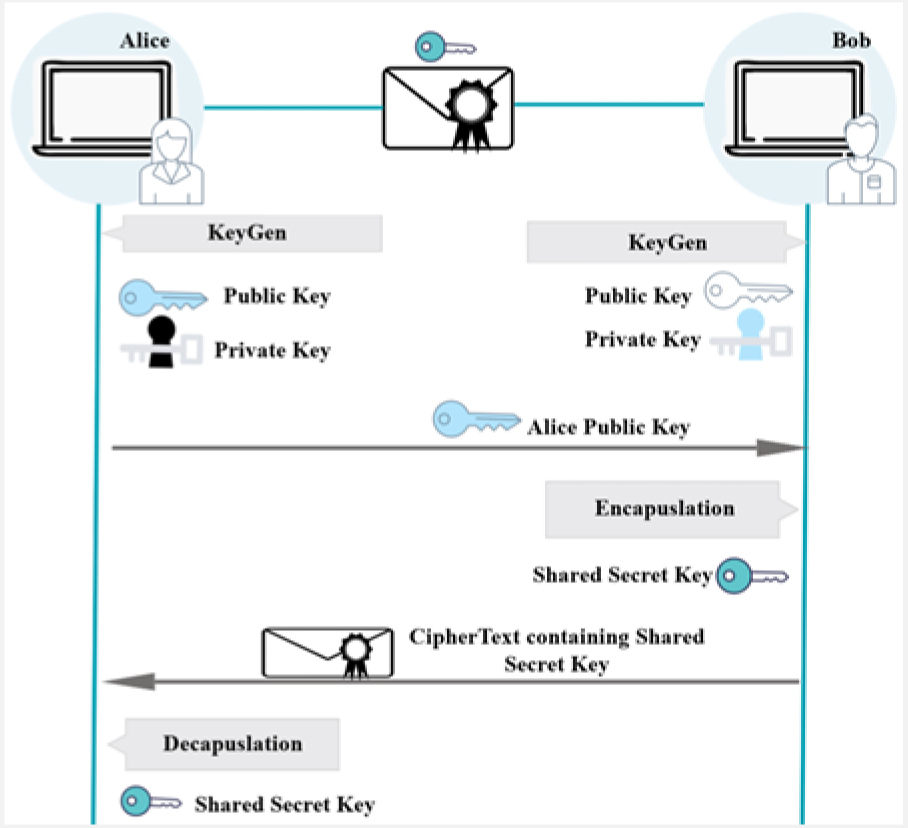
Kyber-KEM > Post-quantum

cryptographic algorithms

Chose to be standard Key encapsulation algorithm by NIST

NIST (National Institute of Standards and Technology) is a U.S. agency that develops and promotes standards for technology, including cybersecurity and cryptography.





- Public key encryption(PKE)
- Encrypted message

- Key Encapsulation Mechanism (KEM)
- Established shared secret key (symmetric key)

https://www.coengoedegebure.com/surviving-an-infosec-job-interview-cryptography/

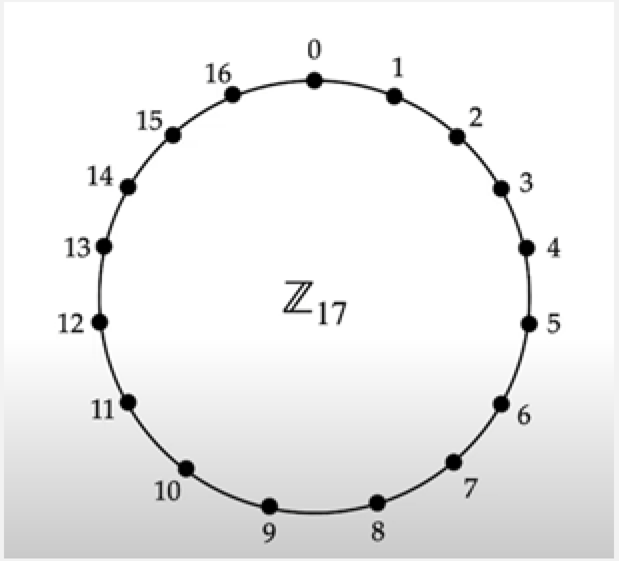
https://www.researchgate.net/publication/371388643_Resilience_Optimization_of_Post-Quantum_Cryptography_Key_Encapsulation_Algorithms

IMPORTANT MATH NOTATION

- Integer modulo $\mathbb{Z}_q$
- Polynomial ring : $R_q = \mathbb{Z}_q[x]/(x^n + 1)$
- Polynomial vector R_q^k
- “ Small” Polynomial : S_η
- bit string $\{0,1\}^n$

- Integer modulo : $\mathbb{Z}_q$
  - Integer 0,1,2,…, q-1
  - After math operation → mod q

e.g. 15+3 = 18%17 = 1



- Polynomial ring : $R_q = \mathbb{Z}_q[x]/(x^n + 1)$
  - q , n
  - Coefficient in $\mathbb{Z}_q \in \{0,1,2,\dots,q-1\}$
  - Degree = $n-1$ → $a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$
  - After multiplying apply modular reduction $/(x^n + 1)$

e.g. q = 17, n = 4

f(x) = $2 + 16x + 3x^2 + 5x^3$

g(x) = $9 + x + 14x^3$

$$f(x)g(x) = 18 + 146x + 43x^2 + 76x^3 + 229x^4 + 42x^5 + 70x^6$$

$\Downarrow$ coef. mod q

$$= 1 + 10x + 9x^2 + 8x^3 + 8x^4 + 8x^5 + 2x^6$$

$\Downarrow$ modular reduction $/(x^n + 1)$

$$= 10 + 2x + 7x^2 + 8x^3$$

Polynomial vector R_q^k

- Matrix size k ;   e.g. k = 3
- Entries are polynomial ring R_q

$$\begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{k\times 1}$$

Polynomial vector $R_q^{k\times k}$

- Matrix size 3 × 3 ; e.g. k = 3
- Entries are polynomial ring R_q

$$\begin{bmatrix} R_q & R_q & R_q \\ R_q & R_q & R_q \\ R_q & R_q & R_q \end{bmatrix}_{k\times k}$$

