Universidad
Rey Juan Carlos

TESIS DOCTORAL

Criptografía segura frente a adversarios cuánticos. Análisis y variantes de propuestas para estandarización

Autor:
José Ignacio Escribano Pablos

Directora:
María Isabel González Vasco

Programa de Doctorado en Ciencias

Escuela Internacional de Doctorado

2022

Universidad
Rey Juan Carlos

TESIS DOCTORAL

Criptografía segura frente a adversarios cuánticos. Análisis y variantes de propuestas para estandarización

Autor:
José Ignacio Escribano Pablos

Directora:
María Isabel González Vasco

Programa de Doctorado en Ciencias

Escuela Internacional de Doctorado

2022

Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica

Criptografía segura frente a adversarios cuánticos. Análisis y variantes de propuestas para estandarización

TESIS DOCTORAL

Autor: José Ignacio Escribano Pablos

Directora: María Isabel González Vasco

2022

Daría todo lo que sé por la mitad de lo que ignoro.

René Descartes (1596–1650)

Si he logrado ver más lejos, ha sido porque he subido a hombros de gigantes.

Isaac Newton (1643–1727)

Agradecimientos

A Maribel, por confiar en mí hace 4 años y por hacer que el proyecto de tesis se haya concretado en esta memoria.

A Alfonso Muñoz, por haberme introducido en el maravillo mundo de la criptografía.

A mis padres y a mi hermana, por estar siempre ahí, en los buenos y en los malos momentos.

A Ángel y Misael, por haberme enseñado tantas cosas del mundo académico que desconocía.

A Coke, Nerea, Miguel y Ruth, por haberme aguantado estos años en el trabajo.

A Adri, Edgar, Germán, Javi y Jesús por haberme acompañado durante este proyecto, aunque no hayamos pasado todo el tiempo juntos que hubiera querido.

A quien confió en mí en algún momento.

Gracias a todos vosotros.

Resumen

Los protocolos de intercambio de clave permiten establecer una comunicación segura entre dos partes a través de un canal inseguro. Estos se pueden generalizar a un número arbitrario de participantes dando lugar a los llamados protocolos de intercambio de clave en grupo.

Esta tesis se enmarca en el estudio e implementación de protocolos de intercambio de clave autenticados en grupo postcuánticos. En concreto, se construyen sobre los mecanismos de encapsulación de clave —conocidos como KEM por su siglas en inglés —finalistas del concurso de estandarización de algoritmos postcuánticos del Instituto de Estándares y Tecnología de los EE.UU. (NIST). Para entender bien la filosofía de este concurso se ha estudiado también el caso de una propuesta fallida: el esquema de firma digital WalnutDSA.

Como principal resultado, se construye un protocolo de intercambio de clave autenticado en grupo postcuántico basado en Kyber, uno de los KEM finalistas del concurso. Demostramos que es seguro en el *Quantum Random Oracle Model* (QROM), un modelo más fuerte que el típicamente considerado. Nuestro diseño se basa en el compilador de Abdalla et al. [1] y las transformaciones de Hövelmanns et al. [63, 64]. El compilador permite convertir un protocolo de intercambio autenticado de clave entre dos partes (abreviado como AKE) en un intercambio de clave en grupo. Las transformaciones de [63, 64] permiten obtener las primitivas criptográficas seguras en el QROM, necesarias para el compilador. En concreto: un AKE y un esquema de compromiso que se derivan a partir de un esquema de cifrado de clave pública. A raíz de la estructura de esta propuesta, diseñamos protocolos para cada uno de los KEM finalistas: Classic McEliece, Kyber, NTRU y Saber. Los protocolos se implementan y se analiza cuál de ellos

ofrece un mejor rendimiento.

Objetivos

Los objetivos de la tesis son los siguientes:

- Diseñar un esquema genérico de intercambio de clave autenticado en grupo postcuántico, que pueda demostrarse seguro formalmente frente a adversarios cuánticos. Identificar las herramientas involucradas en dicho esquema para que pueda completarse la demostración de seguridad.
- Construir, a partir del diseño genérico, nuevos protocolos de intercambio de clave autenticados en grupo postcuántico, usando los KEM finalistas del concurso de estandarización del NIST.
- Implementar estos protocolos para comprobar la viabilidad de los mismos en aplicaciones reales y analizar comparativamente el rendimiento sobre los diferentes KEM finalistas del NIST.

Metodología

La metodología seguida en esta tesis se puede resumir en los siguientes puntos:

- Análisis continuo del estado del arte de la criptografía postcuántica.
- Seguimiento de los avances del concurso de estandarización del NIST.
- Estudio teórico asociado a nuestro diseño de protocolo de intercambio de clave autenticado en grupo.
- Implementación de los diseños propuestos y estudio comparativo de rendimiento de los mismos.

Resultados

Hemos diseñado un protocolo de autenticado de intercambio de clave en grupo postcuántico basado en Kyber, uno de los KEM finalistas del concurso de estandarización del NIST. Este protocolo se apoya en el compilador de Abdalla et al. [1] y las transformaciones de Hövelmanns et al. [63, 64]. El compilador

transforma un protocolo de intercambio de clave entre dos partes (AKE) en un protocolo de intercambio de clave autenticado en grupo. Las transformaciones de [63, 64] permiten conseguir las primitivas necesarias requeridas por el compilador, seguras en el *Quantum Random Oracle Model* (QROM): un AKE y un esquema de compromiso. El primero se obtiene de un esquema de cifrado de clave pública (PKE, por sus siglas en inglés) IND-CPA. El esquema de compromiso se puede construir a partir de cualquier PKE IND-CCA, que se consigue a partir de transformar un PKE IND-CPA en un KEM IND-CCA y este último en un PKE IND-CCA. Demostramos que es posible aplicar las transformaciones de [63, 64] a Kyber, un resultado que no estaba formalmente probado. Además, demostramos que el protocolo es seguro en el QROM, un modelo de seguridad más fuerte que el típicamente considerado, que capta las habilidades de un adversario cuántico. Hasta donde sabemos, este es el primer protocolo de este tipo que es seguro en este modelo de seguridad y que se ha diseñado sobre uno de los finalistas del concurso del NIST. Todo lo anterior se desarrolla en el Capítulo 3.

Este protocolo se implementa sobre el código de Kyber que han hecho público sus autores, realizando todos los cambios necesarios para adecuarse al diseño propuesto. En concreto, los cambios se centran en la implementación de las transformaciones de Hövelmanns et al. [63, 64] a Kyber y en el compilador de Abdalla et al. [1]. Además, se implementan los algoritmos homólogos del protocolo, propuestos en [22], pero seguros en el *Random Oracle Model* (ROM), un modelo que no capta las habilidades de un adversario cuántico. Mostramos experimentalmente que, en este escenario, los algoritmos seguros en el QROM, son más eficientes, que es un resultado contraintuitivo, ya que, a priori, llevaría a pensar que un modelo de seguridad más fuerte implica también una pérdida de rendimiento. Los detalles se pueden encontrar en el Capítulo 4.

Por último, este protocolo se extiende al resto de KEM finalistas del concurso del NIST. En este caso, no es de aplicación el trabajo de Hövelmanns et al. [63, 64] debido a que los KEM finalistas no cumplen las hipótesis de las transformaciones. Por ello, aplicamos la transformación de Fujioka et al. [46], que convierte un KEM en un AKE seguro en el ROM. Por tanto, el protocolo de intercambio de clave en grupo obtenido es seguro en el ROM. Implementamos el protocolo sobre la librería *open source* LibOQS [104], que incluye a todos los finalistas del concurso. Comparamos el rendimiento de los 4 KEM finalistas del concurso: Classic McEliece, Kyber, NTRU y Saber. Se muestra una clara diferencia de rendimiento entre los esquemas basados en retículos (Kyber, NTRU y Saber) y los basados en códigos (Classic McEliece). Este último esquema no es adecuado para este protocolo debido a que es notablemente más lento, debido a que tiene claves públicas significativamente mayores que sus homólogos basados en códigos. En términos de escalabilidad, la mejor opción es Kyber512 en el nivel 1 que

define el NIST (menor nivel de seguridad); Kyber768 y NTRU-HRSS-701 en el nivel 3 (nivel medio) y, Kyber1024 en el nivel 5 (mayor nivel de seguridad). Por último, se compara el rendimiento del protocolo obtenido a partir de los trabajos de Hövelmanns et al. [63, 64] y Fujioka et al. [46] sobre Kyber, resultando que el primero ofrece un mayor rendimiento, además de ser seguro en un modelo de seguridad más fuerte. Los detalles se pueden encontrar en el Capítulo 5.

Publicaciones

Revista

- José Ignacio Escribano Pablos, María Isabel González Vasco, Misael Enrique Marriaga, y Ángel Luis Pérez del Pozo. "The Cracking of WalnutDSA: A Survey" Symmetry 2019, 11, 1072. `https://doi.org/10.3390/sym11091072`.

  JCR 2020: 2.713, Multidisciplinary Sciences, 33/72, Q2.

  Este trabajo recopila los ataques de un esquema postcuántico de firma digital llamado WalnutDSA. Los ataques se describen en la Sección 2.3.2.1.

- José Ignacio Escribano Pablos, María Isabel González Vasco, Misael Enrique Marriaga, y Ángel Luis Pérez del Pozo. "Compiled Constructions towards Post-Quantum Group Key Exchange: A Design from Kyber". Mathematics 2020, 8, 1853. `https://doi.org/10.3390/math8101853`.

  JCR 2020: 2.258, Mathematics, 24/330, Q1.

  Este trabajo contiene los resultados del Capítulo 3.

- José Ignacio Escribano Pablos y María Isabel González Vasco. "Secure Post-Quantum Group Key Exchange: Implementing a Solution Based on Kyber". Enviado para publicación a IET Communications (en revisión).

  JCR 2020: 1.542, Engineering, Electrical & Electronic, 202/273, Q3.

  Este trabajo incluye los resultados del Capítulo 4.

- José Ignacio Escribano Pablos, Misael Enrique Marriaga, y Ángel Luis Pérez del Pozo. "Design and Implementation of a Post-Quantum Group Authenticated Key Exchange protocol with the LibOQS library: a comparative performance analysis from Classic McEliece, Kyber, NTRU, and Saber". Enviado para publicación a IEEE Access (en revisión).

  JCR 2020: 3.367, Computer Science, Information Systems, 65/161, Q2.

  En este artículo, se recogen los resultados del Capítulo 5.

Artículos en actas de congresos

- José Ignacio Escribano Pablos. "Implementación de un protocolo postcuántico de intercambio de clave en grupo seguro en el Quantum Random Oracle Model basado en Kyber". En: Investigación en Ciberseguridad. Ediciones de la Universidad de Castilla-La Mancha, 2021. https://doi.org/10.18239/jornadas_2021.34.56.

  Este artículo contiene los resultados preliminares del Capítulo 4.

Comunicaciones orales en congresos

- José Ignacio Escribano Pablos. "On secure post-quantum group key exchange". V Congreso de Jóvenes Investigadores de la Real Sociedad Matemática Española. 2020.

  En esta comunicación, se presenta un análisis preliminar del diseño teórico del protocolo del Capítulo 3.

- José Ignacio Escribano Pablos. "Implementaciones de criptografía postcuántica". I Encuentro de Tecnologías Cuánticas (QTEC) organizado por el Centro Criptológico Nacional. 2022.

  En esta presentación, se analiza el estado del arte de las implementaciones postcuánticas de la Sección 2.3.3.

Índice general

Introducción

La computación cuántica supone una revolución con dos caras. Por un lado, aborda problemas que no son resolubles eficientemente por ordenadores clásicos, pero, por otro lado, pone en jaque la seguridad de los sistemas criptográficos tal y como los conocemos. Dos de los algoritmos cuánticos con más impacto son los algoritmos de Grover y Shor. El algoritmo de Grover permite reducir el tiempo de búsqueda en un conjunto de elementos no ordenados, haciendo que encontrar una colisión en una función *hash* o una clave de cifrado sea más sencillo. No obstante, la mitigación de este algoritmo es simple: basta con duplicar el tamaño de las claves de cifrado y la salida de las funciones *hash*. El algoritmo de Shor es otro algoritmo cuántico bien conocido. Este sí supone una amenaza real para la criptografía asimétrica actual: permite resolver problemas de forma eficiente tales como la factorización de números grandes o el problema del logaritmo discreto. Sobre estos problemas radica la seguridad de muchos de los criptosistemas de clave pública empleados en el mundo real. Estos sistemas criptográficos se usan para conectarse al banco y realizar un pago, realizar una llamada, enviar mensajes a través de aplicaciones de mensajería o firmar un documento digitalmente, entre muchos otros contextos. Con un ordenador cuántico suficientemente potente, la seguridad de estas herramientas quedará inevitablemente vulnerada. Es por ello, que se deben diseñar e implementar nuevos criptosistemas de clave pública que sean resistentes frente a ataques cuánticos.

En este contexto, surge la *criptografía postcuántica*, que propone construir nuevos criptosistemas que sean seguros frente a adversarios cuánticos. Estos nuevos esquemas son clásicos y se pueden ejecutar en cualquier ordenador, pero se basan en problemas que, se cree, no pueden ser resueltos de forma eficiente por

ordenadores cuánticos ni clásicos. Sin embargo, todo no son buenas noticias: la mayoría de los algoritmos postcuánticos tienen tamaños de clave mayores y son más lentos que sus homólogos tradicionales.

Desde 2017, el Instituto de Tecnología y Estándares de los EE.UU., conocido como NIST, está realizando un proceso de selección y estandarización de esquemas criptográficos postcuánticos, que sustituirán a los esquemas de clave pública actuales, si así fuera necesario. Este proceso busca dos tipos de esquemas: un mecanismo de encapsulación de clave (KEM, por sus siglas en inglés) y un esquema de firma digital. A día de hoy, el proceso ya ha dado a conocer a los finalistas y los candidatos alternativos. Estos últimos son otras opciones para que, en caso de encontrar vulnerabilidades insalvables en los finalistas, haya otros esquemas disponibles.

El tema principal de esta tesis es el diseño e implementación de protocolos de intercambio de clave autenticados en grupo (conocidos como protocolos GAKE) postcuánticos. Este tipo de protocolos permiten a un grupo de usuarios acordar entre ellos una clave de sesión común, que les permite comunicarse entre ellos a través de un canal inseguro. Una vez acordada la clave de sesión, pueden comunicarse entre ellos con un cifrado de clave simétrica como AES. Estos protocolos tienen muchas aplicaciones en el mundo real tales como mensajería instantánea, videollamadas grupales, o las aplicaciones colaborativas, entre muchas otras. En general, se puede emplear un protocolo GAKE en cualquier escenario que involucre a un grupo de usuarios que necesiten garantizar la confidencialidad de sus comunicaciones.

En resumen, esta tesis trata de unir el mundo de los protocolos GAKE con el mundo postcuántico. Para ello, se hará uso de los KEM finalistas del concurso de estandarización del NIST.

En el Capítulo 1, se introducen los conceptos básicos de criptografía, tanto simétrica como asimétrica, así como las nociones de seguridad y las primitivas criptográficas que se utilizarán en capítulos posteriores.

En el Capítulo 2, se presenta la criptografía postcuántica y se justifica su necesidad con el advenimiento de la computación cuántica. Asimismo, se describen cada una de las áreas de las matemáticas de la criptografía postcuántica. En concreto, la criptografía basada en retículos, códigos, *hashes*, isogenias y multivariante. Además, se desarrolla en profundidad el funcionamiento del concurso de estandarización del NIST: proceso, niveles de seguridad establecidos en el mismo, así como los finalistas. La mayor parte del capítulo se dedica a los KEM, que serán de utilidad para las construcciones de los Capítulos 3 y 5. En concreto se explica el funcionamiento y las propiedades de seguridad de Kyber, que es la pieza central del protocolo GAKE del Capítulo 3, así como el resto de KEM finalistas: Classic McEliece, NTRU y Saber. Estos tres últimos serán de interés en

la propuesta del Capítulo 5. Para acabar con los finalistas del concurso del NIST, presentamos los esquemas de firma digital, haciendo una revisión a una propuesta concreta: WalnutDSA, que basa su seguridad en un problema asociado a grupos de trenzas. Por último, presentamos las implementaciones y las librerías *open source*, que se encuentran disponibles y contienen código asociado a estos algoritmos postcuánticos.

El Capítulo 3 es el núcleo de la tesis y del que se derivan los capítulos posteriores. En él, se propone un protocolo GAKE postcuántico basado en Kyber. Nuestra construcción se basa en el compilador de Abdalla et al. [1]. Este permite transformar un protocolo de intercambio autenticado de clave 2-parte (AKE) en un protocolo GAKE. Requiere de dos piezas: un AKE y un esquema de compromiso. Para obtener ambas primitivas se hace uso del trabajo de Hövelmanns et al. [63] en el que propone varias transformaciones que son seguras en el *Quantum Random Oracle Model* (QROM), un modelo de seguridad más fuerte que el típico considerado frente a adversarios clásicos, el *Random Oracle Model* (ROM). En concreto, empleamos dos transformaciones: $\mathsf{FO}_{\mathsf{AKE}}$ y $\mathsf{FO}^{\not\perp}_{m}$. La primera transforma un esquema de cifrado de clave pública (PKE) IND-CPA en un AKE seguro; mientras que la segunda, transforma un PKE IND-CPA en un KEM IND-CCA. Sin embargo, la aplicación de estas técnicas no es directa y el PKE de base debe cumplir la propiedad de *Disjoint Simulatability* (Definición 3.1), abreviada como DS. Demostramos que, el esquema PKE IND-CPA del que proviene Kyber es DS y, por tanto, se pueden aplicar las transformaciones. Esto da lugar a un AKE 2-parte y a un KEM IND-CCA llamado Kyber$^{\not\perp}$. Este último, lo convertimos a un PKE IND-CCA usando el esquema híbrido KEM/DEM [32], que nos permite obtener un esquema de compromiso tal y como se apunta en [1]. Finalmente, demostramos que el protocolo GAKE es seguro en el QROM, probando, en primer lugar, su seguridad en el ROM tal y como se hace en [63].

En el Capítulo 4, implementamos el protocolo GAKE anterior sobre el código de Kyber, que está disponible en un repositorio de GitHub. Nuestra implementación también es pública y se encuentra disponible en `https://github.com/jiep/kyber-gake`. Comparamos el rendimiento de cada uno de los algoritmos que componen el protocolo y, del GAKE en sí mismo, con sus homólogos seguros en el ROM propuestos en [22]. Para ello, desarrollamos un *workflow* de GitHub Actions, que permite automatizar las pruebas en un entorno aislado. Mostramos experimentalmente que, en este protocolo GAKE, el modelo de seguridad QROM no introduce una pérdida de rendimiento, sino que, al contrario, las implementaciones en este modelo son más eficientes.

En el Capítulo 5, extendemos el protocolo GAKE más allá de Kyber para hacerlo aplicable al resto de KEM finalistas del concurso de estandarización del NIST: Classic McEliece, NTRU y Saber, siendo todos ellos, KEM IND-CCA. Ob-

tener el esquema de compromiso a partir de un KEM es sencillo. Sin embargo, no es posible aplicar la transformación $\mathsf{FO}_{\mathsf{AKE}}$ sobre los esquemas PKE originales de estos KEM, debido a que no está demostrado explícitamente que cumplan la propiedad DS. Además, en algunos PKE, incluso no se cumple la noción IND-CPA, que también es necesaria en la transformación. En sustitución, aplicamos la transformación FSXY, que es una transformación genérica aplicable a todos los KEM finalistas. De nuevo, implementamos el protocolo GAKE empleando esta transformación para los 4 KEM finalistas, publicada en `https://github.com/jiep/pq-gake-fsxy`. La implementación se construye sobre la librería *open source* LibOQS [104], que contiene el código de todos los KEM finalistas del concurso del NIST. Demostramos experimentalmente que, existen diferencias significativas en rendimiento entre los basados en retículos (Kyber, NTRU y Saber) y códigos (Classic McEliece). De hecho, este último no es adecuado para este protocolo debido a su bajo rendimiento. Por último, comparamos las transformaciones $\mathsf{FO}_{\mathsf{AKE}}$ y FSXY sobre Kyber, resultando que esta primera logra un mejor rendimiento que la segunda.

En el Capítulo 6, se comentan los resultados de los capítulos anteriores; y en el Capítulo 7, se desarrollan las principales conclusiones obtenidas en esta tesis.

Capítulo

1

Criptografía tradicional

En este capítulo, presentamos los conceptos básicos de criptografía y las primitivas y nociones de seguridad que serán de interés en capítulos posteriores. En concreto, este capítulo se divide en las áreas clásicas de la criptografía: simétrica, asimétrica e híbrida. En la primera, se introducen los esquemas de cifrado, las funciones *hash* y las *extendable output functions*. La segunda incluye los esquemas de cifrado, los mecanismos de encapsulado de clave, los esquemas de firma digital, así como los protocolos de intercambio de clave, tanto entre dos participantes como asociados a un número arbitrario de ellos. Por último, se presenta un esquema híbrido que combina un esquema de cifrado simétrico y un mecanismo de encapsulación para obtener un esquema de cifrado asimétrico.

1.1. Criptografía simétrica

La criptografía simétrica es la rama de la criptografía que presupone que todas las entidades involucradas en un proceso tienen acceso un bloque de información, llamado clave simétrica, impredecible para los adversarios potenciales. Esta debe ser compartida de antemano para que las partes puedas comunicarse

de forma segura.

Dentro de esta rama, tradicionalmente, se incluyen también las funciones *hash*, aunque no requieran de claves para funcionar.

1.1.1. Esquemas de cifrado

Los esquemas de cifrado simétricos se dividen en dos tipos: cifrados de flujo y de bloque. Los primeros permiten cifrar los mensajes bit a bit. Aquí se incluyen esquemas como A5/1, RC4 o Chacha20, entre otros. Los segundos cifran los mensajes en bloques de un tamaño prefijado e incluyen esquemas como DES o AES, entre otros.

1.1.1.1. Cifrado en bloque

Los esquemas de cifrado en bloque se pueden diseñar, típicamente, con una construcción llamada red de Feistel, cuyo ejemplo más conocido es DES [36], o usando redes de permutación-sustitución, como es el caso de AES [3].

Formalmente, un esquema de cifrado en bloque está compuesto por dos algoritmos deterministas: un algoritmo de cifrado E y algoritmo de descifrado D. El algoritmo E toma como entrada un mensaje m y una clave k, devolviendo un texto cifrado c. El algoritmo D toma un texto cifrado c y una clave k, devolviendo un mensaje m o $\perp$, que indica un error de descifrado. Se debe cumplir que $D(E(m,k),k) = m$.

Normalmente, los mensajes que se cifran son más grandes que el tamaño de un bloque, por lo que se deben procesar de algún modo varios bloques para generar el texto cifrado. Se debe hacer de tal manera que no se filtre información sobre el mensaje original. Para ello, se usan los llamados modos de operación. Estos permiten formar un texto cifrado que ocupa más de un bloque, sin filtrar información del mensaje. Los modos de operación más conocidos son CBC y CTR, entre otros. Estos modos logran aportar confidencialidad. Otros modos más avanzados como GCM [44] o CCM [43] aportan integridad además de la ya nombrada confidencialidad. Estos devuelven, además del texto cifrado, un *tag* o etiqueta, que garantiza que el texto cifrado no ha sido modificado antes de descifrarse. Muchos de los modos de operación dependen de una secuencia de bits llamada vector de inicialización (IV, por sus siglas en inglés) que no debe ser repetido y, en algunos modos, es aleatorio.

1.1.1.1.1. AES El Advanced Encryption Standard (AES) es el estándar de cifrado en bloque señalado por el NIST [3]. Admite tres tamaños de clave: 128, 192 y 256 bits. En todos los casos, el tamaño de bloque es de 128 bits.

1.1.2. Funciones hash

Una función *hash* es una función determinista que toma como entrada un número arbitrario y devuelve una salida de longitud fija, llamada *hash* o resumen. Formalmente, es una función $h : \{0,1\}^* \rightarrow \{0,1\}^n$, donde n es el tamaño de salida del *hash*.

Una función *hash* criptográfica debe cumplir las siguientes propiedades:

- Resistencia a la preimagen: dado un *hash* y, debe ser difícil encontrar un mensaje m tal que $y = h(m)$.
- Resistencia a colisiones: debe ser difícil encontrar colisiones, es decir, encontrar dos mensajes distintos m, m' tal que $h(m) = h(m')$.
- Resistencia a segunda preimagen: dado un mensaje m, debería ser difícil encontrar otro mensaje m' con el mismo *hash*, es decir, tal que $h(m) = h(m')$.

Ejemplos típicos de funciones *hash* son SHA-2 y SHA-3, estandarizadas por el NIST.

1.1.2.1. SHA-2 y SHA-3

SHA-2 [33] es un conjunto de 6 funciones *hash*, que se definen en función del tamaño de salida. Las más utilizadas son SHA-256 (*hash* de 256 bits) y SHA512 (*hash* de 512 bits).

SHA-3 [42] es un subconjunto de la solución llamada Keccak [16], que incluye *hashes* de 224, 256, 384 y 512 bits, llamadas SHA3-224, SHA3-256, SHA3-384 y SHA3-512, respectivamente.

1.1.2.2. Random Oracle Model

El *Random Oracle Model* (ROM) es un modelo heurístico empleado para construir demostraciones de seguridad, idealizando las funciones *hash*. En el ROM, todas las partes tienen acceso a un oráculo aleatorio $\mathcal{O}$. Cualquier parte puede realizar una consulta $x \in \{0,1\}^*$ al oráculo $\mathcal{O}$, que devolverá $\mathcal{O}(x) \in \{0,1\}^*$, siendo cada bit de la respuesta elegido independiente y uniformemente al azar. Si x ya ha sido consultado con anterioridad, el oráculo aleatorio devolverá el mismo resultado que en la consulta previa. Este comportamiento del oráculo aleatorio, se puede ver como el comportamiento idílico de una función *hash*. Es por ello que, en los protocolos que se demuestran seguros en el ROM, se reemplaza el oráculo aleatorio por una función *hash*.

Las demostraciones de seguridad que no hacen uso de oráculos aleatorios, se dice que están en el modelo estándar.

1.1.3. Extendable Output Function

Una eXtendable Output Function, abreviada como XOF, es una función en la que se puede extender su salida de forma variable a la longitud deseada [42]. Se pueden ver como una generalización de las funciones *hash*, que tienen una longitud fija de salida.

SHA-3 [42] ofrece funciones XOF denominadas SHAKE128 y SHAKE256, con niveles de seguridad de 128 y 256 bits, respectivamente.

1.2. Criptografía asimétrica

La criptografía asimétrica (también llamada de clave pública), en contraposición a la criptografía simétrica, emplea un par de claves (llamadas pública y secreta[1]) para definir los distintos roles (emisor/receptor, firmante/verificador, etc.) de las entidades involucradas en un proceso.

Esta se basa en problemas computacionales que son eficientes de realizar en un sentido, pero que no es factible realizarlos en sentido contrario. Un ejemplo sencillo es RSA [103], que basa su seguridad en que no existe algoritmo (clásico) eficiente para factorizar números grandes. La clave secreta es un par de números primos grandes y la clave pública es el producto de ellos. Obviamente, multiplicarlos es una operación muy sencilla de realizar, aunque el tamaño de los números sean muy grandes. Sin embargo, realizar la factorización de este producto se vuelve inviable al no tener potencia de cálculo suficiente para realizarla, y llevaría cientos o miles de años hasta completarse con números suficientemente grandes.

A continuación describimos las primitivas asimétricas que serán de utilidad en capítulos posteriores. Si en lo sucesivo no hacemos explícitos los conjuntos de mensajes, textos cifrados y claves involucrados, asumimos que estos elementos son siempre cadenas binarias de longitud polinomial en el parámetro de seguridad.

1.2.1. Esquemas de cifrado

A continuación, definimos formalmente los esquemas de cifrado de clave pública (o PKE, por sus siglas en inglés) y sus propiedades de seguridad.

Definición 1.1 ([72]; definición 11.1)**.** *Un esquema de cifrado de clave pública o PKE es una tripleta de algoritmos* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, donde:*

[1] También recibe el nombre de clave privada.

- *KeyGen, el algoritmo de generación de claves, es un algoritmo probabilístico que toma como entrada el parámetro de seguridad $\mathbf{1}^\lambda$ y devuelve un par de claves público/privada (pk, sk).*

  Escribiremos esto como $(pk, sk) \leftarrow \mathsf{KeyGen}()$, cuando el parámetro de seguridad esté claro y no induzca a confusión.

- *Enc, el algoritmo de cifrado, que toma como entrada una clave pública pk y un mensaje m de un espacio de mensajes $\mathcal{M}$ y, devuelve un texto cifrado c.*

  Escribiremos esto como $c \leftarrow \mathsf{Enc}(pk, m)$.

- *Dec, el algoritmo de descifrado, es un algoritmo determinista que toma como entrada una clave privada sk y un texto cifrado c y, devuelve un mensaje $m \in \mathcal{M}$ o un símbolo especial $\perp$, que denota un error de descifrado.*

  Escribiremos esto como $m := \mathsf{Dec}(sk, c)$.

Se requiere que para cada λ, cada (pk, sk) devuelto por $\mathsf{KeyGen}(\mathbf{1}^\lambda)$ y cada mensaje $m \in \mathcal{M}$, se cumpla

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m.$$

1.2.1.1. Definiciones de seguridad

Antes de introducir las definiciones formales de seguridad, listaremos los diferentes objetivos de los adversarios y los modelos de ataque, que intentan capturar las principales estrategias asociadas a cada adversario específico. Denotaremos con $\mathcal{A}$ a un adversario probabilístico en tiempo polinomial. Asumiremos que $\mathcal{A}$ persigue uno de los siguientes objetivos:

- One-Wayness (OW): invertir la unidireccionalidad de la función de cifrado sin conocer la clave secreta.
- Indistinguishability (IND): conocer cualquier información sobre un texto plano a partir de un texto cifrado.
- Non-Malleability (NM): obtener un texto cifrado válido relacionado a partir de otro. Captura la idea de que los textos cifrados deben ser a prueba de modificaciones.

De forma similar, distinguimos los siguientes modelos de ataque, que capturan las capacidades del adverario:

- CPA (Chosen-Plaintext Attack): el adversario $\mathcal{A}$ puede obtener textos cifrados a partir de textos planos de su elección.

- CCA1 (Non-Adaptive Chosen-Ciphertext Attack): el adversario $\mathcal{A}$ tiene acceso a un oráculo de descifrado, que descifra textos cifrados elegidos por $\mathcal{A}$. Este ataque es no adaptativo, es decir, $\mathcal{A}$ no puede adaptar sus textos cifrados en función de los resultados previamente obtenidos.
- CCA2 (Adaptive Chosen-Ciphertext Attack): el adversario $\mathcal{A}$ tiene acceso a un oráculo de descifrado y puede hacer consultas adaptativas. Lo único que no se le permite es que llame al oráculo con el texto cifrado del desafío. Normalmente, a CCA2 se le conoce simplemente como CCA. A lo largo de la tesis nos referiremos a CCA2 como CCA.

Definición 1.2 ([120]). *Un esquema de cifrado público* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice que es OW-CPA (One-Wayness under Chosen-Plaintext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A}$*, se tiene:*

$$\Pr\left[m = m' : \begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ m \xleftarrow{\$} \mathcal{M};\\ c \leftarrow \mathsf{Enc}(pk, m);\\ m' \leftarrow \mathcal{A}(pk, c);\end{array}\right] \leq \text{negl}(\lambda).$$

La definición anterior establece que, dada la clave pública y un texto cifrado de un mensaje seleccionado uniformemente al azar, un adversario debería seguir teniendo una probabilidad despreciable de obtener el mensaje original.

Definición 1.3 ([120]). *Un esquema de cifrado público* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice que es OW-CCA (One-Wayness under Chosen Ciphertext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, con acceso polinomial a un oráculo de descifrado* $\mathcal{O}_{sk}$ *que produce descifrado válidos con respecto a cierta clave privada* sk*, se tiene:*

$$\Pr\left[m = m' : \begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ st \leftarrow \mathcal{A}_1^{\mathcal{O}_{sk}}(pk);\\ m \xleftarrow{\$} \mathcal{M};\\ c \leftarrow \mathsf{Enc}(pk, m);\\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{sk}}(st, c);\end{array}\right] \leq \text{negl}(\lambda).$$

La definición anterior establece que, un adversario debería tener una probabilidad despreciable de invertir la unidireccionalidad de la función de cifrado, pese a tener acceso a un oráculo de descifrado.

Definición 1.4 ([14]). *Un esquema de cifrado público* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice que es IND-CPA (INDistinguishable under Chosen-Plaintext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, se tiene:*

$$\left| \Pr \left[b = b' : \begin{array}{l} (pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda); \\ (m_0 \in \mathcal{M}, m_1 \in \mathcal{M}, s) \leftarrow \mathcal{A}_1(pk); \\ b \xleftarrow{\$} \{0,1\}; \\ c \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathcal{A}_2(m_0, m_1, s, c); \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

La definición anterior establece que, un adversario debería tener una probabilidad despreciable de distinguir entre dos textos cifrados (excluyendo la probabilidad de acertar por azar), a pesar de conocer los mensajes originales.

Definición 1.5 ([14]). *Un esquema de cifrado público* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice que es IND-CCA (INDistinguishable under Chosen Ciphertext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, con acceso polinomial a un oráculo de descifrado* $\mathcal{O}_{sk}$ *que produce descifrado válidos con respecto a cierta clave privada* sk *se tiene:*

$$\left| \Pr \left[b = b' : \begin{array}{l} (pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda); \\ (m_0 \in \mathcal{M}, m_1 \in \mathcal{M}, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_{sk}}(pk); \\ b \xleftarrow{\$} \{0,1\}; \\ c \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{sk}}(m_0, m_1, s, c); \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

La definición anterior establece que, un adversario debería tener una probabilidad despreciablemente superior a $1/2$ de distinguir entre dos textos cifrados, a pesar de tener acceso a un oráculo de descifrado.

La noción de seguridad IND-CCA es la más fuerte y la estándar entre los esquemas de cifrado. La noción IND-CCA implica también cumplir IND-CPA y esta, a su vez, implica OW-CCA y esta última implica la noción OW-CPA. Asimismo, la noción IND-CCA también implica NM-CCA.

1.2.2. Mecanismos de encapsulado de clave

En esta sección, definimos formalmente los mecanismos de encapsulado de clave o KEM (por sus siglas en inglés) y las nociones de seguridad fundamentales asociadas. Los KEM son la pieza básica de la construcción del Capítulo 3.

Definición 1.6 ([38]). *Un mecanismo de encapsulado de claves o KEM es una tripleta de algoritmos polinomiales* $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$*, junto con un espacio de claves* $\mathcal{K}$*, donde:*

- KeyGen*, el algoritmo de generación de claves, es un algoritmo probabilístico que toma como entrada el parámetro de seguridad* $\mathbf{1}^\lambda$ *y devuelve un par de claves público/privada* (pk, sk)*.*

  Escribiremos esto como $(pk, sk) \leftarrow \mathsf{KeyGen}()$*, cuando el parámetro de seguridad esté claro y no induzca a confusión.*

- Encaps*, el algoritmo de encapsulado, es un algoritmo probabilístico que toma una clave pública* pk *y devuelve una clave* $K \in \mathcal{K}$ *y un texto cifrado* C*. Se dice que* C *es una encapsulación de la clave* K*.*

  Escribiremos esto como $(K, C) \leftarrow \mathsf{Encaps}(pk)$*.*

- Decaps*, el algoritmo de desencapsulado, es un algoritmo determinista que toma como entrada una clave privada* sk *y la encapsulación de una clave* C *y, devuelve una clave* K *o* $\perp$ *si se produce un error.*

  Escribiremos esto como $K := \mathsf{Decaps}(sk, C)$*.*

1.2.2.1. Definiciones de seguridad

En el caso de los KEM, se definen las capacidades del adversario y los modelos de ataque de igual forma que en el caso de los esquemas de cifrado.

Definición 1.7 ([46]). *Un KEM=*$(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *y parámetro de seguridad* λ *se dice que es OW-CPA (One-Wayness under Chosen-Plaintext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A}$*, se tiene:*

$$\Pr\left[K = K' : \begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ (K, C) \leftarrow \mathsf{Encaps}(pk);\\ K' \leftarrow \mathcal{A}^{\mathcal{O}_{sk}}(pk, C);\end{array}\right] \leq \mathrm{negl}(\lambda).$$

La definición establece que, un adversario debería tener una probabilidad despreciable de obtener la clave a partir del texto cifrado.

Definición 1.8 ([46]). *Un KEM=*$(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *y parámetro de seguridad* λ *se dice que es OW-CCA (One-Wayness under Chosen-Ciphertext Attack) si para cualquier adversario polinomial probabilístico* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, con acceso a un oráculo de desencapsulación* $\mathcal{O}_{sk}$ *que produce desencapsulaciones válidas con respecto a cierta clave privada* sk*, se tiene:*

$$\Pr\left[K = K' : \begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ st \leftarrow \mathcal{A}_1^{\mathcal{O}_{sk}}(pk);\\ (K, C) \leftarrow \mathsf{Encaps}(pk);\\ K' \leftarrow \mathcal{A}_2^{\mathcal{O}_{sk}}(pk, C, st);\end{array}\right] \leq \text{negl}(\lambda).$$

La definición establece que, un adversario debería tener una probabilidad despreciable de obtener la clave a partir del texto cifrado, pese a contar con un oráculo de desencapsulación.

Definición 1.9 ([22]). *Un KEM=($\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}$) con espacio de claves $\mathcal{K}$ y parámetro de seguridad λ se dice que es IND-CCA (INDistinguishable under Chosen-Ciphertext Attack) si para cualquier adversario polinomial probabilístico $\mathcal{A}$ con acceso a un oráculo de desencapsulación $\mathcal{O}_{sk}$ que produce desencapsulaciones válidas con respecto a cierta clave privada sk, se tiene:*

$$\left|\Pr\left[b = b' : \begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ b \xleftarrow{\$} \{0,1\};\\ (K_0, C) \leftarrow \mathsf{Encaps}(pk);\\ K_1 \xleftarrow{\$} \mathcal{K};\\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{sk}}(pk, C, K_b);\end{array}\right] - \frac{1}{2}\right| \leq \text{negl}(\lambda).$$

La definición establece que, un adversario debería tener una probabilidad despreciable de distinguir entre dos claves a partir del texto cifrado, pese a contar con un oráculo de desencapsulación (excluyendo la probabilidad de que acierte al azar).

1.2.3. Esquemas de firma digital

La firma digital es una operación fundamental que se puede realizar con criptografía asimétrica. A continuación, definimos formalmente los esquemas de firma y sus propiedades de seguridad.

Definición 1.10. *Un esquema de firma digital es una tripleta de algoritmos polinomiales ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif}$) donde:*

- KeyGen*, el* algoritmo de generación de claves*, es un algoritmo probabilístico que toma como entrada el parámetro de seguridad $\mathbf{1}^\lambda$ y devuelve un par de claves público/privada (pk, sk).*

- `Sign`, *el* algoritmo de firmado, *es un algoritmo probabilístico que toma como entrada un mensaje m del espacio de mensajes $\mathcal{M}$ y una clave secreta sk y devuelve una firma σ.*
- `Verif`, *el* el algoritmo de verificación, *es un algoritmo determinista que toma una firma σ, un mensaje $m \in \mathcal{M}$, y una clave pública pk y devuelve un bit en $\{0, 1\}$, comprobando si σ es una firma válida de m con respecto a pk.*

1.2.3.1. Nociones de seguridad

Antes de dar una definición formal de seguridad, listaremos los diferentes objetivos de los adversarios y los modelos de ataques, que intentan capturar las principales estrategias de ataque que deberían ser prevenidos por cada adversario específico. Denotaremos con $\mathcal{A}$ a un adversario (probabilístico en tiempo polinomial). Asumiremos que $\mathcal{A}$ persigue uno de las siguientes objetivos:

- Existential Forgery (EF): producir una firma válida para un mensaje m, no necesariamente elegido por él.
- Selective Forgery (SF): producir una firma válida para algún mensaje m fijo, elegido por él.
- Universal Forgery (UF): producir una firma válida para *cualquier* mensaje.
- Total Break (TB): recuperar, a partir de información pública, una clave secreta legítima del emisor de la firma.

De forma similar, para capturar las capacidades del adversario, distinguimos entre los siguientes *modelos de ataque*:

- No Message Attack (NMA): $\mathcal{A}$ sólo conoce los parámetros públicos (en particular, la clave pública de firmado).
- Random Message Attack (RMA): $\mathcal{A}$ tiene firmas de una secuencia de mensajes seleccionados uniformemente al azar.
- Chosen Message Attack (CMA): $\mathcal{A}$ tiene acceso a un oráculo de firmado, que firma un mensaje elegido por $\mathcal{A}$. Las consultas a este oráculo pueden ser adaptativas, es decir, $\mathcal{A}$ puede adaptar los mensajes de entrada en función de las firmas obtenidas previamente.

