



Universidad de Valladolid

Escuela de Ingeniería Informática de Segovia.

Grado en Ingeniería Informática
de Servicios y Aplicaciones

Hacia un estándar de criptografía post-cuántica

Alumno: Khalid El Baroudi Rakdou

Tutor: José Ignacio Farrán Martín

Curso: 2022-2023

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

**Hacia un estándar de criptografía post-cuántica: estudio
teórico y práctico.**

Khalid El Baroudi Rakdou.

15 de septiembre de 2023

Índice general

Agradecimientos	8
Introducción	9
1. Descripción del proyecto	13
1.1 Introducción	13
1.1.1 Motivación	13
1.1.2 Contextualización del problema de los ordenadores cuánticos	14
1.1.3 Justificación de la elección del proyecto	17
1.2 Objetivos del proyecto	17
1.3 Estructura del proyecto	18
1.4 Estimación costes.....	18
2. Historia y conceptos de la criptografía.	23
2.1 Antecedentes históricos	23
2.2 Cifrado de clave pública.	29
2.3 Cifrado de clave privada.	31
2.4 Cuerpos finitos.	32
2.5 Compartición de secretos.	34
2.5.2 Algoritmo Shamir.	35

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

2.6 Firma digital.....	36
2.6.1 Firma digital RSA.....	37
2.6.2 Firma digital ElGamal.....	39
2.7 Cifrado de bloque.....	40
2.8 Vernam.....	42
2.9 Criptografía post-cuántica.....	42
2.9.1 Basado en teoría de códigos correctores de errores	43
2.9.2Basado en retículos	43
2.9.3 Basado en isogenias de curvas elípticas.....	44
2.10 Influencia de la criptografía cuántica en criptomonedas	45
3. Criptografía general	46
3.1 Seguridad en la computación post-cuántica.....	47
3.2 Computación cuántica y clásica.....	48
3.3 Algoritmos cuánticos.....	49
3.4 Puertas lógicas vs cuánticas.....	54
3.5 Estándares y protocolos en la criptografía post-cuántica.....	60
3.6 Máquina de Turing.....	64
3.7 Máquina de Turing cuántica.	66
3.8 Qubit.	66
3.9 Algoritmos vulnerables a ataques cuánticos.	66

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

4. Concurso NIST	68
4.1 Descripción del concurso.....	68
4.2 Descripción y análisis de los finalistas.	70
4.3 Estudio comparativo.	76
4.4 Futuro de la criptografía post-cuántica.	85
5. Marco práctico	87
5.1 CRYSTAL-KYBER.	87
5.2 NTRU.....	100
5.3 McEliece.	105
5.4 Pruebas.....	107
Conclusiones	124
Referencias.....	126

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Agradecimientos

Quisiera agradecer a mi tutor por aconsejarme este tema del Trabajo de Fin de Grado, ya que he aprendido otro apartado de la informática fascinante, a todos los profesores que me han impartido clase ya que me han hecho mejorar y aprender en el ámbito académico como en el humano y sobre todo a mi familia y amigos que me han brindado el apoyo necesario para seguir adelante

Introducción

El Instituto Nacional de Estándares y Tecnología más conocido por su acrónimo NIST, es la institución encargada de ayudar a las pequeñas, medianas y grandes empresas a analizar los riesgos de ciberseguridad que existen, además de una protección de datos y redes para poder reducir al máximo posible los riesgos existentes. Este instituto depende del Departamento de Comercio de EE. UU. (Comisión Federal de Comercio, s.f.)

Para profundizar más en el tema, hay que comprender que es la criptografía post cuántica. Consiste en una rama de la criptografía que trata de desarrollar algoritmos que sean capaces de resistir ante ataques de ordenadores cuánticos.

La criptografía tradicional de clave pública se basa en problemas matemáticos complejos, como la factorización del logaritmo discreto, RSA y ElGamal, sin embargo, para los ordenadores cuánticos sería tarea fácil ya que no se rigen mediante el sistema de bit binario (computación clásica) si no por el qubit (computación cuántica), el cual permite a diferencia del bit ordinario (0 o 1) una superposición de ambos al mismo tiempo. Esta cualidad permite a los ordenadores cuánticos ser exponencialmente mejores que los ordenadores tradicionales lo que se denomina paralelización cuántica. (Mena Viveros, 2013)

El concurso NIST sobre criptografía post-cuántica fue creado en 2016 para solucionar entre otros los inconvenientes anteriormente mencionados, podríamos decir que es un concurso en el que la colaboración entre los investigadores ha sido clave para un avance significativo en la seguridad ante los ataques de los ordenadores cuánticos.

La última ronda del concurso comenzó en 2022 y ha finalizado en marzo de 2023. Se evaluaron 9 algoritmos en la tercera ronda, estos fueron estudiados según su seguridad, usabilidad y rendimiento.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Los 9 algoritmos los podemos dividir en 3 categorías:

- **Basado en teoría de códigos correctores de errores:** Consisten en algoritmos que utilizan la teoría de códigos para elaborar algoritmos resistentes a los ordenadores cuánticos. En este grupo el finalista del torneo es el algoritmo Classic McEliece. Fue introducido en 1978 por Robert McEliece, sufrió un cambio en el aumento de parámetros de seguridad para pillar el aumento de la velocidad de los ordenadores además es el algoritmo más investigado del concurso NIST por lo tanto es el que tiene mayor nivel de seguridad. (PQSecure, 2020)

- **Basado en retículos:** Consisten en algoritmos que utilizan técnicas matemáticas basadas en retículos para resolver algoritmos criptográficos, se utilizan para asegurarnos de la seguridad de los sistemas criptográficos. Podemos destacar 3 finalistas que son CRYSTALS-KYBER, NTRU y SABER, cuyo ganador es el primer citado. El CRYSTALS-KYBER es un criptosistema de clave pública, su función es resolver el inconveniente M-LWE. Este algoritmo es IND-CCA2 y consta de 3 conjuntos diferentes para acomodarse al nivel de seguridad NIST 1,3,5. El NTRU es otro criptosistema de clave pública, también seguro ya que cumple el IND-CCA2. Trata sobre la dificultad de resolver problema de aprendizaje en anillos (Ring-LWE).

Se podría considerar que es una fusión del NTRUEncrypt y NTRUHRSS-KEM antes de la ronda 2.

(PQSecure, 2020)

- **Basado en isogenias de curvas elípticas:** Se fundamenta en utilizar las propiedades de las isogenias, las cuales son aplicaciones algebraicas entre curvas elípticas. La clave de estos tipos de algoritmos es encontrar una función que mapee una curva elíptica a otra. En esta sección no hay ningún finalista y solo una alternativa que es el algoritmo SIKE. Este tiene la clave pública más pequeña de todos los candidatos, pero es el más lento.

(PQSecure, 2020)

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Mas adelante se profundizará más en el concurso NIST.

Parte I

Documentación

Capítulo 1

Descripción del proyecto

En este proyecto se va a realizar un estudio teórico-práctico sobre el concurso de criptografía post-cuántica del NIST y la computación cuántica, sobre todo se va a hablar de los concursantes del torneo en esta disciplina.

Además, se va a sustituir el estudio práctico de un algoritmo de los seleccionados por una comparación de todos los algoritmos, comparando sus claves, textos cifrados, ventajas, desventajas, uso, tipos..., esta proposición fue consensuada previamente con el tutor del proyecto.

1.1 Introducción

1.1.1 Motivación

La criptografía es la ciencia que forma parte tanto de la informática como de las matemáticas, que tiene la función de encargarse de transformar información (texto, imágenes, videos etc.) para que terceros no puedan interceptar y leer la información que se quiere ocultar menos a el emisor autorizado mediante un canal seguro, esto ayudando a la privacidad, seguridad e integridad de los datos.

Estas características son esenciales para tratar información bancaria, financiera y sobre todo información personal que en malas manos puede ser muy perjudicial para el usuario.

Podríamos indicar diferentes tipos de cifrado como: clave privada y pública.

Ante la creciente amenaza de los ordenadores cuánticos se han visto obligados desarrolladores de todo el mundo a desarrollar cifrados cuánticos que ayuden a controlar los ataques cuánticos y las amenazas que puedan occasionar.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Actualmente el tema de cifrado cuántico es un tema que se desconoce en la mayoría de la población, están más familiarizados con el cifrado antiguo de textos mediante sustitución de letras por su frecuencia entre otros, además de la idea general que tiene el ciudadano sobre la criptografía y no sobre el conocimiento de lo que puede conllevar este tema.

Este tema propuesto me va a permitir abarcar más conocimiento científico-matemático del que me podía ofrecer otros, además de la posibilidad que se me ofrece de tratar en una materia emergente y que va evolucionando con el paso de los años y el aumento de la evolución tecnológica.

1.1.2 Contextualización del problema de los ordenadores cuánticos

Los ordenadores cuánticos han generado mucha inseguridad y amenaza a los sistemas computacionales clásicos ya que estos no están preparados para ordenadores cuánticos debido a que se realizan cálculos en qubit en vez de bit ordinarios, esto hace que realice cálculos exponencialmente más rápido que los ordinarios.

Por la razón anterior los cifrados hechos originalmente para ordenadores clásicos serán susceptibles a ser vulnerados por los ordenadores cuánticos.

Ejemplos como el RSA y ECC, son sistemas de cifrado actuales, creados ya que están formados por problemas matemáticos de una gran dificultad para ser descifrados por los ordenadores clásicos.

Los ordenadores cuánticos mediante técnicas como superposición pueden resolver estos problemas matemáticos de una forma mucho más rápida que los ordenadores clásicos.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Los ordenadores cuánticos en el ámbito de la inteligencia artificial facilitaran a los equipos aprender información de forma mucho más rápida. Se utiliza sobre todo para métodos predictivos, tema de finanzas, humanidad etc. Todo esto todavía no ha sido puesto en práctica.

La I.A y los ordenadores cuánticos es una unión creada por la evolución del mundo constantemente ya que hay que recoger una serie de datos que cada vez es mayor para técnicas como puede ser el machine learning.

Al hablar de I.A y computación cuántica debemos asociar el termino Quantum Machine Learning (QML).

En el sector de las finanzas y la bancan facilita la detección del fraude y como poder prevenirlo.

Un ejemplo de aplicación de la computación cuántica es la ciencia de los materiales la cuál a la hora de modelar las moléculas de pequeño tamaño necesita una gran capacidad informática. (Claramunt Carriles, 2022)

Existen otras aplicaciones que son de mayor uso como es el reconocimiento de imágenes o la predicción del consumo.

El 4 de mayo de 2022, la presidencia de EE. UU, publicó un escrito para manifestar el liderazgo de EE. UU en la computación cuántica (en el cuál analizaba alguno de los problemas y riesgos de la computación cuántica). (Barbieri, 2022)

Los ordenadores cuánticos utilizan propiedades de la física cuántica, eso hace que pueda obtener datos de forma más rápida y eficiente, esto reduce la complejidad de algunos problemas, como pueden ser los algoritmos criptográficos y la firma electrónica.

Los crímenes en la red podrían ser más recurrentes ya que estos ordenadores

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

cuánticos están hechos para vulnerar las protecciones actuales, como podría ser falsificar firmas. Por todo esto es necesario desarrollar algoritmos post-cuánticos que ayuden a la protección y a la seguridad del usuario.

El memorándum de EE. UU, manifiesta que hay que paliar los riesgos de la computación cuántica lo máximo posible para el año 2035, además de poder ofrecer la capacidad de cambiar de manera inmediata los algoritmos utilizados. (Barbieri, 2022)

Esto último puede ser interpretado de dos formas:

- 1- Para acoger los resistentes a la criptografía post-cuántica.
- 2- Reemplazar los elegidos si son susceptibles a un ataque.

Para los países evolucionar en este tipo de computación es esencial, ya que si otro país pudiera desarrollar este tipo de algoritmos antes de mejorar en la seguridad ante estos algoritmos podría considerarse una situación crítica, a la que ningún país quiere llegar. En cambio, si un país se desarrolla antes que otro, le daría una ventaja inmensa sobre otros respecto a ciberataques.

Debido a que hay países que no muestran su evolución en conocimientos como la computación cuántica hace que no se sepa a ciencia cierta si estos algoritmos puedan ser sólidos (como si ocurre con los actuales).

En Europa se estima en que no haya avances al nivel de EE. UU y China, ya que ya ha manifestado que su objetivo no es ser líder de este sector, aunque ya tiene iniciativas para el sector, todo esto nos hace indicar que cuando esta computación sea algo normal dentro de la sociedad traerá algún fallo de seguridad, solidez...

(Barbieri, 2022)

1.1.3 Justificación de la elección del proyecto

El avance de las tecnologías me hizo plantearme que el tema de mi TFG debía ser un campo de la tecnología que estuviese en desarrollo y en un futuro pudiera ser una revolución. Mi idea inicial era hacer un TFG de criptografía ya que tuve una asignatura en mi centro llamada Protocolos de la Información y Comunicaciones Seguras. Me fascinaba la posibilidad de descifrar algoritmos por diferentes metodologías como la posibilidad de cifrar textos para dificultar la manera de que alguien pudiera descifrarlo.

La película “The Imitation Game” dirigida por Morten Tyldum y estrenada en 2014, fue una película que me maravillo además de introducirme en el campo de la criptografía. Hasta entonces no tenía mucha información y conocimiento sobre ello.

El tema concreto de “Hacia un estándar de criptografía post-cuántica”, fue propuesto por mi tutor, me pareció interesante y acepte de inmediato ya que me iba a permitir complementar y mejorar sobre este campo de la informática.

1.2 Objetivos del proyecto

Se realizará un estudio teórico-práctico sobre el concurso NIST en la modalidad de criptografía post-cuántica. Para resumir, se mostrará los objetivos propuestos desde el inicio del TFG:

- Estudio general de la criptografía post-cuántica.
- Desarrollo del concurso NIST para un estándar de cifrado de clave pública.
- Estudio propuestas finalistas en dicho concurso.
- Aprender nuevos métodos de criptografía.
- Comprender como funciona la computación cuántica.

- Comprender los riesgos que esta computación puede y genera en la sociedad.
- Analizar campos de la informática ligado a la computación cuántica.

1.3 Estructura del proyecto

Capítulo 1 Introducción

Introducción concisa sobre la motivación del proyecto, la contextualización del problema de los ordenadores cuánticos y por último la justificación de la elección del proyecto.

Capítulo 2 Historia y conceptos de la criptografía

Breve descripción de los antecedentes históricos, tratar la seguridad de estos cifrados post-cuánticos y los fundamentos de la criptografía post-cuántica.

Capítulo 3 Criptografía general

Consistirá en el análisis de diferentes tipos de algoritmos, protocolos que ha habido alrededor de la historia, como su funcionamiento.

Capítulo 4 Concurso NIST

Una breve descripción del concurso y de los finalistas por tipo de cifrado. Se procederá a tratar sobre el futuro de la criptografía post- cuántica.

Capítulo 5 Marco práctico

Analizar de forma práctica a los finalistas del certamen, comparación y la evaluación entre ellos.

1.4 Estimación costes

La planificación del proyecto se estimó entre 300 a 360 horas como estipula la guía docente del campus, por lo tanto, mi intención era mantener ese rango, estimo que se ha dedicado al proyecto 350 horas partiendo desde que ya se había seleccionado que

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

TFG tratar. Estas horas son repartidas en las siguientes tareas o actividades.

Lista de actividades:

- T1: Búsqueda de información para dictar los temas necesarios para el TFG, búsqueda de ideas para mejorarlo. Se estima que en esta actividad se dedican 35 horas.
 - T2: Búsqueda de información sobre la criptografía clásica y moderna para poder tener una base de la criptografía. Se estima que en esta actividad se dedican 35 horas.
 - T3: Estudio de las normas APA 7º Edición para documentar el proyecto. Se estima que en esta actividad se dedican 7 horas.
 - T4: Estudio del concurso NIST, especialmente de los algoritmos kem, cuya documentación se puede encontrar en la página oficial como también la de los algoritmos en sus páginas oficiales. Se estima que en esta actividad se dedican 105 horas.
 - T5: Instalación de Linux, de librerías como Openssl y libcekac para el funcionamiento de los algoritmos. En esta tarea se perdió tiempo ya que en Windows no era tan asequible compilar y ejecutar los programas de los algoritmos ganadores, ya que había falta de compatibilidad como diversos fallos de reconocimientos de librerías necesarias. Se estima que en esta actividad se dedican 105 horas.
 - T6: Obtención de los datos de la prueba, comprimidos para luego poder ser visualizados o comparados. Se estima que en esta actividad se dedican 24 horas.
 - T7: Tutorías con el tutor. Se estima que en esta actividad se dedican 10 horas.
 - T8: Redacción del proyecto. Se estima que 75 horas.
- Coste de programador y analista de documentación: Se ha propuesto que hay

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

dos tipos de trabajadores en este proyecto el programador y analista de documentación. El programador es el que se encarga de ejecutar las tareas en Linux, como la configuración de las librerías, programas y demás, sin embargo, el analista de documentación es el que se encarga de redactar la información obtenida por el programador como a su vez del documento de entrega, citas, cumplimiento de normas APA etc.

Con la suma de las horas obtenidas entre las tareas citadas en este punto se reparten de la siguiente manera:

$$T1+T2+T3+T4+T5+T6+T7+T8= 35+35+7+105+105+24+10+75 = 396 \text{ horas.}$$

Las tareas son dedicadas exclusivamente por un empleado. El programador se encarga de realizar T5 y T6, por lo tanto, dedica 129 horas. El analista se encarga de realizar T1,T2,T3,T4,T7,T8, por lo tanto, dedica 267 horas.

Repartición de tareas:

Tarea	T1	T2	T3	T4	T5	T6	T7	T8
Programador					X	X		
Analista	X	X	X	X			X	X

Cargo	Horas estimadas	Coste euros-hora	Coste total euros
Programador	129	15	1935
Analista	267	12	3204
Total	396		5139

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Lo siguiente a tratar es el coste del software utilizado en el proyecto:

Componente	Coste sin I.V.A(euros)	Coste con I.V.A(euros)	Tiempo usado (horas)	Tiempo de licencia(año)
Microsoft Office	54,51	69	75	1
Oracle VM	Gratis		129	
Windows 10 Programas	94,8	120	281	Indefinida
Spyder IDE	Gratis		50	75
Codeblocks	Gratis		55	Indefinida
Github	Gratis		2	Indefinida

Lo siguiente a tratar son los costes técnicos: internet, luz, etc, no supera los 30 euros.

Si la planificación inicial se suponía que iba a ser 300 horas, esto indicaría que las horas del programador estarían en torno a las 85 y las del analista en 215.

Esto se indicaría aquí:

Cargo	Horas estimadas	Coste euros-hora	Coste total euros
Programador	85	15	1275
Analista	215	12	2580
Total	300		3855

Esto indicaría que el costo de personal planificado en un principio alcanza los 3855 y el costo de personal real alcanza 5139, una variación de 1284 euros de más

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

gastados debido al aumento de horas. El coste total del proyecto asciende a la suma del coste de personal, software y técnicos: $5139 + (69 + 120) + 30 = 5358$ euros.

Capítulo 2

Historia y conceptos de la criptografía.

2.1 Antecedentes históricos

El primer indicio de métodos que permitiesen el cifrado de mensajes procede de Grecia, alrededor del siglo VI antes de Cristo, cuyo nombre se conocía como escítalo.

El escítalo consistía en una vara en la cual era rodeada por un pergamino. El mensaje era colocado en el pergamino de forma horizontal pero una vez que se desenrollaba se mostraba de forma vertical, esto hacía que el mensaje no fuese fácil de visualizar. El siguiente indicio importante fue en Roma por los ejércitos romanos el llamado Cifrado Cesar. El Cifrado Cesar, fue como se ha comentado previamente una herramienta para mantener seguros los mensajes en la guerra. El cuál consistía en un cifrado de sustitución el cual cada letra era desplazada un número fijo de posiciones. Por ejemplo, si la letra era la “A” y el desplazamiento era 4, la letra que se mostraba era la “E”, para descifrar el mensaje habría que hacer lo mismo, pero al revés.
(Xifré Solana, 2009)

En el imperio árabe podemos destacar a Al-Kindi (801-873) fue conocido como el filósofo, intelectual, matemático más famoso musulmán. Su libro más famoso redescubierto en 1987 llamado “Sobre el desciframiento de mensajes criptográficos”. En el siglo XIX Thomas Jefferson inventó un mecanismo de cifrado que consistía en un aparato formado por diez cilindros los cuales están colocados en un eje de forma independiente, en esta posición se colocaba el alfabeto y al girar estos cilindros el mensaje a cifrar ya era cifrado.

En la Primera Guerra Mundial (1914-1918) los mensajes eran enviados por los soldados

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

a las zonas de guerra, esto hacía que fuese difícil trasladar la información a zonas conflictivos con lo cual se empezó a usar tecnología como teléfono y la radio. Aun así, esto hacía que fuese asequible interceptar esta información. El siguiente paso en los años siguientes en la criptografía fue el cambio por las denominadas maquinas criptográficas como puede ser la famosa “Enigma” usada por la Alianza contra el ejército nazi, en Estados Unidos poseían SIGABA, en Japón poseían Purple. En 1948-1949 dos artículos redactados por Claude Shannon, “Teoría Matemática de la Comunicación” y “La Teoría de la Comunicación de los Sistemas Secretos”. A partir de la Segunda Guerra Mundial inició la nueva era de la criptografía electrónica, se puede considerar que los avances en criptografía más destacados de estos años eran la Guerra Fría. En 1975 IBM creó el DES (Data Encryption Standard) conocido en el castellano como el Estándar de Encriptación de Datos, que se sigue usando en la actualidad. En 1976 W. Diffie y M. Hellman sucedió un suceso muy importante el cuál fue la creación de la clave pública, esto consistía que la clave de cifrado pertenece a un directorio público, sin embargo, la clave de descifrado es distinta. En 1978 R. L. Rivest, A. Shamir y L. Adleman los cuales pertenecían al MIT crearon el criptosistema de clave pública más usado hasta la fecha el RSA. . (Xifré Solana, 2009)

Se procederá a hacer un resumen de métodos específicos de cifrado a lo largo de la historia:

Criptografía clásica:

- Civilización Egipcia: De la civilización egipcia podemos destacar la escritura jeroglífica la cual está compuesta por caracteres ideográficos combinándolos con caracteres fonéticos los cuales representan uno o varios sonidos. Podemos destacar cuatro características de estos:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

1- Dimensión: La proporción de los signos no se corresponde con la realidad.

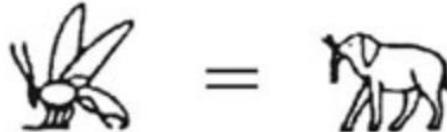


Figura 2.1 Dimensión según la civilización egipcia.

(Domenech Belda, 2004)

2- Direccionalidad: Puede escribirse de horizontal a vertical o al revés, además de izquierda a derecha y al revés. Eso hace que haya 4 orientaciones posibles.



Figura 2.2 Direccionalidad según la civilización egipcia.

(Domenech Belda, 2004)

3- Disposición: Los signos no se encuentran alineados uno con otro, si no que se agrupan en un cuadrado imaginario en el que se ahorra un espacio grande, esto hace que no haya espacios en blanco.



Figura 2.3 Disposición según la civilización egipcia.

(Domenech Belda, 2004)

4- Pronunciación: Este tipo de escritura es consonántica, es decir no se posee vocales.

Los egiptólogos decidieron añadirle una vocal e al resto de consonantes para poder leerlas.

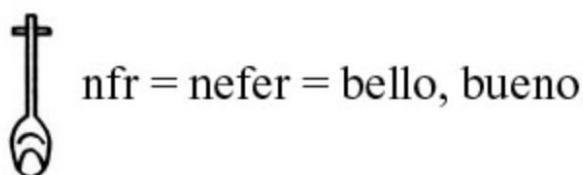
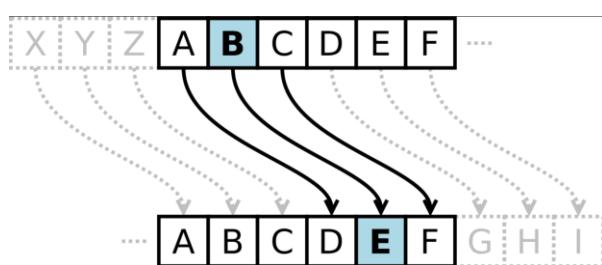


Figura 2.4 Pronunciación según la civilización egipcia.

(Domenech Belda, 2004)

- Civilización Romana: De la civilización romana podemos destacar cuatro criptosistemas el cifrado de césar, cifrado de sustitución, cifrado del grano de trigo y las escítalas.

En primer lugar, el cifrado de césar consiste básicamente en desplazar las letras x posiciones.



HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Figura 2.5 Cifrado César.

(Museu Informática, 2021)

El cifrado de sustitución se compone de dos tipos: sustitución monoalfabética y polialfabética. Este cifrado consiste en cambiar una letra por otra en todo el texto.

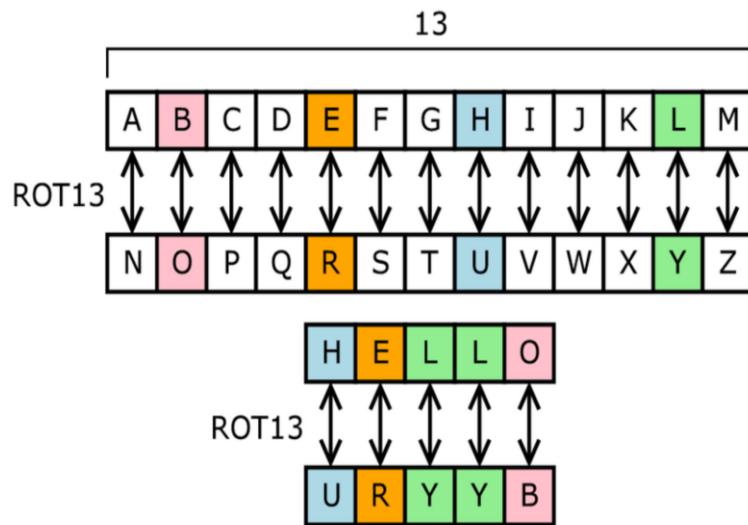


Figura 2.6 Cifrado de sustitución.

(Tecnología + Informática, 2021)

Las escítalas eran unas varillas cilíndricas en la que se enrollaba un pergamo en el cual el mensaje se escribía en espiral a lo largo de la varilla y al desenrollarla se mostraba.



Figura 2.7 Ejemplo de escítala.

(Gutiérrez, 2012)

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Por último, el cifrado de grano de arena consistía en escribir el mensaje en granos de trigo colocados de forma de cuadrado, el mensaje era revelado una vez los granos se dispersaban.

- Civilización Mesopotámica: En el año 3300 A.C. apareció la escritura cuneiforme . Los signos cuneiformes representaban o letras, palabras o sílabas, esto se leía de izquierda a derecha. A veces como los egipcios cambiaban signos cuneiformes por , sin embargo, tenían intención de ocultar el significado de la escritura a diferencia de los egipcios. (Xifré Solana, 2009)

- Civilización Griega: De la civilización griega podemos destacar dos cifrados, el cifrado de Esparta y el cifrado de Atbash.

El cifrado de Esparta era un cifrado basado en la transposición. Básicamente consistía en escribir el mensaje en columnas y leerlo en filas.

El cifrado de Atbash consistía en un cifrado por sustitución en el que sustituye la letra por su opuesta en el alfabeto. Por ejemplo. La A por la Z, la B por la Y.

Cifrado Atbash																									
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Figura 2.8 Cifrado Atbash.

(Castellanos, 2017)

En 1500 A.C se conserva una tablilla de arcilla en la que se escribió una fórmula en barniz, se considera un tesoro de esa edad.

2.2 Cifrado de clave pública.

Este tipo de cifrado usa un par de claves relacionadas entre sí. La información cifrada con la primera clave tiene que desencriptarse/descifrarse con la segunda clave, y la información cifrada con la segunda clave debe desencriptarse/descifrarse con la primera clave .Los participantes en el sistema de clave pública tienen dos claves. Una de ellas es clave privada y es secreta, en cambio la otra es enviada a todo el mundo que quiera, esta va a ser la clave pública. Todos los participantes pueden encriptar/cifrar usando su clave pública, pero solo el usuario puede visualizarlo. Cuando el usuario lo recibe, puede desencriptarlo/descifrarlo con la clave privada. Un mensaje que se quiere cifrar se puede utilizando su clave pública y descifrar con su clave privada. Una vez que se cumplen estas condiciones el mensaje se puede enviar de forma segura mediante un canal no seguro. (IBM, 2021)

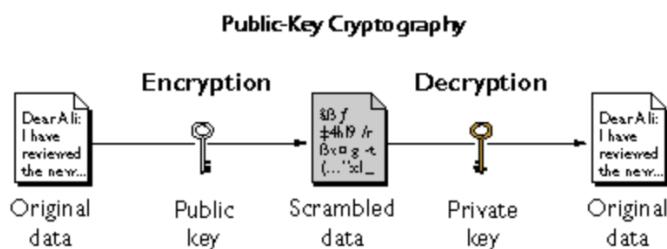


Figura 2.9 Esquema cifrado de clave pública.