$$R_q = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$$

" Small" Polynomial : S_η

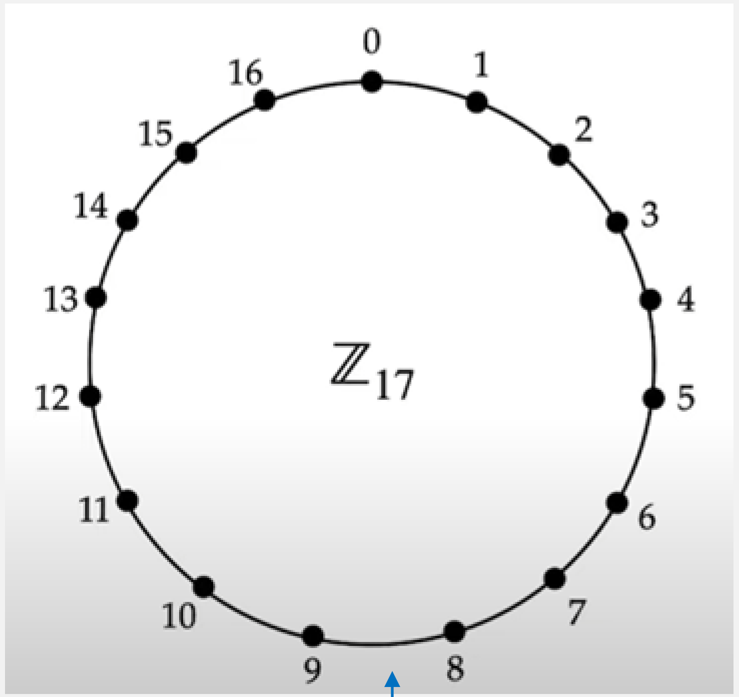
- η (eta) : coefficient $\in [-\eta, \eta]$ when written in symmetric mod
- e.g. $q = 17,\ n = 4, \eta = 2$
- Coefficient can be -2, -1, 0, 1, 2

$$s(x) = 1 + x - 2x^3$$

Noted : $s(x)$ is written is symmetric mods

$15x^3$

Symmetric mod □ mods



$(q-1)/2$

Bit string

- Bits string $\{0,1\}^n$
- e.g. 01101101….0 ← n bits long

Kyber-768 domain Parameters

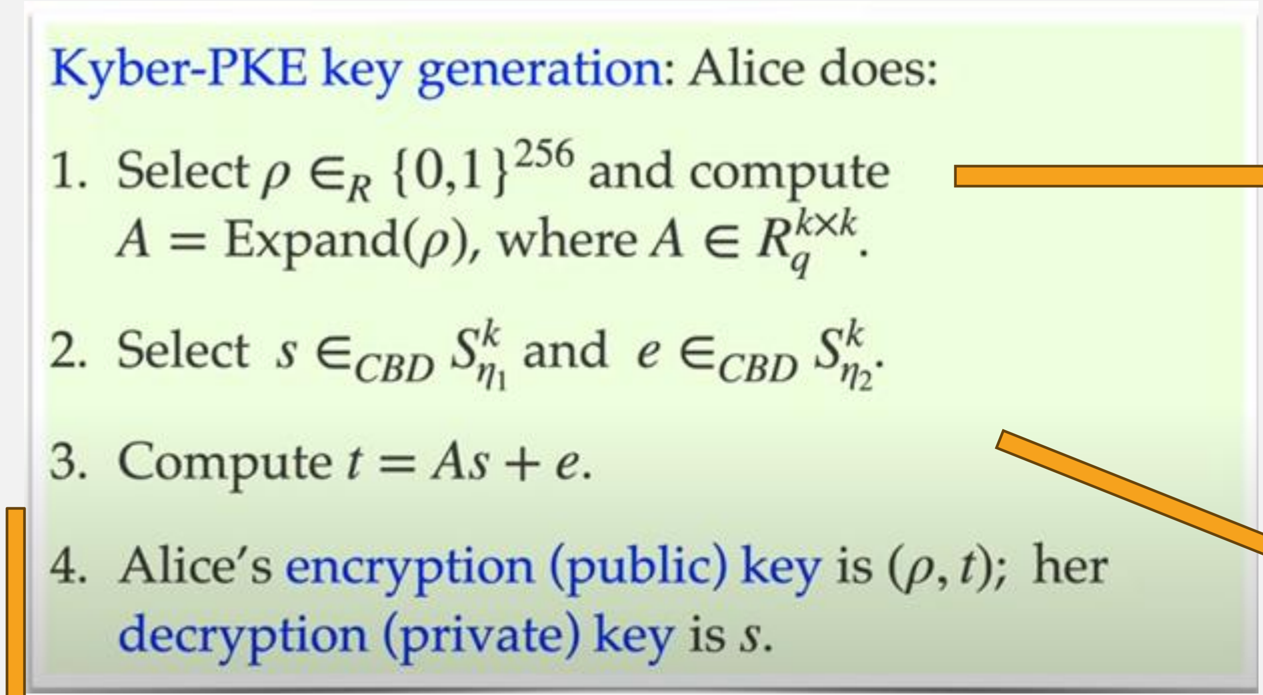
$q = 3329$	Polynomial coefficient can be : 0,1,2,…3328
$n = 256$	Each polynomial has degree 255 : $a_0 + a_1x + \cdots + a_{255}x^{255}$
$k = 3$	For matrix size
$\eta_1 = 2$	size of "small" polynomial
$\eta_2 = 2$	size of "small" polynomial
$d_u = 10$	for compression/decompression of ciphertext
$d_v = 4$	for compression/decompression of ciphertext

KYBER-PKE

- Key generation
- Encryption
- Decryption

Hash function: generate A : SHAKE 128

- Base on Learning with errors problem
- Public key encryption
- Security level : “secure against chosen plain text attack” □ lower than Kyber-KEM



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 $e \in_{CBD} S_{\eta_2}^k$.
3. Compute $t = As + e$.
4. Alice's encryption (public) key is (ρ, t); her decryption (private) key is s.

