



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

**TRABAJO FIN DE GRADO**

# Estudio comparativo de diferentes algoritmos de encriptación para comunicaciones industriales

Bogurad Barañski Barañska

*Cotutor:* Basil Mohammed Al-Hadithi
*Departamento:* ingeniería eléctrica, electrónica, automática y física aplicada.

*Tutor:* Roberto Gonzalez Herranz
*Departamento:* ingeniería eléctrica, electrónica, automática y física aplicada.

Madrid, Septiembre, 2025



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

**TRABAJO FIN DE GRADO**

# Título del trabajo

Firma Autor

*Firma Tutor*

**Título:** Estudio comparativo de diferentes algoritmos de encriptación para comunicaciones industriales
**Autor:** Bogurad Barañski Barañska
**Tutor:** Roberto Gonzalez Herranz

# EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

Agradezco a .....

# Resumen

Este proyecto se resume en.....

**Palabras clave:** palabraclave1, palabraclave2, palabraclave3.

# Abstract

In this project...

**Keywords:** keyword1, keyword2, keyword3.

# Índice general

# Índice de figuras

# Índice de tablas

# Siglas

| | |
|---|---|
| DFT | Transformada Discreta de Fourier. xix, 8 |
| IND-CCA2 | Indistinguibilidad bajo ataque de texto cifrado adaptable. xv, xix, 5 |
| LWE | Aprendizaje Con Errores. xvi, xix, 10 |
| NTT | Transformada Teórica de Números. xv, xvii, xix, 7–10 |

# Capítulo 1

# Introducción

## 1.1. Motivación del proyecto

En el ámbito de las comunicaciones industriales, la seguridad en el intercambio de información es un aspecto crítico, especialmente ante el avance de la computación cuántica y su impacto en los algoritmos criptográficos actuales. Este trabajo se enfoca en el análisis, implementación y evaluación de algoritmos de cifrado asimétrico post-cuántico en sistemas embebidos, con el objetivo de garantizar la seguridad de los protocolos de comunicación en un entorno industrial. Se realiza un estudio detallado de los fundamentos matemáticos de los algoritmos asimétricos clásicos y su vulnerabilidad frente al algoritmo de Shor, así como de los nuevos esquemas criptográficos diseñados para resistir ataques cuánticos.

El cifrado asimétrico es esencial para establecer claves seguras en protocolos de intercambio de claves, permitiendo así la utilización de cifrado simétrico en las comunicaciones industriales. Esto es particularmente relevante en sistemas de control en línea, donde el cifrado simétrico debe operar dentro del bucle de control en tiempo real, garantizando tanto seguridad como eficiencia.

Para ello, en este trabajo se comparan diferentes candidatos propuestos en el estándar NIST para evaluar su rendimiento en términos de velocidad, consumo de recursos y nivel de seguridad.

## 1.2. Objetivos

Para realizar el proyecto, se proponen los siguientes objetivos:

- Estudiar los fundamentos matemáticos de los métodos de cifrado asimétrico clásicos (no seguros cuánticamente) y modernos (seguros cuánticamente).
- Analizar el algoritmo de Shor y su impacto en la seguridad del cifrado asimétrico clásico.
- Implementar un sistema de comunicación entre un PC y un microcontrolador para el intercambio de claves seguras.
- Desarrollar e integrar los siguientes algoritmos de cifrado asimétrico post-cuántico en un microcontrolador y PC:

  - Kyber
  - Saber
  - HQC
  - Bike
- Comparar el rendimiento de los algoritmos de cifrado post-cuántico evaluando velocidad, consumo de recursos y seguridad.
- Estudiar la necesidad de implementar estos algoritmos en FPGAs en caso de que en el microcontrolador el rendimiento no fuera aceptable.
- Implementar y diseñar un sistema de intercambio de claves que permita la escalabilidad de la solución.

## 1.3. Herramientas utilizadas

### 1.3.1. LaTex [1]

Se ha preferido el uso de LaTeX debido a la facilidad que ofrece para el maquetado de textos, superando a otras herramientas de elaboración de documentos. Además, LaTeX permite crear figuras vectorizadas, representar correctamente ecuaciones y ubicar adecuadamente figuras, tablas y bibliografía.

### 1.3.2. TikzMaker [2]

Esta herramienta permite crear figuras vectorizadas de LaTeX mediante el paquete de circuitikz. Su principal ventaja radica en la interfaz gráfica que proporciona y en la facilidad para elaborar figuras.

### 1.3.3. C [3] y C++ [4]

### 1.3.4. Microprocesador CY8CPROTO-063-BLE [5]

## 1.4. Estructura del documento

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo:

- En el capítulo 1 se realiza una introducción.
- En el capítulo 2 se estudia trabajos realizados con relación al tema.
- En el capítulo 3 se desarrollan los fundamentos matemáticos del proyecto.
- En el capítulo 4 se describe la implementación de los algoritmos.
- En el capítulo 5 se presentan y analizan los resultados obtenidos en el capítulo anterior como consecuencia de ejecutar el software realizado.
- En el capítulo 6 se realiza una conclusión.

# Capítulo 2

# Estado del arte

Los artículos que de momento tengo que son relevantes mencionar, lo haré después de los fundamentos: [8] [9] [10] [11] [12] [13] [14] [**?**] [15] [16] [17] [18] [19]

## 2.1. Introducción

a

### 2.1.1. Tipos de cifrado

a

### 2.1.2. Necesidad del cifrado asimétrico

a

## 2.2. Sistemas de intercambio de claves

a

## 2.3. Cambio en el paradigma de cifrado asimétrico. Algoritmo de Shore

a

## 2.4. Evolución de los algoritmos postcuánticos

a

## 2.5. Cifrado asimétrico postcuántico en sistemas embebidos

a

# Capítulo 3

# Fundamentos generales

En este capítulo se desarrollan las bases matemáticas de los distintos algoritmos a implementar.

## 3.1. Introducción

## 3.2. Algoritmos de Hashing

Para elaborar esta seccion se sigue la seccion de recomendaciones del nist [20]

## 3.3. Métodos clásicos de cifrado asimétrico

### 3.3.1. Algoritmo Rivest-Shamir-Adleman (RSA)

### 3.3.2. Criptografía en Curvas Elípticas (ECC)

### 3.3.3. Algoritmo de Shore

Generico: [21] ECC: [22] RSA: [23]

## 3.4. Fundamentos de seguridad de los algoritmos

Recomendaciones NIST para intercambio seguro de claves [24]

### 3.4.1. Generación de números aleatorios criptográficamente seguros

### 3.4.2. Indistinguibilidad bajo ataque de texto cifrado adaptable IND-CCA2

### 3.4.3. Transformadas Fujisaki-Okamoto TFO

[25]

## 3.5. Funcionamiento básico de los algoritmos postcuánticos

En esta sección se describe el funcionamiento de los algoritmos postcuánticos analizados en este trabajo. Dado que no se desarrollaron implementaciones propias, sino que se utilizó el código proporcionado por el NIST en la tercera [26] y cuarta [27] ronda del proceso de estandarización, resulta apropiado presentar su funcionamiento aquí en lugar de en la sección de desarrollo.

### 3.5.1. CRYSTALS-Kyber

Se utilizará la misma notación empleada en el artículo [6]. El conjunto de los enteros sin signo de 8 bits se denota por $\mathcal{B} = \{0, \ldots, 255\}$. Para representar vectores de tamaño $k$, se utiliza la notación $\mathcal{B}^k$, mientras que para vectores de tamaño arbitrario se emplea $\mathcal{B}^*$.

Para trabajar con estos vectores, se utiliza el símbolo $||$ para denotar la concatenación de dos cadenas, y la notación $+k$ para indicar el desplazamiento de $k$ bytes desde el inicio de una cadena. Por ejemplo, si se tiene una cadena $a$ de longitud $l$ y se concatena con una cadena $b$, se obtiene:

$$c = a||b \tag{3.1}$$

Entonces:

$$b = c + l \tag{3.2}$$

Para denotar vectores, se utiliza la notación $v[i]$, donde $v$ es un vector columna e $i$ indica la posición del elemento (empezando desde 0, si no se indica lo contrario). Para las matrices, se emplea la notación $A[i][j]$, donde $i$ representa la fila y $j$ la columna. La transpuesta de una matriz $A$ se denota como $A^T$.

