FIPS 203 (borrador)

Publicación de estándares federales de procesamiento de información

Basado en módulo de celosía Encapsulación de claves Estándar del mecanismo

Categoría: Seguridad Informática **Subcategoría: Criptografía**

Laboratorio de Tecnología de la Información
Instituto Nacional de Estándares y Tecnología
Gaithersburg, MD 20899-8900

Esta publicación está disponible de forma gratuita en:
https://doi.org/10.6028/NIST.FIPS.203.ipd

Publicado el 24 de agosto de 2023

Departamento de Comercio de EE.
UU. *Gina M. Raimondo, Secretaria*

Instituto Nacional de Estándares y Tecnología
Laurie E. Locascio, directora del NIST y subsecretaria de Comercio para Estándares y Tecnología

Prefacio

La Serie de Publicaciones Federales de Estándares de Procesamiento de Información (FIPS) del Instituto Nacional de Estándares y Tecnología es la serie oficial de publicaciones relacionadas con estándares y pautas desarrolladas bajo 15 USC 278g-3 y emitida por el Secretario de Comercio bajo 40 USC 11331.

Los comentarios relacionados con esta publicación del Estándar Federal de Procesamiento de Información son bienvenidos y deben enviarse utilizando la información de contacto en la cláusula "Consultas y comentarios" de la sección de anuncios.

James A. St Pierre, director interino del
Laboratorio de Tecnología de la Información

Abstracto

Un mecanismo de encapsulación de claves (o KEM) es un conjunto de algoritmos que, bajo ciertas condiciones, pueden ser utilizados por dos partes para establecer una clave secreta compartida a través de un canal público. Una clave secreta compartida que se establece de forma segura mediante un KEM se puede utilizar con algoritmos criptográficos de clave simétrica para realizar tareas básicas en comunicaciones seguras, como cifrado y autenticación.

Este estándar especifica un mecanismo de encapsulación de claves llamado ML-KEM. La seguridad de ML-KEM está relacionada con la dificultad computacional del llamado problema de Aprendizaje de Módulos con Errores. Actualmente, se cree que ML-KEM es seguro incluso contra adversarios que posean una computadora cuántica.

Este estándar especifica tres conjuntos de parámetros para ML-KEM. En orden de mayor seguridad (y menor rendimiento), estos conjuntos de parámetros son ML-KEM-512, ML-KEM-768 y ML-KEM-1024.

Palabras clave:la seguridad informática; criptografía; cifrado; Estándares Federales de Procesamiento de Información; criptografía basada en celosía; encapsulación de claves; poscuántico; criptografía de clave pública

Publicación 203 de normas federales de procesamiento de información

Publicado: 24 de agosto de 2023

anunciando el

Encapsulación de claves basada en módulos
Estándar del mecanismo

Las publicaciones de estándares federales de procesamiento de información (FIPS) son emitidas por el Instituto Nacional de Estándares y Tecnología (NIST) según 15 USC 278g-3 y por el Secretario de Comercio según 40 USC 11331.

1.Nombre de la Norma.Estándar de mecanismo de encapsulación de claves basado en celosía de módulo (ML-KEM) (FIPS PUB 203).

2.Categoría de Norma.La seguridad informática.Subcategoría.Criptografía.

3.Explicación.Este estándar especifica un conjunto de algoritmos para aplicaciones que requieren una clave criptográfica secreta compartida por dos partes que solo pueden comunicarse a través de un canal público. Una clave criptográfica (o simplemente "clave") se representa en una computadora como una cadena de bits. A*clave secreta compartida*se calcula conjuntamente por dos partes (por ejemplo, la Parte A y la Parte B) utilizando un conjunto de reglas y parámetros. Bajo ciertas condiciones, estas reglas y parámetros garantizan que ambas partes producirán la misma clave y que esta clave compartida sea secreta para los adversarios. Una clave secreta compartida de este tipo se puede utilizar con algoritmos criptográficos de clave simétrica (especificados en otros estándares del NIST) para realizar tareas, como el cifrado y la autenticación de información digital.

Si bien existen muchos métodos para establecer una clave secreta compartida, el método particular descrito en esta especificación es un mecanismo de encapsulación de claves (KEM). En un KEM, el cálculo de la clave secreta compartida comienza cuando la Parte A genera una*clave de decapsulación*y un*clave de encapsulación*. sesenta y cinco La parte A mantiene privada la clave de decapsulación y pone la clave de encapsulación a disposición de la parte B. La parte B luego usa la clave de encapsulación de la parte A para generar una copia de una clave secreta compartida junto con una clave asociada.*texto cifrado*. Luego, la parte B envía el texto cifrado a la parte A a través del mismo canal. Finalmente, la Parte A utiliza el texto cifrado de la Parte B junto con la clave de decapsulación privada de la Parte A para calcular otra copia de la clave secreta compartida.

La seguridad del KEM particular especificado aquí está relacionada con la dificultad computacional de resolver ciertos sistemas de ecuaciones lineales ruidosas, específicamente las llamadas*Módulo de aprendizaje con errores*(MLWE) problema. Actualmente, se cree que este método particular de establecer una clave secreta compartida es seguro incluso contra adversarios que posean una computadora cuántica. En el futuro, es posible que se especifiquen y aprueben KEM adicionales en publicaciones FIPS o en publicaciones especiales del NIST.

4.Autoridad que lo aprueba.Secretario de Comercio.

5.Agencia de mantenimiento.Departamento de Comercio, Instituto Nacional de Estándares y Tecnología, Laboratorio de Tecnología de la Información (ITL).

6.Aplicabilidad.Los Estándares Federales de Procesamiento de Información se aplican a los sistemas de información utilizados u operados por agencias federales o por un contratista de una agencia u otra organización en nombre de una agencia. No se aplican a los sistemas de seguridad nacional según se definen en 44 USC 3552.

Este estándar debe implementarse siempre que se requiera el establecimiento de una clave secreta compartida para aplicaciones federales, incluido el uso de dicha clave con algoritmos criptográficos de clave simétrica, de acuerdo con la Oficina de Administración y Presupuesto aplicable y las políticas de la agencia. Las agencias federales también pueden utilizar métodos alternativos que el NIST haya indicado que son apropiados para este propósito.

La adopción y el uso de esta norma están disponibles para organizaciones privadas y comerciales.

7.Implementaciones.Se puede implementar un mecanismo de encapsulación de claves en software, firmware, hardware o cualquier combinación de los mismos. Una implementación conforme puede reemplazar la secuencia dada de pasos en los algoritmos de nivel superior de ML-KEM (es decir,ML-KEM.Generación de claves, ML-KEM.encapsula, yML-KEM.decápsulas) con cualquier proceso equivalente. En otras palabras, se permiten diferentes procedimientos que produzcan la salida correcta para cada entrada. En particular, no se requiere que las implementaciones conformes utilicen las mismas subrutinas (de los algoritmos principales antes mencionados) que se utilizan en esta especificación.

NIST desarrollará un programa de validación para probar la conformidad de las implementaciones con los algoritmos de este estándar. La información sobre los programas de validación está disponible enhttps://csrc.nist.gov/projects/cmvp. Los valores de ejemplo para algoritmos criptográficos están disponibles en https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values.

8.Otras funciones de seguridad aprobadas.Implementaciones que cumplen con este estándardeberá emplean algoritmos criptográficos que han sidoaprobadopara proteger información confidencial del gobierno federal.AprobadoLos algoritmos y técnicas criptográficos incluyen aquellos que son:

(a) Especificado en una publicación de Estándares Federales de Procesamiento de Información (FIPS),

(b) Adoptado en una recomendación FIPS o NIST, o

(c) Especificadas en la lista de funciones de seguridad aprobadas para FIPS 140-3.

9. Control de exportación.Ciertos dispositivos criptográficos y los datos técnicos relacionados con ellos están sujetos a controles federales de exportación. Las exportaciones de módulos criptográficos que implementan este estándar y los datos técnicos relacionados con ellos deben cumplir con todas las leyes y regulaciones federales y estar autorizadas por la Oficina de Industria y Seguridad del Departamento de Comercio de EE. UU. La información sobre las regulaciones de exportación está disponible enhttps://www.bis.doc.gov.

10.Patentes.El NIST ha celebrado dos acuerdos de licencia de patentes para facilitar la adopción de la selección anunciada por el NIST del algoritmo PQC de cifrado de clave pública CRYSTALS-K.YBER. El NIST y las partes otorgantes de licencias comparten el deseo, en aras del interés público, de que las patentes con licencia estén disponibles gratuitamente para que las practique cualquier implementador del algoritmo ML-KEM publicado por el NIST. ML-KEM es el nombre que recibe el algoritmo en este estándar derivado de CRYSTALS-KYBER. Para obtener un resumen y extractos de la licencia, consultehttps://csrc.nist.gov/csrc/m edia/Projects/post-quantum-cryptography/documents/selected-algos-2022/nist-pqc-licensesummary-and-excerpts.pdf. La implementación del algoritmo especificado en el estándar puede estar cubierta por patentes estadounidenses y extranjeras de las que el NIST no tiene conocimiento.

11.Calendario de implementación.Esta norma entra en vigor inmediatamente después de su publicación final.

12.Especificaciones.Estándares federales de procesamiento de información (FIPS) 203, Estándar del mecanismo de encapsulación de claves basado en módulos (adjunto).

13.Cualificaciones.En las aplicaciones, las garantías de seguridad de un KEM solo se mantienen bajo ciertas condiciones (consulte NIST SP 800-227 [1]). Una de esas condiciones es el secreto de varios valores, incluida la aleatoriedad utilizada por las dos partes, la clave de decapsulación y la propia clave secreta compartida. Usuariosdeberá,por lo tanto, tenga cuidado con la divulgación de estos valores.

Si bien la intención de este estándar es especificar los requisitos generales para implementar algoritmos ML-KEM, el cumplimiento de este estándar no garantiza que una implementación particular sea segura. Es responsabilidad del implementador garantizar que cualquier módulo que implemente una capacidad de establecimiento clave esté diseñado y construido de manera segura.

De manera similar, el uso de un producto que contenga una implementación que cumpla con este estándar no garantiza la seguridad del sistema general en el que se utiliza el producto. La autoridad responsable en cada agencia o departamento.deberágarantizar que una implementación general proporcione un nivel aceptable de seguridad.

El NIST seguirá siguiendo los avances en el análisis del algoritmo ML-KEM. Al igual que con sus otros estándares de algoritmos criptográficos, el NIST reevaluará formalmente este estándar cada cinco años.

Tanto este estándar como las posibles amenazas que reducen la seguridad proporcionada mediante el uso de este estándar serán revisados por el NIST según corresponda, teniendo en cuenta los análisis y la tecnología recientemente disponibles. Además, el conocimiento de cualquier avance tecnológico o cualquier debilidad matemática del algoritmo hará que el NIST reevalúe este estándar y proporcione las revisiones necesarias.

14.Procedimiento de renuncia.La Ley Federal de Gestión de Seguridad de la Información (FISMA) no permite exenciones a los Estándares Federales de Procesamiento de Información (FIPS) que el Secretario de Comercio hace obligatorias.

15.Dónde obtener copias de la norma.Esta publicación está disponible accediendo https://csrc.nist.gov/publications. Otras publicaciones sobre seguridad informática están disponibles en el mismo sitio web.

dieciséis.Cómo citar esta publicación.NIST ha asignadoNIST FIPS 203 ipdcomo identificador de publicación para este FIPS, según elSintaxis del identificador de publicación de la serie técnica del NIST. NIST recomienda que se cite de la siguiente manera:

> Instituto Nacional de Estándares y Tecnología (2023) Estándar del mecanismo de encapsulación de claves basado en módulos. (Departamento de Comercio, Washington, DC), Publicación de estándares federales de procesamiento de información (FIPS) NIST FIPS 203 ipd.https://doi.org/10.6028/NIST.FIPS.203.ipd

17.Consultas y Comentarios.Las consultas y comentarios sobre este FIPS pueden enviarse a fps-203-comments@nist.gov.

Convocatoria de reclamaciones de patentes

Esta revisión pública incluye una convocatoria de información sobre reivindicaciones de patentes esenciales (reivindicaciones cuyo uso sería necesario para cumplir con la orientación o los requisitos de este borrador de publicación del Laboratorio de Tecnología de la Información (ITL)). Dichas orientaciones y/o requisitos pueden indicarse directamente en esta Publicación ITL o mediante referencia a otra publicación. Esta convocatoria también incluye la divulgación, cuando se conozca, de la existencia de solicitudes de patentes estadounidenses o extranjeras pendientes relacionadas con este borrador de publicación ITL y de cualquier patente estadounidense o extranjera relevante y vigente.

ITL podrá exigir al titular de la patente, o a una parte autorizada para dar garantías en su nombre, en forma escrita o electrónica, ya sea:

a) garantía en forma de descargo de responsabilidad general en el sentido de que dicha parte no posee ni tiene la intención de poseer ninguna reivindicación de patente esencial; o

b) garantía de que una licencia para dicha(s) reivindicación(es) de patente esencial se pondrá a disposición de los solicitantes que deseen utilizar la licencia con el fin de cumplir con la orientación o los requisitos de este borrador de publicación del DIT:

(i) bajo términos y condiciones razonables que estén demostrablemente libres de cualquier discriminación injusta; o

(ii) sin compensación y bajo términos y condiciones razonables que estén demostrablemente libres de cualquier discriminación injusta.

Dicha garantía deberá indicar que el titular de la patente (o un tercero autorizado para dar garantías en su nombre) incluirá en cualquier documento de transferencia de propiedad de patentes sujetas a la garantía, disposiciones suficientes para garantizar que los compromisos de la garantía sean vinculantes para el cesionario. y que el cesionario incluirá igualmente disposiciones apropiadas en caso de futuras transferencias con el objetivo de vincular a cada sucesor en interés.

La garantía también indicará que está destinada a ser vinculante para los sucesores en interés independientemente de si dichas disposiciones están incluidas en los documentos de transferencia pertinentes.

Dichas declaraciones deberán dirigirse a:fps-203-comments@nist.gov.

Publicación 203 de normas federales de procesamiento de información

Especificación para el

Encapsulación de claves basada en módulos

Estándar del mecanismo

Tabla de contenido

Lista de tablas

Lista de Figuras

Lista de algoritmos

1. Introducción

1.1 Propósito y alcance

Esta norma especifica la*Mecanismo de encapsulación de claves basado en celosía de módulo*o ML-KEM. Un mecanismo de encapsulación de claves (o KEM) es un conjunto de algoritmos que se pueden utilizar para establecer una clave secreta compartida entre dos partes que se comunican a través de un canal público. Un KEM es un tipo particular de esquema de establecimiento de claves. NIST actual-aprobadoLos esquemas de establecimiento clave se especifican en NIST SP-800-56A.*Recomendación para esquemas de establecimiento de claves por pares que utilizan criptografía basada en logaritmos discretos*[2] y NIST SP-800-56B,*Recomendación para esquemas de establecimiento de claves por pares que utilizan criptografía de factorización de enteros*[3].

Es bien sabido que los esquemas de establecimiento de claves especificados en NIST SP-800-56A y NIST SP-800-56B son vulnerables a ataques que utilizan computadoras cuánticas suficientemente capaces. ML-KEM es unaprobadoalternativa que actualmente se cree que es segura incluso contra adversarios en posesión de una computadora cuántica. ML-KEM se deriva de la versión de tercera ronda del CRYSTALS-KYBERKEM [4], una presentación en el proyecto de estandarización de criptografía poscuántica del NIST. Para las diferencias entre ML-KEM y CRYSTALS-KYBER, mira la sección1.3.

Este estándar especifica los algoritmos y conjuntos de parámetros del esquema ML-KEM. Su objetivo es proporcionar información suficiente para implementar ML-KEM de una manera que pueda pasar la validación (verhttps://csrc.nist.gov/projects/cryptographic-module-validation-program). Para conocer las definiciones y propiedades generales de los KEM, incluidos los requisitos para el uso seguro de los KEM en aplicaciones, consulte NIST SP 800-227 [1].

Este estándar especifica tres conjuntos de parámetros para ML-KEM. Estos conjuntos de parámetros ofrecen diferentes compensaciones entre la solidez de la seguridad y el rendimiento. Los tres conjuntos de parámetros de ML-KEM sonaprobadopara proteger los sistemas de comunicación sensibles y no clasificados del gobierno federal de EE. UU.