Las nociones formales de seguridad combinan los objetivos del adversario y sus capacidades. Por ejemplo, un esquema de firma es seguro en el sentido de UF-NMA si, dado cualquier adversario probabilístico en tiempo polinomial $\mathcal{A}$,

existe una función despreciable negl que depende del parámetro de seguridad λ, acotando la probabilidad de éxito de un ataque UF, teniendo $\mathcal{A}$ acceso solamente a información pública (NMA). Otras nociones de seguridad se definen de forma análoga; por ejemplo, EUF-CMA captura el hecho de que un adversario CMA no será capaz de producir un ataque EF.

A continuación, definimos las nociones de seguridad más relevantes para esquemas de firma.

Definición 1.11. *Un esquema de firma* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice que es UF-NMA (Universally unForgeable under No-Message Attack) si para cualquier adversario probabilístico en tiempo polinomial* $\mathcal{A}$ *y* $\forall m \in \mathcal{M}$*, entonces:*

$$\Pr\begin{bmatrix}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ \sigma \leftarrow \mathcal{A}(pk, m);\\ \mathsf{Verif}(m, \sigma, pk) = 1\end{bmatrix} \leq \text{negl}(\lambda).$$

La definición establece que, dado un mensaje y una clave pública, un adversario debería tener una probabilidad despreciable de construir una firma válida.

Definición 1.12. *Un esquema de firma* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice UF-RMA (Universally unForgeable under Random-Message Attack) si para cualquier adversario probabilístico en tiempo polinomial* $\mathcal{A}$ *y* $\forall m \in \mathcal{M}$*, entonces:*

$$\Pr\begin{bmatrix}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ \{m_i\}_{i=1}^k \xleftarrow{\$} \mathcal{M} \setminus \{m\};\\ \{\sigma_i\}_{i=1}^k \leftarrow \mathsf{Sign}(sk, \{m_i\}_{i=1}^k);\\ \sigma \leftarrow \mathcal{A}(pk, \{(m_i, \sigma_i)\}_{i=1}^k, m);\\ \mathsf{Verif}(m, \sigma, pk) = 1\end{bmatrix} \leq \text{negl}(\lambda).$$

La definición anterior establece que, dada una lista de pares (mensaje, firma), donde los mensajes son seleccionados uniformemente al azar, el adversario debería seguir teniendo una probabilidad despreciable de construir una firma válida.

Definición 1.13. *Un esquema de firma* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ *con espacio de mensajes* $\mathcal{M}$ *y parámetro de seguridad* λ *se dice EUF-CMA (Existentially unForgeable under adaptive Chosen-Message Attack), si para cualquier adversario probabilístico en tiempo polinomial* $\mathcal{A}$ *con acceso polinomial a un oráculo de firma* $\mathcal{O}_{sk}$ *que produce firmas válidas*

con respecto a una cierta clave secreta sk, entonces:

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda);\\ \{\sigma_i\}_{i=1}^k \leftarrow \mathcal{O}_{\mathrm{sk}}(\{m_i\}_{i=1}^k);\\ (m, \sigma) \leftarrow \mathcal{A}(pk, \{(m_i, \sigma_i)\}_{i=1}^k);\\ \mathsf{Verif}(m, \sigma, pk) = 1 \ \ y \ m \notin \{m_1, \ldots, m_k\}\end{array}\right] \leq \mathrm{negl}(\lambda).$$

En la definición anterior, el adversario tiene acceso a un oráculo de firma que produce firmas válidas con respecto al par de claves que pretende atacar y se enfrenta al reto de producir una firma válida para un mensaje. Este modelo es particularmente relevante para capturar ataques de maleabilidad, que explota la posibilidad de derivar nuevas firmas válidas a partir de otras legítimas.

La noción de seguridad estándar para firmas es EUF-CMA, que es la más fuerte entre las tres nociones introducidas. En concreto, cada esquema de firma EUF-CMA es también UF-RMA y todo esquema de firma UF-RMA es también UF-NMA.

1.2.4. Intercambio de clave

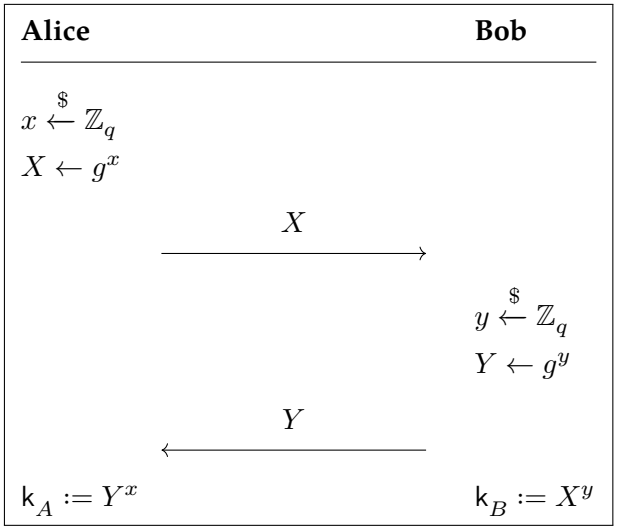


Figura 1.1: Intercambio de clave de Diffie-Hellman.

El intercambio de clave es otra de las primitivas fundamentales en criptografía asimétrica: permite obtener una clave común con la que poder comunicarse a través de un canal inseguro.

El protocolo de intercambio de clave más conocido es el protocolo de Diffie-Hellman, propuesto en 1976 [39].

Alice y Bob quieren acordar una clave común a través de un canal inseguro. Para ello, acuerdan públicamente un elemento generador g de orden q que genera un grupo multiplicativo $\mathbb{G}$. Alice y Bob seleccionan al azar los valores x, y de $\mathbb{Z}_q$, respectivamente. Alice calcula $X = g^x$ y Bob, $Y = g^y$ e intercambian estos valores. Como último paso, Alice y Bob elevan el valor que han recibido del otro con el valor al azar que han calculado originalmente. Ambos obtienen el valor común $\mathsf{k}_A = Y^x = (g^y)^x = g^{yx} = g^{xy} = (g^x)^y = X^y = \mathsf{k}_B$. La Figura 1.1 muestra su funcionamiento.

El protocolo de Diffie-Hellman tiene un problema evidente: ningún mensaje es autenticado, lo que hace a este protocolo susceptible de ataques Man In The Middle (MITM), como el que se ve en la Figura 1.2.

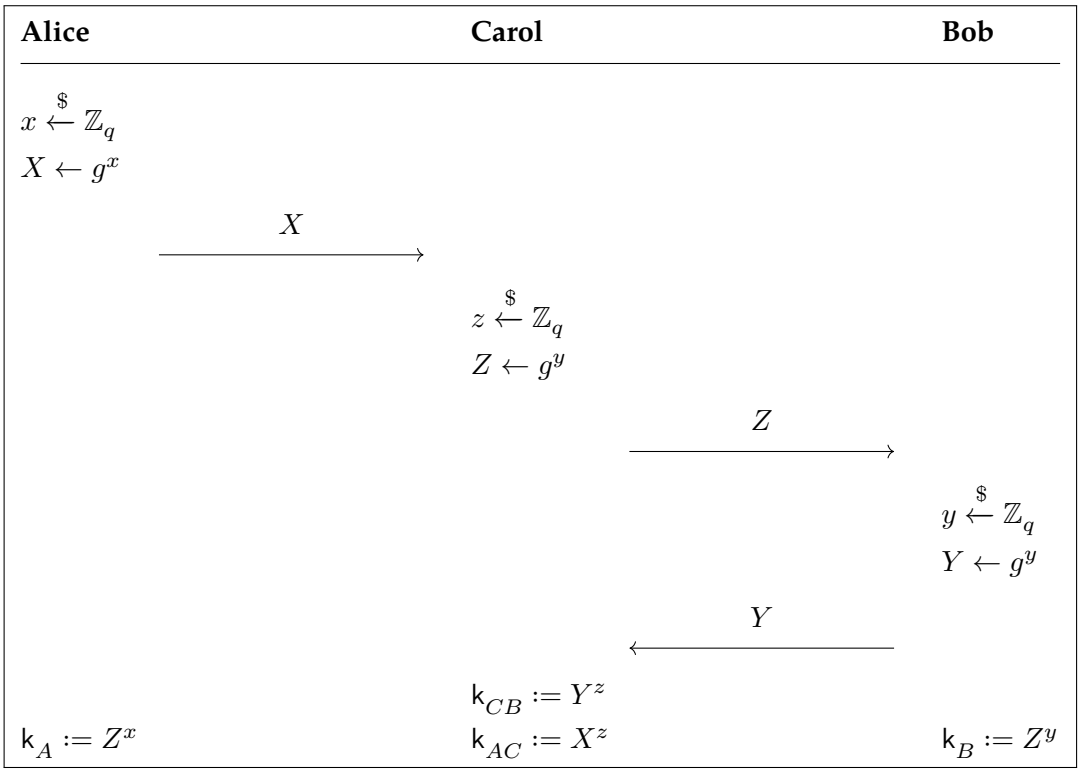


Figura 1.2: Ataque Man In The Middle al protocolo de Diffie-Hellman.

De igual forma que en el caso anterior, Alice y Bob quieren obtener una clave común, pero en esta ocasión Carol intercepta los mensajes entre ellos, haciéndose pasar por Alice o Bob según corresponda. Carol ejecuta el protocolo de Diffie-Hellman con Alice y Bob de forma independiente, pudiendo descifrar a posteriori los mensajes que se intercambien. Se puede comprobar que la clave obtenida por Alice y Bob no coincide. En efecto, $\mathsf{k}_A = Z^x = (g^z)^x \neq (g^z)^y = Z^y = \mathsf{k}_B$.

1.2.4.1. Intercambio de clave autenticado

Así pues, es necesario disponer de protocolos que garanticen la autenticidad de los participantes en el mismo. Los protocolos autenticados manejan dos tipos de claves:

- Claves de sesión: claves que son establecidas durante la ejecución del protocolo. Las claves de sesión se suelen emplear a posteriori con esquemas de cifrado simétricos como AES. Deben ser renegociadas cada cierto tiempo.
- Claves de larga duración (o de autenticación): claves ya existentes antes de la ejecución del protocolo y utilizadas para reconocer a los usuarios legítimos. Estas claves pueden ser de varios tipos:
  - Claves (pre)compartidas: claves para usar con esquemas de cifrado simétricos. Un ejemplo de este tipo es la clave WiFi de los *routers*. Dentro de este grupo se encuentran las contraseñas, que se caracterizan por su baja entropía en contraposición a otro tipo de claves. Un ejemplo de contraseña es el PIN del móvil.
  - Claves basadas en identidad: emplean atributos públicos de una entidad como clave pública. Estos atributos son representados como cadenas de bits. Ejemplos de atributos públicos pueden ser el correo electrónico, el DNI, el número de teléfono, un dominio, etc.
  - Pares de claves público/privadas: para usar este tipo de claves es necesario disponer de infraestructura de clave pública (PKI, por sus siglas en inglés) que usan los participantes en el protocolo para comprobar la validez de las claves públicas, que se realiza a través de certificados digitales. En la mayoría de los protocolos se omiten los detalles de comunicación con la PKI.

A lo largo de esta tesis, las claves de larga duración serán de este último tipo. Además, llamaremos al intercambio de clave autenticado entre dos únicos participantes, AKE o 2-AKE, según convenga.

1.2.5. Intercambio de clave (autenticado) en grupo

El intercambio de clave autenticado en grupo (GAKE, por sus siglas en inglés) es el tema central de esta tesis. Consiste en la generalización de los protocolos AKE a un grupo de $n \geq 2$ participantes. Del mismo que en el caso anterior, el objetivo de los protocolos GAKE es acordar una clave de sesión entre todos los participantes en el mismo (Figura 1.3). Además, por ser autenticado, los participantes disponen de claves de larga duración que, en nuestro caso, serán claves

público/privadas, pero existen protocolos para el resto de tipos de claves (por ejemplo, ver [26]).

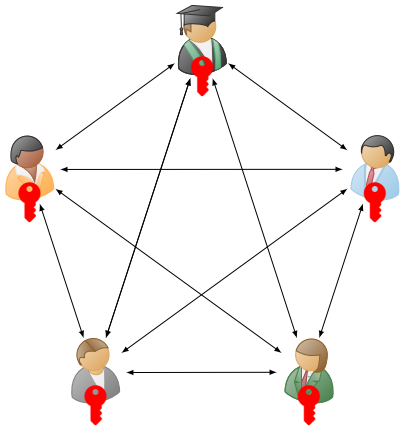


Figura 1.3: Vista general de un protocolo de intercambio de clave en grupo.

A la hora de diseñar o elegir un protocolo GAKE hay que considerar distintos factores:

- Tipo de aplicación: una característica fundamental es el número de participantes que envían información. La transmisión de un mensaje a un grupo, sólo tiene un emisor, mientras que un *chat* de grupo permite a todos sus miembros mandar mensajes.
- Tamaño del grupo: el tamaño del grupo requerido puede limitar cómo se diseña un protocolo; por ejemplo, para un grupo pequeño puede ser factible realizar el intercambio de forma interactiva, pero puede no serlo para un grupo con un tamaño muy grande por el elevado coste de comunicación entre los participantes.
- Escalabilidad: la eficiencia del protocolo puede verse comprometida a medida que el grupo incorpora nuevos miembros. Dos aspectos fundamentales que afectan a la escalabilidad son:
  - Número de rondas: una ronda contiene mensajes que pueden ser enviados simultáneamente. El número de rondas es especialmente importante si depende del número de participantes en el protocolo.
  - Mensajes transmitidos: algunos protocolos requieren la transmisión de algunos mensajes al resto de participantes. Esto puede suponer un coste muy alto si el número de participantes es muy grande. Es importante también medir el número de mensajes transmitidos y enviados punto a punto, es decir, de un participante a otro.

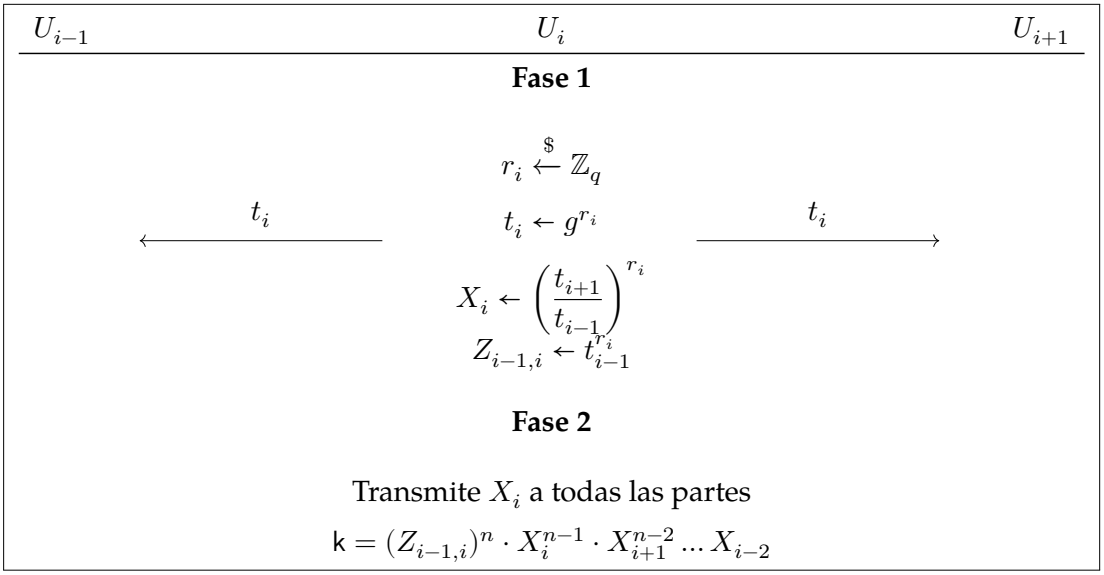


Figura 1.4: Intercambio de clave en grupo de Burmester-Desmedt.

1.2.5.1. Algunos protocolos

El protocolo de Ingemarsson et al. [66], abreviado como ITW es una generalización del protocolo de Diffie-Hellman basado en la idea de funciones simétricas[2], distribuyendo a los participantes en forma de anillo. Klein et al. [74] añade autenticación a ITW. Posteriormente a la publicación de ITW, Steiner et al. [116], propone tres variaciones del protocolo llamadas GDH.1, GDH.2 y GDH.3. Las diferencias entre ellas radican en cómo se envían los mensajes y se realizan los cómputos, dotando de flexibilidad a los protocolos en función de la aplicación. En [12], se publican versiones autenticadas para GDH.2 y GDH.3. Kim et al. [94], proponen un protocolo donde los participantes son las hojas de un árbol binario, añadiendo autenticación con firmas digitales.

El protocolo más conocido de este tipo es el Burmester-Desmedt [27], que propone un protocolo de 2 rondas y un número constante de exponenciaciones modulares por usuario (Figura 1.4). Sólo es seguro frente a adversarios pasivos. Este protocolo ha sido fuente de inspiración para otros diseños posteriores.

Otros protocolos no se basan en el protocolo de Diffie-Hellman. La propuesta de [96], se basa en el esquema de Shamir [111]. En [54], se propone un protocolo de una ronda, seguro en el modelo estándar, que usa un mKEM, que es una generalización de un KEM. Como punto negativo, no es un esquema muy eficiente. Otras propuestas se pueden encontrar en [21, 113, 123].

[2]La j-ésima función simétrica sobre el conjunto $S = \{r_1, r_2, \ldots, r_m\}$ consiste en la suma de todos los posibles productos de j elementos de S [26].

1.2.5.2. Compiladores

A continuación, introducimos la noción de compilador en este contexto. Los compiladores son protocolos que tienen como entrada un esquema criptográfico y devuelven un diseño para un fin similar con ciertas propiedades adicionales. Aquí, nos centraremos en aquellos que producen protocolos de intercambio de clave en grupo.

El compilador de Katz y Yung [73] es un compilador genérico que permite convertir un protocolo de intercambio de clave en grupo no autenticado en uno autenticado. Para ello, hace uso de un esquema de firma digital y añade una ronda de comunicación adicional. Este mecanismo simplifica el diseño de este tipo de protocolos, ya que no es necesario tener en cuenta la autenticación, sino que en primera instancia, se diseña sin autenticación y, luego, se añade con este compilador. Sin embargo, esto puede no ser ideal, ya que se introduce un esquema de firma que puede reducir la practicidad del protocolo.

El compilador de Just y Vaudenay [70], basado en el diseño del protocolo de Burmester-Desmedt [27] convierte un protocolo de intercambio de clave 2-parte en un protocolo de intercambio de clave en grupo autenticado. Sin embargo, añade un esquema de firma digital, que puede hacer que no sea práctico el protocolo. Además, este diseño carece de un análisis de seguridad riguroso.

Figura 1.5: Vista de alto nivel del compilador de Abdalla et al.

1.2.5.2.1. Compilador de Abdalla et al. El compilador de Abdalla et al. [1] permite obtener un protocolo GAKE a partir de un intercambio de clave autenticado 2-parte (Figura 1.5), que llamaremos 2AKE. El compilador no depende de otras técnicas de autenticación distintas a las usadas por 2AKE, ni de hipótesis idealizadas, sólo asume que los participantes están distribuidos en forma de anillo (ver Figura 1.6), inspirado por el protocolo de Burmester-Desmedt (Figura 1.4). No requiere de ningún esquema de firma digital.

Además, si el 2AKE requiere r rondas de comunicación, el protocolo GAKE obtenido requiere $r + 2$ rondas, es decir, el número de rondas es independiente del número de participantes en el protocolo.

Sea $\mathcal{P}$ el conjunto de los usuarios que pueden participar en el protocolo GAKE. El conjunto $\mathcal{G} = \{U_0, U_1, \ldots, U_{n-1}\} \subset \mathcal{P}$ denota al conjunto de los $n > 2$ participantes que desean establecer una clave de sesión de sesión común. Cada

participante en el protocolo $U_i \in \mathcal{G}$, $i = 0, \ldots, n-1$, puede ejecutar distintas ejecuciones del protocolo GAKE.

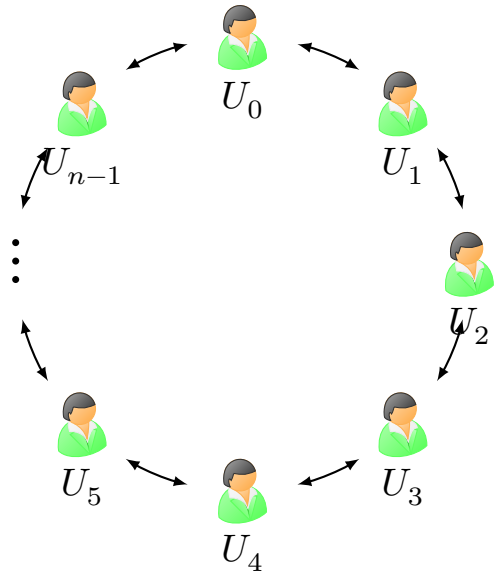


Figura 1.6: Disposición en anillo empleada por el compilador de Abdalla et al.

Como el protocolo 2AKE es un intercambio de clave autenticado, se asume que las claves de larga duración se han establecido durante una fase previa, realizada de forma confiable. En concreto, se asume uno de estos tres casos:

- Cada usuario $U_i \in \mathcal{G}$ posee un par (pk_i, sk_i) consistente en una clave pública pk_i y una clave privada sk_i, y todas las claves públicas necesarias para la ejecución del protocolo se han distribuido a todos los participantes durante una fase de inicialización.
- Cada par de usuarios $U_i, U_j \in \mathcal{G}$, $i \neq j$, comparten una clave simétrica de alta entropía, o el conjunto completo de participantes $\mathcal{G}$ comparte un secreto común (instancias diferentes de un usuario puede usar secretos distintos).
- Cada par de usuarios $U_i, U_j \in \mathcal{G}$, $i \neq j$, comparten una contraseña de baja entropía. En este caso, se asume un diccionario disponible públicamente $\mathcal{D} \subseteq \{0,1\}^*$, del que las contraseñas se eligen uniformemente al azar.

El compilador usa las siguientes herramientas criptográficas:

1. **Un esquema de compromiso no interactivo y no maleable** [41] C *perfectly binding* que consigue *no maleabilidad para múltiples compromisos*, es decir,

que el valor del compromiso es realmente el valor que se comprometió y, que no es posible construir de ninguna manera un compromiso que relacione a este con un conjunto de compromisos dados.

2. **Una familia de funciones pseudoaleatorias resistentes a colisiones** $\mathsf{F} = \{F^\ell\}_{\ell\in\mathbb{N}}$ con $F^\ell = \{F^\ell_\eta\}_{\eta\in\{0,1\}^L}$ indexado por el conjunto $\{0,1\}^L$ de tamaño polinomial, y dos valores conocidos públicamente v_0 y v_1 tal que ningún adversario probabilístico en tiempo polinomial pueda encontrar dos índices distintos $\lambda \neq \mu \in \{0,1\}^L$ tal que $F^\ell_\lambda(v_j) = F^\ell_\mu(v_j)$, $j = 0, 1$.

3. **Una función *hash*** H seleccionada de una familia de funciones *hash* universales que *mapea* la concatenación de cadenas de bits de $\{0,1\}^{kn}$ y el conjunto de participantes $\mathcal{G}$ en $\{0,1\}^L$, donde n el número de participantes de $\mathcal{G}$ y $k \in \mathbb{N}$.

El compilador está descrito en la Figura 1.7.

1.3. Criptografía híbrida

La criptografía híbrida toma las ventajas de la criptografía simétrica y asimétrica para construir esquemas combinando lo mejor de cada una de estas técnicas.

La criptografía simétrica es superior a la asimétrica en velocidad, por lo que es más adecuada para cifrar grandes cantidades de datos. Sin embargo, requiere que la clave de cifrado sea compartida por todas las partes. Esto se puede solucionar con el uso de la criptografía asimétrica.

Sólo nos centraremos en la construcción KEM/DEM, que permite obtener un PKE IND-CCA a partir de un KEM IND-CCA y un esquema de cifrado simétrico (normalmente llamado DEM, que procede de *Data Encapsulation Mechanism*). Esta construcción se debe a Cramer y Shoup [32].

Dado un esquema de cifrado simétrico SKE=(E, D) y un KEM=(KeyGen, Encaps, Decaps), el esquema HPKE=(KeyGen, Enc, Dec) se construye de la siguiente forma:

- HPKE.KeyGen: se emplea KEM.KeyGen.
- HPKE.Enc: toma un mensaje m para cifrar y una clave pública pk y devuelve un texto cifrado C. Primero, se ejecuta KEM.Encaps con la clave pública pk para generar una clave simétrica k y una encapsulación de k denotada por c. Entonces, se cifra m con el esquema simétrico SKE.E, obtenido un texto cifrado c'. La salida del algoritmo es $C = (c, c')$.

Ronda $1 \sim r$:

- Por cada $i = 0, \ldots, n-1$, ejecutar 2AKE con U_i y $U_{[i+1]}$, donde $[k] = k \bmod n$. Por tanto, cada U_i obtiene dos claves $\overrightarrow{K_i}$ y $\overleftarrow{K_i}$, compartidas con $U_{[i+1]}$ y $U_{[i-1]}$, respectivamente. Nótese que $\overrightarrow{K_i} = \overleftarrow{K}_{[i+1]}$ para $i = 0, \ldots, n-1$.

Ronda $r+1$:

- Cada U_i computa $X_i = \overrightarrow{K_i} \oplus \overleftarrow{K_i}$ y elige un r_i al azar para calcular un compromiso $C_i = \mathsf{C}(i, X_i, r_i)$.

- Cada U_i transmite $M_i^1 = (U_i, C_i)$.

Ronda $r+2$:

- Cada U_i transmite $M_i^2 = (U_i, X_i, r_i)$.

- Cada U_i comprueba que $X_0 \oplus X_1 \oplus \ldots \oplus X_{n-1} = 0$ y la corrección de los compromisos. Si cualquiera de estas comprobaciones falla, entonces U_i termina la ejecución del protocolo sin obtener una clave de sesión.

- Cada U_i computa los $n-1$ valores

$$K_{[i-j]} = \overleftarrow{K}_{[i-1]} \oplus X_{[i-j]} \oplus \cdots \oplus X_{[i-j]}, \quad j = 1, 2, \ldots, n-1.$$

  Entonces, U_i define una clave maestra

$$K = (K_0,\, K_1, \ldots,\, K_{n-1},\, \mathsf{pid}_i)\,,$$

  y establece la clave de sesión $\mathtt{sk}_i = F^{\ell}_{\mathsf{H}(K)}(v_1)$ y el identificador de sesión $\mathtt{sid}_i = F^{\ell}_{\mathsf{H}(K)}(v_0)$, donde $\ell \in \mathbb{N}$ es el parámetro de seguridad.

Figura 1.7: Compilador de Abdalla et al.

- `HPKE.Dec`: toma un texto cifrado $C = (c, c')$ y una clave secreta sk y devuelve un mensaje m o un error de descifrado $\perp$. En primer lugar, se desencapsula c con `KEM.Dec` dando lugar a k, que se emplea para descifrar c' usando `SKE.D`, devolviendo el mensaje m o un error $\perp$.

La seguridad del esquema `HPKE` viene dada por el Teorema 5 de [32], que asegura que `HPKE` es un PKE IND-CCA siempre y cuando sean `SKE` y `KEM` también lo sean.

Capítulo

2

Criptografía postcuántica

En este capítulo, se presentan los algoritmos cuánticos y cómo afectan a la criptografía tradicional, dando lugar a la criptografía postcuántica. A continuación, se describe el concurso de estandarización del NIST, que es uno de los principales focos de discusión y desarrollo de herramientas postcuánticas a nivel internacional. En particular, nos centramos en presentar dos tipos de herramientas: los KEM que serán de utilidad en los capítulos posteriores y el esquema de firma digital basada en teoría de trenzas llamado WalnutDSA. Finalmente, se muestran los avances en términos de implementaciones disponibles.

2.1. Computación cuántica

La computación cuántica es una apasionante área de investigación que pone su foco en el diseño y desarrollo de arquitecturas que usan tecnología cuántica, así como en el diseño de algoritmos que puedan ser ejecutados en ellas. Los ordenadores cuánticos tendrán la capacidad de resolver eficientemente problemas que los ordenadores clásicos tardan demasiado tiempo en abordar y esto tiene graves implicaciones en criptografía.

Nadie conoce con exactitud cuándo se conseguirá desarrollar tecnología cuántica para romper *de facto* los esquemas criptográficos conocidos. Michele Mosca estimó en 2017 que hay una posibilidad entre 6 de que RSA-2048 sea roto en una década y un 50 % de posibilidades de que sea roto en los próximos 15 años [82]. En cualquier caso, la comunidad criptográfica internacional asume que es imprescindible disponer a medio plazo de algoritmos y protocolos que sean resistentes frente a ataques implementados con ordenadores cuánticos. Hablamos a continuación de los dos algoritmos cuánticos más relevantes en este contexto.

2.1.1. Algoritmo de Grover

El algoritmo de Grover [55] es un algoritmo cuántico que permite buscar un elemento en una secuencia no ordenada de n datos en tiempo $\mathcal{O}(\sqrt{n})$, frente a la cota clásica de $\mathcal{O}(n)$. En criptografía, esto se traduce en una aceleración cuadrática a la hora de encontrar una colisión de una función *hash* o una clave simétrica frente a los recursos clásicos. La contramedida frente a este algoritmo consiste en duplicar la longitud de clave y el tamaño de los *hash* para alcanzar los niveles de seguridad actual.

2.1.2. Algoritmo de Shor

El algoritmo de Shor [112] es un algoritmo cuántico para descomponer un número entero en sus factores primos y resolver el problema del logaritmo discreto en tiempo polinomial. Esquemas como el cifrado RSA fundamentan su seguridad en la dificultad de factorizar números grandes, por lo que no deberán ser utilizados sin un aumento significativo de los tamaños de clave asociados en presencia de adversarios con acceso a ordenadores cuánticos. De hecho, el alcance del algoritmo de Shor es muy amplio y deja en evidencia la seguridad de numerosas construcciones ampliamente utilizadas más allá de RSA, como aquellas que utilizan el problema del logaritmo discreto en grupos asociados a cuerpos finitos o curvas elípticas. En este sentido es indudable que, en gran medida, la criptografía asimétrica actual no será segura frente a adversarios cuánticos.

2.2. ¿Qué es la criptografía postcuántica?

La criptografía postcuántica tiene por objetivo diseñar e implementar esquemas y protocolos resistentes a ataques que utilicen computación cuántica. Estas nuevas construcciones basan su seguridad en problemas matemáticos de elevada dificultad y para los que la computación cuántica no parece aportar ventajas notables. Es importante destacar que estos nuevos algoritmos son clásicos en

el sentido que pueden ser implementados y ejecutados en cualquier ordenador, sin requerir de computación cuántica.

Se han propuesto aproximaciones a la criptografía postcuántica desde distintas áreas de las matemáticas, algunas muy poco exploradas en criptografía hasta ahora. Habitualmente se establece una clasificación en cinco ámbitos: criptografía basada en retículos, en códigos, multivariante, basada en *hashes*, en códigos y en isogenias.

2.2.1. Demostraciones en el mundo cuántico

Cuando se demuestra que un protocolo es seguro, es necesario definir un modelo de seguridad, haciendo explícitas afirmaciones e hipótesis que puedan ser utilizadas y/o verificadas formalmente. Este no es siempre el caso del mundo postcuántico, ya que los adversarios cuánticos son modelados de forma distinta. A menudo, las construcciones se fundamentan en hipótesis computacionales que explícitamente establecen que un adversario no es capaz de completar eficientemente una tarea computacional. Modelar la interacción entre adversarios cuánticos y otros sistemas (clásicos o cuánticos) plantea dificultades notables. Un caso paradigmático es el caso de las funciones *hash*, que son modeladas típicamente como oráculos aleatorios, dentro del *Random Oracle Model* o ROM (ver Sección 1.1.2.2).

Los oráculos aleatorios son usados para modelar funciones hash idealizadas, que son algoritmos deterministas que seleccionan, para cada nueva entrada, una salida elegida uniformemente al azar de un determinado rango. Se asume que todos los usuarios y procesos de un cierto sistema tienen acceso a los mismos oráculos aleatorios, lo que implica que, en las demostraciones de seguridad, si el entorno criptográfico real es simulado por un adversario, todas las consultas a los oráculos aleatorios deben ser respondidas con valores que son indistinguibles de valores al azar. En el mundo cuántico, las consultas a un oráculo aleatorio pueden realizarse en superposición, con lo que, de algún modo, ha de representarse una evolución simultánea de un número exponencial de valores. Esto complica significativamente cómo trasladar las demostraciones del mundo clásico al escenario cuántico.

2.2.2. Quantum Random Oracle Model

El *Quantum Random Oracle Model* (QROM) es un modelo de seguridad que generaliza al ROM, en el que el adversario tiene acceso a un oráculo aleatorio cuántico que puede realizar consultas en superposición cuántica de estados. Esto no es captado por el ROM y, en un escenario cuántico, un adversario de este tipo tiene una considerable ventaja sobre un adversario clásico. Las demostraciones

que se realizan en el QROM son también válidas en el ROM, pero el recíproco no es cierto.

Como en [64], consideraremos adversarios cuánticos que tienen acceso cuántico a oráculos cuánticos (*offline*), que serán de utilidad en el diseño de la construcción del Capítulo 3. En concreto, necesitaremos hacer uso de dos propiedades básicas de los oráculos aleatorios denominados *quantum-accessible*:

- Resistencia a colisiones (*Collision-freeness*). En [29], se demuestra que el mejor algoritmo para encontrar colisiones en una función aleatoria $H : \{0,1\}^n \to \{0,1\}^n$ (es decir, un par de $x, x' \in \{0,1\}^n; x \neq x'$ tal que $H(x) = H(x')$) es $\tilde{\mathcal{O}}(2^{\frac{n}{5}})$.[1] La cota clásica es $\mathcal{O}(2^{\frac{n}{2}})$. Mientras que es subóptimo en el número de consultas, este algoritmo es el más eficiente en términos de tiempo con poca memoria cuántica. Por tanto, asumiremos en lo sucesivo que encontrar una colisión, incluso para un adversario cuántico, para un oráculo aleatorio *quantum-accessible* sólo puede ser realizado con probabilidad despreciable.

- Pseudoaleatoriedad (*Pseudorandomness*). Siguiendo de nuevo [64], usamos el razonamiento de Zhandry [127] que establece que, ningún algoritmo cuántico, haciendo como mucho q consultas al oráculo aleatorio cuántico $\mathcal{O}_{\mathcal{H}}$ que implementa una función aleatoria $\mathcal{H} : \{0,1\}^m \to \{0,1\}^n$, que puede distinguir entre las salidas de $\mathcal{O}_{\mathcal{H}}$ y una fuente de aleatoriedad real. Como resultado, si la entrada a un oráculo aleatorio cuántico contiene suficiente entropía, la probabilidad de distinguir su salida de un valor elegido uniformemente al azar es despreciable. En otras palabras, cuando la entrada es desconocida y elegida uniformemente al azar, el hecho de que el oráculo aleatorio puede ser consultado en superposición no es de ayuda a distinguir la salida del oráculo de un elemento elegido al azar.

2.2.3. Aproximaciones a la criptografía postcuántica

A continuación, se introducen las áreas más relevantes en criptografía postcuántica y sus problemas computacionales más estudiados en la literatura.

2.2.3.1. Criptografía basada en retículos

Informalmente, un retículo es un conjunto de puntos regularmente espaciados que se extienden hasta el infinito en un espacio n-dimensional. La Figura 2.1 muestra un ejemplo en 2 dimensiones.

[1] La notación $\tilde{\mathcal{O}}$ elimina los factores logarítmicos de $\mathcal{O}$, es decir, $f(n) \in \tilde{\mathcal{O}}(h(n)) \iff \exists k \in \mathbb{N}$ tal que $f(n) \in \mathcal{O}(h(n)\log^k(h(n)))$.

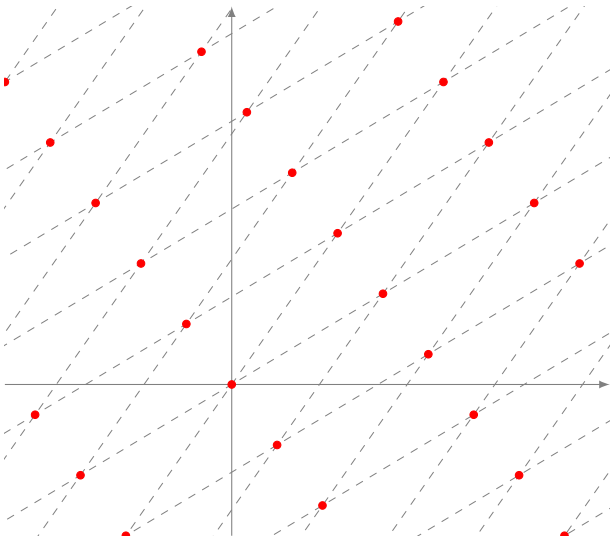


Figura 2.1: Ejemplo de retículo en 2 dimensiones.

Definición 2.1. *Un retículo es el conjunto de todas las combinaciones enteras de n vectores linealmente independientes $\mathbf{b}_1, \dots, \mathbf{b}_n$ en $\mathbb{R}^n$.*

$$\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$$

A los vectores $\mathbf{b}_1, \dots, \mathbf{b}_n$ se les llama base del retículo. Una base se puede representar como una matriz $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ escrita con los vectores de la base como columnas. En notación matricial, se puede definir el retículo generado por la matriz $\mathbf{B} \in \mathbb{R}^{n \times n}$ como $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$.

Los problemas computaciones más conocidos de retículos son los siguientes:

- Problema del vector más corto (Shortest Vector Problem o SVP): dada una base $\mathbf{B}$ de un retículo, encontrar el vector no negativo más corto de $\mathcal{L}(\mathbf{B})$.
- Problema del vector más cercano (Closest Vector Problem o CVP): dada una base $\mathbf{B}$ de un retículo y un vector objetivo $\mathbf{t}$ (no necesariamente en el retículo), encontrar el punto en el retículo $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ más cercano a $\mathbf{t}$.
- Problema del vector independiente más corto (Shortest Independent Vector Problem o SIVP): dada una base $\mathbf{B} \in \mathbb{R}^{n \times n}$ de un retículo, encontrar n vectores linealmente independientes de un retículo $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_n]$, donde $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$ para todo i, minimizando la cantidad $||\mathbf{S}|| = \max_i ||\mathbf{s}_i||$.

El auge de la criptografía postcuántica ha impulsado el estudio de otros problemas, entre los que destaca el llamado problema *Learning With Errors*, abreviado como LWE, que consiste en encontrar un vector secreto $\mathbf{s} \in \mathbb{Z}_q^n$, dada una matriz $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ y $\mathbf{b} = \mathbf{As} + \mathbf{e}$ siendo $\mathbf{e}$ un vector aleatorio de ruido, generado a partir de cierta distribución estadística (conocida).

Este problema ha dado lugar a otras variantes del mismo como *Ring-Learning With Errors* (RLWE), que reemplaza el grupo $\mathbb{Z}_q^n$ por un anillo, típicamente $\mathbb{Z}_q^n[X]/(X^n+1)$ [102].

Otro problema relevante derivado de LWE es Module-LWE (Definición 2.2). La dificulad de dicho problema fundamenta la seguridad de Kyber, un KEM que es central a lo largo de la tesis y que se describe en la Sección 2.3.1.1.

2.2.3.2. Criptografía basada en códigos

Los algoritmos criptográficos basados en códigos se apoyan en códigos correctores de errores. Estos se han empleado típicamente en transmitir información a través de un canal ruidoso. La idea fundamental es basar la seguridad de los criptosistemas en la dificultad de descodificar un código lineal aleatorio, que es un problema NP-hard [78].

El primer esquema de este tipo se debe a McEliece [78] y fue propuesto en 1978. Otros esquemas usan la construcción dual de Niederreiter [87]. Más recientemente, se ha propuesto el esquema *Quasi-Cyclic* [79].

El lector interesado en este tipo de criptografía puede encontrar más información en [92, 122].

2.2.3.3. Criptografía multivariante

La criptografía multivariante trata de construir esquemas criptográficos cuya seguridad se basa en ecuaciones multivariantes, es decir, ecuaciones no lineales con varias incógnitas. Por ejemplo, el siguiente sistema con incógnitas x, y, z es un sistemas multivariante.

$$\begin{cases} 3x^2 + y + z = 2 \\ xy + z^2 = 10 \\ x + y^2 + z^2 = 3 \end{cases}$$

La resolución de este tipo de sistemas es un problema NP-hard [126].

2.2.3.4. Criptografía basada en hashes

La criptografía basada en *hashes* se apoya en la robustez de ciertas funciones *hash*, que no son afectadas en gran medida por un ordenador cuántico (ver

detalles en la Sección 2.1.1).