Se denota mediante $\lfloor x \rceil$ el redondeo de $x$ al entero más cercano. Por ejemplo: $\lfloor 2{,}3 \rceil = 2$, $\lfloor 2{,}5 \rceil = 3$ y $\lfloor 2{,}8 \rceil = 3$.

Se denota mediante $\|x\|$ al valor absoluto de x.

Para las reducciones modulares se emplean dos tipos: una centrada en cero y otra correspondiente a la reducción modular estándar. Para la reducción modular centrada en cero, sea $\alpha$ un entero par. Esta operación se define como:

$$r' = r \bmod^{\pm} \alpha \quad \Longrightarrow \quad -\frac{\alpha}{2} < r' \leq \frac{\alpha}{2} \tag{3.3}$$

Mientras que la reducción modular estándar se denota como:

$$r' = r \bmod^{+} \alpha \quad \Longrightarrow \quad 0 \leq r' < \alpha \tag{3.4}$$

Finalmente, se denota mediante $s \leftarrow S$ la selección de $s$ de manera uniformemente aleatoria del conjunto $S$. Si $S$ representa una distribución de probabilidad, entonces $s$ se selecciona de acuerdo con dicha distribución.

#### 3.5.1.1. Transformada Teórica de Números o Number Theoretic Transform (NTT)

Para acelerar las operaciones de multiplicación en el esquema basado en retículas, se utiliza la Transformada Teórica de Números (NTT, por sus siglas en inglés), la cual permite reducir la complejidad temporal de la multiplicación de polinomios desde $\mathcal{O}(n^2)$, correspondiente al método tradicional, hasta $\mathcal{O}(n\log(n))$. Para más detalles, consúltese [28].

Antes de pasar a explicar el funcionamiento de este método es relevante aclarar que se trabaja sobre el siguiente anillo de polinomios para realizar las operaciones, denotado mediante $R_q$:

$$R_q := \frac{\mathbb{Z}_q[X]}{X^n + 1} \tag{3.5}$$

En la implementación especificada de Kyber, según el artículo [6], se utiliza un valor de $q = 3329$ y $n = 256$. Esta elección es esencial para permitir el uso de la multiplicación mediante la Transformada Teórica de Números (NTT), la cual requiere que $n|(q-1)$, es decir, que $n$ divida a $(q-1)$. Esta condición garantiza la existencia de $n$ raíces enésimas de la unidad en $Z_q$, lo cual es necesario para definir la NTT. La validez de esta afirmación se fundamenta en el siguiente teorema [29]:

**Teorema 1** *Para $n$, $q > 1$, el cuerpo $Z_q$ tiene una raíz enésima de la unidad si y solo sí $n|(q-1)$*

**Demostración 1** *Si $\omega$ es una raíz enésima de la unidad en el conjunto $\mathbb{Z}_q$, entonces el conjunto:*

$$\Omega = \left\{1, \omega, \omega^2, \ldots, \omega^{n-1}\right\} \tag{3.6}$$

*forma un subgrupo cíclico $H$ del grupo multiplicativo $G_{q-1}$. Por el Teorema de Lagrange, se concluye que el orden de $H$ divide al orden de $G_{q-1}$, es decir, $n \mid (q-1)$.*

*Dado que $G_{q-1}$ es también un grupo cíclico, existe un generador $\alpha$ tal que, por el pequeño teorema de Fermat, se cumple:*

$$\alpha^{q-1} = 1 \tag{3.7}$$

*Por lo tanto, el grupo $G_{q-1}$ puede escribirse como:*

$$G_{q-1} = \left\{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\right\} \tag{3.8}$$

*Si se define $\omega$ como:*

$$\omega = \alpha^{\frac{q-1}{n}}, \tag{3.9}$$

*entonces:*

$$\omega^n = \alpha^{q-1} = 1, \tag{3.10}$$

*y además, para todo $0 < k < n$, se cumple:*

$$k \cdot \frac{q-1}{n} < q-1 \quad \Rightarrow \quad \omega^k \neq 1. \tag{3.11}$$

*Por lo tanto, $\omega$ es una raíz enésima de la unidad en $\mathbb{Z}_q$.*

Por tanto, el polinomio $X^{256}+1$ se puede factorizar sobre el cuerpo $\mathbb{Z}_q$. En la implementación concreta del esquema Kyber, este polinomio se descompone en 128 factores cuadráticos.

A este polinomio se le aplica la transformada NTT a sus coeficientes, la cual no es más que una variación de la DFT aplicada a cuerpos finitos $\mathbb{Z}_q$ y aplicada a polinomios de grado $n$. Para ello, se definen dos operaciones fundamentales:

1. La transformada directa:

$$\hat{a}_j = \sum_{i=0}^{n-1} \phi^{i(2j+1)} a_j \bmod q \tag{3.12}$$

2. La transformada inversa:

$$a_i = n^{-1} \sum_{j=0}^{n-1} \phi^{-i(2j+1)} \hat{a}_j \bmod q \tag{3.13}$$

Donde $\phi$ es un valor tal que $\phi^2 = \omega$, con $\omega$ una raíz enésima de la unidad en $\mathbb{Z}_q$ y $n^{-1}$ es la inversa multiplicativa de $n$ en $\mathbb{Z}_q$.

A continuación, se presenta un ejemplo extraído de [28] para ilustrar su funcionamiento. Sea el polinomio $G(x) = 5 + 6x + 7x^2 + 8x^3$, cuyo vector de coeficientes es $g = [5, 6, 7, 8]$. Trabajando en el anillo $\mathbb{Z}_{7681}$, y tomando $\phi = 1925$, se puede calcular la transformada NTT $\hat{g}$. Aplicando luego la transformada inversa, es posible recuperar el vector original $g$.

$$\hat{g} = \begin{bmatrix} \phi^0 & \phi^1 & \phi^2 & \phi^3 \\ \phi^0 & \phi^3 & \phi^6 & \phi^1 \\ \phi^0 & \phi^5 & \phi^2 & \phi^7 \\ \phi^0 & \phi^7 & \phi^6 & \phi^5 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 & 1925 & 3383 & 6468 \\ 1 & 6468 & 4298 & 1925 \\ 1 & 5756 & 3383 & 1213 \\ 1 & 1213 & 4298 & 5756 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 2489 \\ 7489 \\ 6478 \\ 6607 \end{bmatrix} \tag{3.14}$$

Aplicando la transformada inversa, donde la inversa de $\phi = 1925$ en $\mathbb{Z}_{7681}$ es $\phi^{-1} = 1213$ y el inverso del orden del polinomio $n = 4$ es $n^{-1} = 5761$:

$$g = n^{-1} \begin{bmatrix} \phi^0 & \phi^0 & \phi^0 & \phi^0 \\ \phi^{-1} & \phi^{-3} & \phi^{-5} & \phi^{-7} \\ \phi^{-2} & \phi^{-6} & \phi^{-2} & \phi^{-6} \\ \phi^{-3} & \phi^{-1} & \phi^{-7} & \phi^{-5} \end{bmatrix} \cdot \hat{g} = 5761 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1213 & 5756 & 6468 & 1925 \\ 4298 & 3383 & 4298 & 3383 \\ 5756 & 1213 & 1925 & 6468 \end{bmatrix} \cdot \begin{bmatrix} 2489 \\ 7489 \\ 6478 \\ 6607 \end{bmatrix} \tag{3.15}$$

Con estas transformadas definidas se define el producto o convolución negativa en el anillo $R_q \coloneqq \dfrac{\mathbb{Z}_q[X]}{X^n+1}$ entre dos polinomios $g$ y $h$ como:

$$g \cdot h = \mathrm{NTT}^{-1}(\mathrm{NTT}(g) \circ \mathrm{NTT}(h)) \tag{3.16}$$

Donde la operación $\circ$ denota la multiplicación elemento a elemento (o punto a punto) entre los vectores en $Z_q[X]$.