1

$\rho = \{0,1\}^{256}$ 256 bits string

A = Hash(ρ)

$$A = \begin{bmatrix} R_q & R_q & R_q \\ R_q & R_q & R_q \\ R_q & R_q & R_q \end{bmatrix}_{3\times3}$$

2

$$s = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1} \qquad e = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1}$$

Note : coefficient size has to be small according to $\eta_1 = 2$, $\eta_2 = 2$

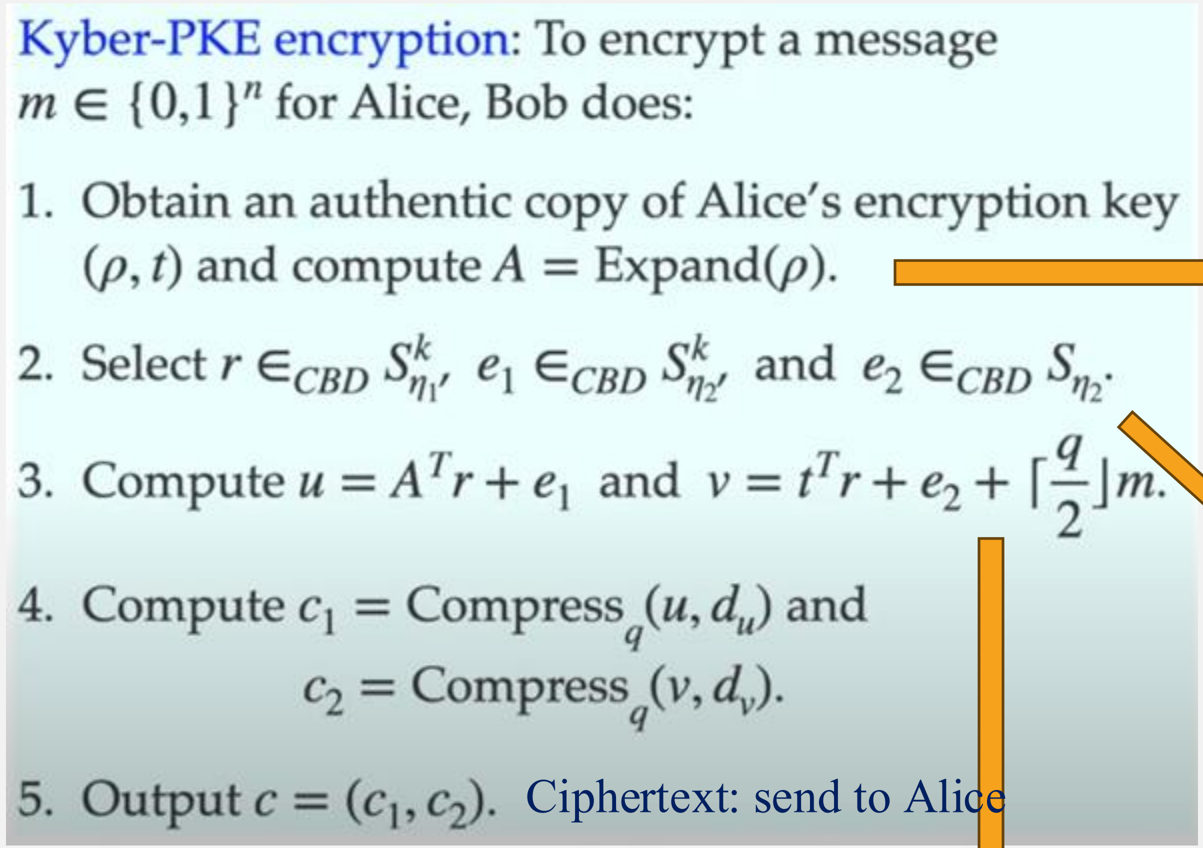
3

$$t = As + e = \left(\begin{bmatrix} R_q & R_q & R_q \\ R_q & R_q & R_q \\ R_q & R_q & R_q \end{bmatrix}_{3\times3} * \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1} \right) + \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1}$$

$$t = As + e = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1} + \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1} = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times1}$$

A As e s e

t



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 $r \in_{CBD} S^k_{\eta_1}$, $e_1 \in_{CBD} S^k_{\eta_2}$, 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 $c = (c_1, c_2)$. Ciphertext: send to Alice

1

$$m = \{0,1\}^{256}$$

$$t = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_3, \rho = \{0,1\}^{256}$$

m, ρ can be written as polynomial e.g. $\rho = x + x^3 + x^5 + \cdots + x^{255}$

2

$$r = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_3 \quad e_1 = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_3 \quad e_2 = R_q$$

R_q are small!

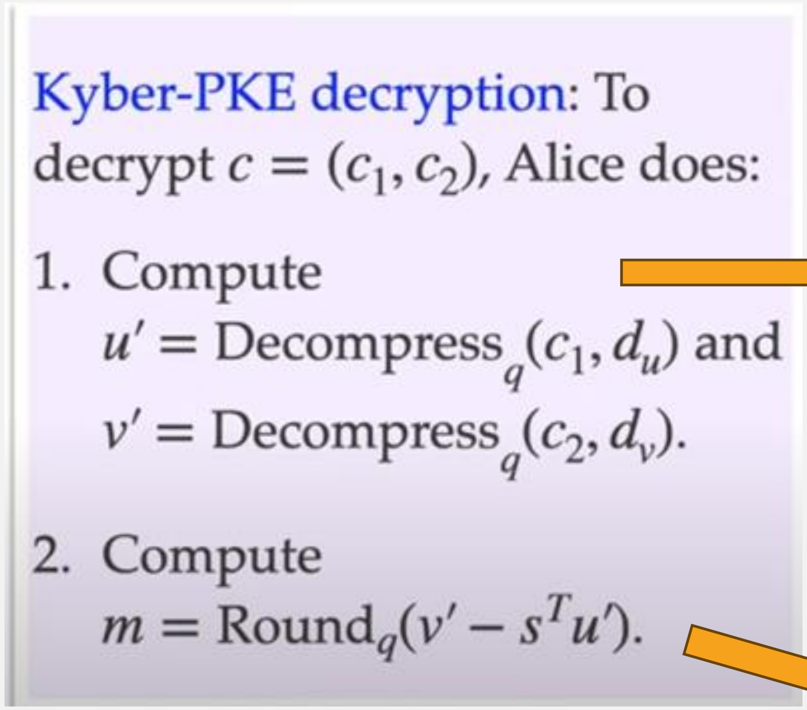
3

$$u = A^T r + e_1 = \begin{bmatrix} R_q & R_q & R_q \\ R_q & R_q & R_q \\ R_q & R_q & R_q \end{bmatrix}_{3\times 3} \times \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1} + \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1}$$