2.2.3.5. Criptografía basada en isogenias

La criptografía basada en isogenias se basa en curvas elípticas. De forma informal, una curva elíptica está formada por los puntos de una determinada estructura algebraica que cumplen la siguiente ecuación $y^2 = x^3 + ax + b$, con $4a^3 + 27b^2 \neq 0$. Sobre las curvas elípticas se define una operación de suma que las dota de estructura de grupo.

Una isogenia entre dos curvas elípticas es una aplicación que puede ser escrita como una fracción de polinomios y que preserva la operación de suma en ambas curvas elípticas. El problema de la isogenia consiste en encontrar una isogenia entre dos curvas elípticas, sabiendo que dicha aplicación existe [18].

2.3. Concurso de estandarización del NIST

Tabla 2.1: Finalistas del concurso de estandarización del NIST.

Estatus	KEM	Firma
Finalistas	Classic McEliece Kyber NTRU SABER	Dilithium Falcon Rainbow
Candidatos alternativos	BIKE FrodoKEM HQC NTRU Prime SIKE	GeMSS Picnic SPHINCS+

En 2017, el NIST (Instituto de Estándares y Tecnología de EE.UU.), comenzó un concurso público con el fin último de estandarizar uno o más algoritmos postcuánticos de clave pública [88]. El concurso busca nuevos esquemas de firma digital y KEM. Se estructura en rondas. En cada una de ellas, se analizan las propuestas en busca de debilidades de seguridad (tanto teóricas como fallos en la implementación) y mejoras en el rendimiento. Al finalizar cada ronda, sólo las mejores propuestas (en términos de seguridad y rendimiento) pasan a la siguiente ronda. Inicialmente, se presentaron 82 propuestas de las que 69 fueron aceptadas, aunque 5 fueron retiradas, siendo 19 propuestas de firma digital y

45 KEM. En la actualidad, tras tres rondas, ya se conocen los 7 finalistas y los 8 candidatos alternativos[2] que se muestran en la Tabla 2.1.

El NIST define 5 niveles de seguridad en el que se categorizan las propuestas y permite comparar las propuestas entre ellas, en términos de uso de recursos computacionales comparables a los requeridos para buscar una clave en un esquema de cifrado de bloque o la búsqueda de colisiones en una función hash. Estos niveles se definen en la Tabla 2.2.

Tabla 2.2: Niveles de seguridad que definido en el concurso de estandarización del NIST.

Nivel	**Descripción** (Tan difícil de romper como búsqueda de...)
Nivel 1	Clave sobre cifrador bloque con clave de 128 bits (p.e. AES128)
Nivel 2	Colisiones sobre función hash de 256 bits (p.e. SHA3-256)
Nivel 3	Clave sobre cifrador bloque con clave de 192 bits (p.e. AES192)
Nivel 4	Colisiones sobre función hash de 384 bits (p.e. SHA3-384)
Nivel 5	Clave sobre cifrador bloque con clave de 256 bits (p.e. AES256)

2.3.1. KEM

La Tabla 2.3 muestra un resumen de las características de los KEM finalistas y candidatos alternativos del concurso del NIST.

A continuación, describimos los KEM finalistas. En concreto, introducimos en profundidad la propuesta llamada Kyber, que será la pieza básica para la construcción presentada en el Capítulo 3; y a Classic McEliece, NTRU y Saber, que se emplearán en el Capítulo 5.

2.3.1.1. Kyber

Kyber [22] es un KEM IND-CCA postcuántico que, junto la firma digital Dilithium, pertenece a la *suite* criptográfica *Cryptographic Suite for Algebraic Lattices (CRYSTALS)*.

Kyber proviene de un esquema de cifrado de clave pública IND-CPA llamado `Kyber.CPA`, que es convertido en un KEM IND-CCA mediante una transforma-

[2]Se han propuesto candidatos alternativos para, en caso de encontrar vulnerabilidades insalvables en los finalistas, poder tener propuestas viables bien estudiadas.

Tabla 2.3: Características de los KEM finalistas y candidatos alternativos del concurso del NIST.

Nombre	**Aproximación utilizada**	**Problema computacional**	**Estatus**
Classic McEliece	Códigos	Versión dual de Niederreiter del PKE de McEliece	Finalista
Kyber	Retículos	Module-LWE	Finalista
NTRU	Retículos	Problema NTRU	Finalista
Saber	Retículos	Module-Learning with Rounding	Finalista
BIKE	Códigos	Quasy-Cyclic-MDPC	Candidato alternativo
FrodoKEM	Retículos	LWE	Candidato alternativo
HQC	Códigos	Hamming Quasi-Cyclic	Candidato alternativo
NTRU Prime	Retículos	Problema NTRU	Candidato alternativo
SIKE	Isogenias	Supersingular walk problem	Candidato alternativo

ción similar a la de Fujisaki-Okamoto[3] [49], pero que puede usarse pese a que el esquema presenta descifrado imperfecto. A continuación, describimos y trabajaremos con una versión modificada de ese esquema, también propuesto en [22] llamada `Kyber.CPA′`.

En primer lugar, introduciremos algunas definiciones necesarias para entender cómo se ha construido el PKE. La Tabla 2.4 resume la notación utilizada. Después, escribimos los algoritmos de generación de claves, de cifrado y de descifrado usados en `Kyber.CPA′`, argumentando la seguridad IND-CPA de que el problema Module-LWE (Definición 2.2) es difícil.

Denotaremos con R al anillo $\mathbb{Z}[X]/(X^n+1)$ y por R_q al anillo $\mathbb{Z}_q[X]/(X^n+1)$, donde $n = 2^{n'-1}$ tal que $X^n + 1$ es el $2^{n'}$-ésimo polinomio ciclotómico. Como en [22], fijamos los valores de n, n' y q a 256, 9 y 7681.

Para algún entero positivo η, se define la distribución binomial centrada como sigue B_η [22]:

$$\text{Muestrear } \{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow (\{0,1\}^2)^\eta$$

$$\text{y devolver como salida } \sum_{i=1}^{\eta}(a_i - b_i).$$

El problema de seguridad subyacente a `Kyber.CPA′` se basa en la dificultad del problema Module-LWE, que generaliza el problema *Learning with Errors* (LWE).

Definición 2.2 (Module-LWE [22])**.** *El problema Module-LWE consiste en distinguir muestras uniformes $(\mathbf{a}_i, b_i) \leftarrow R_q^k \times R_q$ de muestras $(\mathbf{a}_i, \mathbf{a}_i^T\mathbf{s} + e_i) \in R_q^k \times R_q$ donde $\mathbf{a}_i \leftarrow R_q^k$ es uniforme, $\mathbf{s} \leftarrow \beta_\eta^k$ común a todas las muestras, y $e_i \leftarrow \beta_n$ es nueva para cada muestra. La ventaja de un adversario $\mathcal{A}$ se define como*

$$\mathsf{Adv}^{mlwe}_{m,k,\eta}(\mathcal{A}) = \left| \Pr\left[b' = 1 : \begin{array}{l} \mathbf{A} \leftarrow R_q^{m\times k}; \\ (\mathbf{s},\mathbf{e}) \leftarrow \beta_\eta^k \times \beta_\eta^m; \\ \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}; \\ b' = \mathcal{A}(\mathbf{A}, \mathbf{b}) \end{array} \right] - \Pr\left[b' = 1 : \begin{array}{l} \mathbf{A} \leftarrow R_q^{m\times k}; \\ \mathbf{b} \leftarrow R_q^m; \\ b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b}) \end{array} \right] \right|.$$

La hipótesis Module-LWE establece que la ventaja anterior es despreciable para cualquier adversario $\mathcal{A}$.

Los autores de Kyber definen en [22] la función $\mathtt{Compress}_q(x, d)$ que toma un elemento $x \in \mathbb{Z}_q$ y devuelve un entero en $\{0, 1, \ldots, 2^d - 1\}$, donde $d <$

[3] La transformación de Fujisaki-Okamoto convierte un PKE IND-CPA en un PKE IND-CCA, seguro en el *Random Oracle Model*.

Tabla 2.4: Notación usada en `Kyber.CPA′`.

Notación	**Representación**
$y \sim S := \mathtt{Sam}(x)$ donde `Sam` es una eXtendable Output Function (XOF).	Valor y que está distribuido de acuerdo a la distribución S (o uniformemente sobre el conjunto S). Es un procedimiento determinista.
$v \leftarrow \beta_\eta$, $\mathbf{v} \leftarrow \beta_\eta^k$	$v \in R$ es generado a partir de una distribución donde cada uno de sus coeficientes son generados de B_η. Un vector k-dimensional de polinomios $\mathbf{v} \in R^k$ puede ser generado de acuerdo a una distribución β_η^k.
$\lceil \cdot \rfloor$	$\lceil \cdot \rfloor$ es la función redondeo, es decir, $\lceil x \rfloor = \left\lfloor x + \frac{1}{2} \right\rfloor$ donde $x \in \mathbb{Q}$ y $\lfloor \cdot \rfloor$ es la función suelo.
$r' = r \bmod^{\pm} \alpha$	Para cada número entero par (respectivamente, impar) α, $r' = r \bmod^{\pm} \alpha$ es el único elemento r' en el rango $-\frac{\alpha}{2} < r \leq \frac{\alpha}{2}$ (respectivamente, $-\frac{\alpha-1}{2} < r \leq \frac{\alpha+1}{2}$) tal que $r' = r$ mód α.

$\lceil \log_2(q) \rceil$. Es más, se define la función $\texttt{Decompress}_q$ tal que

$$x' = \texttt{Decompress}_q(\texttt{Compress}_q(x, d), d)$$

es un elemento cercano a x. Más específicamente,

$$|x' - x \bmod^{\pm} q| \leq \left\lceil \frac{q}{2^{d+1}} \right\rfloor.$$

Las funciones que satisfacen estos requisitos se definen en [22] como:

$$\begin{aligned} \texttt{Compress}_q(x, d) &= \lceil (2^d/q) \cdot x \rfloor \mod 2^d, \\ \texttt{Decompress}_q(x, d) &= \lceil (q/2^d) \cdot x \rfloor. \end{aligned}$$

El esquema $\texttt{Kyber.CPA}'$ = (KeyGen, Enc, Dec) está parametrizado por los enteros positivos k, d_u, y d_v. Los valores de estos parámetros varían para los diferentes niveles de seguridad. El espacio de mensajes es $\mathcal{M} = \{0,1\}^n$ y el espacio de textos cifrados es de la forma $(\mathbf{u}, v) \in \{0,1\}^{n\,k\,d_u} \times \{0,1\}^{n\,d_v}$. La definición de los algoritmos de generación de claves, cifrado y descifrado vienen dados por los Algoritmos 2.1–2.3 tal y como se definen en [22]. A diferencia de $\texttt{Kyber.CPA}'$, el esquema PKE no modificado $\texttt{Kyber.CPA}$ comprime **t** en la línea 4 del Algoritmo 2.1 y, por tanto, se debe descomprimir **t** en Algoritmo 2.2. El esquema $\texttt{Kyber.CPA}'$ será el punto de partida en los capítulos posteriores.

Algoritmo 2.1: $\texttt{Kyber.CPA}'\texttt{.KeyGen()}$

$\rho, \sigma \leftarrow \{0,1\}^n$
$\mathbf{A} \sim R_q^{k \times k} := \texttt{Sam}(\rho)$
$(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \texttt{Sam}(\sigma)$
$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$
devolver $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algoritmo 2.2: $\texttt{Kyber.CPA}'\texttt{.Enc}(pk = (\mathbf{t}, \rho), m \in \mathcal{M}))$

$r \leftarrow \{0,1\}^n$
$\mathbf{A} \sim R_q^{k \times k} := \texttt{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$
$\mathbf{u} := \texttt{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
$v := \texttt{Compress}_q(\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
devolver $c := (\mathbf{u}, v)$

Algoritmo 2.3: $\texttt{Kyber.CPA}'\texttt{.Dec}(sk = \mathbf{s}, c = (\mathbf{u}, v))$

$\mathbf{u} := \texttt{Decompress}_q(\mathbf{u}, d_u)$
$v := \texttt{Decompress}_q(v, d_v)$
devolver $\texttt{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

Kyber.CPA′ se demuestra que es seguro IND-CPA bajo la hipótesis de la dificultad del problema Module-LWE [22]. Este resultado se formaliza en el siguiente teorema.

Teorema 2.1 ([22]). *Para cualquier adversario $\mathcal{A}$ contra la seguridad CPA de* Kyber.CPA′*, se define la ventaja*

$$Adv^{cpa}_{\mathsf{Kyber.CPA}'}(\mathcal{A}) = \left| \Pr \left[b = b' : \begin{array}{l} (pk,\, sk) \leftarrow \mathsf{KeyGen}(); \\ (m_0,\, m_1,\, s) \leftarrow \mathcal{A}(pk); \\ b \leftarrow \{0,1\};\ c^* \leftarrow \mathsf{Enc}(pk,\, m_b); \\ b' \leftarrow \mathcal{A}(s,\, c^*) \end{array} \right] - \frac{1}{2} \right|.$$

Entonces, existe un adversario $\mathcal{B}$ tal que

$$Adv^{cpa}_{\mathsf{Kyber.CPA}'}(\mathcal{A}) \leq 2\, Adv^{mlwe}_{k+1,k,\eta}(\mathcal{B}).$$

Por último, vale la pena destacar que ni Kyber.CPA ni Kyber.CPA′ tienen corrección perfecta, es decir, hay una pequeña probabilidad de que existan errores al descifrar.

Definición 2.3 ([61]). *Un PKE se dice que es $(1-\delta)$-correcto si*

$$\mathbf{E}\left[\max_{m \in \mathcal{M}} \Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m\right]\right] > 1 - \delta,$$

donde la esperanza se toma sobre $(pk, sk) \leftarrow \mathsf{KeyGen}()$ y la probabilidad se toma sobre el espacio el espacio de aleatoriedades de Enc*.*

Siguiendo la prueba del Teorema 1 de [22], no es difícil probar el siguiente teorema, que provee un valor δ para Kyber.CPA′.

Teorema 2.2 ([22]). *Sea k un entero positivo. Sea $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2$ variables aleatorias que tienen la misma distribución que en los Algoritmos 2.1 y 2.2. Además, sean $\mathbf{c}_u \leftarrow \psi^k_{d_u}$, $\mathbf{c}_v \leftarrow \psi^k_{d_v}$ de acuerdo a la distribución ψ definida como sigue:*
Sea ψ^k_d la siguiente distribución sobre R:

1. *Elegir uniformemente al azar $\mathbf{y} \leftarrow R^k$*
2. ***devolver*** $(y - \mathsf{Descompress}_q(\mathsf{Compress}_q(\mathbf{y}, d), d)) \bmod^{\pm} q$

Denotaremos

$$\delta = \Pr\left[\mathrm{Y}\mathbf{e}^T\mathbf{r} + e_2 + \mathbf{c}_v - \mathbf{s}^T\mathbf{e}_1 - \mathbf{s}^T\mathbf{c}_u\mathrm{Y}_\infty \geq \lceil q/4 \rfloor\right],$$

donde, para $w = w_0 + w_1\, X + \cdots + w_{n-1}\, X^{n-1} \in R$*:*

$$||w||_\infty = \max_i |w_i \bmod^{\pm} q|,$$

y, similarmente, para $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$*:*

$$||\mathbf{w}||_\infty = \max_i ||w_i||_\infty.$$

Entonces, el esquema modificado $\mathsf{Kyber.CPA}'$ *es* $(1-\delta)$*-correcto.*

El KEM IND-CCA, $\mathsf{Kyber} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ viene dado por los Algoritmos 2.1, 2.4 y 2.5, donde KeyGen es el mismo algoritmo que $\mathsf{Kyber.CPA}'$ con la diferencia de que sk también contiene a $pk = (\mathbf{t}, \rho)$ y un secreto aleatorio z de 256 bits. $G\colon \{0,1\}^* \to \{0,1\}^{2\cdot 256}$ y $H\colon \{0,1\}^* \to \{0,1\}^{256}$ son dos funciones *hash*.

Algoritmo 2.4: $\mathsf{Kyber.Encaps}(pk = (\mathbf{t}, \rho))$

$m \leftarrow \{0,1\}^{256}$
$(K', r) := G(H(pk), m)$
$(\mathbf{u}, v) := \mathsf{Kyber.CPA.Enc}((\mathbf{t}, \rho), m; r)$
$c := (\mathbf{u}, v)$
$K := H(K', H(c))$
devolver (c, K)

El siguiente teorema demuestra la seguridad IND-CCA de Kyber cuando H y G están modelados como oráculos aleatorios.

Teorema 2.3 ([22])**.** *Para cualquier adversario* $\mathcal{A}$ *que realiza como máximo* q_{RO} *peticiones a los oráculos aleatorios* H *y* G *y,* q_D *peticiones al oráculo de descifrado, existe un adversario* $\mathcal{B}$ *tal que*

$$\mathsf{Adv}^{cca}_{\mathsf{Kyber}}(\mathcal{A}) \leq 3\, \mathsf{Adv}^{cpa}_{\mathsf{Kyber.CPA}}(\mathcal{B}) + q_{RO} \cdot \delta + \frac{3q_{RO}}{2^{256}}.$$

En [22], también se da una cota que es no ajustada (*non tight*) en el QROM, que sólo sirve de cota asintótica, que se formaliza en el siguiente teorema.

Teorema 2.4 ([22])**.** *Para cualquier adversario* $\mathcal{A}$ *que realiza como máximo* q_{RO} *llamadas a los oráculos aleatorios cuánticos* H *y* G *y,* q_D *llamadas (clásicas) al oráculo de descifrado, existe un adversario* $\mathcal{B}$ *tal que*

$$\mathsf{Adv}^{cca}_{\mathsf{Kyber}}(\mathcal{A}) \leq 8q^2_{RO} \cdot \delta + 4q_{RO}\sqrt{\mathsf{Adv}^{pr}_{\mathsf{Kyber.CPA}}(\mathcal{B})}.$$

Algoritmo 2.5: `Kyber.Decaps`$(sk = (\mathbf{s}, z, \mathbf{t}, \rho), c = (\mathbf{u}, v))$

$m' \leftarrow$ `Kyber.CPA.Dec`$(\mathbf{s}, (\mathbf{u}, v)$
$(K', r') := G(H(pk), m')$
$(\mathbf{u}', v') :=$ `Kyber.CPA.Dec`$((\mathbf{t}, \rho), m'; r')$
si $(\mathbf{u}', v') = (\mathbf{u}, v)$ **entonces**
  devolver $K := H(K', H(c))$
en otro caso
  devolver $K := H(z, H(c))$

Tabla 2.5: Instanciaciones de Kyber.

Instanciación	Nivel de seguridad
Kyber512	1
Kyber768	3
Kyber1024	5

La Tabla 2.5 muestra las instanciaciones de `Kyber` y su nivel de seguridad.

Kyber destaca por su versatilidad y sus autores subrayan que, a partir del KEM IND-CCA, se puede obtener un PKE IND-CCA y un AKE 2-parte.

El PKE IND-CCA se puede construir mediante la construcción KEM/DEM (Sección 1.3), usando `Kyber` como KEM y como DEM cualquier esquema simétrico definido como en [32]. Los autores de Kyber destacan que ejemplos de esto último pueden ser AES256-GCM, AES256-OCB o Chacha20-Poly1305. El PKE obtenido recibe el nombre de `Kyber.PKE` $=$ (`KeyGen`, `Enc`, `Dec`) (Algoritmos 2.6–2.8), donde E y D son, respectivamente, el algoritmo de cifrado y descifrado del esquema simétrico.

Algoritmo 2.6: `Kyber.PKE.KeyGen()`

$(pk, sk) \leftarrow$ `Kyber.KeyGen()`
devolver (pk, sk)

Algoritmo 2.7: `Kyber.PKE.Enc`(pk, m)

$(c, K) \leftarrow$ `Kyber.Encaps`(pk)
$c' := E(K, m)$
devolver $c'' := (c, c')$

Algoritmo 2.8: `Kyber.PKE.Dec`$(sk, c'' = (c, c'))$

$K :=$ `Kyber.Decaps`(sk, c)
devolver $m := D(K, c')$

El AKE 2-parte que se obtiene de `Kyber` sólo requiere el intercambio de 2 mensajes con la estructura que se muestra en la Figura 2.2. El AKE recibe el nombre de `Kyber.AKE` $=$ (`Init`, `AlgB`, `AlgA`) y viene dado por los Algoritmos 2.9–2.11. La seguridad de `Kyber.AKE` se demuestra en el modelo de Canetti–Krawczyk [28] e incluye la propiedad *weak forward secrecy* [22]. Esta propiedad indica que, cuando las claves a largo plazo de los participantes se ven comprometidas, se asegura el secreto de las claves de sesión previamente establecidas, pero sólo en las sesiones en las que el adversario no ha intervenido de manera activa.

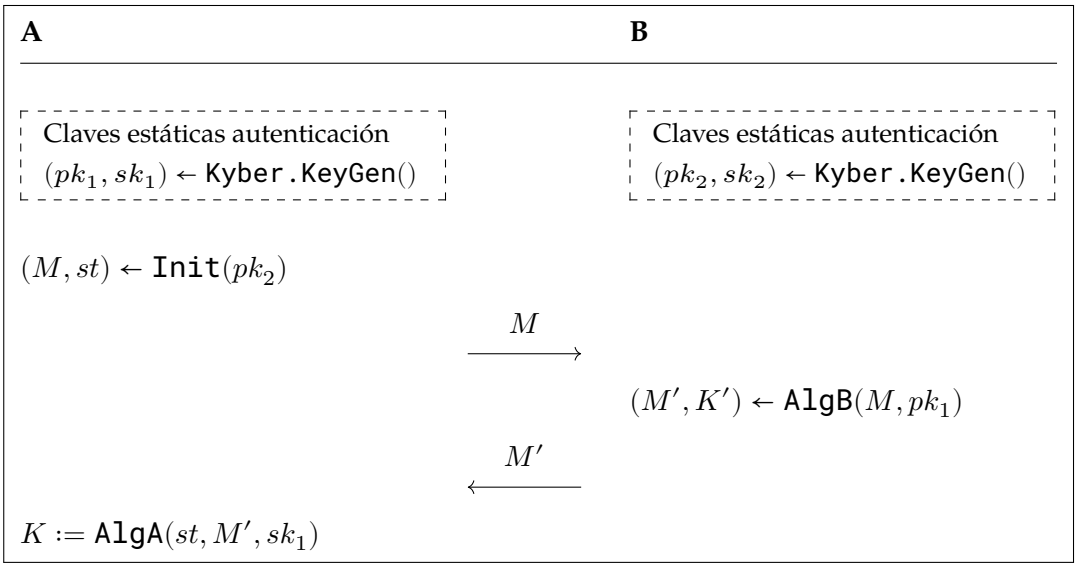


Figura 2.2: Estructura del AKE 2-parte propuesto por Kyber.

Algoritmo 2.9: `Init`(pk_2)

$(pk, sk) \leftarrow$ `Kyber.KeyGen`()
$(c_2, K_2) \leftarrow$ `Kyber.Encaps`(pk_2)
devolver $(M := (pk, c_2), st := (sk, K_2))$

Algoritmo 2.10: $\mathtt{AlgB}(M, pk_1)$

$(pk, c_2) := M$
$(C, K) \leftarrow \mathtt{Kyber.Encaps}(pk)$
$(c_1, K_1) \leftarrow \mathtt{Kyber.Encaps}(pk_1)$
$K_2' := \mathtt{Kyber.Decaps}(sk_2, c_2)$
$K := \mathsf{H}(K, K_1, K_2')$
devolver $(K, M' := (c, c_1))$

Algoritmo 2.11: $\mathtt{AlgA}(st, M', sk_1)$

$(c, c_1) := M'$
$(sk, K_2) := st$
$K' := \mathtt{Kyber.Decaps}(sk, c)$
$K_1' := \mathtt{Kyber.Decaps}(sk_1, c_1)$
$K := \mathsf{H}(K', K_1', K_2)$
devolver K

2.3.1.2. NTRU

NTRU es otro KEM finalista del concurso de estandarización del NIST basado en retículos. Es una mezcla de dos candidatos: NTRUEncrypt y NTRU-HRSS-KEM [89], ambos basados en el criptosistema NTRU [60]. Aunque la versión original de NTRU era un PKE probabilístico parcialmente correcto (sin errores de descifrado), en la Ronda 3, comienza definiendo un PKE determinista y correcto, que se asume que es seguro OW-CPA. Entonces, un KEM IND-CCA se obtiene haciendo pequeñas modificaciones a la variante Saito-Xagawa-Yamakawa de NTRU-HRSS-KEM [109]. Los autores de NTRU hacen dos estimaciones diferentes del nivel de seguridad en cada instanciación del KEM, dependiendo de si el modelo de computación es local o no local. Los detalles de estos modelos y la motivación para diferenciar los modelos de seguridad se pueden encontrar en los documentos subidos por el equipo diseñador durante la Ronda 3 [90]. Los nombres de las instanciaciones y el nivel de seguridad de cada una de ellas se puede ver en la Tabla 2.6.

2.3.1.3. Saber

Saber es un KEM basado en retículos similar a Kyber, en el sentido de que sus autores presentan en la Ronda 3 del concurso del NIST un PKE IND-CPA y aplican una transformación similar a la de Fujisaki-Okamoto [49] para obtener un KEM IND-CCA. Además, el PKE tiene un error de descifrado despreciable

Tabla 2.6: Instanciaciones de NTRU.

Instanciación	Nivel de seguridad (Modelo no local)	Nivel de seguridad (Modelo local)
NTRU-HPS-2048-509	-	1
NTRU-HRSS-701	1	3
NTRU-HPS-2048-677	1	3
NTRU-HPS-4096-821	3	5

y la seguridad se reduce a un problema de retículos, en este caso, al problema *Module Learning With Rounding* (MLWR). Las instanciaciones de Saber y su nivel de seguridad se puede encontrar en la Tabla 2.7.

Tabla 2.7: Instanciaciones de Saber.

Instanciación	Nivel de seguridad
LightSaber-KEM	1
Saber-KEM	3
FireSaber-KEM	5

2.3.1.4. Classic McEliece

Classic McEliece es el único KEM finalista del concurso de estandarización del NIST basado en códigos. El KEM se construye a partir de un PKE OW-CPA determinista llamado *versión dual de Niederreiter del PKE de McEliece*, que usa códigos binarios de Goppa [87]. La actualización de la Ronda 3 ha sido instanciado con 5 parámetros diferentes, cada uno con dos versiones, dependiendo de si la matriz de comprobación de paridad del código ha sido reducida a la forma sistemática o semisistemática. Las distintas instanciaciones y su nivel de seguridad se pueden ver en la Tabla 2.8.

2.3.2. Firma digital

Los esquemas de firma digital no serán el trabajo central de la tesis, pero por completitud, mostramos en la Figura 2.9 las características de los finalistas y candidatos alternativos del concurso del NIST. Además, con el fin de entender en profundidad este tipo de propuestas y sus posibles problemas de seguridad, describimos los ataques a un esquema de firma digital propuesto desde un área

Tabla 2.8: Instanciaciones de Classic McEliece.

Forma sistemática	**Forma semisistemática**	**Nivel de seguridad**
348864	348864f	1
460896	460896f	3
6688128	6688128f	5
6960119	6960119f	5
8192128	8192128f	5

Tabla 2.9: Características de los esquemas de firma digital finalistas y candidatos alternativos del concurso del NIST.

Nombre	**Aproximación utilizada**	**Problema computacional**	**Estatus**
Dilithium	Retículos	Problemas de retículos sobre módulos	Finalista
Falcon	Retículos	Problema NTRU	Finalista
Rainbow	Multivariante	Unbalanced oil and vinager	Finalista
GeMSS	Multivariante	Hidden Field Equations	Candidato alternativo
Picnic	Hashes	Función hash Cifrador en bloque	Candidato alternativo
SPHINCS+	Hashes	Función hash	Candidato alternativo

diferente a los típicos considerados. Este esquema se llama WalnutDSA y se basa en problemas sobre teoría de trenzas.

2.3.2.1. WalnutDSA

WalnutDSA [7] es un esquema de firma digital postcuántico EUF-CMA, que se presentó al concurso de estandarización del NIST, pero que no pasó de la Ronda 1 debido a varios ataques sufridos. Sus autores declararon que era seguro en el sentido EUF-CMA, una noción muy fuerte para esquemas de firma, comparable a IND-CCA en PKE. Este esquema de firma se basa en problemas asociados al grupo de trenzas [11], que es una propuesta que se sale de las aproximaciones habituales de la criptografía postcuántica. A continuación describimos brevemente los ataques sufridos por WalnutDSA. El lector interesado puede encontrar los detalles de los ataques en [45].

2.3.2.1.1. Ataques de factorización Estos ataques tienen por objetivo falsificar una firma para cualquier mensaje, resolviendo un problema de factorización en grupos.

En [58], Hart et al. presentaron un ataque para resolver este tipo de problemas de factorización, permitiendo construir una nueva firma válida a partir de varias firmas válidas, violando la seguridad EUF-CMA. Este ataque generaba firmas de tamaño mayor que las generadas por un usuario legítimo. El diseño de WalnutDSA se modificó para mitigar este ataque. Sin embargo, en [17], se presentó un método que explotaba propiedades de maleabilidad de WalnutDSA para falsificar firmas válidas. De nuevo, las firmas generadas eran varios órdenes de magnitud mayores que las firmas legítimas, por lo que los autores de WalnutDSA impusieron un límite de tamaño de las firmas en la implementación del esquema, haciendo inútil este ataque.

Otro ataque de factorización se encuentra en [80]. Esta vez, el ataque se basa en la llamada forma canónica de Garside [19, 93]. Este ataque no puede ser mitigado cambiando los parámetros del sistema. Se puede prevenir añadiendo *elementos ocultos* a la codificación del mensaje, pero esto repercute en la longitud y tiempo de cómputo de la firma. En [6], los autores de WalnutDSA implementaron el ataque y demostraron que usando los *elementos ocultos* se bloqueaba el ataque.

2.3.2.1.2. Ataques de colisiones En [17], observaron que si existían dos mensajes distintos m_1 y m_2 tal que $f(m_1) = f(m_2)$ para cierta función f definida en la verificación de la firma, entonces se generaría una firma válida tanto para m_1 como para m_2. El ataque se implementó para dicha f con el algoritmo de búsqueda de colisiones de van Oorschot y Wiener [91]. Con un ordenador corriente,

el algoritmo encontró una colisión con tan sólo $2^{32.2}$ evaluaciones de la función, requiriendo sólo una hora de tiempo. Los mensajes m_1 y m_2 encontrados fueron:

$$\begin{cases} m_1 = & \text{“I would like to receive 7181666883746416503 free samples of delicious cookies”.} \\ m_2 = & \text{“I pledge to donate 3519533052089988469 USD to Ward Beullens”.} \end{cases}$$

La mitigación de este ataque se puede hacer de dos formas:

- Aumentando el tamaño de los parámetros, haciendo que el tamaño de la clave pública sea 5 veces mayor y el esquema 25 veces más lento, en el nivel 5 de seguridad.
- Cambiar la codificación y realizar un cambio menor de los parámetros, haciendo la clave pública un 50 % mayor, la firma un 25 % más grande y el algoritmo de verificación 2 veces más lento.

2.3.2.1.3. Inversión de la E-Multiplication WalnutDSA basa su seguridad en la unidireccionalidad de una función llamada E-Multiplication. Sin embargo, en [17], encontraron dos métodos para invertir esta función: un ataque de colisiones basado en el ataque del cumpleaños y un ataque basado en la estructura subyacente de la función. Este segundo ataque es más eficiente que el primero. Ambos ataques se pueden mitigar aumentando el tamaño de los parámetros [6, 17].

2.3.2.1.4. Otros En [75], Kotov et al. presentaron un ataque heurístico sobre WalnutDSA, que funciona exclusivamente sobre las trenzas y no necesita tener en cuenta la función E-Multiplication, con un 100 % de éxito. El ataque funciona de la siguiente manera: un adversario recopila pares arbitrarios de mensajes y firmas válidas y, con ellos, es capaz de calcular una clave secreta alternativa tal que, cuando se usa para firmar cualquier mensaje, produce la misma firma que la clave secreta legítima.

Para mitigar este ataque, los autores de WalnutDSA, introdujeron un nuevo elemento en las firmas, que sólo aumentaba el tamaño de la misma en un 6.7 % [6]. Kotov et al. [31] cuestionaron esta contramedida, señalando que sus métodos estaban diseñados para lidiar con estos elementos y se podían modificar para tratar un nuevo aumento de estos.

2.3.3. Implementaciones

Las implementaciones de las propuestas del concurso de estandarización del NIST se pueden encontrar de diversas formas: la más sencilla es emplear el código proporcionado subido al concurso del NIST [88] al que todos los participantes

Tabla 2.10: Comparativa de las principales librerías de *software*. La interrogación indica que no existe información disponible.

Característica	LibOQS	PQClean	mupq	LibPQCrypto	PQSLib
¿Librería?	✔	✖	✔	✔	✔
Repositorio de implementaciones limpias	✖	✔	✖	✖	✖
Proyecto activo	✔	✔	✔	✖	?
Documentación	✔	✔	✔	✔	?
Implementación en C	✔	✔	✔	✔	✔
Soporte a otros lenguajes	C++ Go Java Python Rust .Net	✖	✖	Python	?
Soporte para hardware específico	✖	✖	Sí, soporte para ARM Cortex M3, M4 y RISC-V (en curso).	✖	?
Implementaciones optimizadas	✔	✔	✔	✔	?
¿Open source?	✔	✔	✔	✔	✖

están obligados a subir en cada ronda. Esta manera permite probar una determinada especificación de los esquemas. Como desventaja, no permite seguir los cambios periódicos que sufre el código en cuanto a *bugs*, optimizaciones, etc. Es por ello, que se puede encontrar el código de algunos de los finalistas en repositorios abiertos como GitHub. Así, es posible hacer un seguimiento detallado de los cambios en el código y obtener las últimas versiones del mismo. Por último, también se pueden encontrar librerías que agrupan los esquemas finalistas y permiten realizar pruebas sobre todos ellos. En este grupo se diferencian dos grupos: librerías de uso general y librerías para *hardware* específico.

Las librerías de uso general incluyen a LibPQCrypto [98], PQSLib [100], LibOQS [115] y PQClean [97]. La primera es *open source* y ha sido desarrollada en el proyecto PQCRYPTO [99], pero no se actualiza desde 2018 e incluye propuestas de las Rondas 1 y 2. La segunda es una librería ligera de algoritmos postcuánticos desarrollada por la empresa PQShield y soporta algunos de algoritmos finalistas del concurso del NIST, pero no es *open source*. Las dos últimas también son *open source* e implementan la mayoría de finalistas del NIST, pero PQClean está concebida como repositorio de implementaciones, es decir, contiene implementaciones específicas para aplicaciones concretas, mientras que LibOQS se concibe para realizar pruebas sobre de distintos protocolos con los diferentes esquemas

postcuánticos del NIST.

Entre las librerías para *hardware* específico, la más conocida es `pqm4` [85] que implementa los finalistas del concurso del NIST para ARM Cortex M4 [10]. En [71] se puede encontrar una comparativa del rendimiento de los candidatos de la Ronda 3 realizada con `pqm4`. La librería para testar los esquemas sobre ARM Cortex M3 [9] es `pqm3` [84], pero contiene menos implementaciones y está menos desarrollada que `pqm4`. También, es posible encontrar una librería para RISC-V [67] llamada `pqriscv`, pero en el momento de escribir estas líneas se encuentra en desarrollo. Todas ellas, se encuentran integradas dentro del proyecto `mupq` [83]. Una comparativa de las principales características de todas estas librerías se puede ver la Tabla 2.10.

En el ámbito del *hardware* también se han propuesto implementaciones, pero estas son menos numerosas debido a algunos retos no resueltos o sólo parcialmente resueltos entre los que se encuentran esquemas no pensados para ser fácilmente implementables en *hardware*, esquemas de diferentes familias (retículos, códigos, isogenias, etc.), piezas nunca antes implementadas, tamaños de clave muy grandes en algunos casos, etc. Los detalles se pueden encontrar en [124]. Una comparativa de las propuestas de la Ronda 2 se pueden encontrar en [13, 114]. Algunas implementaciones *hardware* incluyen: NTRU en Ronda 3 [101], propuestas de retículos como Kyber, Saber y FrodoKEM en Rondas 2 y 3 [34, 35, 65], implementación de isogenias [76] y Lightweight McEliece, que es una modificación de Classic McEliece de la Ronda 2 que permite ejecutarse en *hardware* más eficientemente [4].

Un excelente resumen de los ataques más comunes a implementaciones se puede encontrar en [65].

2.4. Otras iniciativas de estandarización

El concurso del NIST no es el único concurso desarrollado para estandarizar herramientas postcuánticas.

En China, también se ha llevado a cabo un concurso similar, que comenzó en 2018, que buscaba nuevos esquemas de firma digital y PKE/KEM [40]. Ya se conocen los ganadores del mismo: `LAC.PKE` [77] como PKE[4] basado en RLWE y la *suite* `Aigis` [129], que incluye cifrado (`Aigis-enc`) y firma digital (`Aigis-sig`), basada en retículos e inspirados por Kyber y Dilithium, respectivamente.

En Rusia, el proceso de estandarización comenzó en 2020. Se perseguía estandarizar esquemas de firma digital e intercambio de clave. Se consideran las siguientes propuestas [65]:

[4]Esta propuesta también se presentó al concurso del NIST y no pasó de la Ronda 2, por sufrir ciertas vulnerabilidades. Los detalles se pueden consultar en [56].

- Firmas basadas en *hashes*, códigos y retículos.
- Intercambio de clave basado en isogenias llamado `Forsythia`.

El concurso ruso no es una competición al uso, como en EE.UU. y China, sino que, en él, todas las propuestas que cumplan los requisitos de seguridad serán probablemente aceptadas. Una vez lo estén, se integrarán en los estándares gubernamentales GOST [65].

Otros planes de otros países y grupos de trabajo de distintas organizaciones se pueden consultar en [65].

Capítulo

3

Un nuevo protocolo de intercambio de clave en grupo postcuántico basado en Kyber

Este capítulo es el núcleo de la tesis, del que se derivan los capítulos posteriores. Aquí construimos un protocolo autenticado de intercambio de clave en grupo (GAKE, por sus siglas en inglés) postcuántico y demostramos su seguridad en el modelo de seguridad QROM, un modelo de seguridad más fuerte que el típico considerado ROM. Nuestro diseño se basa en dos primitivas: el compilador de Abdalla et al. y Kyber, finalista del concurso de estandarización del NIST. Además, este diseño es el primer protocolo de este tipo que emplea uno de los finalistas del NIST.

Los protocolos de intercambio de clave en grupo (GKE, por sus siglas en inglés) son una construcción fundamental: tienen aplicaciones en numerosos ámbitos del mundo real, sobre todo en los que involucran comunicaciones entre grupos de personas o máquinas, que requieran confidencialidad en las mismas. Los protocolos GKE permiten a un grupo de $n \geq 2$ entidades que interactúan a través de un canal inseguro, establecer una clave común de alta entropía que puede ser empleada para establecer comunicaciones posteriores entre las entidades. En la mayoría de casos, una vez que la clave común es acordada entre

todas las partes, se emplea criptografía simétrica para lograr confidencialidad, y por tanto, todos los participantes en el protocolo pueden comunicarse de forma segura entre ellos.

Los protocolos GKE generalizan a los protocolos de intercambio de clave 2-parte y evitan el uso de estos, ya que establecer claves diferentes de sesión para cada par de participantes obligaría a cada participante a almacenar un gran número de claves. Además, cada mensaje destinado a todo el grupo debería cifrarse $n-1$ veces con claves diferentes, mientras que los protocolos GKE permiten transmitir los mensajes de modo que todos los usuarios procesen estos de la misma forma. Sin embargo, es posible que no haya forma de evaluar el origen y la integridad de los mensajes. En este caso, cuando no se dispone de canales autenticados, los protocolos que persiguen este objetivo se complican mucho más y a menudo tienen que depender de una infraestructura de clave pública (PKI, por sus siglas en inglés) externa para poder autenticar a los miembros legítimos del grupo, lo que suele añadir un coste significativo a las construcciones. Estos se denominan protocolos *Group Authenticated Key Exchange* o abreviadamente como GAKE (ver detalles en la Sección 1.2.5).

Nuestro protocolo GAKE postcuántico seguro en el QROM de 4 rondas se construye a partir de las primitivas criptográficas proporcionadas por el KEM postcuántico Kyber [22] (Sección 2.3.1.1). Más concretamente, es un protocolo compilado que utiliza el compilador de Abdalla et al. [1] como base (Sección 1.2.5.2.1). A partir de los resultados de [63, 64], construimos un esquema de compromiso y un intercambio de clave 2-parte, que pueden ser obtenidos del esquema de clave pública IND-CPA Kyber.CPA′. Para que todas las piezas encajen, demostramos que dicho PKE es seguro en el QROM. Este es el primer GAKE que emplea un finalista del concurso del NIST que provee de seguridad postcuántica basado solamente en la hipótesis Module-LWE, heredada de Kyber. Además, su diseño evita el uso de las costosas firmas postcuánticas. La Figura 3.1 resume los pasos necesarios para obtener el protocolo propuesto.

