Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros de Telecomunicaciones**

Máster Universitario en Ingeniería de Redes y Servicios Telemáticos

Trabajo Fin de Máster

Integración de Tecnologías de Distribución Cuántica de Claves (QKD) en Protocolos Criptográficos Clásicos

Autor(a): Jaime Sáez de Buruaga Brouns
Tutor(a): Juan Pedro Brito Méndez
Ponente: Luis Bellido Triana

Madrid, Junio 2022

Este Trabajo Fin de Máster se ha depositado en la ETSI de Telecomunicaciones de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Ingeniería de Redes y Servicios Telemáticos

Título: Integración de Tecnologías de Distribución Cuántica de Claves (QKD) en Protocolos Criptográficos Clásicos

Junio 2022

Autor(a): Jaime Sáez de Buruaga Brouns
Tutor(a): Juan Pedro Brito Méndez
Lenguajes y Sistemas Informáticos e Ingeniería del Software
ETSI Informáticos
Universidad Politécnica de Madrid
Ponente: Luis Bellido Triana

Resumen

La criptografía clásica se basa en operaciones computacionales con números primos imposibles de calcular para computadores clásicos en un tiempo polinomial. Sin embargo, la computación cuántica supone un peligro para esta estrategia. La computación cuántica consiste en construir ordenadores que hacen uso de los principios de la mecánica cuántica para la computación. El problema criptográfico surge debido a que algunos problemas que no son calculables en tiempo polinomial por un computador clásico, sí lo son por un computador cuántico. Esta es la razón por la que se dice que la criptografía clásica está rota, aunque de momento sólo de manera teórica ya que no se ha logrado construir un ordenador cuántico completo.

La distribución de clave cuántica (*Quantum Key Distribution*) como mecanismo de obtención de clave simétrica se ha desarrollado de manera exponencial en los últimos años, y ha demostrado ser una implementación segura contra ataques de ordenadores cuánticos. Debido al mencionado auge, se han creado estándares para la operación de este esquema. Uno de ellos es el desarrollado por el ETSI, que es el que usaremos en este trabajo (ETSI GS QKD).

El protocolo de transporte *Transport Layer Security* (TLS) es un protocolo criptográfico que proporciona comunicaciones seguras a través de Internet, siendo el más usado entre esquemas cliente-servidor.

El objetivo final de este trabajo es la completa integración de estándares operacionales que ejecutan mecanismos de distribución de clave cuántica en el protocolo de seguridad de transporte TLS (en su versión 1.3) a través de la librería de Python TLSlite, así como su implementación y despliegue en una red real que obtiene clave de dispositivos cuánticos reales.

Palabras clave:

Distribución de Clave Cuántica, Comunicaciones Cuánticas, Protocolo TLS, Criptografía, ETSI GS QKD, Intercambio de Clave, Seguridad Post-Quantum.

Abstract

Classical cryptography is based on computational operations with prime numbers that are impossible for classical computers to compute in polynomial time. However, quantum computing poses a danger to this strategy. Quantum computing involves building computers that use the principles of quantum mechanics for computation. The cryptographic problem arises because some problems that are not computable in polynomial time by a classical computer are computable in polynomial time by a quantum computer. This is the reason why classical cryptography is said to be broken, although only in a theoretical way, since it has not been possible to build a full quantum computer yet.

Quantum Key Distribution as a mechanism to obtain symmetric key has been developed exponentially in recent years, and has proven to be a secure implementation against quantum computer attacks. Due to this boom, standards have been created for the operation of this scheme. One of them is the one developed by ETSI, which is the one we will be using in this work (ETSI GS QKD).

The *Transport Layer Security* protocol is a cryptographic protocol that provides secure communication over the Internet, being the most widely used among client-server schemes.

The final goal of this work is the complete integration of operational standards that implement quantum key distribution mechanisms into the TLS transport security protocol (in its version 1.3) through the Python library TLSlite, as well as its implementation and deployment in a real network that obtains key from real quantum devices.

Keywords:

Quantum Key Distribution, Quantum Communications, TLS Protocol, Cryptography, ETSI GS QKD, Key Exchange, Post-Quantum Security.

Agradecimientos

En primer lugar, agradecérselo a mi tutor Juan Pedro por introducirme en la investigación estos últimos dos años, siendo una fuente de conocimiento ilimitada en el mundo de las redes y las comunicaciones cuánticas. También quiero agradecer al jefe, Vicente Martín, por darme una oportunidad en el equipo. Además, sin mi compañera Laura Ortiz este trabajo nunca había podido salir adelante, muchas gracias por dedicarme todo el tiempo necesario y más. Por último, agradecer a todos mis compañeros del CCS por aguantarme y guiarme en el mundo de la investigación.

Me gustaría también agradecer a los compañeros de Telefónica Investigación y Desarrollo por introducirme en el mundo de las comunicaciones cuánticas.