1.2 Contexto

En los últimos años, ha habido un progreso constante hacia la construcción de computadoras cuánticas. Si se construyen computadoras cuánticas a gran escala, la seguridad de muchos criptosistemas de clave pública de uso común estará en riesgo. Esto incluiría esquemas de establecimiento de claves y esquemas de firma digital que se basan en la factorización de números enteros y logaritmos discretos (tanto en campos finitos como en curvas elípticas). Como resultado, en 2016, el Instituto Nacional de Estándares y Tecnología (NIST) inició un proceso público para seleccionar algoritmos criptográficos de clave pública resistentes a los cuánticos para su estandarización. Se envió un total de 82 algoritmos candidatos al NIST para su consideración para su estandarización.

Después de tres rondas de evaluación y análisis, el NIST seleccionó los primeros cuatro algoritmos para estandarizar como resultado del proceso de estandarización de la criptografía poscuántica (PQC). Estos algoritmos están destinados a proteger la información confidencial del gobierno de EE. UU. en el futuro previsible, incluso después de la llegada de las computadoras cuánticas. Este estándar especifica una variante del algoritmo seleccionado CRYSTALS-KYBER, un mecanismo de encapsulación de claves basado en celosía (KEM) [4]. En este estándar, el KEM especificado aquí se denominará ML-KEM,

ya que se basa en el llamado supuesto de Aprendizaje de Módulos con Errores.

1.3 Diferencias con CRYSTALS-KYBEREnvío

A continuación se muestra una lista de todas las diferencias de esquema entre CRYSTALS-KYBER(como se describe en [4]) y el esquema ML-KEM especificado en este documento. La lista consta únicamente de aquellas diferencias que resultan en un comportamiento diferente de entrada-salida de los algoritmos principales (es decir,KeyGen, encaps, decaps)de CRISTALES-KYBERy ML-KEM. Recuerde que una implementación conforme sólo necesita coincidir con el comportamiento de entrada-salida de estos tres algoritmos (ver"Implementaciones"arriba, y la Sección3.3 abajo). En consecuencia, la siguiente lista no incluye ninguna de las numerosas diferencias en cómo los algoritmos principales realmente producen salidas a partir de entradas (por ejemplo, a través de diferentes pasos computacionales o diferentes subrutinas). La siguiente lista tampoco incluye ninguna diferencia en la presentación entre esta norma y [4].

- En la especificación de tercera ronda [4], la clave secreta compartida se trató como un valor de longitud variable cuya longitud depende de cómo se usaría esta clave en la aplicación correspondiente. En esta especificación, la longitud de la clave secreta compartida se fija en 256 bits. En esta especificación, esta clave se puede utilizar directamente en aplicaciones como clave simétrica; alternativamente, se pueden derivar claves simétricas a partir de esta clave, como se especifica en la Sección3.3.

- ElML-KEM.encapsulayML-KEM.decápsulasLos algoritmos de esta especificación utilizan una variante diferente de la transformada de Fujisaki-Okamoto (ver [5,6]) que la especificación de tercera ronda [4]. Específicamente,ML-KEM.encapsulaya no incluye un hash del texto cifrado en la derivación del secreto compartido, yML-KEM.decápsulasse ha ajustado para que coincida con este cambio.

- En la especificación de tercera ronda [4], la aleatoriedad inicial*metro*en elML-KEM.encapsula El algoritmo fue codificado por primera vez antes de ser utilizado. Específicamente, entre líneas.1y2en algoritmodieciséis, hubo un paso adicional que realizó la operación$metro \leftarrow h(metro)$. El propósito de este paso era proteger contra el uso de procesos de generación de aleatoriedad defectuosos. Como este estándar requiere el uso de generación de aleatoriedad aprobada por NIST, este paso es innecesario y no se realiza en ML-KEM.

- Esta especificación incluye pasos explícitos de validación de entradas que no formaban parte de la especificación de la tercera ronda [4]. Por ejemplo,ML-KEM.encapsularequiere que la matriz de bytes que contiene la clave de encapsulación se decodifique correctamente en una matriz de números enteros módulo$q$sin reducciones modulares.

2. Glosario de términos, acrónimos y símbolos matemáticos

2.1 Términos y definiciones

aprobado	Aprobado por FIPS y/o recomendado por NIST. Un algoritmo o técnica que 1) se especifica en una recomendación FIPS o NIST, 2) se adopta en una recomendación FIPS o NIST, o 3) se especifica en una lista de NIST-aprobadofunciones de seguridad.
decapsulación	El proceso de aplicación deldecápsulasalgoritmo de un KEM. Este algoritmo acepta un texto cifrado KEM y la clave de desencapsulación como entrada y produce una clave secreta compartida como salida.
clave de decapsulación	Una clave criptográfica producida por un KEM durante la generación de claves y utilizada durante el proceso de decapsulación. La clave de decapsulación debe mantenerse privada y debe destruirse cuando ya no sea necesaria.
clave de descifrado	Una clave criptográfica que se utiliza con un PKE para descifrar textos cifrados en textos sin formato. La clave de descifrado debe mantenerse privada y debe destruirse cuando ya no sea necesaria.
destruir	Una acción aplicada a una clave u otro dato secreto. Una vez destruido un dato secreto, no se puede recuperar ninguna información sobre su valor.
encapsulación	El proceso de aplicación delencapsulaalgoritmo de un KEM. Este algoritmo acepta la aleatoriedad privada y la clave de encapsulación como entrada y produce una clave secreta compartida y un texto cifrado asociado como salida.
clave de encapsulación	Una clave criptográfica producida por un KEM durante la generación de claves y utilizada durante el proceso de encapsulación. La clave de encapsulación se puede hacer pública.
Clave de encriptación	Una clave criptográfica que se utiliza con un PKE para cifrar textos sin formato en textos cifrados. La clave de cifrado se puede hacer pública.
proceso equivalente	Dos procesos son equivalentes si se produce el mismo resultado cuando se ingresan los mismos valores a cada proceso (ya sea como parámetros de entrada, como valores disponibles durante el proceso, o ambos).
función hash	Una función en cadenas de bits en la que la longitud de la salida es fija. AprobadoLas funciones hash relevantes para este estándar se especifican en FIPS 202 [7].
Texto cifrado KEM	Una cadena de bits que se produce mediante encapsulación y se utiliza como entrada para la decapsulación.
llave	Una cadena de bits que se utiliza junto con un algoritmo criptográfico. Los ejemplos aplicables a este estándar incluyen: las claves de encapsulación y desencapsulación (de un KEM), la clave secreta compartida (producida por un KEM) y las claves de cifrado y descifrado (de una PKE).

encapsulación de claves mecanismo (KEM)	Un conjunto de tres algoritmos criptográficos (KeyGen, encaps,yDecápsulas) que puede ser utilizado por dos partes para establecer una clave secreta compartida a través de un canal público.
Par de claves	Un conjunto de dos claves con la propiedad de que una clave puede hacerse pública mientras que la otra debe mantenerse privada. En este estándar, esto podría referirse al par de claves (clave de encapsulación, clave de desencapsulación) de un KEM o al par de claves (clave de cifrado, clave de descifrado) de un PKE.
fiesta	Un individuo (persona), organización, dispositivo o proceso. En esta especificación, hay dos partes (Parte A y Parte B, o Alice y Bob), y realizan conjuntamente el proceso de establecimiento de claves mediante un KEM.
pseudoaleatorio	Se dice que un proceso (o datos producidos por un proceso) es pseudoaleatorio cuando el resultado es determinista pero también parece aleatorio siempre que la acción interna del proceso esté oculta a la observación. Para fines criptográficos, "efectivamente aleatorio" significa "computacionalmente indistinguible de aleatorio dentro de los límites de la seguridad prevista".
canal publico	Un canal de comunicación entre dos partes; dicho canal puede ser observado y posiblemente también dañado por terceros.
Llave pública esquema de cifrado (PKE)	Un conjunto de tres algoritmos criptográficos (generación de claves, cifrar,yDescifrar) que puede ser utilizado por dos partes para enviar datos secretos a través de un canal público. También conocido como esquema de cifrado asimétrico.
clave secreta compartida	El resultado final de un proceso de establecimiento de claves KEM. Es una clave criptográfica que se puede utilizar para criptografía de clave simétrica. Debe mantenerse en privado y debe destruirse cuando ya no sea necesario.
categoría de seguridad	Un número asociado con la seguridad de un algoritmo criptográfico poscuántico según lo especificado por NIST (consulte el Apéndice A, Tabla4).
fuerza de seguridad	Un número asociado con la cantidad de trabajo que se requiere para descifrar un algoritmo o sistema criptográfico.
deberá	Se utiliza para indicar un requisito de esta norma.
debería	Se utiliza para indicar una recomendación fuerte pero no un requisito de esta norma. Ignorar la recomendación podría conducir a resultados indeseables.

2.2 Acrónimos

AES	Estándar de cifrado avanzado Distribución
CDB	binomial centrada Estándar federal de
FIPS	procesamiento de información

KEM	Aprendizaje del mecanismo de
LWE	encapsulación de claves con errores
MLWE	Módulo de aprendizaje con errores
NIST	Instituto Nacional de Estándares y Tecnología NIST
nistir	Informe interno o interinstitucional
NTT	Cifrado de clave pública con
PKE	transformación teórica de números
PQC	Función pseudoaleatoria de
PRF	criptografía poscuántica
RBG	Generador de bits aleatorios
sha	Algoritmo hash seguro
AGITAR	Algoritmo hash seguro KECCAK
SP	Publicación especial
XOF	Función de salida extensible

2.3 Símbolos matemáticos

S^*	Si S es un conjunto, esto denota el conjunto de tuplas (o matrices) de longitud finita de elementos del conjunto S, incluida la tupla vacía (o matriz vacía).
S^k	Si S es un conjunto, esto denota el conjunto de k-tuplas (o longitud-k matrices) de elementos del conjunto S.
$\text{BitRev}_7(r)$	Inversión de bits de un entero de siete bits r. Específicamente, si $r = r_0 + 2r_1 + 4r_2 + \cdots + 64r_6$ con $r_i \in \{0,1\}$, entonces $\text{BitRev}_7(r) = r_6 + 2r_5 + 4r_4 + \cdots + 64r_0$.
$\hat{F}$	el elemento de t_q que es igual a la representación NTT de un polinomio $F \in R_q$ (mira la sección 4.3).
$\mathbb{Q}$	El conjunto de los números racionales.
$\mathbb{Z}_{metro}$	El módulo del anillo de números enteros. *metro*, es decir, el conjunto $\{0,1,\ldots,metro-1\}$ equipado con las operaciones de módulo de suma y multiplicación *metro*.
$\mathbb{Z}$	El conjunto de los números enteros.
v^t, A^t	La transposición de una fila o columna. v; Además, la transpuesta de una
$\hat{F}_j$	matriz. A. El coeficiente de X^j de un polinomio $F = F_0 + F_1 X + \cdots + F_{255} X$ El entero $r \in \mathbb{Z}_q$.
$r \bmod^{\pm} metro$	único r' en $\{0,1,\ldots,metro-1\}$ tal que *metro* divide $r - r'$.

$r \bmod^{\pm} metro$	Para *metro* par (respectivamente, impar), esto denota el número entero único r' tal que $-m/2 < r' \leq metro/2$ (respectivamente, $-(metro-1)/2 \leq r' \leq (metro-1)/2$) y *metro* divide $r-r'$.
$\lvert B \rvert$	Si *B* es un número, esto denota el valor absoluto de *B*. Si *B* es una matriz, esto denota su longitud.
$\lceil X \rceil$	el techo de *X*, es decir, el entero más pequeño mayor o igual a *X*.
$\lfloor X \rceil$	El redondeo de *X* al número entero más cercano; si $X = y + 1/2$ para algunos $y \in \mathbb{Z}$, entonces $\lfloor X \rceil = y + 1$.
$\lfloor X \rfloor$	el piso de *X*, es decir, el mayor entero menor o igual a *X*. El
B	conjunto $\{0, 1, \ldots, 255\}$ de enteros de 8 bits sin signo (bytes). La
$A \mid\mid B$	concatenación de dos matrices o cadenas de bits. *A* y *B*.
$B[i]$	La entrada en el índice *i* en la matriz *B*. Todas las matrices tienen índices que comienzan en
$B[k:metro]$	cero. El subarreglo $(B[k], B[k+1], \ldots, B[metro-1])$ de la matriz *B*. Denota el número entero 256
$norte$	en todo este documento.
q	Denota el número entero primo $3329 = 2^8 \cdot 13 + 1$ a lo largo de este documento.
R_q	El anillo $\mathbb{Z}_q[X]/(X^{norte}+1)$ que consta de polinomios de la forma $F = F_0 + F_1 X + \cdots + F_{255} X^{255}$ dónde $F_j \in \mathbb{Z}_q$ para todos *j*, equipado con módulo de suma y multiplicación $X^{norte}+1$.
$s \leftarrow X$	En pseudocódigo, esta notación significa que la variable *s* se le asigna el valor de la expresión *X*.
$s \leftarrow^{\text{ps}} B^{\ell}$	En pseudocódigo, esta notación significa que la variable *s* se le asigna el valor de una matriz de ℓ bytes aleatorios. Los bytes deben generarse utilizando la aleatoriedad a partir de una aprobado RBG (ver Sección 3.3).
t_q	La imagen de R_q bajo la transformada de teoría de números. Sus elementos se denominan "representaciones NTT" de polinomios en R_q (mira la sección 4.3).

2.4 Interpretación del pseudocódigo

Esta sección describe las convenciones del pseudocódigo utilizado para describir los algoritmos de este estándar. Se entiende que todos los algoritmos tienen acceso a dos constantes enteras globales: $norte = 256$ y $q = 3329$. También hay cinco variables enteras globales: $k, \eta_1, \eta_2, d_{tu}$ y d_v. Todas las demás variables son locales. Las cinco variables globales se establecen en valores particulares cuando se selecciona un conjunto de parámetros (consulte la Sección 7).

Cuando los algoritmos de esta especificación invocan otros algoritmos como subrutinas, todos los argumentos (entradas) se pasan por valor. En otras palabras, se crea una copia de las entradas y con la copia se invoca la subrutina. No existe el "pasar por referencia".

Tipos de datos. Para variables que representan la entrada o salida de un algoritmo, el tipo de datos (p. ej.,

bit, byte, matriz de bits) se describirán explícitamente al comienzo del algoritmo. Para la mayoría de las variables locales del pseudocódigo, el tipo de datos se deduce fácilmente del contexto. Para todas las demás variables, el tipo de datos se declarará en un comentario. En un algoritmo único, el tipo de datos de una variable se determina la primera vez que se utiliza la variable y no se cambiará. Los nombres de variables pueden reutilizarse y se reutilizarán en diferentes algoritmos, incluso con diferentes tipos de datos.

Además de los tipos de datos atómicos estándar (p. ej., bits, bytes) y estructuras de datos (p. ej., matrices), el módulo de números enteros $metro$ (es decir, elementos de $\mathbb{Z}_{metro}$) también se utilizará como tipo de datos abstractos. Está implícito que el módulo de reducción $metro$ tiene lugar cada vez que se realiza una asignación a una variable en $\mathbb{Z}_{metro}$. Por ejemplo, para $z \in \mathbb{Z}_{metro}$ y cualquier número entero x, y, la declaración

$$z \leftarrow x + y \tag{2.1}$$

significa que z se le asigna el valor $x + y$ modificación $metro$. El pseudocódigo es agnóstico con respecto a cómo un módulo entero $metro$ se representa en implementaciones reales o cómo se calcula la reducción modular.

Sintaxis de bucle. El pseudocódigo utilizará los bucles " while " y " for ". La sintaxis del "mientras" se explica por sí misma. En el caso de bucles “for”, la sintaxis será del estilo del lenguaje de programación C. En Algoritmo se dan dos ejemplos sencillos. 1.