Kyber [22] es un KEM basado en retículos cuya seguridad depende de la hipótesis Module-LWE (Definición 2.2), que se asume que es postcuántica. Nuestro GAKE hereda, por tanto, esta hipótesis. El AKE 2-parte y el esquema de compromiso se derivan del PKE IND-CPA de [22] denominado Kyber.CPA′. El AKE 2-parte, que en nuestra construcción recibe el nombre de Kyber$^{\pm}$.2AKE es el resultado de aplicar la transformación $\mathsf{FO}_{\mathsf{AKE}}$ de [63, 64] a Kyber.CPA′. Finalmente, un esquema de compromiso se puede conseguir de cualquier PKE IND-CCA, como se apunta en [1]. En nuestra construcción, convertimos Kyber.CPA′ en un KEM IND-CCA aplicando la $\mathsf{FO}^{\pm}_m$ de [63, 64] logrando Kyber$^{\pm}$. Este es es transformado en un PKE IND-CCA, que llamamos Kyber$^{\pm}$.PKE como resultado de aplicar la transformación de [32]. Los detalles de cómo está construido la versión

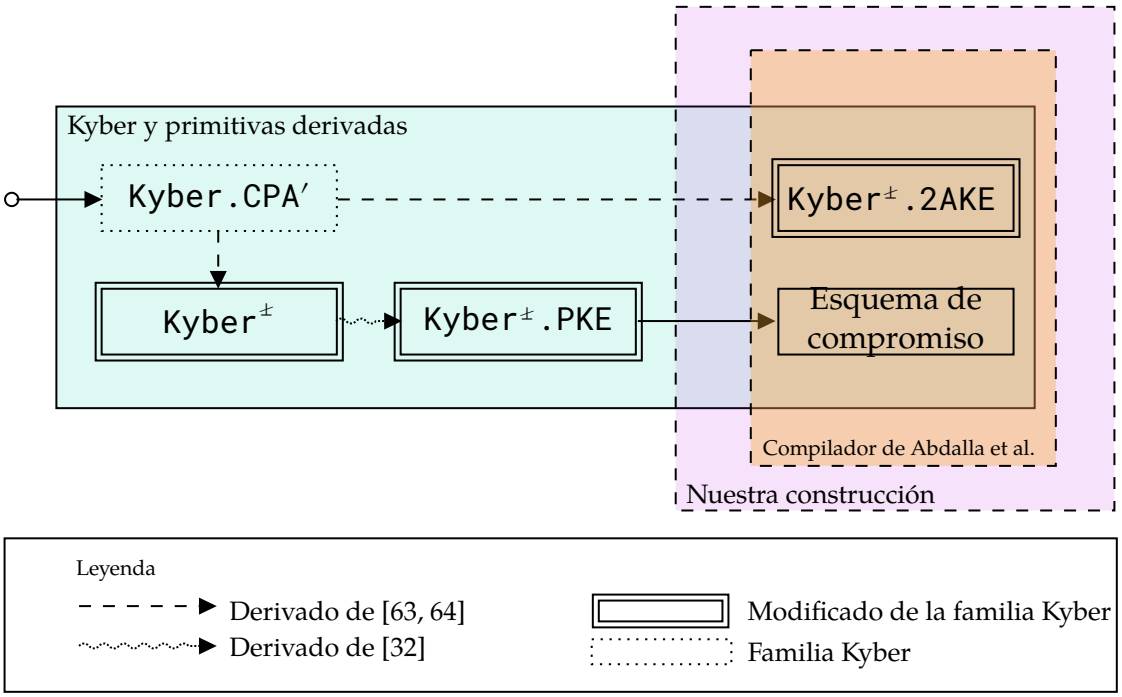


Figura 3.1: Vista general de nuestra construcción.

original de Kyber se puede encontrar en la Sección 2.3.1.1.

3.1. Primitivas postcuánticas

En esta sección, describimos las primitivas postcuánticas usadas en la construcción del protocolo GAKE; concretamente un protocolo de intercambio de clave (AKE) y un esquema de compromiso. Las relaciones entre ellas se resumen en la Figura 3.2.

En la Sección 3.1.1, detallamos cómo se obtiene el esquema $\mathtt{Kyber}^{\pm}\mathtt{.2AKE}$ usando una transformación genérica propuesta en [63, 64]. Para ello, se parte de un esquema PKE que posee las propiedades de *Disjoint Simulatability*, abreviado como DS (Definición 3.1) y seguridad IND-CPA. El esquema resultante es un AKE de dos mensajes que se demuestra seguro en el QROM. En particular, usamos una versión ligeramente modificada llamada $\mathtt{Kyber.CPA}'$ de un PKE IND-CPA introducido en [22] como parte de Kyber. Describimos la transformación $\mathtt{FO}_{\mathtt{AKE}}$ que convierte un PKE IND-CPA en un AKE seguro. La sección acaba probando que $\mathtt{Kyber.CPA}'$ cumple la propiedad de DS y, por tanto, es posible construir un AKE seguro en el QROM aplicando $\mathtt{FO}_{\mathtt{AKE}}$ a este esquema.

La Sección 3.1.2 se dedica a construir el esquema de compromiso postcuán-

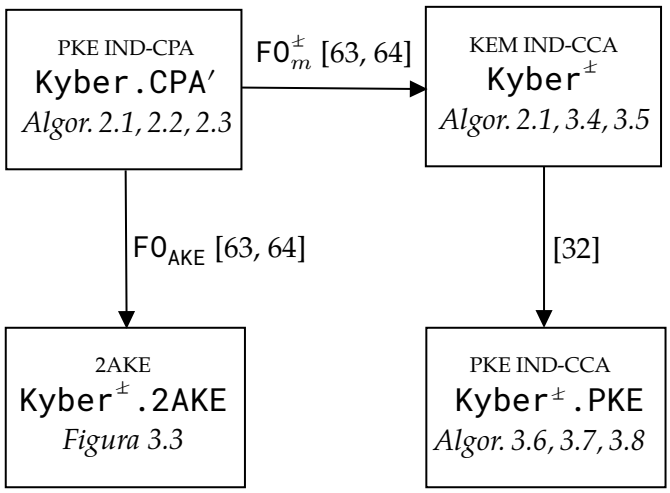


Figura 3.2: Kyber y herramientas derivadas.

tico necesario para el compilador de Abdalla et al. [1]. Debe ser no interactivo y no maleable, *perfectly binding* y no maleable para múltiples compromisos [41]. Como se indica en [1], este tipo de esquemas de compromiso se puede construir a partir de cualquier esquema de clave pública IND-CCA. Para este fin, usamos otra transformación descrita en [63, 64], concretamente FO$_m^{\pm}$, que convierte un PKE IND-CPA en un KEM IND-CCA. A partir de este, se utiliza la aproximación KEM/DEM (Sección 1.3) para obtener el deseado PKE IND-CCA. Combinando todo lo anterior, obtenemos el deseado esquema de compromiso a partir de Kyber.CPA′, la misma primitiva empleada como punto de partida del AKE 2-parte.

3.1.1. Transformación FO$_{\mathsf{AKE}}$: de PKE a AKE

La transformación FO$_{\mathsf{AKE}}$ es una construcción genérica propuesta en [63, 64], que transforma un esquema de clave pública (PKE) IND-CPA en un protocolo AKE, que se demuestra seguro en el QROM. La construcción admite que el PKE no sea perfectamente correcto, con lo que puede aplicarse para el esquema Kyber.CPA′ que se introdujo en la Sección 2.3.1.1. Otra propiedad interesante de la transformación es que evita el uso de las costosas firmas postcuánticas. FO$_{\mathsf{AKE}}$ puede ser vista como una extensión de la transformación de Fujisaki–Okamoto [49], que convierte un esquema PKE IND-CPA en uno IND-CCA.

El AKE 2-parte resultante de aplicar la transformación FO$_{\mathsf{AKE}}$ es bastante eficiente en términos de comunicación. En [63, 64], se le llama protocolo de 2 mensajes, que significa que es un protocolo AKE de 2 rondas en el una parte envía un mensaje en la primera ronda mientras que la otra parte responde con otro

mensaje en la segunda ronda. Como una contribución interesante de [63, 64], los autores definieron un modelo de seguridad y dos nociones de seguridad para los AKE de 2 mensajes: la indistinguibilidad de claves frente a ataques activos (IND-AA) y una noción llamada IND-StAA más débil de indisguibilidad frente a adversarios activos. Estamos interesados en la segunda noción, ya que la seguridad del AKE obtenido de aplicar la transformación $\mathsf{FO}_{\mathsf{AKE}}$ se demuestra seguro en [63, 64] en este modelo más débil. Esta noción es suficiente para nuestros propósitos porque IND-StAA implica la seguridad requerida por el compilador de Abdalla et al.

El modelo IND-StAA a alto nivel, formulado en [63, 64], es el siguiente: IND-StAA establece que la clave de sesión se mantiene indistinguible de una clave aleatoria incluso si:

1. El adversario conoce o bien, la clave (secreta) de larga duración o la información secreta de estado, que incluye, por ejemplo, claves efímeras generadas en la ejecución, pero no ambas, de las partes involucradas en la sesión atacada, siempre que no haya modificado algún mensaje recibido por la sesión atacada.

2. Si el adversario modificó algún mensaje recibido por la sesión atacada, siempre y cuando no haya obtenido ni la clave (secreta) de larga duración de algún usuario involucrado ni el estado de la sesión atacada.

Los autores de la transformación $\mathsf{FO}_{\mathsf{AKE}}$ demuestran la seguridad IND-StAA en el QROM siempre y cuando el PKE cumpla que es IND-CPA y que es posible muestrear textos cifrados falsos que son indistinguibles de textos cifrados reales, asumiendo que la probabilidad de que el algoritmo de muestreo encuentre un cifrado real es pequeña. Esta última propiedad recibe el nombre de *Disjoint Simulatability* (DS) de textos cifrados y se define en [63, 64] como sigue.

Definición 3.1 (Disjoint Simulatability (DS) [64])**.** *Sea* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *un PKE con espacio de mensajes* $\mathcal{M}$ *y espacio de textos cifrados* $\mathcal{C}$*, con un algoritmo adicional probabilístico en tiempo polinomial* $\overline{\mathsf{Enc}}$*. Para adversarios cuánticos* $\mathcal{A}$*, se define la ventaja contra la Disjoint Simulatability de* PKE *como*

$$\mathsf{Adv}^{\mathsf{DS}}_{\mathsf{PKE},\overline{\mathsf{Enc}}}(\mathcal{A}) = \left| \Pr\left[\begin{array}{l} pk \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda), \\ m \leftarrow \mathcal{M}, \\ c \leftarrow \mathsf{Enc}(pk, m) \end{array} : 1 \leftarrow \mathcal{A}(pk, c) \right] \right.$$
$$\left. - \Pr\left[\begin{array}{l} pk \leftarrow \mathsf{KeyGen}(\mathbf{1}^\lambda), \\ c \leftarrow \overline{\mathsf{Enc}}(pk) \end{array} : 1 \leftarrow \mathcal{A}(pk, c) \right] \right|.$$

Cuando no haya confusión, eliminaremos $\overline{\mathsf{Enc}}$ *del subíndice de la ventaja por conveniencia.*

Diremos que PKE *es* ε_{dis}*-disjunto si para todo* $pk \in supp(\mathsf{KeyGen})$*,*

$$Pr[c \leftarrow \overline{\mathsf{Enc}}(pk) \ : \ c \in \mathsf{Enc}(pk, \mathcal{M}; \mathcal{R})] \leq \varepsilon_{dis},$$

donde $\mathcal{R} = \mathcal{R}(pk)$ *es un espacio de aleatoriedades finito definido por* pk*.*

Los autores de la transformación $\mathsf{FO}_{\mathsf{AKE}}$ sugieren que muchos esquemas basados en retículos son DS de una forma natural: los textos cifrados falsos pueden ser muestreados uniformemente al azar de un conjunto que contiene a los textos cifrados válidos. DS debería obtenerse a partir de la hipótesis LWE, ya que las muestras LWE son relativamente dispersas y el muestreo uniforme debería ser disjunto.

En el siguiente teorema establecemos que la seguridad DS de $\mathsf{Kyber.CPA}'$ equipado con un algoritmo adicional $\overline{\mathsf{Enc}}$ se reduce a la seguridad Module-LWE.

Teorema 3.1 (Seguridad DS de $\mathsf{Kyber.CPA}'$)**.** *Sea* η*,* k*,* d_u*, y* d_v *enteros positivos y parámetros de* $\mathsf{Kyber.CPA}'$ *como en la Sección 2.3.1.1. Si* $\mathsf{Kyber.CPA}'$ *está equipado con un algoritmo probabilístico y en tiempo polinomial* $\overline{\mathsf{Enc}}$ *que muestrea un texto cifrado uniformemente al azar para cada clave pública. Entonces, para cualquier adversario* $\mathcal{A}$*, existe un adversario* $\mathcal{B}$ *tal que*

$$Adv^{DS}_{\mathsf{Kyber.CPA}'}(\mathcal{A}) \ \leq \ 2\, Adv^{mlwe}_{k+1,k,\eta}(\mathcal{B}).$$

Más aún, $\mathsf{Kyber.CPA}'$ *es* ε_{dis}-disjunto *con*

$$\varepsilon_{dis} = \frac{1}{2^{n\,(d_u\,k+d_v-2)}}.$$

Demostración. Sea $\mathcal{A}$ un adversario que ataca la seguridad DS de $\mathsf{Kyber.CPA}'$. Obtenemos una cota para $\mathsf{Adv}^{\mathsf{DS}}_{\mathsf{Kyber.CPA}'}(\mathcal{A})$ siguiendo la secuencia de juegos de la demostración del Teorema 2 de [22].

En primer lugar, el valor $\mathbf{t} := \mathbf{As} + \mathbf{e}$ que se usa en KeyGen se sustituye por un valor aleatorio elegido uniformemente al azar. Se sigue de la seguridad Module-LWE de $\mathsf{Kyber.CPA}'$ que el valor $\mathbf{t}$ y el valor uniformemente al azar son indistinguibles uno del otro. Después, los valores $\mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ y $\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m$ usado en la generación del texto cifrado es sustituido simultáneamente con valores elegidos uniformemente al azar. De nuevo, se sigue de la seguridad de la seguridad Module-LWE de $\mathsf{Kyber.CPA}'$ que $\mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ y $\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m$ son indistinguibles de valores al azar. Como en [22], deducimos que existe un adversario $\mathcal{B}$ con el mismo tiempo de ejecución que $\mathcal{A}$ tal que $\mathsf{Adv}^{\mathsf{DS}}_{\mathsf{Kyber.CPA}'}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathcal{B})$.

Para probar que $\mathsf{Kyber.CPA}'$ es ε_{dis}-*disjunto* con $\varepsilon_{\text{dis}} = 2^{n\,(2-d_u\,k-d_v)}$, recordamos que $\mathcal{M} = \{0,1\}^n$, $\mathcal{C} = \{0,1\}^{n\,k\,d_u} \times \{0,1\}^{n\,d_v}$, y $\mathcal{R} = \{0,1\}^n$ son los espacios de mensajes, de textos cifrados y aleatoriedades, respectivamente. Puesto que $|\mathsf{Enc}(pk,\, \mathcal{M};\, \mathcal{R})| \leq |\mathcal{M}|\,|\mathcal{R}| = 2^{2\,n}$, obtenemos

$$\begin{aligned}
\Pr[c \leftarrow \overline{\mathsf{Enc}} \,:\, c \in \mathsf{Enc}(pk,\, \mathcal{M};\, \mathcal{R})] \;&\leq \max_{(pk,\, sk) \in \mathsf{KeyGen}(\mathcal{R})} \frac{|\mathsf{Enc}(pk,\, \mathcal{M};\, \mathcal{R})|}{|\mathcal{C}|} \\
&\leq \frac{2^{2\,n}}{2^{n\,(d_u\,k+d_v)}} \\
&= \frac{1}{2^{n\,(d_u\,k+d_v-2)}},
\end{aligned}$$

que es resultado deseado. □

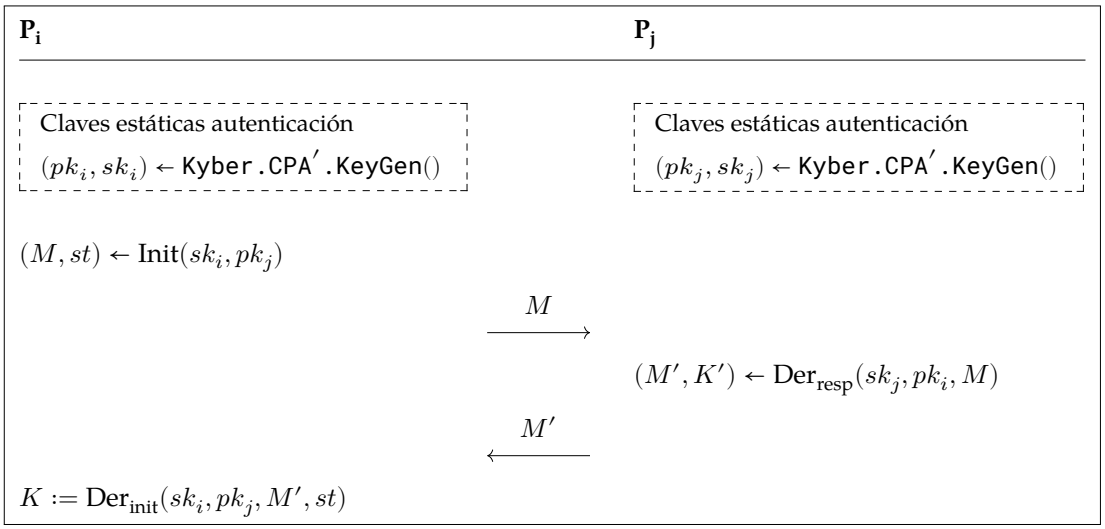


Figura 3.3: Intercambio de clave 2-parte seguro en el QROM.

Los Teoremas 2.1 y 3.1 garantizan que $\mathsf{Kyber.CPA}'$ satisfacen las hipótesis del Teorema 3.2 (que formulamos posteriormente), por lo que podemos usarlo para producir un AKE 2-parte que es seguro IND-StAA en el QROM. La estructura del esquema resultante se muestra en la Figura 3.3. El esquema, que llamaremos $\mathsf{Kyber}^{\pm}\mathsf{.2AKE}$, está compuesto por los Algoritmos 3.1, 3.2 y 3.3. Aquí, G y H son oráculos aleatorios y H'_R, H'_{L1}, H'_{L2}, y H'_{L3} son oráculos aleatorios internos que se pueden implementar con una función pseudoaleatoria. Nótese que este AKE 2-parte no es el mismo que el propuesto en la Sección 2.3.1.1, pero sí mantiene la misma estructura de 2 mensajes.

Algoritmo 3.1: $\mathtt{Init}(sk_i, pk_j)$

$m_j \xleftarrow{\$} \mathcal{M}$
$c_j := \mathsf{Kyber.CPA'.Enc}(pk_j, m_j; \mathsf{G}(m_j))$
$(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{Kyber.CPA'.KeyGen}()$
$M := (\tilde{pk}, c_j)$
$st := (\tilde{sk}, m_j, M)$
devolver (M, st);

Algoritmo 3.2: $\mathtt{Der}_{\text{resp}}(sk_j, pk_i, M)$

$(\tilde{pk}, c_j) := M$
$m_i, \tilde{m} \xleftarrow{\$} \mathcal{M}$
$c_i := \mathsf{Kyber.CPA'.Enc}(pk_i, m_i; \mathsf{G}(m_i))$
$\tilde{c} := \mathsf{Kyber.CPA'.Enc}(\tilde{pk}, \tilde{m}; \mathsf{G}(\tilde{m}))$
$M' := (c_i, \tilde{c})$
$m'_j := \mathsf{Kyber.CPA'.Dec}(sk_j, c_j)$
si $m'_j = \perp$ **o** $c_j \neq \mathsf{Kyber.CPA'.Enc}(pk_j, m'_j; \mathsf{G}(m'_j))$ **entonces**
  $K' := \mathsf{H}'_R(m_i, c_j, \tilde{m}, i, j, M, M')$
en otro caso
  $K' := \mathsf{H}(m_i, m'_j, \tilde{m}, i, j, M, M')$
devolver (M', K');

Algoritmo 3.3: $\mathsf{Der}_{\text{init}}(sk_i, pk_j, M', st)$

$(c_i, \tilde{c}) := M'$
$(\tilde{sk}, m_j, M) := st$
$m'_i := \mathsf{Kyber.CPA'.Dec}(sk_i, c_i)$
$\tilde{m}'_i := \mathsf{Kyber.CPA'.Dec}(\tilde{sk}, \tilde{c})$
si $m'_i = \perp$ **o** $c_i \neq \mathsf{Kyber.CPA'.Enc}(pk_i, m'_i; \mathsf{G}(m'_i))$ **entonces**
  si $\tilde{m}' = \perp$ **entonces**
    $K := \mathsf{H}'_{L1}(c_i, m_j, \tilde{c}, i, j, M, M')$
  en otro caso
    $K := \mathsf{H}'_{L2}(c_i, m_j, \tilde{m}', i, j, M, M')$
si no, si $\tilde{m}' = \perp$ **entonces**
  $K := \mathsf{H}'_{L3}(m'_i, m_j, \tilde{c}, i, j, M, M')$
en otro caso
  $K := \mathsf{H}(m'_i, m_j, \tilde{m}', i, j, M, M')$
devolver K;

Por completitud, a continuación presentamos el teorema de [64] que garantiza la seguridad de la transformación $\mathsf{Kyber}^{\pm}\mathsf{.2AKE}$ a partir de un PKE IND-CPA y DS en el QROM. Este establece que la seguridad IND-StAA de $\mathsf{AKE} = \mathsf{FO}_{\mathsf{AKE}}(\mathsf{PKE}, \mathsf{G}, \mathsf{H})$, donde PKE es un PKE y G, H son oráculos aleatorios, se reduce a la seguridad DS y IND-CPA de PKE en el QROM.

Teorema 3.2 ([64])**.** *Sea* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *un PKE* $(1-\delta)$*-correcto, que tiene un algoritmo de muestreo* $\overline{\mathsf{Enc}}$ *tal que es* ε*-disjunto. Sea* n *el número de partes, y se supone que cualquier adversario tiene acceso a un oráculo REVEAL que revela la clave de sesión (si ya está definida). Entonces, para cualquier adversario IND-StAA* $\mathcal{B}$ *que establece* S *sesiones y realiza como mucho* q_R *consultas (clásicas) a REVEAL, como mucho* q_{G} *consultas (cuánticas) al oráculo aleatorio* G*, y como mucho* q_{H} *consultas (cuánticas) al oráculo aleatorio* H*, existen adversarios* $\mathcal{A}_{DS}$ *y* $\mathcal{A}_{CPA}$ *contra* PKE *tal que*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}StAA}}_{\mathsf{AKE}}(\mathcal{B}) &\leq 2\,S\,(S+3\,n)\,\mathsf{Adv}^{\mathsf{DS}}_{\mathsf{PKE}}(\mathcal{A}_{DS}) \\
&\quad + 4\,S\,(S+3\,n)\sqrt{(q_{\mathsf{G}}+2\,q_H+3\,S)\,\mathsf{Adv}^{cpa}_{\mathsf{PKE}}(\mathcal{A}_{CPA}) + \frac{4\,(q_{\mathsf{G}}+2\,q_{\mathsf{H}}+3\,S)^2}{|\mathcal{M}|}} \\
&\quad + 32\,(S+3\,n)\,(q_{\mathsf{G}}+2\,q_{\mathsf{H}}+3\,S)^2\,(1-\delta) + 4\,S\,(S+n)\,\varepsilon_{dis} \\
&\quad + S^2\,(n+1)\,\mu(\mathsf{KeyGen})\,\mu(\mathsf{Enc}) + 2\,S^2 + \mu(\mathsf{KeyGen}),
\end{aligned}$$

y el tiempo de ejecución de $\mathcal{A}_{DS}$ *y* $\mathcal{A}_{CPA}$ *es el de* $\mathcal{B}$*. Aquí,*

$$\mu(\mathsf{KeyGen}) = Pr[(pk, sk) \leftarrow \mathsf{KeyGen},\ (pk', sk') \leftarrow \mathsf{KeyGen} : pk = pk']$$

y

$$\mu(\mathsf{Enc}) = Pr[(pk, sk) \leftarrow \mathsf{KeyGen};\ m, m' \leftarrow \mathcal{M},\\ c \leftarrow \mathsf{Enc}(pk, m),\ c' \leftarrow \mathsf{Enc}(pk, m') :\ c = c'].$$

3.1.2. Esquema de compromiso

En esta sección, describimos cómo obtener un PKE IND-CCA seguro para utilizarlo como esquema de compromiso a partir de un PKE IND-CPA. Este proceso se puede realizar en dos pasos:

1. Aplicar la transformación $\mathsf{FO}_m^{\not\perp}$ de [63, 64] que convierte un PKE IND-CPA en un KEM IND-CCA.

2. Aplicar la transformación propuesta en la Sección 1.3 para conseguir un PKE IND-CCA a partir de un KEM IND-CCA.

Para conseguir un KEM IND-CCA a partir de $\mathsf{Kyber.CPA}'$, utilizamos la transformación $\mathsf{FO}_m^{\not\perp}$. Es la transformación análoga a $\mathsf{FO}_{\mathsf{AKE}}$, que transforma un esquema PKE que es IND-CPA y DS en un KEM IND-CCA. Como se muestra en [63, 64], a diferencia de otras transformaciones similares, $\mathsf{FO}_m^{\not\perp}$ no requiere que el esquema PKE sea perfectamente correcto y su reducción de seguridad es más ajustada que otras transformaciones. En caso de que el PKE no sea DS, este requisito se puede eliminar con una pérdida de eficiencia despreciable [63, 64]. En el caso de $\mathsf{Kyber.CPA}'$, no existe tal pérdida de eficiencia ya que es IND-CPA y, como se muestra en el Teorema 3.1, es también DS.

Los Algoritmos 2.1, 3.4, y 3.5 muestran el KEM, que llamaremos $\mathsf{Kyber}^{\not\perp} = (\mathsf{Kyber.CPA}'.\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$, que resulta de aplicar la transformación $\mathsf{FO}_m^{\not\perp}$ a $\mathsf{Kyber.CPA}'$. De nuevo, G y H son oráculos aleatorios y H_r es un oráculo aleatorio interno que no se puede acceder directamente y se puede implementar con una función pseudoaleatoria.

Algoritmo 3.4: $\mathsf{Kyber}^{\not\perp}.\mathsf{Encaps}(pk, m)$

$c := \mathsf{Kyber.CPA}'.\mathsf{Enc}(pk, m; \mathsf{G}(m))$
$\mathsf{k} := \mathsf{H}(\mathsf{m})$
devolver (k, c)

Por completitud, presentamos el teorema de [64] que garantiza la seguridad de la transformación $\mathsf{FO}_m^{\not\perp}$ a partir de un PKE IND-CPA y DS en el QROM. Este establece que la seguridad IND-CCA de $\mathsf{FO}_m^{\not\perp} = \mathsf{FO}^{\not\perp}(\mathsf{PKE}, \mathsf{G}, \mathsf{H})$, donde PKE es un PKE y G, H son oráculos aleatorios, se reduce a la seguridad DS e IND-CPA de PKE en el QROM.

Algoritmo 3.5: $\mathsf{Kyber}^{\not\perp}.\mathsf{Decaps}(sk, c)$

$m' := \mathsf{Kyber.CPA}'.\mathsf{Dec}(sk, c)$
si $m' = \perp$ **o** $\mathsf{Kyber.CPA}'.\mathsf{Enc}(pk, m'; \mathsf{G}(m')) \neq c$ **entonces**
  devolver $\mathsf{k} := \mathsf{H}_r(c)$
en otro caso
  devolver $\mathsf{k} := \mathsf{H}(m')$

Teorema 3.3 ([64]). *Sea* PKE*=(*KeyGen*,*Enc*,*Dec*) un PKE* $(1-\delta)$*-correcto, que tiene un algoritmo de muestreo* $\overline{\mathsf{Enc}}$ *tal que es* ε_{dis}*-disjunto. Se supone que cualquier adversario tiene acceso a un oráculo DECAPS. Entonces, para cualquier adversario (cuántico) IND-CCA* $\mathcal{A}$ *realizando como mucho* q_D *consultas (clásicas) al oráculo de desencapsulación DECAPS, como mucho* q_{G} *consultas cuánticas al oráculo aleatorio* G*, y como mucho* q_H *consultas cuánticas al oráculo aleatorio* H*, existen adversarios (cuánticos)* $\mathcal{B}_{\mathsf{DS}}$ *y* $\mathcal{B}_{\mathsf{CCA}}$ *contra* PKE *tal que*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) \leq 8 \cdot (2q_{\mathsf{G}} + q_{\mathsf{H}} + q_D + 4)^2 \cdot \delta + \mathsf{Adv}^{\mathsf{DS}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{DS}}) + 2\sqrt{(q_{\mathsf{G}} + q_{\mathsf{H}}) \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{IND\text{-}CCA}}) + \frac{4(q_{\mathsf{G}} + q_{\mathsf{H}})^2}{|\mathcal{M}|}} + \varepsilon_{dis},$$

y el tiempo de ejecución de $\mathcal{B}_{\mathsf{DS}}$ *y* $\mathcal{B}_{\mathsf{IND\text{-}CCA}}$ *es el de* $\mathcal{A}$*.*

Finalmente, el PKE IND-CCA se obtiene de aplicar la transformación introducida en [32] a $\mathsf{Kyber}^{\not\perp}$ con un esquema de cifrado simétrico, que llamaremos SKE o DEM. Este último esquema viene equipado con un algoritmo de cifrado E y de descifrado D. Llamaremos al esquema PKE IND-CCA resultante $\mathsf{Kyber}^{\not\perp}.\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ formado por los Algoritmos 3.6, 3.7 y 3.8. La seguridad de esta transformación se sigue del Teorema 5 de [32]. Como se apunta en [1], un esquema de compromiso con las propiedades de seguridad requeridas se puede obtener de forma directa de un PKE IND-CCA, con lo que ya tenemos la pieza que buscábamos para nuestro GAKE compilado.

Algoritmo 3.6: $\mathsf{Kyber}^{\not\perp}.\mathsf{PKE}.\mathsf{KeyGen}()$

$(pk, sk) \leftarrow \mathsf{Kyber}^{\not\perp}.\mathsf{KeyGen}()$
devolver (pk, sk)

Algoritmo 3.7: $\mathsf{Kyber}^{\pm}.\mathsf{PKE}.\mathsf{Enc}(pk, m)$

$(c, K) \leftarrow \mathsf{Kyber}^{\pm}.\mathsf{Encaps}(pk)$
$c' := E(K, m)$
devolver $c'' := (c, c')$

Algoritmo 3.8: $\mathsf{Kyber}^{\pm}.\mathsf{PKE}.\mathsf{Dec}(sk, c'' = (c, c'))$

$K := \mathsf{Kyber}^{\pm}.\mathsf{Decaps}(sk, c)$
devolver $m := D(K, c')$

3.2. Nuestra construcción

En esta sección, presentamos nuestra construcción del protocolo GAKE. Informalmente, recalcamos que estamos considerando que los participantes son entidades honestas que pueden ser modeladas como máquinas de Turing probabilísticas en tiempo polinomial, que no tienen acceso a computación cuántica. Estos participantes sólo pueden intercambiar mensajes a través de una red insegura que está bajo el control del adversario, que puede insertar, borrar, retrasar y escuchar mensajes a su elección.

Además, las capacidades computacionales del adversario son superiores a los participantes, ya que asumimos que los adversarios pueden realizar cómputos cuánticos en tiempo polinomial y tienen acceso a cualquier función *hash* (modelada como un oráculo aleatorio) involucrada en el protocolo. El objetivo de este protocolo es garantizar que, cuando los participantes han acordado una clave de sesión a través de la red, esta es indistinguible de un valor al azar para los usuarios fuera del grupo de participantes en esa ejecución del protocolo.

Nótese que, como es norma en los protocolos GKE, no podemos esperar demostrar que el protocolo siempre terminará cuando sea ejecutado por partes honestas, sino que buscamos una garantía formal de que, siempre que el protocolo produzca una clave de salida para un participante, esta clave es segura para su uso posterior.

3.2.1. Modelo de seguridad

Nuestro modelo de seguridad se hereda de [1], que se construye a su vez sobre el trabajo de [15]. Sin embargo, nuestro modelo es menos genérico; mientras que en [1] todas las demostraciones de seguridad se realizan en el modelo CRS (del inglés *Common Reference String*, en el que no hay funciones *hash* idealizadas), nuestras demostraciones están el QROM.

Asumiremos que todas las claves públicas y parámetros necesarias para implementar Kyber$^{\pm}$.2AKE y Kyber$^{\pm}$.PKE son conocidas y están certificadas, así como que todas las funciones *hash* están modeladas como oráculos aleatorios. Además, también asumiremos que las claves de larga duración necesarias para la autenticación de Kyber$^{\pm}$.2AKE han sido generadas y distribuidas a todos los potenciales participantes en una fase de inicialización. Como es habitual, utilizamos variables para detallar la información almacenada por los usuarios en cada ejecución del protocolo, y oráculos para modelar la acción de los adversarios.

3.2.1.1. Instancias del protocolo

Cada participante en el protocolo $U_i \in \mathcal{U}$ ($i \in \mathbb{N}$) puede ejecutar un número polinomial de instancias del protocolo en paralelo. Una instancia $\Pi_i^{s_i}$ puede ser interpretada como un proceso ejecutado por el participante U_i y, en este contexto, refleja la participación en una ejecución concreta del protocolo. La notación $\Pi_i^{s_i}$ se usa para referirse a la instancia s_i del participante $U_i \in \mathcal{U}$. Cada instancia tiene asignadas siete variables:

$\mathsf{used}_i^{s_i}$ indica si esta instancia es o ha sido usada por una ejecución del protocolo. La variable $\mathsf{used}_i^{s_i}$ sólo puede activarse a través de un mensaje recibido por la instancia debido a una llamada a los oráculos Execute o Send (ver detalles abajo).

$\mathsf{state}_i^{s_i}$ mantiene el estado necesario durante la ejecución del protocolo, así como las claves de larga duración necesarias para la autenticación.

$\mathsf{term}_i^{s_i}$ muestra si la ejecución ha terminado.

$\mathsf{sid}_i^{s_i}$ denota un identificador público de la sesión que puede servir a posteriori como identificador de la clave de sesión $\mathsf{sk}_i^{s_i}$. Nótese que, aunque no construyamos identificadores de sesión como transcripciones de sesión, el adversario puede aprender todos los identificadores de sesión.

$\mathsf{pid}_i^{s_i}$ almacena el conjunto de identidades de aquellos usuarios que la instancia $\Pi_i^{s_i}$ trata de establecer una clave, incluyendo a U_i.

$\mathsf{acc}_i^{s_i}$ indica si la instancia del protocolo ha sido exitosa, es decir, si el usuario aceptó la clave de sesión.

$\mathsf{sk}_i^{s_i}$ almacena la clave de sesión una vez que es aceptada por $\Pi_i^{s_i}$. Antes de ser aceptada, su valor es NULL.

3.2.1.2. Red de comunicaciones

Asumimos que existen conexiones punto a punto disponibles entre los usuarios. La red es no privada y es completamente asíncrona: el adversario puede retrasar, escuchar, insertar y mensajes a su elección.

3.2.1.3. Capacidades del adversario

Siguiendo [63, 64], consideramos adversarios que pueden realizar cálculos (cuánticos) en tiempo polinomial y, que tienen acceso clásico a todos los oráculos (*online*) descritos abajo. Además, los adversarios tienen acceso cuántico a cualquier oráculo aleatorio involucrado (*offline*).

Las capacidades de un adversario $\mathcal{A}$ se hacen explícitas mediante acceso a oráculos que permiten a $\mathcal{A}$ comunicarse con las instancias del protocolo ejecutadas por los usuarios:

Send(U_i, s_i, M): envía un mensaje M a la instancia $\Pi_i^{s_i}$ y devuelve la respuesta generada por la instancia. Si $\mathcal{A}$ consulta este oráculo con una instancia no usada $\Pi_i^{s_i}$ y $M \subseteq \mathcal{P}$ es un subconjunto de identidades de los participantes, se activa la variable $\mathsf{used}_i^{s_i}$ y $\mathsf{pid}_i^{s_i}$ es inicializada con $\mathsf{pid}_i^{s_i} := \{U_i\} \cup M$ y el mensaje inicial del protocolo de $\Pi_i^{s_i}$ es devuelto.

Execute$(\{\Pi_{u_1}^{s_{u_1}}, \ldots, \Pi_{u_\mu}^{s_{u_\mu}}\})$: ejecuta una instancia completa entre todas las instancias no usadas de los respectivos usuarios. El adversario obtiene una transcripción de todos los mensajes enviados por la red. Una consulta al oráculo Execute refleja una escucha pasiva.

Reveal(U_i, s_i): devuelve el valor almacenado en $\mathsf{sk}_i^{s_i}$.

Test(U_i, s_i): sea b un bit elegido uniformemente al azar. Siempre que la clave de sesión esté definida, es decir, $\mathsf{acc}_i^{s_i} = \mathsf{true}$ y $\mathsf{sk}_i^{s_i} \neq \textsc{null}$ y la instancia $\Pi_i^{s_i}$ sea fresca[1], $\mathcal{A}$ puede ejecutar este oráculo en cualquier momento al ser activado. Entonces, la clave de sesión $\mathsf{sk}_i^{s_i}$ es devuelta si $b = 0$ y una clave de sesión elegida uniformemente al azar si $b = 1$. En este modelo, un número arbitrario de consultas al oráculo Test está permitido para el adversario $\mathcal{A}$, pero, una vez que el oráculo Test ha devuelto un valor para la instancia $\Pi_i^{s_i}$, devolverá el mismo valor para todas las instancias asociadas con $\Pi_i^{s_i}$ (ver Definición 3.2).

Corrupt(U_i): devuelve todas las claves de larga duración del usuario U_i. En nuestro caso, las claves privadas usadas para la autenticación en Kyber$^{\pm}$.2AKE.

[1] Una sesión fresca es aquella en la que el adversario no conoce la clave establecida por razones triviales, como haber corrompido a un usuario legítimo (ver Definición 3.5).

3.2.2. Descripción del protocolo

3.2.2.1. Corrección, integridad y confidencialidad

A continuación, definimos nuestros objetivos de corrección y seguridad. Introducimos el término *asociación* para indicar qué instancias están asociadas a una sesión común del protocolo.

Definición 3.2 ([1]). *Diremos que las instancias $\Pi_i^{s_i}$ y $\Pi_j^{s_j}$ están asociadas si $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$, $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j}$ y $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.*

Definición 3.3 ([1]). *Llamaremos* correcto *a un protocolo de intercambio de clave en grupo* P*, si en presencia de un adversario pasivos $\mathcal{A}$, es decir, $\mathcal{A}$ no debe usar los oráculos* Send *ni* Corrupt*, si se cumple que: para todo i, j con $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ y $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$, se tiene que $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq \text{NULL}$ y $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$.*

Algún tipo de corrección debería ser también garantizada incluso si los adversarios activamente participan en una ejecución concreta: la noción de *integridad*, introducida en [20], captura esta idea.

Definición 3.4 ([1]). *Decimos que un protocolo de intercambio de clave en grupo correcto cumple la propiedad de* integridad *si, con probabilidad abrumadora, todas las instancias de participantes honestos que han aceptado el mismo identificador de sesión $\mathtt{sid}_j^{s_j}$ tienen claves de sesión $\mathtt{sk}_j^{s_j}$ idénticas y están asociada a esta clave los mismos participantes $\mathtt{pid}_j^{s_j}$.*

A continuación, para detallar la definición de seguridad, tenemos que especificar bajo qué condiciones se puede pueden realizar consultas al oráculo Test.

Definición 3.5 ([1]). *Una consulta al oráculo Test debería sólo ser permitida a aquellas instancias que tienen una clave que es no conocida por el adversario por razones triviales. Para ese propósito, una instancia $\Pi_i^{s_i}$ se dice que es fresca si ninguna de las siguientes condiciones se cumple:*

- *Para algún $U_j \in \mathsf{pid}_i^{s_i}$, una consulta al oráculo $\mathsf{Corrupt}(U_j)$ fue ejecutada antes que una consulta de la forma $\mathsf{Send}(U_k, s_k, M)$ haya tenido lugar, para algún mensaje (o conjunto de identidades) M y algún $U_k \in \mathsf{pid}_i^{s_i}$.*
- *El adversario previamente consultó al oráculo $\mathsf{Reveal}(U_j, s_j)$ con $\Pi_i^{s_i}$ y $\Pi_j^{s_j}$ estando asociados.*

La idea de esta definición es que revelar una clave de sesión de una instancia $\Pi_i^{s_i}$ trivialmente produce la clave de sesión de todas las instancias asociadas con $\Pi_i^{s_i}$ y, por tanto, este tipo de ataque será excluido de la definición de seguridad.