Por último, agradecérselo a mi familia y amigos. Sin el apoyo incondicional de mis padres y mi hermano no estaría aquí.

A todo el que lea este trabajo, gracias infinitas.

Tabla de contenidos

Resumen **i**

Abstract **iii**

Agradecimientos **v**

Tabla de contenidos **vii**

Índice De Figuras **ix**

Índice de Siglas **xi**

1. Introducción **1**
1.1. Objetivos 4

2. Estado del Arte **7**
2.1. Redes Definidas por Software 7
2.2. Protocolos de seguridad de transporte 8
2.3. Distribución de clave cuántica 9
2.4. ETSI GS QKD 12
2.5. Mejoras de protocolos clásicos utilizando tecnología *Quantum Resistant* 16

3. Integración de QKD en el protocolo TLS **19**
3.1. TLS 19
3.2. Integración 22
3.2.1. Envío del ClientHello 23
3.2.2. Envío del ServerHello 24
3.2.3. Envío de los Finished 25
3.3. Casos de Uso: HTTPS 26

4. Despliegue en escenario real **29**
4.1. Escenario y despliegue 29
4.2. Evaluación de resultados 31

5. Conclusiones y trabajo futuro **33**
5.1. Trabajo Futuro 35

Bibliografía **40**

Anexo A: Aspectos éticos, económicos, sociales y ambientales **41**

.1. Introducción 41
.2. Descripción de impactos relevantes relacionados con el proyecto 41
.3. Análisis detallado de alguno de los principales impactos 41
.4. Conclusiones 42

Anexo B: Presupuesto económico **43**

Índice de figuras

1.1. Máquina Enigma alemana. Máquina Enigma alemana usada para cifrar comunicaciones durante la Segunda Guerra Mundial. 1
1.2. Despliegue Madrid Quantum Network. Representación de la red de pruebas de Madrid. Cada nodo posee un nombre y el mapa muestra las longitudes y pérdidas por ruido de cada enlace. 4

2.1. Operación del protocolo BB84. Esquema del funcionamiento por fases del protocolo BB84. 9
2.2. Enfoque Asistido. Definición de una aproximación en la cual una fuente de clave cuántica dota de clave simétrica a ambos extremos de la conexión. 12
2.3. ETSI GS QKD. Esquema del alcance y funcionamiento de los estándares utilizados para la generación de la clave QKD. 13
2.4. Estructura Nodo QKD con SDN. 14
2.5. Interfaz de Aplicación ETSI GS QKD 004. Llamadas de la interfaz de aplicación 004 para la obtención de la clave QKD. 15
2.6. Canal QKD entre Alice y Bob con 3 nodos intermedios que efectúan el reenvío de clave. 16

3.1. TLSv1.3 handshake. Protocolo de enlace TLS *handshake* cliente-servidor en la versión TLSv1.3. 20
3.2. QKD-TLS handshake. Operación del QKD-TLS *handshake* propuesto. . 23
3.3. Mensaje *ClientHello*. Captura *Wireshark* del mensaje *ClientHello*, destacando el identificador de aplicación QKD del cliente. 23
3.4. *ServerHello* message. Captura *Wireshark* del mensaje *ServerHello*, destacando el identificador de aplicación QKD del servidor. 24
3.5. Diagrama de Secuencia del Caso de Uso HTTPS. Diagrama de secuencia que muestra el funcionamiento del caso de uso de HTTPS. 26
3.6. Código que ejecuta el cliente HTTPS. 27

4.1. Red Cuántica Definida por Software de Madrid. Existen distintas redes marcadas por distintos colores, siendo la de TID la roja y la de RM la azul. En enlace morado está aún por licitar. El enlace usado es el que une Norte con Quinto. 30
4.2. QKD-TLS *handshake* realizado de manera exitosa. 31