$$A^T \qquad r \qquad e_1$$

$$v = t^T r + e_2 + \lfloor q/2 \rceil m = [R_q \quad R_q \quad R_q]_{1\times 3} \times \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1} + R_q$$

$$t^T \qquad e \qquad t$$



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
   $m = \text{Round}_q(v' - s^T u')$.

1

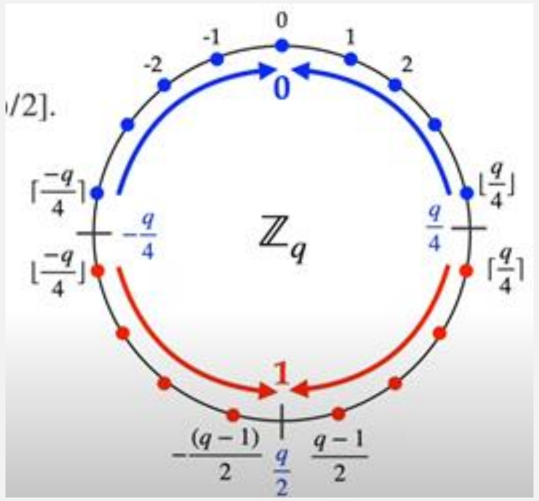
$$u' = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1}, \; v' = R_q$$

2

$$\begin{aligned} m &= Round_q(v' - s^T u') \\ &= Round_q\left(R_q - \; [R_q \quad R_q \quad R_q]_{1\times 3} \times \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1} \right) \\ &= Round_q(R_q) \rightarrow \text{get plaintext} \end{aligned}$$

Bob's original message

Rounding



/2].
0
-1
1
-2
2
0
$\lceil \frac{-q}{4} \rceil$
$\lfloor \frac{q}{4} \rfloor$
$-\frac{q}{4}$
$\frac{q}{4}$
$\mathbb{Z}_q$
$\lfloor \frac{-q}{4} \rfloor$
$\lceil \frac{q}{4} \rceil$
1
$-\frac{(q-1)}{2}$
$\frac{q}{2}$
$\frac{q-1}{2}$

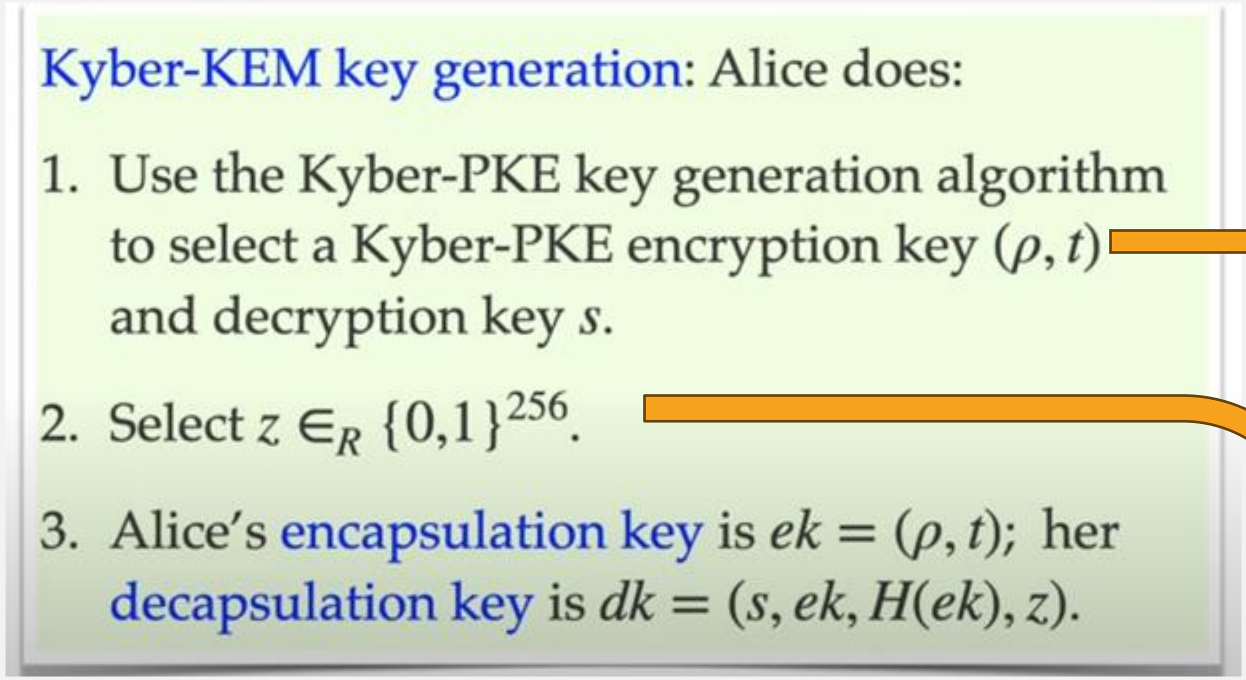
Note: there is small chance that decryption fail but the probability is very low

KYBER-KEM

- Key generation
- Encapsulation
- Decapsulation

Hash function:
generate A : SHAKE 128
G is SHA3-512, H is SHA3-256, J is SHAKE256

- Kyber PKE alone is not secure enough. Applying "Fujisaki Okamoto" transform □ Kyber KEM
- Key Encapsulation mechanism
- Using Kyber-PKE as base building block
- Use Hash() and seed to create Pseudo RNG alongside
  - pure random
  - Central binomial Distribution
- Security : "Secure against chosen ciphertext attack" (more secure than Kyber-PKE)



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 $dk = (s, ek, H(ek), z)$.