Para un protocolo de intercambio de clave en grupo, tenemos que imponer una cota correspondiente a la *ventaja* del adversario: la ventaja $\mathsf{Adv}_{\mathcal{A}}(\ell)$ de un adversario probabilístico en tiempo polinomial $\mathcal{A}$ al atacar el protocolo P es una función del parámetro de seguridad ℓ, definido como

$$\mathsf{Adv}_{\mathcal{A}} := |2 \cdot \mathsf{Succ} - 1|.$$

Aquí, Succ es la probabilidad de que el adversario consulte al oráculo Test sólo es instancias frescas y adivina correctamente el bit b usado por el oráculo Test (sin violar la frescura de las instancias consultadas con Test).

Definición 3.6 ([1])**.** *Decimos que un protocolo autenticado de clave en grupo* P *es* seguro *si para todo adversario probabilístico en tiempo polinomial* $\mathcal{A}$ *la siguiente desigualdad se cumple para alguna función despreciable* negl*:*

$$Adv_{\mathcal{A}}(\ell) \leq \mathrm{negl}(\ell), \tag{3.1}$$

3.2.2.2. Protocolo GAKE

A continuación, describimos el protocolo GAKE que se demuestra seguro contra adversarios cuánticos, construido sobre el AKE postcuántico que denominamos $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ y usando el compilador de Abdalla et al. de la Sección 1.2.5.2.1. Nuestra construcción se muestra en la Figura 3.4. Nótese que tomamos una versión ligeramente modificada del compilador de [1]. Este difiere del original de dos formas:

- Simplificamos el cálculo de la clave de sesión y el identificador de la sesión usando dos funciones *hash* para extraer sendos valores a partir de la clave maestra K. De hecho, como usamos como primitiva el protocolo $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ que se demuestra seguro en el QROM, no tiene sentido emplear la extracción original definida en el compilador original para evitar el uso de funciones *hash*. Por tanto, sustituimos las herramientas mencionadas en la Sección 1.2.5.2.1 y usamos dos funciones *hash* H y F. Al final de la Ronda 4 del protocolo, cada usuario U_i calculará su clave de sesión como $\mathsf{sk}_i = \mathsf{H}(K)$ y el correspondiente identificador de la sesión como $\mathtt{sid}_i = \mathsf{F}(K)$, donde K es la clave maestra compartida por todos los participante en la ejecución del protocolo.
- Además, incluimos un requisito adicional para $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$, necesario para la prueba de seguridad. Como apunta Nam en [86], una condición extra se debe imponer al protocolo 2-parte en el Teorema 1 de [1]. El protocolo $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ debe cumplir la propiedad de integridad para evitar un simple ataque de *replay*. En la demostración del Teorema 1 de [1] se cumple. Debemos cambiar ligeramente el protocolo 2-parte $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ para lograr la integridad del mismo.

- **Ronda 1-2**.
  - Se ejecuta el intercambio de clave 2-parte $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}(U_i, U_{i+1})$ obteniendo cada participante U_i dos claves $\overrightarrow{K_i}$ y $\overleftarrow{K_i}$ compartidas con U_{i+1} y U_{i-1}.

- **Ronda 3**.
  - Cada U_i calcula $X_i := \overrightarrow{K_i} \oplus \overleftarrow{K_i}$ y elige un valor al azar r_i para calcular el compromiso
  $$C_i = \mathsf{Kyber}^{\pm}.\mathsf{PKE}(i, X_i; r_i).$$
  - Cada U_i transmite el valor $M_i^1 := (U_i, C_i)$.

- **Ronda 4**.
  - Cada U_i transmite $M_i^2 := (U_i, X_i, r_i)$.
  - Cada U_i comprueba que $\bigoplus_{i=0}^{n-1} X_i = 0$ y la corrección de los compromisos. Si al menos una de las comprobaciones falla, poner $\mathsf{acc}_i := \mathtt{false}$ y terminar la ejecución del protocolo.
  - Cada U_i define $K_i := \overleftarrow{K_i}$ y calcula los valores
  $$K_{i-j} := \overleftarrow{K_i} \oplus X_{i-1} \oplus \cdots \oplus X_{i-j}$$
  para $j = 1, \ldots, n-1$, que definen la clave maestra
  $$K := (K_1, \ldots, K_n, \mathsf{pid}_i)$$
  y asigna
  $$\mathsf{sk}_i := \mathsf{H}(K), \mathsf{sid}_i := \mathsf{F}(K) \text{ y } \mathsf{acc}_i := \mathtt{true}$$
  donde $\mathsf{F}, \mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\ell$ son funciones *hash* modeladas como oráculos aleatorios.

Figura 3.4: Protocolo postcuántico de intercambio de clave en grupo seguro en el QROM.

3.2.2.3. Argumentos de seguridad y demostraciones

Para demostrar que nuestra construcción es segura, usamos los siguientes resultados:

1. $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$, representado en la Figura 3.3 y los Algoritmos 3.1, 3.2 y 3.3, es seguro en el sentido de IND-StAA y puede ser modificado para obtener integridad (Definición 3.4). Nótese que, como se explica en la Sección 5 de [63, 64], IND-StAA implica la seguridad requerida en el compilador original de [1].

2. El esquema de cifrado de clave pública $\mathsf{Kyber}^{\pm}.\mathsf{PKE}$ consigue un esquema de compromiso no interactivo que es no maleable para múltiples compromisos y *perfectly binding*. Es un resultado directo de que el esquema es IND-CCA.

3.2.2.4. Una variante del AKE con integridad

Es fácil modificar la construcción de $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ para obtener integridad. La principal idea es añadir un segundo oráculo aleatorio F que, antes de realizar la derivación de la clave de sesión, será aplicada a la misma entrada que H para derivar el identificador de la sesión. Es trivial ver que, de esta forma, la integridad de esta versión modificada de $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ se logra tanto en el ROM como el QROM, debido a la resistencia a colisiones de los oráculos aleatorios. De hecho, supongamos que $\mathsf{k} = \mathsf{k}'$. Como H y F son oráculos aleatorios, su resistencia a colisiones garantiza que, con probabilidad abrumadora, ambos participantes tienen los mismos identificadores de sesión, y por tanto, la misma clave de sesión k. Este argumento es válido para los oráculos aleatorios clásicos y cuánticos del tipo *quantum-accessible* (ver Sección 2.2.2). En lo sucesivo, asumiremos que esta modificación tiene lugar y que $\mathsf{Kyber}^{\pm}.\mathsf{2AKE}$ tiene integridad.

3.2.2.5. Seguridad del protocolo

A continuación, presentamos las demostraciones de seguridad de seguridad del protocolo: la primera tiene lugar en el ROM y la segunda en el QROM, siguiendo [63, 64].

Teorema 3.4. *En el Random Oracle Model, el protocolo presentado en la Figura 3.4 es un protocolo autenticado de intercambio de clave en grupo que es correcto, íntegro y seguro en el sentido de las Definiciones 3.3, 3.4 y 3.6.*

Demostración. Esta demostración es una adaptación de la prueba de seguridad del Teorema 1 de [1], que usamos como herramienta principal en la construcción del protocolo.

Corrección. En una ejecución honesta del protocolo, es fácil verificar que todos los participantes terminarán aceptando y calculando la misma clave de sesión y el mismo identificador de la sesión.

Integridad. Debido a la resistencia a colisiones del oráculo aleatorio F, todos los oráculos que aceptan con idénticos identificadores de sesión también obtendrán con probabilidad abrumadora la misma clave maestra K y sid (que puede ser extraída de K) derivará, por tanto, la misma clave de sesión $\mathsf{H}(K)$.

Seguridad de la clave. La demostración de la seguridad de la clave en el sentido de la Definición 3.6 se realizará en una secuencia de juegos, comenzando con el ataque real contra la seguridad de la clave del protocolo de intercambio de clave en grupo y terminando con un juego en el que la ventaja del adversario es 0, y para que el podemos acotar la diferencia en la ventaja del adversario para dos juegos consecutivos. Siguiendo la notación estándar, denotaremos con $\mathsf{Adv}(\mathcal{A}, G_i)$ la ventaja del adversario $\mathcal{A}$ en el Juego i. Por claridad, clasificaremos las consultas al oráculo Send en tres categorías, dependiendo a qué ronda del protocolo esté asociada la consulta, comenzando con Send-0 y terminando Send-2. Send-t denota la consulta al oráculo Send asociada a la ronda t para $t = 0, 1, 2$.

Los primeros tres juegos de esta demostración son exactamente los mismos que en la demostración del Teorema 1 de [1].

Juego 0. Este primer juego corresponde al ataque real, en el que todos los parámetros, como los parámetros públicos y las claves de larga duración se asocian a cada usuario, se eligen como en el protocolo. Por definición,

$$\mathsf{Adv}(\mathcal{A}, G_0) = \mathsf{Adv}(\mathcal{A}).$$

Juego 1. En este juego, para $i = 1, \ldots, n$, modificamos la simulación de los oráculos Send y $\mathsf{Execute}$ para que, cuando una instancia $\Pi_i^{s_i}$ todavía sea considerada fresca al final de la Ronda 2, las claves $\overleftarrow{K_i}$ y $\overrightarrow{K_i}$ que se comparten con las instancias $\Pi_{i-1}^{s_{i-1}}$ y $\Pi_{i+1}^{s_{i+1}}$, se reemplacen con valores elegidos al azar de un conjunto apropiado.

Es fácil ver que la distancia entre este juego y el anterior está acotada por la probabilidad de que el adversario rompa la seguridad del protocolo subyacente 2-AKE. Como resultado, se cumple que

$$\left|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_0)\right| \leq 2 \cdot \mathsf{Adv}_{\mathsf{2\text{-}AKE}}(\ell, 2 \cdot q_{send}),$$

donde q_{send} representa el número de consultas de las instancias diferentes del protocolo al oráculo Send.

Juego 2. En este juego, cambiamos la simulación del oráculo Send para que, una instancia fresca $\Pi_i^{s_i}$ no acepte en la Ronda 4, cuando un compromiso C_j para

$j \neq i$, que se recibe de la Ronda 3, haya sido generado por el simulador, pero no por su respectiva instancia $\Pi_j^{s_j}$, $j \neq i$ en la misma sesión.

El adversario $\mathcal{A}$ puede detectar la diferencia del Juego G_1 si $\mathcal{A}$ fabrica un compromiso que debería haber llevado a la aceptación en la Ronda 4 en ese juego. Como el valor comprometido X_i es un valor aleatorio independiente de los mensajes anteriores, la probabilidad de esto es despreciable.

$$\left|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)\right| \leq \mathrm{negl}(\ell)$$

Juego 3. Este juego reproduce la modificación para los compromisos generados por el adversario. La simulación del oráculo Send cambia para que, una instancia fresca $\Pi_i^{s_i}$ no acepte en Ronda 4, cuando un compromiso C_j para $j \neq i$ recibido en la Ronda 3 haya sido por el adversario. La ventaja del adversario diverge sólo en una probabilidad despreciable del juego anterior:

$$\left|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)\right| \leq \mathrm{negl}(\ell)$$

Juego 4. Ahora, las simulaciones de los oráculos Execute y Send son modificados cuando se va a calcular la clave de sesión. La simulación mantiene una lista de cadenas $(K_1, \ldots, K_n, \mathcal{G})$. Una vez que la instancia recibe la última consulta a Send-2, el simulador calcula $K_1, \ldots, K_n$ y comprueba si para la correspondiente cadena $(K_1, \ldots, K_n, \mathcal{G})$, una clave maestra fue ya emitida. Si ese es el caso, se asigna la correspondiente clave maestra a la instancia. Si no existe en la lista, el simulador elige una clave de sesión $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformemente al azar. Nótese que, incluso si los mensajes de la Ronda 4 se envían, la clave maestra todavía contiene suficiente entropía para que la salida del oráculo aleatorio H sea indistinguible de una clave aleatoria $\mathsf{sk}_i^{s_i}$, sólo con probabilidad despreciable. Como resultado,

$$\left|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)\right| \leq \mathrm{negl}(\ell).$$

Claramente, en el Juego G_4, todas las claves de sesión son elegidas uniformemente al azar y el adversario no obtiene ninguna ventaja.

$$\mathsf{Adv}(\mathcal{A}, G_4) = 0.$$

□

Teorema 3.5. *En el Quantum Random Oracle Model, el protocolo presentado en la Figura 3.4 es un protocolo autenticado de intercambio de clave en grupo que es correcto, íntegro y seguro en el sentido de las Definiciones 3.3, 3.4 y 3.6.*

Demostración. La demostración sigue el mismo razonamiento del Teorema 3.4; Sólo destacamos que el argumento del Juego 4 sigue siendo válido cuando se

consideran oráculos aleatorios *quantum-accessible*. De hecho, en el último juego, las simulaciones de los oráculos Execute y Send son modificados en el punto de calcular la clave de sesión. El simulador mantiene una lista de cadenas de la forma $(K_1, \dots, K_n, \mathcal{G})$, y, tras recibir la última consulta al oráculo Send-2, se calculan los valores $K_1, \dots, K_n$ y se comprueba si la clave maestra correspondiente ya se ha emitido previamente. Si este es el caso, esta clave maestra se asignará a esta instancia. En otro caso, el simulador elige una clave de sesión $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformemente al azar. En este punto, todas las claves $K_1, \dots, K_n$ son elegidas uniformemente al azar y son desconocidas para el adversario. El adversario sólo puede advertir este último cambio si ya se ha consultado la misma clave al oráculo aleatorio cuántico H. Este evento sucederá con probabilidad despreciable. Como resultado, la salida de H es indistinguible de una clave aleatoria $\mathsf{s}k_i^{s_i}$ con probabilidad abrumadora. Por tanto, tenemos

$$\left|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)\right| \leq \mathrm{negl}(\ell).$$

Claramente, en el Juego G_4, todas las claves de sesión son elegidas al azar uniformemente al azar y el adversario no tiene ninguna ventaja.

$$\mathsf{Adv}(\mathcal{A}, G_4) = 0.$$

□

3.3. Trabajo relacionado

Numerosos protocolos GKE que se pueden considerar postcuánticos han sido propuestos. Entre los que basan su seguridad en isogenias (Sección 2.2.3.5) se encuentra una construcción llamada SIBD propuesta en [50], que sólo considera adversarios pasivos. Sobre este trabajo, se han diseñado construcciones basadas en árboles que incluyen un diseño autenticado que se puede encontrar en [62]. En particular, la propuesta A-SIT y su versión *peer to peer* llamada A-P2P-SIT resiste a ataques activos y los participantes en el protocolo se autentican a través de un esquema de firma postcuántico. Sin embargo, estas construcciones asumen un algoritmo de configuración de un grafo de doble árbol (*double tree graph*) que se ejecuta en cada instancia del protocolo, y además es necesario suponer que cada participante es consciente de la posición que ocupa en esa estructura. Por otro lado, en Fujioka et al. [48] se presentan dos protocolos autenticados de una ronda, llamados n-UM y BC n-DH, cuya seguridad reside en problemas asociados a isogenias entre dos curvas elípticas supersingulares con el mismo número de puntos. El primero se demuestra seguro en el QROM, mientras el segundo se demuestra seguro en el ROM.

Otros protocolos usan problemas sobre retículos (Sección 2.2.3.1) como base. Apon et al. [8] construyeron un protocolo de tres rondas no autenticado que se demuestra seguro bajo la hipótesis *Ring Learning With Errors* (RLWE). Este esquema se puede transformar en un protocolo autenticado mediante el uso del compilador de Katz y Yung [73]. Sin embargo, el esquema resultante añade una ronda adicional de comunicación y cada mensaje que se envía debe ser firmado, añadiendo un sobrecoste en cómputo y comunicación si se emplea un esquema de firma postcuántico. Usando el mismo problema como base, Choi et al. [30] construyeron sobre el compilador de Katz y Yung [73] tres protocolos: el primero es autenticado, el segundo añade autenticación y el tercero es dinámico[2]. El segundo protocolo llamado STAG es un protocolo de tres rondas en el que cada participante en el protocolo debe realizar dos firmas. Todas estas propuestas se construyen sobre el compilador de Burmester-Desmedt (Figura 1.4) y sus variantes, mientras que una nueva propuesta de [117] permite la implementación de un GAKE basado en isogenias y retículos, usando el compilador de Just y Vaudenay [70]. Este se puede instanciar tomando el protocolo de Apon et al. [8] o usar una variante del esquema de la propuesta basada en isogenias de [50].

Por último, otras propuestas se basan en KEM, en vez de en protocolos de intercambio de clave 2-parte. Persichetti et al. [95] presentó un protocolo de tres rondas a partir de un KEM y un esquema de firma. Cada participante necesita firmar un mensaje. González Vasco et al. [119] introdujeron un GAKE de dos rondas derivado de un KEM y un código de autenticación de mensaje (MAC, por sus siglas en inglés). Sin embargo, en este último trabajo, se asume que la seguridad sólo se cumple en el escenario denominado *future-quantum*, donde el adversario que tiene acceso a computación cuántica sólo cuando la ejecución del protocolo se ha completado. Un protocolo de transporte de clave se ha propuesto en [37], que puede ser instanciado usando KEM postcuánticos. El escenario considerado en esta construcción es que cada participante de un servicio en la nube quiere distribuir una clave criptográfica al resto de usuarios, con la cooperación de un tercero no confiable.

3.3.1. Comparativa entre el estado del arte y nuestra construcción

A continuación, comparamos nuestra construcción con el estado del arte, fijándonos en algunas de las características más relevantes para su evaluación. La Tabla 3.1 resume algunos parámetros relacionados con el rendimiento de los protocolos. En este tipo de protocolos, el parámetro más importante es el número

[2]Este tipo de protocolos permiten añadir y eliminar miembros del grupo. Los protocolos que no permiten realizar esto reciben el nombre de estáticos.

de rondas. Además, para cada protocolo, indicamos si se evita el uso de firmas postcuánticas, que es una buena propiedad ya que este tipo de firmas son muy costosas en términos de computación y tamaño. Nótese que el protocolo de [8] no usa firmas postcuánticas, pero no es autenticado. Finalmente, incluimos el número total de mensajes enviados en la ejecución del protocolo con n participantes, los mensajes retransmitidos y los mensajes enviados punto a punto (PtP, por sus siglas en inglés), es decir, mensajes enviados de una parte a otra.

Tabla 3.1: Algunos parámetros de eficiencia de algunos protocolos GKE/GAKE postcuánticos.

Protocolo	**Rondas**	**Evita Firmas**	**Mensajes retransmitidos**	**Mensajes PtP**
n-UM [48]	1	Sí	n	0
BC n-DH [48]	1	Sí	n	0
Apon et al. [8]	3	Sí (no auten.)	$2n+1$	0
STAG [30]	3	No	$2n+1$	0
Nuestra construcción	4	Sí	$2n$	$2n$

La Tabla 3.2 se centra en la seguridad de los protocolos. La primera columna incluye el tipo de hipótesis en la que el protocolo basa su seguridad. La segunda columna, establece en qué modelo de seguridad se demuestra que es seguro: el Random Oracle Model (ROM) o en el Quantum Random Oracle Model (QROM). Como ya hemos comentado, el último modelo es más fuerte que el primero porque se asume un adversario más poderoso. Después, se especifica en qué escenario se cumplen las propiedades del protocolo: en el escenario *future quantum* (Fut-Q) o en el escenario postcuántico (PostQ). El último, que asume que el adversario tiene acceso a computación cuántica es preferible al primero donde la seguridad de clave sólo se garantiza contra adversarios que no pueden hacer cálculos cuánticos durante la ejecución del protocolo, pero tendrán está opción en algún momento en el futuro. Finalmente, indicamos si el intercambio de clave es autenticado.

Resumen del capítulo

En este capítulo, hemos diseñado un protocolo GAKE postcuántico de 4 rondas basado en el compilador de Abdalla et al. y en el finalista del concurso de estandarización del NIST Kyber, demostrando que el protocolo es seguro en el

Tabla 3.2: Seguridad de algunos protocolos GKE/GAKE postcuánticos.

Protocolo	**Tipo aproximación**	**Modelo**	**FutQ/PostQ**	**Autenticado**
n-UM [48]	Isogenias	QROM	PostQ	Sí
BC n-DH [48]	Isogenias	ROM	PostQ	Sí
Apon et al. [8]	Retículos	ROM	PostQ	No
STAG [30]	Retículos	ROM	PostQ	Sí
Nuestra construcción	Retículos	QROM	PostQ	Sí

QROM. En el siguiente capítulo, implementaremos dicho protocolo y realizaremos un análisis de rendimiento.

Capítulo

4

Implementación del protocolo GAKE

Este capítulo se centra en la implementación del protocolo GAKE propuesto en el capítulo anterior, construido sobre el compilador de Abdalla et al. y Kyber, cuya seguridad demostramos en el modelo de seguridad QROM. Recordemos que el compilador asume que los participantes en el protocolo están distribuidos en un anillo (Figura 1.6). Uno de sus componentes esenciales, Kyber, es un KEM finalista del concurso de estandarización del NIST (ver detalles en la Sección 2.3.1.1).

Además de describir la implementación de este protocolo, comparamos los distintos algoritmos que componen el mismo con sus homólogos que sólo se demuestran seguros en el ROM (propuestos por los autores de Kyber). La implementación realizada se encuentra disponible en GitHub: `https://github.com/jiep/kyber-gake`.

4.1. Implementación de Kyber

Comenzamos describiendo la implementación de Kyber de la que parte nuestro desarrollo, ya que la mayoría del código del protocolo GAKE se basa en el

código que tienen disponible los autores de Kyber en GitHub[1].

Los autores de Kyber han instanciado el esquema con tres niveles de seguridad: KYBER512 (menor nivel de seguridad), KYBER768 (nivel medio), y KYBER1024 (mayor nivel de seguridad). Los oráculos aleatorios de cada uno de ellos han sido realizados con funciones derivadas de Keccak, estandarizado en FIPS 202 [42]. Concretamente, Kyber usa las funciones SHAKE-128 y SHAKE-256; y SHA3-256 y SHA3-512. Nótese que, Kyber implementa, para cada nivel de seguridad, las variantes que denominan 90s, que están diseñadas para ofrecer soporte para *hardware* antiguo, donde Keccak puede ser lento. Estas variantes usan AES256-CTR y SHA2 en vez de SHAKE y SHA3, por lo que no las consideraremos a lo largo de este capítulo.

El repositorio original de Github de Kyber contiene dos implementaciones escritas en C99 [68]: la implementación de referencia y la implementación optimizada. Esta última emplea instrucciones de AVX2 para paralelizar las funciones de Keccak [22] y sólo funciona cuando estas instrucciones se encuentran disponibles en la CPU. En lo sucesivo, nos referiremos a la implementación de referencia como ref y a la implementación optimizada como avx2.

Para implementar el protocolo GAKE, el código original de Kyber ha sido modificado para adaptarse a las especificaciones de $\text{Kyber}^{\pm}$.2AKE, $\text{Kyber}^{\pm}$ y $\text{Kyber}^{\pm}$.PKE. La única primitiva que queda inalterada es Kyber.CPA′, que sirve como punto de partida del protocolo (ver Figura 3.1). Los tamaños de clave de Kyber.CPA′ se muestran en la Tabla 4.1.

Tabla 4.1: Tamaño (en bytes) de los textos cifrados, claves públicas y secretas del esquema Kyber.CPA′ por nivel de seguridad.

	KYBER512	KYBER768	KYBER1024
Texto cifrado	736	1088	1568
Clave pública	800	1184	1568
Clave secreta	736	1152	1536

[1] https://github.com/pq-crystals/kyber

4.2. Implementación del AKE y del esquema de compromiso

4.2.1. Kyber$^{\pm}$.2AKE

Kyber$^{\pm}$.2AKE es el protocolo AKE 2-parte del GAKE, que se forma a partir de la transformación $\mathsf{FO}_{\mathsf{AKE}}$. Recordemos que dicha transformación convierte un esquema PKE IND-CPA — en nuestro caso, Kyber.CPA′ — en un AKE. Este envía dos mensajes que llamaremos M y M' (ver Figura 3.3). M está formado por una clave pública y un texto cifrado, mientras que M' está compuesto de dos textos cifrados (ver Figura 4.1). El esquema Kyber$^{\pm}$.2AKE hereda los tamaños de clave de Kyber.CPA′ (ver Tabla 4.1).

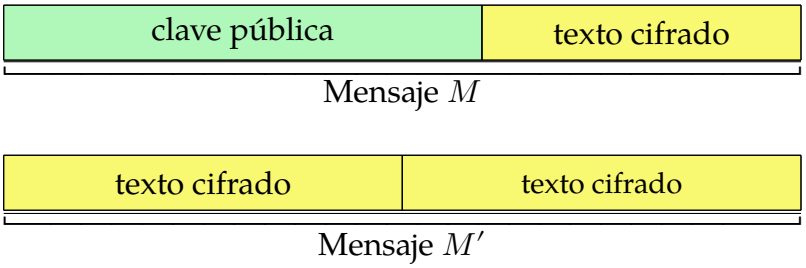


Figura 4.1: Mensajes enviados durante la Ronda 1-2.

4.2.2. Esquema de compromiso

El esquema de compromiso se deriva de Kyber$^{\pm}$.PKE, que se obtiene, a su vez, de Kyber$^{\pm}$, un KEM IND-CCA seguro en el QROM que, por su parte, se deriva de Kyber.CPA′.

4.2.2.1. Kyber$^{\pm}$

Kyber$^{\pm}$ también hereda los tamaños de clave de Kyber.CPA′. La Tabla 4.2 muestra las longitudes de las claves, del texto cifrado y del secreto compartido. Kyber$^{\pm}$ se usa como un esquema determinista, en el que los *nonces* empleados para generar el compromiso se almacenan y, posteriormente, en la Ronda 4 del protocolo (ver Figura 3.1), se envían para volver a generar cada compromiso y comprobar si es válido.

Tabla 4.2: Tamaño (en bytes) de los textos cifrados, claves públicas y secretas y tamaño del secreto compartido del esquema Kyber$^{\pm}$, por nivel de seguridad.

	KYBER512	KYBER768	KYBER1024
Texto cifrado	736	1088	1568
Clave pública	800	1184	1568
Clave secreta	736	1152	1536
Clave simétrica	32	32	32

4.2.2.2. Kyber$^{\pm}$.PKE

Kyber$^{\pm}$.PKE se apoya en la construcción híbrida KEM/DEM, descrita en la Sección 3.1.2, con Kyber$^{\pm}$ como KEM. Como DEM se puede usar cualquier cifrado simétrico como, por ejemplo, AES256-GCM, CHACHA20-POLY1305, etc. Nótese que todos estos esquemas simétricos tienen una clave de 32 bytes (256 bits), que coincide con la longitud de la clave que genera Kyber$^{\pm}$. Estos DEM son autenticados, por lo que devuelven un *tag* que garantiza la integridad del texto cifrado. Como en el KEM, debemos almacenar la aleatoriedad, es decir, el vector de inicialización (IV, por sus siglas en inglés) para enviarlo en la Ronda 4 del protocolo (ver Figura 3.1). La construcción KEM/DEM se resume en la Figura 4.2.

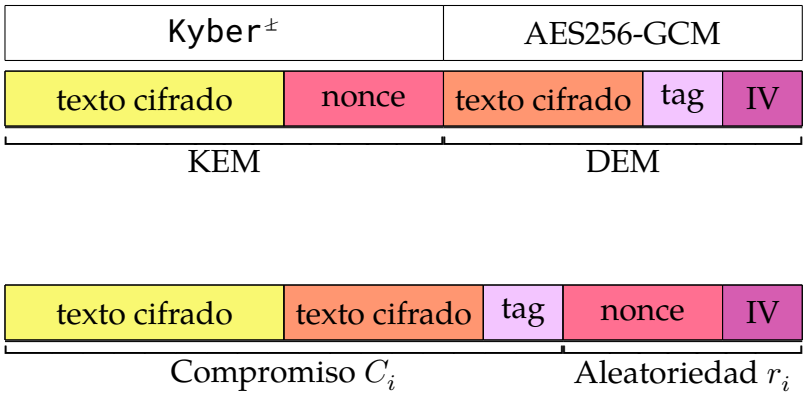


Figura 4.2: Composición del compromiso a partir del KEM y DEM.

Finalmente, el compromiso consiste en los textos cifrados del KEM y DEM y del *tag* del DEM. Los valores aleatorios involucrados son el *nonce* del KEM y el IV del DEM (ver Figura 4.2 y sus correspondientes tamaños en la Tabla 4.3).

Tabla 4.3: Tamaño (en bytes) del IV y el tag para AES256-GCM y CHACHA20-POLY1305.

	AES256-GCM	CHACHA20-POLY1305
Tamaño IV	12	12
Tamaño *tag*	16	16
Tamaño de la clave	32	32

4.3. Implementación del protocolo

La implementación del protocolo GAKE (Figura 3.1) está escrita en C99 y hereda las funciones hash del código original, es decir, las funciones `G`, `H`, etc. de los algoritmos son funciones de la familia `SHA3`. Por otro lado, las implementaciones del DEM provienen de `OpenSSL 1.1.1f` [118]. Esta implementación asume una red de comunicaciones sin latencia.

El protocolo permite ejecutar un número polinomial de instancias en paralelo. Por tanto, es necesario definir una serie de variables para mantener el estado de cada instancia; sus definiciones vienen heredadas del compilador de Abdalla et al. [1]:

- `public_key` contiene la clave pública de autenticación.
- `secret_key` contiene la clave secreta de autenticación.
- `pid` contiene los identificadores de los usuarios participantes U_i ejecutando la instancia del protocolo.
- `sk` es la clave de sesión. Su tamaño es de 32 bytes. Por defecto, su valor es 0^{256}.
- `sid` es el identificador público de la clave de sesión `sk`. Su tamaño es de 32 bytes.
- `term` es una variable booleana que indica si la instancia ha terminado. En la implementación, 0 indica `false` y 1, `true`.
- `acc` es una variable booleana que indica si la instancia ha sido aceptada.
- Otras variables que contienen todo lo necesario para ejecutar el protocolo, como por ejemplo, $\overleftarrow{K_i}, \overrightarrow{K_i}, X_i, r_i, K$, etc.

4.3.1. Fase de inicialización

En la implementación, se ha definido una fase de inicialización, que llamaremos `Init`, en la que todos los participantes en el protocolo generan sus claves de larga duración para autenticarse como miembros legítimos del grupo. Se asume que los de participantes conocen las claves pública del resto de participantes. Además, durante esta fase, también se reserva memoria para la estructura de datos que almacena todas las variables que contienen el estado de la instancia.

4.3.2. Ronda 1-2

Durante la Ronda 1-2, dos tipos de mensajes (M y M') son intercambiados (ver Figura 4.1). En total, se envían n mensajes de cada tipo durante esta fase del protocolo, donde n es el número de participantes en el protocolo.

4.3.3. Ronda 3

Durante la Ronda 3, los mensajes M_i^1 para $i = 0, \ldots, n-1$ se transmiten al resto de participantes. Estos mensajes están descritos en la Figura 4.3. El tamaño de U_i se ha fijado en 20 bytes y se requieren 36 bytes para cifrar $i \,||\, X_i$ ($|X_i| = 32$ y $|i| = 4$). El tamaño del *tag* depende de DEM empleado (ver Tabla 4.3).

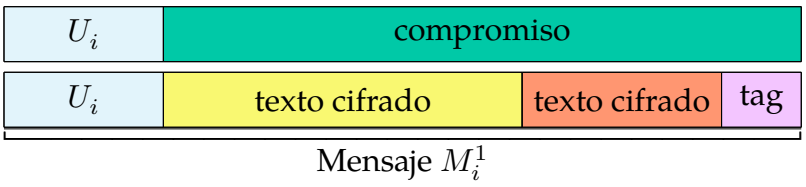


Figura 4.3: Mensaje enviado durante la Ronda 3.

4.3.4. Ronda 4

En la Ronda 4, n mensajes M_i^2 se retransmiten al resto de participantes en el protocolo, que están descritos en la Figura 4.4. El tamaño de U_i es de 20 bytes, como en la Ronda 3. X_i tiene un tamaño de 32 bytes, el *nonce* tiene 32 bytes y el IV depende del DEM empleado (ver Tabla 4.3).

La Figura 4.5 muestra una ejecución del protocolo GAKE con 100 participantes para cada una de las implementaciones disponibles (`ref` y `avx2`).

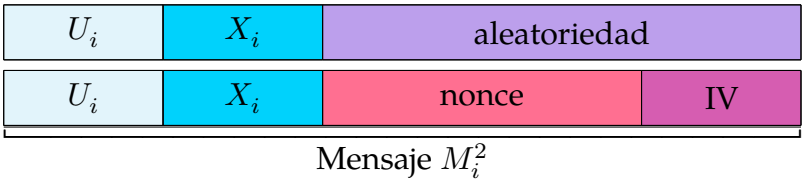


Figura 4.4: Mensaje enviado durante la Ronda 4.

```
> ./ref/test_gake_qrom1024_ref 100
Round 1-2
Round 3
Round 4
All keys are equal!
Session key: ab7e8742974376d0821a7f27871a8946fd165d023d5a5f52fdb82aafe80d6297
Session id:  6985f5ce0fa3e74906e270ff42db4bf24b74e524c86fa29e9420ca39637c42fe


Time stats
        Init time      : 0.031s (2.35%)
        Round 1-2 time : 0.141s (10.59%)
        Round 3 time   : 0.000s (0.00%)
        Round 4 time   : 1.156s (87.06%)
        Total time     : 1.328s (100.00%)
```

(a) Implementación `ref`

```
> ./avx2/test_gake_qrom1024_avx2 100
Round 1-2
Round 3
Round 4
All keys are equal!
Session key: a6666aa09022f5c3fece6c7607b948605218c94e34a8ff7a4a3bda53f693dff9
Session id:  c2d1d11c02e3187faf133e865037a77dfc26de7e4fb37569b4692a729e56f6db


Time stats
        Init time      : 0.016s (5.56%)
        Round 1-2 time : 0.031s (11.11%)
        Round 3 time   : 0.000s (0.00%)
        Round 4 time   : 0.234s (83.33%)
        Total time     : 0.281s (100.00%)
```

(b) Implementación `avx2`

Figura 4.5: Ejecución del protocolo GAKE con 100 participantes para las implementaciones `ref` y `avx2`.

4.4. Resultados experimentales

En esta sección, mostramos los resultados experimentales de la implementación del protocolo. Hemos implementado el mismo protocolo GAKE usando las primitivas que sólo se demuestran seguras en el ROM (ver detalles en la Sección 2.3.1.1) para poder comparar el rendimiento entre ese marco y el QROM. De aquí en adelante, nos referiremos con `QROM` a los algoritmos y esquemas que son seguros en el QROM, mientras que denotaremos con `ROM` a sus homólogos seguros en el ROM. La Tabla 4.4 describe los algoritmos específicos para cada uno de estos escenarios.

Tabla 4.4: Comparativa de los algoritmos implementados en la los modelos de seguridad `QROM` y `ROM`.

Esquema	`QROM`	`ROM`
IND-CCA PKE	`Kyber.CPA'`	`Kyber.CPA'`
KEM IND-CCA	$\texttt{Kyber}^{\perp}$ Algoritmos 2.1, 3.4, y 3.5	`Kyber` Algoritmos 2.1, 2.4, y 2.5
AKE 2-parte	`Init` (Algoritmo 3.1) $\texttt{Der}_{\text{resp}}$ (Algoritmo 3.2) $\texttt{Der}_{\text{init}}$ (Algoritmo 3.3)	`Init` (Algoritmo 2.9) `AlgB` (Algoritmo 2.10) `AlgA` (Algoritmo 2.11)
PKE IND-CCA	Algoritmos 3.6–3.8	Algoritmos 2.6–2.8
Esquema compromiso	Aplicación directa del PKE IND-CCA	Aplicación directa del PKE IND-CCA

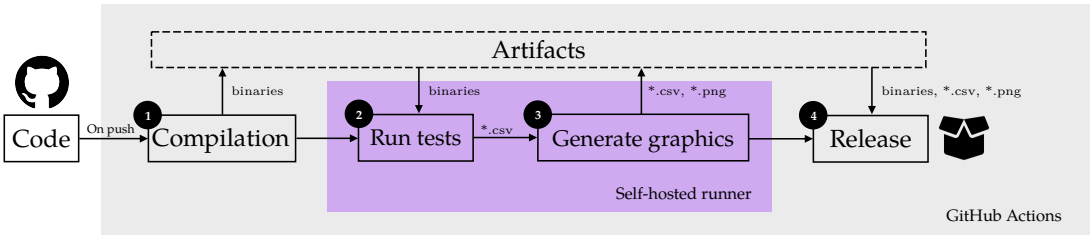


Figura 4.6: *Workflow* que ejecuta los experimentos en GitHub Actions.

Por reproducibilidad, hemos desarrollado un *workflow* de CI/CD [51] para ejecutar los experimentos en un entorno aislado. El *workflow* (Figura 4.6) se ha

creado en GitHub Actions y contiene todas todos los pasos necesarios desde que se sube nuevo código al repositorio de GitHub hasta que los experimentos se ejecutan. Está compuesto de las siguientes acciones:

1. **Compilation**: en cada nuevo *commit* al repositorio de GitHub, se lanza un evento que desencadena el *workflow*. En este paso, se compila el código en una máquina[2] con sistema operativo Ubuntu 20.04 hospedado en GitHub Actions. Se ha decidido emplear `CMake` para automatizar la construcción de los ficheros binarios de las dos implementaciones, aplicando la opción `-DMAKE_BUILD_TYPE=Release` y emplear `gcc` como compilador de C. Todos los binarios se enlazan estáticamente con sus dependencias (`OpenSSL`). La implementación `ref` se compila con las siguientes opciones en `gcc`: `-O3 -fwrapv`, mientras que la implementación `avx2` se compila con las opciones `-O3 -mavx2 -mbmi -mpopcnt -maes -fomit-frame-pointers`.

2. **Run tests**: este paso ejecuta todos los experimentos en un *runner* auto-hospedado[3] porque los *runner* ejecutados por GitHub Actions tienen un tiempo máximo de ejecución de 6 horas [53], que no es suficiente para ejecutar todos los experimentos definidos. Las especificaciones del *runner* auto-hospedado están descritas en la Tabla 4.5.

   Tabla 4.5: Especificaciones de *hardware* del *runner* auto-hospedado.

Característica	**Valor**
Sistema operativo	Ubuntu 20.04.1 LTS
CPU	i7-6700HQ@2.60 GHz
RAM	8 GB
AVX2 habilitado	Sí

   Cada uno de los *tests* mide el tiempo de ejecución de cada uno de los algoritmos de los que está compuesto el protocolo GAKE. Para cada implementación (`ref` y `avx2`), para cada nivel de seguridad (`512`, `768` y `1024`) y para cada modelo de seguridad (`QROM` y `ROM`) se ejecutan los siguientes *tests*:

   - `test_speed_kem`: mide el tiempo medio de ejecución de 10 000 iteraciones del KEM IND-CCA de la Tabla 4.4. Además, mide el tiempo

[2]Las máquinas que ejecutan el código reciben el nombre de *runner* en GitHub Actions.

[3]Un *runner* auto-hospedado es una máquina que ejecuta uno o varios pasos de un *workflow* fuera de las instalaciones de GitHub Actions y no está sujeto a las restricciones impuestos por ellos.

de ejecución de los algoritmos de generación de clave, de encapsulación y de desencapsulación por separado.

- `test_speed_ake`: mide el tiempo medio de ejecución de 10 000 iteraciones del protocolo AKE 2-parte de la Tabla 4.4. También, mide el tiempo de ejecución de cada uno de los tres algoritmos que los está compuesto el protocolo AKE.
- `test_speed_commitment`: mide el tiempo medio de ejecución de 10 000 iteraciones del esquema de compromiso de la Tabla 4.4. También, mide, individualmente, cada uno de los algoritmos del esquema: la generación de claves, la generación del compromiso (`Commit`) y la comprobación del mismo (`Check`).
- `test_speed_gake`: mide el tiempo medio de ejecución de 10 iteraciones del protocolo en función del número de participantes en el mismo, denotado por n. Se ejecuta para $n = 2, 2^2, ..., 2^{11} = 2048$. Además, se mide el tiempo de ejecución para cada una de las rondas definidas y el tiempo de la fase de inicialización.

Todos los *tests* anteriores generan ficheros CSV que son procesados posteriormente.

3. **Generate graphics**: este paso procesa los ficheros CSV y genera los gráficos de este capítulo. Se han generado de forma automática con Python y la librería de visualización `seaborn` [121].

4. **Create release**: finalmente, se añade una *tag* al repositorio de GitHub y se genera una nueva *release*. Esta contiene todos los binarios resultantes de la compilación, los ficheros CSV y todos los gráficos.