Índice de Siglas

QKD: Quantum Key Distribution

TLS: Transport Secure Layer

SSL: Secure Socket Layer

TRL: Technology Readiness Level

SHA: Secure Hash Algorithm

GCM: Galois-Counter Mode

HTTPS: HyperText Transport Protocol Secure

RSA: Rivest, Shamir and Adleman

PKI: Public Key Infrastructure

ETSI: European Telecommunications Standards Institute

NIST: National Institute of Standards and Technology

BB84: Bennett and Brassard 1984

SDN: Software-Defined Network

YANG: Yet Another Next Generation

MAC: Media Access Control

DH: Diffie-Hellman

ECDHE: Elliptic-Curve Diffie-Hellman

ECDSA: Elliptic-Curve Digital Signature Algorithm

AES: Advanced Encryption Standard

IPsec: Internet Protocol Security

API: Application Programming Interface

KM: Key Manager

ID: Identifier

KMS: Key Management System

ITS: Information Theoretically Secure

KSID: Key Stream ID

TID: Telefónica Investigación y Desarrollo

RM: RediMadrid

LKMS: Local Key Management System

QUAI: Quantum Abstraction Interface

SMTP: Simple Mail Transfer Protocol

Capítulo 1

Introducción

La seguridad en las comunicaciones es la disciplina encargada de prevenir que cualquier entidad no autorizada intercepte la comunicación y pueda acceder de forma inteligible a la información, como se muestra en este trabajo [1]. La criptografía se encarga de cifrar o codificar mensajes para evitar que su contenido pueda ser leído por un tercero no autorizado [1]. Si miramos en el pasado, desde hace miles de años existen técnicas de codificación que han cambiado el curso de la historia, habiendo un punto de inflexión en la Segunda Guerra Mundial, en la que la criptografía fue clave hasta el punto de cambiar el curso de la guerra. Hablamos de la famosa máquina Enigma alemana [2], que podemos ver en la figura 1.1, creada poco antes del comienzo de la Segunda Guerra Mundial, cuyo cifrado ya había sido roto (descifrado) por matemáticos polacos y compartida con los ejércitos inglés y francés pocos meses antes del comienzo de la invasión a Polonia.

Figura 1.1: Máquina Enigma alemana. Máquina Enigma alemana usada para cifrar comunicaciones durante la Segunda Guerra Mundial.

En 1948, la criptografía experimentó un importante desarrollo gracias a Claude Shannon, cuando publicó este artículo [3], en el que se modernizaron las técnicas de codificación para transformarlas en procesos matemáticos avanzados, conduciendo a la criptografía a un punto más cercano al actual.

En el año 1976 se publica en este artículo [4] el protocolo *Diffie-Hellman*, siendo el primer protocolo de cifrado asimétrico, capaz de permitir acordar una clave secreta entre dos máquinas a través de un canal inseguro. Esta clave es imposible de romper, en términos computacionales clásicos, por un atacante aunque obtenga los mensajes enviados por el protocolo. En 1977, los científicos Ronald Rivest, Adi Shamir y Leonard Adleman crearon otro esquema de cifrado asimétrico (RSA [5]) que utiliza la factorización de números enteros. Este esquema se sigue utilizando hoy en día, por lo que ha estado en uso durante más de 40 años para conexiones https, transacciones de bancos, etc.

La criptografía de clave pública o criptografía asimétrica introducida por estos dos últimos trabajos consiste en un sistema que utiliza pares de claves para cifrar y autenticar la información, donde autenticar quiere decir confirmar la identidad del emisor. En estos esquemas existen dos tipos de claves: la clave pública y la clave privada, y cada extremo posee los dos tipos de clave. La clave pública puede distribuirse libremente sin afectar a la seguridad mientras que la privada sólo es conocida por el propietario. La clave privada permite al propietario decodificar mensajes cifrados por la clave pública, de manera que cualquiera puede usar la clave pública para cifrar información pero sólo el propietario de la clave privada es capaz de descifrarla. Este esquema se da entre todo par de sistemas que están intercambiando clave; es decir, en una arquitectura cliente-servidor, el servidor cifra con la clave pública del cliente, que este puede descifrar fácilmente con su clave privada y viceversa.

Estos protocolos de criptografía están basados en operaciones matemáticas complejas con números primos incapaces de ser resueltas mediante la computación clásica en un tiempo razonable (tales como la curva elíptica o la factorización de enteros logarítmicos discretos [5]). Como propone D. E. Denning en este trabajo [6], para romper una clave de 128 bits, los computadores más rápidos del mundo necesitarían trillones de años para encontrar la clave correcta. Además, el mismo artículo [6] indica que esta seguridad se ve altamente amenazada por los avances en la computación cuántica. De hecho, se ha dado una especificación teórica de que un algoritmo cuántico es capaz de resolver estos problemas en tiempo polinomial, rompiendo toda la infraestructura de clave pública (PKI) tal y como la conocemos [7]. Los autores M. Elboukhari, A. Azizi y M. Azizi han demostrado que es posible romper RSA en un tiempo razonable [8].

Como mecanismo de defensa ante esta posible amenaza, denominada comúnmente amenaza cuántica, ha surgido una nueva tecnología conocida como Comunicaciones Cuánticas, que se basa en la generación de una clave simétrica en dos puntos diferentes de la red a través del envío de fotones de luz desde un punto a otro, como explican M. A. Nielsen y I. L. Chuang en [9]. Esto es posible gracias a un par de dispositivos conectados punto a punto (Dispositivos de Criptografía Cuántica) que permiten una transferencia incondicionalmente segura.