(IBM, 2021)

Características de la clave pública:

- Este tipo de cifrado hace que cada participante tenga dos claves.
- Cualquier participante que sepa la clave pública puede enviar el mensaje, pero solo el usuario podría visualizarlo.
- La forma de mantenerse seguro es mantener la clave privada en secreto.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- También es llamado cifrado asimétrico, esto es porque no se puede usar la misma clave para cifrar y descifrar.
- En el cifrado de clave pública, cualquier usuario puede crear dos claves y compartir la clave pública. Surge un dilema a la hora de poder comprobar la identidad del creador de la clave pública, ya que este usuario crear las claves públicas con un nombre falso. El creador de la clave pública no puede visualizar los mensajes cifrados por esa clave ya que no tiene la clave privada. El usuario anteriormente citado si tiene la posibilidad de interceptar los mensajes anteriormente citados, le permitiría descifrar y visualizar mensajes que iban dirigidos a otro usuario. (Real Casa de la Moneda, s.f)

Infraestructura de claves públicas: La infraestructura de claves públicas también llamada PKI es la infraestructura que gracias a las tecnologías de clave pública facilita que las aplicaciones se comuniquen de forma segura, esto da seguridad. En la realidad una pequeña cantidad de información es cifrada de esta manera. PKI usa la clave privada para que la aplicación pueda firmar documentos. Para el receptor pueda autenticar al emisor, se debe obtener la clave pública del emisor, esta clave se obtiene

de un certificado digital, que es vigilado por una entidad emisora de certificados de terceros que sea fiable. (IBM, 2021)

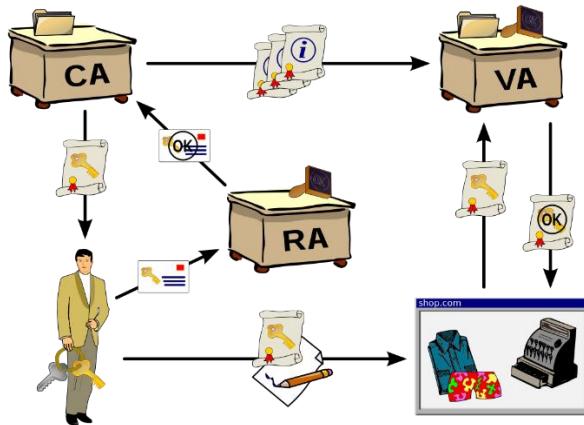


Figura 2.10 Infraestructura de clave pública. (Wikipedia, 2022)

2.3 Cifrado de clave privada.

Este tipo de cifrado consta solo de una clave, la cual compartirán el emisor y el receptor. El funcionamiento es simple, para cifrar el mensaje el emisor utiliza la clave y para descifrar el mensaje del receptor utiliza también la clave.

Esta clave debe no compartirse para poder garantizar seguridad en el envío.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA



Figura 2.11 Cifrado de clave privada. (Moreland, 2022)

Características de la clave privada:

- Las claves deben ser enviadas a todos los participantes ya que la necesitan para cifrar y descifrar. Esto hace que sean más vulnerables a ser robadas.
- Este cifrado necesita que haya una clave por emisor y receptor si desean enviar información de forma privada. El número de claves aumenta si se aumenta el número de participantes.
- Se necesita un acuerdo o un pacto para mantener comunicaciones entre ellos. No pueden enviar información a alguien al azar.

(IBM, 2021)

2.4 Cuerpos finitos.

Un cuerpo K es finito una vez que posee un numero finito de elementos. Los cuerpos finitos también son conocidos como cuerpos de Galois, esto sucedió en honor a Évariste Galois, matemático del siglo XIX, su contribución más destacada es la teoría que lleva su apellido, además del estudio de los grupos y cuerpos algebraicos. Los cuerpos y grupos algebraicos son fundamentales para entender el funcionamiento de

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

los algoritmos clásicos, curvas elípticas... (Raposo Briceño, 2019)

La teoría de Galois: Uno de los problemas que surgió en la antigüedad en el campo del álgebra, es buscar soluciones en una ecuación polinomial. Los polinomios de grado 2 ya tenían soluciones en la antigüedad, en el siglo XVI hubo soluciones para polinomios de grado 3 y 4. Sin embargo los polinomios de grado 5 durante más de 300 años no se podían solucionar. Alrededor del siglo XIX, Galois fue el indicado de mostrar que polinomios podían ser resueltos y cuáles no. Básicamente esta teoría muestra que hay una gran independencia entre la teoría de grupos y la de cuerpos. (Judson, 2017)

Demostraciones:

- Un cuerpo finito tiene una característica llamada p , la cual es un número primo.

Si tenemos un cuerpo K el cual es finito que posee n elementos, p será un divisor de n , lo cual hace que $nk = 0$ para todo $k \in K$.

- Si K es un cuerpo finito que posee una característica llamada p que es un número primo. Esto conlleva a que el orden de K es p^n para cierto natural n distinto de 0. (Raposo Briceño, 2019)

Construcción de un cuerpo finito:

- Sea K el cuerpo de escisión (consiste en el campo de extensiones de menor tamaño que contiene las raíces del polinomio) del polinomio $f(x) = x^{p^n} - x \in \mathbb{Z}_p$. (Raposo Briceño, 2019)

Ejemplo de teorema:

- El polinomio $x^{q^n} - x \in F_q$ es el resultado del producto de todos los polinomios

irreducibles cuyo grado divide a n. (Raposo Briceño, 2019)

2.5 Compartición de secretos.

En la criptografía clásica, en la criptografía de clave privada y pública solo había dos participantes: el emisor y el receptor. Sin embargo, en la criptografía moderna el avance de las redes, y la obligación de mantener seguridad y privacidad de la información. Esto hace que la información mediante el uso de protocolos hace que haya más participantes. La información secreta no es creada para solo personas físicas, sino también para empresas y entidades. Esto hace que no esté muy claro si la información ha de caer solo en una persona si no en varias. Un esquema para compartir secretos es un protocolo criptográfico, el secreto se divide en diversas personas para mantenerse seguro. La división del secreto entre los participantes debe cumplir:

- Solo hay ciertos participantes que pueden volver a construir el secreto original.
- Los participantes no autorizados no pueden obtener información del secreto original.

El secreto final no consiste en unir los fragmentos de cada participante, sino en ejecutar un algoritmo cuya entrada van a ser los fragmentos anteriores. En 1979 Shamir y Blackley fue el primero en crear los primeros esquemas para la compartición de secretos. Un algoritmo debe ser robusto ante personas externas. El ejemplo más significativo es en el que un participante del esquema tendría la capacidad de engañar a otros participantes, dándoles información falsa, como consecuencia un secreto erróneo. El inconveniente del algoritmo Shamir consiste en que el participante que quiere hacer trampa con un mensaje correcto y un secreto incorrecto podría recuperar el secreto original. Para las votaciones electrónicas también se usa la compartición de

secretos, cuando se procede a abrir la urna tiene un código que se reconstruye con la clave secreta que ha sido compartida a los miembros de la sala. La compartición de secretos ofrece una seguridad incondicional, pero también hay un tipo que es menos estricto llamado seguridad computacional segura, de la cual se debe cumplir lo siguiente:

- Algunos participantes autorizados pueden volver a construir el mensaje.
- Los participantes no autorizados no tienen la capacidad de obtener información sobre el secreto original.

(Villar Santos y otros, s.f)

2.5.2 Algoritmo Shamir.

Para tratar sobre el algoritmo de Shamir se va a considerar un conjunto de participantes $P = \{P_1, P_2, \dots, P_n\}$ y un entero al que vamos a llamar t el cual está dentro de este dominio $1 \leq t \leq n$. El conjunto de secretos va a consistir en un cuerpo finito, el cual lo vamos a llamar Z_p , la p va a consistir en un número primo. P debe ser mayor que el número de participantes.

En la primera fase del algoritmo se le proporcionara a cada participante un elemento x_i que pertenece al cuerpo Z_p . Los valores de x_i son no nulos y diferentes cada uno de ellos. La distribución del secreto llamado $s \in Z_p$ se lleva a cabo mediante la toma de un polinomio al azar menor o igual que $t-1$, que $r(x) \Rightarrow r(0) = s$. El participante recoge como fragmento $s_i = r(x_i)$. Se necesita mantener en secreto tanto al valor del secreto como al coeficiente del polinomio, de esto se encarga el distribuidor D .

El algoritmo de Shamir es incondicional, ya que, aunque el individuo que quisiera romperlo tuviese una capacidad computacional ilimitada no podría obtener el secreto aun conociendo $t-1$ fragmentos.

Posee propiedades homomórficas, en algunos esquemas de compartición de secretos hay posibilidad de realizar operaciones simples sobre secretos, sin tener que volver a construirlos otra vez.

Un esquema para compartir secretos es homomórfico sobre Z_p , esto sucede si cada participante que hay P_i tiene la posibilidad de combinar los fragmentos $S_i^{(a)}$ y $S_i^{(b)}$ los cuales han sido recibidos por los secretos $s^{(a)}$ y $s^{(b)}$, si se quiere obtener un nuevo fragmento del secreto hay que sumar $s^{(a)} + s^{(b)}$. (Villar , 2017)

2.6 Firma digital.

Es el concepto que engloba a los datos asociados a un mensaje, el cual permite mantener segura la identidad del usuario que firma. La firma digital no implica que el mensaje sea cifrado, es decir, el mensaje que ha sido firma es descifrable si está cifrado o no. El usuario que se encarga de firmar mediante el uso de una función generara la huella digital. Esta huella digital se cifrará con la clave privada propia y esto generara la firma digital, esta va a ir adherida al mensaje original.

El receptor del mensaje a su vez puede comprobar si el mensaje no ha sido modificado utilizando la misma función que antes para generar la huella digital que se encuentra adherida al mensaje. Se puede verificar su autenticidad, descifrando la firma con la clave pública del que firma, esto genera la huella digital.

Los algoritmos de clave pública generan algunos inconvenientes ya que son lentos, y aumentan con el tamaño del mensaje que se quiere cifrar. Para esto la firma digital utiliza funciones hash.

Ejemplo de firma digital:

- Ana y Carlos tienen cada uno su clave.
- Ana formula un mensaje para Carlos. Carlos debe comprobar si es Ana la que está enviando el mensaje, mediante 3 pasos: Utiliza una función hash para reducir el mensaje, cifra el resultado de la función hash el cual hace obtener la firma digital, y por último se envía a Carlos el mensaje con la firma adherida.
- Carlos ya ha recibido el mensaje con la huella digital. Se comprueba la validez del mensaje y reconocer al creador del mensaje, esto se hace mediante 3 pasos: descifrar el resumen del mensaje con la clave pública del emisor es decir Ana, se aplica al mensaje la función hash esto hace obtener el resumen del mensaje, se compara el resumen recibido con el que ha salido del resultado de la función hash. Si son iguales, el mensaje es seguro que lo ha enviado Ana.

La firma digital permite las 3 siguientes características:

- Autentificación.
- Imposibilidad de negar que se ha enviado el mensaje por el usuario.
- Integridad.

(Real Casa de la Moneda, s.f)

2.6.1 Firma digital RSA.

En 1977, fue mostrado al mundo el algoritmo RSA, de la mano de Rivest, Adleman y Shamir. Es el ejemplo de clave pública más usado del mundo. El éxito de este criptosistema radica en la dificultad a nivel computacional de resolver la factorización de números enteros. La

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

factorización de números enteros se basa en un numero entero llamado n, y se intenta localizar la descomposición como producto de factores primos. Es un problema si el numero n no tiene factores pequeños.

Para generar la clave pública se tiene que considerar que $n = p \cdot q$, p y q serían números primos de alto valor.

El algoritmo RSA consta de 3 partes:

- Obtención de claves:

Todos los usuarios S tienen que elegir sus claves públicas y privadas.

S elige dos números de gran valor que van a ser p y q. Se procede a calcular su producto, es decir, $n = p \cdot q$, la colección de mensajes que usa el usuario S es el grupo multiplicativo. La cantidad de elementos del grupo es $\phi(n)$, ϕ es el indicador de n.

$\phi(n)$ se calcula mediante la siguiente fórmula: $\phi(n) = \phi(p \cdot q) = (p - 1)(q - 1)$.

Si el usuario S elige un número entero llamado e, que se encuentra en el siguiente intervalo $2 < e < \phi(n)$, esto hace que $\text{mcd}(e, \phi(n)) = 1$.

El usuario S va a calcular d, es decir, el inverso de e módulo $\phi(n)$, esto comprueba que $1 < d < \phi(n)$, además de $e \cdot d \equiv 1 \pmod{\phi(n)}$.

Con estos pasos ya se conocen la clave pública, n, e y la clave privada que va a consistir en el valor de d.

- Cifrado del mensaje: Suponemos que tenemos dos usuarios Ana y Carlos, los cuales quieren comunicarse con un mensaje que vamos a llamar m, está escrito en el alfabeto A y mediante una función llamada mensaje numérico

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

pueden convertirse en un entero. Carlos es el que va a enviar el mensaje a Ana. Se deben seguir los siguientes pasos:

Carlos pregunta la clave pública de Ana, es decir (n_A, e_A) es el par que es parte de la clave pública de Ana.

Carlos va a asignar el mensaje como un elemento Z_n , el rango al que pertenecerá el mensaje es el siguiente $\{0, 1, 2, \dots, n_A - 1\}$, también tiene que ser primo de n_A . A veces hay que dividir en diferentes y pequeños trozos el mensaje m para que sea menor que n_A si no lo es.

Carlos calcular el valor del criptograma con la siguiente función $c = m^{e_A} \pmod{n_A}$, el cuál va a ser enviado a Alicia.

- Descifrado del mensaje: Alicia recibe el criptograma de Carlos, si se desea recuperar el mensaje original se deben seguir los siguientes pasos:

Ana utiliza su clave privada d_A y utiliza la siguiente función en todos los elementos del criptograma, para poder recuperar el mensaje.

$$C^{d_A} \pmod{n_A} = (m^{e_A})^{d_A} \pmod{n_A} = m^{e_A d_A} \pmod{n_A} \equiv m \pmod{n_A}$$

(Giematic , s.f)

2.6.2 Firma digital ElGamal.

La firma digital ElGamal fue creada por Taher ElGamal en el año 1984.

Se va a usar el mismo ejemplo de Alicia y Carlos.

- Alicia debe crear un número aleatorio llamado a que cumpla:

$$\text{mod}(l, p-1) = 1.$$

- Alicia también debe calcular el elemento que vamos a llamar e ,

$$e \equiv a^l \pmod{p}.$$

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Alicia debe resolver la siguiente congruencia: $m \equiv a \cdot e + l \cdot s \pmod{p-1}$.
La firma digital está formada por el par (e, s) .
- Carlos debe comprobar el mensaje m , para ello debe seguir los dos siguientes pasos: Carlos calcula $e^s \equiv (a^l)^s \pmod{p}$, además de:
$$(a^a)^e \pmod{p}.$$
- A su vez, Carlos debe calcular $(a^a)^e \cdot (a^l)^s \pmod{p}$, luego hay que comprobar el resultado anterior si es igual a $a^m \pmod{p}$.

(Xifré Solana, 2009)

2.7 Cifrado de bloque.

Es un cifrado de tipo de clave simétrica, cuya función consiste en dividir el mensaje en bloques (unidades de tamaño fijas), para posteriormente aplicarles un cifrado en específico a cada una. Este tipo de cifrado consume mucho a nivel computacional debido a que hay que guardar archivos en memoria y por usar operaciones complicadas. Actualmente un cifrado de bloque de 64 bits con operaciones convencionales no puede ser seguro y es vulnerable a ataques.

Hay 4 funciones básicas en este tipo de cifrado:

- Difusión: Es la difusión del impacto de un bit con formato de texto simple a numerosos bits de texto cifrado. Todo esto se hace para ocultar la estadística de texto simple. Se usan las Cajas P-Box.
- Confusión: Es un cambio, el cual tiene como función cambiar la relación entre las estadísticas del texto plano a las del texto cifrado, todo esto tapando la relación entre clave y el mensaje que se ha cifrado. Se usan las Cajas S-Box.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Expansión de claves: Se utiliza la puerta XOR a la clave y la información del mensaje, para obtener una nueva. Si el tamaño de la clave no es igual al tamaño de bloque hay que expandir, no replicarla para que los cálculos sean más simples y el algoritmo sea productivo.
- Rondas: Los dos primeros conceptos que hemos tocado: la difusión y confusión, son un éxito cuando se usa cifrados iterados, en estos cifrados se pueden unir las cajas-S y las cajas-P con otros elementos.

(Numerentur Org, s.f)

Características:

- Mayor tamaño de bloque conlleva a que sea menos rápido, pero más seguro.
- Si hay más rondas hay más seguridad.
- Mayor tamaño de bloque conlleva a que sea menos rápido, pero más seguro.
- Mayor nivel de complejidad de las subclaves, conllevan a un alto nivel de complejidad del criptoanálisis.
- Mayor nivel de complejidad de las rondas, conllevan a alto nivel de complejidad del criptoanálisis.

(Numerentur Org, s.f)

Tipos de cifrado:

- Feistel: Se fracciona un bloque de texto en dos o más partes. Posteriormente se usan operaciones en estas. Ejemplos: DES,3DES, RC5...
- No Feistel: Se transforma los bloques de texto mediante números rondas

diferentes y subclaves. Como se comentó anteriormente se realizan operaciones S-Box y P-Box. Ejemplos: AES, SHARK, SQUARE...

(Numerentur Org, s.f)

2.8 Vernam.

También denominado cifrado de flujo y fue creado por Gilbert S. Vernam en el año 1917. El algoritmo permite relacionar un texto con una clave de un mismo tamaño lo cual hace que se cree un criptosistema eficiente, el cuál si se utilizase una clave al azar haría que fuese más seguro. El cifrado consiste en transformar cada carácter del texto en una cadena binaria. Esta cadena es transformada por una puerta XOR bit a bit con una clave aleatoria generada por un algoritmo, es decir parece aleatoria pero no lo es. Todo esto se puede resumir en asignar n bits los cuales son sumados por la puerta XOR mod 2 con una clave.

La función que representa todo esto es la siguiente:

$C_i = M_i \oplus K_i$ para $i = 1, 2, \dots, n$. Esto es para el cifrado.

$C_i \oplus K_i = M_i \oplus K_i \oplus K_i$. Esto es para el descifrado.

M_i son los n bits de cada carácter del texto, K_i la clave y C_i el mensaje cifrado.

(Urrego Urrego, 2019)

2.9 Criptografía post-cuántica.

La criptografía post-cuántica es aquella que engloba algoritmos, criptosistemas y protocolos criptográficos que de momento son invulnerables por la criptografía cuántica. Al hablar de criptografía post-cuántica hay que hablar sobre la factorización de enteros y el problema del logaritmo discreto.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Factorización de enteros: RSA, y KMOV que es usado en cuerpo elípticos.
- Logaritmo discreto: Podemos destacar ElGamal y firma digital (DSA) de los que se hablaran más adelante, y el intercambio de claves del protocolo Diffie-Hellman.

(Farrán, 2016)

2.9.1 Basado en teoría de códigos correctores de errores.

En la actualidad estos algoritmos no han reflejado vulnerabilidades a ataques cuánticos, esto hace que sea una de las áreas más prometedoras en la criptografía cuántica.

La investigación en este tipo de área viene dada por las convocatorias de estandarización post-cuántica del NIST.

El objetivo de este concurso es abrir al mundo estas áreas, dar charlas informativas, invitaciones de individuos conocedores de estos algoritmos.

(Farrán, 2016)

2.9.2 Basado en retículos.

Para explicar porque se debe tratar la criptografía basada en retículos

Debemos explicar las características y las razones de su estudio:

- Este tipo de criptografía es bastante fácil de comprender y de implementar.
- Son unos candidatos ideales para la criptografía post-cuántica ya que

no hay algoritmos cuánticos que resuelvan retículas que funcionen de mejor manera que los cásicos.

- Han provocado grandes avances en la criptografía homomórfica (capacidad que permite trabajar con información cifrada sin tener que descifrarlos)

Estos tienen una complejidad computacional menor para cifrar y descifrar que los criptosistemas famosos.

(Farrán, 2016)

2.9.3 Basado en isogenias de curvas elípticas.

Esta área de la criptografía es relativamente joven ya que surgió en la década de los 2000. Deriva de la criptografía de curvas elípticas la cual pertenece al área de clave pública, la criptografía de curvas elípticas surgió en 1980 cuando Miller y Koblitz convencieron de usar curvas elípticas en el protocolo de intercambio de claves de Diffie-Hellman.

En la década de 1990 gracias al algoritmo de Schoof facilitase localizar curvas elípticas de orden primo grande permitió un gran avance.

En la década del 2000 hubo dos innovaciones importantes en la criptografía en curvas elípticas:

-Aparición de la criptografía basada en parejas como es el intercambio de claves de Diffie-Hellman. Permite el intercambio entre ambas partes sin compartir la clave secreta.

- La criptografía basada en isogenias fue iniciada por Couveignes, Teske, Rostovtsev y Stolbunov. Mientras la criptografía basada en parejas tuvo mayor apogeo la criptografía basada en isogenias. Hasta la segunda parte de la década

del 2010, en el que este tipo de criptografía tomó importancia debido del miedo de la inminente llegada del ordenador cuántico de uso general.

(Farrán, 2016)

2.10 Influencia de la criptografía cuántica en criptomonedas.

La conexión entre la criptografía y las criptomonedas es un tema que llama mucho la atención actualmente debido a que al ser las criptomonedas una moneda para realizar pagos interesa que esos pagos se realicen de forma segura.

La inclusión de Taproot en Bitcoin hizo que la red de Bitcoin fuese más privada y segura. Una de las funcionalidades que permite esto es que no se pueda diferenciar entre transacciones de firma única y firma múltiple, esto indica que no hay posibilidad de verificar que tipo de transacciones involucra la apertura de canales de Lightning Network con las transacciones en la capa base. Con la evolución de los nuevos ordenadores cuánticos indica que se va a tener que modificar las técnicas criptográficas de las Blockchain. El Dr. Joel Alwen, cuyo puesto es criptógrafo jefe de Wickr, una de las mejores aplicaciones actuales de mensajería que utilizan cifrados para los mensajes indicó las siguientes premisas, las cuales tiene una visión general del tema:

- Los ordenadores cuánticos actuales no están todavía preparados para romper los estándares de cifrado actuales. El avance actual no es muy significativo.
- Se necesita diálogo entre expertos en computación cuántica y cifrado para llegar a un consenso común y evolucionar en el tema. Retracta que no hay suficiente comunicación entre estos expertos, por lo tanto, el avance de la computación cuántica es más lento que el del cifrado.

Hay países como China que no han revelado prácticamente ningún avance en el

sector eso hace que se crea que los avances son ocultados para no mostrar los verdaderos conocimientos en este sector que se tiene.

Todos estos factores indican que no habrá mucho avance a no ser que haya colaboración.

(Huang, 2021)

Capítulo 3

Criptografía general

En este apartado se va a tratar ejemplos que van a ayudar a comprender diferentes términos y técnicas importantes en la criptografía.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Se introducirán términos como la clave pública, cuerpos finitos, AES, Des, compartición de secretos, firma digital y por último se concluirá con los cifrados de bloque.

3.1 Seguridad en la computación post-cuántica.

Actualmente los cifrados que se encuentran en la red la mayoría utilizan la clave pública y privada. Teóricamente los ordenadores cuánticos tienen la capacidad de quebrar estos tipos de cifrado, actualmente no pueden quebrarlos debido a que se utilizan claves muy extensas y se carece de muchos qubit, en un futuro se prevé que podrán quebrarlos con el avance de la tecnología, además de que se podría usar este tipo de computación para simulaciones en física, aeroespacial entre otras.

Por eso se crearon tipo de competiciones como la de NIST de criptografía post-cuántica para prevenir y crear algoritmos post-cuánticos que sean resistentes a ataques de ordenadores cuánticos.

El proceso para verificar la seguridad de este tipo de algoritmos es la siguiente:

- Se diseñan los sistemas para cifrar y descifrar.
- Se pone a prueba este tipo algoritmos.
- Se implementan los algoritmos más rápidos.

Lo que se está haciendo actualmente para protegerse de este tipo de ataques cuánticos es aumentar el tamaño de las claves, ya que esto hace que el ordenador cuántico le cueste descifrarlo.

Se considera que las claves cuya longitud es 256 bits son prácticamente seguras, pero en un futuro se supone que no lo llegarán a ser y habrá que buscar formas nuevas para protegerse.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Para RSA se recomienda que sea de 2048-4096 bits, las implementaciones de RSA que utilizaban 768-1024 bits ya fueron quebrantadas en 2010.

Se estima que en 20 años ya un ordenador cuántico podrá romper los grandes criptosistemas actuales.

Actualmente existen varios algoritmos de cifrados resistentes a los ordenadores cuánticos, pero no son visibles a gran escala por el tiempo y espacio que ocupan, hay que generar una confianza al usuario y eso lleva tiempo, necesitamos realizar muchas pruebas para asegurarnos de que el producto es bueno y seguro, eso quiere decir que no genere contratiempos en un futuro.

(Orange, 2022)

3.2 Computación cuántica y clásica.

Se procederá a hacer una introducción de cada una para luego para terminar diferenciando una de otra.

Computación clásica: Los computadores clásicos se rigen mediante el uso de bits que pueden tomar dos estados: 0 y 1. Esto quiere reflejar que una computadora de N bits hace que la información que contiene es N en un estado concreto.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Si se aumenta en 1 el número de bits, el resultado sería $N+1$, esto hace que cambie de N a $N+1$.

(Salas Peralta & Sanz Sáenz, 2000)

Los ordenadores cuánticos utilizan puertas lógicas.

Computación cuántica: Los computadores cuánticos se rigen mediante el uso de qubit, esto hace que los estados no sean 0 y 1 si no ambos. Esto quiere reflejar que una computadora de N bits hace que la información que contiene es 2^N en un estado concreto.

Si se aumenta en 1 el número de bits, el resultado sería 2^{N+1} , esto hace que cambie de 2^N a 2^{N+1} .

Los ordenadores cuánticos utilizan puertas cuánticas esto hace que se hagan operaciones a un nivel óptimo y rápido.

(Salas Peralta & Sanz Sáenz, 2000)

3.3 Algoritmos cuánticos.

En este apartado se va a tratar de introducir a los cuatro algoritmos cuánticos más conocidos:

- Algoritmo de Shor: La función primordial de este algoritmo consiste en descomponer un número entero de gran tamaño en sus factores primos. Es un tipo de algoritmo cuántico de factorización. Presenta una ventaja exponencial ($O(\sqrt{N})$). Se

puede utilizar para solucionar el problema del subgrupo escondido también llamado HSP. El HSP se soluciona con la parallelización cuántica, la cuál es una propiedad que tienen los ordenadores cuánticos de realizar diferentes cálculos al mismo tiempo, de las que podemos destacar el logaritmo discreto para un grupo cíclico finito llamado G y para la factorización de enteros para $G = \mathbb{Z}$. El algoritmo Shor en ordenadores cuánticos ya establecidos podría romper algoritmos como el RSA y ECC. Este algoritmo está formado por dos pasos:

- El primer paso consiste en encontrar el período de una función el cuál es la longitud más pequeña de un intervalo, o coloquialmente la distancia que hay entre el eje x y dos puntos que se encuentran de forma sucesiva en los que la función posee un mismo valor y se trata de un patrón de repetición.

$$f(x) = f(x + T)$$

Figura 3.1 Fórmula para encontrar el período de una función.

- Esto indica que si le sumo un valor T a x la función tiene el mismo valor, en esto consiste el encontrar el período de una función.

Para la obtención del periodo se va a realizar una parte práctica para mostrar un ejemplo de la obtención:

- Se procede a escoger un número pseudoaleatorio (un número que parece aleatorio pero que no lo es, es generado por un algoritmo) que sea menor que N (número entero elegido previamente para el algoritmo Shor). El número pseudoaleatorio lo vamos a llamar y. $y < N$.

- Se procede a computar el máximo común divisor de N e Y.

Aquí puede haber dos soluciones:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

$\text{mcd}(y, N)$ es distinto a 1 es un factor no trivial de N .