Todos los ficheros necesarios en los pasos posteriores del *workflow* se almacenan como artefactos en GitHub Actions [52].

4.4.1. Comparativa entre implementaciones

La Tabla 4.6 muestra una comparativa entre la implementación de referencia (`ref`) frente a la implementación optimizada (`avx2`). Se puede observar que la implementación optimizada es entre 6 y 16 veces más rápida que la implementación de referencia. Estos resultados no son sorprendentes, pero muestra hasta qué punto es recomendable usar la implementación optimizada siempre que el *hardware* la soporte. Es importante destacar que, si algunos participantes en el protocolo usan la implementación `avx2` y otros la implementación `ref` para

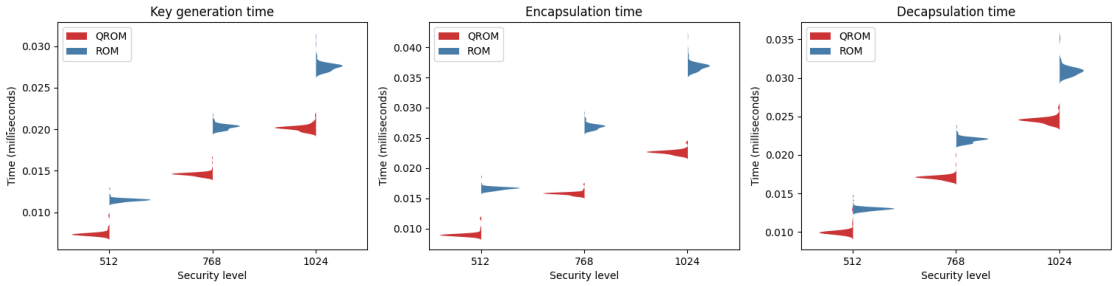
ejecutar el protocolo, los usuarios usando esta última crearán un cuello de botella, ya que es mucho más lenta y el resto de participantes se quedarán esperando, lo que reducirá el rendimiento drásticamente.

Tabla 4.6: Comparación de la velocidad de las dos implementaciones para las distintas operaciones de las primitivas, dependiendo el nivel de seguridad. Se muestra cuántas veces es más rápida la implementación `avx2` con respecto a `ref`.

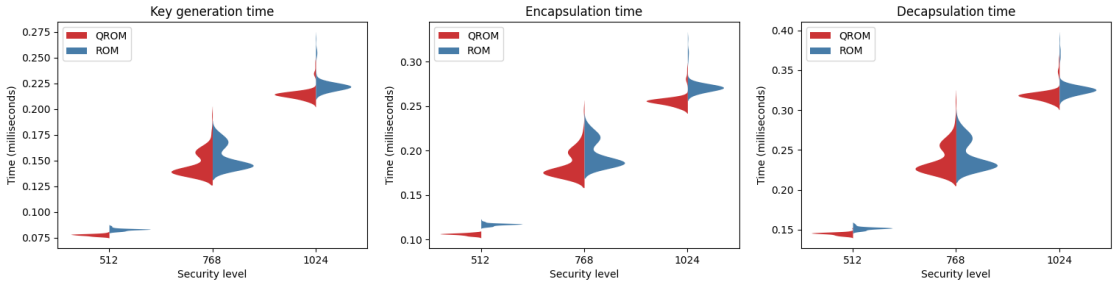
	KYBER512		KYBER768		KYBER1024	
	QROM	ROM	QROM	ROM	QROM	ROM
KEM						
KeyGen	10.42x	7.07x	9.15x	6.87x	10.82x	7.79x
Encaps	11.56x	6.91x	10.64x	6.75x	11.46x	7.16x
Decaps	14.32x	11.23x	12.87x	9.93x	13.04x	10.20x
Esquema compromiso						
Commit	10.56x	6.53x	10.57x	6.47x	10.96x	6.99x
Check	10.66x	6.58x	10.57x	6.46x	10.93x	6.96x
2-AKE						
Init	11.34x	7.09x	10.21x	6.77x	10.91x	7.67x
Der_{resp} AlgB	12.79x	8.23x	11.51x	7.62x	11.80x	8.27x
Der_{init} AlgA	16.48x	11.37x	14.32x	10.01x	13.80x	10.43x

4.4.2. KEM

La Figura 4.7 muestra la comparativa de rendimiento (medido en tiempo) entre Kyber (ROM) y $\text{Kyber}^{\pm}$ (QROM). Se puede observar que todos los algoritmos del KEM en el modelo de seguridad QROM (KeyGen, Encaps y, Decaps) son más rápidos que sus homólogos en el modelo de seguridad ROM. Esta diferencia de rendimiento es mayor en la implementación avx2 y es mayor a medida que el nivel de seguridad aumenta.



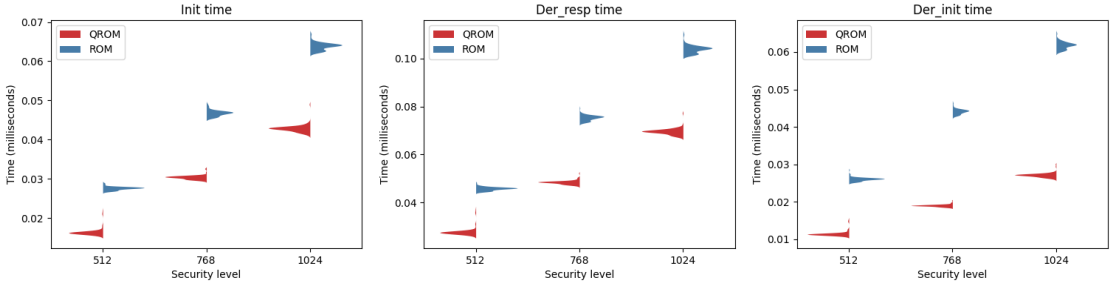
(a) Implementación `avx2`.



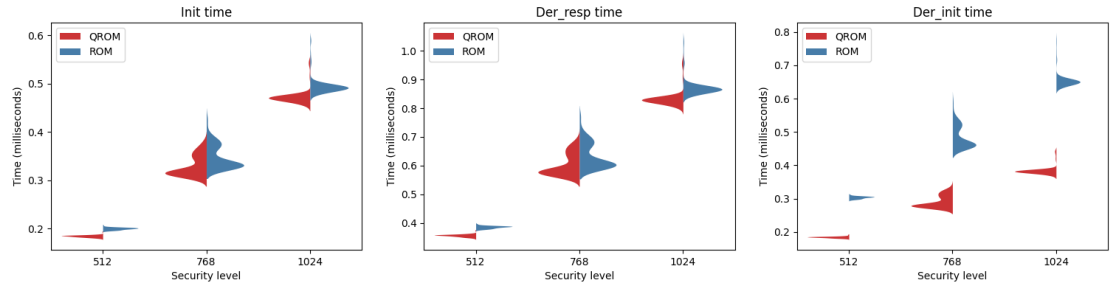
(b) Implementación `ref`.

Figura 4.7: Tiempo de ejecución de los algoritmos del KEM, dependiendo del nivel de seguridad y modelo de seguridad (`QROM` o `ROM`).

4.4.3. AKE 2-parte



(a) Implementación `avx2`.



(b) Implementación `ref`.

Figura 4.8: Tiempo de ejecución de los algoritmos del protocolo AKE 2-parte, dependiendo del nivel de seguridad y el modelo de seguridad (`QROM` o `ROM`).

La Figura 4.8 muestra el tiempo de ejecución de los algoritmos del protocolo AKE 2-parte para los modelos de seguridad `QROM` y `ROM`. Se observa el mismo comportamiento que en el caso del KEM: los algoritmos del modelo `QROM` son más rápidos que sus homólogos en el modelo de seguridad `ROM` y, a medida que el nivel de seguridad aumenta, la diferencia es mayor. Esto se ve especialmente acentuado en el caso de la implementación `avx2`.

4.4.4. Esquema de compromiso

Existen muchas opciones para elegir un DEM para derivar el esquema de compromiso que necesitamos. La Tabla 4.7 muestra el tiempo de ejecución de 10 000 iteraciones, eligiendo `AES256-GCM` y `CHACHA20-POLY1305` como DEM en el esquema de compromiso, para cada nivel de seguridad. Ambas implementaciones provienen de `OpenSSL 1.1.1f` [118]. Se puede observar que el tiempo de ejecución entre ellos es muy similar en todos casos: `CHACHA20-POLY1305` es más rápido con `KYBER512` y `KYBER768`, mientras que `AES256-GCM` es más

rápido con KYBER1024). Debido a esta pequeña diferencia entre estos DEM, preferimos el uso de AES256-GCM al estar estandarizado por el NIST [44]. Así, en lo sucesivo, el esquema de compromiso se fijará usando AES256-GCM como DEM.

Tabla 4.7: Tiempo medio de ejecución (en milisegundos) en la generación de 10 000 compromisos con AES256-GCM y CHACHA20-POLY1305 como DEM.

	KYBER512	KYBER768	KYBER1024
AES256-GCM	0.12406	0.19454	0.27534
CHACHA20-POLY1305	0.12207	0.19183	0.27664

La Figura 4.9 muestra el tiempo de ejecución para los modelos de seguridad QROM y ROM. Se exhibe el mismo comportamiento que en los casos anteriores.

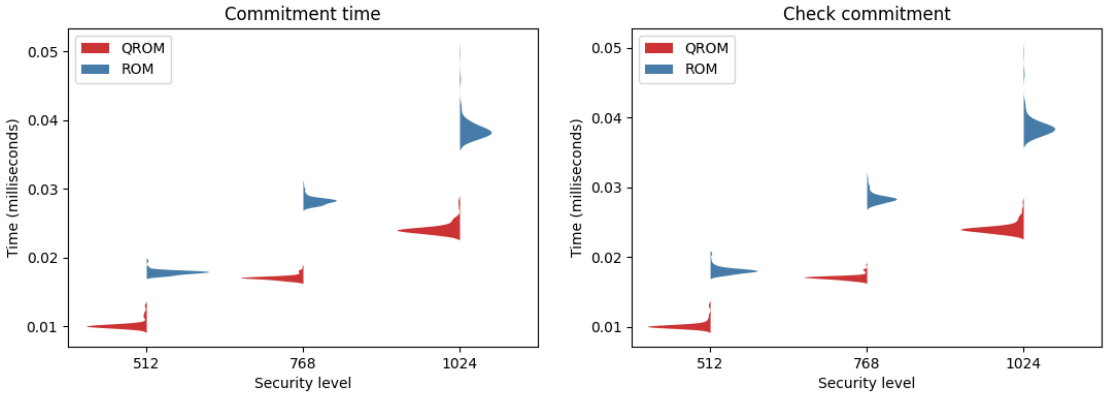
4.4.5. Protocolo

La Figura 4.10 muestra el porcentaje del tiempo consumido en cada ronda del protocolo. Se puede observar que no hay diferencias significativas entre el modelo de seguridad QROM y ROM. La ronda que consume más tiempo es la Ronda 4, durante la cual se comprueban si los compromisos recibidos del resto de participantes son válidos, se deriva la clave maestra y se genera la clave de sesión. Se muestra también la Ronda Init, en la que se reserva memoria para las estructuras de datos que almacenan los estado de la instancia del protocolo y se generan las claves de autenticación de cada participante.

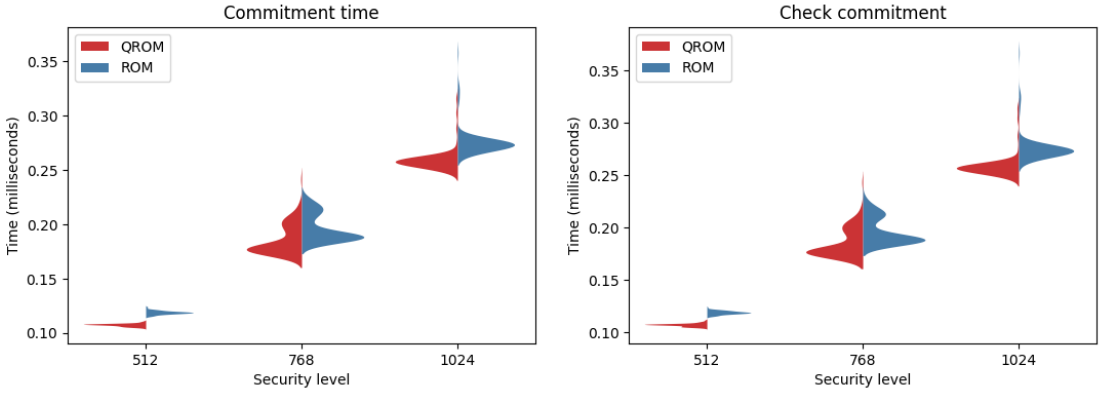
Con respecto a la escalabilidad del protocolo, la Figura 4.11 muestra el tiempo de ejecución (que cada participante ejecuta en paralelo) en función del número de participantes involucrados. Se puede observar un crecimiento lineal, que es el esperado atendiendo al diseño del protocolo.

4.4.6. Discusión de los resultados

Los experimentos anteriores muestran que, en este protocolo, la elección de las primitivas que están demostradas seguras en el modelo de seguridad QROM no conlleva una pérdida de rendimiento; al contrario, las primitivas implementadas en el modelo de seguridad ROM son menos eficientes. Este hecho es más evidente en la implementación optimizada avx2 y a medida que el nivel de seguridad es mayor. Como resultado, concluimos que un modelo de seguridad más fuerte como QROM no siempre implica una pérdida de rendimiento en un protocolo. En esta implementación concreta, esta diferencia de rendimiento se puede deber a los distintos tamaños de clave secreta involucrados en cada modelo de seguridad. En el modelo de seguridad QROM, el punto de partida es el

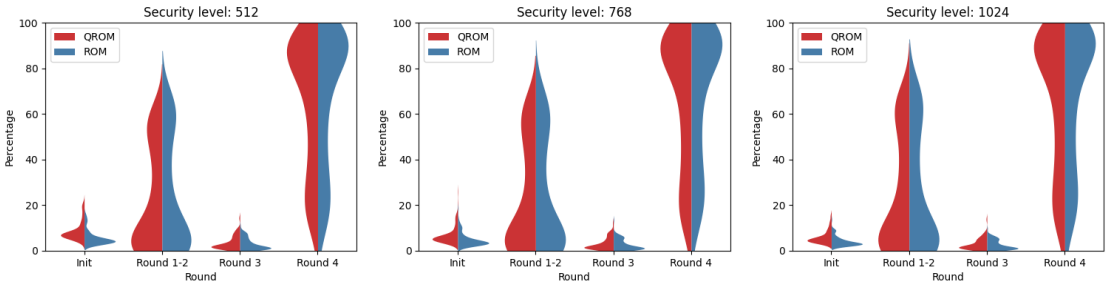


(a) Implementación `avx2`.

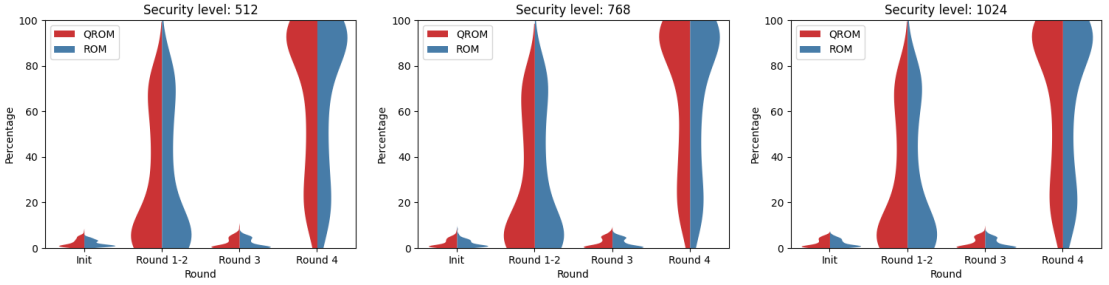


(b) Implementación `ref`.

Figura 4.9: Tiempo de ejecución de los algoritmos del esquema de compromiso, dependiendo del nivel de seguridad y el modelo de seguridad (`QROM` o `ROM`).

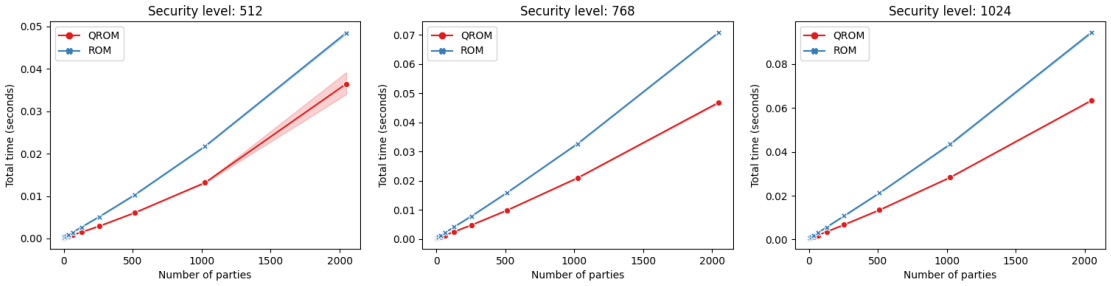


(a) Implementación `avx2`.

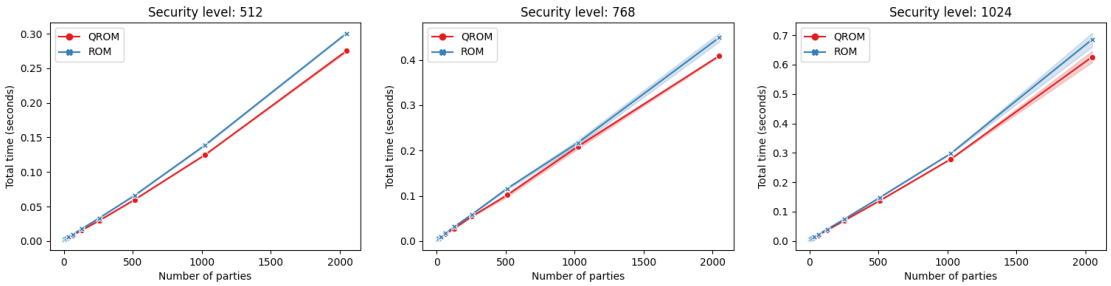


(b) Implementación `ref`.

Figura 4.10: Porcentaje del tiempo total del protocolo empleado en cada ronda, en función del nivel de seguridad y del modelo de seguridad (`QROM` o `ROM`).



(a) Implementación `avx2`.



(b) Implementación `ref`.

Figura 4.11: Tiempo total de ejecución (en milisegundos) de cada participante en el protocolo, dependiendo del número de partes involucradas, el nivel de seguridad y el modelo de seguridad (`QROM` o `ROM`).

esquema de clave pública IND-CPA `Kyber.CPA′`, del que se derivan el resto de primitivas. En el caso del modelo de seguridad `ROM`, se emplea en `Kyber` como punto de partida, del que se derivan el resto de primitivas, que tiene un tamaño de claves mayor (ver Tablas 4.2 y 4.8).

Tabla 4.8: Tamaño (en bytes) de los textos cifrados, claves pública y secreta de `Kyber`, según su nivel de seguridad.

	KYBER512	KYBER768	KYBER1024
Texto cifrado	736	1088	1568
Clave pública	800	1184	1568
Clave secreta	1632	2400	3168

4.5. Trabajo relacionado

Una introducción genérica sobre las implementaciones del concurso del NIST se puede encontrar en la Sección 2.3.3. Aquí, nos centraremos en implementaciones de Kyber. La mayoría de propuestas de la literatura se centran en la optimización, la extensión a arquitecturas *hardware* y la protección de las implementaciones frente a ataques de canal lateral. En [125], los autores implementan Kyber sobre una FPGA con bajos recursos computaciones, consiguiendo un rendimiento razonable. En concreto, sobre Xiling Artix-7. En [128], proponen una optimización *hardware* para Kyber, testado sobre la misma FPGA anterior, logrando un rendimiento mejor entre 1.5 y 3.95 veces. Más recientemente, en [57], proponen un nuevo diseño con un menor tiempo de ejecución que requiere del uso de menor registros que sus predecesores.

En ARM, también se han desarrollado implementaciones, la mayoría sobre Cortex-M4. En [24], proponen una implementación *software* optimizada que es un 18 % más rápida que la implementación de los autores de Kyber, minimizando, además, el uso de memoria. En [2], presentan una implementación optimizada que mejora entre un 15.9 % y 17.8 % las operaciones más limitantes. Además, la implementación es pública y se puede encontrar en GitHub[4]. En [110], proponen una implementación optimizada para los procesadores Cortex-A de 64 bits.

Sobre la arquitectura *open hardware* RISC-V, también hay implementaciones. En [5], presentan una extensión al conjunto de instrucciones de RISC-V, que evalúan sobre Kyber en varias FPGAs.

[4] `https://github.com/FasterKyberDilithiumM4/FasterKyberDilithiumM4`

Por otro lado, se han propuesto implementaciones resistentes a ataques de canal lateral. En [23], se presenta un diseño específico para ARM Cortex-M0+ y Cortex M4F y en [59] para Cortex M4, siendo *open source*[5]. En [69], implementan protección frente a ataques de canal lateral introduciendo sólo un 5 % de sobrecoste.

Resumen del capítulo

En este capítulo hemos implementado el protocolo GAKE propuesto en el capítulo anterior, comparando el rendimiento de cada primitiva y del protocolo completo con primitivas que son seguras en el ROM. Los resultados experimentales muestran que todas las primitivas y el protocolo completo son más eficientes en el caso del modelo de seguridad QROM, lo que muestra un hecho contraintuitivo: un modelo de seguridad más fuerte como es QROM, en este caso, no introduce una pérdida de rendimiento. En el capítulo siguiente, veremos cómo extender el mismo diseño de protocolo GAKE a otros KEM finalistas del NIST, más allá de Kyber.

[5] `https://github.com/masked-kyber-m4/mkm4`

Capítulo

5

Extendiendo el protocolo GAKE más allá de Kyber

En este capítulo, extendemos el protocolo GAKE del Capítulo 3 al resto de KEM finalistas del concurso de estandarización del NIST. En nuestra construcción, evitamos tener que imponer que el KEM de base cumpla la propiedad de DS (Definición 3.1). Para ello, empleamos la transformación FSXY propuesta por Fujioka et al. en [46], que obtiene un AKE 2-parte de un KEM, seguro en el ROM.

Nuestro objetivo final es obtener una variante del protocolo GAKE que pueda ser implementado con cualquiera de los KEM finalistas del concurso del NIST (cumplan o no la propiedad de DS): Classic McEliece, Kyber, NTRU y Saber. De hecho, conseguimos un protocolo, que puede ser usado con cualquier KEM IND-CCA (aunque, por supuesto, no será seguro frente a adversarios cuánticos si no se garantiza que el KEM es postcuántico). Recordemos que el protocolo del Capítulo 3 aplica la transformación $\mathsf{FO}_{\mathsf{AKE}}$, que tiene alguna ventaja sobre FSXY como que es segura en el QROM en vez del ROM, pero no puede ser aplicada a cualquier KEM, sino que tiene que cumplir la propiedad de DS y ser un esquema IND-CPA. Esta última noción de seguridad no se cumple en el caso de Classic McEliece y NTRU por ser esquemas deterministas.

De nuevo, el protocolo resultante se ha implementado dejando el código dis-

ponible en GitHub: `https://github.com/jiep/pq-gake-fsxy`.

5.1. Diseño teórico

Existen algunas transformaciones genéricas que convierten KEM en AKE 2-parte en el modelo estándar como [25] y [47]. Estas transformaciones generan protocolos AKE a partir de KEM IND-CCA usando funciones pseudoaleatorias. Los AKE resultantes se demuestran seguros en los modelos de seguridad CK [28] y CK+ [47], respectivamente. Desgraciadamente, los KEM seguros en el modelo estándar son computacionalmente ineficientes tanto para comunicaciones clásicas como postcuánticas.

5.1.1. FSXY: una construcción genérica de KEM a AKE 2-parte

En [46], Fujioka et al., proponen una construcción genérica de protocolos AKE a partir de KEM OW-CCA, que llamaremos FSXY, relajando el modelo de seguridad al ROM. El AKE resultante se demuestra seguro en el modelo CK+ en el ROM. De hecho, se muestra que las hipótesis acerca de las funciones de una vía postcuánticas como (Ring-)LWE, McEliece, NTRU, etc. se pueden usar para construir protocolos AKE seguros. Además, evidenciamos que adaptando el ROM en la demostración de seguridad de FSXY, los protocolos AKE obtenidos para cada hipótesis postcuántica se vuelven eficientes en términos del coste de comunicación.

La transformación FSXY se construye a partir de dos KEM como sigue. Sea $\mathsf{KEM}_1 = (\mathsf{KeyGen}_1, \mathsf{Encaps}_1, \mathsf{Decaps}_1)$ un KEM OW-CCA y $\mathsf{KEM}_2 = (\mathsf{KeyGen}_2, \mathsf{Encaps}_2, \mathsf{Decaps}_2)$ un KEM OW-CPA. Sea ℓ el parámetro de seguridad y $H_1 : \{0,1\}^* \to \mathcal{R}$ y $H_2 : \{0,1\}^* \to \{0,1\}^\ell$ dos funciones *hash* modeladas como oráculos aleatorios, donde $\mathcal{R}$ es un espacio del que extraemos valores uniformente al azar. Los valores aleatorios r y r_1 se eligen de $\{0,1\}^{f(\ell)}$, donde f es una función polinómica en el parámetro de seguridad. La Figura 5.1 muestra el protocolo AKE que involucra a los usuarios U_A (la parte iniciadora) y U_B (la parte que responde). A los algoritmos de esta transformación los llamaremos `Init`, `AlgB` y `AlgA`.

Nótese que esta transformación es de dos mensajes y sigue la misma estructura que la transformación $\mathsf{FO}_{\mathsf{AKE}}$ (ver detalles en la Sección 3.1.1).

Teorema 5.1 ([46]). *Si $\mathsf{KEM}_1 = (\mathsf{KeyGen}_1, \mathsf{Encaps}_1, \mathsf{Decaps}_1)$ es un KEM OW-CCA y $\mathsf{KEM}_2 = (\mathsf{KeyGen}_2, \mathsf{Encaps}_2, \mathsf{Decaps}_2)$ es un KEM OW-CPA, entonces el AKE de la Figura 5.1 es seguro en el modelo CK+, donde H_1 y H_2 son funciones hash, modeladas como oráculos aleatorios.*

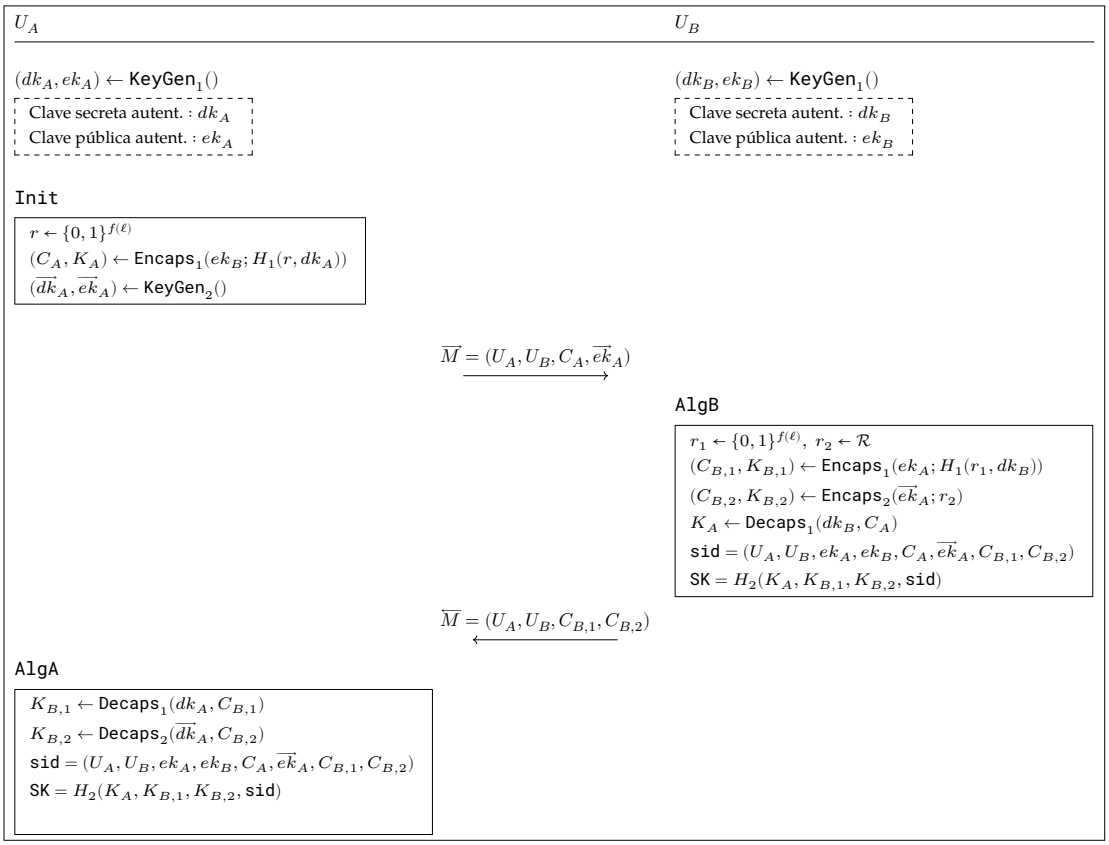


Figura 5.1: Transformación FSXY.

5.1.2. Construcción del esquema de compromiso

Como en los capítulos precedentes, necesitamos construir un esquema de compromiso no interactivo y no maleable que sea *perfectly binding* y no maleable para múltiples compromisos. Este se puede construir a partir de cualquier PKE IND-CCA. De nuevo, emplearemos la aproximación KEM/DEM (ver detalles en la Sección 1.3), siendo el KEM cualquiera entre Classic McEliece, Kyber, NTRU y Saber y, el DEM será AES256-GCM.

5.1.3. Protocolo GAKE

La construcción del protocolo GAKE sólo requiere de las siguientes primitivas (ver Figura 5.2):

- KEM = (KeyGen, Encaps, Decaps), que es un KEM IND-CCA.
- SKE = (Enc, Dec) es un esquema simétrico de cifrado que se usa como DEM.
- H, G son funciones *hash*, modeladas como oráculos aleatorios.

Para obtener el GAKE, se instancia el protocolo AKE 2-parte con la transformación FSXY, usando KEM como KEM_1 y KEM_2. Nótese que la noción de IND-CCA implica tanto la seguridad OW-CPA como OW-CCA requeridos por KEM_1 y KEM_2. El protocolo AKE 2-parte resultante satisface el modelo de seguridad CK+ [47], que es suficiente para el compilador. También se necesita la propiedad de integridad (Definición 3.4) para el AKE 2-parte con el fin de alcanzar la noción de seguridad necesaria. Es una comprobación directa que el AKE obtenido de FSXY tiene integridad debido a la forma en que se calculan los identificadores de sesión. El otro ingrediente necesario para el compilador es un esquema de compromiso, que también se obtiene de KEM. Obsérvese que todos los KEM finalistas del concurso del NIST alcanzan la seguridad IND-CCA y pueden ser utilizados en nuestra construcción. Por último, las funciones *hash* G y H se utilizan para derivar los identificadores de sesión y las claves en el AKE y en el paso final del protocolo, respectivamente.

La Figura 5.3 muestra el protocolo GAKE completo. Nótese que la única diferencia entre este protocolo y el propuesto en la Figura 3.4 es cómo se genera el AKE 2-parte. En la Ronda 4, la función hash H se elige al doble del tamaño necesario de salida para generar la clave de sesión sk_i a partir de la primera mitad de $\mathsf{H}(K)$ y construir $\mathtt{sid}_i$ con la segunda mitad.

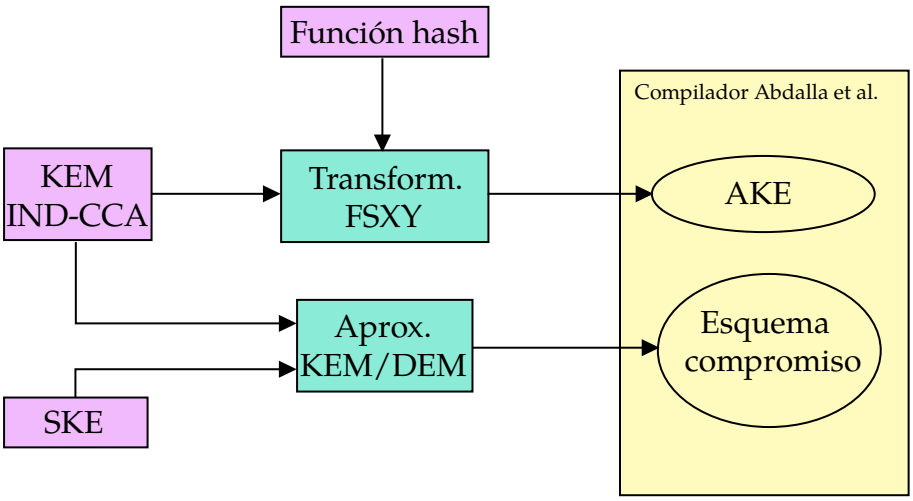


Figura 5.2: Relaciones entre las primitivas y las transformaciones aplicadas al protocolo GAKE.

5.2. Implementación

A continuación, describimos la implementación del protocolo GAKE. Para ello, describimos individualmente las primitivas del protocolo.

5.2.1. KEM

Los KEM se han tomado de la librería *open source* LibOQS [115], que provee todos los KEM finalistas del concurso de estandarización del NIST.

5.2.1.1. LibOQS

LibOQS [104, 115] es una librería escrita en C99 [68] que contiene todos los esquemas postcuánticos finalistas del concurso de estandarización del NIST. La Tabla 5.1 muestra el estado de los esquemas implementados. Ha sido desarrollada dentro del proyecto Open Quantum Safe (OQS), cuyo objetivo es el desarrollo y la implementación de la criptografía postcuántica. A fecha de escritura de estas líneas, no se recomienda su uso en producción ni para proteger datos sensibles, sólo para realizar pruebas de concepto. Entre sus principales ventajas se encuentran:

- Permite la gestión dinámica de los esquemas, haciendo posible intercambiar uno por otro sin necesidad de modificar el código de un protocolo.

- **Ronda 1-2**.
  - Se ejecuta el intercambio de clave 2-parte de la transformación FSXY, obteniendo cada participante U_i dos claves $\overrightarrow{K_i}$ y $\overleftarrow{K_i}$ compartidas con U_{i+1} y U_{i-1}.

- **Ronda 3**.
  - Cada U_i calcula $X_i := \overrightarrow{K_i} \oplus \overleftarrow{K_i}$ y elige un valor al azar r_i para calcular el compromiso C_i a partir del esquema PKE IND-CCA de la aproximación KEM/DEM
    $$C_i = \mathsf{PKE}(i, X_i; r_i).$$
  - Cada U_i transmite el valor $M_i^1 := (U_i, C_i)$.

- **Ronda 4**.
  - Cada U_i transmite $M_i^2 := (U_i, X_i, r_i)$.
  - Cada U_i comprueba que $\bigoplus_{i=0}^{n-1} X_i = 0$ y la corrección de los compromisos. Si una de las comprobaciones falla, poner $\mathtt{acc}_i := \mathtt{false}$ y terminar la ejecución del protocolo.
  - Cada U_i define $K_i := \overleftarrow{K_i}$ y calcula los valores
    $$K_{i-j} := \overleftarrow{K_i} \oplus X_{i-1} \oplus \cdots \oplus X_{i-j}$$
    para $j = 1, \ldots, n-1$, que definen la clave maestra
    $$K := (K_1, \ldots, K_n, \mathsf{pid}_i)$$
    y asigna
    $$(\mathsf{sk}_i, \mathtt{sid}_i) := \mathsf{H}(K) \text{ y } \mathtt{acc}_i := \mathtt{true}$$
    donde $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\ell}$ es una función *hash* modelada como un oráculo aleatorio.

Figura 5.3: Protocolo postcuántico de intercambio de clave en grupo a partir de la transformación FSXY.

Tabla 5.1: Esquemas implementados en LibOQS. El color verde indica esquema implementado; el rojo, no implementado; y, el naranja indica que está implementado, pero hace un gran uso de la pila.

Estatus	KEM	Firma
Finalistas	Classic McEliece	Dilithium
	Kyber	Falcon
	NTRU	Rainbow
	SABER	
Candidatos alternativos	BIKE	GeMSS
	FrodoKEM	Picnic
	HQC	SPHINCS+
	NTRU Prime	
	SIKE	

- Posibilita construir la librería con sólo los esquemas necesarios en la aplicación.
- Simplifica la compilación cruzada. Además, LibOQS proporciona los *toolchain* para las arquitecturas más comunes.
- Provee de utilidades como funciones *hash*, funciones de generación de bits aleatorios o sobreescritura de secretos con ceros, entre otras.
- Se integra con las librerías, protocolos y aplicaciones criptográficas más utilizadas (por ejemplo, OpenSSL y BoringSSL, OpenSSH, curl, Chromium, entre otras). Más detalles se pueden encontrar en [105, 106, 107, 108].
- Soporta otros lenguajes de programación.

LibOQS provee 10 instanciaciones[1] para Classic McEliece, 3 para Kyber[2], 4 para NTRU, y 3 para Saber. La Tabla 5.2 muestra los tamaños de clave (pública y secreta), del secreto compartido y de los textos cifrados, así como el nivel y el modelo de seguridad. Nótese que el tamaño de la clave pública de Classic McEliece es varios órdenes de magnitud mayor que el resto de KEM. Por otro

[1]Sólo nos centraremos en los KEM finalistas del concurso de estandarización. Los detalles sobre las firmas digitales y los candidatos alternativos de los KEM se pueden encontrar en [104].

[2]LibOQS provee 6 instanciaciones, pero las variantes llamadas `90s` no se consideran porque son soportadas por *hardware* antiguo y no soportan SHA-3 y nuestras implementaciones dependen de ella.

lado, el tamaño del texto cifrado de Classic McEliece es más pequeño que el resto de finalistas.

Dos implementaciones diferentes vienen para cada instanciación: la implementación de referencia (llamada ref, clean o vec por los KEM) y la implementación optimizada (llamada avx2 o avx). Toda la información sobre estas implementaciones se puede encontrar en la Tabla 5.3. Nótese que las implementaciones de referencia no presentan ninguna limitación con respecto a la arquitectura o sistema operativo de base, mientras que la implementación optimizada sólo funciona en la arquitectura x86_64 para los sistemas operativos macOS y Linux. También, es destacable que, las implementaciones de Classic McEliece hacen un gran uso de la pila, lo que puede causar fallos cuando se ejecuta sobre aplicaciones multihilo o en entornos con recursos limitados [104].

Tabla 5.2: Propiedades de las diferentes instanciaciones de los KEM finalistas en LibOQS. Todos los tamaños vienen dados en bytes. Fuente: [104].

KEM	Instanciación	Modelo de seguridad	Nivel de seguridad	Tamaño clave pública	Tamaño clave secreta	Tamaño texto cifrado	Tamaño secreto compartido	Tamaño aleat.
Classic McEliece	Classic-McEliece-348864	IND-CCA	1	261 120	6 452	128	32	436
	Classic-McEliece-348864f	IND-CCA	1	261 120	6 452	128	32	436
	Classic-McEliece-460896	IND-CCA	3	524 160	13 568	188	32	576
	Classic-McEliece-460896f	IND-CCA	3	524 160	13 568	188	32	576
	Classic-McEliece-6688128	IND-CCA	5	1 044 992	13 892	240	32	836
	Classic-McEliece-6688128f	IND-CCA	5	1 044 992	13 892	240	32	836
	Classic-McEliece-6960119	IND-CCA	5	1 047 319	13 908	226	32	870
	Classic-McEliece-6960119f	IND-CCA	5	1 047 319	13 908	226	32	870
	Classic-McEliece-8192128	IND-CCA	5	1 357 824	14 080	240	32	1 024
	Classic-McEliece-8192128f	IND-CCA	5	1 357 824	14 080	240	32	1 024
Kyber	Kyber512	IND-CCA	1	800	1 632	768	32	32
	Kyber768	IND-CCA	3	1 184	2 400	1 088	32	32
	Kyber1024	IND-CCA	5	1 568	3 168	1 568	32	32
NTRU	NTRU-HPS-2048-509	IND-CCA	1	699	935	699	32	2413
	NTRU-HPS-2048-677	IND-CCA	3	930	1 234	930	32	3 211
	NTRU-HPS-4096-821	IND-CCA	5	1 230	1 590	1 230	32	3 895
	NTRU-HRSS-701	IND-CCA	3	1 138	1 450	1 138	32	1 400
Saber	LightSaber-KEM	IND-CCA	1	672	1 568	736	32	32
	Saber-KEM	IND-CCA	3	992	2 304	1 088	32	32
	FireSaber-KEM	IND-CCA	5	1 312	3 040	1 472	32	32

5.2.2. AKE 2-parte

El protocolo AKE 2-parte se ha implementado siguiendo la transformación `FSXY` de la Figura 5.1, que se ha divido en tres algoritmos:

- `Init`: algoritmo que ejecuta U_A al principio del protocolo y devuelve el mensaje M.
- `AlgB`: algoritmo que ejecuta U_B tomando el mensaje M y devuelve el mensaje M' y la clave de sesión `SK`.