La tecnología QKD permite a las dos partes hacer uso de una clave secreta aleatoria para cifrar y descifrar mensajes. Debido a la naturaleza cuántica de la señal, puede asegurarse que sólo es conocida por las partes implicadas en la comunicación. Se po-

dría considerar los dispositivos QKD como un par de fuentes conectadas que generan números aleatorios de manera continua y sincronizada y con la peculiaridad de darse cuenta si se está interceptando la secuencia. Esta capacidad hace que un par QKD (siguiendo los estándares de redes, son normalmente llamados Alice y Bob) sea una tecnología idónea para la generación simétrica de clave en dos puntos separados en el espacio incondicionalmente segura, Information Theoretic Secure (ITS). Esto implica que independientemente de la capacidad computacional y tiempo de un atacante el soporte QKD no se verá afectado, ya que ha cambiado la naturaleza de la señal.

Actualmente, esta tecnología nueva y revolucionaria ha despertado el interés de numerosas compañías y organismos gubernamentales, que están invirtiendo capital y esfuerzo en su implantación y desarrollo. Hay ejemplos de empresas más especializadas (como puede ser IDQuantique, dedicada al cifrado de red cuántica, generadores de números aleatorios cuánticos y sistemas QKD) como generales (como pueden ser Huawei o Toshiba).

Además, existe una gran cantidad de proyectos europeos muy diversos con el objetivo común del desarrollo de estas tecnologías, como el *OpenQKD* [10], el *CiViQ* [11], el *Quantum Flagship* [12] y el *EuroQCI* [13].

Para introducir esta tecnología en las redes de comunicación se ha utilizado en la red donde se ha realizado este trabajo de investigación, el paradigma SDN. Tradicionalmente, tanto los elementos del plano de control como los del plano de datos de una arquitectura de red estaban empaquetados en un código propietario e integrado distribuido por uno o más proveedores. El estándar de código abierto OpenFlow [14], creado en 2008, fue reconocido como la primera arquitectura SDN (Software-Defined Networking) que definió cómo los elementos del plano de control y de datos se separarían y se comunicarían entre sí utilizando el protocolo OpenFlow.

La Madrid Quantum Network comenzó en 2006 [16] y ha ido creciendo y evolucionando a lo largo de los años, contando ahora con 12 nodos distribuídos entre diferentes centros de investigación, empresas y universidades. El medio de transmisión de la información cuántica es a través de fibra óptica, estando los canales cuánticos perfectamente integrados en la infraestructura de telecomunicaciones de Telefónica y RediMadrid, proveedores de servicios de telecomunicaciones.

A diferencia de otras redes, la Madrid Quantum Network es desde su segunda iteración una red SDN. Este enfoque encaja en la flexibilidad que requieren las comunicaciones cuánticas, implementando una interfaz de abstracción que permite la integración de dispositivos QKD de diferentes proveedores en la misma red, haciendola *multi-vendor*, teniendo ahora integrados de manera transparente dispositivos de IdQuantique [15], Huawei (como se muestra en la segunda iteración de la red descrita en el trabajo [17]) y Toshiba, operando con distintos dispositivos, protocolos y tecnologías. Además, posee distintos encriptadores de ADVA [18] y Rhode and Swarz [19] para el cifrado de la información.

Hoy, en la última iteración de esta red [20], la Madrid Quantum Network es una red heterogénea capaz de integrar dispositivos de diferentes marcas comerciales y proveedores, así como diferentes protocolos criptográficos cuánticos y post-cuánticos. Además, los dispositivos utilizados tienen diferentes características: pueden transmitir en variable contínua (utiliza sistemas dimensionales infinitos, por lo que se puede obtener un continuo de resultados de medición) o variable discreta (se tiene

un espacio de Hilbert de dimensión finita, por lo que los resultados de las mediciones provienen de un conjunto finito). También pueden proporcionar clave cruda o destilada [21]. La clave cruda es aquella que se ha generado en cada extremo de la conexión y no ha pasado por ningún proceso de destilación, es decir, el resultado directo de las mediciones. La clave destilada ya ha pasado por estos procesos, consiguiendo una clave simétrica idéntica en ambos puntos de la conexión. En la Figura 1.2 podemos ver el despliegue de la Madrid Quantum Network, con distintas redes (Telefónica en rojo, RediMadrid en azul) con distancias y ruido introducido en el canal.