Si es 1, debemos utilizar un subprograma para hallar x el periodo de la función explicado anteriormente en el primer paso.

- Si y es un número impar tenemos que volver a escoger un número pseudoaleatorio.

- Si $y^{x/2} \equiv -1 \pmod{N}$ tenemos que volver a escoger un número pseudoaleatorio.

- Los factores que tiene N son $\text{mcd}(y^{x/2} \pm 1, N)$.

El segundo paso consiste en encontrar un algoritmo cuántico para averiguar el período. Esto se realiza mediante la capacidad de un ordenador cuántico de estar en diferentes estados simultáneamente lo que es denominado como superposición cuántica. Hay que evaluar la función en todos los puntos de forma simultánea, esto la física cuántica no lo permite ya que una medición cuántica dará solamente un valor posible eliminando los demás, por lo tanto, tendríamos que buscar una forma de transformar la superposición a otro estado para que nos devuelva una respuesta correcta, para ello se utilizara la transformada de Fourier cuántica.

A continuación, se va a reflejar los pasos del subprograma para obtener el período.

- Inicializar dos registros de qubit de entrada y salida con $\log_2 N$ qubit cada uno.

$$N^{-1/2} \sum_x |x\rangle |0\rangle$$

(Wikimedia, s.f)

Figura 3.2 Fórmula para inicializar dos registros de qubit de entrada y salida.

El valor que recoge x es de 0 a $N-1$.

Se procede a construir $f(x)$ para luego aplicarla al estado antedicho. El resultado es:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

$$N^{-1/2} \sum_x |x\rangle |f(x)\rangle$$

(Wikimedia, s.f)

Figura 3.3 Fórmula f(x).

- Se aplica la transformada de Fourier cuántica a la entrada.

$$U_{QFT} |x\rangle = N^{-1/2} \sum_y e^{2\pi i xy/N} |y\rangle$$

Figura 3.4 Transformada de Fourier cuántica a la entrada. (Wikimedia, s.f)

-El resultado que nos deja el estado es:

$$N^{-1} \sum_x \sum_y e^{2\pi i xy/N} |y\rangle |f(x)\rangle$$

(Wikimedia, s.f)

Figura 3.5 Resultado del estado.

Se procederá a realizar una medición. Se obtiene un resultado y en el registro de entrada y $f(x_0)$ en el registro de salida. F es periódica:

$$N^{-1} \left| \sum_{x: f(x)=f(x_0)} e^{2\pi i xy/N} \right|^2 = N^{-1} \left| \sum_b e^{2\pi i (x_0 + rb)y/N} \right|^2$$

(Wikimedia, s.f)

Figura 3.6 Medición.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Esto indica que hay mayor probabilidad cuando y/N es un número casi igual a un número entero.

-Convertir y/N en una función irreducible, extraer r prima la cual es un candidato a ser r.

-Si $f(x) = f(x + r \text{ prima})$ ya habremos acabado ya que r prima seria nuestra r.

- Si $f(x)$ es distinto $f(x + r \text{ prima})$, probar valores más cercanos a y, o múltiplos de r prima.

- Si no se encuentra ni uno tendremos que inicializar otro par de registros y volver a hacer todos los pasos otra vez.

(Wikimedia, s.f)

- Algoritmo de Grover: Consiste en un algoritmo creado en 1996 por Lov Grover. Es junto al algoritmo de Shor los primeros algoritmos cuánticos que resolvían un problema de una forma óptima que los clásicos. Lov Grover con este algoritmo consiguió encontrar el elemento óptimo entre un conjunto de datos. Presenta una ventaja cuadrática($O(N)$). Este algoritmo en concreto es usado para acelerar diferentes algoritmos, se ha demostrado aceleraciones en problemas de caja negra en la complejidad de la consulta cuántica, la distinción de elementos y el problema de colisión. (Hmong, s.f)

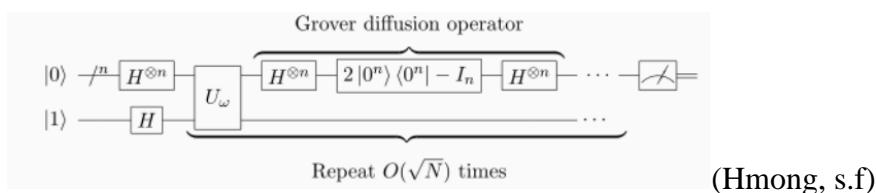


Figura 3.7 Circuito cuántico con algoritmo de Grover.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Este algoritmo está formado por tres pasos:

- Inicializar el sistema de superposición uniforme en todos los estados que hay:

Hay que realizar una iteración Grover $r(N)$ veces:

- Se debe aplicar el operador U_w

Se debe aplicar también el operador de difusión de Grover

$$U_s = 2|s\rangle\langle s| - I.$$

$|s\rangle$ es el estado de superposición uniforme.

I consiste en la matriz de identidad.

$|s\rangle$ consiste en el estado.

Se debe medir para finalizar el estado cuántico resultante en la base computacional.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle. \quad (\text{Hmong, s.f})$$

Figura 3.8 Estado cuántico.

Esta medición da un resultado que se aproxima al estado de mayor probabilidad.

(Hmong, s.f)

3.4 Puertas lógicas vs cuánticas.

- Computación clásica:

Para la computación clásica el manejo de los bits se realiza mediante puertas lógicas.

-Computación cuántica:

Para la computación cuántica el manejo de los qubit se realiza mediante

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

puertas cuánticas.

Estos son unos ejemplos de puertas lógicas:

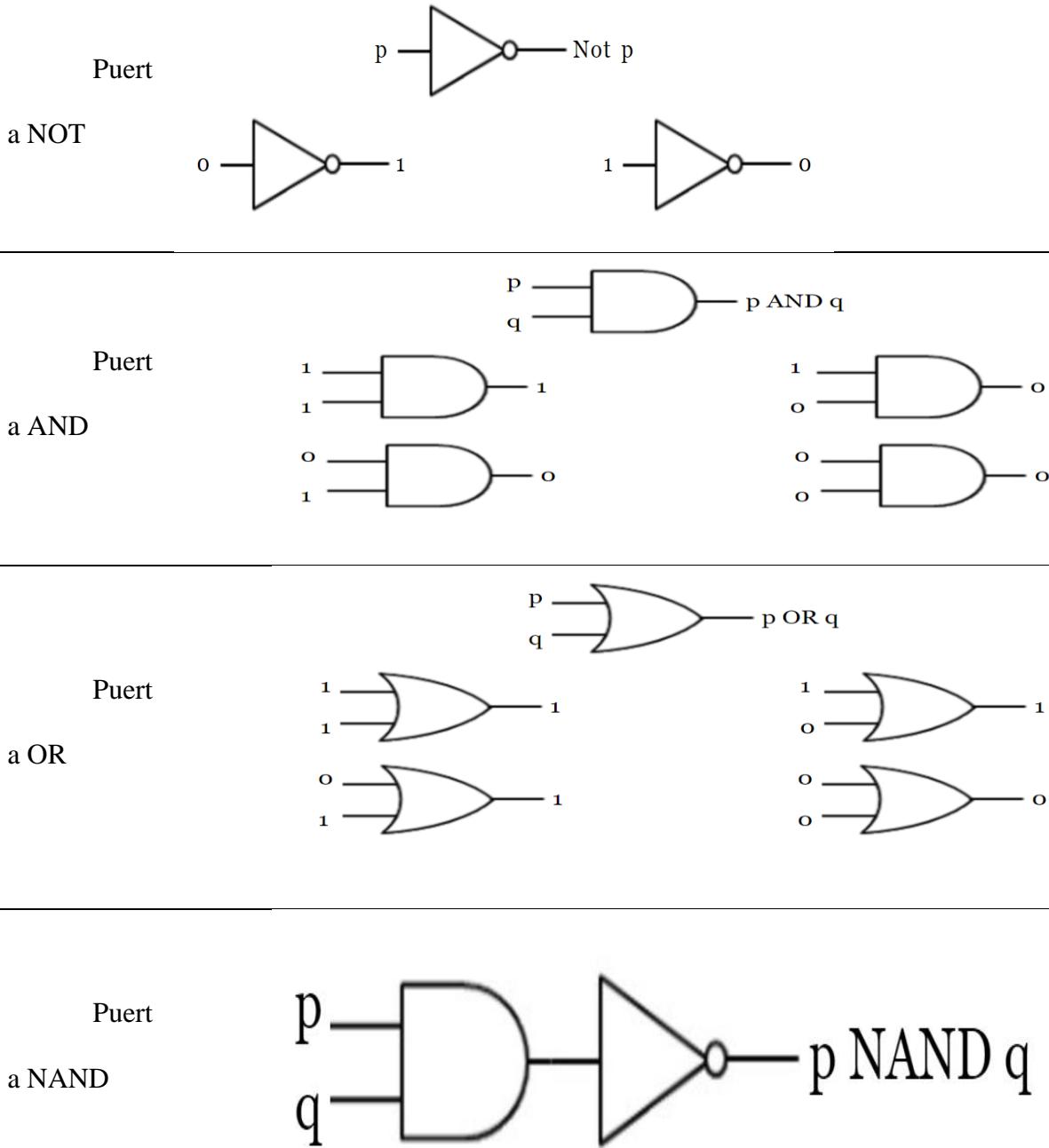


Figura 3.9 Puerta NOT. (González Cornejo, 2020)

Figura 3.10 Puerta AND. (González Cornejo, 2020)

Figura 3.11 Puerta OR. (González Cornejo, 2020)

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Figura 3.12 Puerta NAND. (González Cornejo, 2020)

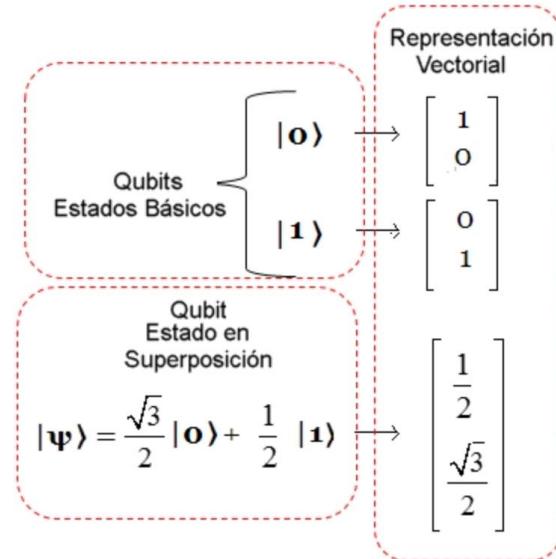


Figura 3.13 Representación vectorial de Qubit básicos y en superposición. (González Cornejo, 2020)

Podemos destacar seis diferentes puertas cuánticas:

- Puerta Toffoli:

Consiste en una puerta cuántica de tres bits de entrada y tres bits de salida.

Es conocida como la puerta NOT de doble control también llamada CCNOT. Si los dos primeros bits son 1, el ultimo bit se invierte.

Esta puerta no es universal para la computación cuántica.

Matriz de permutación

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 3.14 Matriz de permutación. (Frwiki, s.f)

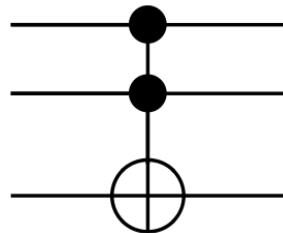


Figura 3.15 Puerta Toffoli. (Frwiki, s.f)

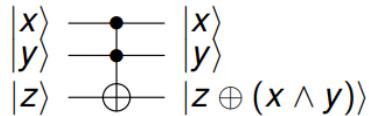


Figura 3.16 Entrada de 3 bits Toffoli. (Frwiki, s.f)

- Puerta CNOT:

Es una puerta de dos qubit o más. Es la puerta NOT controlada, la operación NOT se realiza en el segundo qubit si el primer qubit es $|1\rangle$, si no fuese el caso se queda igual.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 3.17 Matriz CNOT. (Wikipedia, s.f)

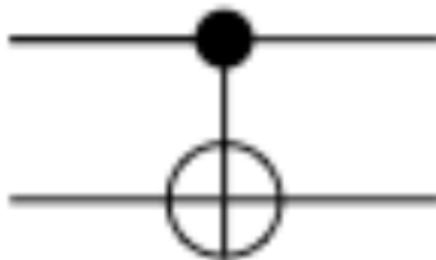


Figura 3.18 Puerta CNOT controlada en circuito. (Wikipedia, s.f)

- Puerta de Hadamard:

Es una puerta de un solo qubit. Se produce una superposición de los estados 0 y 1.

El estado base $|0\rangle$ pasa a $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ y $|1\rangle$ a $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Equivale a una rotación de la esfera Bloch alrededor del eje x y z en radianes.



Figura 3.19 Puerta cuántica de Hadamard. (Wikipedia, s.f)

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Figura 3.20 Matriz de Hadamard. (Wikipedia, s.f)

- Puerta de Pauli-Z:

Es una puerta de un solo qubit. Equivale a una rotación de la esfera Bloch alrededor del eje z en radianes. (Wikipedia, s.f)

Su matriz es la siguiente:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figura 3.21 Matriz de Pauli-Z. (Wikipedia, s.f)

- Puerta de Pauli-Y:

Es una puerta de un solo qubit. Equivale a una rotación de la esfera Bloch alrededor del eje Y en radianes. (Wikipedia, s.f)

Su matriz es la siguiente:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Figura 3.22 Matriz de Pauli-Y. (Wikipedia, s.f)

- Puerta Pauli-X:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Es una puerta de un solo qubit. La puerta lógica que equivale a esta es la puerta NOT. Equivale a una rotación de la esfera Bloch alrededor del eje X en radianes. (Wikipedia, s.f)

Su matriz es la siguiente:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figura 3.23 Matriz de Pauli-X. (Wikipedia, s.f)



Figura 3.24 Diagrama cuántico CNOT. (Wikipedia, s.f)

3.5 Estándares y protocolos en la criptografía post-cuántica.

Hay diferentes tipos de estándares y protocolos cuánticos, pero se va a tratar solo cuatro que se han considerado de los más importantes:

La función que representa todo esto es la siguiente:

$C_i = M_i \oplus K_i$ para $i = 1, 2, \dots, n$. Esto es para el cifrado.

$C_i \oplus K_i = M_i \oplus K_i \oplus K_i$. Esto es para el descifrado.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

M_i son los n bits de cada carácter del texto, K_i la clave y C_i el mensaje cifrado.

- BB84: Consiste un protocolo de criptografía cuántica el cual fue creado por Charles Bennett y Gilles Brassard en el año 1984. Su función primordial es el intercambio de claves a través de la comunicación cuántica.

Se procede a representar a dos tipos de personajes ficticios los cuales son Alice y Bob para explicar el movimiento de datos.

Ambos se comunican por Internet y por un canal cuántico. Este protocolo facilita la detección de un espía o por un medio cuántico o por uno clásico, nunca en ambos.

El BB84 se divide en diversos pasos en los que se trata la información entre Alice y Bob:

- Paso 1: Alice debe generar N bits, para el bit 0 se otorga una base cuántica y para el bit 1 otra diferente.

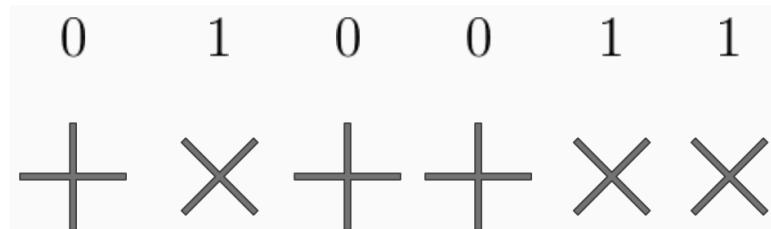


Figura 3.25 Otorgamiento de bases. (Owasp, s.f)

- Paso 2: Alice vuelve a generar N bits de forma aleatoria, en la posición de un 0 asigna un estado asociado con 0 y cuando es un 1 con un 1. Una vez asignados Alice se encarga de enviar estos estados a Bob.

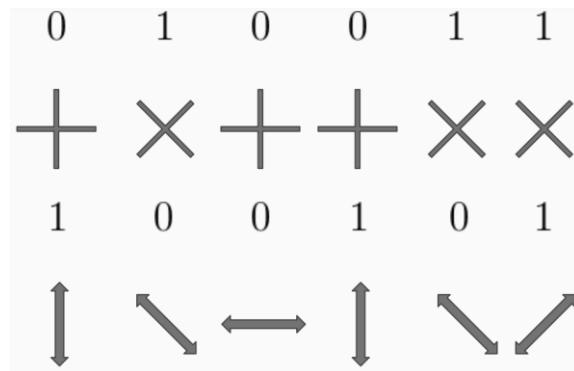


Figura 3.26 Asignar estados y envío. (Owasp, s.f)

- Paso 3: Bob genera N bits de forma aleatoria y los asocia a las bases cuánticas.

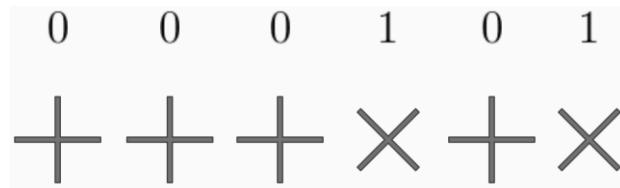


Figura 3.27 Asociar N bits a bases cuánticas. (Owasp, s.f)

- Paso 4: Bob se encarga de medir en qubit el mensaje que le envía Alice.

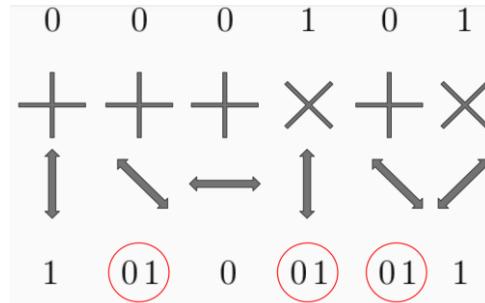


Figura 3.28 Medición del mensaje. (Owasp, s.f)

- Paso 5: Alice y Bob proceden a intercambiar las bases por Internet, las comparan, si son iguales se marcan como medidos, si no se marcan como inválidas.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

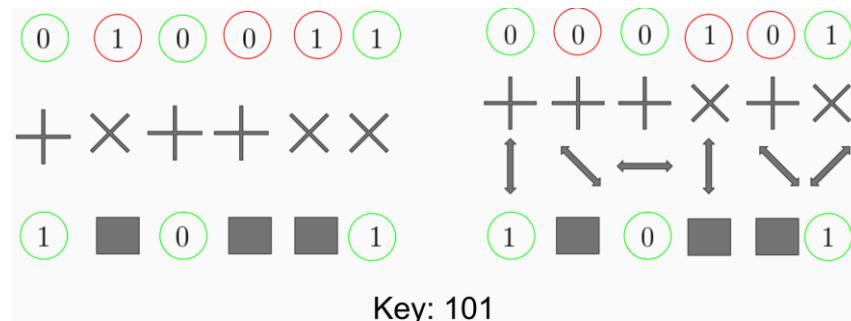


Figura 3.29 Intercambio de bases cuánticas. (Owasp, s.f)

(Owasp, s.f)

- E91: Consiste en un protocolo de criptografía cuántica creado por el físico polaco Artur Ekert en 1991. Se hizo famoso por ser uno de los primeros protocolos que utilizaba el entrelazamiento (correlación entre partículas cuánticas) o también llamado intrincación cuántica, también es el primer protocolo en usar pares entrelazados, esto consigue una transmisión de información entre Alice y Bob de forma más segura alrededor de un canal cuántico. Estos pares entrelazados son recibidos por los dos emisores y a la vez interpreta el estado que ha recibido por el canal mediante el uso de una base elegida aleatoriamente. Al acabar, los interlocutores intercambian por un canal público las bases, y se desechan los valores con bases distintas. El resultado correcto final consiste en una clave de ambos interlocutores la cual es idéntica o similar.

El E91 se divide en diversos pasos en los que se trata la información entre Alice y Bob:

- Paso 1: Alice genera una secuencia de bases aleatoria.
- Paso 2: Bob genera una secuencia de bases aleatoria.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Paso 3: La fuente el cuál sistema que se encarga de generar pares de partículas entrelazadas envía pares entrelazados a ambos extremos.
- Paso 4: Alice interpreta los estados recibidos, según la secuencia creada por la misma.
- Paso 5: Bob interpreta los estados recibidos, según la secuencia creada por el mismo.
- Paso 6: Alice y Bob repiten los pasos 4 y 5 hasta que no se deje de emitir los pares ERP. Una vez que se dejan de emitir ambos interlocutores intercambian la secuencia de bases que se ha utilizado por medida.
- Paso 7: Alice o Bob ejerce una respuesta sobre el otro para manifestar la secuencia de valores indicando la base que se usó para hacer la medida.

(Martínez Mateo, 2008)

- SARG04: Consiste en un protocolo que surgió al usar el protocolo BB84 con otro método de descifrar la información. Utiliza pulsos de laser en vez de fotones. Todo esto hace que sea más robusto que el BB84.

Es más seguro contra ataques incoherentes de PNS

3.6 Máquina de Turing.

La máquina de Turing consiste en un modelo computacional el cual tiene funciones de escritura y lectura automáticamente sobre una cinta, la cual se genera una respuesta en esta misma cinta. En 1936, Alan Turing introdujo la máquina de Turing en el proyecto “On computable numbers, with an application to the

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Entscheidungsproblem” en Londres. Este proyecto trataba el tema de si las matemáticas son decidibles, lo cual es lo mismo que poder comprobar en todos los algoritmos su veracidad. Esta máquina verifica que hay problemas que una máquina no puede realizar o resolver. (EcuRed, s.f)



Figura 3.32 Máquina de Turing . (El País, s.f)

Está formada por:

- Alfabeto de entrada y respuesta.
- Un símbolo denominado blanco.
- Conjunto de transiciones entre estados y estados finitos.

El funcionamiento es simple, la máquina de Turing procede a leer una celda de la cinta por cada iteración, elimina el símbolo donde está ubicado el cabezal y añadiendo el nuevo símbolo de salida, después se mueve el cabezal a la izquierda o derecha. Este proceso se reitera hasta finalizar en el estado final. (EcuRed, s.f)

3.7 Máquina de Turing cuántica.

En 1985, Deutsch creó la máquina de Turing cuántica. Está formada por las mismas piezas que la máquina original lo único es que el elemento es un qubit.

El procesador de la máquina posee una guía con las indicaciones que se aplican al elemento de la cinta que marca el cabezal y realiza una instrucción por unidad de tiempo. (EcuRed, s.f)

3.8 Qubit.

Un qubit o también llamado bit cuántico es la unidad de información de la computación cuántica.

Al hablar de un qubit podemos relacionar el término de superposición, gracias a la superposición se logra una combinación lineal entre dos estados.

El bit ordinario solo se puede representar con un cero o uno, en cambio el qubit se puede representar por un cero, un uno o alguna proporción de cero y uno en la superposición de ambos, como se puede mostrar en la siguiente fotografía. (Azure Microsoft, s.f)

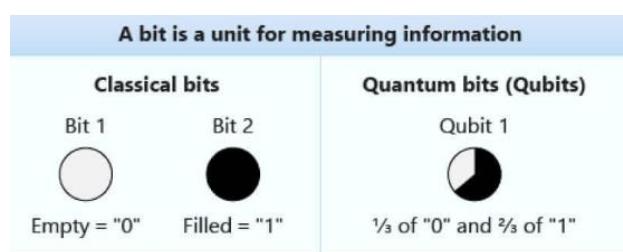


Figura 3.33 Diferencia entre qubit y bit . (Azure Microsoft, s.f)

3.9 Algoritmos vulnerables a ataques cuánticos.

Podemos destacar ciertos algoritmos vulnerables a ataques cuánticos. Basados en factorización destaca RSA, y en logaritmo discreto destaca ElGamal y curvas

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

elípticas.

De los algoritmos no vulnerables podemos destacar los de cifrado simétrico y la compartición de secretos como el algoritmo Shamir.

“Durante la Conferencia Enigma 2023 celebrada en Santa Clara (California) el pasado 24 de enero, el científico informático y experto en seguridad y privacidad, **Simson Garfinkel**, aseguró públicamente que la computación cuántica tenía pocas aplicaciones y recursos para **acabar con el cifrado RSA**. Además, añadía que si nos centramos en este asunto dejaremos la puerta abierta a otras amenazas inminentes.” (Delgado, 2023)

Sin embargo, el algoritmo RSA es vulnerable debido a que realiza operaciones de factorización de números grandes esto hace que los ordenadores cuánticos estén más preparados. Mediante el algoritmo Shor pueden hacerlo.

ElGamal es vulnerable debido al problema del logaritmo discreto, el cual también puede ser resuelto por un ordenador cuántico. Un ordenador cuántico con demasiada potencia podría romperlo.

Las curvas elípticas como el cifrado ECC también son vulnerables debido al problema del logaritmo discreto en curvas elípticas.

Los algoritmos de clave simétrica en teoría no son vulnerables a ataques cuánticos ya que no poseen problemas matemáticos como los asimétricos, un ejemplo de este es el algoritmo AES el cuál para aliviar un ataque se puede utilizar una clave más grande y esto complicara el ataque.

Capítulo 4

Concurso NIST

4.1 Descripción del concurso.

Es un concurso cuya finalidad consiste en ayudar a empresas, bancos, y distintas organizaciones a evitar riesgos de seguridad, mantener la información segura para que no caiga en malas manos en un futuro. En estos últimos años ha habido un aumento de investigación sobre la computación cuántica, el problema surge cuando se creen

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

ordenadores cuánticos a escala mundial, estos tendrían la capacidad de poder romper bastantes criptosistemas de clave pública. El objetivo primordial de la criptografía post-cuántica es que se puedan generar criptosistemas seguros para los ordenadores clásicos y los cuánticos. Algunos expertos indican que en 20 años se crearán ordenadores cuánticos de grandes tamaños que podrán romper todos los criptosistemas de clave pública. Hay 5 áreas esenciales en el marco de la ciberseguridad del concurso NIST:

- Identificación: Se debe identificar todos los elementos electrónicos, toda la parte de software que son usados por la empresa. Se debe elaborar una política de ciberseguridad la cual tiene que cumplir lo siguiente: Las funciones y responsabilidades de los integrantes y terceros de la empresa, una forma de actuar ante ataques para reducir riesgos.
- Protección: Utilización de programas seguros, mantener un control de quien accede a nuestros equipos y red, realizar copias de seguridad, eliminar datos irrelevantes, encriptar alguna información relevante.
- Detección: Examinar que usuarios no están autorizados a acceder a la red de la empresa, revisar la actividad maliciosa en la persona o en la red de la empresa.
- Respuesta: Revisar y mantener a raya un ataque, ir mejorando la política de ciberseguridad, poner en conocimiento de las autoridades competentes si ha habido alguna irregularidad, avisar a los clientes que han sido afectados por un ataque.
- Recuperación: Avisar a los empleados y clientes si se va a hacer un recuperado, revisar y restaurar los equipos afectados.

En el año 2020, el NIST anuncio los finalistas de la Ronda 3 mediante un proceso de estandarización post-cuántica. Se inicio el proceso con 69 algoritmos en la Ronda 1 en el año 2017, posteriormente en la Ronda 2 se redujo a 26 algoritmos y

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

finalmente en la Ronda 3 se redujo a 15 algoritmos. De estos 15 algoritmos se van a dividir en finalistas y suplentes, los suplentes serán estandarizados en la Ronda 4.

(Comisión federal de comercio, s.f)

4.2 Descripción y análisis de los finalistas.

Los algoritmos de firma digital vamos a omitirlos ya que no son relevantes para este trabajo. En la siguiente tabla van a venir especificados los algoritmos de la Ronda 3 por tipo, y una breve descripción de ellos.

Mecanismo de encapsulamiento de clave		
Tipo	Finalista	Alternativa
Basado en teoría de códigos	Classic McEliece	BIKE

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

correctores	HQC
	CRYSTALS-KYBER
Basado en retículos	NTRU
	SABER
Basado en isogenias de curvas elípticas	SIKE
(PQSecure, 2020)	

Para empezar, vamos a tratar el algoritmo ganador el cuál ha resultado ser el CRYSTAL-KYBER, posteriormente el McEliece, NTRU, SABER los cuáles han resultado finalistas, luego se procederá a describir brevemente las alternativas.