Tabla 5.3: Características de cada instanciación de los KEM finalistas de LibOQS. Fuente: [104].

KEM	Instanciación	Identificador implementación	Arquitecturas soportadas	SO soportados	Extensiones de CPU	¿Sin ramificaciones en los secretos?	¿Gran uso de la pila?
Classic McEliece	Classic-McEliece-348864	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
	Classic-McEliece-348864f	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT,BMI1	✘	✔
	Classic-McEliece-460896	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
	Classic-McEliece-460896f	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT,BMI1	✘	✔
	Classic-McEliece-6688128	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
	Classic-McEliece-6688128f	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT,BMI1	✘	✔
	Classic-McEliece-6960119	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
	Classic-McEliece-6960119f	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT,BMI1	✘	✔
	Classic-McEliece-8192128	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
	Classic-McEliece-8192128f	vec	Todos		Ninguna	✔	✔
		avx	x86_64		AVX2,POPCNT	✘	✔
Kyber	Kyber512	ref	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,POPCNT,BMI1	✔	✘
	Kyber768	ref	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,POPCNT,BMI1	✔	✘
	Kyber1024	ref	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,POPCNT,BMI1	✔	✘
NTRU	NTRU-HPS-2048-509	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,BMI2	✔	✘
	NTRU-HPS-2048-677	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,BMI2	✔	✘
	NTRU-HPS-4096-821	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,BMI2	✔	✘
	NTRU-HRSS-701	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2,BMI2	✔	✘
Saber	LightSaber-KEM	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2	✘	✘
	Saber-KEM	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2	✘	✘
	FireSaber-KEM	clean	Todos		Ninguna	✔	✘
		avx2	x86_64		AVX2	✘	✘

- `AlgA`: algoritmo que ejecuta U_A al final del protocolo tomando como entrada el mensaje M' y devuelve la clave de sesión `SK`.

En la implementación, $\mathsf{KEM}_1 = \mathsf{KEM}_2$ y la funciones hash H_1 y H_2 son `SHA3-256`, procedentes de LibOQS.

5.2.3. Esquema de compromiso

El esquema de compromiso se ha implementado con la construcción híbrida KEM/DEM, siendo el KEM cualquiera de los implementados en LibOQS y el DEM es `AES256-GCM` tomado de `OpenSSL 1.1.1f` [118]. El compromiso está formado por los textos cifrados del KEM y el DEM y la *tag* del DEM. La aleatoriedad r_i está formada por el *nonce* del KEM y el IV del DEM (ver Figura 4.2). Nótese que es necesario almacenar en memoria r_i ya que durante la Ronda 4 se debe transmitir al resto de participantes, por lo que los KEM deben ser deterministas, lo que implica modificar todas las implementaciones de LibOQS (ver Tabla 5.3).

Se han implementados tres algoritmos:

- `Init` reserva memoria para los textos cifrado del KEM y DEM.
- `Commit` genera un compromiso a partir del esquema PKE.
- `Check` crea un compromiso y comprueba si es igual a un compromiso creado anteriormente.

5.2.4. Protocolo GAKE

El protocolo GAKE se ha implementado con las primitivas anteriores. Las funciones hash vienen de `SHA-3`, procedentes de LibOQS. Además, como en el capítulo anterior, la implementación asume que una red de comunicaciones no tiene retardo.

El protocolo permite ejecutar un número polinomial de instancias del protocolo en paralelo, que son modeladas mediante una serie de variables, como en la Sección 4.3.

5.2.4.1. Fase de inicialización

Durante la fase de inicialización `Init`, se generan las claves de larga duración para autenticar a todos los participantes en el protocolo y se asume que las claves públicas son conocidas por todos ellos. Además, se reserva espacio en memoria para la estructura de datos que almacena toda la información de la instancia del protocolo.

5.2.4.2. Ronda 1-2

Durante la Ronda 1-2, se intercambian dos tipos de mensajes:

- Mensaje M, que contiene una clave pública, un texto cifrado, y los identificadores de los participantes U_A y U_B. Los tamaños de U_A y U_B se ha fijado a 20 bytes.
- Mensaje M', que contiene dos textos cifrados y, U_A y U_B.

La Figura 5.4 muestra la composición de los mensajes M y M'. Su tamaño depende del KEM en uso (ver Tabla 5.2).

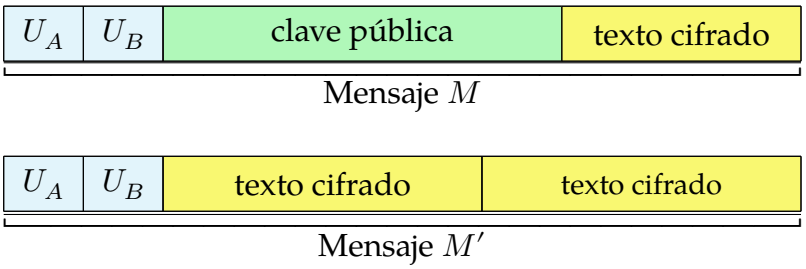


Figura 5.4: Mensajes enviados durante Ronda 1-2 en la transformación FSXY.

5.2.4.3. Ronda 3

Durante la Ronda 3, se transmiten los mensajes M_i^1, $i = 0, \ldots, n-1$ al resto de participantes. El mensaje contiene el identificador del usuario U_i, los textos cifrados del KEM y DEM, y la *tag* del DEM. La aleatoriedad r_i contiene el *nonce* del KEM y el IV del DEM. El tamaño de U_i se ha fijado a 20 bytes y $i \mid\mid X_i$ se puede cifrar con 36 bytes. La *tag* es de 16 bytes.

5.2.4.4. Ronda 4

En la Ronda 4, se retransmiten n mensajes M_i^2. Este mensaje contiene a U_i, X_i y la aleatoriedad r_i. El tamaño de X_i es 32 bytes el IV es 12 bytes. El tamaño del *nonce* del KEM depende del KEM usado (ver Tabla 5.2).

La clave de sesión `sk` y el identificador de la sesión `sid` se genera a partir de la clave maestra K utilizando la función *hash* SHA3-512, donde los primeros 32 bytes se corresponden con la clave de sesión `sk` y los últimos 32 bytes es el identificador de la sesión `sid`.

La Figura 5.5 muestra una ejecución del protocolo GAKE para Kyber1024 y Classic-McEliece-8192128f con 100 participantes.

```
> ./test_gake 100 Kyber1024
Round 1-2
Round 3
Round 4
All keys are equal!
Session key: ef169be7a5f6b717253827c4825c3397fc094aebdf963bfba1a4cc790adb931a
Session id:  83011836db7eaeb851474b24e70b78166d7bd8b9c960c2fa9190ec53865ab4e8


Time stats
        Init time      : 0.016s (4.35%)
        Round 1-2 time : 0.047s (13.04%)
        Round 3 time   : 0.016s (4.35%)
        Round 4 time   : 0.281s (78.26%)
        Total time     : 0.359s (100.00%)
```

(a) Kyber1024

```
> ./test_gake 100 Classic-McEliece-8192128f
Round 1-2
Round 3
Round 4
All keys are equal!
Session key: b5ad675e622d95d71b8fa5f5e6955a3f58eb89be17de357117332171b2de59d5
Session id:  554737f22ee28a4e4b2907ceff803188cc63b697679c31e03a38276758980b2f


Time stats
        Init time      : 38.328s (33.56%)
        Round 1-2 time : 74.750s (65.44%)
        Round 3 time   : 0.016s (0.01%)
        Round 4 time   : 1.125s (0.98%)
        Total time     : 114.219s (100.00%)
```

(b) Classic-McEliece-8192128f

Figura 5.5: Ejecución del protocolo GAKE con 100 participantes para dos KEM diferentes.

5.3. Entorno de pruebas

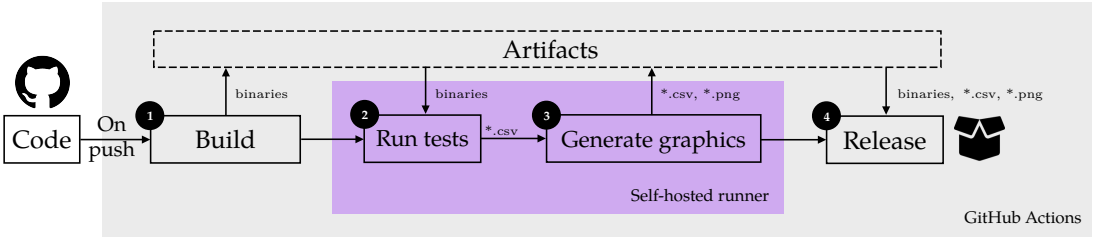


Figura 5.6: *Workflow* para el protocolo GAKE basado en la transformación FSXY.

Como en el capítulo anterior, se ha desarrollado un *workflow* de GitHub para reproducir los experimentos en un entorno aislado. El *workflow* se muestra en la Figura 5.6 y se ejecuta en un *runner* con Ubuntu 20.04 hospedado en GitHub Actions. Este consta de cuatro pasos:

1. **Build**: construye todos los binarios y las librerías de las que depende. La Figura 5.7 muestra este proceso.

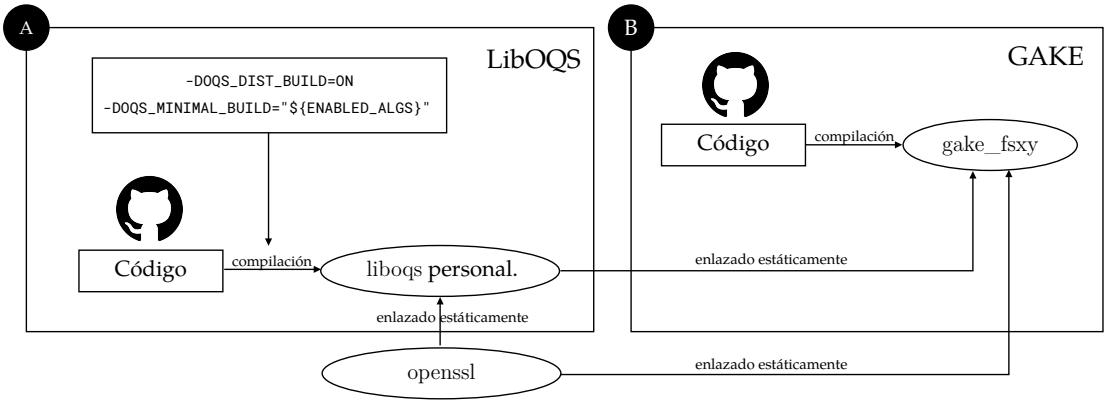


Figura 5.7: Construcción de la librería personalizada de LibOQS y la compilación del protocolo GAKE con la transformación FSXY.

   A. Se construye una versión personalizada de LibOQS [115] a partir de su versión `v0.7.0`. Esta contiene versiones modificadas de todos los KEM para ser deterministas y poder almacenar la aleatoriedad del KEM durante la generación del compromiso en la Ronda 3 del protocolo. La construcción de esta librería se ha automatizado con CMake, habilitando las opciones
      - `-DOQS_DIST_BUILD=ON` y,

- `-DOQS_MINIMAL_BUILD="${ENABLED_ALGS}"`,

donde la variable `ENABLED_ALGS` es un *array* que contiene los nombres de los KEM que se desea habilitar en LibOQS (los detalles se pueden ver en [104]). De hecho, sólo se habilitan los esquemas en la Tabla 5.2. Además, OpenSSL 1.1.1f se enlaza estáticamente, que es una dependencia requerida por LibOQS. Con estas opciones habilitadas, se genera una librería estática usando `gcc` como compilador de C.

B. El protocolo GAKE se construye con CMake y `gcc` como en el paso anterior, habilitando las opciones `-O3` y `-fwrapv` en este último. Las librería personalizada de LibOQS y OpenSSL 1.1.1f se enlazan estáticamente. La última se usa para implementar AES256-GCM en el esquema de compromiso. Una vez compilado, se lanza una serie de *tests* con `ctest` para garantizar que los binarios generados funcionan adecuadamente. Estos incluyen el correcto funcionamiento de las primitivas requeridas que integran el protocolo: AES256-GCM, el AKE, el esquema de compromiso, y la propia implementación del protocolo GAKE para cada uno de las instanciaciones de los KEM en LibOQS.

2. **Run tests**: este paso mide el rendimiento de cada una de las primitivas del protocolo GAKE. El rendimiento se mide en términos del número de ciclos consumidos por la CPU y el tiempo de ejecución. Para este propósito, empleamos la cabecera `ds_benchmark.h` incluida en LibOQS, que se puede encontrar en `https://github.com/open-quantum-safe/liboqs/blob/0.7.0/tests/ds_benchmark.h`. Esta implementa dos macros para medir el rendimiento:

   - `TIME_OPERATION_ITERATIONS`: ejecuta una porción de código un número dado de iteraciones.
   - `TIME_OPERATION_SECONDS`: ejecuta una porción de código durante un número dado de segundos.

   Todos los experimentos se ejecutan con la primera macro.

   Este se ejecuta en un *runner* autoalojado con Ubuntu 20.04 en WSL2 [81] en Windows 10 con las especificaciones dadas en la Tabla 5.4. Los *tests* definidos en este paso son los siguientes:

   - `test_speed_kem`: este *test* mide el rendimiento en ciclos de CPU y tiempo de ejecución para cada uno de los KEM implementados de la Tabla 5.2. Además, se miden independientemente, la generación de clave, la encapsulación y la desencapsulación. El resultado es la media de 10 000 iteraciones.

Tabla 5.4: Especificaciones de *hardware* para el *runner* autoalojado para la transformación FSXY.

Característica	Valor
Sistema Operativo	Ubuntu 20.04.1 LTS
CPU	i7-6700HQ@2.60 GHz
RAM	16 GB
Extensiones de CPU habilitadas	AVX2, BMI2, POPCNT

- `test_speed_ake`: mide el rendimiento de la transformación FSXY para cada uno de los KEM de LibOQS. Se mide el rendimiento de cada uno de los algoritmos que componen la transformación. El resultado es la media de la ejecución de 10 000 iteraciones.
- `test_speed_commitment`: mide el rendimiento del esquema de compromiso, a partir de los KEM implementados en LibOQS. El DEM se ha fijado a AES256-GCM. Se ejecutan 10 000 iteraciones de la generación de clave, la generación del compromiso y la comprobación del mismo.
- `test_speed_gake`: mide el rendimiento del protocolo GAKE para cada uno de los KEM implementados en LibOQS, en función del número de participantes en el protocolo, n. Se lanza el *test* para $n = 2, 2^2, \dots, 2^{11} = 2048$. Además, se mide el rendimiento de cada una de las rondas individualmente.

Todos los *tests* mencionados anteriormente generan tablas que son convertidas a formato CSV, que son procesados en pasos posteriores.

3. **Generate graphics**: este paso genera los gráficos de la Sección 5.4. Los gráficos se han representado usando el lenguaje de programación Python y la librería de visualización `seaborn` [121]. Todos los gráficos se guardan con el formato png.

4. **Release**: crea una nueva *release* en GitHub con todos los ficheros que se han generado durante el proceso, que se ha ido almacenando como artefactos en GitHub Actions [52]: binarios, gráficos, y ficheros CSV.

5.4. Resultados experimentales

A continuación, describimos los resultados experimentales obtenidos de los *tests* obtenidos anteriormente. Nótese que sólo comparamos las implementacio-

nes optimizadas de cada instanciación de cada KEM (ver Tabla 5.3).

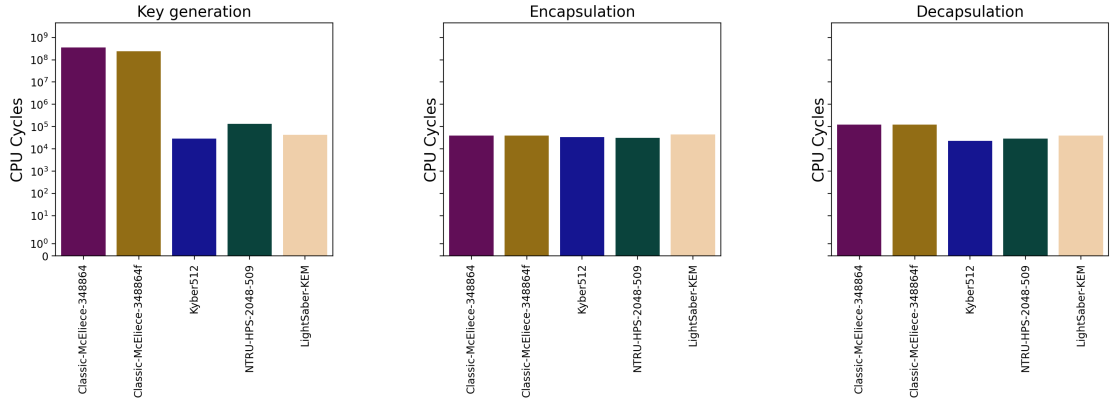
La Figura 5.8 muestra el rendimiento de cada algoritmo del KEM por cada nivel de seguridad que establece el concurso del NIST (ver Tabla 2.2). Se puede observar que, para todos los niveles de seguridad, el algoritmo `KeyGen` es significativamente más lento en Classic McEliece que el resto de KEM. Esto se debe al tamaño enorme de las claves de Classic McEliece (ver Tabla 5.2 para los tamaños concretos). El algoritmo `Encaps` no muestra una diferencia significativa, mientras el algoritmo `Decaps` muestra la misma diferencia entre Classic McEliece frente al resto de KEM, pero no es tan significativa como en el caso del algoritmo `KeyGen`. En resumen, se muestra una clara división en rendimiento entre los KEM de retículos (Kyber, NTRU y Saber) frente a los de códigos (Classic McEliece).

La Figura 5.9 muestra el rendimiento del protocolo AKE 2-parte resultado de la transformación `FSXY` (Figura 5.1). Se puede observar que, en cada nivel de seguridad y cada algoritmo del AKE, la diferencia de rendimiento entre Classic McEliece y el resto de los esquemas es significativo. Esto se debe a que el AKE depende del KEM, siendo los algoritmos de `KeyGen` y `Decaps` significativamente más lentos en Classic McEliece que en el resto de KEM.

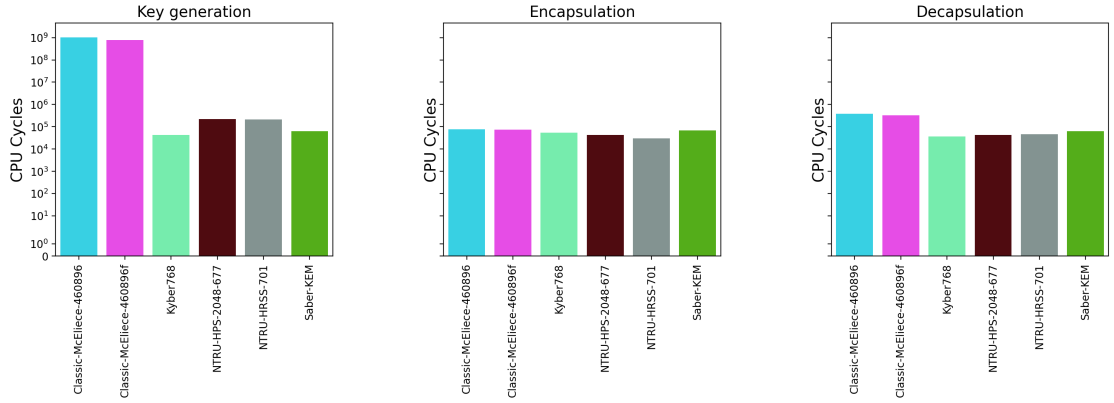
Con respecto al esquema de compromiso (Figura 5.10), Classic McEliece es más rápido durante la fase de inicialización `Init`. Esto se debe principalmente a que los textos cifrados de Classic McEliece son más pequeños que los del resto (ver Tabla 5.2 para los tamaños concretos). Los algoritmos `Commit` y `Check` tienen un mayor rendimiento en NTRU, en cualquier nivel de seguridad.

La Figura 5.11 muestra el rendimiento del protocolo GAKE en cada ronda del mismo. La ronda `Init` inicializa la estructura de datos necesaria para almacenar las variables de la instancia del protocolo y genera las claves de larga duración de autenticación. Se puede observar que, Kyber es más eficiente que el resto de esquemas, en cualquier nivel de seguridad. En la Ronda 1-2 (ejecución del AKE) y Ronda 3 (generación del compromiso), se produce el mismo comportamiento: las instanciaciones de Kyber ofrecen el mejor rendimiento. Finalmente, en la Ronda 4 (comprobación del compromiso, derivación de la clave maestra, y generación de la clave de sesión) el rendimiento en el nivel 1 de seguridad es muy similar entre las instanciaciones. El esquema más eficiente en el nivel 3 de seguridad es NTRU-HPS-2048-677 y NTRU-HPS-4096-821 para nivel 5.

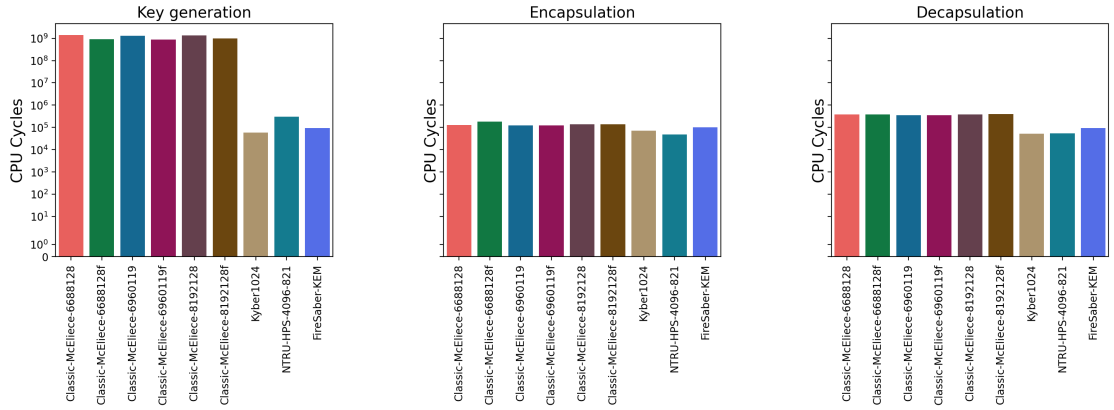
La Figura 5.12 muestra el rendimiento del protocolo GAKE en función del número de participantes. Se puede notar, que en todos los niveles de seguridad, Classic McEliece es significativamente menos eficiente que el resto de KEM y esto empeora a medida que se aumenta el número de participantes. El KEM más eficiente en términos de escalabilidad es Kyber512 en el nivel 1; Kyber768 y NTRU-HRSS-701 en el nivel 3 y, Kyber1024 en el nivel 5 de seguridad.



(a) Nivel 1

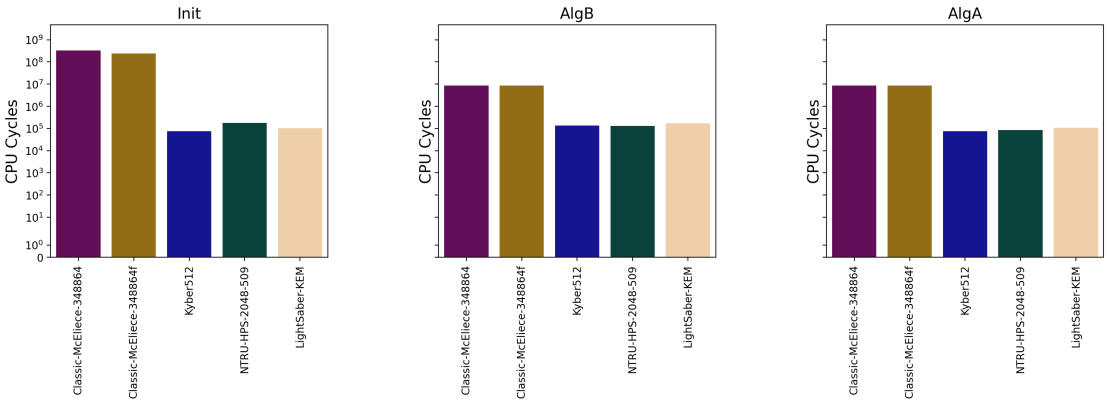


(b) Nivel 3

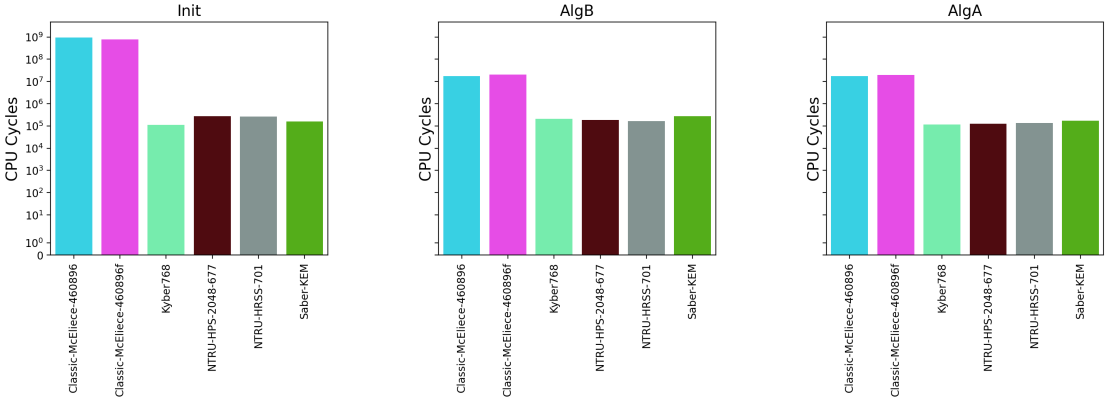


(c) Nivel 5

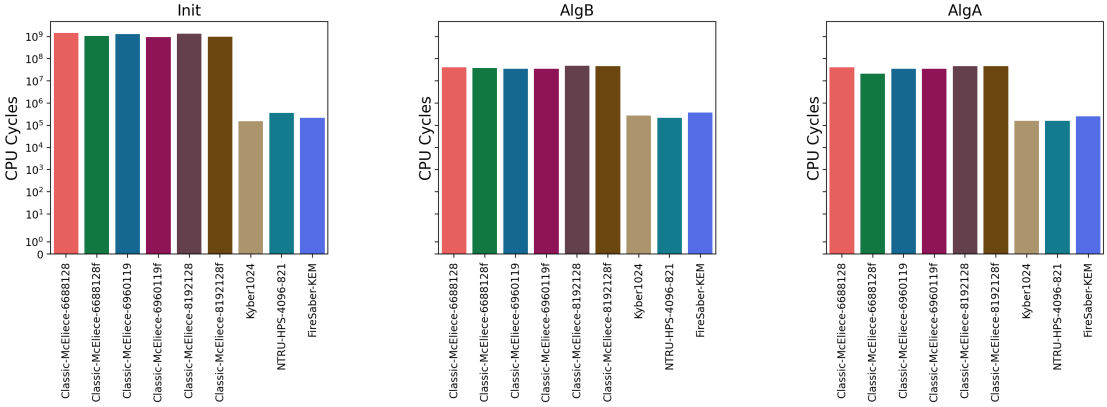
Figura 5.8: Rendimiento en ciclos de CPU para cada uno de los algoritmos de los KEM, divididos en niveles 1, 3 y 5 de seguridad.



(a) Nivel 1

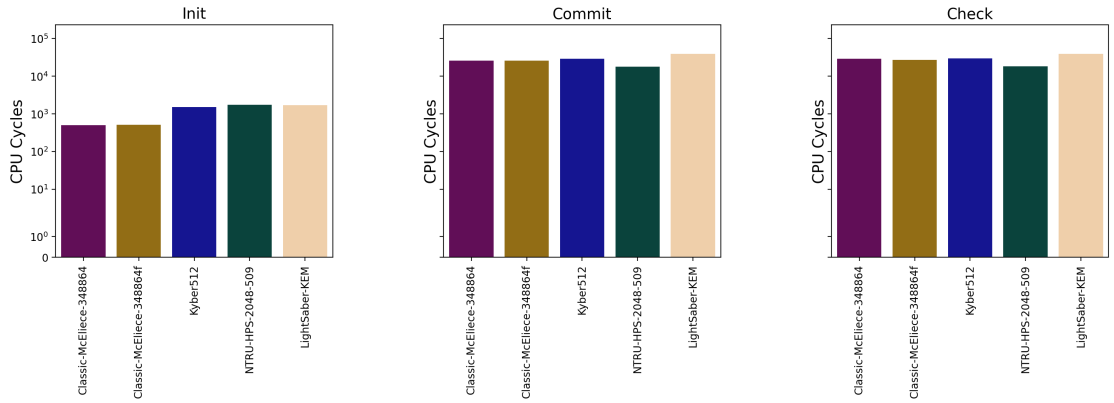


(b) Nivel 3

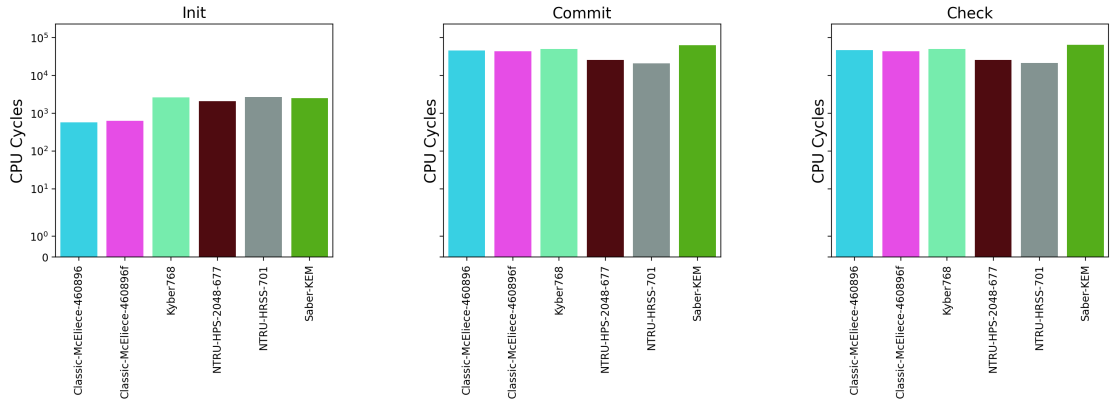


(c) Nivel 5

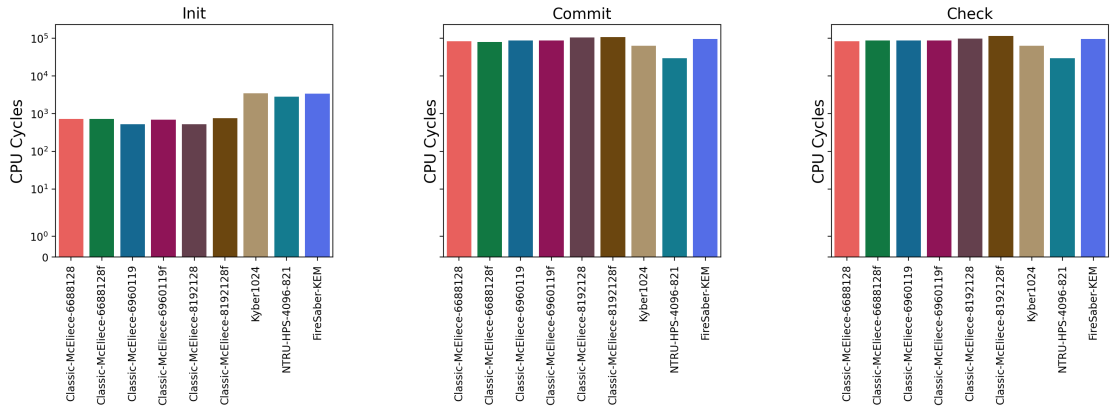
Figura 5.9: Rendimiento en ciclos de CPU para cada uno de los algoritmos del protocolo AKE 2-parte (`Init`, `AlgB` y `AlgA`), divididos en niveles 1, 3 y 5 de seguridad.



(a) Nivel 1

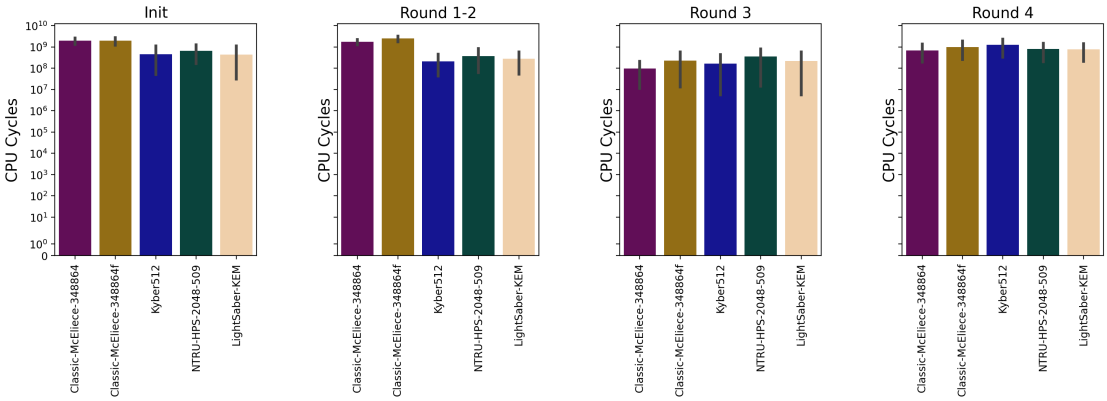


(b) Nivel 3

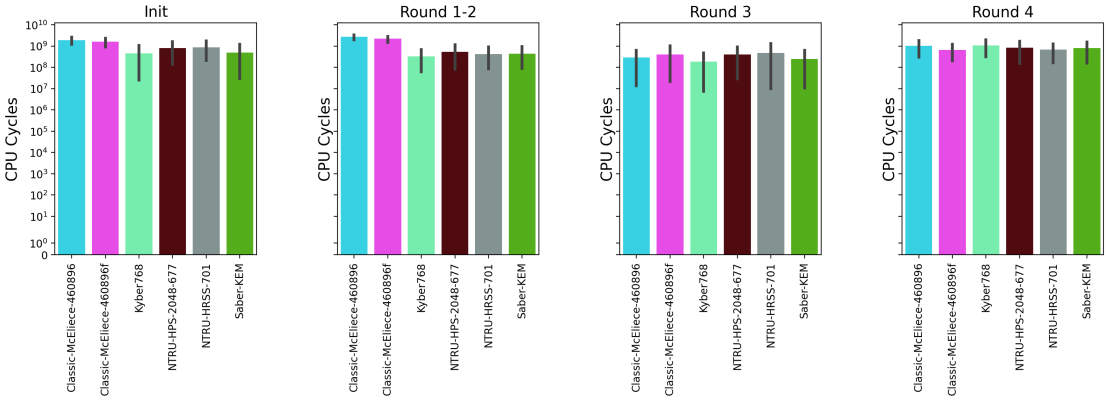


(c) Nivel 5

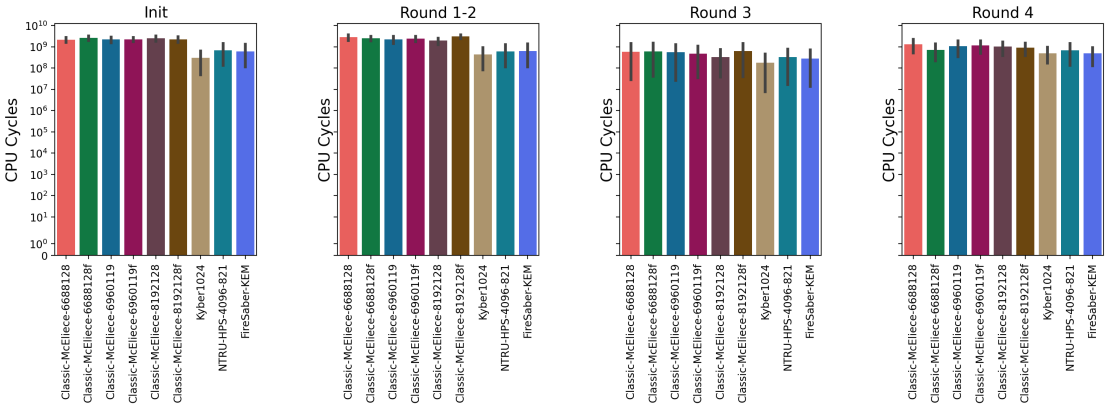
Figura 5.10: Rendimiento en ciclos de CPU para cada uno de los algoritmos del esquema de compromiso (`Init`, `Commit` y `Check`), divididos en niveles 1, 3 y 5 de seguridad.



(a) Nivel 1

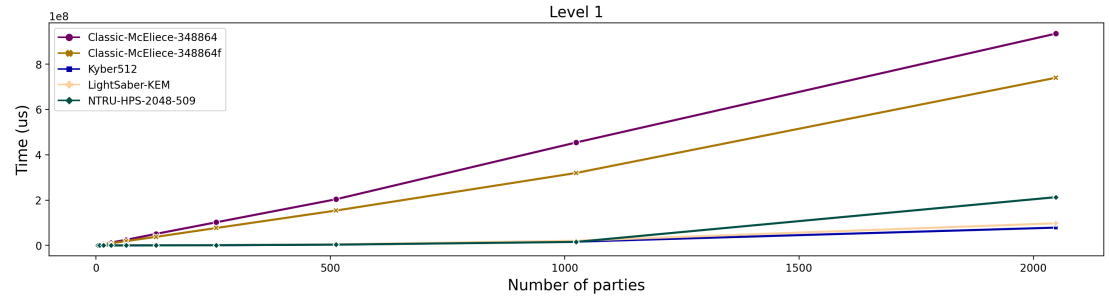


(b) Nivel 3

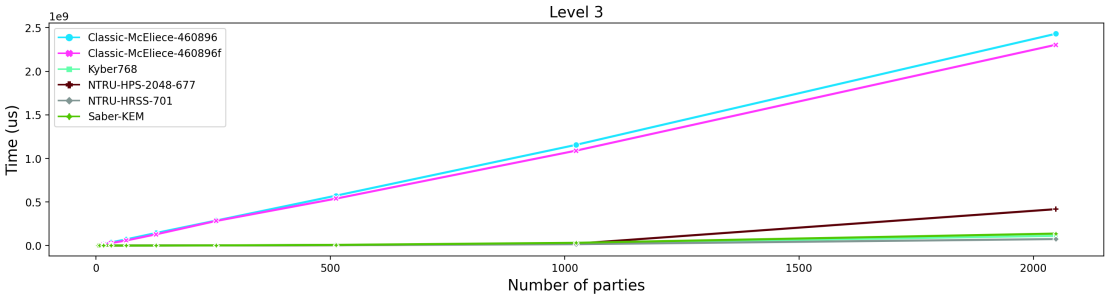


(c) Nivel 5

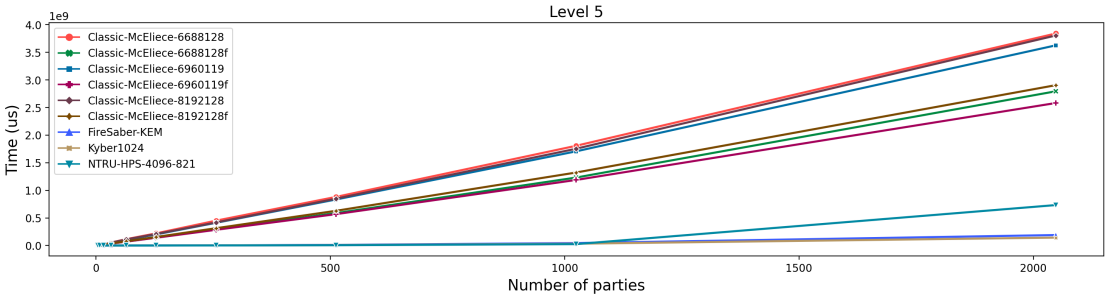
Figura 5.11: Rendimiento en ciclos de CPU para cada una de las rondas del protocolo GAKE (Init, Ronda 1-2, Ronda 3, y Ronda 4), divididos en niveles 1, 3 y 5 de seguridad.



(a) Nivel 1



(b) Nivel 3



(c) Nivel 5

Figura 5.12: Rendimiento, en tiempo de ejecución total, del protocolo GAKE en función del número de participantes, separados por el nivel de seguridad del KEM (niveles 1, 3 y 5).