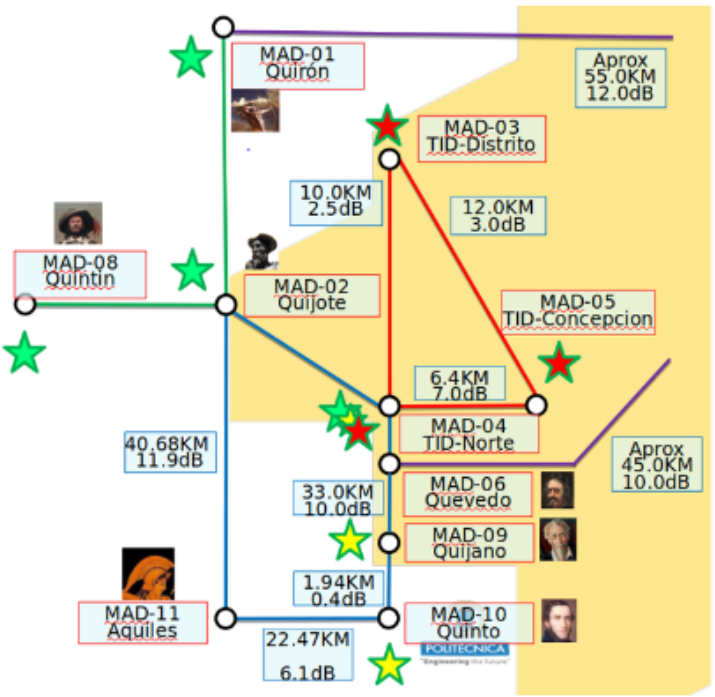

Figura 1.2: Despliegue Madrid Quantum Network. Representación de la red de pruebas de Madrid. Cada nodo posee un nombre y el mapa muestra las longitudes y pérdidas por ruido de cada enlace.

1.1. Objetivos

Lo que se propone en este trabajo es la sobrescritura del protocolo TLS para que en lugar de estar basado en algoritmos de clave pública y privada, aproveche de manera transparente para el usuario las capacidades de las comunicaciones cuánticas. Esto ofrecería al usuario la posibilidad de su utilización de manera trivial sin necesidad de ningún conocimiento adicional, dotando al protocolo de capacidades *quantum-resistant* y haciéndolo aprovechable por diversos tipos de aplicaciones tales como conexiones https y smtp. Por último, como parte de este trabajo, esta infraestructura se ha desplegado en un entorno real de distribución de clave cuántica, la Madrid Quantum Network. La aplicación de la propuesta realizada en este trabajo es en una red real, utilizando dispositivos QKD reales.

Este proyecto propone tres objetivos finales. El primero es la integración transparen-

te de la tecnología QKD en protocolos de seguridad clásica como TLS, de manera que cualquier usuario que haga uso del protocolo TLS para conexiones HTTPS pueda beneficiarse de la seguridad cuántica de manera trivial y transparente. Como segundo objetivo se pretende aumentar la seguridad del protocolo TLS mediante el uso de tecnología QKD. El último objetivo sería el despliegue de la integración en un escenario QKD real, con acceso a dispositivos cuánticos, como es la Madrid Quantum Network.

Capítulo 2

Estado del Arte

En este capítulo se presentan de manera breve ciertos aspectos necesarios para el entendimiento de este proyecto. Se comenzará explicando de manera breve el concepto de redes definidas por software (SDN), después se explicará el funcionamiento de los protocolos de seguridad de transporte (TLS en concreto), a continuación se presentarán los aportes de la tecnología QKD y el protocolo a utilizar, para seguir se dará una breve pincelada a los estándares utilizados y por último se explicarán los protocolos criptográficos barajados y los trabajos anteriores relacionados.

Existe una gran variedad de trabajos previos que buscan la integración de mecanismos de distribución de clave cuántica en el protocolo TLS. La mayoría de ellos se basan en la integración de algoritmos post-quantum en TLS, pero otros de ellos sí se centran en la integración pura de QKD en TLS. En el capítulo3 haremos un estudio detallado de estos, exponiendo lo que ha contribuido cada uno en el resultado final de esta integración.

2.1. Redes Definidas por Software

Se podría considerar SDN como el conjunto de técnicas que permiten la programabilidad a alto nivel de las redes hoy en día, separando los planos de datos y de control de la arquitectura de la red. Esta desvinculación permite a los administradores ajustar dinámicamente el flujo de tráfico en toda la red para satisfacer las necesidades cambiantes que pudiera tener la propia red. Otra característica relevante de las redes SDN es la gestión centralizada de la red, gestión que realizan los llamados controladores SDN, cuya principal labor es mantener una visión global de la red para dar soporte a todas aquellas aplicaciones que tuvieran lugar dentro de esta red SDN.

Los gestores de redes SDN podrían configurar, gestionar y asegurar los recursos de la red rápidamente gracias a otros programas SDN que escribieran ellos mismos ya que no dependen de software propietario. Por último, SDN está basada en estándares e interfaces abiertas y además es independiente del fabricante. Gracias a la implementación basada en protocolos bien conocidos y adaptados por las empresas de telecomunicaciones, se simplifica el diseño y el funcionamiento de la red ya que las instrucciones las proporciona el Controlador SDN en lugar dispositivos y protocolos específicos de cada fabricante.

2.2. Protocolos de seguridad de transporte

La información que viaja de un punto a otro de una red de telecomunicaciones necesita ser segura, por lo que a lo largo de la historia han surgido distintos protocolos de capa de transporte encargados de cifrar la información de un extremo a otro.