CRYSTALS-KYBER: Es un tipo de mecanismo de encapsulamiento de clave lo que se denomina también como KEM, la complejidad de este algoritmo se basa en resolver el aprendizaje con errores en las redes de módulos. Este algoritmo tiene tres niveles de seguridad. Kyber-512 cuya seguridad es similar a AES-128, Kyber-768 a AES-192 y Kyber-1024 a AES-256. Este esquema se basa en un esquema de cifrado seminal realizado por Regev. Lo bueno de este tipo de esquema están formados por la misma disposición que el ruido y además permite formar esquemas parecidos al LWE usando un cuadrado en vez de un rectángulo para una clave pública. Recientemente se añadió otro tipo que se creó e integro en la Ronda 2 del concurso NIST. (PQSecure,

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

2020)

McEliece: Es un criptosistema de clave pública, el cuál toma el nombre de su creador McEliece en el año 1978. La clave pública detalla un código Goppa binario aleatorio, la clave privada tiene la función de decodificar un código de forma eficiente.

El proceso de la clave privada consta en tres pasos:

- Obtener la clave del texto cifrado.
- Identificar los errores.
- Corregir los errores.

El algoritmo McEliece fue diseñado de tal forma que se utilizó una de las propiedades de la clave pública la unidireccional, esto hace que no se pueda averiguar de forma eficaz la clave a través de la clave pública y el texto cifrado, esto se debe a que la clave se toma al azar.

Actualmente el algoritmo McEliece ha sabido mantenerse en el tiempo, además de ofrecer una buena seguridad. Se diseño para 2^{64} de seguridad, esto quiere decir que un ataque 2^{64} son las operaciones que necesita un intruso para poder deshacer el algoritmo mediante un ataque de fuerza bruta. Lo bueno es que es un sistema escalable esto va a provocar que haya seguridad para las nuevas técnicas de la informática, como pueden ser los ordenadores cuánticos.

El algoritmo genera demasiado trabajo de seguimiento a la comunidad, esto genera mejoras. Gracias a este trabajo se ha producido mejoras en la eficiencia además en la seguridad, se podría hablar de mejoras sobre el PKE propuesto por Niederreiter, mejoras en la velocidad del software y hardware. Actualmente existe la capacidad de convertir un OW-CPA PKE es un KEM el cuál es IND-CCA2 lo que permite que sea

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

seguro ante ataque de ROM. El OW-CPA es un ataque de texto en claro de una sola dirección, es un esquema PKE el cuál es robusto ante ataques OW-CPA. IND-CCA2 indica que es resistente a ataques en las que el intruso o atacante tienen la capacidad de formular consultas de descifrado adaptativo. ROM es un modelo de la criptografía que tiene la función de revisar la seguridad del algoritmo. La conversión se rige mediante las siguientes normas:

- PKE es determinista.
- PKE no tiene fallos en texto cifrados correctos.

Para resumir el Classic McEliece consiste en un KEM que permite la seguridad IND-CCA2, además en ordenadores cuánticos. Este nivel de seguridad va aumentando para que en el tiempo se pueda proporcionar la seguridad necesaria ante ordenadores cuánticos. (PQSecure, 2020)

NTRU: Es un KEM que consta de dos algoritmos, el NTRUEncrypt que tiene la función de cifrar y descifrar y el NTRUSign se usa para clave pública. Aunque esta “suite” (término para referirnos a NTRUEncrypt y NTRUSign como un conjunto) fue patentada, hasta el año 2017 no fue mostrada de forma pública, se puede usar en aplicaciones con licencia GPL. Tiene la capacidad de realizar operaciones de clave privada de forma más rápida que el RSA y manteniendo la seguridad del RSA. Esto se debe a que el RSA realiza operaciones incrementando con el cubo de la clave, sin embargo, en NTRU es cuadrático. El NTRU es invulnerable a los ataques cuánticos por ahora, y el RSA es vulnerable. En el año 2009 Perlner y Cooper indicó que el algoritmo NTRU podría ser la alternativa post-cuántica en cifrado y firma digital al algoritmo Shor. (PQSecure, 2020)

SABER: Es un KEM que permite la seguridad IND-CCA2, además en

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

ordenadores cuánticos, a través del problema del módulo con redondeo. Este algoritmo consta de tres niveles de seguridad:

- SABER: Parecido en términos de seguridad al AES-192.
- FireSABER: Parecido en términos de seguridad al AES-256.
- LightSABER: Parecido en términos de seguridad al AES-128.

El algoritmo tiene tres características claves:

- Se usa el aprendizaje con redondeo.
- Cada módulo son potencia de dos.
- Esta estructura que posee el módulo ofrece flexibilidad.

Ventajas del algoritmo SABER:

- Ideal para el anonimato en la comunicación, ejemplos como el navegador Tor.
- Es eficiente además de ser simple.

(PQSecure, 2020)

BIKE: Es un KEM basado en códigos correctores de errores, el cuál consiguió llegar a la ronda final del concurso NIST. Su seguridad es IND-CCA. El algoritmo BIKE ha ido evolucionando a lo largo del concurso NIST. En la primera ronda solo era seguro en ataques de texto plano. En la segunda ronda se realizaron tres nuevas variantes que ofrecían seguridad en ataques de cifrado. El proceso del algoritmo nos hace primero realizar una implementación de forma segura del protocolo de ataques de texto cifrado y mostramos que el rendimiento es ligeramente peor que una implementación segura del protocolo de ataques de texto plano. Una de las fases del algoritmo llamada desencapsulación, el algoritmo BIKE usa otro algoritmo para

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

decodificar que tiene un porcentaje de fallo llamado tasa de decodificación. El siguiente proceso consiste en analizar la tasa de decodificación de dos decodificadores diferentes de BIKE los cuales son Back-Flip y Black-Gray, posteriormente utilizaremos un nuevo decodificador llamado Black-Gray-Flip el cual tiene la misma tasa de decodificación que los anteriores, pero es el doble de rápido. (PQSecure, 2020)

Finalmente se plantea un algoritmo para la inversión de polinomios binarios de la variante de BIKE llamada BIKE-2. Esta implementación nos indica que BIKE-2 es menor que sus variantes, esto la hace la favorita. (PQSecure, 2020)

HQC: Es un KEM basado en códigos correctores de errores, el cuál es considerado como una alternativa y no un algoritmo finalista. Su seguridad es IND-CPA, seguro frente ataques de texto plano. Es dirigido a los niveles 1, 3 y 5 del NIST. Mas eficiente que el algoritmo BIKE, además de poseer el texto cifrado y las claves públicas levemente superiores a BIKE. (PQSecure, 2020)

FrodoKEM: Es un KEM basado en retículos, el cuál es considerado como una alternativa y no un algoritmo finalista. Su seguridad es IND-CCA. Tiene una clave pública mayor pero menos restricciones de parámetros. Es dirigido a los niveles 1, 3 y 5 del NIST. (PQSecure, 2020)

NTRU Prime: Consiste en una modificación del NTRU de los años 90, para poder usar anillos sin las mismas estructuras requeridas. Su seguridad es IND-CCA2. (PQSecure, 2020)

SIKE: Es el único KEM del concurso NIST que está basado en isogenias de curvas elípticas. Utiliza operaciones de criptografía con curvas elípticas. Tiene la clave pública más pequeña que hay entre todos los candidatos y a su vez es la más lenta. (PQSecure, 2020)

4.3 Estudio comparativo.

En las siguientes tablas se va a mostrar un ejemplo del rendimiento de los diferentes tipos del algoritmo que hay por seguridad, va a estar medida por un procesador Intel-Core-i7 4770K, se va a tomar de referencias una implementación en C y otra en AVX2.

Kyber-512

Tamaño(bytes)	Ciclos Haswell (C)	Ciclos Haswell (avx2)
---------------	--------------------	-----------------------

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Clave secreta:	1632	Generación de claves:	122684	Generación de claves:	33856
Paquete de datos:	800	Número de ciclos de CPU para cifrar datos:	154524	Número de ciclos de CPU para cifrar datos:	45200
Connecticut:	768	Número de ciclos de CPU para operación de diccionario:	187960	Número de ciclos de CPU para operación de diccionario:	34572

(Pq-Crystals, 2020)

Kyber-768

Tamaño(bytes)	Ciclos Haswell (C)	Ciclos Haswell (avx2)
Clave secreta:	2400	Generación de claves: 199408
Paquete de datos:	1184	Número de ciclos de CPU para 235260
		Número de ciclos de CPU para 67624

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

cifrar datos:	cifrar datos:
Número de ciclos de	Número de ciclos de
Connecticut: 1088	CPU para 274900
operación de diccionario:	operación de diccionario:

(Pq-Crystals, 2020)

Kyber-1024

Tamaño(bytes)	Ciclos Haswell (C)	Ciclos Haswell (avx2)
Clave secreta: 3168	Generación de claves: 307148	Generación de claves: 73544
Paquete de datos: 1568	Número de ciclos de CPU para 346648	Número de ciclos de CPU para 97324

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

cifrar datos:	cifrar datos:
Número de ciclos de	Número de ciclos de
Connecticut: 1568	CPU para 396584
operación de diccionario:	operación de diccionario:

(Pq-Crystals, 2020)

Kyber-512-90s

Ciclos Haswell (C)	Ciclos Haswell (avx2)
Generación de claves: 213156	Generación de claves: 21880
Número de ciclos de CPU para cifrar 213156 datos:	Número de ciclos de CPU para cifrar 28592 datos:
Número de ciclos de CPU para operación 277612 de diccionario:	Número de ciclos de CPU para operación 20980 de diccionario:

(Pq-Crystals, 2020)

Kyber-768-90s

Ciclos Haswell (C)	Ciclos Haswell (avx2)
--------------------	-----------------------

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Generación de claves:	389760	Generación de claves:	30460
Número de ciclos de datos:	432764	Número de ciclos de datos:	40140
Número de ciclos de de diccionario:	473984	Número de ciclos de de diccionario:	30108
(Pq-Crystals, 2020)			

Kyber-1024-90s

Ciclos Haswell (C)	Ciclos Haswell (avx2)
Generación de claves:	636380
Número de ciclos de datos:	672644
Número de ciclos de de diccionario:	724144
(Pq-Crystals, 2020)	

Hay diferencias entre cada una de estas versiones del algoritmo, se comparan las versiones implementadas de referencia y después las versiones optimizadas en avx2.

Se puede apreciar que las dos claves y el texto cifrado es mayor de una versión a otra,

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

como se muestra en la anterior figura. En las versiones avx2 el número de ciclos para cifrar, el número de ciclos para descifrar y la generación de claves es menor respecto al de referencia, lo cual tiene su lógica debido a que es una versión optimizada. En las versiones especiales Kyber-90 ocurren situaciones diferentes, si comparamos kyber-512 y kyber-512-90s la generación de claves, el número de ciclos para cifrar, el número de ciclos para descifrar en referencia es mayor en 90s, sin embargo, en avx2 es menor que la generada en Kyber-512, ocurre lo mismo en las otras versiones.

En la siguiente tabla se muestra los tamaños tanto del texto cifrado, la clave privada y la clave pública de las versiones del algoritmo McEliece

Tipo	Texto cifrado	Clave pública	Clave privada	Clave sesión
McEliece348864	96	261120	6492	32
McEliece460896	156	524160	13608	32
McEliece6688128	208	1044992	13932	32
McEliece6960119	194	1047319	13948	32
McEliece8192128	208	1357824	14120	32

(McEliece, 2022)

Se puede observar que en cuanto mayor es la versión del algoritmo mayor es el texto cifrado, clave pública y privada, exceptuando la versión McEliece6688128 que tiene una mayor clave pública y texto cifrado que McEliece6960119. Se puede observar que la clave de sesión la cuál va a ser temporal es de 32 bytes o lo que quiere decir 256 bits.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Mceliece	Cuartil	Mediana	Cuartil
mceliece348864 cifrado	34951	36457	38980
mceliece460896 cifrado	69674	76086	88956
mceliece6688128 cifrado	165296	171442	185077
mceliece6960119 cifrado	139980	144678	149592
mceliece8192128 cifrado	155174	156945	159040
mceliece348864 descifrado	127036	127140	127256
mceliece460896 descifrado	262919	263046	263225
mceliece6688128 descifrado	305910	306212	306925
mceliece6960119 descifrado	286353	286596	287038
mceliece8192128 descifrado	309938	310097	310475
mceliece348864 clave compartida	35039714	56705880	67615011
mceliece348864f clave compartida	35970884	35976620	35981416

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

mceliece460896	clave compartida	116209838	153266214	264539700
mceliece460896f	clave compartida	117267744	117297677	117331130
mceliece6688128	clave compartida	265554240	443746986	532990499
mceliece6688128f	clave compartida	274329761	274384229	274430338
mceliece6960119	clave compartida	241288202	316995472	468394597
mceliece6960119f	clave compartida	240198020	240226771	240254131
mceliece8192128	clave compartida	308008713	486195290	664466919
mceliece8192128f	clave compartida	306203040	306238935	306280509

(McEliece, 2022)

Se muestra que se ha dividido los ciclos en tres fases, la del primer cuartil la cuál muestra el valor más pequeño del 25 por ciento de menor valor de las mediciones. La mediana es el valor medio de todas las mediciones hechas. Este ayuda a reflejar el rendimiento típico. El tercer cuartil muestra el valor más pequeño del 25 por ciento de mayor valor de las mediciones

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

ntruhaps2048509

Tamaño(bytes)	Ciclos Haswell(ref)	Ciclos Haswell(avx2)
sk: 935	gen: 12506668	gen: Aún no medido
pk: 699	enc: 761236	enc: Aún no medido
ct: 699	dec: 1940870	dec: Aún no medido

ntruhaps2048677

Tamaño(bytes)	Ciclos Haswell(ref)	Ciclos Haswell(avx2)
sk: 1235	gen: 21833048	gen: Aún no medido
pk: 931	enc: 1313454	enc: Aún no medido
ct: 931	dec: 3399726	dec: Aún no medido

ntruhaps4096821

Tamaño(bytes)	Ciclos Haswell(ref)	Ciclos Haswell(avx2)
sk: 1592	gen: 31835958	gen: Aún no medido
pk: 1230	enc: 1856936	enc: Aún no medido
ct: 1230	dec: 4920436	dec: Aún no medido

ntruhrss701

Tamaño(bytes)	Ciclos Haswell(ref)	Ciclos Haswell(avx2)
sk: 1452	gen: 23302424	gen: 381476
pk: 1138	enc: 1256210	enc: 71238
ct: 1138	dec: 3642966	dec: 77848

(Chen, y otros, 2019, p. 29)

Se muestra la información de las distintas versiones de ntru. En las diferentes versiones el valor en bytes de la clave secreta es mayor que el de la clave privada y el

texto cifrado, y cuanto mayor es la versión mayor es el tamaño de los parámetros citados. Respecto a los ciclos Haswell implementados en referencia en la generación de claves el que mayor tamaño tiene es el ntruhp4096821 seguido de ntruhrss701, ntruhp2048677 y ntruhp2048509 sucesivamente, en el proceso de cifrado el mayor tamaño es de ntruhp4096821 nuevamente seguido de ntruhp2048677, ntruhrss701 y ntruhp2048509, en el proceso de descifrar el mayor tamaño es de ntruhp4096821 nuevamente seguido de ntruhrss701, ntruhp2048677 y ntruhp2048509 como la generación de claves. En los ciclos Haswell de generación de claves, cifrado y descifrado de ntruhp2048509, ntruhp2048677, ntruhp4096821 aún está por definir, sin embargo, para ntruhrss701 sí que tenemos los tres parámetros, comparándolos con los de referencia de la misma versión son bastante inferiores.

Los datos reflejados nos indican que el algoritmo kyber, en especial kyber-90s en su versión optimizada es más rápido que los demás, en el tamaño de clave privada y pública la más grande es la de McEliece sobre todo la de mcEliece8192128. En términos de generación de clave el más rápido es el Kyber512-90s.

4.4 Futuro de la criptografía post-cuántica.

El futuro de la criptografía post-cuántica por ahora es incierto, aunque se tratará los riesgos, beneficios y consecuencias que puede acarrear este nuevo sector.

Riesgos:

- Mejoras o inclusión de nuevas técnicas que serían capaz de romper los algoritmos post- cuánticos, aunque estos en un futuro sean resistentes a ataques cuánticos. La evolución constante de la tecnología y por lo tanto de la criptografía.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- La falta de estudio actualmente puede hacer que en un futuro surjan nuevos inconvenientes que hagan retrasar el proceso de estandarización.
- Se necesitarán actualizaciones o mejoras en el algoritmo constantemente para asegurar la seguridad.
- Pasa de un modelo tradicional a uno cuántico no es tarea fácil, produciría un alto costo económico y logístico.

Beneficios:

- Seguridad ante ataques de algoritmos cuánticos por ordenadores cuánticos. Debido a que estaríamos preparados ante factorizaciones de números de gran tamaño.
- Mantener un control o preparación ante ataques cuánticos en un futuro hace que estemos más preparados.
- Evolucionar en los aparatos o sistemas que estaban acostumbrados al sistema clásico a ahora el sistema cuántico.
- Avance en las matemáticas, programación.

Consecuencias:

- Obligación de las sociedades, organizaciones o empresas a realizar avances en la computación cuántica para poder mantener sus intereses seguros.
- Realizar transformaciones en todos los aparatos, electrónica para que sea capaz de mantener los equipos seguros.
- Cooperación entre países para mantener la seguridad de ellos a salvo.
- Mantener un control sobre los avances en el sector.

Capítulo 5

Marco práctico

5.1 CRYSTAL-KYBER.

Se ha procedido a descargar el algoritmo ganador del concurso NIST. En un primer tema se intentó ejecutar en Windows, pero al dar complicaciones a la hora de compatibilidades con la instalación de alguna biblioteca requerida de OpenSSL.

El programa es el oficial facilitado por la página oficial del algoritmo. Consta de dos tipos de implementaciones, ref que es la que se va a usar y avx2 la cual está diseñada para CPU x86.

Como requisitos previos se exige la instalación de OpenSSL ya que hay funciones que se aprovechan de sus bibliotecas. Los pasos en la terminal constan de diversos pasos:

- Instalar dependencias: Debemos tener instalado en la terminal las siguientes dependencias: make y update.

Los comandos necesarios serían:

- sudo apt-get update. Para instalar la dependencia update.
- sudo apt-get install make. Para instalar la dependencia make.
- Clonar el repositorio de GitHub del algoritmo:
 - git clone <https://github.com/pq-crystals/kyber.git>
 - Descargar el archivo .gz y descomprimirlo en el directorio /usr/local/src.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Instalación OpenSSL:

- Actualizar los paquetes: sudo apt-get update.
- Instalación del paquete libssl-dev: sudo apt-get install libssl-dev.
- Dirigirse al escritorio /usr/local/src.
- Descarga del archivo .gz desde la web: wget

<https://www.openssl.org/source/openssl-3.0.2o.tar.gz>.

- Extraer el archivo: tar -xf openssl-3.0.2v.tar.gz.
- Dirigirse al directorio de openssl: cd openssl-3.0.2v.
- Comprobar si se ha instalado viendo la versión: openssl versión -a.

- Declarar las variables de entorno:

- export CFLAGS="-I/usr/local/opt/openssl.3.0.2/include".
- export NISTFLAGS="-I/usr/local/opt/openssl.3.0.2/include".
- export LDFLAGS="-L/usr/local/opt/openssl.3.0.2/lib".
- Configuración y compilación de OpenSSL:

- ./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared zlib.
- make install: Se instala OpenSSL en el directorio /usr/local/ssl.

- Configuración de bibliotecas compartidas:

- Acceder al archivo de configuración que se ha creado el cuál cuelga del directorio /usr/local/ssl/lib/etc.
- Se accede al archivo cd /etc/ld.so.conf.d/.
- Cargar datos del archivo a un archivo binario: vim opens-3.0.2v.conf.
- Se nos abrirá un panel donde tenemos que colocar la ruta de la biblioteca OpenSSL: /usr/local/ssl/lib. Procedemos a guardar.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Cargamos el enlace de configuración y veremos las bibliotecas cargadas:

Sudo ldconfig -v.

- Configuración archivos binarios:

- Realizamos una copia de seguridad:

- mv /usr/bin/c_rehash /usr/bin/c_rehash.BEKUP.

- mv /usr/bin/openssl /usr/bin/openssl.BEKUP.

- Edición del archivo enviroment que se encuentra en el directorio etc y a añadir el nuevo PATH que queremos tener.

- vim /etc/enviroment.

-PATH=“

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:

/usr/games:/usr/local/games:/usr/local/ssl/bin”

- Compilar el directorio ref:

- cd ref. Para dirigirnos al escritorio ref.

- make. Una vez que nos encontramos en el directorio para compilar los programas.

- make speed. Compila los programas benchmarking.

- Compilar el directorio avx2:

- cd avx2. Para dirigirnos al escritorio ref.

- make. Una vez que nos encontramos en el directorio para compilar los programas.

- make speed. Compila los programas benchmarking.

- Directorio del algoritmo Crystals-Kyber: Se accede mediante el comando ls

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

dentro del directorio kyber para mostrar su contenido. Podemos visualizar en la captura diferentes archivos y directorios.

```
admin@ubuntu:~/kyber$ ls
AUTHORS          Kyber1024-90s_META.yml  Kyber768-90s_META.yml  ref
avx2            Kyber1024_META.yml      Kyber768_META.yml      runlcov.sh
CMakeLists.txt   Kyber512-90s_META.yml  LICENSE           runtests.sh
Common_META.yml  Kyber512_META.yml     README.md         SHA256SUMS
```

Figura 5.1 Directorio algoritmo del Kyber .

Podemos observar el siguiente contenido:

- Authors: Archivo que indica los autores del algoritmo.
- Avx2: Directorio donde están las instrucciones del algoritmo en la versión

avx2 la cuál fue desarrollada por Intel para procesadores x86.

```
admin@ubuntu:~/kyber/avx2$ ls
aes256ctr.c  fips202x4.c  ntt.h          rng.c          test_kyber1024
aes256ctr.h  fips202x4.h  ntt.s          rng.h          test_kyber512
align.h       fq.inc      params.h        sha2.h         test_kyber768
api.h         fq.S       poly.c          shuffle.inc   test_kyber.c
basemul.s    indcpa.c   poly.h          shuffle.S     test_speed.c
cbd.c         indcpa.h   polyvec.c      speed_print.c test_vectors1024
cbd.h         invntt.S  polyvec.h      speed_print.h test_vectors512
consts.c     keccak4x   PQGenKAT_kem.c symmetric.h   test_vectors768
consts.h     kem.c      randombytes.c  symmetric-shake.c test_vectors.c
cpucycles.c  kem.h      randombytes.h  test_kex1024   verify.c
cpucycles.h  kex.c      reduce.h       test_kex512   verify.h
fips202.c    kex.h      rejsample.c   test_kex768
fips202.h    Makefile   rejsample.h   test_kex.c
```

Figura 5.2 Figura del directorio avx2 del algoritmo Kyber.

- Aes256ctr.c: Archivo el cuál es una implementación del algoritmo AES-256 en modo contador o también llamado CTR, bajo una representación en avx2 optimizada para procesadores x86. Tiene funciones para encriptar, para generar PRF (secuencia de bytes pseudoaleatorios), para inicializar un contexto de cifrado y para generar una secuencia de bloques cifrados en CTR.

- Aes256ctr.h: Encabezado del archivo aes256ctr.c.
- Align.h: Encabezado usado para alinear datos. Esto ayuda a la mejora del rendimiento.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Basemul.S: Archivo en ensamblador para operaciones de multiplicación.
- Api.h: Define constantes para luego ser usadas en el algoritmo KYBER.
- Cbd.c: Archivo para generación de polinomios.
- Cbd.h: Encabezado del archivo Cbd.c.
- Consts.c: Archivo que define constantes usadas posteriormente en el algoritmo Kyber.
- Consts.h: Encabezado del archivo consts.h.
- Cpucycles.c: Llamada en comando de Linux al directorio ref que se encuentra colgando del padre de avx2 a un mismo nivel.
- Flips202.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Flips202.h: Archivo de encabezado.
- Flips202x4.c: Archivo cuya función es la implementación de SHAKE128 y SHAKE 256 para procesar varios bloques de datos de manera simultánea.
- Flips202x4.h: Archivo de encabezado.
- Fq.inc: Archivo que tiene la función de usarse en multiplicaciones y otras operaciones en registros de 256 bits en avx2.
- Fq.S: Archivo en ensamblador para operaciones de campo finito.
- Indcpa.c: Archivo que se encarga de verificar la protección de la comunicación.
- Indcpa.h: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Invntt.S: Archivo en ensamblador que implementa la transformación numérica inversa teórica.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Kem.c: Archivo que contiene una implementación KEM del algoritmo Kyber.
- Kem.h: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Kex.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Kex.h: Archivo de encabezado.
- Makefile: Archivo de configuración para la compilación en Linux. Con escribir make en la terminal en el mismo directorio bastaría para compilar el programa.
- Ntt.h: Archivo de encabezado de la transformación numérica teórica.
- Ntt.S: Archivo en ensamblador dedicado a la transformación numérica teórica.
- Params.h: Archivo de encabezado para definir contantes del algoritmo kyber.
- Poly.c: Archivo cuya función es realizar operaciones polinómicas.
- Poly.h: Archivo de encabezado para operaciones polinómicas.
- Polyvec.c: Archivo cuya función es realizar operaciones polinómicas.
- Polyvec.h: Archivo de encabezado para operaciones polinómicas.
- PQCgenKAT_kem.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Randombytes.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Randombytes.h: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Reduce.h: Archivo de encabezado el cual contiene operaciones para la transformación a Montgomery.
- Rejsample.c: Archivo en el que se define una matriz de dos dimensiones la cuál es llamada idx.
- Rejsample.h: Archivo de encabezado para el algoritmo kyber.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Rng.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Rng.h: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Sha2.h: Archivo de encabezado para facilitar la simplificación del hash sha-256 y sha-512.
- Shuffle.inc: Archivo usado para cambiar de orden los elementos de matrices etc.
- Shuffle.S: Archivo en lenguaje ensamblador, su función es realizar operaciones shuffle en registros de 256 bits.
- Speed_print.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
 - Speed_print.h: Llamada en comando de Linux al mismo archivo en el directorio ref.
 - Symmetric.h: Archivo de encabezado para versiones de kyber-90s.
 - Symmetric-shake.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Test_kex.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
 - Test_kex512: Test para ver si el sistema de intercambio de claves funciona en UAKE y AKE. Usa una longitud de clave de 512 bits.
 - Test_kex768: Test para ver si el sistema de intercambio de claves funciona en UAKE y AKE. Usa una longitud de clave de 768 bits.
 - Test_kex1024: Test para ver si el sistema de intercambio de claves funciona en UAKE y AKE. Usa una longitud de clave de 1024 bits.
- Test_kyber.c: Llamada en comando de Linux al mismo archivo en el directorio ref.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Test_kyber512: Test para verificar la seguridad del algoritmo kyber en diferentes situaciones. Usa una longitud de clave de 512 bits.
- Test_kyber768: Test para verificar la seguridad del algoritmo kyber en diferentes situaciones. Usa una longitud de clave de 768 bits.
- Test_kyber1024: Test para verificar la seguridad del algoritmo kyber en diferentes situaciones. Usa una longitud de clave de 1024 bits.
- Test_speed.c: Archivo en el que se realizan diferentes pruebas 1000 veces con el fin de medir el rendimiento mediante el tiempo de ejecución en ciclos de CPU.
- Test_vectors.c: Llamada en comando de Linux al mismo archivo en el directorio ref.
- Test_vectors512: Archivo para comprobar si se crean las claves, y si hay un cifrado y descifrado de forma correcta. Usa una longitud de clave de 512 bits.
- Test_vectors768: Archivo para comprobar si se crean las claves, y si hay un cifrado y descifrado de forma correcta. Usa una longitud de clave de 768 bits.
- Test_vectors1024: Archivo para comprobar si se crean las claves, y si hay un cifrado y descifrado de forma correcta. Usa una longitud de clave de 1024 bits.
- Verify.c: Archivo cuya función principal es que no haya fugas de información en el paso del mensaje.
- Verify.h: Archivo de encabezado para aliviar las posibles fugas de información.

- Ref: Directorio donde están las instrucciones del algoritmo en la versión de referencia.

```
admin@ubuntu:~/kyber/ref$ ls
aes256ctr.c      ntt.h          symmetric-aes.c    test_speed1024-90s
aes256ctr.h      params.h       symmetric.h        test_speed512
api.h             poly.c         symmetric-shake.c test_speed512-90s
cbd.c             poly.h         test_kex1024       test_speed768
cbd.h             polyvec.c     test_kex1024-90s  test_speed768-90s
CMakeLists.txt    polyvec.h     test_kex512        test_speed.c
cpucycles.c      PQCgenKAT_kem.c test_kex512-90s  test_vectors1024
cpucycles.h      randombytes.c  test_kex768        test_vectors1024-90s
fips202.c         randombytes.h  test_kex768-90s  test_vectors512
fips202.h         reduce.c       test_kex.c        test_vectors512-90s
indcpa.c          reduce.h      test_kyber1024    test_vectors768
indcpa.h          rng.c         test_kyber1024-90s test_vectors768-90s
kem.c              rng.h         test_kyber512    verify.c
kem.h              sha256.c      test_kyber512-90s verify.h
kex.c              sha2.h        test_kyber768    verify.h
kex.h              sha512.c      test_kyber768-90s
Makefile           speed_print.c  test_kyber.c      test_speed.c
htt.c              speed_print.h test_speed1024
```

Figura 5.3 Figura del directorio ref del algoritmo Kyber.