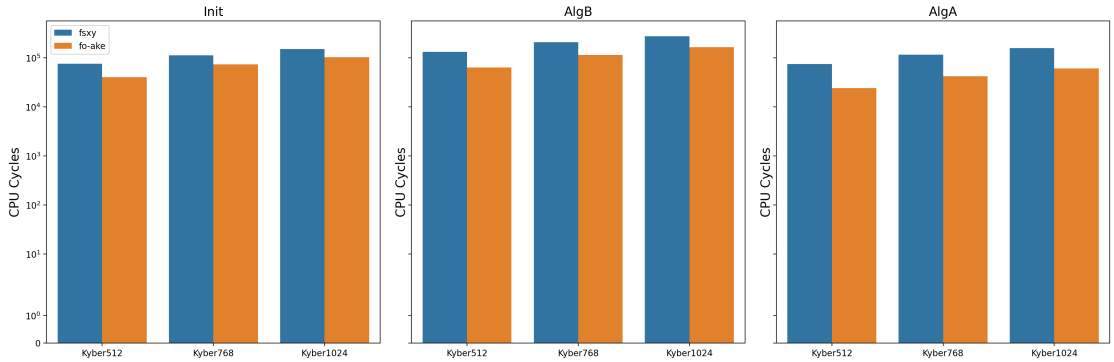
5.4.1. Comparativa entre FSXY y $\mathsf{FO}_{\mathsf{AKE}}$

A continuación, comparamos el rendimiento de las transformaciones FSXY frente a $\mathsf{FO}_{\mathsf{AKE}}$ (Sección 3.1.1) sobre Kyber. Recordemos que esta última se demuestra segura en el QROM, pero no se puede aplicar a cualquier KEM, sino sólo a aquellos que cumplen la propiedad de DS (Definición 3.1). Nótese que ambas transformaciones usan una estructura de 2 mensajes con la única diferencia de que en la transformación FSXY se envían en ambos mensajes U_A y U_B, haciendo que se produzcan mensajes de mayor tamaño que la transformación $\mathsf{FO}_{\mathsf{AKE}}$.

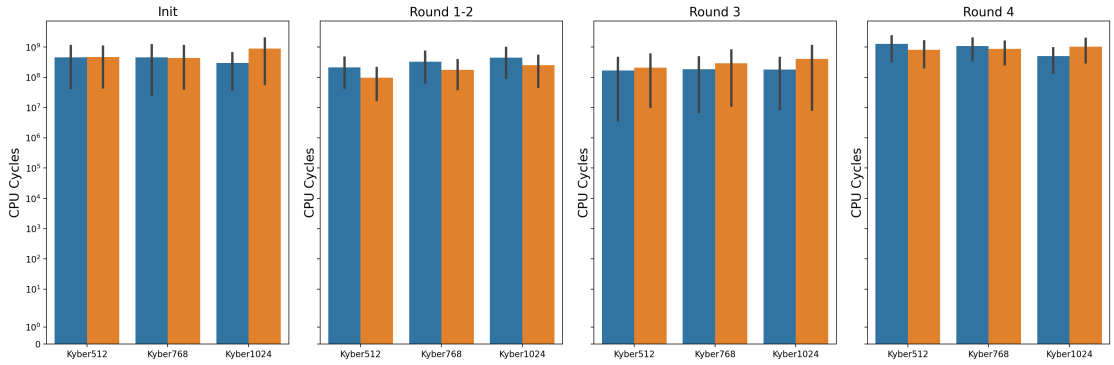
La Figura 5.13 muestra la comparación entre estas dos transformaciones sobre Kyber. Se puede observar que la transformación $\mathsf{FO}_{\mathsf{AKE}}$ tiene un mejor rendimiento que FSXY. Los resultados muestran que, si Kyber se aplica como KEM en este protocolo, el AKE resultante de $\mathsf{FO}_{\mathsf{AKE}}$ debería ser considerado en vez FSXY, proveyendo, además, un mayor nivel de seguridad al ser demostrado seguro en el QROM.

Resumen del capítulo

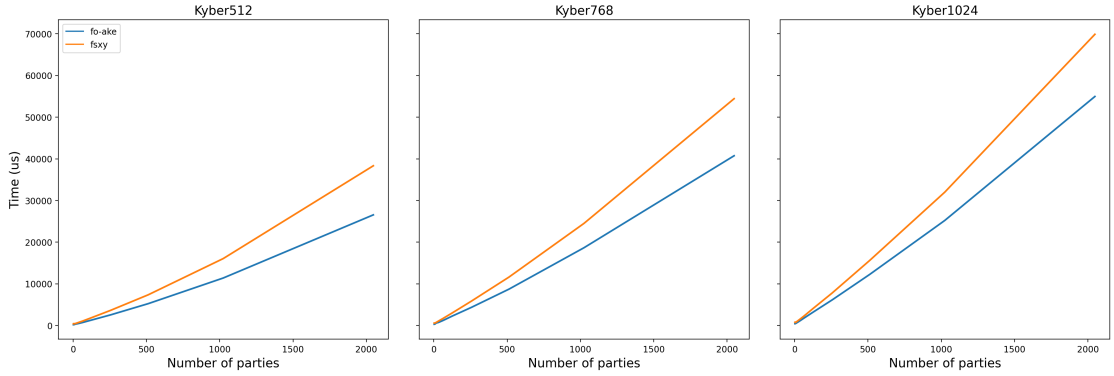
Este capítulo muestra cómo es posible extender el protocolo GAKE de los capítulos previos a otros KEM finalistas del concurso del NIST, más allá de Kyber. Esto ha sido posible porque la transformación FSXY, que no requiere ningún tipo de propiedad adicional para aplicarse. Se ha implementado el protocolo GAKE con LibOQS, mostrando experimentalmente que Classic McEliece no es adecuado para este protocolo porque es significativamente más lento que el resto de KEM finalistas. El KEM más apropiado, atendiendo a su escalabilidad, es Kyber 512 en el nivel 1, Kyber768 y NTRU-HRSS-701 para nivel 3 y, Kyber1024 para nivel 5.



(a) Rendimiento de cada uno de los algoritmos del KEM



(b) Rendimiento en cada ronda del protocolo GAKE



(c) Rendimiento en escalabilidad

Figura 5.13: Comparativa entre las transformaciones FSXY y $\mathsf{FO}_{\mathsf{AKE}}$.

Capítulo

6

Resultados y discusión general

Esta tesis se centra en los esquemas finalistas del concurso de estandarización de criptografía postcuántica del NIST. Nuestro análisis se enfoca en los mecanismos de encapsulación de clave (KEM), estudiando también soluciones para firma digital.

El núcleo de la tesis es el diseño e implementación de protocolos autenticados de intercambio de clave en grupo (GAKE) postcuánticos empleando los KEM finalistas del concurso del NIST, evitando el uso de las costosas firmas postcuánticas. Los protocolos GAKE tienen como objetivo acordar una clave de sesión entre un grupo de participantes a través de un canal inseguro. Esta clave se usa, por ejemplo, para cifrar las comunicaciones posteriores haciendo uso de un esquema de cifrado simétrico.

Los principales resultados de la tesis se describen a continuación:

- En el Capítulo 3, hemos desarrollado un protocolo de intercambio de clave autenticado en grupo postcuántico a partir del compilador de Abdalla et al. [1] y Kyber [22], uno de los KEM finalistas del concurso de estandarización del NIST. Este protocolo se demuestra seguro en el *Quantum Random Oracle Model* (QROM), un modelo de seguridad más fuerte que

el que tradicionalmente se considera, el *Random Oracle Model* (ROM). El QROM captura la capacidad de un adversario de realizar cómputos cuánticos relacionados con funciones *hash*, tales como realizar una búsqueda de colisiones en superposición.

- El protocolo depende de dos primitivas: un esquema de compromiso y un esquema de clave autenticado (AKE) 2-parte. Ambas se construyen a partir de las transformaciones de Hövelmanns et al. [63, 64], que se demuestran seguras en el QROM. Estas reciben el nombre de $\mathsf{FO}_{\mathsf{AKE}}$ y $\mathsf{FO}_m^{\not\perp}$ y, requieren que se cumpla la propiedad de *Disjoint Simulatability* (DS), cuya formalización viene dada por la Definición 3.1.
- Demostramos que, `Kyber.CPA′`, el PKE IND-CPA a partir del cual se obtiene `Kyber`, cumple la propiedad de *Disjoint Simulatability*. Este resultado es imprescindible para aplicar las transformaciones anteriores (Teorema 3.1).
- El AKE 2-parte se construye a partir de `Kyber.CPA′` aplicando la transformación $\mathsf{FO}_{\mathsf{AKE}}$, que resulta en un AKE seguro en el QROM.
- El esquema de compromiso se construye a partir de `Kyber.CPA′` mediante la transformación $\mathsf{FO}_m^{\not\perp}$, dando lugar a un KEM IND-CCA llamado $\mathtt{Kyber}^{\not\perp}$. Este se emplea para construir un PKE IND-CCA denominado $\mathtt{Kyber}^{\not\perp}\mathtt{.PKE}$ a través de la aproximación KEM/DEM. Este último, se puede emplear como esquema de compromiso tal y como se indica en [1].
- Finalmente, demostramos que el protocolo GAKE diseñado es seguro en el QROM (Teorema 3.5), previamente habiendo probado que es seguro en el ROM (Teorema 3.4).

- El Capítulo 4 implementa el protocolo GAKE anterior, utilizando entre otros recursos, el código C99 publicado por los autores de Kyber en GitHub. Nuestra implementación se encuentra disponible en `https://github.com/jiep/kyber-gake`. En concreto:
  - Se ha modificado el código para aplicar las transformaciones $\mathsf{FO}_{\mathsf{AKE}}$ y $\mathsf{FO}_m^{\not\perp}$, siendo el protocolo instanciado en tres niveles de seguridad, tal y como hacen los autores de Kyber, llamados Kyber512 (menor nivel de seguridad), Kyber768 y Kyber1024 (mayor nivel de seguridad). Las funciones *hash* de las primitivas pertenecen a `SHA-3`, procedente de OpenSSL.

  - De igual forma, se implementan las primitivas del protocolo con sus homólogos seguros en el ROM, propuestos en [22]. Se ha comparado el rendimiento de cada una de ellas, mostrando experimentalmente que, en este protocolo, un modelo de seguridad más fuerte como QROM no introduce una pérdida de rendimiento.
  - Asimismo, se compara el rendimiento de un par de DEM para obtener el mejor candidato para realizar el compromiso. Se comparan `AES256-GCM` y `CHACHA20-POLY1305`, sin apenas diferencias significativas de rendimiento, eligiendo el primero por estar estandarizado por el NIST.

- En el Capítulo 5, se extiende el protocolo GAKE más allá de Kyber, es decir, sin depender que el KEM subyacente tenga que cumplir la propiedad de DS, que reduce su aplicabilidad. Para ello, se emplea la transformación `FSXY` [46], que es segura en el ROM. Esta transformación es análoga a $\texttt{FO}_{\texttt{AKE}}$, pero convirtiendo un KEM en un AKE 2-parte. Los KEM sólo deben cumplir que son IND-CCA, una condición que se da en todos los finalistas del concurso del NIST.

  - Esta nueva variante del protocolo se implementa con la librería *open source* LibOQS para hacerlo extensible a los KEM finalistas del NIST: Classic McEliece, Kyber, NTRU y Saber. De igual forma que en la implementación anterior, se implementa en C99 usando las funciones *hash* de `SHA-3`, proporcionadas por LibOQS. El código se encuentra disponible públicamente en `https://github.com/jiep/pq-gake-fsxy`.
  - Se muestra, experimentalmente, que existe una clara diferencia de rendimiento entre los esquemas basados en retículos (Kyber, NTRU y Saber) frente a los esquemas basados en códigos (Classic Eliece), haciendo que este último no sea adecuado para este protocolo GAKE, debido a su rendimiento significativamente inferior, principalmente a causa de que los tamaños de clave pública son notablemente mayores en los KEM finalistas basados en códigos. En términos de escalabilidad, la mejor opción es Kyber512 en el nivel 1 (menor nivel de seguridad); Kyber768 y NTRU-HRSS-701 en el nivel 3; y Kyber1024 en el nivel 5 (mayor nivel de seguridad).
  - Por último, se compara el rendimiento de las transformaciones `FSXY` y $\texttt{FO}_{\texttt{AKE}}$ sobre Kyber. Esta última es más eficiente que `FSXY`, siendo preferible el uso de $\texttt{FO}_{\texttt{AKE}}$ en Kyber, siendo, además, seguro en el QROM, un modelo de seguridad más fuerte.

- En la Sección 2.3.2.1, analizamos, para obtener una visión global del escenario postcuántico, los ataques sufridos por un esquema de firma digital postcuántico llamado WalnutDSA, también candidato en el concurso de estandarización del NIST. Este diseño basa su seguridad en problemas que proceden de un área no contemplada en los otros esquemas estudiados: la teoría de trenzas.

6.1. Trabajo futuro

El trabajo realizado en esta tesis se puede extender tanto en la parte teórica como en las implementaciones.

- **Análisis teórico**.
  - Extensión del protocolo GAKE para su implementación con otro tipo de claves de larga duración de autenticación, como contraseñas o basadas en identidad.
  - Conversión del protocolo en dinámico, es decir, permitir añadir y eliminar participantes en el grupo sin volver a ejecutar el protocolo de nuevo. Esto, posiblemente, requiera romper la estructura de anillo que asumimos presentan los participantes en el protocolo, en favor de una estructura que permita el dinamismo de los participantes.
  - Creación de una transformación genérica, segura en el QROM, que convierta un PKE OW-CPA en un AKE 2-parte, que permita aplicarse a esquemas como Classic McEliece.
- **Implementaciones.**
  - Extensión a entornos con redes reales de comunicaciones y a dispositivos con pocos recursos computacionales y de red, tales como dispositivos IoT.
  - Evaluación del impacto de ataques de canal lateral (*side channels*).
  - Comparación del consumo de recursos (por ejemplo, energéticos) del protocolo GAKE, en función del KEM empleado.
  - Análisis del comportamiento del protocolo cuando se emplean implementaciones optimizadas y no optimizadas del KEM. Detección de cuellos de botella y retrasos significativos debido a diferencias en el rendimiento. Estimación del umbral de participantes a partir del cual el protocolo se vuelve inviable o deja de funcionar correctamente.

Capítulo

7

Conclusiones generales

A continuación, resumimos las conclusiones generales de la tesis:

- Es posible construir un protocolo de intercambio de clave autenticado en grupo (GAKE) postcuántico a partir de herramientas conocidas y validadas tales como mecanismos de encapsulación de clave (KEM), funciones *hash* y esquemas de cifrado simétrico.
- El GAKE compilado propuesto puede demostrarse seguro en un marco muy general en el que encajan todos los KEM finalistas del concurso de estandarización del NIST.
- De los diseños testados, el peor en la práctica es Classic McEliece debido al gran tamaño de sus claves públicas. En términos de escalabilidad, las mejores opciones son Kyber512 en nivel 1 (menor nivel de seguridad); Kyber768 y NTRU-HRSS-701 en nivel 3 (nivel medio) y, Kyber1024 en el nivel 5 (mayor nivel de seguridad).
- La demostración de seguridad del protocolo GAKE en un modelo de seguridad más fuerte como QROM no introduce una pérdida de rendimien-

to; al contrario, las primitivas implementadas en el modelo de seguridad ROM son menos eficientes.

- En el caso de Kyber, la transformación $\mathsf{FO}_{\mathsf{AKE}}$ logra un mayor rendimiento que FSXY, siendo la primera segura en el QROM, un modelo de seguridad más fuerte que el ROM, considerado en FSXY.

Bibliografía

[1] Michel Abdalla, Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "(Password) Authenticated Key Establishment: From 2-Party to Group". In: *TCC*. Vol. 4392. Lecture Notes in Computer Science. Springer, 2007, pp. 499–514.

[2] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Daan Sprenkels. "Faster Kyber and Dilithium on the Cortex-M4". In: *IACR Cryptol. ePrint Arch.* (2022), p. 112.

[3] *Advanced Encryption Standard (AES)*. Tech. rep. Nov. 2001. DOI: 10.6028/nist.fips.197. URL: https://doi.org/10.6028/nist.fips.197.

[4] Rashmi Agrawal, Lake Bu, and Michel A. Kinsy. "Quantum-Proof Lightweight McEliece Cryptosystem Co-processor Design". In: *2020 IEEE 38th International Conference on Computer Design (ICCD)*. 2020, pp. 73–79. DOI: 10.1109/ICCD50377.2020.00029.

[5] Erdem Alkim, Hülya Evkan, Norman Lahr, Ruben Niederhagen, and Richard Petri. "ISA Extensions for Finite Field Arithmetic". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 219–242. DOI: 10.46586/tches.v2020.i3.219-242. URL: https://doi.org/10.46586/tches.v2020.i3.219-242.

[6] Iris Anshel, Derek Atkins, Dorian Goldfeld, and Paul E Gunnells. "Defeating the Hart et al, Beullens-Blackburn, Kotov-Menshov-Ushakov, and

Merz-Petit Attacks on WalnutDSA (TM).” In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 472.

[7] Iris Anshel, Derek Atkins, Dorian Goldfeld, and Paul E. Gunnells. “WalnutDSA: A Quantum Resistant Digital Signature Algorithm”. In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 58.

[8] Daniel Apon, Dana Dachman-Soled, Huijing Gong, and Jonathan Katz. “Constant-Round Group Key Exchange from the Ring-LWE Assumption”. In: *PQCrypto*. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 189–205.

[9] ARM. *Cortex-M3 - Arm Developer*. URL: `https://developer.arm.com/Processors/Cortex-M3`.

[10] ARM. *Cortex-M4 - Arm Developer*. URL: `https://developer.arm.com/Processors/Cortex-M4`.

[11] Emil Artin. “Theory of braids.” English. In: *Annals of Mathematics (2)* 48 (1947), pp. 101–126. ISSN: 0003-486X; 1939-8980/e.

[12] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. “Authenticated Group Key Agreement and Friends”. In: *CCS*. ACM, 1998, pp. 17–26.

[13] Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri. “NIST Post-Quantum Cryptography- A Hardware Evaluation Study”. In: *IACR Cryptol. ePrint Arch.* (2019), p. 47. URL: `https://eprint.iacr.org/2019/047`.

[14] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. “Relations Among Notions of Security for Public-Key Encryption Schemes”. In: *CRYPTO*. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 26–45.

[15] Mihir Bellare, David Pointcheval, and Phillip Rogaway. “Authenticated Key Exchange Secure against Dictionary Attacks”. In: *EUROCRYPT*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 139–155.

[16] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche1, and Ronny Van Keer1. *Team Keccak*. `https://keccak.team`. 2022.

[17] Ward Beullens and Simon R. Blackburn. “Practical Attacks Against the Walnut Digital Signature Scheme”. In: *ASIACRYPT (1)*. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 35–61.

[18] Ward Beullens, Jan-Pieter D'Anvers, Andreas Hülsing, Tanja Lange, Lorenz Panny, Cyprien de Saint Guilhem, and Nigel P. Smart. *Post-Quantum Cryptography: Current state and quantum mitigation*. Tech. rep. https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation. ENISA, 2021.

[19] Joan S. Birman and James Cannon. *Braids, Links, and Mapping Class Groups. (AM-82)*. Princeton University Press, 1974. ISBN: 9780691081496. URL: http://www.jstor.org/stable/j.ctt1b9rzv3.

[20] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "Secure group key establishment revisited". In: *Int. J. Inf. Sec.* 6.4 (2007), pp. 243–254. DOI: 10.1007/s10207-007-0018-x. URL: https://doi.org/10.1007/s10207-007-0018-x.

[21] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. "Secure group key establishment revisited". In: *Int. J. Inf. Sec.* 6.4 (2007), pp. 243–254.

[22] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *EuroS&P*. IEEE, 2018, pp. 353–367.

[23] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine Van Vredendaal. "Masking Kyber: First- and Higher-Order Implementations". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Aug. 2021), pp. 173–214. DOI: 10.46586/tches.v2021.i4.173-214. URL: https://doi.org/10.46586/tches.v2021.i4.173-214.

[24] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. "Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4". In: *AFRICACRYPT*. Vol. 11627. Lecture Notes in Computer Science. Springer, 2019, pp. 209–228.

[25] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. "Efficient One-Round Key Exchange in the Standard Model". In: *ACISP*. Vol. 5107. Lecture Notes in Computer Science. Springer, 2008, pp. 69–83.

[26] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020.

[27] Mike Burmester and Yvo Desmedt. "A secure and scalable Group Key Exchange system". In: *Inf. Process. Lett.* 94.3 (2005), pp. 137–143. DOI: 10.1016/j.ipl.2005.01.003. URL: https://doi.org/10.1016/j.ipl.2005.01.003.

[28] Ran Canetti and Hugo Krawczyk. "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels". In: *EUROCRYPT*. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 453–474.

[29] André Chailloux, María Naya-Plasencia, and André Schrottenloher. "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography". In: *ASIACRYPT (2)*. Vol. 10625. Lecture Notes in Computer Science. Springer, 2017, pp. 211–240.

[30] Rakyong Choi, Dongyeon Hong, and Kwangjo Kim. "Constant-round Dynamic Group Key Exchange from RLWE Assumption". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 35.

[31] *Comments to WalnutDSA^{TM} proposal to NIST PQC project*. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf.

[32] Ronald Cramer and Victor Shoup. "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack". In: *SIAM J. Comput.* 33.1 (2003), pp. 167–226.

[33] Quynh H. Dang. *Secure Hash Standard*. Tech. rep. July 2015. DOI: 10.6028/nist.fips.180-4. URL: https://doi.org/10.6028/nist.fips.180-4.

[34] Viet Ba Dang, Farnoud Farahmand, Michal Andrzejczak, Kamyar Mohajerani, Duc Tri Nguyen, and Kris Gaj. "Implementation and Benchmarking of Round 2 Candidates in the NIST Post-Quantum Cryptography Standardization Process Using Hardware and Software/Hardware Co-design Approaches". In: *IACR Cryptol. ePrint Arch.* (2020), p. 795. URL: https://eprint.iacr.org/2020/795.

[35] Viet Ba Dang, Kamyar Mohajerani, and Kris Gaj. *High-Speed Hardware Architectures and Fair FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber*. URL: https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/gaj-high-speed-hardware-gmu-pqc2021.pdf.

[36] *Data Encryption Standard (DES)*. Tech. rep. 1999. URL: https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf.

[37] Gareth T. Davies, Herman Galteland, Kristian Gjøsteen, and Yao Jiang. "Cloud-Assisted Asynchronous Key Transport with Post-Quantum Security". In: *Information Security and Privacy - 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30 - December 2, 2020, Proceedings*. Vol. 12248. Lecture Notes in Computer Science. Springer, 2020, pp. 82–101. DOI: 10.1007/978-3-030-55304-3_5. URL: https://doi.org/10.1007/978-3-030-55304-3_5.

[38] Alexander W. Dent. "A Designer's Guide to KEMs". In: *IMACC*. Vol. 2898. Lecture Notes in Computer Science. Springer, 2003, pp. 133–151.

[39] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Trans. Inf. Theory* 22.6 (1976), pp. 644–654.

[40] Jintai Ding. *The latest Progress of PQC Competition in China - ETSI*. URL: https://docbox.etsi.org/Workshop/2019/201911_QSCWorkshop/TECHNICAL_TRACK/02_COLLABORATIVEEFFORTS/DING_CHONGQINGUNIVERSITY.pdf.

[41] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-Malleable Cryptography (Extended Abstract)". In: *STOC*. ACM, 1991, pp. 542–552.

[42] Morris J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Tech. rep. National Institute of Standards and Technology, July 2015. DOI: 10.6028/nist.fips.202. URL: https://doi.org/10.6028/nist.fips.202.

[43] Morris J. Dworkin. *SP 800-38C. Recommendation for block cipher modes of operation: the CCM Mode for Authentication and Confidentiality*. Tech. rep. 2007. DOI: 10.6028/nist.sp.800-38c. URL: https://doi.org/10.6028/nist.sp.800-38c.

[44] Morris J. Dworkin. *SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Tech. rep. Gaithersburg, MD, USA, 2007. DOI: 10.6028/nist.sp.800-38d.

[45] José Ignacio Escribano Pablos, María Isabel González Vasco, Misael Enrique Marriaga, and Ángel Luis Pérez del Pozo. "The Cracking of WalnutDSA: A Survey". In: *Symmetry* 11.9 (2019). ISSN: 2073-8994. DOI: 10.3390/sym11091072. URL: https://www.mdpi.com/2073-8994/11/9/1072.

[46] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. "Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism". In: *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. 2013, pp. 83–94.

[47] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. "Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices". In: *Designs, Codes and Cryptography* 76.3 (2015), pp. 469–504.

[48] Atsushi Fujioka, Katsuyuki Takashima, and Kazuki Yoneyama. "One-Round Authenticated Group Key Exchange from Isogenies". In: *ProvSec*. Vol. 11821. Lecture Notes in Computer Science. Springer, 2019, pp. 330–338.

[49] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure integration of asymmetric and symmetric encryption schemes". In: *CRYPTO'99*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 537–554.

[50] Satoshi Furukawa, Noboru Kunihiro, and Katsuyuki Takashima. "Multiparty Key Exchange Protocols from Supersingular Isogenies". In: *International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28-31, 2018*. IEEE, 2018, pp. 208–212. DOI: 10.23919/ISITA.2018.8664316. URL: https://doi.org/10.23919/ISITA.2018.8664316.

[51] *GitHub Actions. About continuous integration*. https://docs.github.com/es/actions/automating-builds-and-tests/about-continuous-integration. 2022.

[52] *GitHub Actions. Storing workflow data as artifacts*. https://docs.github.com/es/actions/advanced-guides/storing-workflow-data-as-artifacts. 2022.

[53] *GitHub Actions. Usage limits, billing, and Administration*. https://docs.github.com/en/actions/learn-github-actions/usage-limits-billing-and-administration. 2022.

[54] M. Choudary Gorantla, Colin Boyd, Juan Manuel González Nieto, and Mark Manulis. "Generic One Round Group Key Exchange in the Standard Model". In: *ICISC*. Vol. 5984. Lecture Notes in Computer Science. Springer, 2009, pp. 1–15.

[55] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[56] Qian Guo, Thomas Johansson, and Jing Yang. "A Novel CCA Attack Using Decryption Errors Against LAC". In: *ASIACRYPT (1)*. Vol. 11921. Lecture Notes in Computer Science. Springer, 2019, pp. 82–111.

[57] Wenbo Guo, Shuguo Li, and Liang Kong. "An Efficient Implementation of KYBER". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 69.3 (2022), pp. 1562–1566. DOI: 10.1109/TCSII.2021.3103184.

[58] Daniel Hart, DoHoon Kim, Giacomo Micheli, Guillermo Pascual-Perez, Christophe Petit, and Yuxuan Quek. "A Practical Cryptanalysis of WalnutDSA TM". In: *Public Key Cryptography (1)*. Vol. 10769. Lecture Notes in Computer Science. Springer, 2018, pp. 381–406.

[59] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Daan Sprenkels. "First-Order Masked Kyber on ARM Cortex-M4". In: *IACR Cryptol. ePrint Arch.* (2022), p. 58.

[60] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem". In: *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*. Vol. 1423. Lecture Notes in Computer Science. Springer, 1998, pp. 267–288. DOI: 10.1007/BFb0054868. URL: https://doi.org/10.1007/BFb0054868.

[61] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *TCC (1)*. Vol. 10677. Lecture Notes in Computer Science. Springer, 2017, pp. 341–371.

[62] Hector B. Hougaard and Atsuko Miyaji. "Authenticated logarithmic-order supersingular isogeny group key exchange". In: *International Journal of Information Security* 21.2 (2022), pp. 207–221.

[63] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. "Generic Authenticated Key Exchange in the Quantum Random Oracle Model". In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 928. URL: https://eprint.iacr.org/2018/928.

[64] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. "Generic Authenticated Key Exchange in the Quantum Random Oracle Model". In: *Public-Key Cryptography – PKC 2020*. Cham: Springer International Publishing, 2020, pp. 389–422. ISBN: 978-3-030-45388-6.

[65] James Howe, Thomas Prest, and Daniel Apon. "SoK: How (not) to Design and Implement Post-quantum Cryptography". In: *CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021*. Vol. 12704. Springer, 2021, pp. 444–477.

[66] Ingemar Ingemarsson, Donald Tang, and C. K. Wong. "A conference key distribution system". In: *IEEE Transactions on Information Theory* 28.5 (1982), pp. 714–719.

[67] RISC-V International. *RISC-V: The Free and Open RISC Instruction Set Architecture*. URL: https://riscv.org.

[68] ISO. *ISO C Standard 1999*. Tech. rep. ISO/IEC 9899:1999 draft. 1999. URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf.

[69] Arpan Jati, Naina Gupta, Anupam Chattopadhyay, and Somitra Kumar Sanadhya. "A Configurable Crystals-Kyber Hardware Implementation with Side-Channel Protection". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1189.

[70] Mike Just and Serge Vaudenay. "Authenticated Multi-Party Key Agreement". In: *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*. Vol. 1163. Lecture Notes in Computer Science. Springer, 1996, pp. 36–49. DOI: 10.1007/BFb0034833. URL: https://doi.org/10.1007/BFb0034833.

[71] Matthias J. Kannwischer and Richard Petri. *pqm4: NISTPQC Round 3 Results on the Cortex-M4*. URL: https://csrc.nist.gov/CSRC/media/Presentations/pqm4-nistpqc-round-3-results-on-the-cortex-m4/images-media/session-3-kannwischer-pqm4.pdf.

[72] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2015.

[73] Jonathan Katz and Moti Yung. "Scalable Protocols for Authenticated Group Key Exchange". In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Vol. 2729. Lecture Notes in Computer Sci-

ence. Springer, 2003, pp. 110–125. DOI: 10.1007/978-3-540-45146-4_7. URL: https://doi.org/10.1007/978-3-540-45146-4_7.

[74] Birgit Klein. "Conference key distribution protocols in distributed systems". In: *Proc. Codes and Ciphers: Cryptography and Coding IV, 1995* (1995), pp. 225–242.

[75] Matvei Kotov, Anton Menshov, and Alexander Ushakov. "Attack on Kayawood Protocol: Uncloaking Private Keys." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 604.

[76] Brian Koziel, Reza Azarderakhsh, Mehran Mozaffari Kermani, and David Jao. "Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.1 (2017), pp. 86–99. DOI: 10.1109/TCSI.2016.2611561.

[77] *LAC.PKE: LAttice-based Cryptosystems: Public Key Encryption*. URL: https://sfjs.cacrnet.org.cn/upload/5de89b6e38a93.pdf.

[78] Robert J McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[79] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. "Efficient Encryption From Random Quasi-Cyclic Codes". In: *IEEE Trans. Inf. Theory* 64.5 (2018), pp. 3927–3943.

[80] Simon-Philipp Merz and Christophe Petit. "Factoring Products of Braids via Garside Normal Form". In: *Public Key Cryptography (2)*. Vol. 11443. Lecture Notes in Computer Science. Springer, 2019, pp. 646–678.

[81] Microsoft. *What is the Windows Subsystem for Linux?* URL: https://docs.microsoft.com/en-us/windows/wsl/about.

[82] Michele Mosca. "Cybersecurity in an Era with Quantum Computers: Will We Be Ready?" In: *IEEE Secur. Priv.* 16.5 (2018), pp. 38–41. DOI: 10.1109/MSP.2018.3761723. URL: https://doi.org/10.1109/MSP.2018.3761723.

[83] mupq. *mupq GitHub organization*. URL: https://github.com/mupq.

[84] mupq. *Post-quantum crypto library for the ARM Cortex-M3*. URL: https://github.com/mupq/pqm3.

[85] mupq. *Post-quantum crypto library for the ARM Cortex-M4*. URL: https://github.com/mupq/pqm4.

[86] Junghyun Nam, Juryon Paik, and Dongho Won. "A security weakness in Abdalla et al.'s generic construction of a group key exchange protocol". In: *Inf. Sci.* 181.1 (2011), pp. 234–238.

[87] Harald Niederreiter. "Knapsack-type Cryptosystems And Algebraic Coding Theory". In: *Prob. Contr. Inform. Theory* 15.2 (1986), pp. 157–166.

[88] NIST. *Post-Quantum Cryptography. Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[89] NIST. *Post-Quantum Cryptography. Round 1 submissions.* URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/Round-1-Submissions.

[90] NIST. *Post-Quantum Cryptography. Round 3 submissions.* URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[91] Paul C. van Oorschot and Michael J. Wiener. "Parallel Collision Search with Cryptanalytic Applications". In: *J. Cryptology* 12.1 (1999), pp. 1–28.

[92] Raphael Overbeck and Nicolas Sendrier. "Code-based cryptography". In: *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 95–145. ISBN: 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7_4. URL: https://doi.org/10.1007/978-3-540-88702-7_4.

[93] Luis Paris. *Braid groups and Artin groups*. 2007. DOI: 10.48550/ARXIV.0711.2372. URL: https://arxiv.org/abs/0711.2372.

[94] Adrian Perrig. "Efficient collaborative key management protocols for secure autonomous group communication". In: *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*. 1999, pp. 192–202.

[95] Edoardo Persichetti, Rainer Steinwandt, and Adriana Suárez Corona. "From Key Encapsulation to Authenticated Group Key Establishment - A Compiler for Post-Quantum Primitives". In: *Entropy* 21.12 (2019), p. 1183.

[96] Joseph Pieprzyk and C-H Li. "Multiparty key agreement protocols". In: *IEE Proceedings-Computers and Digital Techniques* 147.4 (2000), pp. 229–236.

[97] PQClean. *GitHub repository*. URL: https://github.com/PQClean/PQClean.

[98] PQCRYPTO. *libpqcrypto*. URL: https://libpqcrypto.org.

[99] PQCRYPTO. *Post-quantum cryptography for long-term security PQCRYPTO ICT-645622*. URL: https://pqcrypto.eu.org.

[100] PQSlib. *Lightweight Post-Quantum Cryptography Library for Embedded, IoT, Secure Elements*. URL: https://pqshield.com/solutions/pqslib.

[101] Zhenhui Qin, Rui Tong, Xingjun Wu, Guoqiang Bai, Liji Wu, and Linlin Su. "A Compact Full Hardware Implementation of PQC Algorithm NTRU". In: *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. 2021, pp. 792–797. DOI: 10.1109/CISCE52179.2021.9446042.

[102] Oded Regev. "The Learning with Errors Problem (Invited Survey)". In: *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*. CCC '10. USA: IEEE Computer Society, 2010, pp. 191–204. ISBN: 9780769540603. DOI: 10.1109/CCC.2010.26. URL: https://doi.org/10.1109/CCC.2010.26.

[103] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126.

[104] Open Quantum Safe. *Algorithms in liboqs*. URL: https://openquantumsafe.org/liboqs/algorithms.

[105] Open Quantum Safe. *LibOQS Demos*. URL: https://github.com/open-quantum-safe/oqs-demos.

[106] Open Quantum Safe. *OQS-BoringSSL*. URL: https://github.com/open-quantum-safe/boringssl.

[107] Open Quantum Safe. *OQS-OpenSSH*. URL: https://github.com/open-quantum-safe/openssh.

[108] Open Quantum Safe. *OQS-OpenSSL*. URL: https://github.com/open-quantum-safe/openssl.

[109] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. "Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 520–551.

[110] Pakize Sanal, Emrah Karagoz, Hwajeong Seo, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "Kyber on ARM64: Compact Implementations of Kyber on 64-Bit ARM Cortex-A Processors". In: *Security and Privacy in Communication Networks*. Cham: Springer International Publishing, 2021, pp. 424–440. ISBN: 978-3-030-90022-9.

[111] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[112] Peter Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[113] Ruslan V. Skuratovskii and Aled Williams. "Application of commutator subgroups of Sylow 2-subgroups of alternating group and Miller-Moreno groups to Key Exchange Protocol". In: *IACR Cryptol. ePrint Arch.* (2020), p. 234.

[114] Deepraj Soni, Kanad Basu, Mohammed Nabeel, and Ramesh Karri. *A Hardware Evaluation Study of NIST Post-Quantum Cryptographic Signature schemes*. URL: https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/soni-hardware-evaluation.pdf.

[115] Douglas Stebila and Michele Mosca. "Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project". In: *SAC*. Vol. 10532. Lecture Notes in Computer Science. Springer, 2016, pp. 14–37.

[116] Michael Steiner, Gene Tsudik, and Michael Waidner. "Diffie-Hellman Key Distribution Extended to Group Communication". In: *ACM Conference on Computer and Communications Security*. ACM, 1996, pp. 31–37.

[117] Katsuyuki Takashima. "Post-Quantum Constant-Round Group Key Exchange from Static Assumptions". In: *International Symposium on Mathematics, Quantum Theory, and Cryptography*. Singapore: Springer Singapore, 2021, pp. 251–272.

[118] The OpenSSL Project. *OpenSSL: The Open Source toolkit for SSL/TLS*. www.openssl.org. 2022.

[119] María Isabel González Vasco, Ángel L. Pérez del Pozo, and Rainer Steinwandt. "Group Key Establishment in a Quantum-Future Scenario". In: *Informatica* 31.4 (2020), pp. 751–768. ISSN: 0868-4952. DOI: 10.15388/20-INFOR427.

[120] Jorge L. Villar. *Security Models Notes*. 2021. URL: https://web.mat.upc.edu/jorge.villar/doc/notes/DataProt/sec.html.

[121] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[122] Violetta Weger, Niklas Gassner, and Joachim Rosenthal. "A Survey on Code-Based Cryptography". In: *CoRR* abs/2201.07119 (2022).

[123] Qianhong Wu, Yi Mu, Willy Susilo, Bo Qin, and Josep Domingo-Ferrer. "Asymmetric Group Key Agreement". In: *EUROCRYPT*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 153–170.

[124] Jiafeng Xie, Kanad Basu, Kris Gaj, and Ujjwal Guin. "Special Session: The Recent Advance in Hardware Implementation of Post-Quantum Cryptography". In: *2020 IEEE 38th VLSI Test Symposium (VTS)*. 2020, pp. 1–10. DOI: 10.1109/VTS48691.2020.9107585.

[125] Yufei Xing and Shuguo Li. "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Feb. 2021), pp. 328–356. DOI: 10.46586/tches.v2021.i2.328-356. URL: https://doi.org/10.46586/tches.v2021.i2.328-356.

[126] Bo-Yin Yang and Ward Beullens. *WG 2 SD8 (Post-Quantum Cryptography) – Part 5: Multivariate Cryptography*. Tech. rep. https://www.din.de/resource/blob/721042/4f1941ac1de9685115cf53bc1a14ac61/sc27wg2-sd8-data.zip. ISO/IEC JTC 1/SC27, 2020.

[127] Mark Zhandry. "Secure Identity-Based Encryption in the Quantum Random Oracle Model". In: *CRYPTO*. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 758–775.

[128] Cong Zhang, Dongsheng Liu, Xingjie Liu, Xuecheng Zou, Guangda Niu, Bo Liu, and Quming Jiang. "Towards Efficient Hardware Implementation of NTT for Kyber on FPGAs". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401170.

[129] Jiang Zhang, Yu Yu, Shuqin Fan, Zhenfeng Zhang, and Kang Yang. "Tweaking the Asymmetry of Asymmetric-Key Cryptography on Lattices: KEMs and Signatures of Smaller Sizes". In: *Public Key Cryptography (2)*. Vol. 12111. Lecture Notes in Computer Science. Springer, 2020, pp. 37–65.

Índice de figuras

Índice de tablas

Índice de algoritmos

Acrónimos

AES Advanced Encryption Standard.

AKE Authenticated Key Exchange.

ARM Advanced RISC Machine.

AVX Advanced Vector Extensions.

BMI Bit Manipulation Instructions.

CBC Cipher Block Chaining.

CCM Counter with CBC-MAC.

CI/CD Continuous Integration and Continuous Delivery/Continuous Deployment.

CPU Central Processing Unit.

CRS Common Reference String.

CSV Comma-Separated Values.

DEM Data Encapsulation Mechanism.

DES Data Encryption Standard.

EUF-CMA Existential Unforgeability under Chosen Message Attack.

FPGA Field Programmable Gate Arrays.

GAKE Group Authenticated Key Exchange.

GCM Galois/Counter Mode.

GKE Group Key Exchange.

GOST Gosudarstvenny Standart.

IND-CCA INDistinguishability under Chosen-Ciphertext Attack.

IND-CPA INDistinguishability under Chosen-Plaintext Attack.

IoT Internet of Things.

IV Initialization Vector.

KEM Key Encapsulation Mechanism.

LWE Learning With Errors.

MAC Message Authentication Code.

MITM Man In The Middle.

MLWE Module Learning With Errors.

MLWR Module Learning With Rounding.

NIST National Institute of Standards and Technology.

OW-CPA One-Wayness under Chosen Ciphertext Attack.

OW-CPA One-Wayness under Chosen Plaintext Attack.

PKE Public Key Encryption.

PKI Public Key Infrastructure.

QROM Quantum Random Oracle Model.

RISC Reduced Instruction Set Computer.

RLWE Ring Learning With Errors.

ROM Random Oracle Model.

SHA Secure Hash Algorithm.

SKE Symmetric Key Encryption.

XOF eXtendable Output Function.