En 1995 se desarrolló en este trabajo [22] el protocolo Secure Sockets Layer basado en el cifrado encargado de garantizar la privacidad (esconder los datos transferidos a terceras partes), autenticación (asegurar que todas las partes que intercambian información sean quienes dicen ser) e integridad (verificar que los datos recibidos no han sido alterados ni falsificados) de los datos en las comunicaciones de Internet. Ésto lo consigue cifrando los datos que se transmiten por la web. Actualmente, se considera que este protocolo está obsoleto ya que la última versión re-escrita en este trabajo [23] es la 3.0 y data de 1996.

En 1999, se el protocolo SSL se estandarizó en este trabajo [24], no existiendo grandes diferencias. En el año 2006 en este trabajo [25] se lanza la segunda versión, TLSv1.1, dada la necesidad de cubrir una vulnerabilidad del protocolo frente a ataques CBC. Es en el año 2008 cuando se lanza la versión TLSv1.2 en este trabajo [26], mejorando los algoritmos de cifrado cambiando métodos depreciados y realizando mejoras simples en el *handshake*. Por último, en el año 2018 se lanza en este trabajo [27] la versión TLSv1.3, significando un impulso esencial a nivel de seguridad y en la prevención de posibles ataques como los sufridos en las versiones anteriores. Además, esta última versión acelera el rendimiento del *handshake*. TLS es el protocolo más utilizado en internet para la securización de las comunicaciones (como muestra este informe de Google [28], casi el 95 % de las conexiones de Google usan TLS sobre HTTP). Este protocolo criptográfico proporciona comunicaciones seguras a través de internet, y hace uso de algoritmos de clave pública para la generación de la clave de cifrado simétrico que utilizarán tanto cliente como servidor. Como señala V. Mavroeidis y compañía en este articulo [29], este protocolo se ve altamente amenazado por el auge de la computación cuántica, y no disponemos de ninguna otra alternativa clara para el cifrado de la inmensa mayoría de conexiones en internet.

La integración de TLS con las comunicaciones actuales es trivial, ya que cada extremo de la conexión necesita únicamente una clave con la que cifrar y descifrar la información, y a través de la interacción con TLS cualquier aplicación punto a punto es capaz de recuperarla.

Hemos elegido este protocolo puesto que es el protocolo de capa de transporte más moderno, estable y seguro que existe, además del más usado [28], ya que es utilizado para securizar las comunicaciones de una gran cantidad de aplicaciones.

Como consecuencia de ello, existen muchas implementaciones distintas de este protocolo. En primer lugar están las implementaciones nativas de los sistemas operativos, ya que en los núcleos de todos ellos se implementa este protocolo. Además, existen implementaciones en librerías escritas en distintos lenguajes de programación para que una aplicación desarrollada en un lenguaje específico pueda hacer uso de este protocolo.

En el trabajo [30] se presenta OpenSSL, un proyecto de software libre desarrollado en el lenguaje de programación C que implementa este protocolo. Es una librería muy compleja y muy completa. Paramiko [31] es una implementación en Python del

protocolo SSHv2, que hace uso también del esquema RSA para la generación de una clave simétrica. También existe una librería en Python denominada TLSlite-ng [32], una biblioteca Open Source que implementa SSL y TLS.

Se han analizado los distintos *frameworks* y librerías que cumplían con los requisitos necesitados para el desarrollo correcto del proyecto, pero desafortunadamente no existía ninguna oficial en la versión TLSv1.3, por lo que se ha utilizado una adaptación de un fork del usuario tomato_42 de GitHub en una versión temprana pero completa de la versión TLSv1.3 de la librería TLSlite-ng, que está desarrollada en Python.

2.3. Distribución de clave cuántica

Como se muestra en el artículo [33], el protocolo BB84 es uno de los tantos protocolos que implementa la distribución de clave cuántica, un método de comunicación seguro que permite que dos partes separadas en el espacio compartan una clave secreta aleatoria simétrica que solo ellos conocen, para posteriormente cifrar las comunicaciones entre ellas [34].

En primer lugar, se va a describir una explicación sencilla del funcionamiento del protocolo BB84. Como se detalla en este trabajo [35], el BB84 es el primer esquema de distribución de clave cuántica. La seguridad de este protocolo radica en las comunicaciones cuánticas, más concretamente en el hecho de que los dos conjuntos de polarización son indistinguibles si se codifican en una sola partícula de luz [35], asegurando que si un intruso intenta interceptar la comunicación creará errores en el canal, notificando así a los usuarios legítimos.

Para realizar una explicación más detallada del funcionamiento del esquema, se va a proceder a una explicación mas detallada de la operación del protocolo, dividiéndolo para ello en cuatro fases: preparación aleatoria, transmisión, medida aleatoria y reconciliación. En la Figura 2.1 se muestra la ejecución del protocolo dividida en estas fases.