Este directorio cuenta con los siguientes archivos:

- Aes256ctr.c: Consiste en una variante del algoritmo AES en CTR(Counter), permite trabajar en paralelo en bloques de 256 bits.
- Aes256ctr.h: Archivo de encabezado del algoritmo aes.
- Api.h: Archivo que consta de implementaciones del algoritmo Kyber en c.
- Cbd.c: Archivo que contiene implementaciones del algoritmo sobre generación de claves y cifrado.
- Cbd.h: Archivo de encabezado del módulo cbd.
- CMakeLists.txt: Archivo de texto donde se encuentra la configuración que queramos hacer para hacer diferentes pruebas en el algoritmo, diferentes versiones del algoritmo según la longitud de la clave etc.
- Cpucycles.c: Archivo que tiene la función de calcular la sobrecarga mínima, esto ayuda a obtener medidas más concretas.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Cpuycles.h: Archivo de encabezado que ayuda a la hora de medir los ciclos de la CPU.
- Flips202.c: Archivo que se encarga de generar un hash SHA-3 de longitud de clave 256 bits mediante la función Keccak.
- Flips202.h: Archivo de encabezado que permite usar algoritmos como SHA-3 de 256 bits y de 512 bits, SHAKE128 y SHAKE256.
- Indcpa.c: Mismo significado que la función en el directorio avx2.
- Indcpa.h: Mismo significado que la función en el directorio avx2.
- Kem.c: Archivo que implementa un modelo de clave pública para proporcionar seguridad en el algoritmo Kyber.
- Kem.h: Archivo de encabezado utilizado para usar un modelo de clave pública.
- Kex.c: Archivo cuya función primordial es generar claves compartidas entre el emisor y el receptor.
- Kex.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Kex.c.
- Makefile: Mismo significado que la función en el directorio avx2.
- Ntt.c: Archivo en el que se realiza operaciones NTT en el algoritmo kyber.
- Ntt.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Ntt.c.
- Params.h: Mismo significado que la función en el directorio avx2.
- Poly.c: Mismo significado que la función en el directorio avx2.
- Poly.h: Mismo significado que la función en el directorio avx2.
- Polyvec.c: Mismo significado que la función en el directorio avx2.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Polyvec.h: Mismo significado que la función en el directorio avx2.
- PQCgenKAT_kem.c: Archivo que tiene la función de testear el algoritmo kyber, utiliza diferentes casos para comprobar si funciona correctamente.
- Randombytes.c: Archivo cuya función es generar bytes aleatorios.
- Randombytes.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Randombytes.c.
- Reduce.c: Archivo de verificación para tratar sobre el resultado de operaciones modulares este en el rango correcto.
- Reduce.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Reduce.c.
- Rng.c: Archivo cuya función consiste en generar números aleatorios y números pseudoaleatorios que estén basado en AES-256 en modo CTR.
- Rng.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Rng.c.
- Sha256.c: Archivo que implementa SHA-256 del cuál se obtiene un hash de los parámetros de entrada.
- Sha2.h: Mismo significado que la función en el directorio avx2.
- Sha512.c: Archivo que implementa SHA-512 del cuál se obtiene un hash de los parámetros de entrada.
- Speed_print.c: Archivo en el que se obtienen parámetros estadísticos como la mediana, promedio, media etc.
- Speed_print.h: Archivo de encabezado que define las constantes y funciones usadas en el archivo Speed_print.c.
- Symmetric-aes.c: Archivo encargado de realizar operaciones de cifrado AES-

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

256 en CTR.

- Symmetric.h: Mismo significado que la función en el directorio avx2.
- Symmetric-shake.c: Archivo que permite el uso de SHAKE-128 y SHAKE-256.
 - Test_kex.c: Archivo que trata de verificar si hay un intercambio UAKE y AKE correctamente.
 - Test_kex512,test_kex512-90s,test_kex768,test_kex768-90s,test_kex1024,test_kex1024-90s: Mismo significado que la función en el directorio avx2.
 - Test_kyber.c: Archivo para verificar la seguridad del algoritmo kyber en diferentes situaciones.
 - Test_kyber512,test_kyber512-90s,test_kyber768,test_kyber768-90s,test_kyber1024, test_kyber1024-90s: Mismo significado que la función en el directorio avx2.
 - Test_speed.c: Mismo significado que la función en el directorio avx2.
 - Test_speed512: Test de velocidad con un nivel de seguridad similar a AES-128.
 - Test_speed512-90s: Test de velocidad con un nivel de seguridad similar a AES-256 y SHA2 en vez de SHAKE.
 - Test_speed768: Test de velocidad con un nivel de seguridad similar a AES-192.
 - Test_speed768-90s: Test de velocidad con un nivel de seguridad similar a AES-256 y SHA2 en vez de SHAKE.
 - Test_speed1024: Test de velocidad con un nivel de seguridad similar a AES-256.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- `Test_speed1024-90s`: Test de velocidad con un nivel de seguridad similar a AES-256 y SHA2 en vez de SHAKE.
- `Test_vectors.c`: Archivo que se encarga de comprobar si el sistema KEM funciona como tiene que ser y que la clave que ha sido compartida sea la misma por parte del emisor y receptor.
- `Test_vectors512`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-128.
- `Test_vectors512-90s`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-256.
- `Test_vectors768`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-192.
- `Test_vectors768-90s`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-256.
- `Test_vectors1024`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-256.
- `Test_vectors1024-90s`: Test de comprobación de uso correcto de kem con un nivel de seguridad similar a AES-256.
- `Verify.c`: Mismo significado que la función en el directorio avx2.
- `Verify.h`: Mismo significado que la función en el directorio avx2.
- `CMakeLists.txt`: Archivo de configuración para generar scripts y archivos para algunos sistemas operativos distintos.
- `Archivos yml`: Son metadatos los cuáles sirven para la configuración y descripción de una instancia del algoritmo. Sirve como guía al programador o usuario que va a utilizar en el algoritmo donde aparecen diferentes campos como: el nombre

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

del algoritmo, el nivel de seguridad del NIST, el tipo de seguridad que requerimos(el usado es IND-CAA2), el tamaño de la clave pública, del texto cifrado, de la clave secreta y la clave secreta que ha sido compartido, tipo de algoritmo, un valor del hash sha256 usado para la integridad de los vectores que se generan en las pruebas de vectores y otro hash que se encarga de comprobar el resultado del algoritmo, información sobre los integrantes que han colaborado y participado en la creación del algoritmo, una sección llamada implementación generada en json en la cual da información del algoritmo.

Tenemos un ejemplo diferente a los otros tipos de archivos yml, el cuál es Common_META el cuál muestra las bibliotecas comunes del proyecto con una serie de especificaciones.

- Archivos sh: Runcov.sh consiste en un script cuya función es identificar la parte de código ejecutada en pruebas y generar el informe correspondiente. Runtests.sh tiene la función de realizar pruebas de forma automática.
- README: Archivo de texto que explica el funcionamiento y requisitos para trabajar con este algoritmo.
- SHA256SUMS: Consiste en un archivo donde se va a encontrar los valores de suma de verificación también llamado checksum en el inglés, se muestra los checksum correspondientes a los archivos tvecs512, tvecs512-90s, tvecs768, tvecs768-90s, tvecs1024, tvecs1024-90s. Esta información nos ayuda a comprobar si ha habido archivos que durante la transmisión o guardado han sido dañados o modificados.

5.2 NTRU.

- Directorio del algoritmo NTRU: Se accede mediante el comando ls dentro del directorio ntru para mostrar su contenido. Podemos visualizar en la captura diferentes

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

archivos y directorios.

```
admin@ubuntu:~$ ls
Descargas Escritorio kyber Música Plantillas snap
Documentos Imágenes mcelliece ntru Público Vídeos
admin@ubuntu:~$ cd ntru
admin@ubuntu:~/ntru$ ls
avx2-common knownanswertest ref-hps2048509 ref-hrss701
avx2-hps2048509 LICENSE ref-hps2048677 test
avx2-hps2048677 package_supercop.sh ref-hps40961229 test_compatibility.sh
avx2-hps4096821 README.md ref-hps4096821
avx2-hrss701 ref-common ref-hrss1373
```

Figura 5.4 Figura del directorio del algoritmo NTRU.

Contiene los siguientes directorios, archivo script de shell, un archivo de licencia, archivos de guía como README:

- Avx2-common: Directorio el cual contiene cuatro archivos. Directorio del algoritmo ntru con la instrucción avx2 para procesadores Intel x86, en la cual se espera una mejora de la velocidad de carga. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:~/ntru/avx2-common$ ls
crypto_sort_int32.c crypto_sort_int32.h poly.h sample_iid.c
```

Figura 5.5 Figura del directorio avx2-common del algoritmo NTRU.

- Avx2-hps2048509: Versión optimizada del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-128, esto indica que admite una clave de 128 bits. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:~/ntru/avx2-hps4096821$ ls
api_bytes.h crypto_int8.h owcpa.c poly_r2_inv.h
api.h crypto_sort_int32.c owcpa.h randombytes.c
asngen crypto_sort_int32.h pack3.c randombytes.h
bitpermutations fips202.c packq.c sample.c
cmov.c fips202.h params.h sample.h
cmov.h kem.c poly.c sample_iid.c
cpucycles.c kem.h poly.h test
cpucycles.h Makefile poly_lift.c
crypto_hash_sha3256.h Makefile-NIST poly_r2_inv.c
```

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Figura 5.6 Figura del directorio avx2-hps4096821 del algoritmo NTRU.

- Avx2-hps2048677: Versión optimizada del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-192, esto indica que admite una clave de 192 bits. Otorga más seguridad que el anterior. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:/ntru/avx2-hps2048677$ ls
api_bytes.h bitpermutations cpucycles.c      crypto_int8.h    fips202.c    ken.h        owcpa.c    packq.c    poly.h        poly_mod_q_phi_n.s    poly_r2_mul.s    poly_s3_inv.c    sample.c
square_10_677_shufbytes.s square_21_677_shufbytes.s square_3_677_patience.s square_84_677_shufbytes.s
api.h          cpucycles.h      crypto_sort_int32.c    fips202.h   Makefile     owcpa.h    params.h    poly_lift.c    poly_r2_inv.c    poly_rq_mul.s    randombytes.c    sample.h
square_1_677_patience.s square_2_677_patience.s square_42_677_shufbytes.s test
isngen         cmov.h        crypto_hash_sha3256.h  crypto_sort_int32.h  ken.c        Makefile-NIST pack3.c    poly.c        poly_mod_3_phi_n.s    poly_r2_inv.h    poly_rq_to_s3.s    randombytes.h    sample_id.c
square_168_677_shufbytes.s square_338_677_shufbytes.s square_5_677_patience.s vec32_sample_id.s
```

Figura 5.7 Figura del directorio avx2-hps2048677 del algoritmo NTRU.

- Avx2-hps4096821: Versión optimizada del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-256, esto indica que admite una clave de 256 bits. Otorga más seguridad que los anteriores.

Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:/ntru/avx2-hps4096821$ ls
api_bytes.h  cpucycles.c    fips202.c    owcpa.c    poly.h        poly_r2_mul.s    sample.c        square_204_821_shufbytes.s    test
api.h        cpucycles.h    fips202.h   owcpa.h    poly_lift.c    poly_rq_mul.s    sample.h        square_24_821_shufbytes.s    vec32_sample_id.s
isngen       crypto_hash_sha3256.h  kem.c        pack3.c    poly_mod_3_phi_n.s    poly_rq_to_s3.s    sample_id.c    square_3_821_patience.s
bitpermutations crypto_int8.h  ken.h        packq.c    poly_mod_q_phi_n.s    poly_s3_inv.c    square_102_821_shufbytes.s    square_408_821_shufbytes.s
cmov.c       crypto_sort_int32.c  Makefile    params.h    poly_r2_inv.c    randombytes.c    square_12_821_shufbytes.s    square_51_821_shufbytes.s
cmov.h       crypto_sort_int32.h  Makefile-NIST poly.c        poly_r2_inv.h    randombytes.h    square_1_821_patience.s    square_6_821_patience.s
```

Figura 5.8 Figura del directorio avx2-hps4096821 del algoritmo NTRU.

- Avx2-hrss701: Versión optimizada del algoritmo ntru cuya matriz tiene una longitud de 701 unidades. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:/ntru/avx2-hrss701$ ls
api_bytes.h    cnov.h      fips202.h   owcpa.c   poly.c      poly_r2_inv.c  poly_s3_inv.c  sample_id.c    square_27_701_shufbytes.s  square_84_701_shufbytes.s
api.h          cpucycles.c  ken.c      owcpa.h   poly.h      poly_r2_inv.h  randombytes.c  square_12_701_shufbytes.s  square_336_701_shufbytes.s  test
isngen        cpucycles.h  ken.h      pack3.c   poly_lift.s  poly_r2_mul.s  randombytes.h  square_15_701_shufbytes.s  square_3_701_patience.s  vec32_sample_id.s
bitpermutations crypto_hash_sha3256.h  Makefile  packq.c   poly_mod_3_Phi_n.s  poly_rq_mul.s  sample.c     square_168_701_shufbytes.s  square_42_701_shufbytes.s
cmov.c        fips202.c    Makefile-NIST  params.h  poly_mod_q_Phi_n.s  poly_rq_to_s3.s  sample.h     square_1_701_patience.s  square_6_701_patience.s
```

Figura 5.9 Figura del directorio avx2-hrss701 del algoritmo NTRU.

- Knownanswertest: Directorio para verificar la seguridad del algoritmo mediante una técnica llamada known answer test. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:~/ntru/knownanswertest$ ls
avx2  Makefile  PQGenKAT_kem.c  ref  rng.c  rng.h
admin@ubuntu:~/ntru/knownanswertest$ cd avx2
admin@ubuntu:~/ntru/knownanswertest/avx2$ ls
PQKemKAT_1234.req  PQKemKAT_1234.rsp  PQKemKAT_1450.req  PQKemKAT_1450.rsp  PQKemKAT_1590.req  PQKemKAT_1590.rsp  PQKemKAT_935.req  PQKemKAT_935.rsp
admin@ubuntu:~/ntru/knownanswertest$ cd ..
admin@ubuntu:~/ntru/knownanswertest$ cd ref
admin@ubuntu:~/ntru/knownanswertest/ref$ ls
PQKemKAT_1234.req  PQKemKAT_1450.req  PQKemKAT_1590.req  PQKemKAT_2366.req  PQKemKAT_2983.req  PQKemKAT_935.req
PQKemKAT_1234.rsp  PQKemKAT_1450.rsp  PQKemKAT_1590.rsp  PQKemKAT_2366.rsp  PQKemKAT_2983.rsp  PQKemKAT_935.rsp
```

Figura 5.10 Figura del directorio Knownanswertest del algoritmo NTRU.

- License: Archivo en el que viene los términos y condiciones del algoritmo ntru.
- Package_supercop.sh: Archivo en script que realiza tareas de empaquetado de diferentes versiones de NTRU y obtiene metadatos.
- Readme.md: Archivo de guía para configuración e implementación del algoritmo.
- Ref-common: Directorio que contiene diferentes archivos comunes dentro de la implementación de referencia. Este directorio contiene los siguientes archivos mostrados en la siguiente imagen.

```
admin@ubuntu:~/ntru/ref-common$ ls
cmov.c  cpucycles.h  crypto_sort_int32.h  kem.c  owcpa.h  pack3.c  poly.c      poly_mod.c      poly_rq_mul_schoolbook.c  randombytes.c  sample.h
cmov.h  crypto_hash_sha3256.h  fips202.c      kem.h  pack16384.c  pack4096.c  poly.h      poly_r2_inv.c  poly_rq_mul_toom4_k2x2.c  randombytes.h  sample_id.c
cpucycles.c  crypto_sort_int32.c  fips202.h      owcpa.c  pack2048.c  pack8192.c  poly_lift.c  poly_rq_mul_k2x4.c  poly_s3_inv.c  sample.c
```

Figura 5.11 Figura del directorio ref-common del algoritmo NTRU.

- Ref-hps2048509: Versión de referencia del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-128, esto indica que admite una clave de 128 bits. Este directorio

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

contiene los siguientes archivos mostrados en la siguiente imagen, comparten los mismos archivos con los tres siguientes directorios.

```
admin@ubuntu:/ntru/ref-hps2048509$ ls
api_bytes.h    cmov.h      crypto_hash_sha3256.h  fips202.c  kem.h        owcpa.c  packq.c  poly.h      poly_r2_inv.c  PQCgenKAT_kem sample.c  test
api.h          cmov.c      cpucycles.c      crypto_sort_int32.c  fips202.h  Makefile   owcpa.h  params.h  poly_lift.c  poly_rq_mul.c  randombytes.c  sample.h
cmov.c         cpucycles.h cpucycles.h      crypto_sort_int32.h   kem.c     Makefile-NIST pack3.c  poly.c      poly_mod.c   poly_s3_inv.c  randombytes.h  sample_id.c
```

Figura 5.12 Figura del directorio ref-hps2048509 del algoritmo NTRU.

- Ref-hps2048677: Versión de referencia del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-192, esto indica que admite una clave de 192 bits. Otorga más seguridad que el anterior.

-Ref-hps4096821: Versión de referencia del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-256, esto indica que admite una clave de 256 bits. El espacio de clave es 2^{256} .

-Ref-hps40961229: Versión de referencia del algoritmo ntru cuya implementación contiene unos parámetros que se asemejan a la seguridad ofrecida por el cifrado AES-256, esto indica que admite una clave de 256 bits. El espacio de clave es 2^{256} .

- Ref-hrss1373: Versión de referencia del algoritmo ntru, el tamaño tanto de la clave pública como privada es de 1373 bits. El nivel de seguridad está en torno a 128 bits.

Este directorio contiene los siguientes archivos mostrados en la siguiente imagen, comparten los mismos archivos con el siguiente directorio.

```
admin@ubuntu:/ntru/ref-hrss1373$ ls
api_bytes.h    cmov.h      crypto_hash_sha3256.h  fips202.h  kem.h        Makefile-NIST  owcpa.h  packq.c  poly.c      poly_r2_inv.c  poly_s3_inv.c  randombytes.c  sample.c  sample_id.c
api.h          cmov.c      cpucycles.c      crypto_sort_int32.c  fips202.h  kem.c     Makefile   owcpa.c  pack3.c  params.h  poly_lift.c  poly_rq_mul.c  PQCgenKAT_kem  randombytes.h  sample.h  test
```

Figura 5.14 Figura del directorio ref-hrss1373 del algoritmo NTRU.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Ref-hrss701: Versión de referencia del algoritmo ntru, el tamaño tanto de la clave pública como privada es de 701 bits. El nivel de seguridad está en torno a 128 bits.

- Test: Directorio en el cual se encuentran los diferentes testeos que se pueden realizar al compilar cada directorio del algoritmo.

```
admin@ubuntu:~/ntru/test$ ls
decap.c  encap.c  gen_owcpa_vecs.c  keypair.c  speed.c  test_gp_compat.c  test_ntru.c  test_owcpa.c  test_pack.c  test_polymul.c
```

Figura 5.15 Figura del directorio test del algoritmo NTRU.

5.3 McEliece.

- Directorio del algoritmo McEliece: Se accede mediante el comando ls dentro del directorio ntru para mostrar su contenido. Podemos visualizar en la captura diferentes archivos y directorios.

```
admin@ubuntu:~/mceliece-20221023$ ls
Additional_Implementations  Optimized_Implementation  README  Reference_Implementation  Supporting_Documentation  XKCP
```

Figura 5.16 Figura del directorio del algoritmo McEliece.

- Additional_Implementations: Directorio en el que se encuentra las implementaciones adicionales del algoritmo. El contenido del directorio se repite en los dos siguientes directorios, es el siguiente:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/mceliece-20221023/Additional_Implementations$ ls
...
admin@ubuntu:~/mceliece-20221023/Additional_Implementations$ cd ken
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken$ ls
mceliece348864 mceliece40896 mceliece6688128f mceliece6960119 mceliece6960119f mceliece8192128 mceliece8192128f
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken$ cd mceliece348864
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ ls
...
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ cd avx
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864/avx$ ls
api.h const.data decrypt.c fft_tr.h kat_kem_int operations.h scalars_2x.data transpose_64x256_sp.asm.q update_asm.S vec256_mul_asm.q vec_mul_sp_asm.S
architectures const.S decrypt.h gf.c kat_kem.rsp pk_gen.c sk_gen.h transpose_64x256_sp.asm.S util.h vec256_mul_asm.S vec_reduce_asm.q
benes.c controlbits.c encrypt.c gf.h kat_kem.rsp pk_gen.c sk_gen.c transpose_64x64.asm.q vec128.h vec.c vec_reduce_asm.S
benes.h controlbits.h encrypt.h int32_minmax_x86.c KATNUM pk_gen.h sk_gen.h transpose_64x64.asm.S vec128_mul_asm.q vec.h
bm.c crypto_hash.h fft.c int32_sort.c Makefile powers.data subroutines transpose.h vec128_mul_asm.S vec_mul_asm.q
bm.h crypto_kem.h fft.h int32_sort.h KATNUM randombytes.h syndrome.asm.q uint64_sort.h vec.c vec_mul_asm.S
build crypto_kem_mceliece348864.h fft_tr.c kat_kem_int operations.c run syndromes.asm.q update_asm.q vec256.h vec_mul_sp_asm.q
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ cd ..
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ ls
...
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ sse
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864/sse$ ls
...
api.h const.data decrypt.c fft_tr.h kat_kem.rsp pk_gen.c sk_gen.c transpose_64x64.asm.q util.h vec_mul_asm.q
architectures const.S decrypt.h gf.c KATNUM pk_gen.h sk_gen.h transpose_64x64.asm.S vec128.c vec_mul_asm.S
benes.c controlbits.c encrypt.c gf.h Makefile powers.data subroutines transpose.h vec128.h vec_reduce_asm.q
benes.h controlbits.h encrypt.h int32_sort.h randombytes.h syndrome.asm.q uint64_sort.h vec128_mul_asm.q vec_reduce_asm.S
bm.c crypto_hash.h fft.c kat_kem_int operations.c run syndromes.asm.q uint64_sort.h vec128_mul_asm.S vec_reduce_asm.S
bm.h crypto_kem.h fft.h int32_sort.h kat_kem_req params.h scalars.data transpose_64x128_sp.asm.q update_asm.q vec.c
build crypto_kem_mceliece348864.h fft_tr.c kat_kem_req params.h scalars.data transpose_64x128_sp.asm.S update_asm.S vec.h
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ sse$ cd ..
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864$ cd vec
admin@ubuntu:~/mceliece-20221023/Additional_Implementations/ken/mceliece348864/vec$ ls
...
api.h bm.h controlbits.h decrypt.c encrypt.h fft_tr.h kat KATNUM operations.h powers.data scalars.data transpose.h vec.c
benes.c build crypto_hash.h decrypt.h fft.c gf.c kat_kem_int Makefile params.h randombytes.h sk_gen.c uint16_sort.h vec.h
benes.h const.data crypto_kem.h destino.txt fft.h gf.h kat_kem_req fft.h pk_gen.c run sk_gen.h uint64_sort.h
bm.c controlbits.c crypto_kem_mceliece348864.h encrypt.c fft_tr.c int32_sort.h kat_kem_rsp operations.c pk_gen.h scalars_2x.data subroutines util.h
```

Figura 5.17 Figura del directorio Additional_Implementations del algoritmo McEliece.

Para poder compilar cualquiera de estos directorios tenemos que usar los siguientes comandos:

```
export CPATH="$CPATH:$HOME/include".
```

```
export LIBRARY_PATH="$LIBRARY_PATH:$HOME/lib".
```

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HOME/lib".
```

Se puede apreciar en la fotografía tres directorios dentro de cada configuración del algoritmo que indica las tres implementaciones adicionales.

- Optimized_Implementation: Directorio en el que se encuentra las implementaciones optimizadas avx2 del algoritmo.

- Reference_Implementation: Directorio en el que se encuentra las implementaciones de referencia del algoritmo.

- Supporting_Documentation: Archivos de texto cuya función consiste en facilitar ayuda o soporte a los usuarios que deseen utilizar el algoritmo como prueba. El contenido del directorio es el siguiente:

```
admin@ubuntu:~/mceliece-20221023$ cd Supporting_Documentation
admin@ubuntu:~/mceliece-20221023/Supporting_Documentation$ ls
impl.pdf mods2.pdf mods3.pdf mods.pdf pc.pdf rationale.pdf security.pdf spec.pdf submission.pdf
```

Figura 5.18 Figura del directorio Supporting_Documentation del algoritmo McEliece.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

-XCP: Librería necesaria a parte de Openssl instalación explicada en el punto 5.3. El contenido del directorio es el siguiente:

```
admin@ubuntu:~/mceliece-20221023$ cd XKCP
admin@ubuntu:~/mceliece-20221023/XKCP$ ls
bin doc lib LICENSE Makefile Makefile.build README.markdown Standalone support tests util
```

Figura 5.19 Figura del directorio XKCP del algoritmo McEliece.

5.4 Pruebas.

Se va a testear los diferentes archivos de prueba que hay en cada algoritmo. Para ello nos vamos a ayudar de la terminal de Linux, en la cuál con las configuraciones que se han realizado en pasos anteriores. El primero en ser testeado es el algoritmo kyber.

Kyber: Accedemos al perfil sudoers para no tener ningún problema de permisos, introducimos nuestra contraseña, al estar ya en el usuario admin, se procede a desplazarse al directorio de inicio donde se tiene descomprimido e instalado el algoritmo kyber.

```
khalid@ubuntu:/home/admin$ su admin
Contraseña:
admin@ubuntu:~$ cd ~
admin@ubuntu:~$ ls
Descargas Escritorio include lib mceliece-20221023.tar.gz Música Plantillas snap XKCP
Documentos Imágenes kyber mceliece-20221023 mceliece ntrn Público Videos
admin@ubuntu:~$ cd kyber
admin@ubuntu:~/kyber$ ls
AUTHORS CMakelists.txt Kyber1024-90s_META.yml Kyber512-90s_META.yml Kyber768-90s_META.yml LICENSE ref runtests.sh
avx2 Common_META.yml Kyber1024_META.yml Kyber512_META.yml Kyber768_META.yml README.md runlcov.sh SHA256SUMS
```

Figura 5.20 Acceso a kyber.

El siguiente paso es elegir si queremos testear la implementación en referencia o avx2.

Accedemos al directorio referencia.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/kyber$ cd ref
admin@ubuntu:~/kyber/ref$ ls
aes256ctr.c    fips202.h   ntt.h        reduce.c      symmetric-aes.c  test_kex.c      test_speed1024-90s  test_vectors512-90s
aes256ctr.h    indcpa.c    params.h    reduce.h      symmetric.h     test_kyber1024  test_speed512       test_vectors768
api.h          indcpa.h    poly.c      rng.c       symmetric-shake.c test_kyber1024-90s  test_speed512-90s  test_vectors768-90s
tbd.c          kem.c      poly.h      rng.h       test_kex1024    test_kyber512  test_speed768      verify.c
tbd.h          kem.h      polyvec.c  sha256.c    test_kex1024-90s  test_kyber512-90s  test_speed768-90s  verify.h
MakeLists.txt   kex.c      PQGenKAT_kem.c sha512.c    test_kex512    test_kyber768  test_speed.c     test_vectors1024
tpucycles.c   kem.h      randombytes.c speed_print.c test_kex768    test_kyber.c   test_vectors1024-90s
tpucycles.h   Makefile   randombytes.h speed_print.h test_kex768-90s  test_speed1024  test_vectors512
tips202.c     ntt.c      randombytes.h speed_print.h test_kex768-90s  test_speed1024  test_vectors512
```

Figura 5.21 Acceso a kyber/ref.

La información de todos los textos está en un archivo comprimido en la subida de la memoria por eso se pondrá de ejemplo una prueba de cada tipo.

Para ello se va a seleccionar kyber 512 y posteriormente su salida por pantalla.

- Test_Kex512: Se verifica si hay intercambio de claves con UAKE y AKE, la

longitud de la clave es 512 bits. La salida es la siguiente:

```
admin@ubuntu:~/kyber/ref$ ./test_kex512
KEX_UAKE_SENDABYTES: 1568
KEX_UAKE_SENDBBYTES: 768
KEX_AKE_SENDABYTES: 1568
KEX_AKE_SENDBBYTES: 1536
```

Figura 5.22 Ejemplo de salida de datos de Test Kex 512.

KEX_UAKE_SENDABYTES: El tamaño de los datos de UAKE enviados de Alice a Bob en el proceso de intercambio a.

KEX_UAKE_SENDBBYTES: El tamaño de los datos enviados de UAKE enviados de Bob a Alice en el proceso de intercambio b.

KEX_AKE_SENDABYTES: El tamaño de los datos de AKE enviados de Alice a Bob en el proceso de intercambio a.

KEX_AKE_SENDBBYTES: El tamaño de los datos enviados de AKE enviados de Bob a Alice en el proceso de intercambio b.

Se observa solo diferencia en el tamaño de datos de UAKE en el intercambio b y el tamaño de datos de UAKE en el intercambio b, el tamaño de datos de UAKE en el

intercambio a es igual al tamaño de datos de AKE en el intercambio a.

- Test_Kyber512: Se verifica si la seguridad del algoritmo kyber cuya longitud de clave es 512 bits.