Algoritmo 1 Por ejemplo

Realiza dos bucles “for” simples.

```
1: para (i ← 0; yo < 10; i++)
2:     A[i] ← i                ▷ A es una matriz de números enteros de longitud 10
3: fin para                    ▷ A ahora tiene el valor (0,1,2,3,4,5,6,7,8,9)
4: j ← 0
5: para (k ← 256; k > 1; k ← k/2)
6:     B[j] ← k                ▷ B es una matriz de números enteros de longitud 8
7:     j ← j + 1
8: fin para                    ▷ B ahora tiene el valor (256,128,64,32,dieciséis,8,4,2)
```

Aritmética con matrices de números enteros. Este estándar hace un uso extensivo de matrices de números enteros. módulo $metro$ (es decir, elementos de $\mathbb{Z}_{\ell\ metro}$). En un caso típico, los valores relevantes son $metro = q$ y $\ell = norte = 256$. Aritmética con matrices en $\mathbb{Z}_{\ell\ metro}$ se realizará de la siguiente manera. Dejar $a \in \mathbb{Z}_{metro}$ y $X, Y \in \mathbb{Z}_{\ell\ metro}$. Las declaraciones

$$z \leftarrow a \cdot X$$
$$W. \leftarrow X + Y$$

resultará en dos matrices $Z, W \in \mathbb{Z}_{\ell\ metro}$, con la propiedad que $z[i] = a \cdot X[i]$ y $W.[i] = X[i] + Y[i]$ para todos i. Multiplicación de matrices en $\mathbb{Z}_{\ell\ metro}$ sólo será significativo cuando $metro = q$ y $\ell = norte = 256$, en cuyo caso corresponde a la multiplicación en un anillo particular. Esta operación se describirá en (2.2) abajo.

Representaciones de objetos algebraicos. Una operación esencial en ML-KEM es la numeración.

transformada teórica (NTT), que mapea un polinomio F en cierto anillo R_q a su "representante NTT" "sentación" $\hat{F}$ en un anillo diferente t_q. Los anillos R_q y t_q y el NTT se analizan en detalle en la Sección 4.3. Esta norma representará elementos de R_q y elementos de t_q en pseudocódigo usando matrices de módulo de números enteros q, como sigue.

Un elemento F de R_q es un polinomio de la forma

$$F = F_0 + F_1 X + \cdots + F_{255} X^{255} \in R_q$$

y será representado en pseudocódigo por la matriz

$$(F_0, F_1, \ldots, F_{255}) \in \mathbb{Z}_q^{256}$$

cuyas entradas contienen los coeficientes de F. Abusando un poco de la notación, esta matriz también se denotará por F. El i-ésima entrada de la matriz F contendrá así el i-ésimo coeficiente del polinomio F (es decir, $F[i] = F_i$).

Un elemento (a veces llamado "representación NTT") *gramo* de t_q es una tupla de 128 polinomios, cada uno de grado como máximo uno. Específicamente,

$$gramo = (gramo_{0,0} + gramo_{0,1} X,\ \hat{g}_{1,0} + gramo_{1,1} X,\ \ldots,\ \hat{g}_{127,0} + gramo_{127,1} X) \in t_q.$$

Un objeto algebraico de este tipo estará representado en pseudocódigo por la matriz

$$(gramo_{0,0},\ \hat{g}_{0,1},\ \hat{g}_{1,0},\ \hat{g}_{1,1},\ \ldots,\ \hat{g}_{127,0},\ \hat{g}_{127,1}) \in \mathbb{Z}_q^{256}.$$

Abusando un poco de la notación, esta matriz también se denotará por *gramo*. En este caso, el mapeo entre las entradas de la matriz y los coeficientes es $gramo[2i] = gramo_{i,0}$ y $gramo[2i+1] = gramo_{i,1}$ para $i \in \{0,1,\ldots,127\}$.

Convertir entre un polinomio $F \in R_q$ y su representación NTT $\hat{F} \in t_q$ se realizará a través del algoritmos NTT (Algoritmo 8) y NTT^{-1} (Algoritmo 9). Estos algoritmos actúan sobre matrices de coeficientes, como se describió anteriormente, y satisfacen $\hat{F} = \text{NTT}(F)$ y $F = \text{NTT}^{-1}(\hat{F})$.

Aritmética con polinomios y representaciones NTT. Las operaciones algebraicas de suma y multiplicación escalar en R_q y t_q se realizan en forma de coordenadas. Por ejemplo, si $a \in \mathbb{Z}_q$ y $F \in R_q$, el i-ésimo coeficiente del polinomio $a \cdot F \in R_q$ es igual a $a \cdot F_i$ modificación q. En pseudocódigo, elementos de ambos R_q y t_q están representados por matrices de coeficientes (es decir, elementos de $\mathbb{Z}_q^{256}$), como se describió anteriormente. Las operaciones algebraicas de suma y multiplicación escalar se realizan así mediante la suma y multiplicación escalar de las matrices de coeficientes correspondientes. Por ejemplo, la adición de dos representaciones NTT en pseudocódigo se realiza mediante una declaración de la forma $\hat{h} \leftarrow \hat{f} + gramo$, dónde $\hat{h}, \hat{f}, \hat{g} \in \mathbb{Z}_q^{256}$ son matrices de coeficientes.

Las operaciones algebraicas de multiplicación en R_q y t_q son tratados de la siguiente manera. Por razones de eficiencia, la multiplicación en R_q no será utilizado. El significado algebraico de la multiplicación en t_q se analiza en la sección 4.3.1. En pseudocódigo, será realizado por el algoritmo. Multiplicar NTT (Algoritmo 10). Específicamente, si $\hat{f}, \hat{g} \in \mathbb{Z}_q^{256}$ son un par de matrices (cada una representa el NTT de

algún polinomio), entonces

$$\hat{h} \leftarrow \hat{f} \times_{t_q} gramo \quad \text{medio} \quad \hat{h} \leftarrow \text{Multiplicar NTT}(\hat{f}, \hat{g}). \tag{2.2}$$

El resultado es una matriz $\hat{h} \in (\mathbb{Z}_{256})_q$.

Matrices y vectores.Además de matrices de módulo de números enteros q, el pseudocódigo también hará uso de matrices cuyas entradas son en sí mismas elementos de $\mathbb{Z}_{256}$. Por ejemplo, un elemento $v \in (\mathbb{Z}_{256})^3_q$ será una matriz de longitud tres cuyas entradas $v[0], v[1]$ y $v[2]$ son en sí mismos elementos de $\mathbb{Z}_{256}$ (es decir, matrices). Se puede pensar que cada una de estas entradas representa un polinomio en R_q, de modo que v en sí mismo representa un elemento del módulo R^3_q.

Cuando se utilizan matrices para representar matrices y vectores cuyas entradas son elementos de R_q, se indicarán con letras en negrita (p. ej., $\mathbf{v}$ para vectores y $\mathbf{A}$ para matrices). Cuando se utilizan matrices para representar matrices y vectores cuyas entradas son elementos de t_q, se denotarán con una "sombrero" (p. ej., $\hat{\mathbf{v}}$ y $\hat{\mathbf{A}}$). A menos que se realice una operación de transposición explícita, se entiende que los vectores son vectores de columna. Entonces se pueden ver los vectores como el caso especial de matrices con una sola columna.

Conversión entre matrices sobre R_q y matrices sobre t_q se realizará de forma coordinada. Específicamente, si $A \in (\mathbb{Z}_{256})^{k \times \ell}_q$, entonces la declaración

$$\mathbf{A} \leftarrow \text{NTT}(\mathbf{A})$$

resultará en $\in (\mathbb{Z}_{256})^{k \times \ell}_q$ tal que $[yo, j] = \text{NTT}(\mathbf{A}[yo, j])$ para todos yo, j. Esto implica correr NTT un total de $k \cdot \ell$ veces. Tenga en cuenta que el caso de los vectores corresponde a $\ell = 1$.

Aritmética con matrices y vectores.A continuación se describe cómo realizar aritmética con matrices sin dejar de ver los vectores como un caso especial de matrices.

La suma y la multiplicación escalar se realizan por coordenadas. Suma de matrices sobre R_q y t_q entonces es sencillo. En el caso de t_q, la multiplicación escalar se realiza mediante (2.2). Para ejemplo, si $F \in \mathbb{Z}_{256}$ y tú, $\hat{\mathbf{v}} \in (\mathbb{Z}_{256})^k_q$, entonces

$$\hat{\mathbf{w}} \leftarrow \hat{f} \cdot \hat{\mathbf{u}}$$
$$\hat{\mathbf{z}} \leftarrow \hat{\mathbf{u}} + \hat{\mathbf{v}}$$

resultará en $\hat{\mathbf{w}}, \hat{\mathbf{z}} \in (\mathbb{Z}_{256})^k_q$ satisfactorio $\hat{\mathbf{w}}[i] = F \times_{t_q} \hat{\mathbf{u}}[i]$ y $\hat{\mathbf{z}}[i] = \hat{\mathbf{u}}[i] + \hat{\mathbf{v}}[i]$ para todos i. Tenga en cuenta que la multiplicación y suma de entradas individuales aquí se realiza utilizando la aritmética apropiada para matrices de coeficientes de elementos de t_q.

También será necesario multiplicar matrices con entradas en t_q. Esto se hace usando la multiplicación de matrices estándar siendo la multiplicación del caso base (es decir, la multiplicación de entradas individuales) la multiplicación en t_q. Si $\hat{\mathbf{A}}$ y $\hat{\mathbf{B}}$ son dos matrices con entradas en t_q, su producto matricial se denotará $\circ$. A continuación se proporcionan algunos ejemplos de declaraciones en pseudocódigo que implican la multiplicación de matrices. En estos ejemplos, $\hat{\mathbf{A}}$ es un $k \times k$ matriz, mientras que $\hat{\mathbf{u}}$ y $\hat{\mathbf{v}}$ son vectores de longitud k. Todo

Tres de estos objetos están representados en pseudocódigo mediante matrices: a$k \times k$matriz para y longitud-k matrices para $\hat{u}$ y $\hat{v}$. Las entradas de $\hat{A}$, $\hat{u}$ y $\hat{v}$ son elementos dez256 $_q$.Los dos primeros pseudocódigos Las siguientes declaraciones producen una nueva longitud.$k$vector cuyas entradas se especifican en el lado derecho. La tercera declaración de pseudocódigo calcula un producto escalar; el resultado está por lo tanto en el anillo base (es decir,t_q), y está representado por un elemento$\hat{z}$dez$_{q}$256.

$$\hat{w} \leftarrow \hat{A} \circ \hat{u} \qquad \hat{w}[i] = \sum_{j=0}^{k-1} \hat{A}[yo, j] \times_{tq} \hat{u}[j]$$

$$\hat{y} \leftarrow \hat{A}^{T} \circ \hat{u} \qquad \hat{y}[i] = \sum_{j=0}^{k-1} \hat{A}[j, i] \times_{tq} \hat{u}[j]$$

$$\hat{z} \leftarrow \hat{u}^{T} \circ \hat{v} \qquad \hat{z} = \sum_{j=0}^{k-1} \hat{u}[j] \times_{tq} \hat{v}[j]$$

la multiplicacion$\times_{tq}$de las entradas individuales anteriores se realiza utilizandoMultiplicar NTT, como se describe en (2.2) arriba.

Aplicar algoritmos a arrays.Las convenciones de la aritmética de coordenadas descritas anteriormente también se extenderán a los algoritmos que actúan sobre (y/o producen) un tipo de datos atómicos. Cuando el pseudocódigo invoca dicho algoritmo en una entrada de matriz, se da a entender que el algoritmo se invoca repetidamente para cada entrada de la matriz. Por ejemplo, la funciónComprimir$_d$:z$_q \to$z$_{2^d}$ definido en la sección4se puede invocar en una entrada de matriz$F \in$z256$_q$ con la declaración

$$k \leftarrow \text{Comprimir}_d(F). \tag{2.3}$$

El resultado será que$k \in$z256y$_{2^d}$[i] =Comprimir$_d$(F[i])para cada$i$. implica Este cálculo ejecutar elComprimiralgoritmo 256 veces.

3. Descripción general del esquema ML-KEM

Esta sección ofrece una descripción general de alto nivel del esquema ML-KEM.

3.1 Mecanismos clave de encapsulación

La siguiente es una descripción breve e informal de los mecanismos de encapsulación de claves (o KEM). Para obtener más detalles, consulte NIST SP 800-227 [1].

Un mecanismo de encapsulación de claves (o KEM) es un conjunto de algoritmos que pueden usarse, bajo ciertas condiciones, para establecer una clave secreta compartida entre dos partes que se comunican. Esta clave secreta compartida se puede utilizar para criptografía de clave simétrica.

Un KEM consta de tres algoritmos y una colección de conjuntos de parámetros. Los tres algoritmos son:

- un algoritmo de generación de claves denotado porGeneración de claves;
- un algoritmo de "encapsulación" indicado porEncapsula;
- un algoritmo de "decapsulación" indicado porDecápsulas.

La colección de conjuntos de parámetros se utiliza para seleccionar un equilibrio entre seguridad y eficiencia. Cada parámetro establecido en la colección es una lista de valores numéricos específicos, uno para cada parámetro requerido por los algoritmos anteriores.

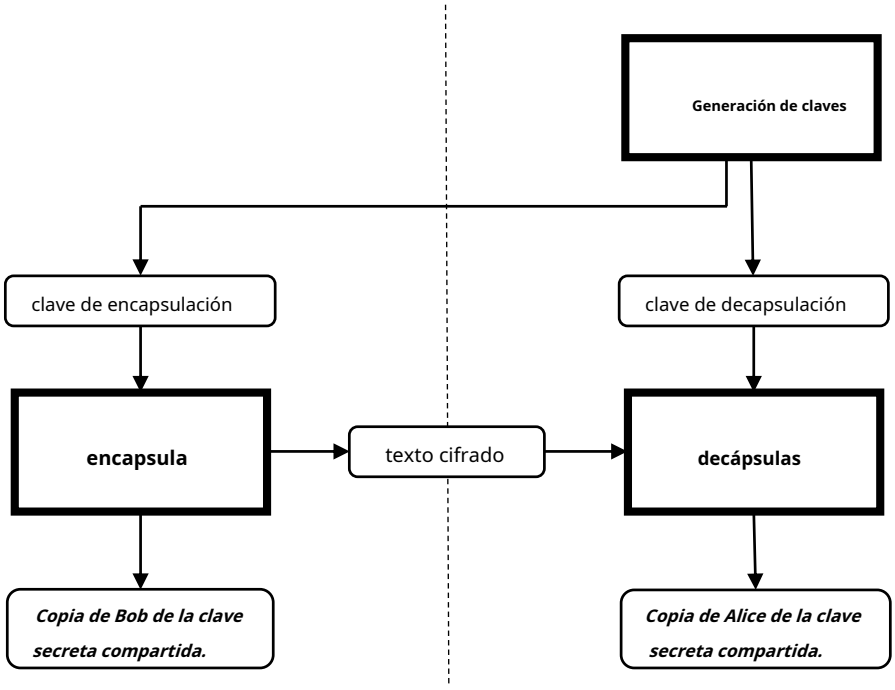

Figura 1. Una vista simple del establecimiento de claves utilizando un KEM

Se puede utilizar un KEM para establecer una clave secreta compartida entre dos partes (consulte la Figura1) referido aquí como Alice y Bob. Alice comienza a correr.Generación de clavespara generar una clave de encapsulación (pública) y una clave de decapsulación (privada). Al obtener la clave de encapsulación de Alice,

Bob dirige elencapsulaalgoritmo; esto produce la copia de Bob$k_B$de la clave secreta compartida junto con un texto cifrado asociado. Bob envía el texto cifrado a Alice, y Alice completa el proceso ejecutando el decápsulasalgoritmo que utiliza su clave de decapsulación y el texto cifrado; este paso produce la copia de Alice$k_A$de la clave secreta compartida.

Después de completar el proceso anterior, a Alice y Bob les gustaría concluir que sus resultados individuales satisfacen$k_A = k_B$y que este valor es una clave secreta segura, aleatoria y compartida. Sin embargo, estas propiedades sólo se mantienen bajo ciertos supuestos importantes, como se analiza en NIST SP 800-227 [1].

3.2 El esquema ML-KEM

ML-KEM es un mecanismo de encapsulación de claves basado en CRYSTALS-KYBER[4], un esquema que fue descrito inicialmente en [8]. La siguiente es una descripción breve e informal de la suposición computacional subyacente a ML-KEM y cómo se construye el esquema ML-KEM.

El supuesto computacional.La seguridad de ML-KEM se basa en la presunta dificultad de resolver el problema denominado Aprendizaje de Módulos con Errores (MLWE) [9], una generalización del problema de aprendizaje con errores (LWE) introducido por Regev en 2005 [10]. La dureza del problema MLWE se basa en la supuesta dureza de ciertos problemas computacionales en redes de módulos [9]. De ahí el nombre del esquema ML-KEM.