Ahora, queda demostrar que el tiempo de ejecución de la NTT es de $\mathcal{O}(n\log(n))$. Para lograr este cometido, se aprovechan dos propiedades que cumplen estas raíces calculadas:

1. Peridicidad:
$$\phi^{k+2n} = \phi^k \tag{3.17}$$

2. Simetría:
$$\phi^{k+n} = \phi^{-k} \tag{3.18}$$

Con estas propiedades, se puede implementar el algoritmo de Cooley-Tukey [30] que consiste en ir descomponiendo el problema en mitades de manera recursiva para reducir al máximo la cantidad de cálculos realizados. Partiendo de la transformada directa y desarrollando:

$$\hat{a}_j = \sum_{i=0}^{n-1} \phi^{i(2j+1)} a_j \bmod q = \sum_{i=0}^{n/2-1} \phi^{4ij+2i} a_{2i} + \phi^{2j+1} \sum_{i=0}^{n/2-1} \phi^{4ij+2i} a_{2i+1} \bmod q \tag{3.19}$$

Si se sustituye $A_j = \sum_{i=0}^{n/2-1} \phi^{4ij+2i} a_{2i}$ y $B_j = \sum_{i=0}^{n/2-1} \phi^{4ij+2i} a_{2i+1}$. Ahora aplicando simetría en $\phi$:

$$\hat{a}_j = A_j + \phi^{4ij+2i} B_j \bmod q \tag{3.20}$$

$$\hat{a}_{j+n/2} = A_j - \phi^{4ij+2i} B_j \bmod q \tag{3.21}$$

Donde las matrices $A_j$ y $B_j$pueden obtenerse como el resultado de aplicar la NTT sobre la mitad de los puntos, gracias a la estructura recursiva del algoritmo. Esto implica que, si $n$ es una potencia de 2, el proceso puede repetirse recursivamente sobre subproblemas de tamaño cada vez menor, hasta alcanzar el caso base.

De manera similar, se puede mostrar esta propiedad para la transformada inversa. Por tanto, se demuestra que este algoritmo tiene complejidad $\mathcal{O}(n\log(n))$.

En el caso concreto de Kyber [6], la Transformada (NTT) de un polinomio en el anillo $R_q$ se representa como un vector de 128 polinomios de grado 1.

Sean las 256 raíces enésimas de la unidad $\{\xi, \xi^3, \ldots, \xi^{255}\}$, con $\xi = 17$ como primera raíz primitiva. Entonces, se cumple que:

$$X^{256} + 1 = \prod_{i=0}^{127} \left(X^2 - \xi^{2i+1}\right) \tag{3.22}$$

Aplicando la NTT propuesta en [6] a un polinomio $f \in R_q$ se obtiene:

$$NTT(f) = \hat{f} = \left(\hat{f}_0 + \hat{f}_1 X, \hat{f}_2 + \hat{f}_3 X, ..., \hat{f}_{254} + \hat{f}_{255} X\right) \tag{3.23}$$

Con

$$\begin{aligned} \hat{f}_{2i} &= \sum_{j=0}^{127} f_{2j} \xi^{(2i+1)j} \\ \hat{f}_{2i+1} &= \sum_{j=0}^{127} f_{2j+1} \xi^{(2i+1)j} \end{aligned} \tag{3.24}$$

Donde mediante la transformada directa NTT e inversa $\text{NTT}^{-1}$ se puede realizar el producto de $f$,$g \in R_q$ de manera eficiente de la siguiente manera:

$$h = f \cdot g = \text{NTT}^{-1}\left[\text{NTT}(f) \circ \text{NTT}(g)\right] \tag{3.25}$$

Siendo $\hat{h} = \hat{f} \circ \hat{g} = \text{NTT}(f) \circ \text{NTT}(g)$ la multiplicación base definida como:

$$\hat{h}_{2i} + \hat{h}_{2i+1}X = \left(\hat{f}_{2i} + \hat{f}_{2i+1}\right)\left(\hat{g}_{2i} + \hat{g}_{2i+1}\right) \mod \left(X^2 - \xi^{2i+1}\right) \tag{3.26}$$

En la figura 3.5.1.1 se puede ver una representación gráfica de como funciona este proceso.

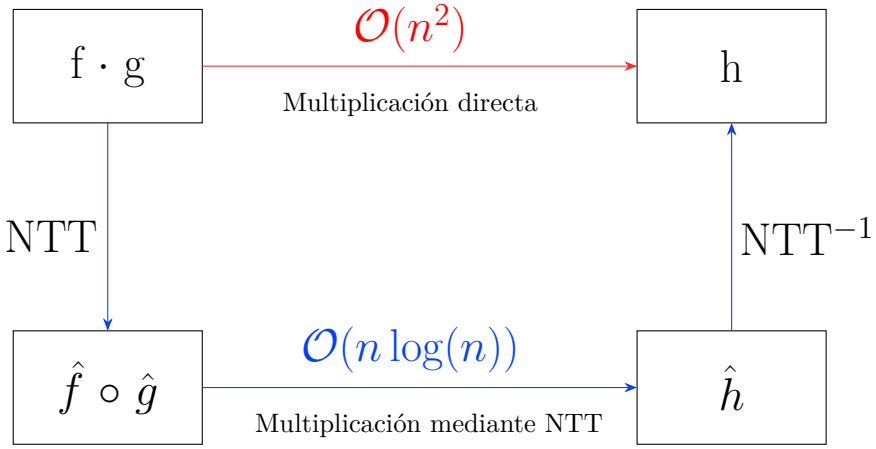


Figura 3.1: Representación del cálculo de un producto de polinomios mediante NTT en comparación con los algoritmos clásicos.

#### 3.5.1.2. Aprendizaje Con Errores o Learning With Errors (LWE)

Para la elaboración de esta sección, se ha utilizado la información contenida en el artículo [31], el cual expone los fundamentos matemáticos del problema de Aprendizaje con Errores (Learning With Errors, LWE). Este problema constituye la base teórica sobre la que se sustenta el esquema criptográfico Kyber.

Para ello, los esquemas basados en LWE trabajan con un objeto matemático conocido como retícula. Una retícula es un conjunto discreto de puntos en el espacio, generado por todas las combinaciones lineales con coeficientes enteros de un conjunto de $n$ vectores linealmente independientes que conforman una base $\mathbb{B} = \{b_1, ..., b_n\}$:

$$\mathcal{A} = \mathcal{L}(\mathbb{B}) = \left\{\sum_{i \in n} z_i b_i : z \in Z^n\right\} \tag{3.27}$$

Esta definición será útil en la demostración de la seguridad de los esquemas basados en LWE. Aun así, antes de pasar a la explicación del algoritmo de intercambio de claves públicas, es necesario definir el problema de Aprendizaje con Errores.

El Aprendizaje con Errores consiste en la tarea de distinguir entre parejas de la forma $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, donde $b_i \approx \langle a_i, s\rangle$, y parejas generados de manera completamente aleatoria. Donde, $s \in \mathbb{Z}_q^n$ es el secreto que se mantiene fijo para todas

las muestras, $a_i \in \mathbb{Z}_q^n$ es un vector elegido uniformemente al azar, y $\langle , \rangle$ denota el producto escalar Euclídeo.

La principal utilidad criptográfica del problema LWE radica en que, para quien no conoce el secreto, resulta computacionalmente difícil distinguir entre pares estructurados y pares aleatorios. En cambio, quien sí conoce el secreto puede identificar fácilmente qué parejas son consistentes con este, lo que permite construir esquemas seguros de cifrado e intercambio de claves.

#### LWE en anillos

En Kyber no se parte del problema LWE "puro", ya que los esquemas basados directamente en LWE suelen presentar claves de tamaño y tiempos de cálculo con un orden de magnitud $\mathcal{O}(n^2)$ [32]. Esto se debe a que, al trabajar con matrices genéricas, se carece de la estructura de anillo necesaria para aprovechar multiplicaciones mediante convolución rápida, es decir, la NTT. En cambio, Kyber se fundamenta en una variante conocida como Modulus-LWE (M-LWE), donde las operaciones de multiplicación matricial se pueden implementar como multiplicaciones de polinomios en $\dfrac{\mathbb{Z}_q[x]}{X^n+1}$, lo cual permite usar la NTT.