1

Follow Kyber-PKE Key-gen (page 15)

$$\rho = \{0,1\}^{256} \quad , t = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1}$$

$$s = \begin{bmatrix} R_q \\ R_q \\ R_q \end{bmatrix}_{3\times 1}$$

encryption key $ek = (\rho, t)$

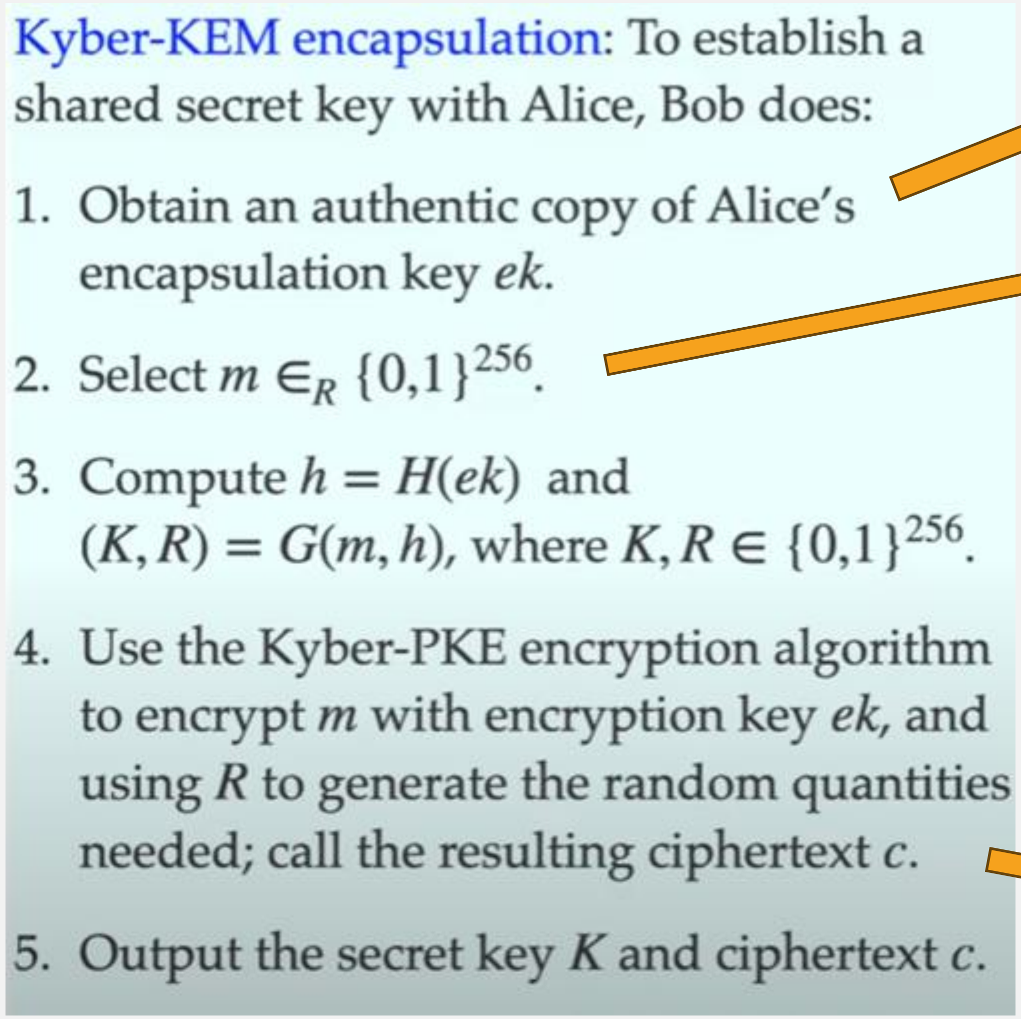
2

$$z = \{0,1\}^{256}$$

	Public key	**Private key**
Kyber-KEM		
Kyber-PKE		

Encapsulation key = Encryption key

Decapsulation key != Decryptionkey



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.