En el problema LWE, la entrada es un conjunto de ecuaciones lineales aleatorias "ruidosas" en algún secreto variables$X \in \mathbb{Z}_{norte}^{q}$, y la tarea es recuperar$X$. El ruido en las ecuaciones es tal que el estándar Los algoritmos (por ejemplo, la eliminación gaussiana) son intratables. El problema LWE se presta naturalmente a aplicaciones criptográficas. Por ejemplo, si$X$se interpreta como una clave secreta, entonces se puede cifrar un valor de un bit muestreando una ecuación lineal aproximadamente correcta (si el valor del bit es cero) o una ecuación lineal que dista mucho de ser correcta (si el valor del bit es uno). Es plausible que sólo una parte en posesión de$X$ Entonces podemos distinguir estos dos casos. Luego, el cifrado se puede delegar a otra parte mediante la publicación de una gran colección de ecuaciones lineales ruidosas, que la parte que cifra puede combinar adecuadamente. El resultado es un esquema de cifrado asimétrico.

En un nivel alto, el problema MLWE plantea la misma tarea que LWE pero con$\mathbb{z}_{norte}^{q}$reemplazado con el móduloR_k^qconstruido tomando el$k$-pliegue del producto cartesiano de un determinado anillo polinómico$R_q$para algún número entero$k > 1$. En particular, el secreto es ahora un elementoXdel móduloR_k^q.

La construcción ML-KEM.A alto nivel, la construcción del ML-KEM se desarrolla en dos pasos. Primero, la idea mencionada anteriormente se utiliza para construir un esquema de cifrado de clave pública a partir del problema MLWE. En segundo lugar, este esquema de cifrado de clave pública se convierte en un mecanismo de encapsulación de claves mediante la denominada transformación Fujisaki-Okamoto (FO).11,12]. Además de producir un KEM, la transformación FO también pretende proporcionar seguridad en un modelo de ataque adversario significativamente más general. Como resultado, se cree que ML-KEM satisface la llamada seguridad IND-CCA [1,4,13].

La especificación de los algoritmos ML-KEM en este estándar seguirá el patrón anterior. Específicamente, este estándar describirá primero un esquema de cifrado de clave pública llamado K-PKE y luego utilizará los algoritmos de K-PKE como subrutinas al describir los algoritmos de ML-KEM. La transformación criptográfica de K-PKE a ML-KEM es crucial para lograr una seguridad total.

El esquema K-PKE no es lo suficientemente seguro yno debeutilizarse como un esquema independiente (consulte la Sección3.3).

Una característica notable de ML-KEM es el uso de la*transformada teórica de números*(NTT). El NTT convierte un polinomio$F\in R_q$a una representación alternativa como un vector$\hat{F}$de polinomios lineales. Aunque las representaciones NTT permiten una multiplicación rápida, se deben aplicar otras operaciones, como el redondeo y el muestreo, a las representaciones polinómicas estándar.

ML-KEM satisface las propiedades clave de la corrección de KEM y se conoce una prueba de seguridad teórica asintótica (en un determinado modelo heurístico) [4]. Cada uno de los conjuntos de parámetros de ML-KEM viene con una fortaleza de seguridad asociada, que se estimó en base al criptoanálisis actual (consulte la Sección7para detalles).

Conjuntos de parámetros y algoritmos.Recuerde que un KEM consta de algoritmos.KeyGen, encaps, y decápsulas,junto con una colección de conjuntos de parámetros. En el caso de ML-KEM, los tres algoritmos antes mencionados son:

- ML-KEM.Generación de claves(Algoritmo15);
- ML-KEM.encapsula(Algoritmodieciséis);
- ML-KEM.decápsulas(Algoritmo17).

Estos algoritmos se describen y analizan en detalle en la Sección6. ML-KEM viene equipado con tres conjuntos de parámetros:

- ML-KEM-512 (categoría de seguridad 1);
- ML-KEM-768 (categoría de seguridad 3);
- ML-KEM-1024 (categoría de seguridad 5).

Estos conjuntos de parámetros se describen y analizan en detalle en la Sección7; Las categorías de seguridad 1-5 se definen en el Apéndice.A. Cada conjunto de parámetros asigna un valor numérico particular a cinco variables enteras:k,η_1,η_2,d_{tu}, yd_v. Los valores de estas variables en cada conjunto de parámetros se dan en la Tabla2de Sección7. Además de estos cinco parámetros variables, también existen dos constantes:$norte$=256 yq=3329.

Fallos de decapsulación.Siempre que todas las entradas estén bien formadas, el procedimiento de establecimiento de claves de ML-KEM nunca fallará explícitamente. Específicamente, elML-KEM.encapsulayML-KEM.decápsulas Los algoritmos siempre generarán un valor con el mismo tipo de datos que una clave secreta compartida y nunca generarán un símbolo de error o falla. Sin embargo, es posible (aunque extremadamente improbable) que el proceso falle en el sentido de que Alice (a través deML-KEM.decápsulas) y Bob (víaML-KEM.encapsula) producirán resultados diferentes, aunque ambos se comporten honestamente y no haya ninguna interferencia adversaria. En este caso, Alice y Bob claramente no lograron producir una clave secreta compartida. Este evento se llama falla de decapsulación. La probabilidad de falla de decapsulación se define como la probabilidad de que el proceso

1. (ek,NS)←ML-KEM.Generación de claves()

2. $(c,k) \leftarrow$ML-KEM.encapsula(ek)

3. $k' \leftarrow$ML-KEM.decápsulas(C,NS)

resultados en$k \neq k$(es decir, la clave encapsulada es diferente de la clave decapsulada). Las estimaciones de la probabilidad (o tasa) de falla de decapsulación para cada uno de los conjuntos de parámetros ML-KEM se dan en la Tabla1(ver [4]).

Tabla 1. Tasas de falla de decapsulación para ML-KEM

Conjunto de parámetros	Tasa de fracaso de decapsulación
ML-KEM-512	2^{-139}
ML-KEM-768	2^{-164}
ML-KEM-1024	2^{-174}

Una nota sobre la terminología de las claves.Un KEM implica tres tipos diferentes de claves: claves de encapsulación, claves de decapsulación y claves secretas compartidas. ML-KEM se basa en el esquema de cifrado de clave pública K-PKE, y K-PKE tiene dos tipos de claves adicionales: claves de cifrado y claves de descifrado. En la literatura, las claves de encapsulación y las claves de cifrado a veces se denominan "claves públicas", mientras que las claves de decapsulación y descifrado a veces se pueden denominar "claves privadas". Para reducir la confusión, este estándar no utilizará los términos "clave pública" y "clave privada". En su lugar, nos referiremos a las claves utilizando los términos más específicos anteriores (es decir, clave de encapsulación, clave de decapsulación, clave de cifrado, clave de descifrado o clave secreta compartida).

3.3 Requisitos para implementaciones ML-KEM

Esta sección describe varios requisitos para implementar los algoritmos de ML-KEM. Los requisitos para utilizar ML-KEM en aplicaciones específicas se proporcionan en NIST SP 800-227 [1].

K-PKE es sólo un componente.El esquema de cifrado de clave pública K-PKE descrito en la Sección 5no debeutilizarse como un esquema criptográfico independiente. En cambio, los algoritmos que componen K-PKE solo pueden usarse como subrutinas en los algoritmos de ML-KEM. En particular, los algoritmosK-PKE.Generación de claves(Algoritmo12),K-PKE.cifrar(Algoritmo13), yK-PKE.Descifrar (Algoritmo14) sonno aprovadopara su uso como esquema de cifrado de clave pública.

Implementaciones equivalentes.Cada uno de los tres algoritmos de nivel superior (es decir,ML-KEM.Generación de claves, ML-KEM.encapsula, yML-KEM.decápsulas) define una operación matemática particular, asignando cualquier entrada dada a una salida correspondiente. Por ejemplo, la operación definida por el algoritmo ML-KEM.encapsula toma una matriz de bytes como entrada y produce dos matrices de bytes como salida.

En esta norma, las tres operaciones defnidas porML-KEM.Generación de claves,ML-KEM.encapsula, y ML-KEM .decápsulasse describen utilizando secuencias particulares de pasos computacionales. Una implementación conforme puede reemplazar cada una de estas secuencias con una secuencia diferente de pasos, siempre que la operación resultante sea un proceso equivalente al especificado en este estándar.

Por ejemplo, una implementación conforme de la operación de encapsulación debe tener la propiedad de que, para cualquier conjunto de parámetros y cualquier matriz de bytes de entradabueno,la distribución de las matrices de bytes de salida es idéntica a la distribuciónML-KEM.encapsula(ek)como se especifica en esta norma.

Uso aprobado de la clave secreta compartida.La salida de los algoritmos de encapsulación y decapsulación de ML-KEM es siempre un valor de 256 bits. En condiciones apropiadas (ver arriba; ver también NIST SP 800-227 [1]), esta salida es una clave secreta compartida*k*. Esta clave secreta compartida*k*se puede utilizar directamente como clave para criptografía simétrica. Cuando se necesita la derivación de claves, las claves simétricas finalesdeberáderivarse de esta clave secreta compartida de 256 bits*k*en unaprobadomanera, como se especifica en NIST SP 800-108 [14].

Generación de aleatoriedad.Tres algoritmos de este estándar requieren la generación de aleatoriedad: K-PKE. Generación de claves,ML-KEM.Generación de claves, yML-KEM.encapsula. En pseudocódigo, el paso en el que esta aleatoriedad se genera se indica mediante una declaración en pseudocódigo de la forma$metro\leftarrow_{ps}B32$. un fresco Se debe generar una cadena de bytes aleatorios para cada invocación de este tipo. Estos bytes aleatoriosdeberá ser generado utilizando unaprobadoRBG, según lo prescrito en NIST SP 800-90A, NIST SP 800-90B y NIST SP 800-90C [15,dieciséis,17]. Además, el RBG utilizódeberátener una seguridad de al menos 128 bits para ML-KEM-512, al menos 192 bits para ML-KEM-768 y al menos 256 bits para ML-KEM-1024.

Validación de entrada.Los algoritmosML-KEM.encapsulayML-KEM.decápsulasrequieren validación de entrada. Implementadoresdeberáasegurarse de queML-KEM.encapsulayML-KEM.decápsulassólo se ejecutan en entradas validadas, como se describe en la Sección6. Como se analizó anteriormente, los implementadores pueden optar por implementar los algoritmos de nivel superior (es decir,ML-KEM.encapsula,ML-KEM.decápsulas, o ML-KEM. Generación de claves) utilizando cualquier proceso equivalente; la validación de los insumos se considera parte de este proceso. Una implementación conformedeberáser equivalente a validar primero la entrada y luego ejecutar el algoritmo apropiado.

Destrucción de valores intermedios.Los datos utilizados internamente por los algoritmos KEM en pasos de cálculo intermedios podrían ser utilizados por un adversario para comprometer la seguridad. Implementadoresdeberá, por lo tanto, asegúrese de que dichos datos intermedios se destruyan tan pronto como ya no sean necesarios.

Sin aritmética de punto flotante.Implementaciones de ML-KEMno debeUtilice aritmética de punto flotante. Todos los pasos de división y redondeo de los algoritmos de ML-KEM se pueden realizar dentro del conjunto de números racionales.

4. Algoritmos auxiliares

4.1 Funciones criptográficas

Los algoritmos especificados en esta publicación requieren el uso de varias funciones criptográficas. Cada funcióndeberáser instanciado mediante unaprobadofunción hash o unaaprobado Función de salida extensible (XOF), como se describe a continuación. Las funciones hash y XOF relevantes se describen en detalle en FIPS 202 [7]. Se utilizarán de la siguiente manera.

SHA3-256 y SHA3-512 son funciones hash con entrada de longitud variable y salida de longitud fija. En este estándar, las invocaciones de estas funciones en una entrada$METRO$se indicará con SHA3-256($METRO$) y SHA3-512($METRO$), respectivamente.

SHAKE128 y SHAKE256 son XOF con entrada y salida de longitud variable. Invocaciones de estas funciones en una entrada$METRO$se indicará de dos maneras diferentes, dependiendo de si la longitud de salida deseadaℓ(en bytes) se conoce en el momento de la invocación. Siℓse conoce en el momento de la invocación, la invocación se indicará con SHAKE128($METRO, \ell$)o AGITAR256($METRO, \ell$). Para SHAKE128, a veces no se conocerá la longitud de salida en el momento de la invocación; en esos casos, la invocación se indicará con SHAKE128($METRO$)y la rutina de hash se comportará como un flujo de bytes que proporciona bytes pseudoaleatorios (realizando rondas de "compresión" adicionales [7]) hasta que no se necesiten más bytes.

Las funciones anteriores desempeñarán varios roles diferentes en los algoritmos especificados en este estándar. Será conveniente asignar una notación específica a cada uno de estos roles, como sigue.

Función pseudoaleatoria (PRF).La funciónPRFtoma un parámetro$\eta \in \{2,3\}$, una entrada de 32 bytes y una entrada de 1 byte. Produce uno $(64 \cdot \eta)$-byte de salida. Se denotará por$\text{PRF}: \{2,3\} \times \mathbb{B}^{32} \times \mathbb{B} \rightarrow \mathbb{B}^{64\eta}$,y esodeberáser instanciado como

$$\text{PRF}_\eta(s, b) := \text{AGITAR256}(s \| b, 64 \cdot \eta), \tag{4.1}$$

dónde$\eta \in \{2,3\}$,$s \in \mathbb{B}^{32}$, y$b \in \mathbb{B}$.Aquí,ηsolo se utiliza para especificar la longitud de salida deseada y no para realizar la separación de dominios. Tenga en cuenta que el parámetro de longitud de salida para SHAKE256 se especifica en bytes.

Función de salida extensible (XOF).La funciónXOFtoma una entrada de 32 bytes y dos entradas de 1 byte. Produce una salida de longitud variable. Esta función se denotará por$\text{XOF}: \mathbb{B}^{32} \times \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}^*$, y esodeberáser instanciado como

$$\text{XOF}(\rho, yo, j) := \text{AGITAR128}(\rho \| i \| j), \tag{4.2}$$

dónde$\rho \in \mathbb{B}^{32}$,$i \in \mathbb{B}$,y$j \in \mathbb{B}$.La funciónXOFsólo se invocará para proporcionar un flujo de bytes pseudoaleatorios para el algoritmo de muestreoMuestraNTT(Algoritmo6). ComoMuestraNTT realiza un muestreo de rechazo, el número total de bytes necesarios no se conocerá en el momento en que XOF es invocado.

Tres funciones hash.La especificación también hará uso de tres instancias de función hash. h,j, y $GRAMO$, como sigue.

Las funciones$h$y$j$cada uno toma una entrada de longitud variable y produce una salida de 32 bytes. Se denotarán por$h$:$B^{*}\rightarrow B^{32}$yj:$B^{*}\rightarrow B^{32}$, respectivamente, ydeberáser instanciado como

$$h(s) := \text{SHA3-256}(s) \qquad \text{y} \qquad j(s) := \text{AGITAR256}(s, 32) \tag{4.3}$$

dónde$s\in B^{*}$.

La función$GRAMO$toma una entrada de longitud variable y produce dos salidas de 32 bytes. Se denotará por $GRAMO$:$B^{*}\rightarrow B^{32}\times B^{32}$. Las dos salidas de$GRAMO$se denotará por, por ejemplo, $(a, b)\leftarrow GRAMO(C)$, dónde $a, b\in B^{32}$, $C\in B^{*}$, y$GRAMO(C) = a\|b$. La función$GRAMO$deberáser instanciado como

$$GRAMO(C) := \text{SHA3-512}(C). \tag{4.4}$$

4.2 Algoritmos generales

Esta sección especifica una serie de algoritmos que se utilizarán como subrutinas en los principales algoritmos de ML-KEM. Para una discusión sobre cómo interpretar el pseudocódigo de estos algoritmos, consulte la Sección2.4.

4.2.1 Algoritmos de conversión y compresión

Esta sección especifica varios algoritmos para convertir entre matrices de bits, matrices de bytes y matrices de números enteros.$metro$. También especifica una determinada operación de compresión para módulos de números enteros.q, así como la correspondiente operación de descompresión.

Conversión entre bits y bytes.Algoritmos2y3convertir entre matrices de bits y matrices de bytes. Las entradas aBits a bytesy las salidas deBytes a bitsson matrices de bits, en las que cada segmento de 8 bits representa un byte en orden little-endian.

Algoritmo 2Bits a bytes(b)

Convierte una cadena de bits (de longitud múltiplo de ocho) en una matriz de bytes.

Aporte:matriz de bits$b\in\{0,1\}^{8\cdot\ell}$.

Producción:matriz de bytes$B\in B^{\ell}$.