Antes de describir en detalle el funcionamiento de la M-LWE, conviene introducir primero el problema Ring-LWE (R-LWE). La M-LWE puede entenderse como una extensión modular (un vector) de R-LWE. Este enfoque permite romper la simetría inherente a un anillo y aporta mayor flexibilidad en la selección de parámetros para lograr una mejor relación entre eficiencia y resistencia criptográfica [33].

En cuanto a la eficiencia en la reducción del tamaño de las claves, esta puede observarse claramente a través del siguiente ejemplo ilustrativo:

| Esquema | LWE | R-LWE |
|---|---|---|
| Tamaño clave pública | $\mathcal{O}(n^2)$ bits | $\mathcal{O}(n)$ |
| Operación para generar la pareja | $b_i = \langle a_i, s\rangle + e_i$ | $b[x] = a[x] \cdot s[x] + e[x]$ |
| Matriz $A$ | $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ | 3+11X |
| Secreto $s$ | $\begin{pmatrix} 5 \\ 6 \end{pmatrix}$ | 5+6X |
| Error $e$ | $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ | 1+X |
| Resultado $b$ | $\begin{pmatrix} 1 \\ 6 \end{pmatrix}$ | 1+6X |

Tabla 3.1: Tabla con un ejemplo numérico del calculo del valor $b$ necesario para el problema LWE como en R-LWE para ver como efectivamente en la clave pública $(a, b)$ el tamaño es menor. En este ejemplo se trabaja en $\mathbb{Z}_{17}$ y $X^2 + 1$. En M-LWE las claves son de tamaño similar que en R-LWE pues la matriz $A$ se puede reconstruir a través de una semilla y la matriz $b$ solo ve reescalada en función de la seguridad empleada.

**Aplicación del R-LWE para el intercambio de claves públicas**

A continuación, se presenta el algoritmo principal empleado para el intercambio de claves basado en el problema R-LWE que fundamenta matemáticamente el funcionamiento de Kyber. El uso de la M-LWE es similar y se puede consultar en la sección 3.5.1.5.

El algoritmo de generación de claves emplea los valores $q$ y $n$ , que definen la estructura algebraica, para calcular la llave privada $sk$ y la llave pública $pk$.

**Algoritmo 1** Generación claves R-LWE en $\mathbb{Z}_q[x]/(X^n+1)$ [0]

**Entrada:** $q$, $n$

**Salida:** $sk$, $pk$

1: Generar $a \in R_q$ al azar según una distribución uniforme.
2: Generar $sk, e \in R_q$ como elementos pequeños según la distribución del error.
▷ Binomial en Kyber, discreta Gaussiana en otros esquemas
3: Calcular

$$b := a \cdot sk + e \tag{3.28}$$

4: **return** $(sk, pk := a||b)$

En el algoritmo de cifrado a partir de la llave pública, $pk$ y un mensaje $z \in \{0,1\}^n$ se obtienen los textos cifrados $u$ y $v$ que serán enviados al poseedor de la clave privada para descifrar el mensaje.

**Algoritmo 2** Cifrado R-LWE [0]

**Entrada:** $pk$, $z$, $q$, $n$

**Salida:** $u$, $v$

1: Generar $r, e_1, e_2 \in R_q$ como elementos pequeños según la distribución del error.
2: Calcular el primer texto cifrado $u$:

$$u := a \cdot r + e_1 \bmod^{+} \mathrm{q} \tag{3.29}$$

3: Calcular el segundo texto cifrado $v$:

$$v := b \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil \cdot z \bmod^{+} \mathrm{q} \tag{3.30}$$

4: **return** $(u, v)$

En el algoritmo de descifrado, a partir de los textos cifrados $u$ y $v$se recupera el mensaje original $z$. La seguridad del esquema radica en el término de ruido ($\varepsilon = r \cdot e - s \cdot e_1 + e_2$) que ofusca el mensaje, el cual solo puede recuperarse correctamente usando la clave secreta siempre que $\|\varepsilon\| < q/4$.

Para mantener este error dentro de niveles aceptables, es fundamental ajustar adecuadamente los parámetros de las distribuciones de ruido en el algoritmo Kyber. En la versión empleada en este trabajo (Kyber1024), esto se traduce en una tasa de error de $2^{-174}$ [34].

[0] En Kyber se usa la variante M-LWE, donde $s \in R_q^d$, $A \in R_q^{d \times d}$ y $b = A \cdot s + e \in R_q^d$; el pseudocódigo completo en M-LWE aparece en la sección 3.5.1.5.

**Algoritmo 3** Descifrado R-LWE [0]

**Entrada:** $sk$, $u$, $v$, $q$, $n$

**Salida:** $z$

1: Calcular el polinomio $z'$ a partir del cual se calcula el mensaje.

$$z' := v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \left\lfloor \frac{q}{2} \right\rceil \cdot z \bmod^{+} \mathrm{q} \tag{3.31}$$

2: Calcular la distancia de cada coeficiente ($z'_i$) de $z'$:

**1.1:** Distancia a 0:

$$\mathrm{d}_i(0) := \left\| z'_i \bmod^{\pm} \left\lfloor \frac{q}{2} \right\rceil \right\| \tag{3.32}$$

**1.2:** Distancia a $\left\lfloor \frac{q}{2} \right\rceil$:

$$\mathrm{d}_i\left(\frac{q}{2}\right) = \left\| \left\lfloor \frac{q}{2} \right\rceil - \mathrm{d}_i(0) \right\| \tag{3.33}$$

3: **if** $\mathrm{d}_i(0) < \mathrm{d}_i\left(\frac{q}{2}\right)$ **then**

4: Descifrar el bit $i$ del mensaje $z$ como un 0.

5: **else**

6: Descifrar el bit $i$ del mensaje $z$ como un 1.

7: **end if**

8: **return** $z$

En el anexo B se puede ver un breve ejemplo de aplicación numérica de funcionamiento de los algoritmos anteriores.

**Uso de M-LWE en Kyber**

A partir del siguiente articulo se define la M-LWE que es el esquema de fondo en Kyber [33].

Sean $R_q$ el anillo a trabajar y $d$ la dimensión de la retícula. Se define:

$s \in R_q^d$ (el vector secreto)

$A \in R_q^{d \times d}$ (la matriz pública, muestreada $R_q^{d \times d}$ a partir de una semilla $\rho$)

$e \in R_q^d$ (el vector de error, con coeficientes "pequeños")

$b \in R_q^d$ (la pareja resultante)

$b = A \cdot s + e \quad \in R_q^d$

Entonces, el problema M-LWE (como extensión modular de R-LWE) se define como el problema de distinguir muestras $(A, b)$ de parejas uniformes en $R_q^{d \times d} \times R_q^d$.

Como se observa, esta definición resulta relativamente sencilla de entender a partir del ejemplo anterior. Sin embargo, las razones que avalan su seguridad son diferentes: al presentar una estructura menos uniforme, la M-LWE ofrece mejores garantías frente a ataques que explotan la regularidad de los anillos.

Por tanto, la M-LWE es un punto intermedio entre ambos esquemas que permite:

[0] En Kyber se usa la variante M-LWE, donde $s \in R_q^d$, $A \in R_q^{d \times d}$ y $b = A \cdot s + e \in R_q^d$; el pseudocódigo completo en M-LWE aparece en la sección 3.5.1.5.

- Escalabilidad en seguridad mediante el parámetro $d$.
- Mejor eficiencia que en LWE, al poder emplear la NTT sobre anillos.
- Reducción del tamaño de claves, pues la matriz $A$ puede reconstruirse a partir de una semilla.
- Mayor robustez, dado que su estructura "menos simétrica" que un anillo puro dificulta algunos ataques específicos a R-LWE.