1

$ek = (\rho, t)$

2

$m = 11000101 \dots 1 \leftarrow$ 256 bits

3

Using hash function H and G

$K = 11000101 \dots 1 \leftarrow$ 256 bits string

$R = 11000101 \dots 1 \leftarrow$ 256 bits string

K is secret key

R is seed for Kyber-PKE step 4

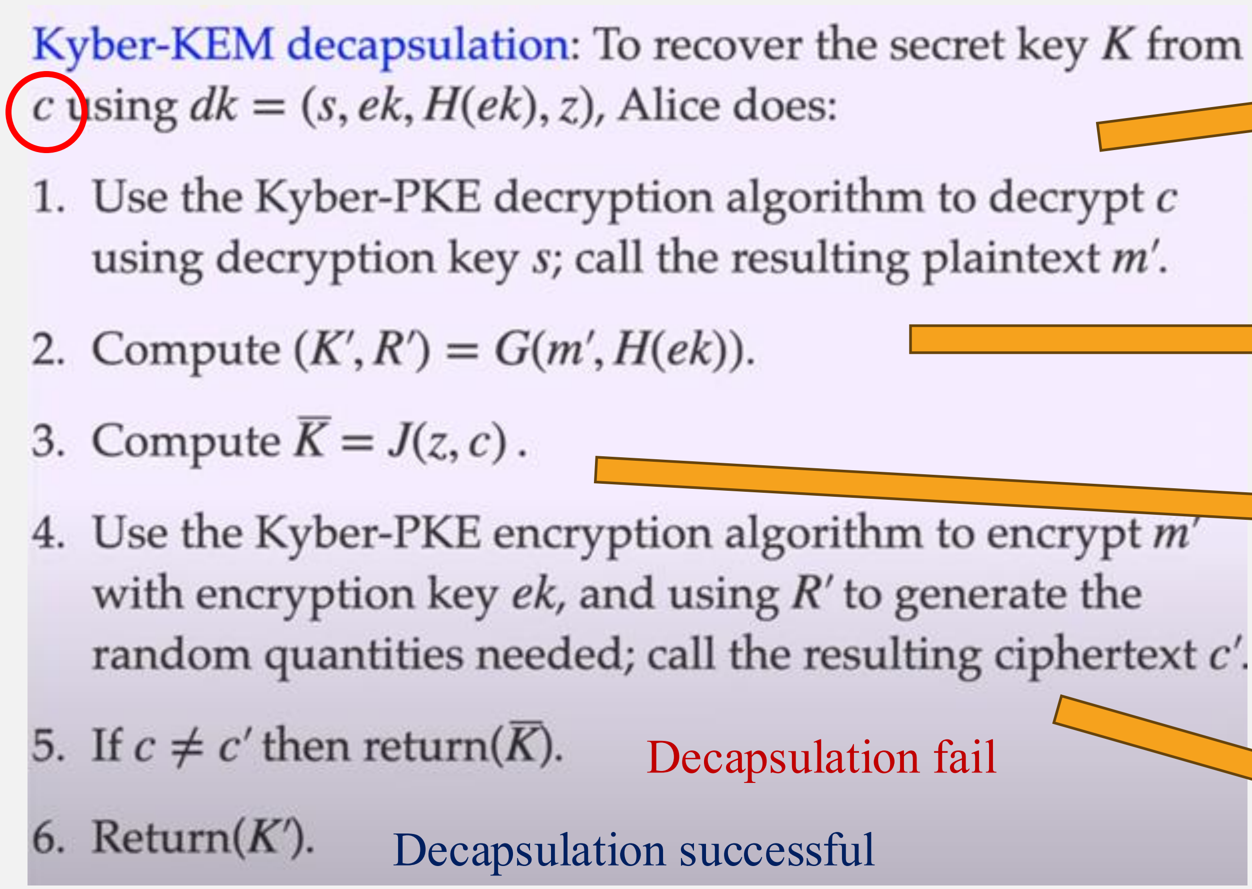
4

Use R as seed $\rightarrow r, e_1, e_2$ to encrypt m

Follow Kyber-PKE encryption (page 16)

Get ciphertext $c = (c_1, c_2)$

Send ciphertext c to Alice



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$(\overline{K})$. Decapsulation fail
6. Return(K'). Decapsulation successful

1

Follow Kyber-PKE decryption (page17)
$m' = \{0,1\}^{256}$

2

Get $K', R' = \{0,1\}^{256}$

K' is the candidate of secret key

R' is the seed use in step 4

3

$\overline{K}$ is return value when the decapsulation fail

4

Use R' as seed $\rightarrow r, e_1, e_2$ to encrypt m' again!!

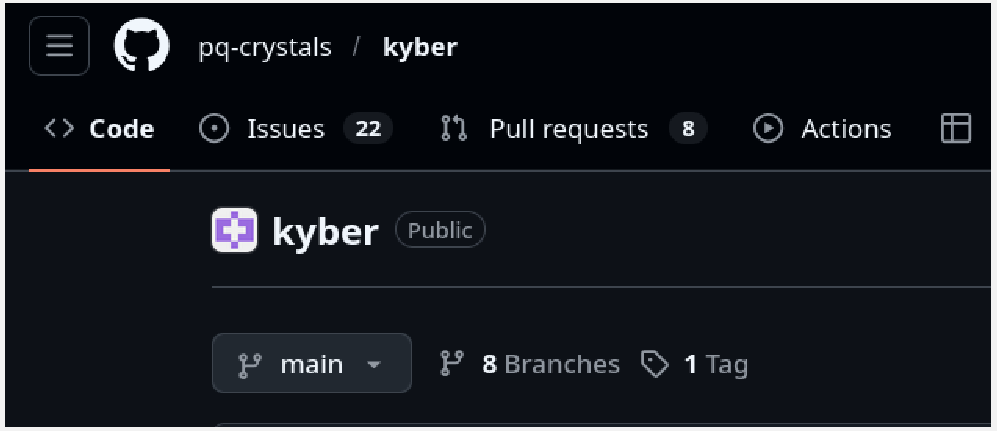
Follow Kyber-PKE encryption (page 16)

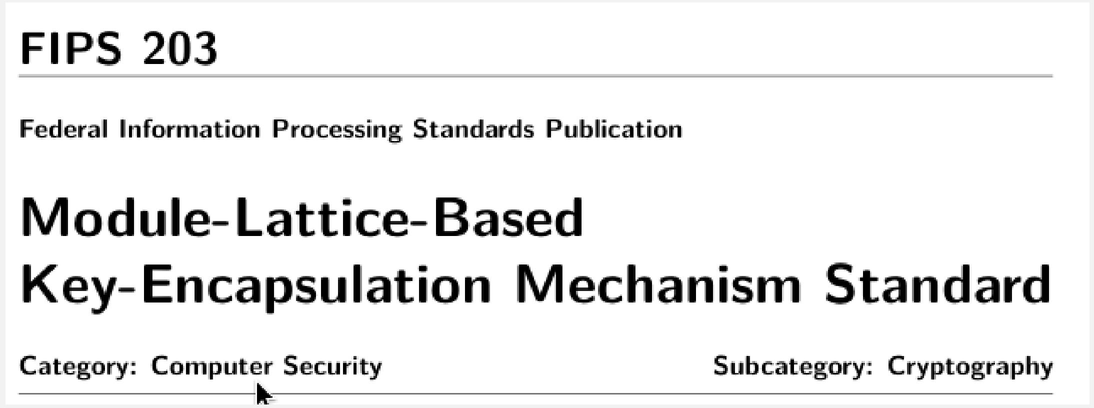
Get ciphertext $c' = (c_1, c_2)$

Current work

Design and implementation Kyber's modules

- Based on official documentations & C-implementation reference
- Design module in System Verilog
- Verification compare RTL output vs C-reference output