1: $B\leftarrow(0,\ldots,0)$
2: para ($i\leftarrow 0$; $yo < 8\ell$; i++)
3: $B[\lfloor i/8\rfloor]\leftarrow B[\lfloor i/8\rfloor]+b[i]\cdot 2^{i \bmod 8}$
4: fin para
5: devolver$B$

Algoritmo 3 Bytes a bits(B)

Realiza la inversa de Bits a bytes*, convirtiendo una matriz de bytes en una matriz de bits.*

Aporte: matriz de bytes $B \in \mathbb{B}^{\ell}$.

Producción: matriz de bits $b \in \{0,1\}^{8 \cdot \ell}$.

```
para (i←0; yo <ℓ; i++)
    para (j←0; j <8; j++)
        b[8i+j] ←B[i] mod 2 B[
        i] ←⌊B[i]/2⌋ fin para

fin para
devolver b
```

Compresión y descompresión. Recordar que q=3329, y tenga en cuenta que la longitud de bits de q es 12. Para re <12, definir

$$\text{Comprimir}_d : \mathbb{Z}_q \longrightarrow \mathbb{Z}_{2^d} \tag{4.5}$$
$$X \mapsto \lceil (2^d/q) \cdot X \rfloor.$$
$$\text{Descomprimir}_d : \mathbb{Z}_{2^d} \longrightarrow \mathbb{Z}_q \tag{4.6}$$
$$y \mapsto \lceil (q/2^d) \cdot y \rfloor.$$

Tenga en cuenta que los tipos de entrada y salida de estas funciones son números enteros módulo $metro$ (ver discusión de tipos en la Sección 2.4). La división y el redondeo en el cálculo de las funciones anteriores se realizan en el conjunto de números racionales. Cálculos de punto flotante no debe ser usado.

Informalmente, Comprimir descarta bits de orden inferior de la entrada, y Descomprimir agrega bits de orden inferior establecidos en cero. Estos algoritmos satisfacen dos propiedades importantes. Primero, la descompresión seguida de la compresión preserva la entrada, es decir, $\text{Comprimir}_d(\text{Descomprimir}_d(y)) = y$ para todos $y \in \mathbb{Z}_q$ y todo re < 12. En segundo lugar, si d es grande (es decir, cerca de 12), lo que significa que el número de bits descartados es pequeño; la compresión seguida de la descompresión no altera significativamente el valor. Específicamente,

$$[\text{Descomprimir}_d(\text{Comprimir}_d(X)) - x] \text{ modificación} \pm q \leq \lceil q/2^{d+1} \rfloor \tag{4.7}$$

para todos $X \in \mathbb{Z}_q$ y todo re <12.

Codificación y decodificación. Los algoritmos Código de bytes (Algoritmo 4) y Decodificación de bytes (Algoritmo 5) se utilizará para la serialización y deserialización de matrices de módulo de números enteros $metro$. Todas las matrices serializadas tendrán una longitud $norte$=256. Código de bytes_d serializa una serie de d enteros de bits en una matriz de $32 \cdot d$ bytes. $\text{Decodificación de bytes}_d$ realiza la operación de deserialización correspondiente, convirtiendo una matriz de $32 \cdot d$ bytes en una matriz de d enteros de bits.

Para la siguiente discusión, es conveniente ver Decodificación de bytes y Código de bytes como convertir entre números enteros y bits. (La conversión entre bits y bytes es sencilla y se realiza utilizando Bits a bytes y Bytes a bits.)

El rango válido de valores para el parámetro d es $1 \le d \le 12$. Las matrices de bits se dividen en d-segmentos de bits. En el caso donde $1 \le d \le 11$, ByteEncode$_d$ convierte cada uno d segmento de bits de la entrada en un entero módulo 2^d, y ByteDecode$_d$ realiza la operación inversa. En este caso la conversión es uno a uno.

El caso $d = 12$ recibe un trato diferente. En este caso, ByteEncode$_{12}$ recibe módulo de números enteros q como entrada, y ByteDecode$_{12}$ produce enteros módulo q como salida. ByteDecode$_{12}$ convierte cada segmento de 12 bits de la entrada en un entero módulo $2^{12} = 4096$, y luego reduce el módulo de resultado q. Esta ya no es una operación uno a uno. De hecho, algunos segmentos de 12 bits podrían corresponder a un número entero mayor que $q = 3329$ pero menos de 4096; sin embargo, esto no puede ocurrir para matrices producidas por ByteEncode$_{12}$. Estos aspectos de ByteDecode$_{12}$ y ByteEncode$_{12}$ será importante al considerar la validación de la clave de encapsulación ML-KEM en la Sección 6.

Algoritmo 4 ByteEncode$_d(F)$

Codifica una matriz de enteros de d bits en una matriz de bytes, por ejemplo $1 \le d \le 12$.

Aporte: matriz de enteros $F \in \mathbb{Z}_{m}^{256}$ donde $m = 2^d$ si $d < 12$ y $m = q$ si $d = 12$.
Producción: matriz de bytes $B \in \mathbb{B}^{32d}$.

```
para (i ← 0; i < 256; i++)
    a ← F[i]                                          ▷ a ∈ ℤ_{2^d}
    para (j ← 0; j < d; j++)
        b[i·d + j] ← a mod 2                          ▷ b ∈ {0,1}^{256·d}
        a ← (a − b[i·d + j])/2                        ▷ nota a − b[i·d + j] siempre es parejo.
    fin para
fin para
B ← BitsToBytes(b)
devolver B
```

Algoritmo 5 ByteDecode$_d(B)$

Decodifica una matriz de bytes en una matriz de enteros de d bits, por ejemplo $1 \le d \le 12$.

Aporte: matriz de bytes $B \in \mathbb{B}^{32d}$.
Producción: matriz de enteros $F \in \mathbb{Z}_{m}^{256}$, donde $m = 2^d$ si $d < 12$ y $m = q$ si $d = 12$.

```
b ← BytesToBits(B)
para (i ← 0; i < 256; i++)
    F[i] ← Σ_{j=0}^{d−1} b[i·d + j] · 2^j   (modificación m)
fin para
devolver F
```

4.2.2 Algoritmos de muestreo

Los algoritmos de ML-KEM requieren dos subrutinas de muestreo que se especifican en Algoritmos 6 y 7. Ambos algoritmos se pueden utilizar para convertir un flujo de bytes uniformemente aleatorios en una muestra de alguna distribución deseada. En este estándar, estos algoritmos se invocarán con un flujo de bytes pseudoaleatorios como entrada. De ello se deduce que el resultado será una muestra de una distribución que es computacionalmente indistinguible de la distribución deseada.

Muestreo uniforme de representaciones NTT.el algoritmoMuestraNTT(Algoritmo6) convierte un flujo de bytes en un polinomio en el dominio NTT. Si el flujo de entrada consta de bytes uniformemente aleatorios, entonces el resultado se extraerá uniformemente al azar det_q. La salida es una matriz. enz256que contiene los coeficientes del elemento muestreado det_q(mira la sección2.4).

Algoritmo 6MuestraNTT(B)

Si la entrada es un flujo de bytes uniformemente aleatorios, la salida es un elemento uniformemente aleatorio de T_q.

Aporte:flujo de bytes$B \in \mathbb{B}^*$.
Producción:formación$a \in \mathbb{Z}_q^{256}$. ▷los coeficientes del NTT de un polinomio

$i \leftarrow 0$
$j \leftarrow 0$
mientras$j < 256$hacer
  $d_1 \leftarrow B[i] + 256 \cdot (B[i+1] \bmod 16)$ $d_2 \leftarrow \lfloor B$
  $[i+1] /$dieciséis$\rfloor + 16 \cdot B[i+2]$ si$d_1 < q$
  entonces
    $a[j] \leftarrow d_1$ ▷$a \in \mathbb{Z}_q^{256}$
    $j \leftarrow j + 1$
  terminara si
  si$d_2 < q$y$j < 256$entonces
    $a[j] \leftarrow d_2$
    $j \leftarrow j + 1$
  terminara si
  $i \leftarrow i + 3$
terminar mientras
devolver$a$

Algoritmo 7MuestraPolyCBD$_\eta$(B)

Si la entrada es un flujo de bytes uniformemente aleatorios, genera una muestra de la distribución.$D_\eta(R_q)$.

Aporte:matriz de bytes$B \in \mathbb{B}^{64\eta}$.
Producción:formación$F \in \mathbb{Z}_q^{256}$. ▷los coeficientes del polinomio muestreado

$b \leftarrow$Bytes a bits(B)
para ($i \leftarrow 0$;$yo < 256$;i++)
  $x \leftarrow \sum_{j=0}^{\eta-1} b[2i\eta + j]$
  $y \leftarrow \sum_{j=0}^{\eta-1} b[2i\eta + \eta + j]$ $F[i] \leftarrow$
  x-$y$modificaciónq ▷$F \in \mathbb{Z}_q^{256}$
fin para
devolver$F$

Muestreo a partir de la distribución binomial centrada.ML-KEM hace uso de una distribución especial$D_\eta(R_q)$de polinomios en$R_q$con coeficientes pequeños. Estos polinomios a veces

denominarse “errores” o “ruido”. La distribución está parametrizada por un número entero.$\eta \in \{2,3\}$. Para muestrear un polinomio de$D_\eta(R_q)$, cada uno de sus coeficientes se muestrea independientemente a partir de una determinada distribución binomial centrada (CBD) enz_q. el algoritmoMuestraPolyCBD(Algoritmo 7) muestra la matriz de coeficientes de un polinomio$F \in R_q$según la distribución$D_\eta(R_q)$, siempre que su entrada sea un flujo de bytes uniformemente aleatorios.

4.3 La transformada de la teoría de números

La transformada de teoría de números (o NTT) puede verse como una versión exacta y especializada de la transformada discreta de Fourier. En el caso de ML-KEM, el NTT se utiliza para mejorar la eficiencia de la multiplicación en el anillo.R_q. Recordar que$R_q$es el anillo$z_q[X]/(X^{norte}+1)$ que consta de polinomios de la forma$F = F_0 + F_1X + \cdots + F_{255}X^{255}$dónde$F_j \in z_q$para todos$j$, equipado con módulo aritmético$X^{norte}+1$.

El anillo$R_q$es naturalmente isomorfo a otro anillo, denotadot_q, que es una suma directa de 128 extensiones cuadráticas dez_q. El NTT es un isomorfismo computacionalmente eficiente entre estos dos anillos. Al ingresar un polinomio$F \in R_q$, el NTT genera un elemento$F := \text{NTT}(F)$del anillo t_q, dónde$F$se llama la “representación NTT” de$F$.La propiedad de isomorfismo implica que

$$F \times_{R_q} gramo = \text{NTT}^{-1}(F \times_{t_q} gramo), \tag{4.8}$$

dónde$\times_{R_q}$y$\times_{t_q}$denotar multiplicación enR_qyt_q, respectivamente. Es más, desde$t_q$es un producto de 128 anillos, cada uno de los cuales consta de polinomios de grado uno, la operación$\times_{t_q}$es mucho más eficiente que la operación$\times_{R_q}$.Por estas razones, el NTT se considera una parte integral de ML-KEM y no simplemente una optimización.

como los anillosR_qy$t_q$tener una estructura de espacio vectorial sobrez_q, el tipo de datos abstracto más natural representar elementos de cualquiera de estos anillos esz_q^{norte}. Por esta razón, la elección de la estructura de datos para las entradas y salidas deNTTyNTT^{-1}son longitud-$norte$matrices de números enteros módulo$q$; Se entiende que estas matrices representan elementos det_qoR_q, respectivamente (ver Sección2.4). AmbosNTT yNTT^{-1}se puede calcular in situ. De hecho, los algoritmos8y9demostrar un medio eficiente de computaciónNTTyNTT^{-1}en su lugar. Sin embargo, para mayor claridad en la comprensión de la distinción de los objetos algebraicos antes y después de la conversión, los algoritmos se escriben con entradas y salidas explícitas.

La estructura matemática de una NTT simple.Recordemos que, en ML-KEM,$q$es el primo $3329 = 2^8 \cdot 13 + 1$ y$norte$=256. Hay 128 raíces primitivas de unidad 256 y ninguna raíz primitiva de unidad 512 enz_q. Tenga en cuenta que$\zeta = 17 \in z_q$es una raíz primitiva número 256 del módulo unitario$q$. De este modo$\zeta^{128} \equiv -1$.

Definir$\text{BitRev}_7(i)$ser el número entero representado mediante la inversión de bits del valor de 7 bits sin signo que corresponde al número entero de entrada$i \in \{0,\ldots,127\}$.

El polinomio$X^{256}+1$ factoriza 128 polinomios de grado 2 módulo$q$como sigue:

$$X^{256}+1 = \prod_{k=0}^{127} X^2 - \zeta^{2\,\text{BitRev}_7(k)+1} \quad . \tag{4.9}$$

Por lo tanto, $R_q := \mathbb{Z}_q[X]/(X^{256}+1)$ es isomorfo a una suma directa de 128 campos de extensión cuadrática dez$q$, denotadot_q. En concreto, este anillo tiene la estructura

$$t_q \overset{\text{METRO}}{:=} \prod_{k=0}^{127} \mathbb{Z}_q[X]/\left(X^2 - \zeta^{2\,\text{BitRev}_7(k)+1}\right). \tag{4.10}$$

Así, la representación de NTT$F \in t_q$de un polinomio$F \in R_q$es el vector que consta de los correspondientes residuos de grado uno:

$$\hat{F} := (\hat{F}\text{modificación } (X^2 - \zeta^{2\,\text{BitRev}_7(0)+1}), \ldots, \hat{F}\text{modificación } (X^2 - \zeta^{2\,\text{BitRev}_7(127)+1})). \tag{4.11}$$

En los algoritmos siguientes,$\hat{F}$se almacena como una matriz de 256 números enteros módulo$q$. Específicamente,

$$\hat{F}\text{modificación } (X^2 - \zeta^{2\,\text{BitRev}_7(i)+1}) = \hat{F}[2i] + \hat{F}[2i+1]X.$$

para$i$de 0 a 127.

Algoritmo 8NTT(F)

Calcula la representación NTT $\hat{f}$ del polinomio f dado$\in R_q$.

Aporte:formación$F \in \mathbb{Z}_{256}^{q}$. ▷los coeficientes del polinomio de entrada

Producción:formación$\hat{F} \in \mathbb{Z}_{256}^{q}$. ▷los coeficientes del NTT del polinomio de entrada

```
1: F̂ ← F                                   ▷calculará NTT in situ en una copia de la matriz de entrada
2: k ← 1
3: para (len ← 128; len ≥ 2; len ← len/2)
4:     para (comenzar ← 0; inicio < 256; comenzar ← comenzar + 2·len)
5:         zeta ← ζ^BitRev7(k) modificación q
6:         k ← k + 1
7:         para (j ← comenzar; j < inicio + len; j++)
8:             t ← zeta · f̂[j + len]                ▷pasos 8-10 módulo hecho q
9:             f̂[j + len] ← f̂[j] − t
10:            f̂[j] ← f̂[j] + t
11:        fin para
12:    fin para
13: fin para
14: devolver F̂
```

Los algoritmos ML-KEM NTT.Un algoritmo para laNTTse describe en Algoritmo8. Un algoritmo para el NTT inverso se describe en Algoritmo9. Estos dos algoritmos están sobrecargados en este estándar. Primero, representan la transformación utilizada para mapear elementos deR_qa elementos det_q(usandoNTT) y viceversa (usandoNTT^{-1}). En segundo lugar, representan la transformación coordinada de estructuras sobre esos anillos; Específicamente, asignan matrices/vectores con entradas enR_qa matrices/vectores con entradas ent_q(usandoNTT) y viceversa (usandoNTT^{-1}).

Algoritmo 9 $\text{NTT}^{-1}(F)$

Calcula el polinomio $f \in R_q$ correspondiente a la representación NTT dada $\hat{f} \in t_q$.