```
admin@ubuntu:~/kyber/ref$ ./test_kyber512
CRYPTO_SECRETKEYBYTES: 1632
CRYPTO_PUBLICKEYBYTES: 800
CRYPTO_CIPHERTEXTBYTES: 768
```

Figura 5.23 Ejemplo de salida de datos de Test Kyber 512.

CRYPTO_SECRETKEYBYTES: Valor de la clave secreta en bytes.

CRYPTO_SECRETKEYBYTES: Valor de la clave secreta en bytes

CRYPTO_SECRETKEYBYTES: Valor de la clave secreta en bytes

Gracias a estas variables se pueden obtener el tamaño de diferentes parámetros como claves etc.

- Test_Speed512: En esta función se mide la mediana y el promedio del ciclo en CPU de diferentes funciones. Esto mide la velocidad que tarda en procesarse cada función.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/kyber/rei$ ./test
gen_a:
median: 18479 cycles/ticks
average: 18906 cycles/ticks

poly_getnoise_eta1:
median: 3569 cycles/ticks
average: 3693 cycles/ticks

poly_getnoise_eta2:
median: 1973 cycles/ticks
average: 2007 cycles/ticks

NTT:
median: 8735 cycles/ticks
average: 8854 cycles/ticks

INVNTT:
median: 13355 cycles/ticks
average: 13599 cycles/ticks

polyvec_basemul_acc_montgomery:
median: 8315 cycles/ticks
average: 8403 cycles/ticks

poly_tomsg:
median: 965 cycles/ticks
average: 1017 cycles/ticks

poly_frommsg:
median: 293 cycles/ticks
average: 275 cycles/ticks

poly_compress:
median: 629 cycles/ticks
average: 641 cycles/ticks

poly_decompress:
median: 83 cycles/ticks
average: 84 cycles/ticks

polyvec_compress:
median: 2771 cycles/ticks
average: 2800 cycles/ticks

polyvec_decompress:
median: 1805 cycles/ticks
average: 1821 cycles/ticks

lndcpa_keypair:
median: 102773 cycles/ticks
average: 105413 cycles/ticks

lndcpa_enc:
median: 124319 cycles/ticks
average: 128281 cycles/ticks

lndcpa_dec:
median: 45317 cycles/ticks
average: 45743 cycles/ticks

kyber_keypair:
median: 111509 cycles/ticks
average: 113780 cycles/ticks

kyber_encaps:
median: 144899 cycles/ticks
average: 147987 cycles/ticks

kyber_decaps:
median: 180809 cycles/ticks
average: 185245 cycles/ticks

kex_uake_initA:
median: 256661 cycles/ticks
average: 263498 cycles/ticks

kex_uake_sharedB:
median: 326927 cycles/ticks
average: 334870 cycles/ticks

kex_uake_sharedA:
median: 181691 cycles/ticks
average: 185570 cycles/ticks

kex_ake_initA:
median: 256325 cycles/ticks
average: 259010 cycles/ticks

kex_ake_sharedB:
median: 472373 cycles/ticks
average: 486050 cycles/ticks

kex_ake_sharedA:
median: 362837 cycles/ticks
average: 370637 cycles/ticks
```

Figura 5.24 Ejemplo de salida de datos de Test Speed 512.

Test_Vectors512: Prueba para ver cómo se generan las claves, encapsulamiento y des encapsulamiento en hexadecimal. Y como último la clave que comparte Alice y Bob. _No se puede mostrar todo el proceso porque se excede en caracteres, pero se puede mostrar las claves compartidas de Alice y Bob, el texto cifrado y la clave secreta y pública.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
d0f3a323dfff69be0c1a3f39b59ea8cf255514c6cdf78f7fe5d7919e966ceac6b  
8646ee272f1428d0a13a3f39b59ea8cf255514c6cdf78f7fe5d7919e966ceac6b  
Public Key: 49f679f1a4390e917ac13090e9d3137cc1b363ab90e4381e9bd0a25b0abee4e3c963b66b9dc66fbc495b022a238fe0885ff4bb0972c9cd197e7f2d1bae48a5a4bb98834b220429abb0a39314626  
25d27d53e2ff8c8c964a85af0c591836c981be08d190190504aeecc97a5602a366bfce132e39c3e3895681e4eef3873bbdb4877559b9d4d751f1d420f7499a68676  
7fd80877feebd01f5b9ee15b973028b7d72d112d685ba426614ff925b9a2187c47463e3144a95958bf684733c3122b980a14864d36f827ac153ab52ad0d1cf50b364a7e465b29a722e976f6c0836a1ghab73c37730332  
728bea0c8826225c1a330368923632829eac62344eb0d748f2328daebba60a0fb813677979a9e4785135b01f15b60177e70f320817e64a40f749c54b60387798751857f633a4c59828338cc9eccb30899a6835  
Secret Key: acfc24cb016f1ed6a518b5b0d620a5db4a4f5c6167e7aa8055882a088458ad426897b6c163ac66d351c834b2c76f76195d79b4753a079889128ea7c92492e0e6925aeaf46675c3736cc61fc82b8c0e9e  
aa0e51a9d2dfab4eb922537e76f1a6426a14732fa65a502f5bf09e55d0d1b150d1c440e6222525d40e893827957bc3073b4c87a5bb5282f1585c2b56779260cbb22a3dbbbd10893d1172a417ccfbfb6191d49698  
1a885795c92dd9f77a6e0e669f99788c367a3bf2c6e55d0184e0754d4d18c48cabe97470f8091c4013214e0607f3ce50044708721a9c00aa3971549946f1b87f9ade22cha0108c8ccdb3c94d79abeb3bc095955ch7d4a  
c722d9b5c913e0887e28999a0873671d12c347aae0ba99c3b434edc76380e279417fa99f91a32542459647772fae415ddc912178015fe77c006496e0e2137d9c19eba1ba2a7e00e0c484919a3575bf2b67d1cf7ccdd74cd  
f2d1bae49ia5ab9884b4h228429abb0a391146265b7145ce14ab89217ba8a3c26ef834a4e629re0d08784bf8c5587120b11691eaabdbac12f79cbceca9a9c0669a6af58a04a4d1b140e226871051ce254bfc5a3h09  
7abbd4b4877559b59d0d751f1d142c8f7499a86876fa116c2b87919c866ddbf917d14076a6e99879830cb1ad3b599118aaaba42b1a5fa0e64a7e5e59000c77302d494f7285ad8ba85dd29a73979c3d7b87a1ba14e465baa5  
a2e465bc9a722e976fc0836a1ba673c37320332bf9ec45a2a04c556992339abae43823c1b097b5c4a2e8208d763738ae360440e0c774934385d5332154531488b7a567be8892f21ba04f4d1c492f00413e91b731  
87541852f1633a296926071b626136f0f1seeds;ffed3ed0857f4668226742eb752ebebf86  
c571a3a296926071b626136f0f1seeds;ffed3ed0857f4668226742eb752ebebf86  
CipherText: 71cd765fccc794442492b1be9f9286f99800c9d7e627785bc07feefadbe4d6343c49fecc9e81bb02a20174ef9c835de845d979cc89d9436797189be405c10ae14844febe7efbc38141a273a6bce35  
683e0ec1a71c1f08e6358255899eae2f29c62b8465911e067f13fb13c52462a0e84c1502191d136f8a0e1cod5c2de0e6fd0c796e96899c4b54d4e0f6e6af519c26928ee38e728666eb81c17ae02911584de1902b5b9  
0c5477fd79b94a2884783b3d4f16df8d3d76a81e566f7989d2c540009c83217296e4437a6ea021b3f56bfac04681f5dc5adb42345129f6cd492117b7ae040672383e76c8b472e9750d19a9b0f425e54d67f81e1fb1  
7c13fd4500e4a8b585ea901b522119b7594bf4783bf032a28bb997b5bd57c9812a5cdcbc19fb818591f35d9ba95837e6bb0a9e0d56058bcc767ebeb0af92c70edbcce24ad8c5fe04bc2e59839c02fc1c6b0ab76ce88f  
Shared Secret B: 3e95155286193494deaa4883fdb19f54cc8ee305de9e7c971247ee09164041c  
Shared Secret A: 3e95155286193494deaa4883fdb19f54cc8ee305de9e7c971247ee09164041c
```

Figura 5.25 Ejemplo de salida de datos de Test Vectors 512.

Accedemos al directorio avx2.

```
admin@ubuntu:~/kyber$ ls  
AUTHORS  avx2  CMakelists.txt  Common META.yml  Kyber1024-90s META.yml  Kyber1024 META.yml  Kyber512-90s META.yml  Kyber512 META.yml  Kyber768-90s META.yml  Kyber768 META.yml  LICENSE  README  
admin@ubuntu:~/kyber$ cd avx2  
admin@ubuntu:~/kyber/avx2$ ls  
aes256ctr.c  api.h  cbd.h  cpucycles.c  fips202.h  fq.inc  lndcpa.h  kem.c  kex.h  ntt.S  poly.h  PQCgenKAT_kem.c  reduce.h  rng.h  shuffle.inc  speed_print.h  
aes256ctr.h  basemul.S  consts.h  cpucycles.h  fips202x4.c  fq.S  invntt.S  kem.h  Makefile  params.h  polyvec.c  randombytes.c  rejsample.h  rng.h  shuffle.S  symmetric.h  
align.h  cbd.c  consts.h  fips202.c  fips202x4.h  lndcpa.c  keccak4x.c  kex.c  ntt.h  poly.c  polyvec.h  randombytes.h  rejsample.h  sha2.h  speed_print.c  symmetric-shake.c  
admin@ubuntu:~/kyber/avx2$
```

Figura 5.26 Acceso a directorio avx2.

Las funciones que nos interesan son las siguientes:

```
test_kex1024  test_kex512-90s  test_kex.c      test_kyber512  test_kyber768-90s  test_speed1024-90s  test_speed768  test_vectors1024  test_vectors512-90s  test_vectors.c  
test_kex1024-90s  test_kex768  test_kyber1024  test_kyber512-90s  test_kyber.c  test_speed512  test_speed768-90s  test_vectors1024-90s  test_vectors768  verify.c  
test_kex512  test_kex768-90s  test_kyber1024-90s  test_kyber768  test_speed1024  test_speed512-90s  test_speed.c  test_vectors512  test_vectors768-90s  verify.h
```

Figura 5.27 Funciones importantes del directorio avx2.

La información de todos los textos está en un archivo comprimido en la subida

de la memoria por eso se pondrá de ejemplo una prueba de cada tipo.

Para ello se va a seleccionar kyber 512 y posteriormente su salida por pantalla.

- Test_Kex512: Misma función que en la versión de referencia.

```
admin@ubuntu:~/kyber/avx2$ ./test_kex512  
KEX_UAKE_SENDABYTES: 1568  
KEX_UAKE_SENDBBYTES: 768  
KEX_AKE_SENDABYTES: 1568  
KEX_AKE_SENDBBYTES: 1536
```

Figura 5.28 Ejemplo de salida de datos de Test Kex 512 en avx2.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Test_Kyber512: Misma función que en la versión de referencia.

```
admin@ubuntu:~/kyber/avx2$ ./test_kyber512
CRYPTO_SECRETKEYBYTES: 1632
CRYPTO_PUBLICKEYBYTES: 800
CRYPTO_CIPHERTEXTBYTES: 768
```

Figura 5.29 Ejemplo de salida de datos de Test Kyber 512 en avx2.

- Test_Vectors512: Misma función que en la versión de referencia.

```
734c158122b8c84cf57e845da8adc6eadc368b329b55108c01d0b1790ea807d1
3cd0887ca97373879e8a540836e197fe7b0a19513dc234ec8add3df7a9ca
Public Key: 1abb4c85569ce09a92b008080f47e56bd7915eb478641852fd2483c4923ee95ac5c019a080fc6465f90f9c9b8c0ad800c3a67aaeb1ccb7b83a7430bddfc8a7e08612829a75de9
86c2e0815d05699326625db2c52e02896722778944860b5a37a213a7287f4876652291e5cb37f78641852fd2483c4923ee95ac5c019a080fc6465f90f9c9b8c0ad800c3a67aaeb1ccb7b83a7430bddfc8a7e08612829a75de9
0111b4c4682b96909106d13030b1ac44823545a384fb29919da87a16451c75a19d0ba3d3693222c0d3c20a44d9b704440c5439a2b32e6147667504f7a7d437f621b058d1a53d94456e56d22ca3da627b9653d
91e85963b791fd72c5ed6c18249b7f82898ee08c46bbbe0b79367290483879898c83b155e601b98996c5fb358ab79db3b599f8abbe07bcc2514b406374c40519291e88ddc11abf1d8127f22fb26b007b7e17767fbeb1874ee374a05a11e53
971868c74945a7bed99b7949317497e44988a3568acf0a76124c45f83d4233c05a3b9259543d04e2f35a9e52ab36e4a16c40538947832847885750a5902f1cb2e452c818c9fe804ff1fb00f01c3002342949daa28e2f93e8a638f8ccb1
30eet7cbsde231b57883f12784adff1b5a1c3b6bc89
Secret Key: 90652e29a537f78cb24994ba0550ba04aabb6811c383074365b39ff5108177842b0d0286273a4b126f3ae9adaef679fd1c1a2ad0b345864860ba0ca3ed5162eadcd264e0b344cc59abf4b23db4c67298a80c7f562d5516767cccd93a
33a2baa35b364ad477062f77842f3e9187b842d4f8a46175157b5clb2e0f9e777864bbd632bb396e5733bf04d841daea92eecd1bb9d0815ab54c3de2733cb5a31d400239e4a3086388a2fb3368a0b60c366bcc0eac70d7f926c347b53906784e908f
48ff12387b2b0623531383f197989c371c03a5b336832e0301ed63a74ec857f1a2df16496c8a162876130357d5f9c577c4c111d1a288539a9eaedadc9c4b0781c9ed774582c6930139fc4c483346ae0e2134abe814954c3481320e7a101
82c4ad0f0236fb710dc424b657c35ab3810c4cbf181b2216b91c3fb1017b7a178820c47a15f24b12bz29a20e179a8955742a880364aad6c11ced033389ce256fa2a371059d1ea93386e2818c211b747962849rebd8a813d25956b2916a29
c75343015b478918c0134cd1a0743c183158c9c113b7a4872cafa547d311efb0b456c5a45f83d4233c05a3b9259543d04e2f35a9e52ab36e4a16c40538947832847085750a5902f1cb2e452c818c9fe804ff1fb00f01c3002342949daa28e2f93e8a638f8ccb19284c1e6cc0716e32a5671cd36a9145668
0080f47e56b07915cb470c78641852fd2f483c4923ee95ac5c019a00f699d5ac5f83d4236e6f3960c9781d661a088fc4665f90f9c9b8c0ad800c3a67aaeb1ccb7b83a7430bddfc8a7e08621829475de699c4b79ba0f0c99a0ef393febe43023508b4c
96722778944860b65a37a21b784f8766522912e13f213a777600128f9c5809444a3f317986c527bb0c999bf1c158d15a267b32f7544456e56d822ca3d627b9653d428456d5e0a19816b29e0f4b682d2f7
3545a304fb8b9d91c9da87a16451c75a19d0ba3d3693222c0d3c20a44d91b9704440c5439a2b32e6147667504f7a7d7a437f621a058d15b3a9d1b9023948b27dd8adfcdf93407eac16d5a019d4c19c1289d45c0e118c5b9934ce
898ee80c46bbbe0b79367290483879898c83b155e601b98996c5fb358ab79db3b599f8abbe07bcc2514b406374c40519291e88d0d1a51b7d18127f22fb26b007b7e17767fbeb1874ee374a05a11e05328ea424c6c94fc58423ccc49e2b4b86
44d988a93568acf0a76124c45f83d4233c05a3b9259543d04e2f35a9e52ab36e4a16c40538947832847085750a5902f1cb2e452c818c9fe804ff1fb00f01c3002342949daa28e2f93e8a638f8ccb19284c1e6cc0716e32a5671cd36a9145668
c3b0b8c90df6d30f612a29dc3a5f14386c8f87ec6ba0bc59f64ba563bc7c285c58e9df5a3cd0873ca9c7737799e8a540836e197f7b0aa195d13d0c234ec8addd3df7a9ca
3989d9a1e491f38e79d4ff73368339960a051ca9a333cd0ca894c7579f
ciphertext: 173235eae3dc9b5c056122d8f13685bd9547c29a15b98a6a6573c3942a518b44b7c60182be98d1b2a65c4219ad07e1a2bccf99797a34d1d50e0df9597cac9ff8fa7aa2595a8cfa160989c8673fb01cheae2600f7467cd7
d5180570e6260dc3a6b64974c0b26c79978e70f684649901969d7f1985cbddc758b1a6632d060108a46675f90eaa0d8bd94dc5a5ceeb0947ad493826830ed41df8e6859967f8e081bdead1344a499837486440447c05825d7f10b510113a0350c1c
a43dad0b0686293f9b449fcfa2ec4e8dbf5f7fcb7f7a933381d5f83315b84e1b916df7b13fe2ea2c5a2bdaac4f251ac7b125087d291904e38e0f8e7d92c1d408c060bf8e6968717166f645034277f04797b4a
351231e963b3d6eb2c48c044392c778ea109f29c55648c17946542537f7d1dc1a1bbad28868802a1e9d579f8ee51ac7967bae13b7dfa9c2a33759f6ff4369be3d5d0286f448c9b5137c39d30af0e7802f9cb207086753c5451664
e7f9fc13e2a461b184533c8b8d0746596170d13c42159b52d536f57bdb2e2adff018eae2d88eaaf17e0e29709dcaaf9c3e02e017f5dbfb8818c7dfeb5cfe7138ad7f6461bcffbb572632d11e57e089fb2a6f424360b8d0b15a74b84364
Shared Secret B: da485505ed5afabc0f987caf1542de931440e2c57e3b8213e309ff0d75eb1519b
Shared Secret A: da485505ed5afabc0f987caf1542de931440e2c57e3b8213e309ff0d75eb1519b
[...]
```

Figura 5.30 Ejemplo de salida de datos de Test Vectors 512 en avx2.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Test_Speed512:_Misma función que en la versión de referencia.

```
admin@ubuntu:~/kyber/avx2$ ./test_speed512
gen_a:
median: 8231 cycles/ticks
average: 10141 cycles/ticks

poly_getnoise_eta1:
median: 2435 cycles/ticks
average: 2513 cycles/ticks

poly_getnoise_eta2:
median: 1595 cycles/ticks
average: 1590 cycles/ticks

poly_getnoise_eta1_4x:
median: 4787 cycles/ticks
average: 4860 cycles/ticks

NTT:
median: 209 cycles/ticks
average: 193 cycles/ticks

INVNTT:
median: 209 cycles/ticks
average: 204 cycles/ticks

polyvec_basemul_acc_montgomery:
median: 167 cycles/ticks
average: 195 cycles/ticks

poly_tomsg:
median: 41 cycles/ticks
average: 38 cycles/ticks

poly_frommsg:
median: 41 cycles/ticks
average: 38 cycles/ticks

poly_compress:
median: 41 cycles/ticks
average: 36 cycles/ticks

poly_decompress:
median: 41 cycles/ticks
average: 40 cycles/ticks

polyvec_compress:
median: 209 cycles/ticks
average: 198 cycles/ticks

polyvec_decompress:
median: 83 cycles/ticks
average: 99 cycles/ticks

indcpa_keypair:
median: 16169 cycles/ticks
average: 16497 cycles/ticks

indcpa_enc:
median: 16589 cycles/ticks
average: 17361 cycles/ticks

indcpa_dec:
median: 1007 cycles/ticks
average: 1000 cycles/ticks

kyber_keypair:
median: 23099 cycles/ticks
average: 24734 cycles/ticks

kyber_encaps:
median: 34019 cycles/ticks
average: 35037 cycles/ticks

kyber_decaps:
median: 38555 cycles/ticks
average: 40342 cycles/ticks

kex_uake_initA:
median: 79925 cycles/ticks
average: 74384 cycles/ticks

kex_uake_sharedB:
median: 62831 cycles/ticks
average: 77647 cycles/ticks

kex_uake_sharedA:
median: 39941 cycles/ticks
average: 37776 cycles/ticks

kex_ake_initA:
median: 56951 cycles/ticks
average: 62468 cycles/ticks

kex_ake_sharedB:
median: 96935 cycles/ticks
average: 114570 cycles/ticks

kex_ake_sharedA:
median: 80765 cycles/ticks
average: 85737 cycles/ticks
```

Figura 5.31 Ejemplo de salida de datos de Test Speed 512 en avx2.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Ntru: Accedemos al perfil sudoers para no tener ningún problema de permisos, introducimos nuestra contraseña, al estar ya en el usuario admin, se procede a desplazarse al directorio de inicio donde se tiene descomprimido e instalado el algoritmo ntru.

```
khalid@ubuntu: ~$ su admin  
Contraseña:  
admin@ubuntu:~/home/khalid$ cd ~  
admin@ubuntu:~$ ls  
Descargas Documentos Escritorio Imágenes include kyber lib mceliece-20221023 mceliece-20221023.tar.gz mcelliece Música ntru  
Plantillas Pública snap Videos XKCP  
admin@ubuntu:~$ cd ntru  
admin@ubuntu:~/ntru$ ls  
avx2-common avx2-hps2048509 avx2-hps2048677 avx2-hps4096821 avx2-hrss701 knownanswertest LICENSE package_supercop.sh README.md  
ref-common ref-hps2048509 ref-hps2048677 ref-hps40961229 ref-hps4096821 ref-hrss1373 ref-hrss701 test test_compatibility.sh  
admin@ubuntu:~/ntru$
```

Figura 5.32 Acceso al directorio ntru.

Respecto a kyber, en ntru no hay directorio referencia ni avx2, si no que se divide en directorios de cada versión algoritmo con implementación ya sea red y avx2. Avx2-common y Ref-common son las funciones comunes de los algoritmos. Voy a tomar de ejemplo uno de avx2 y otro de referencia.

De referencia voy a seleccionar ref-hps2048509 y de avx2 avx2-hps2048509 para comparar la misma versión de algoritmo con diferente implementación.

Accedemos al directorio ref-hps2048509, y a su vez dentro del directorio test:

```
admin@ubuntu:~/ntru$ cd ref-hps2048509  
admin@ubuntu:~/ntru/ref-hps2048509$ ls  
api_bytes.h cpcycles.h fips202.h owcpa.c poly.c poly_rq_mul.c sample.c  
api.h crypto_hash_sha3256.h kem.c owcpa.h poly.h poly_s3_inv.c sample.h  
cmov.c crypto_sort_int32.c kem.h pack3.c poly_lift.c PQCgenKAT_kem sample_iid.c  
cmov.h crypto_sort_int32.h Makefile packq.c poly_mod.c randombytes.c test  
cpucycles.c fips202.c Makefile-NIST params.h poly_r2_inv.c randombytes.h  
admin@ubuntu:~/ntru/ref-hps2048509$ cd test  
admin@ubuntu:~/ntru/ref-hps2048509/test$ ls  
decap encaps gen_owcpa_vecs.c keypair.c speed.c test_gp_compat.c test_ntru.c test_owcpa.c test_pack.c  
decap.c encaps.c keypair speed test_gp_compat test_ntru test_owcpa test_pack test_polymul.c  
admin@ubuntu:~/ntru/ref-hps2048509/test$
```

Figura 5.33 Contenido de ref-hps2048509/test.

Se va a realizar el test_gp_compat, test_ntru, test_pack y test_owcpa:

- Test_gp_compat: Prueba que verifica si las operaciones de polinomios en ntru son compatibles con PARI/GP.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/ntru/ref-hps2048509/test$ ./test_owcpa
success
```

Figura 5.37 Ejemplo de salida de datos de Test owcpa.

Acedemos al directorio avx2-hps2048509:

Se va a realizar el test_gp_compat, test_ntru, test_pack y test_owcpa:

- Test_gp_compat: Misma función que la implementación de referencia.

```
admin@ubuntu:~/ntru/avx2-hps2048509/test$ ./test_gp_compat
ALLOK=1;
plift(x) = lift(lift(x));
clift(x) = if(type(x) == "t_POLMOD", -centerlift(-lift(x)), -centerlift(-x));
q = (1 << 11);
n = 509;
Phi = polcyclo(n);
in = Mod(1,3) * Polrev([0,0,1,1,2,1,2,0,2,1,2,2,0,0,0,0,1,2,1,2,0,0,1,1,1,1,0,0,2,2,0,1,0,1,2,2
,2,1,0,2,0,0,2,0,2,1,2,2,0,0,0,2,0,2,1,0,1,0,1,2,0,1,0,1,1,1,2,1,1,0,1,0,2,2,2,0,0,0,1,2,
1,1,0,0,0,2,0,0,2,1,0,2,0,1,0,0,0,1,0,2,1,1,2,1,2,1,1,0,2,2,2,1,0,1,2,1,2,2,1,0,1,2,1,0,2,0,1
,2,2,1,1,2,1,1,0,2,0,0,2,2,2,2,0,1,0,2,2,1,2,0,0,0,2,2,2,0,2,1,0,0,0,1,2,2,2,2,0,0,0,1,0,0,2,1,1,
0,0,1,0,1,2,0,0,0,2,2,2,1,1,1,1,2,2,0,1,1,0,1,1,1,0,1,2,2,1,1,0,1,0,1,1,2,2,2,0,0,1,0,0,0,0,1,1
,2,0,0,0,1,1,0,1,1,2,2,0,0,1,1,0,1,0,2,2,0,2,1,2,1,1,0,2,0,1,1,2,2,2,1,1,0,2,2,0,1,1,2,2,2,1,2,
1,1,1,0,1,1,1,1,1,0,2,0,1,2,1,1,1,2,2,2,1,1,0,1,0,0,1,1,1,2,2,2,0,1,1,0,0,1,1,2,2,2,1,0,1,1,0
,1,1,0,1,0,2,2,0,1,1,1,2,2,2,0,2,0,1,1,1,2,1,2,0,2,1,1,1,0,1,2,1,1,1,0,1,0,0,2,0,2,1,0,0,0,1,2,2,1,0,0
,1,0,0,1,0,2047,1,2047,1,2047,1,2047,2047,0,0,0,0,1,2047,1,2047,0,0,1,1,1,1,0,0,2047
2047,1,2047,0,0,1,1,1,1,0,1,1,2047,0,0,2047,2047,1,0,2047,0,0,0,2047,0,2047,0,2047,1,2047,1,2047,2047,0,0
,1,1,2047,1,1,0,1,0,2047,2047,2047,0,0,0,1,2047,2047,0,2047,1,0,0,1,1,1,1,1,2047,2047,1,1,1,1,0
,1,0,0,0,1,0,2047,1,1,2047,1,2047,1,1,0,2047,2047,1,0,1,2047,1,2047,1,0,1,2047,2047,1,0,1,2047,1,0,2047
```

Figura 5.38 Ejemplo de salida de datos de Test Gp Compat.

- Al estar ALLOK a 1 quiere decir que ha funcionado perfectamente la prueba.

- Test_ntru: Misma función que la implementación de referencia.

```
admin@ubuntu:~/ntru/avx2-hps2048509/test$ ./test_ntru
success
```

Figura 5.39 Ejemplo de salida de datos de Test ntru.

El resultado es success por lo tanto ha sido un éxito.