#### 3.5.1.3. Fundamentos de seguridad de Kyber

**Seguridad esperada**

**Análisis respecto a ataques conocidos**

#### 3.5.1.4. Parámetros Kyber empleados

A continuación, se presenta una tabla con los parámetros del esquema Kyber1024 empleado en este trabajo fin de grado.

| | $n$ | $k$ | $q$ | $\eta_1$ | $\eta_2$ | $d_u$ | $d_v$ | $\delta$ | $pk(bytes)$ | $sk(bytes)$ | $ct(bytes)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Kyber1024 | 256 | 4 | 3329 | 2 | 2 | 11 | 5 | $2^{-174}$ | 3168 (32) | 1568 | 1568 |

Tabla 3.2: Tabla con los parámetros utilizados por Kyber1024. El valor de $n = 256$ se elige para permitir una escalabilidad sencilla y permitir distintos niveles de seguridad sin perder capacidad de tener un nivel de ruido aceptable. El valor de $k = 4$ fija el tamaño de la retícula e implica una seguridad de 256 bits. El valor de $q = 3329$ es un primo que satisface $n|(q-1)$ para permitir la NTT, los primos anterior y siguiente que también satisfacen esta propiedad conllevan probabilidades de fallo demasiado altas. Los valores $\eta_1$ y $\eta_2$ definen el ruido y junto a $d_u$ y $d_v$ se usan para equilibrar la seguridad y la tasa de fallos $\delta$. El valor (32) en la llave pública es el tamaño de la semilla necesaria para reconstruirla.

#### 3.5.1.5. Algoritmos principales [6]

El algoritmo de generación de llaves genera las llaves pública ($pk$) y privada ($sk$) a partir de los parámetros de la tabla 3.3.

- La función Parse($x$) se encarga de convertir cadenas de bits a su representación NTT garantizando que los coefientes $a_i$ sean del tamaño adecuado $\log_2(q) \approx 11{,}7$ y no permitiendo desbordamientos $a_i < q$.
- La función $\text{CBD}_\eta(x)$ muestrea el ruido a partir mediante una distribución binomial. Convierte un vector de bits $B \in \mathcal{B}^{64\eta}$ a un polinomio $f \in R_q$.
- La función $\text{Encode}_k(x)$ convierte de un vector de bits $B \in \mathcal{B}^{32l}$ a un polinomio $f \in R_q$.

**Algoritmo 4** Generación llaves en Kyber

**Salida:** $pk \in \mathcal{B}^{12\cdot k\cdot n/8+32}$, $sk \in \mathcal{B}^{24\cdot k\cdot n/8+96}$

1: Obtener $d, z \in \mathcal{B}^{32}$ de manera aleatoria usando una distribución uniforme.
2: Obtener dos nuevos parámetros $\rho, \sigma$ expandiendo el valor inicial $d$:

$$(\rho, \sigma) := \text{SHA3-512}(d) \tag{3.34}$$

Se crean las matrices para realizar las operaciones de los algoritmos de la R-LWE.
3: **for** i=0:k-1 **do**
4: **for** j=0:k-1 **do**
5: $\hat{A}[i][j] := \text{Parse}[\text{SHAKE-128}(\rho, j, i)]$ ▷ Muestreo matriz en el dominio NTT.
6: **end for**
7: **end for**
8: $N := 0$
9: **for** i=0:k-1 **do**
10: $s[i] := \text{CBD}_{\eta_1}[\text{SHAKE-256}(\sigma, N)]$ ▷ Muestreo secreto.
11: $N := N + 1$
12: **end for**
13: **for** i=0:k-1 **do**
14: $e[i] := \text{CBD}_{\eta_1}[\text{SHAKE-256}(\sigma, N)]$ ▷ Muestreo error.
15: $N := N + 1$
16: **end for**
17: Se convierten las magnitudes mediante la NTT para agilizar los cálculos:

$$\begin{aligned} \hat{s} &:= \text{NTT}(s) \\ \hat{e} &:= \text{NTT}(e) \\ \hat{t} &:= \hat{A} \circ \hat{s} + \hat{e} \end{aligned} \tag{3.35}$$

18: Se calcula la llave pública:

$$pk := \text{Encode}_{12}(\hat{t} \bmod^{+} \text{q})||\rho \tag{3.36}$$

▷ Se envía $\hat{b}$ junto con la semilla $\rho$ para el calculo de $\hat{A}$.
▷ Así se reduce el tamaño de la clave enviada.
19: Se calcula la llave secreta:

$$sk := \text{Encode}_{12}(\hat{s} \bmod^{+} \text{q})||pk||\text{SHA3-256}(pk)||z \tag{3.37}$$

▷ Se realiza esta concatenación para cumplir la seguridad IND-CCA2 mediante la TFO.
20: **return** $(pk, sk)$

En el algoritmo de cifrado Kyber se obtiene el texto cifrado $c$ a partir de la llave pública $pk$, un mensaje $m$ y una semilla aleatoria $\gamma$ mediante el uso de la NTT.

- Las funciones $\text{Compress}_q(x, y)$ y $\text{Decompress}_q(x, y)$ se usan para reducir el tamaño de los textos cifrados basándose en el fundamento descrito para los mecanismos basados en la R-LWE.
- La función $\text{Decode}_k(x)$ convierte de un polinomio $f \in R_q$ a un vector de bits $B \in \mathcal{B}^{32l}$.

**Algoritmo 5** Cifrado Kyber

**Entrada:** $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$, $m \in \mathcal{B}^{32}$, $\gamma \in \mathcal{B}^{32}$

**Salida:** $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

Calcular los parámetros necesarios:
$N := 0$
**for** i=0:k-1 **do**
    $r[i] := \text{CBD}_{\eta_1}[\text{SHAKE-256}(\gamma, N)]$ ▷ Muestreo elemento $r$.
    $N := N + 1$
**end for**
**for** i=0:k-1 **do**
    $e_1[i] := \text{CBD}_{\eta_2}[\text{SHAKE-256}(\gamma, N)]$ ▷ Muestreo del primer error.
    $N := N + 1$
**end for**
$e_2[i] := \text{CBD}_{\eta_2}[\text{SHAKE-256}(\gamma, N)]$ ▷ Muestreo del segundo error.
Calcular los valores de los textos cifrados:

$$\begin{aligned} \hat{r} &:= \text{NTT}(r) \\ u &:= \text{NTT}^{-1}(\hat{A}^T \circ \hat{r}) + e_1 \\ v &:= \text{NTT}^{-1}(\hat{t}^T \circ \hat{r}) + e_2 + \text{Decompress}_q[\text{Decode}_1(m), 1] \\ c_1 &:= \text{Encode}_{d_u}[\text{Compress}_q(u, d_u)] \\ c_2 &:= \text{Encode}_{d_v}[\text{Compress}_q(v, d_v)] \end{aligned} \tag{3.38}$$

**return** $(c := c1||c2)$

En el algoritmo de encapsulado Kyber a partir de la llave pública $pk$ se obtiene el texto cifrado $c$ y el secreto compartido $k$.

**Algoritmo 6** Encapsulado Kyber

**Entrada:** $pk \in \mathcal{B}^{12\cdot k\cdot n/8+32}$

**Salida:** $c \in \mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$, $k \in \mathcal{B}^*$

1: Obtener los valores necesarios a partir de la llave pública:

$$\begin{aligned} \hat{t} &:= \text{Decode}_{12}(pk) \\ p &:= pk + 12\cdot k\cdot n/8 \end{aligned} \tag{3.39}$$

2: Calcular la matriz $\hat{A}^T$ a partir de $\rho$ codificado en la llave pública.
3: Obtener $m'$ de manera aleatoria usando una distribución uniforme.
4: Obtener los parámetros $m, \kappa, \gamma$ a partir de $m'$ y la llave pública:

$$\begin{aligned} m &:= \text{SHA3-256}(m') \\ (\kappa,\gamma) &:= \text{SHA3-512}[m||\text{SHA3-256}(pk)] \end{aligned} \tag{3.40}$$

5: Obtener el texto cifrado:

$$c \leftarrow \text{Cifrado Kyber}(pk, m, \gamma) \tag{3.41}$$

6: Calcular el secreto compartido:

$$k := \text{SHAKE-256}[\kappa||\text{SHA3-256}(c)] \tag{3.42}$$

7: **return** $(c, k)$

En el algoritmo de decapsulado Kyber a partir del texto cifrado $c$ y la llave secreta $sk$ se puede obtener el secreto compartido $k$.