Aporte: formación $F \in \mathbb{Z}_{256}^{q}$. ▷ los coeficientes de representación NTT de entrada
Producción: formación $F \in \mathbb{Z}_{256}^{q}$. ▷ los coeficientes del NTT inverso de la entrada

1: $F \leftarrow \hat{f}$ ▷ calculará en el lugar en una copia de la matriz de entrada
2: $k \leftarrow 127$
3: para ($len \leftarrow 2; len \leq 128; len \leftarrow 2 \cdot len$)
4: para ($comenzar \leftarrow 0; inicio < 256; comenzar \leftarrow comenzar + 2 \cdot len$)
5: $zeta \leftarrow \zeta^{\text{BitRev}_7(k)} \bmod q$
6: $k \leftarrow k - 1$
7: para ($j \leftarrow comenzar; j < inicio + len; j{+}{+}$)
8: $t \leftarrow F[j]$
9: $F[j] \leftarrow t + F[j + len]$ ▷ pasos 9-10 módulo hecho q
10: $F[j + len] \leftarrow zeta \cdot (F[j + len] - t)$ fin
11: para
12: fin para
13: fin para
14: $F \leftarrow F \cdot 3303 \bmod q$ ▷ multiplica cada entrada por $3303 \equiv 128^{-1} \bmod q$
15: devolver F

4.3.1 Multiplicación en el Dominio NTT

Como se discutió en la Sección 2.4, suma y multiplicación escalar de elementos de t_q es sencillo: se puede hacer usando las correspondientes operaciones aritméticas de coordenadas en las matrices de coeficientes. Esta sección describe cómo realizar la operación del anillo restante (es decir, multiplicación en t_q).

Recordar de (4.11) eso $F \in t_q$ es un vector de polinomios cuadráticos de módulo de residuo uno de grado 128. Algebraicamente, multiplicación en el anillo. t_q Consiste en una multiplicación independiente en cada una de las 128 coordenadas con respecto al módulo cuadrático de esa coordenada. Específicamente, el i-ésima coordenada en t_q del producto $\hat{h} = F \times_{t_q} gramo$ está determinado por el cálculo

$$\hat{h}[2i] + \hat{h}[2i+1]X = (F[2i] + F[2i+1]X)(gramo[2i] + gramo[2i+1]X) \bmod (X^2 - \zeta^{2\,\text{BitRev}_7(i)+1}). \quad (4.12)$$

Por tanto, se puede calcular el producto de dos elementos de t_q usando el algoritmo Multiplicar NTT (Algoritmo 10). Tenga en cuenta que Multiplicar NTT usos BaseCaseMultiplicar (Algoritmo 11) como una subrutina. Como se discutió en la Sección 2.4, Multiplicar NTT permite realizar operaciones aritméticas algebraicas lineales con matrices y vectores con entradas en t_q.

Algoritmo 10Multiplicar NTT($\hat{f}$, $\hat{g}$)

Calcula el producto (en el anillo T_q) de dos representaciones de NTT.

Aporte:Dos matrices$F\in\mathbb{Z}_q^{256}$y$gramo\in\mathbb{Z}_q^{256}$. ▷los coeficientes de dos representaciones NTT

Producción:Una matriz$\hat{h}\in\mathbb{Z}_q^{256}$. ▷los coeficientes del producto de las entradas

1:para ($i\leftarrow 0$;yo <128;i++)

2: ($\hat{h}[2i]$, $\hat{h}[2i+1]$)$\leftarrow$BaseCaseMultiplicar($F[2i]$, $f[2i+1]$, $\hat{g}[2i]$, $\hat{g}[2i+1]$,$\zeta^{2\text{BitRev}_7(i)+1}$)

3:fin para

4:devolver$\hat{h}$

Algoritmo 11BaseCaseMultiplicar(a_0,a_1,b_0,b_1, y)

Calcula el producto de dos polinomios de grado uno con respecto a un módulo cuadrático.

Aporte:$a_0,a_1,b_0,b_1\in\mathbb{Z}_q$. ▷los coeficientes de$a_0+a_1Xyb_0+b_1X$

Aporte:$y\in\mathbb{Z}_q$. Producción: ▷el módulo esX^2-y

$C_0,C_1\in\mathbb{Z}_q$. ▷los coeficientes del producto de los dos polinomios

1:$C_0\leftarrow a_0\cdot b_0+a_1\cdot b_1\cdot y$ ▷pasos1-2módulo hecho$q$

2:$C_1\leftarrow a_0\cdot b_1+a_1\cdot b_0$

3:devolverC_0,C_1

5. El esquema de componentes K-PKE

Esta sección describe el esquema de componentes K-PKE. Como se discutió en la Sección3.3, K-PKE es no aprovadopara su uso de forma independiente. Sirve sólo como una colección de subrutinas para su uso en los algoritmos delaprobadoesquema ML-KEM, como se describe en la Sección6.

K-PKE consta de tres algoritmos:

1. Generación de claves (K-PKE.Generación de claves);
2. Cifrado (K-PKE.cifrar);
3. Descifrado (K-PKE.Descifrar).

Cuando se crea una instancia de K-PKE como parte de ML-KEM, K-PKE hereda el conjunto de parámetros seleccionado para ML-KEM. Cada conjunto de parámetros especifica valores numéricos para cada parámetro. Mientras$norte$es siempre 256 y$q$es siempre 3329, los valores de los parámetros restantes$k$,η_1,η_2,d_{tu}, y$d_v$varían entre los tres conjuntos de parámetros. Los parámetros individuales y los conjuntos de parámetros se describen en el capítulo7.

Los algoritmos de esta sección no realizan ninguna validación de entrada. Esto se debe a que sólo se invocan como subrutinas de los principales algoritmos ML-KEM. Los algoritmos de ML-KEM realizan la validación de entradas según sea necesario; también garantizan que todas las entradas a los algoritmos K-PKE (invocadas como subrutinas) serán válidas.

Cada uno de los algoritmos de K-PKE a continuación va acompañado de una breve descripción informal en texto. Para simplificar, esta descripción está escrita en términos de vectores y matrices cuyas entradas son elementos deR_q. En el algoritmo real, la mayoría de los cálculos ocurren en el dominio NTT para mejorar la eficiencia de la multiplicación. Los vectores y matrices relevantes tendrán entonces entradas ent_q. Aritmética algebraica lineal con tales vectores y matrices (ver, por ejemplo, línea19 deK-PKE.Generación de claves) se realiza como se describe en las Secciones2.4y4.3.1. La clave de cifrado y descifrado de K-PKE también se almacena en formato NTT.

5.1 Generación de claves K-PKE

El algoritmo de generación de claves.K-PKE.Generación de clavesde K-PKE (Algoritmo12) no recibe información, requiere aleatoriedad y genera una clave de cifradoek$_{\text{PKE}}$y una clave de descifradodk$_{\text{PKE}}$. Desde el punto de vista típico del cifrado de clave pública, la clave de cifrado puede hacerse pública, mientras que la clave de descifrado y la aleatoriedad deben permanecer privadas. Este será el caso también en el contexto de esta norma. De hecho, la clave de cifrado de K-PKE servirá como clave de encapsulación de ML-KEM (verML-KEM.Generación de clavesa continuación) y por lo tanto puede hacerse público; mientras tanto, la clave de descifrado y la aleatoriedad deK-PKE.Generación de clavesdeben permanecer privados ya que se pueden usar para realizar la decapsulación en ML-KEM.

Descripción informal.La clave de descifrado deK-PKE.Generación de claveses una longitud-$k$vectorsde elementos deR_q, es decir,$s \in R_q^k$. Mas o menos,ses un conjunto de variables secretas, mientras que la clave de cifrado es un colección de ecuaciones lineales "ruidosas" (A,Como+e)en las variables secretass.Las filas de la matriz. A Forme los coeficientes de la ecuación. Esta matriz se genera pseudoaleatoriamente usandoxof,con solo la semilla almacenada en la clave de cifrado. El secretosy el “ruido”mise toman muestras de la

Algoritmo 12K-PKE.Generación de claves()

Genera una clave de cifrado y una clave de descifrado correspondiente.

Producción:Clave de encriptaciónek$_{PKE} \in B^{384k+32}$.

Producción:clave de descifradodk$_{PKE} \in B^{384k}$.

1: $d \leftarrow B^{32}$ ▷des de 32 bytes aleatorios (consulte la Sección3.3)

2: $(\rho,\sigma) \leftarrow G(d)$ ▷expandirse a dos semillas pseudoaleatorias de 32 bytes

3: $N \leftarrow 0$

4: para ($i \leftarrow 0; i < k; i$++) ▷generar matriz$\in (\mathbb{Z}_{256}^{256})^{k \times k}$

5: para ($j \leftarrow 0; j < k; j$++)

6: $A[i,j] \leftarrow$MuestraNTT(XOF(ρ, i, j)) ▷cada entrada de uniforme en el dominio NTT

7: fin para

8: fin para

9: para ($i \leftarrow 0; i < k; i$++) ▷generars$\in (\mathbb{Z}_{256}^{256})^{k}$

10: $s[i] \leftarrow$MuestraPolyCBD$_{\eta_1}$(PRF$_{\eta_1}(\sigma, N)$) ▷s$[i] \in \mathbb{Z}_{256}^{256}$muestreado de CBD

11: $N \leftarrow N+1$

12: fin para

13: para ($i \leftarrow 0; i < k; i$++) ▷generarmi$\in (\mathbb{Z}_{256}^{256})^{k}$

14: mi$[i] \leftarrow$MuestraPolyCBD$_{\eta_1}$(PRF$_{\eta_1}(\rho, N)$) ▷mi$[i] \in \mathbb{Z}_{256}^{256}$muestreado de CBD

15: $N \leftarrow N+1$

dieciséis: fin para

17: $s \leftarrow$ NTT(s) ▷NTTse ejecuta$k$veces (una vez por cada coordenada des)

18: mi $\leftarrow$ NTT(mi) ▷NTTse ejecuta$k$veces

19: $\hat{t} \leftarrow \hat{A} \circ \hat{s} + \hat{e}$ ▷sistema lineal ruidoso en el dominio NTT

20: ek$_{PKE} \leftarrow$Código de bytes$_{12}(\hat{t}) \| \rho$ ▷Código de bytes$_{12}$se ejecuta$k$veces; incluir semilla para

21: dk$_{PKE} \leftarrow$Código de bytes$_{12}$(s) ▷Código de bytes$_{12}$se ejecuta$k$veces

22: devolver (ek$_{PKE}$,dk$_{PKE}$)

906 distribución binomial centrada usando aleatoriedad expandida desde una semilla víaPRF.Una vez
907 Ays ymise generan, el cálculo de la parte finalt = Como+ede la clave de cifrado tiene lugar.

908 EnK-PKE.Generación de claves, la elección del conjunto de parámetros afecta la longitud del secretos (a través del
909 parámetro k) y, como consecuencia, los tamaños del vector de ruido.miy la matriz pseudoaleatoriaA.La elección del
910 conjunto de parámetros también afecta a la distribución del ruido (a través del parámetroη_1) utilizado para muestrear las
911 entradas desymi.

912 5.2 Cifrado K-PKE

913 El algoritmo de cifradoK-PKE.cifrarde K-PKE (Algoritmo13) toma una clave de cifrado ek$_{PKE}$y un texto plano
914 *metro*como entrada, requiere aleatoriedad$r$y genera un texto cifrado$C$. Si bien muchos algoritmos
915 especificados en este documento requieren aleatoriedad, sólo la descripción deK-PKE.cifrar interpreta esta
916 aleatoriedad como parte de la entrada. Esto se debe a que ML-KEM necesitará invocar K-PKE.cifrarcon una
917 elección específica de aleatoriedad (ver Algoritmodieciséispara detalles).

918

Descripción informal.el algoritmoK-PKE.cifrarcomienza extrayendo el vectorty la semilla de la clave de cifrado. Luego la semilla se expande para regenerar la matriz.A,de la misma manera que se hizo enK-PKE.Generación de claves. SityAse derivan correctamente de una clave de cifrado producida porK-PKE. Generación de claves, entonces son iguales a sus valores correspondientes enK-PKE.Generación de claves.

Algoritmo 13K-PKE.cifrar(ek_{PKE}, *señor*)

Utiliza la clave de cifrado para cifrar un mensaje de texto sin formato utilizando la aleatoriedad r.

Aporte:Clave de encriptación$ek_{PKE} \in B^{384k+32}$.
Aporte:mensaje*metro*$\in B^{32}$.
Aporte:aleatoriedad del cifrado*r*$\in B^{32}$.
Producción:texto cifrado$C \in B^{32(d_{tu}k+d_v)}$.

1: $norte \leftarrow 0$
2: $\hat{t} \leftarrow$ Decodificación de bytes$_{12}(ek_{PKE}[0:384k])$
3: $\rho \leftarrow ek_{PKE}[384k:384k+32]$ ▷extraer semilla de 32 bytes deek_{PKE}
4: para ($i \leftarrow 0; yo < k; i$++) ▷regenerar matriz$\in (Z_{256})_q^{k \times k}$
5: para ($j \leftarrow 0; j < k; j$++)
6: $A[yo, j] \leftarrow$ MuestraNTT(XOF(ρ,*yo*, *j*)) fin
7: para
8: fin para
9: para ($i \leftarrow 0; yo < k; i$++) ▷generarr$\in (Z_{256})_q^k$
10: $r[i] \leftarrow$ MuestraPolyCBD$_\eta$(PRF$_1$*norte* η_1 (*r*,*norte*)) ▷r[*i*]$\in Z_{256}$ muestreado de CBD
11: $\leftarrow norte+1$
12: fin para
13: para ($i \leftarrow 0; yo < k; i$++) ▷generarmi 1$\in (Z_{25_q})_k^6$
14: $mi_1[i] \leftarrow$ MuestraPolyCBD$_\eta$(PRF$_2$ *norte*η_2(*r*,*norte*)) ▷mi1[*i*]$\in Z_{256}$muestreado de CBD
15: $\leftarrow norte+1$
dieciséis: fin para
17: $mi_2 \leftarrow$ MuestraPolyCBD$_\eta$(PRF $_{\eta_2}$(*r*,*norte*)) ▷muestra$mi_2 \in Z_{256}$del CDB
18: $\hat{r} \leftarrow$ NTT(r) ▷NTTse ejecuta*k*veces
19: $tu \leftarrow NTT^{-1}(A^T \circ \hat{r}) + mi_1$ ▷NTT^{-1}se ejecuta*k*veces
20: $\mu \leftarrow$ Descomprimir$_1$(Decodificación de bytes$_1$(*metro*)))
21: $v \leftarrow NTT^{-1}(\hat{t}^T \circ \hat{r}) + mi_2 + \mu$ ▷codificar texto plano*metro*en polinomio*v*.
22: $C_1 \leftarrow$ Código de bytes$_{d_{tu}}$(Comprimir$_{d_{tu}}$(*tu*)) ▷Código de bytes$_{d_{tu}}$se ejecuta*k*veces
23: $C_2 \leftarrow$ Código de bytes$_{d_v}$(Comprimir$_{d_v}$(*v*))
24: devolver $C \leftarrow (C_1 \| C_2)$

Recuerde de la descripción de la generación de claves que el par (A,t = Como+e)puede considerarse como un sistema de ecuaciones lineales ruidosas en las variables secretass.Se puede generar una ecuación lineal ruidosa adicional en las mismas variables secretas, sin sabers -eligiendo una combinación lineal aleatoria de las ecuaciones ruidosas del sistema (A,t).Luego se puede codificar información en el "término constante" (es decir, la entrada que es una combinación lineal de entradas det)de tal ecuación combinada. Esta información puede luego ser descifrada por una parte en posesión des.Por ejemplo, se podría codificar un solo bit decidiendo si alterar significativamente o no el término constante,

generando así una ecuación casi correcta (correspondiente al valor del bit descifrado de 0) o una ecuación que dista mucho de ser correcta (correspondiente al valor del bit descifrado de 1). En el caso de K-PKE, el término constante es un polinomio con 256 coeficientes, por lo que se puede codificar más información: un bit en cada coeficiente.

Para tal fin,K-PKE.cifrarprocede generando un vector$r \in R_k$ $_q$y términos de ruido$mi_1 \in R_k$ $_q$y $mi_2 \in R_q$, todos los cuales se muestrean a partir de la distribución binomial centrada utilizando pseudoaleatoriedad expandida (a través dePRF)de la aleatoriedad de entrada$r$. Luego se calcula la "nueva ecuación ruidosa" que (hasta algunos detalles) se calcula mediante $(tu, v) \leftarrow (A_T r + e_1, t_T r + mi_2)$. Una codificación apropiada μdel mensaje de entrada *metro*luego se agrega al términot$_T r + mi_2$. Finalmente, la pareja (tu,v)se comprime, se serializa en una matriz de bytes y se genera como texto cifrado.

5.3 Descifrado K-PKE

El algoritmo de descifradoK-PKE.Descifrarde K-PKE (Algoritmo14) toma una clave de descifrado dk$_{PKE}$y un texto cifrado$C$como entrada, no requiere aleatoriedad y genera un texto sin formato*metro*.