FIPS 203

Federal Information Processing Standards Publication

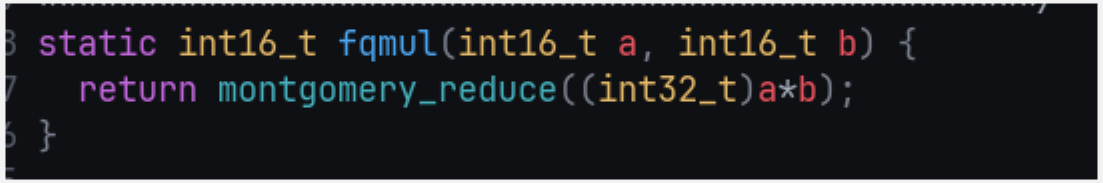
Module-Lattice-Based
Key-Encapsulation Mechanism Standard

Category: Computer Security Subcategory: Cryptography

https://github.com/pq-crystals/kyber

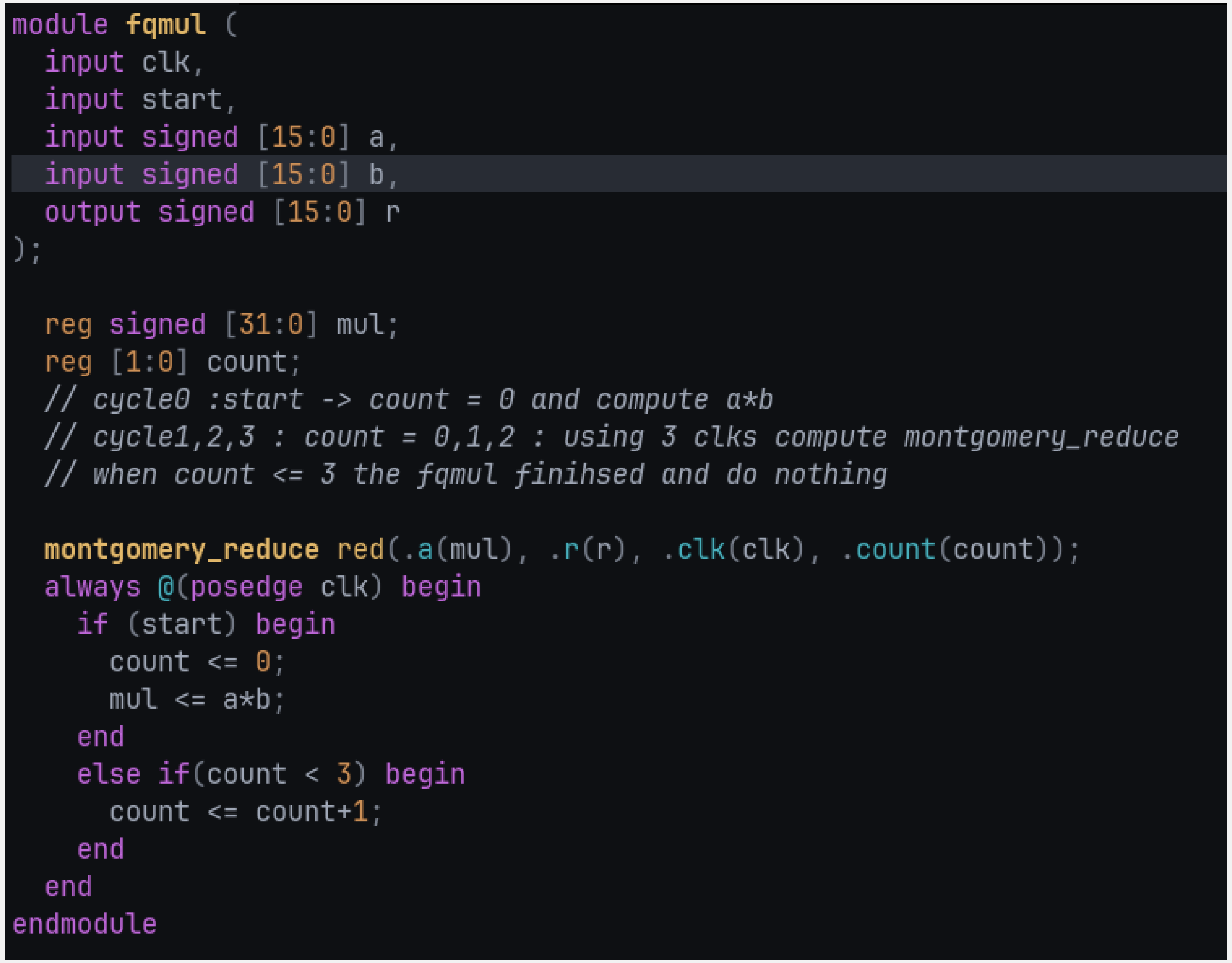
fqmul

1. multiply a*b
2. Montgomery reduction



```c
static int16_t fqmul(int16_t a, int16_t b) {
  return montgomery_reduce((int32_t)a*b);
}
```