**Algoritmo 7** Decapsulado Kyber

**Entrada:** $c \in \mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$, $sk \in \mathcal{B}^{24\cdot k\cdot n/8+96}$

**Salida:** $k \in \mathcal{B}^*$

1: Obtener los valores descomprimidos $u$, $v$ y el valor de la llave secreta $s$ en el dominio NTT:

$$\begin{aligned} u &:= \text{Decompress}_q[\text{Decode}_{d_u}(c, d_u)] \\ v &:= \text{Decompress}_q[\text{Decode}_{d_v}(c + d_u\cdot k\cdot n/8, d_v)] \\ \hat{s} &:= \text{Decode}_{12}(sk) \end{aligned} \tag{3.43}$$

2: Obtener el mensaje $m'$ cifrado anteriormente:

$$m' := \text{Encode}_1[\text{Compress}_q\left(v - \text{NTT}^{-1}(\hat{s}^T \circ \text{NTT}(u)), 1\right)] \tag{3.44}$$

Para Garantizar la seguridad ante ataques de canal lateral se vuelve a calcular el texto cifrado.
3: Cifrar $m'$ con la llave pública $pk$ y el parámetro $\gamma$ para obtener el texto cifrado $c'$:

$$\begin{aligned} pk &:= sk + 12\cdot k\cdot n/8 \\ h &:= sk + 24\cdot k\cdot n/8 + 32 \\ (\kappa,\gamma) &:= \text{SHA3-512}[m'||h] \\ c' &\leftarrow \text{Cifrado Kyber}(pk, m', \gamma) \end{aligned} \tag{3.45}$$

4: Comparar los textos cifrados obtenidos, añadiendo un nuevo parámetro $z$ para textos cifrados no válidos.

$$z := sk + 24\cdot k\cdot n/8 + 64 \tag{3.46}$$

5: **if** $c == c'$ **then**
6: **return** $K := \text{SHAKE-256}[\kappa||\text{SHA3-256}(c)]$ ▷ Mismo secreto compartido.
7: **else**
8: **return** $K := \text{SHAKE-256}[z||\text{SHA3-256}(c)]$ ▷ No distinguible de llaves válidas.
9: **end if**

### 3.5.2. Saber

Para elaborar esta sección se parte del artículo de Saber adjuntado con la submission al NIST [7]. En Saber al igual que en Kyber se trabaja en un anillo de la forma:

$$R_q = \frac{\mathbb{Z}_q}{X^n + 1} \tag{3.47}$$

Con $n = 256$, pero con la diferencia de que $q = 2^{13}$, no es posible utilizar la NTT, ya que esta requiere un módulo primo adecuado. No obstante, el uso de un módulo en base 2 ofrece ciertas ventajas [35] frente a esquemas basados en M-LWE:

- Uso de LWR: a diferencia de los esquemas basados en LWE, donde es necesario muestrear errores desde una distribución aleatoria, en LWR el error se introduce mediante redondeo determinista.
- Uso de potencias de 2: al trabajar con módulos del tipo $2^k$, las reducciones modulares se pueden implementar de forma eficiente mediante operaciones de desplazamiento de bits (bitshifts), eliminando la necesidad de reducciones modulares explícitas.

#### 3.5.2.1. Algoritmos de Toom-Cook y Karatsuba

Dado que en Saber no se cumple la condición $n|(q-1)$ para poder utilizar la NTT, se recurre a los algoritmos de Karatsuba [36] y Toom-Cook-4 [37] para acelerar las multiplicaciones polinómicas. Para ello se sigue la estructura del diagrama de la figura 3.5.2.1.

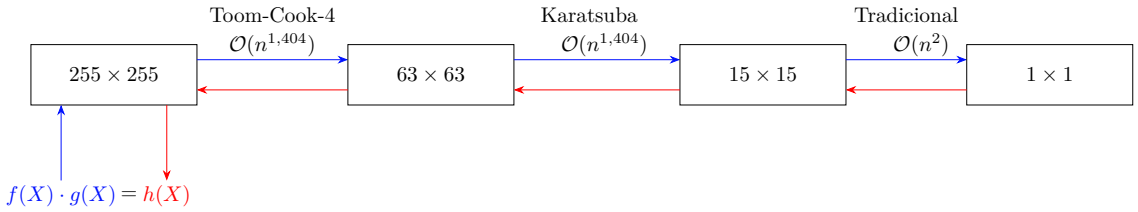


Figura 3.2: Representación del cálculo de un producto de polinomios en Saber. Inicialmente, para polinomios de grado 255, se aplica el algoritmo Toom-Cook-4 que reduce los productos a polinomios de grado 63, ya que ofrece el mejor rendimiento asintótico. Sin embargo, conforme disminuye el tamaño de los polinomios, este rendimiento asintótico se vuelve menos eficiente, por lo que se utiliza el algoritmo de Karatsuba para reducir a polinomios de grado 15, y finalmente se realiza el producto mediante multiplicación tradicional de polinomios.

El algoritmo de Toom-Cook-4 usado en Saber [39] para hacer más eficiente la multiplicación de dos polinomios $c = f \cdot g$ con una complejidad de $\mathcal{O}(n^{\log_4(7)}) \approx \mathcal{O}(n^{1,404})$, sigue los siguientes pasos:

- **Partición**: en este paso se representan los operandos $f$, $g$ mediante 4 bloques iguales teniendo en cuenta que $n-1$ es el grado del polinomio:

$$\begin{aligned} &h(T) = h(X,T) = h_0(X) + h_1(X) \cdot T + h_2(X) \cdot T^2 + h_3(X) \cdot T^3 \\ &T = X^{n/4} \end{aligned} \tag{3.48}$$

De esta manera, con la introducción de una nueva variable $T$ se consigue reducir el tamaño efectivo de los polinomios para el paso de la evaluación. La variable $X$ se utiliza como un parámetro. Aplicado a los polinomios $f$ y $g$ con $n = 256$ en Saber:

$$\begin{aligned} f(T) &= f_0(X) + f_1(X) \cdot T + f_2(X) \cdot T^2 + f_3(X) \cdot T^3 \\ g(T) &= g_0(X) + g_1(X) \cdot T + g_2(X) \cdot T^2 + g_3(X) \cdot T^3 \end{aligned} \tag{3.49}$$

Donde cada $f_i$ y $g_i$ son polinomios de grado 63.

- **Evaluación**: se eligen 7 puntos $v_i$ en los que se evalúan ambos polinomios en $T$. En [39] se desarrolla un algoritmo eficiente para luego recuperar adecuadamente los coeficientes en la interpolación y para ello, se deben utilizar los siguientes puntos:

$$v_i = \left\{ \infty, 2, 1, -1, \frac{1}{2}, -\frac{1}{2}, 0 \right\} \tag{3.50}$$

De esta manera el producto se convierte en un polinomio de grado 6 en $T$ de la forma:

$$c(T) = c_0 + c_1 \cdot T + c_2 \cdot T^2 + c_3 \cdot T^3 + c_4 \cdot T^4 + c_5 \cdot T^5 + c_6 \cdot T^6 \tag{3.51}$$

Con cada $c_i$ siendo un polinomio de grado 126 de la forma:

$$c_i(X) = c_{i,0} + c_{i,1} \cdot X + c_{i,2} \cdot X^2 + ... + c_{i,126} \cdot X^{126} \tag{3.52}$$

Analizando este polinomio en todos los puntos se obtiene el producto $c = A_n \cdot \omega$:

$$\begin{pmatrix} c_6(X) \\ c_5(X) \\ c_4(X) \\ c_3(X) \\ c_2(X) \\ c_1(X) \\ c_0(X) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1/64 & 1/32 & 1/16 & 1/8 & 1/4 & 1/2 & 1 \\ 1/64 & -1/32 & 1/16 & -1/8 & 1/4 & -1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \omega_6(X) \\ \omega_5(X) \\ \omega_4(X) \\ \omega_3(X) \\ \omega_2(X) \\ \omega_1(X) \\ \omega_0(X) \end{pmatrix} \tag{3.53}$$

Para implementar las divisiones por potencias de 2 en la práctica se utilizan bitshifts. Mientras que para implementar las multiplicaciones de los polinomios de grado 63 se utiliza el algoritmo de Karatsuba.

- **Interpolación**: se resulve el problema de interpolación $w(v_i) = \omega_i$ invirtiendo la matriz de Vandermonde $A_n$ generada mediante $v_i$ y computando el vector de coeficientes $w = A_n^{-1} c$.