Descripción informal.el algoritmoK-PKE.Descifrarcomienza calculando la “ecuación ruidosa” (tú,v)subyacente al texto cifrado$C$, como se comenta en la descripción deK-PKE.cifrar. Aquí uno puede pensar entucomo los coeficientes de la ecuación y$v$como término constante. Recuerde que la clave de descifradodk$_{PKE}$contiene el vector de variables secretass.Por tanto, el algoritmo de descifrado puede utilizar la clave de descifrado para calcular el término constante verdadero.$v' = s_T tu$y luego calcular$v - v'$. El algoritmo de descifrado finaliza decodificando el mensaje de texto plano.*metro*de$v - v'$y salida*metro*.

Algoritmo 14K-PKE.Descifrar(no sé$_{PKE}$, C)

Utiliza la clave de descifrado para descifrar un texto cifrado.

Aporte:clave de descifradodk$_{PKE} \in B^{384k}$.
Aporte:texto cifrado$C \in B^{32(d_{tu}k + d_v)}$.
Producción:mensaje*metro*$\in B^{32}$.

1: $C_1 \leftarrow C[0 : 32 d_{tu} k]$
2: $C_2 \leftarrow C[32 d_{tu} k : 32(d_{tu} k + d_v)]$
3: tu $\leftarrow$ Descomprimir$_d$(Decodificación de bytes$_{d_{tu}}$(C_1)) ▷Decodificación de bytes$_{d_{tu}}$invocado$k$veces
4: $v \leftarrow$ Descomprimir$_d$(Decodificación de bytes$_{d_v}$(C_2)) ▷decodificar texto plano*metro*del polinomio$v$
5: s $\leftarrow$ Decodificación de bytes$_{12}$(no sé$_{PKE}$)
6: $w \leftarrow v - $NTT^{-1}($s_T \circ$NTT(tú)) ▷NTT^{-1}yNTTinvocado$k$veces
7: *metro* $\leftarrow$ Código de bytes$_1$(Comprimir$_1$(w))
8: devolver*metro*

---

6. El mecanismo de encapsulación de claves ML-KEM

El esquema ML-KEM consta de tres algoritmos:

1. Generación de claves (ML-KEM.Generación de claves)
2. Encapsulación (ML-KEM.encapsula)
3. Decapsulación (ML-KEM.decápsulas)

Para crear una instancia de ML-KEM, se debe seleccionar un conjunto de parámetros, cada uno de los cuales está asociado con una compensación particular entre seguridad y rendimiento. Los tres conjuntos de parámetros posibles se denominan ML-KEM-512, ML-KEM-768 y ML-KEM-1024 y se describen en detalle en la Tabla2de Sección7 . Cada conjunto de parámetros asigna valores numéricos específicos a los parámetros individuales.$norte$, q,k,η_1,η_2,d_{tu}, yd_v. Mientras$norte$es siempre 256 y$q$es siempre 3329, los parámetros restantes varían entre los tres conjuntos de parámetros. Implementadoresdeberáasegúrese de que los tres algoritmos de ML-KEM enumerados anteriormente solo se invoquen con un conjunto de parámetros válido y que este conjunto de parámetros se seleccione de manera adecuada para la aplicación deseada. Además, los algoritmosML-KEM.encapsula yML-KEM. decápsulasrequieren validación de las entradas, como se analiza a continuación.

6.1 Generación de claves ML-KEM

El algoritmo de generación de claves.ML-KEM.Generación de clavespara ML-KEM (Algoritmo15) no acepta ninguna entrada, requiere aleatoriedad y produce una clave de encapsulación y una clave de decapsulación. Si bien la clave de encapsulación se puede hacer pública, la clave de decapsulación debe permanecer privada.

Descripción informal.La subrutina central deML-KEM.Generación de claveses el algoritmo de generación de claves de K-PKE (Algoritmo12). La clave de encapsulación ML-KEM es simplemente la clave de cifrado de K-PKE. La clave de desencapsulación de ML-KEM se compone de la clave de descifrado de K-PKE, la clave de encapsulación, un hash de la clave de encapsulación y un valor pseudoaleatorio de 32 bytes. Este valor aleatorio se utilizará en el mecanismo de "rechazo implícito" del algoritmo de decapsulación (Algoritmo17).

Algoritmo 15ML-KEM.Generación de claves()

Genera una clave de encapsulación y una clave de decapsulación correspondiente.

Producción:Clave de encapsulaciónek$\in B^{384k+32}$.

Producción:Clave de decapsulaciónndk$\in B^{768k+96}$.

1:$z \leftarrow_{ps} B^{32}$ ▷zes de 32 bytes aleatorios (consulte la Sección3.3)

2:$(ek_{PKE}, dk_{PKE}) \leftarrow$K-PKE.Generación de claves() ▷ejecutar generación de claves para K-PKE

3:$ek \leftarrow ek_{PKE}$ ▷La clave de encapsulación KEM es solo la clave de cifrado PKE

4:$dk \leftarrow (dk_{PKE} \| ek \| h(ek) \| z)$ ▷La clave de descifrado de KEM incluye la clave de descifrado de PKE

5:devolver (ek,NS)

6.2 Encapsulación ML-KEM

El algoritmo de encapsulaciónML-KEM.encapsulade ML-KEM (Algoritmodieciséis) acepta una clave de encapsulación como entrada, requiere aleatoriedad y genera un texto cifrado y una clave compartida.

Validación de entrada.Para validar una entrada determinada[1]mimikaML-KEM.encapsula, realice las siguientes comprobaciones.

1. *(Escriba verificación.)*Simimikno es una matriz de bytes de longitud 384k+32 por el valor de$k$especificado por el conjunto de parámetros correspondiente, la entrada no es válida.

2. *(Verificación de módulo.)*realizar el cálculoek←Código de bytes12(Decodificación de bytes12(mimik)).Si ek ≠ mi mik,la entrada no es válida. (Mira la sección4.2.1.)

Si cualquiera de las comprobaciones anteriores declara que la entrada no es válida, entoncesML-KEM.encapsulano debeser realizado con entradamimik.En cambio, los pasos apropiados para la aplicacióndeberáser llevado a abortar. Si se pasan las dos comprobaciones anteriores (es decir, ninguna de ellas declara que la entrada no es válida), entonces la entrada es considerado válido yML-KEM.encapsulase puede realizar con entradaek = mimik.

Es importante tener en cuenta que el proceso de validación de entrada anterior no garantiza quemimikes una salida real deML-KEM.Generación de claves. De hecho, la capacidad de garantizar eso (sin utilizar la clave de decapsulación) violaría el supuesto de seguridad.

Recuerde que, como se discutió en la Sección3.3, las implementaciones solo son necesarias para reproducir correctamente el comportamiento de entrada-salida de los algoritmos de nivel superior. En el caso deML-KEM.encapsula, esto significa que una implementación puede realizar cualquier proceso que sea equivalente a ejecutar las comprobaciones 1 y 2 anteriores y luego ejecutar el algoritmodieciséis. (Por ejemplo, la segunda verificación podría realizarse durante la ejecución deDecodificación de bytes12en línea2deK-PKE.cifrar.)

Algoritmo 16ML-KEM.encapsula(ek)

Utiliza la clave de encapsulación para generar una clave compartida y un texto cifrado asociado.

Entrada validada:clave de encapsulaciónek$\in \mathbb{B}^{384k+32}$.
Producción:llave compartida$k\in\mathbb{B}^{32}$. Producción:texto cifrado$C\in\mathbb{B}^{32(d_tuk+d_v)}$.

1: $metro \xleftarrow{ps} \mathbb{B}^{32}$ ▷$metro$es de 32 bytes aleatorios (consulte la Sección3.3)
2: $(k,r)\leftarrow G(metro\|h(\text{ek}))$ ▷derivar clave secreta compartida$k$y aleatoriedad$r$
3: $C\leftarrow$K-PKE.cifrar(ek,$metro$,r) ▷cifrar$metro$usando K-PKE con aleatoriedad$r$
4:devolver (k,c)

Descripción informal.La subrutina central deML-KEM.encapsulaes el algoritmo de cifrado de K-PKE, que se utiliza para cifrar un valor aleatorio$metro$en un texto cifrado$C$. Una copia del secreto compartido.ky la aleatoriedad utilizada durante el cifrado se derivan de$metro$y la encapsulación

[1] En discusiones sobre validación de entradas, la tilde en la notación indica que es posible que la entrada no esté formada correctamente, p.ej,mimikpara una entrada clave de encapsulación candidata, a diferencia deekpara una entrada válida.

llaveekmediante hash. Específicamente,*h*es aplicado abueno,y el resultado se concatena con*metro*y luego hash usando*GRAMO*. El algoritmo se completa generando el texto cifrado.*C*y el secreto compartido*k*.

6.3 Decapsulación ML-KEM

El algoritmo de decapsulaciónML-KEM.decápsulasde ML-KEM (Algoritmodieciséis) acepta una clave de decapsulación y un texto cifrado ML-KEM como entrada, no utiliza ninguna aleatoriedad y genera un secreto compartido.

Validación de entrada.Para validar un par de entradas determinado*C*mi(texto cifrado candidato) ydmik (clave de decapsulación candidata) aML-KEM.decápsulas, realice las siguientes comprobaciones.

1. *(Verificación del tipo de texto cifrado.)*Si*C*mino es una matriz de bytes de longitud 32($d_{tu}k+d_v$)por los valores ded_{tu}, d_v, y*k* especificado por el conjunto de parámetros correspondiente, la entrada no es válida.

2. *(Verificación del tipo de clave de decapsulación).*Sidmikno es una matriz de bytes de longitud 768*k*+96 por el valor de*k* especificado por el conjunto de parámetros correspondiente, la entrada no es válida.

Si cualquiera de las comprobaciones anteriores declara que la entrada no es válida, entoncesML-KEM.decápsulasno debe realizarse con entrada (*C*mi,dmik).En cambio, los pasos apropiados para la aplicacióndeberáser llevado a abortar. Si ambas comprobaciones pasan (es decir, ninguna declara que la entrada no sea válida), entonces la entrada es considerado válido yML-KEM.decápsulasse puede realizar con entrada (*C*,dk) = (*C*mi,dmik).

Para algunas aplicaciones, validación adicional de la clave de decapsulación.dmikpuede ser apropiado. Por ejemplo, en los casos en quedmikfue generado por un tercero, los usuarios pueden querer asegurarse de que los cuatro componentes dedmiktener la relación correcta entre sí, como en línea4deML-KEM.Generación de claves. En todos los casos, los implementadoresdeberávalidar las entradas paraML-KEM.decápsulasde manera adecuada para su aplicación.

Descripción informal.el algoritmoML-KEM.decápsulascomienza analizando los componentes de la clave de decapsulacióndkde ML-KEM. Estos componentes son un par (clave de cifrado, clave de descifrado) para K-PKE, un valor hash*h*y un valor aleatorio*z*. Luego, la clave de descifrado de K-PKE se utiliza para descifrar el texto cifrado de entrada.*C*para obtener un texto plano$metro'$. Luego, el algoritmo de decapsulación vuelve a cifrar*metro*y calcula una clave secreta compartida candidatak'de la misma manera que debería haberse hecho en encapsulación. Específicamente,k'y la aleatoriedad del cifrador'se calculan mediante hash*metro*y la clave de cifrado de K-PKE, y un texto cifradoC'se genera cifrando$metro'$ usando aleatoriedadr'. Finalmente, la decapsulación comprueba si el texto cifrado resultanteC'coincide con el texto cifrado proporcionado*C*. Si no es así, el algoritmo realiza un “rechazo implícito”: el valor dek'se cambia a un hash de*C*junto con el valor aleatorio*z*almacenado en la clave secreta ML-KEM (consulte la discusión sobre fallas de decapsulación en la Sección3.2). En cualquier caso, la decapsulación genera la clave secreta compartida resultante.k'.

Algoritmo 17ML-KEM.decápsulas(C,NS)

Utiliza la clave de decapsulación para generar una clave compartida a partir de un texto cifrado.

Entrada validada:texto cifrado $C \in B32(d_{tuk}+d_v)$. Entrada validada:clave de decapsulacióndk $\in B768k+96$. Producción: llave compartida $k \in B32$.

1: $dk_{PKE} \leftarrow dk[0: 384k]$ ▷extraer (de la clave de descodificación de KEM) la clave de descifrado de PKE
2: $ek_{PKE} \leftarrow dk[384k: 768k+32]$ ▷extraer la clave de cifrado PKE
3: $h \leftarrow dk[768k+32: 768k+64]$ ▷extraer el hash de la clave de cifrado PKE
4: $z \leftarrow dk[768k+64: 768k+96]$ ▷extraer el valor de rechazo implícito
5: $metro' \leftarrow$K-PKE.Descifrar(no sé$_{PKE}$, C) ▷descifrar texto cifrado
6: $(k', r') \leftarrow GRAMO(metro' \| h)$
7: $\bar{K} \leftarrow J(z \| C, 32)$
8: $C' \leftarrow$K-PKE.cifrar(ek_{PKE}, $metro', r'$) ▷volver a cifrar utilizando la aleatoriedad derivada r'
9: si $C \neq C'$ entonces
10: $k' \leftarrow \bar{K}$ ▷si los textos cifrados no coinciden, "rechazar implícitamente"
11: terminara si
12: devolver k'

7. Conjuntos de parámetros

ML-KEM está equipado con tres conjuntos de parámetros. Cada uno de los tres conjuntos de parámetros se compone de cinco parámetros individuales: k, η_1, η_2, d_{tu}, y d_v. También hay dos constantes: $norte$=256 y q=3329. La siguiente es una descripción breve e informal de los roles que desempeñan los parámetros variables en los algoritmos de K-PKE (y por lo tanto en ML-KEM). Mira la sección5para detalles.

- El parámetro$k$determina las dimensiones de los vectoressymienK-PKE.Generación de claves, así como las dimensiones de la matriz Â y los vectoresre$_1$, ymi$_2$enK-PKE.cifrar.
- El parámetro$\eta_1$es necesario para especificar la distribución para generar los vectoressy mien K-PKE.Generación de clavesy el vectorrenK-PKE.cifrar.
- El parámetro$\eta_2$es necesario para especificar la distribución para generar los vectoresmi$_1$ ymi_2enK-PKE.cifrar.
- Los parametrosd_{tu}y$d_v$sirven como parámetros y entradas para las funcionesComprimir, Descomprimir,Código de bytes, yDecodificación de bytesenK-PKE.cifraryK-PKE.Descifrar.

Esta norma aprueba los conjuntos de parámetros que figuran en la tabla2. Cada conjunto de parámetros está asociado con una fuerza de seguridad requerida para la generación de aleatoriedad (consulte la Sección3.3). Los tamaños de las claves ML-KEM y los textos cifrados para cada conjunto de parámetros se resumen en la Tabla3.

	$norte$	q	k	η_1	η_2	d_{tu}	d_v	fuerza RBG requerida (bits)
ML-KEM-512	256	3329	2	3	2	10	4	128
ML-KEM-768	256	3329	3	2	2	10	4	192
ML-KEM-1024	256	3329	4		2	2	11	5	256

Tabla 2. Conjuntos de parámetros aprobados para ML-KEM

	clave de encapsulación	clave de decapsulación	texto cifrado	clave secreta compartida
ML-KEM-512	800	1632	768	32
ML-KEM-768	1184	2400	1088	32
ML-KEM-1024	1568	3168	1568	32

Tabla 3. Tamaños (en bytes) de claves y textos cifrados de ML-KEM

También se puede decir que el nombre de un conjunto de parámetros denota un KEM (sin parámetros). Específicamente, ML-KEM-X se puede utilizar para indicar el KEM libre de parámetros que resulta de crear una instancia del esquema ML-KEM con el conjunto de parámetros ML-KEM-X.

Los tres conjuntos de parámetros incluidos en la Tabla2fueron diseñados para cumplir con ciertas categorías de resistencia de seguridad definidas por NIST en su convocatoria de propuestas original [4,18]. Estas categorías de fortaleza de seguridad se explican con más detalle en el Apéndice.A.

Con este enfoque, la fortaleza de la seguridad no se describe con un solo número, como “128 bits de seguridad”. En cambio, se afirma que cada conjunto de parámetros ML-KEM es al menos tan seguro como un conjunto de parámetros genérico.

cifrado de bloque con un tamaño de clave prescrito o una función hash genérica con una longitud de salida prescrita. Más precisamente, se afirma que los recursos computacionales necesarios para descifrar ML-KEM son mayores o iguales a los recursos computacionales necesarios para descifrar el cifrado de bloque o la función hash, cuando estos recursos computacionales se estiman utilizando cualquier modelo de cálculo realista. Los diferentes modelos de cálculo pueden ser más o menos realistas y, en consecuencia, conducir a estimaciones más o menos precisas de la solidez de la seguridad. Algunos modelos comúnmente estudiados se analizan en [19].