C



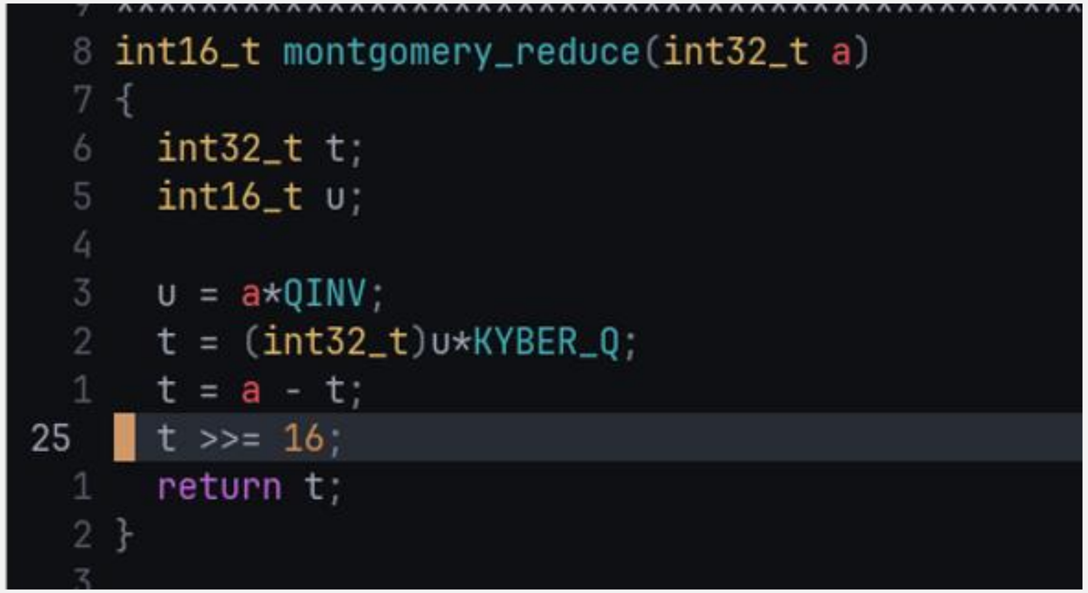
```verilog
module fqmul (
  input clk,
  input start,
  input signed [15:0] a,
  input signed [15:0] b,
  output signed [15:0] r
);

  reg signed [31:0] mul;
  reg [1:0] count;
  // cycle0 :start -> count = 0 and compute a*b
  // cycle1,2,3 : count = 0,1,2 : using 3 clks compute montgomery_reduce
  // when count <= 3 the fqmul finihsed and do nothing

  montgomery_reduce red(.a(mul), .r(r), .clk(clk), .count(count));
  always @(posedge clk) begin
    if (start) begin
      count <= 0;
      mul <= a*b;
    end
    else if(count < 3) begin
      count <= count+1;
    end
  end
endmodule
```

System Verilog

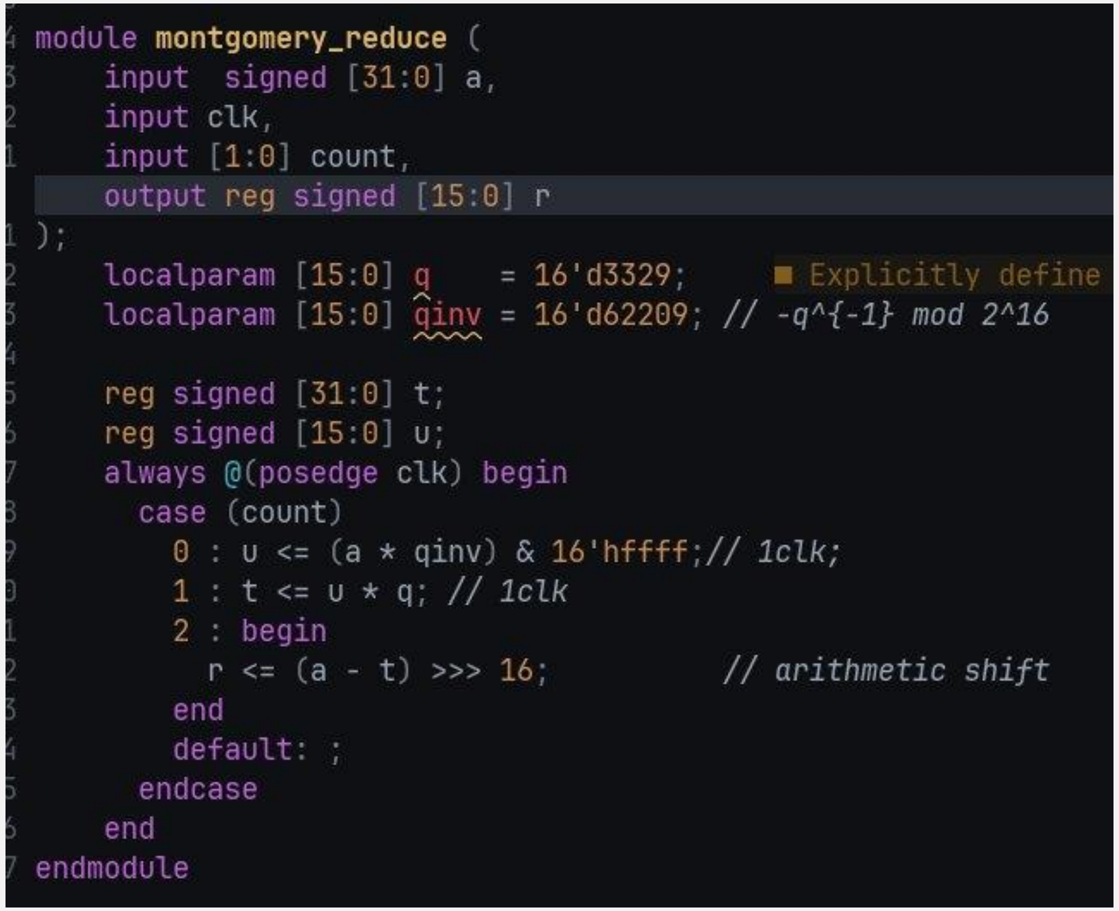
montgomery_reduce



```c
int16_t montgomery_reduce(int32_t a)
{
  int32_t t;
  int16_t u;

  u = a*QINV;
  t = (int32_t)u*KYBER_Q;
  t = a - t;
  t >>= 16;
  return t;
}
```

C



```verilog
module montgomery_reduce (
    input  signed [31:0] a,
    input clk,
    input [1:0] count,
    output reg signed [15:0] r
);
    localparam [15:0] q    = 16'd3329;      ■ Explicitly define
    localparam [15:0] qinv = 16'd62209; // -q^{-1} mod 2^16

    reg signed [31:0] t;
    reg signed [15:0] u;
    always @(posedge clk) begin
      case (count)
        0 : u <= (a * qinv) & 16'hffff;// 1clk;
        1 : t <= u * q; // 1clk
        2 : begin
          r <= (a - t) >>> 16;          // arithmetic shift
        end
        default: ;
      endcase
    end
endmodule
```

System Verilog

REFERENCES

- https://eee.poriyaan.in/topic/fpga--field-programmable-gate-arrays--11689/
- https://www.researchgate.net/figure/Digital-circuit-with-two-inputs-OR-and-AND-gates_fig11_309907692
- https://cryptography101.ca/kyber-dilithium/