En este problema de interpolación la matriz se tiene precomputada y una vez obtenidas las inversas se reconstruye el polinomio teniendo en cuenta que este producto se repite para cada uno de los 127 coeficientes del polinomio de salida.

Por último, se deshace la división en bloques sustituyendo $T = X^{64}$, que equivale a desplazar los coeficientes, y se aplica la reducción modular correspondiente a $X^{256} + 1$.

El algoritmo de Karatsuba aplica el paradigma de "divide y vencerás" para multiplicar los polinomios dividiéndolos en dos partes más pequeñas. No obstante, en la implementación de Saber no se usa Karatsuba "tradicional" pues se divide en cuatro bloques [7]. Para ello, utilizando los polinomios de grado 63 anteriormente divididos en Karatsuba se vuelve a introducir una variable $T = X^{16}$ para particionar:

$$\begin{aligned} f &= f_0 + f_1 \cdot T + f_2 \cdot T^2 + f_3 \cdot T^3 \\ g &= g_0 + g_1 \cdot T + g_2 \cdot T^2 + g_3 \cdot T^3 \end{aligned} \tag{3.54}$$

Entonces su producto queda como:

$$f \cdot g = \begin{cases} T^6 \cdot (f_3 \cdot g_3) + \\ T^5 \cdot (f_3 \cdot g_2 + f_2 \cdot g_3) + \\ T^4 \cdot (f_3 \cdot g_1 + f_2 \cdot g_2 + f_1 \cdot g_3) + \\ T^3 \cdot (f_3 \cdot g_0 + f_2 \cdot g_1 + f_1 \cdot g_2 + f_0 \cdot g_3) + \\ T^2 \cdot (f_2 \cdot g_0 + f_1 \cdot g_1 + f_0 \cdot g_2) + \\ T^1 \cdot (f_1 \cdot g_0 + f_0 \cdot g_1) + \\ T^0 \cdot (f_0 \cdot g_0) \end{cases} \tag{3.55}$$

Descomponiendo adecuadamente en 9 productos de polinomios más pequeños $z_i$ de grado 15 que se realizan mediante el método tradicional:

$$\begin{aligned} z_0 &= f_0 \cdot g_0 \\ z_1 &= f_1 \cdot g_1 \\ z_2 &= f_2 \cdot g_2 \\ z_3 &= f_3 \cdot g_3 \\ z_4 &= (f_0 + f_1)\,(g_0 + g_1) \\ z_5 &= (f_0 + f_2)\,(g_0 + g_2) \\ z_6 &= (f_1 + f_3)\,(g_1 + g_3) \\ z_7 &= (f_2 + f_3)\,(g_2 + g_3) \\ z_8 &= (f_0 + f_1 + f_2 + f_3)\,(g_0 + g_1 + g_2 + g_3) \end{aligned} \tag{3.56}$$

Con estos términos se puede ver como la ecuación 3.56 se puede escribir como una combinación lineal de los términos anteriores:

$$f \cdot g = C_0 + C_1 T + C_2 T^2 + C_3 T^3 + C_4 T^4 + C_5 T^5 + C_6 T^6 \tag{3.57}$$

Donde $C_i$ es:

$$\begin{aligned} C_0 &= z_0 \\ C_1 &= z_4 - z_1 - z_0 \\ C_2 &= z_5 - z_2 - z_0 + z_1 \\ C_3 &= z_8 - (z_7 + z_6 + z_5 + z_4) + z_3 + z_2 + z_1 + z_0 \\ C_4 &= z_6 - z_3 - z_1 + z_2 \\ C_5 &= z_7 - z_3 - z_2 \\ C_6 &= z_3 \end{aligned} \tag{3.58}$$

Es relevante señalar que el algoritmo Karatsuba empleado en Saber tiene la misma complejidad asintótica que el tradicional $\mathcal{O}(n^{\log_2(3)}) \approx \mathcal{O}(n^{1,585})$, demostrable de forma directa mediante el teorema maestro [38].

Una vez hechas estas descomposiciones recursivas se van deshaciendo las particiones hasta llegar al producto deseado. De esta manera, se consigue reducir considerablemente el tiempo de cálculo.

#### 3.5.2.2. Aprendizaje Con Redondeo Modular o Modular Learning With Rounding (Mod-LWR)

uasd [35]

#### 3.5.2.3. Fundamentos de seguridad de Saber

**Seguridad esperada**

**Análisis respecto a ataques conocidos**

#### 3.5.2.4. Parámetros Saber empleados

A continuación, se presenta una tabla con los parámetros del esquema FireSaber empleado en este trabajo fin de grado.

| | $n$ | $l$ | $q$ | $p$ | $T$ | $\mu$ | $\delta$ | $pk(bytes)$ | $sk(bytes)$ | $ct(bytes)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| FireSaber | 256 | 4 | $2^{13}$ | $2^{10}$ | $2^6$ | 6 | $2^{-165}$ | 1312 | 3040 (1760) | 1472 |

Tabla 3.3: Tabla con los parámetros utilizados por FireSaber. TODO EXPLICAR CADA PARAMETRO

#### 3.5.2.5. Algoritmos principales [7]

### 3.5.3. Hamming Quasi-Cyclic (HQC)

Para [40]

# Capítulo 4

# Desarrollo

## 4.1. Implementación comunicación serie

### 4.1.1. Parámetros generales y formato mensajes

### 4.1.2. Implementación en el ordenador

### 4.1.3. Implementación en el microprocesador

## 4.2. Interfaz para los algoritmos de cifrado asimétrico

### 4.2.1. Interfaz Kyber

### 4.2.2. Interfaz Saber

### 4.2.3. Interfaz HQC

## 4.3. Implementación algoritmos en el PC

### 4.3.1. Compilación en librerías

### 4.3.2. Diagrama de uso

### 4.3.3. Diagrama funcional

### 4.3.4. Diagrama de clases

## 4.4. Implementación de algoritmos en el microcontrolador

### 4.4.1. Diagrama de uso

### 4.4.2. Diagrama funcional

## 4.5. Implementación del intercambio de claves. Creación del secreto compartido

Hablar de los modelos de comunicaciones a implementar .... (msg teams)

### 4.5.1. Modelo 1

### 4.5.2. Modelo 2

### 4.5.3. Modelo 3

## 4.6. Tests de rendimiento realizados

# Capítulo 5

# Resultados y discusión

En este capítulo se muestran los resultados obtenidos de aplicar las rutinas desarrolladas con anterioridad.

## 5.1. Resultados

### 5.1.1. (No sé si lo metere) Resultado de ejecución de los algoritmos clásicos

#### 5.1.1.1. RSA

#### 5.1.1.2. ECC

### 5.1.2. Resultados de ejecución de los algoritmos post-cuánticos

#### 5.1.2.1. Kyber

#### 5.1.2.2. Saber

#### 5.1.2.3. HQC

### 5.1.3. Resultados de los distintos modelos de comunicación

#### 5.1.3.1. Modelo 1

#### 5.1.3.2. Modelo 2

#### 5.1.3.3. Modelo 3

## 5.2. Discusión

### 5.2.1. Comparativa entre los distintos algoritmos post-cuánticos analizados

### 5.2.2. Comparativa entre los distintos modelos de comunicación

# Capítulo 6

# Conclusiones

Se presentan a continuación las conclusiones del proyecto y desarrollos futuros para mejorar la implementación.

## 6.1. Conclusión

Una vez finalizado el proyecto...

## 6.2. Desarrollos futuros

Un posible desarrollo...

# Apéndice A

# Definiciones básicas

# Apéndice B

# Ejemplo R-LWE

Sea el espacio de trabajo en $R_{17} = \mathbb{Z}_{17}[X]/\left(X^2+1\right)$ con un mensaje $z \in \{0,1\}^2$ y la distribución del error $e \in \{-1,0,1\}$.