Concretamente, se afirma que ML-KEM-512 está en la categoría de seguridad 1, ML-KEM-768 se afirma que está en la categoría de seguridad 3 y ML-KEM-1024 se afirma que está en la categoría de seguridad 5. Para una discusión adicional sobre la fortaleza de seguridad de los criptosistemas basados en MLWE, consulte [4].

Seleccionar un conjunto de parámetros apropiado.Al establecer inicialmente protecciones criptográficas para los datos, se debe utilizar el conjunto de parámetros más sólido posible.deberíaser usado. Esto tiene una serie de ventajas, incluida la reducción de la probabilidad de costosas transiciones a conjuntos de parámetros de mayor seguridad en el futuro. Al mismo tiempo, cabe señalar que algunos conjuntos de parámetros pueden tener efectos adversos en el rendimiento de la aplicación correspondiente (por ejemplo, el algoritmo puede ser inaceptablemente lento).

NIST recomienda utilizar ML-KEM-768 como conjunto de parámetros predeterminado, ya que proporciona un gran margen de seguridad a un costo de rendimiento razonable. En los casos en los que esto no sea práctico o cuando se requiera una seguridad aún mayor, se pueden utilizar otros conjuntos de parámetros.

Referencias

[1] NIST. Publicación especial 800-227: Recomendaciones para mecanismos de encapsulación de claves, 2024.

[2] Elaine B. Barker, Lily Chen, Allen L. Roginsky, Apostol Vassilev y Richard Davis. Recomendación para esquemas de establecimiento de claves por pares utilizando criptografía de logaritmos discretos. Informe técnico, publicación especial 800-56A Revisión 3, Departamento de Comercio de EE. UU., Washington, DC, abril de 2018.

[3] Elaine B. Barker, Lily Chen, Allen L. Roginsky, Apostol Vassilev, Richard Davis y Scott Simon. Recomendación para el establecimiento de claves por pares utilizando criptografía de factorización de enteros. Informe técnico, publicación especial 800-56B Revisión 2, Departamento de Comercio de EE. UU., Washington, DC, marzo de 2019.

[4] Robert Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler y Damien Stehlé. Especificaciones del algoritmo CRYSTALS-Kyber y documentación de respaldo. Presentación de tercera ronda al proceso de estandarización de criptografía poscuántica del NIST, 2020.https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[5] Equipo de presentación de CRYSTALS-Kyber. "Discusión sobre la transformación FO modificada de Kyber". Publicación del foro en pqc-forum, disponible enhttps://groups.google.com/a/list.nist.gov/g/pqcforum/c/WFRDl8DqYQ4, 2023.

[6] Equipo de presentación de CRYSTALS-Kyber. "Decisiones Kyber, parte 2: Transformación FO". Publicación del foro en pqc-forum, disponible enhttps://groups.google.com/a/list.nist.gov/g/pqc-forum/c/C0D3 W1KoINY/m/99kIvydoAwAJ, 2023.

[7] Instituto Nacional de Normas y Tecnología. Estándar SHA-3: hash basado en permutaciones y funciones de salida extensible. (Departamento de Comercio de EE. UU., Washington, DC), Publicación de estándares federales de procesamiento de información (FIPS) 202, agosto de 2015.https://doi.org/10.6028/NIST.FIPS.202.

[8] Joppe Bos, Léo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler y Damien Stehlé. CRYSTALS-Kyber: un KEM basado en celosía de módulo seguro CCA. En*Simposio europeo IEEE sobre seguridad y privacidad 2018 (EuroS&P)*, páginas 353–367, 2018.

[9] Adeline Langlois y Damien Stehlé. Reducciones del peor de los casos al promedio para celosías de módulos.*Diseños, Códigos y Criptografía*, 75(3):565–599, 2015.

[10] Oded Regev. Sobre celosías, aprendizaje con errores, códigos lineales aleatorios y criptografía. En *Actas del trigésimo séptimo simposio anual ACM sobre teoría de la computación*, STOC '05, páginas 84–93, Nueva York, NY, EE. UU., 2005. Asociación de Maquinaria de Computación.

[11] Eiichiro Fujisaki y Tatsuaki Okamoto. Integración segura de esquemas de cifrado simétricos y asimétricos.*Revista de criptología*, 26:80–101, 2013.

[12] Dennis Hofheinz, Kathrin Hövelmanns y Eike Kiltz. Un análisis modular de la transformación Fujisaki-Okamoto. En Yael Kalai y Leonid Reyzin, editores, *Teoría de la criptografía*, páginas 341–371, Cham, 2017. Springer International Publishing.

[13] Jonathan Katz y Yehuda Lindell. *Introducción a la criptografía moderna*. Chapman & Hall/CRC, tercera edición, 2020.

[14] Lirio Chen. Recomendación para la derivación de claves utilizando funciones pseudoaleatorias. (Instituto Nacional de Estándares y Tecnología, Gaithersburg, MD), Publicación especial (SP) del NIST 800-108 Rev. 1, agosto de 2022. https://doi.org/10.6028/NIST.SP.800-108r1.

[15] Elaine B. Barker y John M. Kelsey. Recomendación para la generación de números aleatorios utilizando generadores deterministas de bits aleatorios. (Instituto Nacional de Estándares y Tecnología, Gaithersburg, MD), Publicación especial (SP) del NIST 800-90A, Rev. 1, junio de 2015. https://doi.org/10.6028/NIST.SP.800-90Ar1.

[16] Meltem Sönmez Turan, Elaine B. Barker, John M. Kelsey, Kerry A. McKay, Mary L. Baish y Mike Boyle. Recomendación para las fuentes de entropía utilizadas para la generación de bits aleatorios. (Instituto Nacional de Estándares y Tecnología, Gaithersburg, MD), Publicación especial (SP) del NIST 800-90B, enero de 2018. https://doi.org/10.6028/NIST.SP.800-90B.

[17] Elaine B. Barker, John M. Kelsey, Kerry McKay, Allen Roginsky y Meltem Sönmez Turan. Recomendación para construcciones de generadores de bits aleatorios (RBG). (Instituto Nacional de Estándares y Tecnología, Gaithersburg, MD), Publicación especial (SP) del NIST 800-90C (Tercer borrador público), septiembre de 2022. https://csrc.nist.gov/publications/detail/sp/800-90c/draft.

[18] Instituto Nacional de Normas y Tecnología. Requisitos de presentación y criterios de evaluación para el proceso de estandarización de la criptografía poscuántica, 2016. https://csrc.nist. gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposalsfnal-dec-2016.pdf.

[19] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, René Peralta, Ray Perlner, Angela Robinson y Daniel Smith-Tone. Informe de estado de la tercera ronda del proceso de estandarización de la criptografía poscuántica del NIST. Informe técnico NIST Interagencial o Informe interno (IR) 8413, Instituto Nacional de Estándares y Tecnología, Gaithersburg, MD, julio de 2022.

[20] Samuel Jaques, Michael Naehrig, Martin Roetteler y Fernando Virdia. Implementación de oráculos Grover para búsqueda de claves cuánticas en AES y LowMC. En Anne Canteaut y Yuval Ishai, editores, *Avances en Criptología – EUROCRYPT 2020*, páginas 280–310, Cham, 2020. Springer International Publishing.

[21] Lov K. Grover. Un algoritmo rápido de mecánica cuántica para la búsqueda de bases de datos. En *Actas del vigésimo octavo simposio anual ACM sobre teoría de la computación*, STOC '96, páginas 212–219, Nueva York, NY, EE. UU., 1996. Asociación de Maquinaria de Computación.

Apéndice A: Categorías de resistencia de la seguridad

El NIST comprende que existen importantes incertidumbres al estimar las fortalezas de seguridad de los criptosistemas poscuánticos. Estas incertidumbres provienen de dos fuentes: primero, la posibilidad de que se descubran nuevos algoritmos cuánticos, lo que conducirá a nuevos ataques criptoanalíticos; y segundo, nuestra capacidad limitada para predecir las características de rendimiento de las futuras computadoras cuánticas, como su costo, velocidad y tamaño de memoria.

Para abordar estas incertidumbres, NIST propuso el siguiente enfoque en su convocatoria de propuestas original [18]. En lugar de definir la fortaleza de un algoritmo utilizando estimaciones precisas del número de “bits de seguridad”, el NIST definió una colección de categorías amplias de fortaleza de seguridad. Cada categoría se define mediante una primitiva de referencia comparativamente fácil de analizar, cuya seguridad servirá como base para una amplia variedad de métricas que el NIST considera potencialmente relevantes para la seguridad práctica. Se puede crear una instancia de un criptosistema determinado utilizando diferentes conjuntos de parámetros para poder encajar en diferentes categorías. Los objetivos de esta clasificación son:

- Facilitar comparaciones significativas de rendimiento entre varios algoritmos poscuánticos garantizando, en la medida de lo posible, que los conjuntos de parámetros que se comparan proporcionen una seguridad comparable.
- Para permitir que NIST tome decisiones futuras prudentes sobre cuándo realizar la transición a claves más largas
- Ayudar a los remitentes a tomar decisiones consistentes y sensatas con respecto a qué primitivas simétricas usar en los mecanismos de relleno u otros componentes de sus esquemas que requieren criptografía simétrica.
- Comprender mejor las compensaciones entre seguridad y rendimiento involucradas en un enfoque de diseño determinado.

De acuerdo con el segundo y tercer objetivo anteriores, el NIST basó su clasificación en la gama de fortalezas de seguridad que ofrecen los estándares existentes del NIST en criptografía simétrica, que el NIST espera que ofrezca una resistencia significativa al criptoanálisis cuántico. En particular, NIST definió una categoría separada para cada uno de los siguientes requisitos de seguridad (enumerados en orden creciente):

1. Cualquier ataque que viole la definición de seguridad relevante debe requerir recursos computacionales comparables o mayores que los necesarios para la búsqueda de claves en un cifrado de bloque con una clave de 128 bits (por ejemplo, AES-128).
2. Cualquier ataque que viole la definición de seguridad relevante debe requerir recursos computacionales comparables o mayores que los necesarios para la búsqueda de colisiones en una función hash de 256 bits (por ejemplo, SHA-256/SHA3-256).
3. Cualquier ataque que viole la definición de seguridad relevante debe requerir recursos computacionales comparables o mayores que los necesarios para la búsqueda de claves en un cifrado de bloque con una clave de 192 bits (por ejemplo, AES-192).
4. Cualquier ataque que viole la definición de seguridad relevante debe requerir recursos computacionales comparables o mayores que los necesarios para la búsqueda de colisiones en una función hash de 384 bits (por ejemplo, SHA-384/SHA3-384).
5. Cualquier ataque que rompa la definición de seguridad relevante debe requerir recursos computacionales

comparables o superiores a los necesarios para la búsqueda de claves en un cifrado de bloque con una clave de 256 bits (por ejemplo, AES-256).

Tabla 4. Categorías de fortaleza de seguridad del NIST

Categoría de seguridad	Tipo de ataque correspondiente	Ejemplo
1	Búsqueda de claves en cifrado de bloques con clave de 128 bits	AES-128
2	Búsqueda de colisiones en función hash de 256 bits	SHA3-256
3	Búsqueda de claves en cifrado de bloques con clave de 192 bits	AES-192
4	Búsqueda de colisiones en función hash de 384 bits	SHA3-384
5	Búsqueda de claves en cifrado de bloques con clave de 256 bits	AES-256

Aquí, los recursos computacionales se pueden medir usando una variedad de métricas diferentes (por ejemplo, número de operaciones elementales clásicas, tamaño del circuito cuántico). Para que un criptosistema satisfaga uno de los requisitos de seguridad anteriores, cualquier ataque debe requerir recursos computacionales comparables o superiores al umbral establecido, con respecto a todas las métricas que el NIST considera potencialmente relevantes para la seguridad práctica.

El NIST pretende considerar una variedad de métricas posibles, que reflejan diferentes predicciones sobre el desarrollo futuro de la tecnología informática clásica y cuántica, y el costo de diferentes recursos informáticos (como el costo de acceder a cantidades extremadamente grandes de memoria).[2]El NIST también considerará las aportaciones de la comunidad criptográfica con respecto a esta cuestión.

En una métrica de ejemplo proporcionada a los remitentes, el NIST sugirió un enfoque en el que los ataques cuánticos se restringen a un tiempo de ejecución fijo o a una profundidad del circuito. Llame a este parámetro MAXDEPTH. Esta restricción está motivada por la dificultad de ejecutar cálculos en serie extremadamente largos. Los valores plausibles para MAXDEPTH oscilan entre 2^{40}puertas lógicas (el número aproximado de puertas que actualmente se espera que funcionen en serie las arquitecturas de computación cuántica en un año) hasta 2^{64} puertas lógicas (el número aproximado de puertas que las arquitecturas informáticas clásicas actuales pueden realizar en serie en una década), a no más de 2^{96}puertas lógicas (el número aproximado de puertas que los qubits de escala atómica con tiempos de propagación de la luz podrían realizar en un milenio). La versión más básica de esta métrica de costos ignora los costos asociados con los bits o qubits que se mueven físicamente para que estén físicamente lo suficientemente cerca como para realizar operaciones de puerta. Esta simplificación puede resultar en una subestimación del costo de implementar cálculos con uso intensivo de memoria en hardware real.

La complejidad de los ataques cuánticos puede medirse entonces en términos del tamaño del circuito. Estos números se pueden comparar con los recursos necesarios para romper AES y SHA-3. Durante el proceso de estandarización poscuántica, el NIST proporcionó las siguientes estimaciones para los recuentos de puertas clásicas y cuánticas[3]para la recuperación óptima de claves y ataques de colisión en AES y SHA-3, respectivamente, donde

[2]Vea la discusión en [19, Apéndice B].

[3]Los tamaños de los circuitos cuánticos se basan en el trabajo en [20].

1225la profundidad del circuito está limitada a MAXDEPTH][4].

Tabla 5. Estimaciones de recuentos de puertas clásicas y cuánticas para la recuperación óptima de claves y ataques de colisión en AES y SHA-3

AES-128	2^{157}/MAXDEPTH puertas cuánticas o 2^{143}puertas clásicas
SHA3-256	2^{146}puertas clásicas
AES-192	2^{221}/MAXDEPTH puertas cuánticas o 2^{207}puertas clásicas
SHA3-384	2^{210}puertas clásicas
AES-256	2^{285}/MAXDEPTH puertas cuánticas o 2^{272}puertas clásicas
SHA3-512	2^{274}puertas clásicas

1226 Vale la pena señalar que las categorías de seguridad basadas en estas primitivas de referencia proporcionan
1227 sustancialmente más seguridad cuántica de lo que podría sugerir un análisis ingenuo. Por ejemplo, las categorías
1228 1, 3 y 5 se definen en términos de cifrados en bloque, que pueden descifrarse utilizando el algoritmo de Grover [21
1229], con una aceleración cuántica cuadrática. Sin embargo, el algoritmo de Grover requiere un cálculo en serie de
1230 larga duración, lo cual es difícil de implementar en la práctica. En un ataque realista, hay que ejecutar muchas
1231 instancias más pequeñas del algoritmo en paralelo, lo que hace que la aceleración cuántica sea menos dramática.

1232 Finalmente, para los ataques que utilizan una combinación de computación clásica y cuántica, se puede utilizar una
1233 métrica de costos que califique las puertas cuánticas lógicas como varios órdenes de magnitud más caras que las
1234 puertas clásicas. Las arquitecturas de computación cuántica imaginadas actualmente suelen indicar que el costo
1235 por puerta cuántica podría ser miles de millones o billones de veces el costo por puerta clásica. Sin embargo,
1236 especialmente cuando se consideran algoritmos que afirman tener una alta seguridad (por ejemplo, equivalente a
1237 AES-256 o SHA-384), probablemente sea prudente considerar la posibilidad de que esta disparidad se reduzca
1238 significativamente o incluso se elimine.

[4] NIST cree que las estimaciones anteriores son precisas para la mayoría de los valores de MAXDEPTH que son relevantes para su análisis de seguridad, pero las estimaciones anteriores pueden subestimar la seguridad de SHA para valores muy pequeños de MAXDEPTH y pueden subestimar la seguridad cuántica de AES para valores muy grandes de MAXDEPTH.