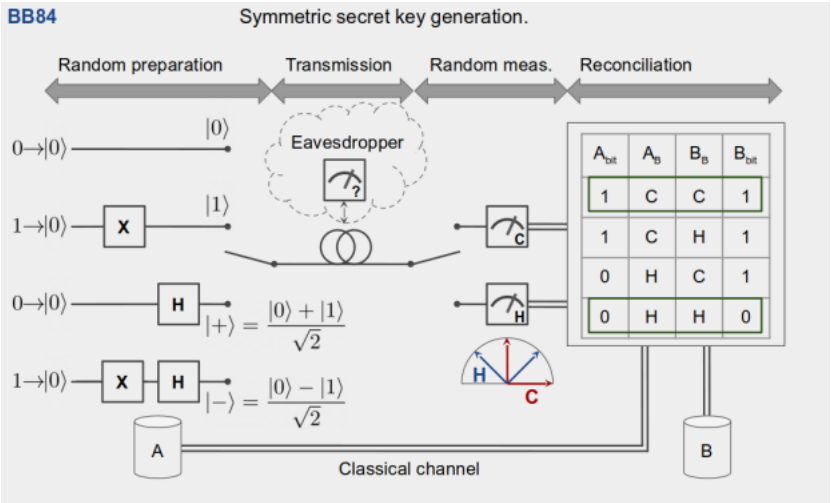

Figura 2.1: Operación del protocolo BB84. Esquema del funcionamiento por fases del protocolo BB84.

Para ilustrar la explicación y para ayudar a la correcta comprensión, se proporciona un ejemplo de funcionamiento del protocolo. Antes de comenzar, vamos a explicar el concepto de qubit.

Como se explica en el trabajo [9], un qubit o bit cuántico es un sistema cuántico con dos estados propios y constituye la unidad básica de información en la computación cuántica. Un bit clásico puede representar un único valor binario (0 o 1). Sin embargo, un qubit puede representar un 0, un 1 o cualquier proporción de 0 y 1 en la superposición de ambos estados. Hay muchas implementaciones físicas del qubit. En este trabajo, el qubit está representado por dos fotones individuales preparados con una polarización determinada.

1. *Preparación aleatoria*

   Cada polarización tiene dos posibles estados respecto a una base, los dos estados son 1 y 0 (relajado y excitado). Respecto a la otra base son las superposición de los estados 1 y 0. De esta forma, si no se conoce la base en la que se ha preparado el estado no se puede conocer el estado; es decir, hay que medir en la misma base en la que se prepara para asegurar la corrección de la medida (es muy importante que los estados que se quieran distinguir sean ortogonales [9]).

   En esta fase, el emisor prepara una secuencia de qubits [36], eligiendo la base en la que se preparan de manera aleatoria y guardando tanto el estado del qubit transmitido como la secuencia de bases en las que han sido preparados, sin ser ésta transmitida aún.

   Se va a ilustrar esta preparación aleatoria con un ejemplo: el emisor, Alice, prepara una ristra de qubits para enviar al receptor, Bob. Estos qubits están preparados en una base que puede ser C, o puede ser H, y únicamente el emisor conoce las bases elegidas. Todavía no han sido transmitidas al receptor.

2. *Transmisión*

   Esta fase es la encargada de transmitir los qubits a través del canal cuántico a la otra parte. Como se muestra en el trabajo [9], debido a los principios básicos de la mecánica cuántica, esta información no puede ser interceptada ni duplicada, ya que cuando se lee no se tiene información suficiente para recrear el estado debido al principio de no clonación. En esta fase se transmiten los qubits preparados en una base aleatoria desde el emisor hasta el receptor.

   En sintonía con el anterior ejemplo; el emisor, Alice, envía la secuencia de qubits al receptor, Bob.

3. *Medida Aleatoria*

   En este momento, el receptor ha recibido una secuencia de qubits pero no sabe en qué base se han preparado. Es aquí donde el receptor elige, también de manera aleatoria, las bases con las que va a medir el qubit, y efectúa la medición. Esta medición es almacenada en una tabla junto a qué base se ha utilizado para efectuar dicha medida. Es importante destacar que la tabla de la figura 2.1 consiste en bits; es decir, información clásica (no se necesita memoria cuántica) porque ya se han realizado las mediciones, transformando el qubit en bit.

   Continuando con el ejemplo, esta fase es el momento en el cual el receptor, Bob, realiza las mediciones sobre la secuencia de qubits recibida del emisor, Alice.

Estas mediciones se harán de forma aleatoria, ya que Bob no sabe qué medidas ha utilizado Alice. Además de apuntar el resultado medido, Bob se encarga de apuntar la base con la que ha realizado la medición asociada al resultado. Nótese que las mediciones aquí pueden efectuarse de manera errónea si se elige de manera equívoca otra base. Como se muestra en la Figura 2.1: Alice prepara un 0 con una base H, y Bob al realizar la medida con una base C lee un 1, lo que resultaría en un error.