- Test_pack: Misma función que la implementación de referencia.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/ntru/avx2-hps2048509/test$ ./test_pack  
success
```

Figura 5.40 Ejemplo de salida de datos de Test pack.

El resultado es success por lo tanto ha sido un éxito.

- Test_owcpa: Misma función que la implementación de referencia.

```
admin@ubuntu:~/ntru/avx2-hps2048509/test$ ./test_owcpa  
success
```

Figura 5.41 Ejemplo de salida de datos de Test owcpa.

El resultado es success por lo tanto ha sido un éxito.

McEliece: Accedemos al perfil sudoers para no tener ningún problema de permisos, introducimos nuestra contraseña, al estar ya en el usuario admin, se procede a desplazarse al directorio de inicio donde se tiene descomprimido e instalado el algoritmo McEliece.

```
khalid@ubuntu:/home/admin$ su admin  
Contraseña:  
admin@ubuntu:~$ ls  
Descargas Escritorio include lib mceliece-20221023.tar.gz Música Plantillas snap XKCP  
Documentos Imágenes kyber mceliece-20221023 mceliece ntru Público Vídeos  
admin@ubuntu:~$ cd mceliece-20221023  
admin@ubuntu:~/mceliece-20221023$ ls  
Additional_Implementations Optimized_Implementation README Reference_Implementation Supporting_Documentation XKCP  
admin@ubuntu:~/mceliece-20221023$ █
```

Figura 5.42 Acceso al directorio mceliece.

Se procederá a explicar una versión de cada directorio que hay. Una de referencia, una optimizada y una adicional. Para empezar, se accederá al directorio Reference_Implementation.

- Reference_Implementation: Acceder directorio de referencia y después al directorio intermedio llamado kem cuyo contenido es el siguiente:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/mceliece-20221023$ cd Reference_Implementation
admin@ubuntu:~/mceliece-20221023/Reference_Implementation$ ls
kem
admin@ubuntu:~/mceliece-20221023/Reference_Implementation$ cd kem
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem$ ls
mceliece348864  mceliece460896  mceliece6688128  mceliece6960119  mceliece8192128
mceliece348864f  mceliece460896f  mceliece6688128f  mceliece6960119f  mceliece8192128f
```

Figura 5.43 Acceso al directorio mceliece/Reference_Implementation.

Escogemos mceliece348864: Compilamos y arreglamos el error surgido de biblioteca exportando las etiquetas a una dirección para que reconozca el encabezado.

```
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ make
./run
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ ./run
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ ./build
In file included from operations.c:5:
crypto_hash.h:3:10: fatal error: libkeccak.a.headers/SimpleFIPS202.h: No existe el archivo o el directorio
  3 | #include <libkeccak.a.headers/SimpleFIPS202.h>
   | ^_____________________________________
compilation terminated.
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ export CPATH="$CPATH:$HOME/include"
export LIBRARY_PATH="$LIBRARY_PATH:$HOME/lib"
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HOME/lib"
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ ./build
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$
```

Figura 5.44 Arreglo de encabezado libkeccak no encontrado.

Mostramos el contenido del directorio:

```
admin@ubuntu:~/mceliece-20221023/Reference_Implementation/kem/mceliece348864$ ls
api.h  controlbits.c      decrypt.h    kat      nist      randombytes.h subroutines util.c
benes.c controlbits.h     encrypt.c    kat_kem.int operations.c root.c    synd.c      util.h
benes.h crypto_hash.h    encrypt.h    kat_kem.req operations.h root.h    synd.h
bm.c   crypto_kem.h      gf.c        kat_kem.rsp  params.h   run      transpose.c
bm.h   crypto_kem_mceliece348864.h gf.h        KATNUM    pk_gen.c  sk_gen.c  transpose.h
build  decrypt.c         int32_sort.h Makefile   pk_gen.h  sk_gen.h  uint64_sort.h
```

Figura 5.45 Contenido de directorio mceliece/Reference_Implementation/kem/mceliece348864.

Tenemos que abrir los archivos generados kat_kem.int, kat_kem.req y kat_kem.int:

- Kat_kem.int: Archivo que contiene parámetros ya definidos para ser usados en diferentes pruebas.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:/mccllece-2021023/Reference_Implementation/ken/mceliece348864$ cat kat_kem.int
encrypt e: positions 0 81 92 105 216 378 389 468 504 586 617 738 928 1030 1061 1070 1072 1177 1294 1316 1366 1389 1510 1670 1721 1728 1746 1783 1888 1893 1
1995 2003 2053 2068 2082 2113 2202 2363 2418 2508 2553 2603 2641 2704 2731 2951 2974 3091 3132 3158 3174 3182 3215 3330 3348 3377 3382 3392
decrypt e: positions 0 81 92 105 216 378 389 468 504 586 617 738 928 1030 1061 1070 1072 1177 1294 1316 1366 1389 1510 1670 1721 1728 1746 1783 1888 1893 1
1995 2003 2053 2068 2082 2113 2202 2363 2418 2508 2553 2603 2641 2704 2731 2816 2951 2974 3091 3132 3157 3158 3174 3182 3215 3330 3348 3377 3382 3392
encrypt e: positions 75 117 159 212 254 277 297 307 466 667 745 790 828 846 871 873 878 885 946 977 1067 1106 1268 1327 1415 1541 1557 1596 1641 1666 1711
1844 1863 1868 1893 1902 1947 1994 2024 2061 2216 2252 2464 2544 2565 2570 2751 2860 2864 2865 2885 2941 2984 2986 3087 3100 3239 3252 3278 3292 3397 3416
decrypt e: positions 75 117 159 212 254 277 297 307 466 667 745 790 828 846 871 873 878 885 946 977 1067 1106 1268 1327 1415 1541 1557 1596 1641 1666 1711
1844 1863 1868 1893 1902 1947 1994 2024 2061 2216 2252 2464 2544 2565 2570 2751 2860 2864 2865 2885 2941 2984 2986 3087 3100 3239 3252 3278 3292 3397 3416
encrypt e: positions 31 34 290 351 363 486 505 537 567 655 681 683 705 794 831 843 861 869 928 1115 1151 1163 1350 1367 1489 1490 1511 1517 1757 1901 1915
1939 1986 2032 2117 2136 2167 2189 2195 2246 2300 2353 2374 2387 2405 2408 2430 2434 2597 2705 2734 2763 3102 3104 3107 3193 3199 3207 3337 3400 3428
decrypt e: positions 31 34 290 351 363 486 505 537 567 655 681 683 705 794 831 843 861 869 928 1115 1151 1163 1350 1367 1489 1490 1511 1517 1757 1901 1915
1939 1986 2032 2117 2136 2167 2189 2195 2246 2300 2353 2374 2387 2405 2408 2430 2434 2597 2705 2734 2763 2785 3102 3104 3107 3193 3199 3207 3337 3400 3428
encrypt e: positions 1 81 239 261 308 350 385 496 530 550 658 709 716 939 1021 1021 1027 1137 1192 1245 1294 1376 1502 1600 1655 1658 1846 1871 1897 1912 1986 2
2107 2180 2254 2263 2397 2407 2427 2433 2447 2466 2466 2533 2542 2563 2682 2718 2763 2897 2906 2963 2988 3110 3144 3172 3178 3218 3286 3362 3365 3381 3389
decrypt e: positions 1 81 239 261 308 350 385 496 530 550 658 709 716 939 1021 1027 1137 1192 1245 1294 1376 1502 1600 1655 1658 1846 1871 1897 1912 1986 2
2107 2180 2254 2263 2397 2407 2427 2433 2447 2466 2466 2533 2542 2563 2682 2718 2763 2897 2906 2963 2988 3110 3144 3172 3178 3218 3286 3362 3365 3381 3389
encrypt e: positions 62 71 73 75 134 190 262 361 441 510 552 723 783 809 1039 1080 1084 1152 1172 1283 1288 1372 1388 1500 1523 1589 1672 1717 1728 1731 17
894 1949 2020 2093 2147 2159 2206 2305 2400 2427 2433 2457 2478 2497 2536 2558 2569 2632 2675 2703 2708 2843 2847 2957 3024 3025 3153 3221 3265 3317 3406 3
decrypt e: positions 62 71 73 75 134 190 262 361 441 510 552 723 783 809 1039 1080 1084 1152 1172 1283 1288 1372 1388 1500 1523 1589 1672 1717 1728 1731 17
894 1949 2020 2093 2147 2159 2206 2305 2400 2427 2433 2457 2478 2497 2536 2558 2569 2632 2675 2703 2708 2843 2847 2957 3024 3025 3153 3221 3265 3317 3406 3
encrypt e: positions 97 236 253 386 416 444 488 556 568 626 731 784 834 911 952 1063 1130 1231 1234 1235 1242 1341 1417 1419 1432 1463 1468 1537 1829 1846
1883 2044 2165 2316 2352 2398 2400 2434 2447 2527 2617 2653 2687 2750 2761 2825 2902 2912 2938 2980 2982 3111 3114 3143 3148 3188 3339 3355 3411 3413 3422
decrypt e: positions 97 236 253 386 416 444 488 556 568 626 731 784 834 911 952 1063 1130 1231 1234 1235 1242 1341 1417 1419 1432 1463 1468 1537 1829 1846
1883 2044 2165 2316 2352 2398 2400 2434 2447 2527 2617 2653 2687 2750 2761 2825 2902 2912 2938 2980 2982 3111 3114 3143 3148 3188 3339 3355 3411 3413 3422
encrypt e: positions 21 69 141 160 198 218 257 294 296 346 368 389 463 474 568 645 728 845 853 1047 1058 1085 1166 1291 1311 1312 1342 1382 1486 1510 1520 1567
1659 1682 1726 1751 1796 1881 1935 1961 1966 2004 2047 2082 2155 2355 2368 2435 2455 2494 2598 2635 2654 2700 2765 2794 2827 2864 2878 2898 3003 3394 3401
decrypt e: positions 21 69 141 160 198 218 257 294 296 346 368 389 463 474 568 645 728 845 853 1047 1058 1085 1166 1291 1311 1312 1342 1382 1486 1510 1520 1567
1659 1682 1726 1751 1796 1881 1935 1961 1966 2004 2047 2082 2155 2355 2368 2435 2455 2494 2598 2635 2654 2700 2765 2794 2827 2864 2878 2898 3003 3394 3401
encrypt e: positions 65 150 239 332 377 496 498 521 575 708 754 775 943 1043 1046 1060 1075 1174 1221 1262 1372 1380 1553 1571 1596 1632 1666 1686 168
13 1748 1987 2028 2096 2205 2216 2275 2291 2296 2303 2322 2324 2357 2422 2640 2644 2713 2740 2770 2796 2968 2996 3071 3249 3283 3320 3351 3383 3398 34
decrypt e: positions 65 150 239 332 377 496 498 521 575 708 754 775 943 1043 1046 1060 1075 1174 1221 1262 1372 1380 1553 1571 1596 1632 1666 1686 168
13 1748 1987 2028 2096 2205 2216 2275 2291 2296 2303 2322 2324 2357 2422 2640 2644 2713 2740 2770 2796 2968 2996 3071 3249 3283 3320 3351 3383 3398 34
encrypt e: positions 157 332 355 432 471 519 532 565 683 772 790 808 812 943 959 994 1053 1054 1074 1117 1172 1263 1290 1433 1477 1511 1627 1637 1758 1781
2086 2115 2128 2194 2224 2306 2319 2364 2398 2415 2425 2505 2693 2638 2727 2742 2806 2847 2916 2925 2965 3005 3044 3048 3054 3207 3264 3289 3348 3365 3401
decrypt e: positions 157 332 355 432 471 519 532 565 683 772 790 808 812 943 959 994 1053 1054 1074 1117 1172 1263 1290 1433 1477 1511 1627 1637 1758 1781
2086 2115 2128 2194 2224 2306 2319 2364 2398 2415 2425 2505 2693 2638 2727 2742 2806 2847 2916 2925 2965 3005 3044 3048 3054 3207 3264 3289 3348 3365 3401
encrypt e: positions 4 40 47 59 77 78 138 184 259 282 298 522 569 648 742 784 1110 1185 1194 1206 1270 1276 1413 1437 1464 1512 1586 1592 1596 1608 1627 16
796 1797 1848 1908 1942 1946 1975 1994 2112 2188 2294 2298 2385 2486 2508 2510 2563 2590 2633 2804 2848 2870 2920 3101 3206 3247 3312 3317 3420 3477 3481
decrypt e: positions 4 40 47 59 77 78 138 184 259 282 298 522 569 648 742 784 1110 1185 1194 1206 1270 1276 1413 1437 1464 1512 1586 1592 1596 1608 1627 16
796 1797 1848 1908 1942 1946 1975 1994 2112 2188 2294 2298 2385 2486 2508 2510 2563 2590 2633 2804 2848 2870 2920 3151 3206 3247 3312 3317 3420 3477 3481
```

Figura 5.46 Resultado de kat_kem.int.

- Kat_kem.req: Archivo que utiliza los parámetros generados por kat_kem.int para después realizar peticiones al algoritmo.

```
admin@ubuntu:/mccllece-2021023/Reference_Implementation/ken/mceliece348864$ cat kat_kem.req
count = 0
seed = 061550234D158C5EC95595FE04EF7A25767F2E24CC28C479D09086DC9ABC7DE7056A8C26F9E97E085410B02E1FFA1
pk =
sk =
ct =
ss =

count = 1
seed = D81408D734FCBFBEADE303F8A039FAA2A2C957E835AD55B22E75BF57B8556AC81A0DE6AEB45A875C3BFCAF958F
pk =
sk =
ct =
ss =

count = 2
seed = 972365386416448855682673178483491195210631130123112341235124213414171419143214631468153718294
pk =
sk =
ct =
ss =

count = 3
seed = 18832044216523162352239824002434244725271672653268727502761282529022912293829802982311311431483111343341134133422
pk =
sk =
ct =
ss =

count = 4
seed = 22055C5E2CEAC61930A07503FB59F7C2F936A3E0754810DA3CA299A80F8C5DF9223A073E7B90E02EBF98CA2227EBA38C1A
pk =
sk =
ct =
ss =

count = 5
seed = 64335BF29E5DE62842C941766BA1298064385E7121CA26CFC190E7DC3543830557FDD5C03CF123A456D48EFEA43C868
pk =
sk =
ct =
ss =

count = 6
seed = 225D5CE2CEAC61930A07503FB59F7C2F936A3E0754810DA3CA299A80F8C5DF9223A073E7B90E02EBF98CA2227EBA38C1A
pk =
sk =
ct =
ss =

count = 7
seed = EDC76E7C1523E3862552133FEA4D2AB05C69FB54A9354F8B46456A2A407E071DF4650E0E0A5666A52CD094620BC51F9
pk =
sk =
ct =
ss =

count = 8
seed = AA96349193C2C5985ACF89E6AC50C36AE16A2526D7C684F7A3B84ABC7D86FF790E82BADCE89B7C380D66251F97AAAA
pk =
sk =
ct =
ss =

count = 9
seed = 2E0140C7C2696B9F0D4AF555CB4A8931B34863F60E234104FDFE472FEF2FE2C33E0813FC5CAFDE4E30277FE522A9049
pk =
sk =
ct =
ss =

count = 10
seed = AEFB28FD034E0AB403A703B53296E3A545CA479C1D8148E2D501B3C8D88B1034BD986F13F1A7B4671B7E79359FD2AA
pk =
sk =
ct =
ss =

count = 11
seed = CBE5161E8DE02DDA7DE204AE0FB84CA81344B8A8C30FE357A4664E5D298BA03B64184D7DC69F8D367550E5FEA0876D41
pk =
sk =
ct =
ss =
```

Figura 5.47 Resultado de kat_kem.req.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Kat_kem.rsp: Archivo que utiliza las peticiones de Kat_kem.req para generar una serie de respuestas correctas.

```
#5E7EFAA4C0C12C34F5BC2284376D7C4AA5B22F/F8367680CFC2E983FA80686769A37952A79E357421239F7BD8AE4D44A4BB6D05A  
CF59182A4A8F90340405A9F500E45F57736855385A8DD6ECE4A58FA87FAE6DC23C521DB88350B44C8C6CBF5D29D6A9A5AF3D717292  
5D109E424BD9E21A991EBBD69AC0FEBAD5D67E440D6818575EFF67CDEDA4E03C0308A1BE398A1C50D79320EB6C28FC63267F882D10  
5526D76C93681AE8FFDC5058995117AE3158885F2F53D25705A7043281E4103DE87E4FB532A399A39FAB7CD51DEEE4E08F0DBE8F0  
D95AB5756BC025CBF808514941AFEF28CEF49AE8D39044BEA68841D1ACB331FA4048834EE98C5280F2921BF6D5730E37406EBDBD2  
BA5B8900194045601789920FAB8B3479242261284A067A2FBCE77F56DC07E2FA9A7594783971A970F54FFEE4B04296DB17EC4F8EB  
3370B19ECD7D111AE77E021D807EAFCAT1DC485A996A23ADCA043D625703DBCA886BEDE5BABC47C8997FA18DEFDDC5A6BC6C370B  
12C260DBA6064894186F012AFC770C925C393BBCD77D6E23DE0AA470A4E7FCAC14A27A6186888BDAAEF5D1B41E662A76008F3CF  
C5F266ED6D2C12E04FC52CDE7E801DB0A852EB7C1F9E05A6A76D2E66E7AFBB6340520FB62E80313503A48F0BD16086830D72EDE88  
BA862F7B568284AC28811D185A2DB2551E0EA8E042A78188AB68FEF34AF37D4960D922D64F343334F863CF404B1AC0413ECCFDEC  
BAAE5A32EEA3C548E8E1A2E98767A179259BF31D1EC9785BF137E6AC472AB6D77EF09CA9D170446A4B2A9A39BCCED29BF810E7F361  
E518D0820373377D05B0FA7350299F85171BE4C1E68284B5D5440B96B34802B8064D6299D862D334A2B31A98499041CE5289BBC  
CCC10D1A5F4D8E06F30AA22061F55246EF291D94E0E28C2337B4800650328296D35018C16FFF7003144D3A69CFE5057960CBD652  
537E2AD1723F66258BD966E0CF5C8A59636BDFB84E294AD26BB3CB6D758F58D4D94792BB2C38DDC7A5894D6BFFA3C5A88BA113620  
BFDBC76B080B0490C8EC187DA30E8A29E06AD9ABB4C41AAA218306E37CBF7B0934943377F8189FF35945BDD2225FF73F5B845558  
14A8047C27960F27E7A3A7BD79E77D5B7C19A8BECF9527A8138B31C8D3B8CAA21A685E7E421A49219F69873F38D9B9CDA40F88C80  
461F98B6160BD7BB217D2F02EB3549E94F87555BC92D73D1A8C8D248D06A02E2E8942994E7E30FF1640CBC7CE131E10ACA122B5AE4  
321CAC18C24CF89A1506B3D6146559219FC5BEF8F1DD450F28D5AESEC671A4F76BBB5FDD207BF57AA6CB43D5F8558A047FD187054  
B10FE8238A2AB4759F87A683AE90BA4636367E85EB2AF11D0667EBF3D686CB50B93BF14522B3146F98BBC0FCF2AC2A3F5B74A59  
98DB400E386B6DFDSC667BF72D8DF0FE8D45A9E3539B098667CDC99EA8A3A6EB9E551D9AF96DC3A8BBED60F6565D38D163C9868C  
36695CC71E7572FF39F322607ADD9D3B08DAFC55E8CF10606E6937D25E461043C3D53B4443AAB45E684AE698FD5EE193194672EDA  
A96BBB1961CDB873B10251587C943A55C9908152D8C54311133541D6BE0C2974BF7734C877DD5DA4A265306775700A745AFF64913BA  
B0795BAF09092D4B94D109BD8AD095EE5D0456F080C0C24D4FB09BAC7F8089E312E8112055F9D27351DC6338F126AFA079DEE9  
B89B2899CED492FD7158ED24FA9B514D261F49EC217D60EDCF05EEC1402BB9F5D6FD03758C105DD54A7494B82D3C086BE7FE470F2  
B915142CE303AF7DCB152B52697A897DBB0729CB889A91D63547FE798ADB049B2CD39EEAE476B1CB5E35BDF389E0E71BE592903E  
B3D891C230C941A060466C74B5DD846E56173F99E2E8D53101A7FAE8A31F1A87DEDB8CB24DD7A376B5FE77D3EEC700AC89610DA4DE  
B8E2C49AA1C0FAEC98758B0B57B26A481AF12D062325E20F70EE75E7E84395B60088C311D001CE48FBF88DF0994BE834DD1334C9980  
B220BC8ED570878BAF3A9D9F4B9DA80EE33066E12F4CA7F1F8531CA77537EDEFE317A1EE80A19C36C22B5E2145934B6BD33CE5FF124F  
B2EDC19FB22B0A5A3F4121D72ABC101B2EDCEA8D97DF65DC1C0C4F74ECFDCCF4FBF27148A5EAE244222BCBBE9792484D6131672  
B8B19295B1E995EA25AEB7E9710D97D13E15C37F81E4A915D13DF428291371AAB8A5E7267F2B03FED024B6EAFE7582105A227035557  
9F29A946AFA005E619697F787927E89A2  
ct = 2BFA3FF4C00D9E3FEB6765B8CF1360F4A2846AA6706B83BF34B58D4EC53FF6B77861066CA410CF35D0DF442C71D4C3F4D3E62  
BD313956ED8C7ED83E13B29B4D97A9ED9  
ss = F36899E20FE20AE71BD9C4179CE9B9FB9398174181AE277E254F9CDC1B095254
```

Figura 5.48 Resultado de Kat_kem.rsp.

Muestra todos los parámetros de kat_kem.req, todos los datos almacenados están guardados en el archivo comprimido de la entrega.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Posteriormente se accede al directorio Optimized_Implementation.

- Optimized_Implementation: Acceder directorio optimizado y después al directorio intermedio llamado kem cuyo contenido es el siguiente:

```
admin@ubuntu:~/mceliece-20221023$ cd Optimized_Implementation
admin@ubuntu:~/mceliece-20221023/Optimized_Implementation$ ls
kem
admin@ubuntu:~/mceliece-20221023/Optimized_Implementation$ cd kem
admin@ubuntu:~/mceliece-20221023/Optimized_Implementation/kem$ ls
mceliece348864  mceliece460896  mceliece6688128  mceliece6960119  mceliece8192128
mceliece348864f  mceliece460896f  mceliece6688128f  mceliece6960119f  mceliece8192128f
```

Figura 5.49 Accedemos a Optimized_Implementation.

Escogemos mceliece348864 como en el directorio de referencia.

```
admin@ubuntu:~/mceliece-20221023/Optimized_Implementation/kem$ cd mceliece348864
admin@ubuntu:~/mceliece-20221023/Optimized_Implementation/kem/mceliece348864$ ls
api.h      build      crypto_kem_mceliece348864.h  gf.c      kat_kem.req  operations.c  randombytes.h  sk_gen.h      transpose.c
apenes.c   controlbits.c  decrypt.c        gf.h      kat_kem.rsp  operations.h  root.c       subroutines  transpose.h
apenes.h   controlbits.h  decrypt.h        int32_sort.h  KATNUM     params.h     root.h      synd.c      uint64_sort.h
pm.c       crypto_hash.h encrypt.c        kat        Makefile    pk_gen.c     run        synd.h      util.c
pm.h       crypto_kem.h  encrypt.h       kat_kem.int  nist      pk_gen.h    sk_gen.c    THIS_IS_A_COPY_OF_REF  util.h
```

Figura 5.50 Mostramos contenido del directorio mceliece348864.

Tenemos que abrir los archivos generados kat_kem.int, kat_kem.req y kat_kem.int:

- Kat_kem.int: Misma función que la implementación de referencia.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

```
admin@ubuntu:~/ncalice-20221023/optimized_Implementation/kem/ncalice34886$ cat kat_kem.int
encrypt e: positions 0 81 92 105 216 378 389 468 504 586 617 738 928 1030 1061 1070 1072 1177 1294 1316 1366 1389 1510 1670 1721 17
1995 2003 2053 2068 2082 2113 2202 2363 2418 2508 2541 2553 2603 2641 2704 2731 2818 2951 2974 3091 3132 3157 3158 3174 3182 3215 3
decrypt e: positions 0 81 92 105 216 378 389 468 504 586 617 738 928 1030 1061 1070 1072 1177 1294 1316 1366 1389 1510 1670 1721 17
1995 2003 2053 2068 2082 2113 2202 2363 2418 2508 2541 2553 2603 2641 2704 2731 2818 2951 2974 3091 3132 3157 3158 3174 3182 3215 3
encrypt e: positions 75 117 159 212 254 277 297 307 466 667 745 790 828 846 871 873 878 885 946 977 1067 1106 1268 1327 1415 1541 1
1844 1863 1868 1893 1902 1947 1994 2024 2061 2216 2252 2464 2544 2565 2570 2751 2866 2864 2865 2885 2941 2984 2986 3087 3100 3239
decrypt e: positions 75 117 159 212 254 277 297 307 466 667 745 790 828 846 871 873 878 885 946 977 1067 1106 1268 1327 1415 1541 1
1844 1863 1868 1893 1902 1947 1994 2024 2061 2216 2252 2464 2544 2565 2570 2751 2866 2864 2865 2885 2941 2984 2986 3087 3100 3239
encrypt e: positions 31 34 296 351 363 486 505 537 567 651 681 683 705 794 831 843 861 869 928 1115 1151 1163 1350 1367 1489 1490 1
1939 1986 2032 2117 2136 2167 2189 2195 2246 2300 2353 2374 2387 2405 2498 2430 2434 2597 2795 2734 2763 2785 3102 3104 3107 3193
decrypt e: positions 31 34 296 351 363 486 505 537 567 651 681 683 705 794 831 843 861 869 928 1115 1151 1163 1350 1367 1489 1490 1
1939 1986 2032 2117 2136 2167 2189 2195 2246 2300 2353 2374 2387 2405 2498 2430 2434 2597 2705 2734 2763 2785 3102 3104 3107 3193
encrypt e: positions 1 84 239 261 308 358 385 496 530 556 658 709 716 939 1021 1027 1137 1192 1245 1294 1378 1502 1600 1655 1658 18
2107 2180 2253 2263 2397 2407 2427 2433 2447 2460 2466 2533 2542 2563 2682 2718 2763 2897 2906 2963 2988 3110 3144 3172 3178 3218 3
decrypt e: positions 1 84 239 261 308 358 385 496 530 556 658 709 716 939 1021 1027 1137 1192 1245 1294 1378 1502 1600 1655 1658 18
2107 2180 2253 2263 2397 2407 2427 2433 2447 2460 2466 2533 2542 2563 2682 2718 2763 2897 2906 2963 2988 3110 3144 3172 3178 3218 3
encrypt e: positions 62 71 73 75 134 190 262 361 441 510 552 723 783 809 1039 1080 1084 1152 1172 1283 1288 1372 1388 1500 1523 158
394 1949 2020 2093 2147 2159 2286 2305 2400 2427 2433 2457 2478 2497 2536 2558 2569 2632 2675 2703 2708 2843 2847 2957 3024 3025 31
decrypt e: positions 62 71 73 75 134 190 262 361 441 510 552 723 783 809 1039 1080 1084 1152 1172 1283 1288 1372 1388 1500 1523 158
394 1949 2020 2093 2147 2159 2286 2305 2400 2427 2433 2457 2478 2497 2536 2558 2569 2632 2675 2703 2708 2843 2847 2957 3024 3025 31
encrypt e: positions 97 236 253 386 416 440 480 550 568 626 731 784 834 911 952 1062 1130 1231 1234 1235 1242 1341 1417 1419 1432 1
1883 2044 2165 2316 2352 2398 2400 2434 2447 2527 2617 2653 2687 2750 2761 2852 2902 2912 2938 2980 2982 3111 3114 3143 3148 3188
decrypt e: positions 97 236 253 386 416 440 480 550 568 626 731 784 834 911 952 1062 1130 1231 1234 1235 1242 1341 1417 1419 1432 1
1883 2044 2165 2316 2352 2398 2400 2434 2447 2527 2617 2653 2687 2750 2761 2852 2902 2912 2938 2980 2982 3111 3114 3143 3148 3188
encrypt e: positions 21 69 141 160 198 218 257 294 296 341 368 389 463 474 568 645 728 845 853 1047 1058 1085 1160 1291 1312 1342 1
1659 1682 1726 1751 1796 1881 1935 1961 1966 2004 2047 2082 2157 2355 2368 2435 2455 2494 2598 2635 2654 2700 2765 2796 2827 2864
decrypt e: positions 21 69 141 160 198 218 257 294 296 341 368 389 463 474 568 645 728 845 853 1047 1058 1085 1160 1291 1312 1342 1
1659 1682 1726 1751 1796 1881 1935 1961 1966 2004 2047 2082 2157 2355 2368 2435 2455 2494 2598 2635 2654 2700 2765 2796 2827 2864
encrypt e: positions 65 150 239 332 377 496 498 521 575 708 754 775 943 1043 1046 1060 1075 1174 1221 1262 1267 1372 1380 1553 1571
13 1748 1987 2028 2090 2205 2212 2275 2291 2296 2303 2322 2323 2357 2422 2640 2644 2713 2740 2770 2796 2968 2996 3071 3249 328
decrypt e: positions 65 150 239 332 377 496 498 521 575 708 754 775 943 1043 1046 1060 1075 1174 1221 1262 1267 1372 1380 1553 1571
13 1748 1987 2028 2090 2205 2212 2275 2291 2296 2303 2322 2323 2357 2422 2640 2644 2713 2740 2770 2796 2968 2996 3071 3249 328
encrypt e: positions 157 332 355 471 519 532 565 683 772 790 808 812 943 959 994 1053 1054 1074 1117 1172 1263 1290 1433 1477 1
2086 2115 2128 2194 2224 2306 2319 2364 2398 2415 2425 2505 2603 2638 2672 2727 2742 2806 2847 2916 2925 2965 3005 3044 3048 3054 3207
decrypt e: positions 157 332 355 471 519 532 565 683 772 790 808 812 943 959 994 1053 1054 1074 1117 1172 1263 1290 1433 1477 1
2086 2115 2128 2194 2224 2306 2319 2364 2398 2415 2425 2505 2603 2638 2672 2727 2742 2806 2847 2916 2925 2965 3005 3044 3048 3054 3207
encrypt e: positions 4 40 47 59 77 78 138 184 259 282 298 522 569 648 742 784 1110 1185 1194 1206 1270 1276 1413 1437 1464 1512 158
796 1797 1848 1908 1942 1946 1975 1994 2112 2188 2294 2298 2385 2486 2508 2510 2563 2598 2633 2804 2848 2870 2920 3151 3206 3247 33
decrypt e: positions 4 40 47 59 77 78 138 184 259 282 298 522 569 648 742 784 1110 1185 1194 1206 1270 1276 1413 1437 1464 1512 158
796 1797 1848 1908 1942 1946 1975 1994 2112 2188 2294 2298 2385 2486 2508 2510 2563 2598 2633 2804 2848 2870 2920 3151 3206 3247 33

```

Figura 5.51 Resultado de kat_kem.int.

- Kat_kem.req: Misma función que la implementación de referencia.