En el primer paso se generan $a, s$ y $e$:

$$\begin{array}{l} a[X] = 3 + 4X \\ s[X] = 1 + 0X \\ e[X] = -1 + 1X \end{array} \tag{B.1}$$

Una vez inicializados los parámetros, se procede al cálculo de $b$. La reducción módulo $X^2+1$ equivale a sustituir $X^2$ por $-1$ cada vez que aparezca.

$$b[X] = a[X] \cdot s[X] + e[X] = 2 + 5X \tag{B.2}$$

Con la clave pública calculada $a||b$ se puede cifrar un mensaje $z$, pero antes se generan los valores de $z, r, e_1$ y $e_2$:

$$\begin{array}{l} z[X] = 1 + 0X \\ r[X] = 1 + 1X \\ e_1[X] = 0 + 1X \\ e_2[X] = -1 + 0X \end{array} \tag{B.3}$$

Con estos valores se calculan los textos cifrados:

$$\begin{array}{l} u[X] = a[X] \cdot r[X] + e_1[X] = 16 + 8X \\ v[X] = b[X] \cdot r[X] + e_2[X] + \left\lfloor \dfrac{q}{2} \right\rceil \cdot z[X] = 5 + 7X \end{array} \tag{B.4}$$

Por último, se comprueba que el mensaje se descifra correctamente.

$$z'[X] = v[X] - u[X] \cdot s[X] = 6 + 16X \rightarrow \begin{cases} z'_0: & \begin{array}{l} \mathrm{d}_0(0) = 6 \\ \mathrm{d}_0(9) = 3 \end{array} \\ z'_1: & \begin{array}{l} \mathrm{d}_1(0) = 1 \\ \mathrm{d}_1(9) = 8 \end{array} \end{cases} \tag{B.5}$$

Con estas distancia se obtiene que $z = (1,0)$. No obstante, aunque este descifrado se comprueba que se cumple que el error no supera la magnitud límite $q/4 = 4{,}25$.

$$\varepsilon[X] = r[X] \cdot e[X] - s[X] \cdot e_1[X] + e_2[X] = 14 + 16X \tag{B.6}$$

Para cumplirse la distancia a 0 debe ser menor a $q/4$ para cada coeficiente:

$$\begin{array}{l} \mathrm{d}_0(0) = 3 \\ \mathrm{d}_1(0) = 1 \end{array} \tag{B.7}$$

# Bibliografía


[1] Leslie Lamport et al. The latex project, 2024.

[2] Tikzmaker. `https://tikzmaker.com/editor`, 2024.

[3] ISO/IEC 9899:2018 Information technology ? Programming languages ? C. `https://www.iso.org/standard/82075.html`, 2024. International Organization for Standardization, Geneva, Switzerland.

[4] ISO/IEC 14882:2025 Information technology ? Programming languages ? C++. `https://www.iso.org/standard/83626.html`, 2024. International Organization for Standardization, Geneva, Switzerland.

[5] Infineon Technologies AG. CY8CPROTO-063-BLE PSoC 6 BLE Prototyping Kit. `https://www.infineon.com/cms/en/product/evaluation-boards/cy8cproto-063-ble/`, 2020. Consultado: 2025-05-05.

[6] 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 3.01). Technical report, CRYSTALS Project, January 2021. NIST PQC Round 3 submission.

[7] Andrea Basso, José María Bermudo Mera, Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Michiel Van Beirendonck, and Frederik Vercauteren. SABER: Mod-LWR based KEM (Round 3 Submission). Technical report, Katholieke Universiteit Leuven and University of Birmingham, 2020. NIST PQC Round 3 submission.

[8] 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. NIST Interagency Report NIST IR 8413 (Updated), National Institute of Standards and Technology (NIST), July 2022.

[9] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. Status report on the fourth round of the nist post-quantum cryptography standardization process. NIST Internal Report NIST IR 8545, National Institute of Standards and Technology (NIST), Gaithersburg, MD, March 2025.

[10] Yichen Zhang, Yiming Bian, Zhengyu Li, Song Yu, and Hong Guo. Continuous-variable quantum key distribution system: Past, present, and future. *Applied Physics Reviews*, 11(1):011318, 03 2024.

[11] Mandeep Kumar and Bhaskar Mondal. A brief review on quantum key distribution protocols. *Multimedia Tools and Applications*, January 2025.

[12] Akoh Atadoga, Oluwatoyin Ajoke Farayola, Benjamin Samson Ayinla, Olukunle Oladipupo Amoo, Temitayo Oluwaseun Abrahams, and Femi Osasona. A comparative review of data encryption methods in the usa and europe. 5:447–460, Feb. 2024.

[13] Qixin Zhang. An overview and analysis of hybrid encryption: The combination of symmetric encryption and asymmetric encryption. In *2021 2nd International Conference on Computing and Data Science (CDS)*, pages 616–622, 2021.

[14] Basel Halak, Yildiran Yilmaz, and Daniel Shiu. Comparative analysis of energy costs of asymmetric vs symmetric encryption-based security applications. *IEEE Access*, 10:76707–76719, 2022.

[15] Mila Anastasova, Panos Kampanakis, and Jake Massimo. PQ-HPKE: Post-quantum hybrid public key encryption. Cryptology ePrint Archive, Paper 2022/414, 2022.

[16] C. H. Ugwuishiwu, U. E. Orji, C. I. Ugwu, and C. N. Asogwa. An overview of quantum cryptography and shor?s algorithm. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(5):7487–7495, September 2020.

[17] David Aasen, Morteza Aghaee, Zulfi Alam, and Mariusz Andrzejczuk et al. Roadmap to fault tolerant quantum computation using topological qubit arrays, 2025.

[18] Fouzia Samiullah, Ming-Lee Gan, Sedat Akleylek, and Y. Aun. Quantum resistance saber-based group key exchange protocol for iot. *IEEE Open Journal of the Communications Society*, 6:378–398, 2025.

[19] National Institute of Standards and Technology. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication FIPS 203, National Institute of Standards and Technology, Gaithersburg, MD, USA, August 2024.

[20] National Institute of Standards, Technology (NIST), and Morris J. Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions, 2015-08-04 00:08:00 2015.

[21] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[22] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2004.

[23] Siyon Singh and Eric Sakk. Implementation and analysis of shor?s algorithm to break rsa cryptosystem security. January 2024.

[24] Gorjan Alagic, Elaine Barker, Lily Chen, Dustin Moody, Angela Robinson, Hamilton Silberg, and Noah Waller. Recommendations for key-encapsulation mechanisms: Initial public draft. NIST Special Publication NIST SP 800-227 ipd, National Institute of Standards and Technology (NIST), January 2025.

[25] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Berlin, Heidelberg, 1999. Springer.

[26] National Institute of Standards and Technology. Post-quantum cryptography - round 3 submissions. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`, 2020. Consultado: 2025-05-05.

[27] National Institute of Standards and Technology. Post-quantum cryptography - round 4 submissions. `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`, 2022. Consultado: 2025-05-05.

[28] Ardianto Satriawan, Rella Mareta, and Hanho Lee. A complete beginner guide to the number theoretic transform (NTT). Cryptology ePrint Archive, Paper 2024/585, 2024.

[29] Miguel A. Moreno. Primitive $n$-th roots of unity of finite fields. `https://www.csd.uwo.ca/~mmorenom/CS874/Lectures/Newton2Hensel.html/node9.html`, n.d. Consultado: 2025-05-05.

[30] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[31] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6), November 2013.

[32] Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[33] Yang Wang and Mingqiang Wang. Module-LWE versus ring-LWE, revisited. Cryptology ePrint Archive, Paper 2019/930, 2019.

[34] Léo Ducas and John Schanck. Security estimation scripts for kyber and dilithium. `https://github.com/pq-crystals/security-estimates`, 2023. Consultado: 2025-05-31.

[35] Jan-Pieter D?Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. Cryptology ePrint Archive, Paper 2018/230, 2018.

[36] Anatolii Karatsuba and Yu Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595, 12 1962.

[37] A. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Doklady Akademii Nauk SSSR*, 3, 01 1963.

[38] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. A general method for solving divide-and-conquer recurrences. *SIGACT News*, 12(3):36?44, September 1980.

[39] Marco Bodrato and Alberto Zanoni. Integer and polynomial multiplication: towards optimal toom-cook matrices. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, page 17?24, New York, NY, USA, 2007. Association for Computing Machinery.

[40] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, and Gilles Zémor. HQC: Hamming Quasi-Cyclic (Fourth Round Submission). Technical report, HQC Team, October 2022. NIST PQC Round 4 submission.