```
admin@ubuntu:~/ncalice-20221023/optimized_Implementation/kem/ncalice34886$ cat kat_kem.req
count = 0
seed = 0E1550234D15BCE5C5E9559FE04E7425767F2E24CC28C479D09B06DC9ABCDFE7056A8C266F9EF97ED085410BD2E1FFA1
pk =
sk =
ct =
ss =

count = 1
seed = D81C4D80734FCFBFBEADE303F8A039FAA2A2C9957E835AD55B2E75BF57B556ACB1ADDE6AEEB4A5A875C3BFCAFDA958F
pk =
sk =
ct =
ss =

count = 2
seed = 64335BF29E50E62842C9417668A129B0643B5E7121CA26CF198EC7DC3543838557F005C03C123A456D048FEA43C868
pk =
sk =
ct =
ss =

count = 3
seed = 225D5CE2CEAC61930A07503FB59F7C2F936A3E0754810A3CA299A80FBC5DF9223A073E7B90E02EBF98CA2227EBA38C1A
pk =
sk =
ct =
ss =

count = 4
seed = EDC76E7C1523E3862552133FEA402AB05C69FB54A9354F0846456A2A407E071DF4650EC0E0A5666A52CD09462DBC51F9
pk =
sk =
ct =
ss =

count = 5
seed = AA93649193C2C5985ACF8F9E6AC50C36AE16A2526D7C684F7A3B84ABC0D7B6FF790E828ADCE89BC7380066251F97AAAAA
pk =
sk =
ct =
ss =

count = 6
seed = E2014DC7C269689F6D4AF555CBA4B931B3483FF0E2341D4FDFE472FEE2F2E2C33E0813FC5CAFDE4E30277FE522A9049
pk =
sk =
ct =
ss =

count = 7
seed = AEFB561E80E0200A7D0E204AE0B0FBB4CA81344BA8C30FE357A4664E5D2988A03B64184D7DC69F8D367550E5FEA0876D41
pk =
sk =
ct =
ss =

count = 8
seed = CBE5161E80E0200A7D0E204AE0B0FBB4CA81344BA8C30FE357A4664E5D2988A03B64184D7DC69F8D367550E5FEA0876D41
pk =
sk =
ct =
ss =

count = 9
seed = B4663A7A9883386A2AE4CB093787E247BF260887E3826D1B8DBEB679E49C08B286E114F0E9F42F61F630EC42B4F974B46
pk =
sk =
ct =
ss =
```

Figura 5.52 Resultado de kat_kem.req.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

- Kat_kem.rsp: Misma función que la implementación de referencia

```
B0B158EAD13FCDE9E718F509CB267318288765CD2AE94F7C0D5E0086BD97EE2DF10487C000D8AE889929DCB698C7B6CDB5C1E6CA5820  
04FF1CF47AA601D2A400076F109318CDB81B33103800EF5E5E32CD2849C3A68F285AF1DE34E60FB52F5A310E78CBCEFA494941CE98E  
ABFA076D9AF3DE11F987E5C9CA9E88704CF9DA4060D3E14E05249582785F1D647E3B762297F9C4877DBB093C90E0000A2E2A889C346  
1181D50288265EE5F129E4035698083BDEC329BF339471AA9A9293480B185978CF83C7C2646261090D0830F840534723D40123F0774E368  
SBFBAB2E4EB2A523F23D4FC41E68B8405EF04A51DAB4A00F9C314528CB8F8FEEDADD2A09CB13FAC4D0D757A74B3FA741FC1915EF429B39  
1A15072F5A08A232F87C565996C848EBA907344E8FB0D131E7651401BCFA7313D6E18359155FA347602E847765D2670184E  
DCCB69C7E778AC322AA09A156288C9863D25677368C201102ECCD41625251F5A142ED0A009162015C953D3052684EFD8C61B82F80CB3A2  
522C79D24AB5E887BEAC7045EF8023A427CCE93877889E08A8F68A1F047D7D8468B04B2D1931332E0D832C0B990F30182B04B8808C623E2  
9C34250EAA889599A79215582C5EA76495E1F8A9E489E43548DE7CBA137A6352406E5730E6D9585B2E0A  
3F82C2A901DA03580112912C20412D86110667D91D90408FS8A1F8417525CC3704387C89C6447D7A8E2E930B4D197056736E8D310  
0556229338280682040789178FC572E169899372D6FC0329A216CF88B78D7A74E422B97D22972170DD14446E5DB5D52084  
DA62D4C996C6971245779E0266789D9ESEC25BCE83E2F4CA81B084E8CCB18D01AD4625CDB9E8FEC90178F2E6472E4615837389F38FEC4  
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
33D70862604022E46492562562C0083947493682E60558A26624124906A456A0AAC596E1717273C9F699C30013030B9C5996E98BE51  
243A01F4DE2C2944B06A507A0FC72BF801E8646112E208808073D06E9FA74894B0282ED9617381C7E99A5B16AD399D91D56C815F9E2  
5494646990000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
01043E572200C50D62E2DB64980E08323E2044D0A285082CA03D61106021A042908E0831882446E16F5558824342D403261B840  
0280114E80F7A8058D3A68983B206A603C916159E409C5E0713942789C431850F6E21BF84E83E45E00212FC01500E6A5968994FB34  
576C46821448B018866E05D4011600000000000000000000000000000000000000000000000000000000000000000000000000000000000  
002C25009980B3E2F215050318160063E008085E49153C7B014038C605306320604434740F91424ZC6E99206262A9901609297102525  
965862B82706212C212A35038A8E94A6C15C2148F85E0FFC8484924A738E14A281C59D478850AC6A5E13C700B83584D95CC1800461E7A  
1446F0A1C495A196CC808890C3C73078704980B3C1E00EA0000000000000000000000000000000000000000000000000000000000000000000  
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
C2C0280311B06140480040142CAC00D14228001A189C039407812305D17405A016C80A22886370898578D9492108  
41065C48480022A49588012409588E06D035C871095B10408193272048028A49882C024A778A99544CE74044A8107BF37D610B1D0EE73249  
899F6AD00AECC33B2A214976B68C889963A64E84B4CE99C86F919D2EDC9894C993C9E187BB8C4FF3EDCB03230257064D206E62B8948  
46EABA305FF2C5A0982E1A0F1511B648593B5A8E8E298366D672555D6C7788C87B0D017AD830D611C6B0F8D7B3B3C10D8  
4A8C88B226C323761CD04E63984BEC840094E5D5E7981F972D757A93C586F3D415DEC19B9C6116A87B7C25812F949E951A23A9A  
AA8F87E025F5CFBCE025A51E08A2C0D5C52A5B87E34E0FAE28765D07810F824112D4A6E74E9B8C618997033F98F888BA8A83E90318  
B52EC70852E0351350B8D3A5637082E84C244E3D60153661808781D1087381D2A52C83B68A5639A8ECD8A441E88BA44B421349780  
65D904E35668CB0D8E40F3109E8893CF89B0E80081C93FCE84077F73666CBFCB7C35B6E8D8B0D71A45E95322B5306E4CE2A60A20  
9462B0EF5ABE5B241E90B4031C4F8487D59A696BE8083170DC1E2A75982F78E2F4242D47873812E200223E061  
D45DECA2E0B39323E502385F80259858138F7A1984F282DBCCF46188EBFF8A843A5D5982F38B9823E98CAE8D12F6280FD4E8B4  
E49CA88A15900A41964D28FCE418593234D3C1CA409C80B43ADC3804F8750C980719049C932A5194F2E4770341863A89462735958  
C2A7BF1E493C02A594527D9F120D30181B8C476D4B4A3D90A633E9C6684B8D9696C10473D8B7E7F75F4665074ED0  
880F10CE81C80D9E5945D2A8056F6D3E11CC54D5E860C72055121C8CAF744CC76C830E1A24F1985DCEAC1237C39DC30F411129F27E  
B54D9318F80718C06F3055A7F98C43FB4885E0FCFAC1D43DEE4E96A4A08D06CA2845C245D698E2F3E7B00EFD4A2DB746575C2780C81  
857E7EFA44C9D12C345F5B2284376DC7C4A5B2F78376800FCF8379838F7A68766769A37952F79E375421239F7BDBAE4D444B86005A57D8  
CF59182A4A8F90348405A9P500E45F57736855385A8D06ECE4A58F87FAE60D2C321D88350844C6CBF6C8F5D29D69A5A8F3071292A178  
S1D918E424B9D9E1A9197B8D699A09E05F0E083170C2F658459828D93A56E47D4A6E41376D628F8F882D10B106  
6526D76C3681A8E905F0C850959117A8E158885F2F53D25705A7043281E4103DE87E4F8532A399A39F487C5D10EE4E08F0D8E08F02D2A  
D9585756BC025CBF8038A594830948841D1A331F0A408838A6894833F802D172046E46B8F2921BF6D5730E37406E8D0B2D2E8A8  
8A588900194045601789920FABF8742261284A067942F8E77F56D0C7E2F9A7594783971A970F54F7EE4B804296D8B17EC4F8EBB93  
8370B19E70111A7E77E021D887EFA7C7C10C458A96A23A0C403D625C703DCB868EDE5B8C47C8997F1A80EFD5C6A8C6C37082FEE  
12C260D8B4606489186F012AF377E8893CF89B0E80081C93FCE84077F73666CBFCB7C35B6E8D8B0D71A45E95322B5306E4CE2A60A20  
C5F266E6D2C12E04FC52CD7E0801D0B0A852E7B7C1F9E05M6A76D2E66E7A7FB8340520F6B62E80313503A48F0B16086830D072EDE88A20  
8A862F78568248C428811D185A2D82551E0E8A042A78188A686B87F7E0D4796D20D64F34334F863C40481A0413ECCF0D8C622  
BAAE532EEA3C5488E81E2A9E8761A79259BF32D7E6A472AB6D77EF09C9A01704464A6B2A9A39BCCED29BF810E7F361F79A  
E510D802373777D580F7A350299F85171B8E4C116E882845D5440B98B4380B28064D629986D62D334A2B31A98499041CE5289B8ECA58R  
CCC10D158568243F8062E0645817B788E06D28337B48006058328960350146F703014D93A69CFE057F960CB6D5241C8  
537E2AD1723F662588B09660C5F8A5A96368DBF8E2944D2683B86D758F158D04D94792B82C38D075A8F7351DC689823F5C8  
3F0BC7F68B8800F498C8E187D030A829E06A9BA8441AA218306E37C87B9034943778189F73594580D22258F73F5B8845558F8F0  
14A8047C27960FF27E7A37B79E77D5B7C19A8EBCF9527A83C8D31C8D3B8C8A21A685B77E421A9219F69873F38D9B9CDA4088C807F9  
461F98B8160B0D7B8217D2F02E85349E94F857558C92D7D3A18C8D248D06A8B942994E3701F6408C7CE131018A122B5AE4D481  
821CAC7424C89A1590B3D61456592159C85B5E5CE67D1405F2875A6C843D5F8558A047F187054D63A  
B10F8E238A24B74579F87A683A8E908A4636367E85E2F11D0667E87F3D0668C8505893BF1452283146F98BEC0FFC2A23F5874A594C88  
98DB400E386860FD5C6678F72D8D0F8E0D58A9E9F53598B908667DC99E8A3A6E89B551D9F90C3A8BBED60F6565D38D163C9868CCCC  
86695717E572F8B08027060248464645667C917C9091528C4311333541D6E80C29748F7734C87D5A4265306775700A745AFF49138A1409  
A96B8161CD873B1021587C9A55C9908152D8C4311333541D6E80C29748F7734C87D5A4265306775700A745AFF49138A1409  
3B07958A9096204894098D955A8D5045F800C24D4FB098ABAC7F8089E312E81120559F9D27351DC6338F126FA0790EE98020  
88982899C8D492F7D158E2D4F9B814D261F49E2C17D60E8DC05EEC14028B9F5D6F0D37585C105D54A7494882D0C0868E7F4E70F278C  
8915142CE3E03A7FDC8152B52697A897DB0729CB889A91D63547F798A80B098B2CD39EEAE74681CB5E35BD8F389E0E71B8E592903E7CF  
3D891C230C941A0604667C45BD846E5617398E2E8D53101A7FAE8A31F1A87DDEB8CB24D7A376B5F77D3EEC700AC89610D44ECD50  
3E2C49A1C0FAEC98758B0857B26A481F12D0623252E8F7EE7459358E0088C311D01CE48FB8F8D0F0994B8E34D01334C9986E2A  
0226BC8E05708778F3A9D9F4B9D9A88E33066E12F4CA7F1F8531CA77537E8F317A1EE80A19C36C2285E2145934B68033CE5FF124F752  
32EDC19F22B0A5A3F4121D72ABC7F101B2EDEA80D97D65DC1C04F74E4CFDCCF48B271418A5EA2E442228CBBE97924840D131672B872  
3B19295B1E995EA25A87E971009D7D13E15C37F81E4A915013D4F28291371A8A5E7267F2B03FED024B6EAF8758210522763557927D  
9F29A946FA005E616967787927E89A2  
ct = 2FA3FF4C009F3FFB67658B8C1360F4A2846A6706B83BF34858D4E53F6B77861066CA410CF35D0F442C71D4C3F4D3F625963  
3D313956ED8C7ED83E13B2984D97A9ED9  
ss = F3699E820F20AE71BD9C4179CE9B9F893174181AE277E254F9C6C1B095254
```

Figura 5.53 Resultado de Kat_kem.rsp.,

Muestra todos los parámetros de kat_kem.req, todos los datos almacenados están guardados en el archivo comprimido de la entrega.

Para Additional_Implementations no se va a tratar ya que es similar a los otros dos directorios y por lo tanto refleja datos similares.

Capítulo 6

Conclusiones

Reflexionando ha sido un proyecto que me ha hecho mejorar mi capacidad de captar información de fuentes fiables, como también obtener conocimientos nuevos en Linux y en criptografía. Al principio me surgió varias dudas por ser un tema nuevo tratado apenas en la asignatura Protocolos y Comunicaciones Seguras impuesta por mi tutor.

Decidí escoger este tema debido a que me daba curiosidad el funcionamiento de la computación cuántica como también pensar en cómo esta puede cambiar el mundo para bien o para mal. No todo ha sido un camino de rosas ya que ha habido algunas características que se sobrepasaba el plazo inicial como puede ser enunciar matemáticamente los algoritmos del punto cuatro, pero como no era el objetivo de mi TFG decidí no explicarlas, ya que el objetivo es tener una toma de contacto y experimentar con los algoritmos. Como conclusión de cuál es el algoritmo óptimo del concurso NIST tengo que decir que el finalista el algoritmo kyber según los datos arrojados es el más completo de los tres finalistas, aunque me quedo con las

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

ganas de realizar más pruebas para ver diversas diferencias entre los algoritmos.

Como implementación futura me gustaría hacer un análisis matemático de cada algoritmo además de realizar pruebas a individuales a cada uno.

Considero que he cumplido los objetivos establecidos en el TFG y aprendido una nueva rama que me puede servir en el futuro.

Capítulo 7

Referencias

[1] FREDY CHALA, Y. (2019). *Unad. IMPORTANCIA DE LA APLICACIÓN DEL MECANISMO DE CIFRADO DE INFORMACIÓN EN LAS EMPRESAS PARA LA PREVENCIÓN DE RIESGOS COMO ATAQUES, PLAGIO Y PERDIDA DE CONFIDENCIALIDAD:*

<https://repository.unad.edu.co/bitstream/handle/10596/30745/yfchala..pdf>

Azure Microsoft. (s.f). *Azure Microsoft*. Explicación de los cúbits:

<https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-qubit/#qubit-vs-bit>

Barbieri, A. (3 de Noviembre de 2022). *Orange. ¿Se convertirá la computación cuántica en un problema de seguridad? Los retos que nos esperan:*

<https://blog.orange.es/innovacion/computacion-cuantica-problema-seguridad/>

Castellanos, L. R. (19 de Enero de 2017). *Seguridad Informática WordPress*. 6. Criptografía:

<https://lcseguridadinformatica.wordpress.com/2017/01/19/6-criptografia/>

Chen, C., Danba , O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J., . . . Zhang, Z. (30 de Marzo de 2019). *NTRU*. NTRU Algorithm Specifications And Supporting Documentation: <https://ntru.org/f/ntru-20190330.pdf>

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Claramunt Carriles, S. (Mayo de 2022). *Escuela Politécnica Superior de Alicante.*

Investigación sobre la computación cuántica:

https://rua.ua.es/dspace/bitstream/10045/124691/1/Estudio_de_la_computacion_cuantica_en_los_diferent_Claramunt_Carriles_Sergio.pdf

Comisión Federal de Comercio. (s.f.). *Marco de ciberseguridad del NIST.* Qué es y como funciona EL MARCO DE CIBERSEGURIDAD DEL NIST:

<https://www.ftc.gov/es/guia-para-negocios/protegiendo-pequenos-negocios/ciberseguridad/marco-ciberseguridad-nist#top>

Comisión federal de comercio. (s.f.). *Comisión federal de comercio. EL MARCO DE CIBERSEGURIDAD DEL NIST:* <https://www.ftc.gov/es/guia-para-negocios/protegiendo-pequenos-negocios/ciberseguridad/marco-ciberseguridad-nist>

Delgado, S. (26 de Enero de 2023). *RSA se mantiene intacto frente al algoritmo de Shor y otros ataques cuánticos.* MCPRO:

<https://www.muycomputerpro.com/2023/01/26/rsa-intacto-algoritmo-shor-ataques-cuanticos#:~:text=RSA%20se%20mantiene%20intacto%20frente%20al%20algoritmo%20de%20Shor%20y%20otros%20ataques%20cu%C3%A1nticos>

Domenech Belda, C. (2004). *BIBLIOTECA VIRTUAL MANUEL DE CERVANTES.* La escritura jeroglífica egipcia: https://www.cervantesvirtual.com/obra-visor/la-escritura-jeroglifica-egipcia-0/html/001c6860-82b2-11df-acc7-002185ce6064_2.html

EcuRed. (s.f.). *EcuRed. Máquina de Turing:*

https://www.ecured.cu/M%C3%A1quina_de_Turing

El Cedazo. (1 de Abril de 2014). *El Cedazo. Computación Cuántica I – Introducción:*

<https://eltamiz.com/elcedazo/2014/01/04/computacion-cuantica-i-introduccion/>

El País. (s.f.). *El País. El País:*

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

[https://imagenes.elpais.com/resizer/00aRr553FHRPv7h11eRsVXQo4mE=/1960x0/filters:focal\(53x134:63x144\)/cloudfront-eu-central-1.images.arcpublishing.com/prisa/3FKNA3NWMRFJBOIDWNISFFW4KY.jpg](https://imagenes.elpais.com/resizer/00aRr553FHRPv7h11eRsVXQo4mE=/1960x0/filters:focal(53x134:63x144)/cloudfront-eu-central-1.images.arcpublishing.com/prisa/3FKNA3NWMRFJBOIDWNISFFW4KY.jpg)

Farrán, J. (2016). *Revista de la Escuela de Ingeniería Informática de Segovia*. Criptografías post-cuántica: <http://www.inf5g.uva.es/node/469>

Frwiki. (s.f). *Frwiki*. Puerta de Toffoli: https://es.frwiki.wiki/wik/Porte_de_Toffoli

Giematic . (s.f). *Giematic*. EasyRSA:

<https://giematic.etsisi.upm.es/EasyRSA/EasyRSA1.html>

González Cornejo, J. (Marzo de 2020). *DocIRS Technology*. Puertas Lógicas Clásicas y Cuánticas: https://www.docirs.cl/Puertas_Circuitos_Cuanticos.asp

Gutiérrez, P. (28 de Diciembre de 2012). *Genbeta:dev*. ¿Qué es y como surge la criptografía?: un repaso por su historia: <https://www.genbeta.com/desarrollo/que-es-y-como-surge-la-criptografia-un-repaso-por-su-historia>

Hmong. (s.f). *Hmong*. Algoritmo de Grover: https://hmong.es/wiki/Grover%27s_algorithm

Huang, R. (15 de Junio de 2021). *Forbes*. Here's How Quantum Computers Will Really Affect Cryptocurrencies: <https://www.forbes.com/sites/rogerhuang/2021/06/15/heres-how-quantum-computers-will-really-affect-cryptocurrencies/?sh=6e74a0853229>

IBM. (1 de Marzo de 2021). *IBM*. Criptografía de clave pública:

<https://www.ibm.com/docs/en/ztpf/2020?topic=concepts-public-key-cryptography>

Judson, T. (6 de Diciembre de 2017). *Algebra Abstracta: Teoría y Aplicaciones*. Teoría de Galois: <http://abstract.ups.edu/aata-es/galois.html>

Martínez Mateo, J. (Abril de 2008). *Universidad Politécnica Madrid*. Criptografía cuántica aplicada: https://oa.upm.es/1298/1/PFC_JESUS_MARTINEZ_MATEO.pdf

McEliece. (23 de Octubre de 2022). *McEliece*. Classic McEliece Implementation:

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

<https://classic.mceliece.org/impl.html>

Mena Viveros, I. (19 de Noviembre de 2013). *Artículo de Computación cuántica*.

Computación cuántica:

https://d1wqtxts1xzle7.cloudfront.net/33329861/articulo_compcuantica-libre.pdf?1395964176=&response-content-disposition=inline%3B+filename%3DComputacion_Cuantica.pdf&Expires=1694658160&Signature=XL5N87iTH~TBkJvPCE4cGamzY27aYp6EfLF7Tu4YaE~fGu7k~77Zpm~7bCRJ

Moreland, K. (26 de Abril de 2022). *Ledger academy*. ¿Qué son las claves públicas y privadas?: <https://www.ledger.com/es/academy/que-son-las-claves-publicas-y-privadas>

Museu Informática. (2021). *Museu Informática*. Cifrado de César:

<https://museo.inf.upv.es/blog/2021/05/14/cifrado-de-cesar/>

Numerentur Org. (s.f). *Numerentur Org*. Cifrado en bloque I: <https://numerentur.org/cifrado-en-bloque/>

Orange. (3 de Noviembre de 2022). *Orange*. ¿Se convertirá la computación cuántica en un problema de seguridad? Los retos que nos esperan:

<https://blog.orange.es/innovacion/computacion-cuantica-problema-seguridad/>

Owasp. (s.f). *Owasp*. QKD BB84: <https://owasp.org/www-pdf-archive/BB84.pdf>

Pq-Crystals. (23 de Diciembre de 2020). *Pq-Crystals*. CRYSTALS Cryptographic Suite for Algebraic Lattices: <https://pq-crystals.org/kyber/>

Pq-Crystals. (23 de Diciembre de 2020). *Pq-Crystals*. CRYSTALS Cryptographic Suite for Algebraic Lattices: <https://pq-crystals.org/kyber/>

PQSecure. (2020). *PQSecure*. Overview of NIST Round 3 Post-Quantum cryptography

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Candidates: <https://www.pqsecurity.com/wp-content/uploads/2020/07/Round-3.pdf>

PQSecure. (2020). *PQSecure*. Overview of NIST Round 3 Post-Quantum cryptography

Candidates: <https://www.pqsecurity.com/wp-content/uploads/2020/07/Round-3.pdf>

Raposo Briceño, S. A. (2019). *Universidad Politécnica de Madrid*. Criptografía de curvas elípticas. Fundamentos matemáticos e implementación.:
https://oa.upm.es/56215/1/TFG_SANTIAGO_ALFONSO_RAPOSO_BRICENO.pdf

Real Casa de la Moneda. (s.f). *Real Casa de la Moneda*. Infraestructura de clave pública:

<https://www.cert.fnmt.es/curso-de-criptografia/infraestructura-de-clave-publica>

Real Casa de la Moneda. (s.f). *Real Casa de la Moneda*. Firma digital:

<https://www.cert.fnmt.es/curso-de-criptografia/criptografia-de-clave-asimetrica/firma-digital>

Real Casa de la Moneda. (s.f). *Real Casa de la Moneda Fabrica Nacional de la Moneda y Timbre*. Criptografía de clave asimétrica: <https://www.cert.fnmt.es/curso-de-criptografia/criptografia-de-clave-asimetrica>

Salas Peralta, P., & Sanz Sáenz, Á. (15 de Mayo de 2000). *Universidad Politécnica de Madrid*. COMPUTACIÓN CUÁNTICA: NUEVAS PERSPECTIVAS:

<https://upcommons.upc.edu/bitstream/handle/2099/9946/Article009.pdf>

Tecnología + Informática. (2021). *Tecnología + Informática*. Qué es Tecnología? Definición y evolución: <https://www.tecnologia-informatica.com/que-es-tecnologia/>

Urrego Urrego, J. J. (2019). *Intitución Universitaria*. Método criptográfico para cifrar información usando los estados cuánticos de polarización de fotones individuales:

https://repositorio.itm.edu.co/bitstream/handle/20.500.12622/2090/Rep_Itm_mae_Urrego.pdf?sequence=1&isAllowed=y

Villar , J. (Enero de 2017). *Web Mat Upc*. Informe eSAMCid sobre estructuras de acceso- 2.

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA

Esquemas para compartir secretos:

<https://web.mat.upc.edu/jorge.villar/esamcid/rep/accs/reportaccessse2.html#x4-40002>.

Villar Santos, J., Padró Laimón, C., & Sáez Moreno, G. (s.f). *Universidad Politecnica de Cataluña. COMPARTICIÓN DE SECRETOS EN CRIPTOGRAFÍA :*

<https://upcommons.upc.edu/bitstream/handle/2099/9843/Article018.pdf?sequence=1&isAllowed=y>

Wikimedia. (s.f). *Wikimedia. Algoritmo de Shor:*

https://www.wikiwand.com/es/Algoritmo_de_Shor#Referencias

Wikipedia. (6 de Diciembre de 2022). *Wikipedia. Infraestructura de clave pública:*

https://es.wikipedia.org/wiki/Infraestructura_de_clave_p%C3%BAblica

Wikipedia. (s.f). *Wikipedia. Puertas Cuánticas:* https://es.wikipedia.org/wiki/Puerta_cu%C3%A1ntica

Xifré Solana, P. (2009). *Universidad Carlos III. Antecedentes y perspectivas de estudio en historia de la Criptografía:* <https://e-archivo.uc3m.es/handle/10016/6173>

HACIA UN ESTÁNDAR DE CRIPTOGRAFÍA POST-CUÁNTICA