4. *Reconciliación*

   Durante este proceso, la tabla de resultados de Bob es distribuida (por medio de envío a través de canales clásicos, por ello era necesario un canal clásico autenticado) para establecer cualquier comunicación cuántica, como podemos deducir del trabajo [37, 41]. Durante el proceso de reconciliación, ambas partes recorren la tabla y desechan todos los bits que no procedan de una medida con la misma base. Como podemos observar en la figura 2.1 que ilustra el ejemplo seguido, sólo son válidos el primer y último qubit, ya que son los que se han medido con la misma base. El resto se desechan porque no podemos saber en qué estado se preparó.

   Cuando termina la reconciliación, tanto emisor como receptor han desechado las medidas no válidas y disponen de la misma información en ambos puntos. Además, como se explica en [38] es teóricamente imposible que un tercero haya interceptado la transmisión (debido a las propiedades de no clonación de la mecánica cuántica), además de ser imposible la reconciliación completa por parte de un tercero.

   Del trabajo [9] se deduce que sólo habría dos maneras en las que un tercero podría interceptar la comunicación. La primera consiste en copiar el qubit para luego reenviarlo, lo cuál no es posible debido al principio de no clonación de la mecánica cuántica. La segunda manera sería leyendo y reenviando los qubits que le han llegado; es decir, transformar los qubits leídos en bits a través de la medición; y después preparar otra serie de qubits en bases aleatorias para ser reenviadas. Esto tampoco es posible, puesto que el atacante no conoce la secuencia de bases que ha enviado Alice a Bob en primer lugar, por lo que debería usar una secuencia de bases aleatorias, que resultaría en un aumento en la tasa de error en el receptor, que sería capaz de saber que la comunicación ha sido alterada.

   Es en este momento cuando podemos asegurar que tanto Alice como Bob disponen de una secuencia simétrica de bits de información distribuida de manera cuántica, lo que implica la imposibilidad de que un tercero haya interceptado la comunicación.

Una vez comprendido el funcionamiento del BB84, vamos a dar unas pinceladas de cómo es el funcionamiento actual de las tecnologías de distribución de clave cuántica. El esquema más utilizado en la actualidad es el que denominamos **Enfoque Asistido**, que consiste en dotar tanto a emisor como a receptor de una fuente de clave cuántica capaz de aprovisionar a ambas partes con la misma clave simétrica, como se muestra en la figura 2.2:

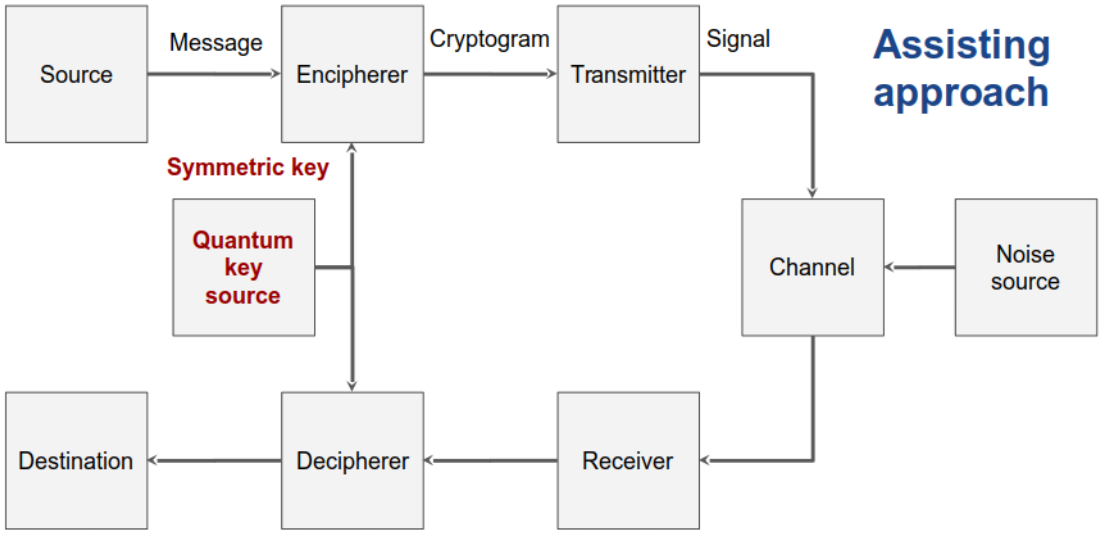

Figura 2.2: Enfoque Asistido. Definición de una aproximación en la cual una fuente de clave cuántica dota de clave simétrica a ambos extremos de la conexión.

Podría implementarse una fuente de clave cuántica simulada que recupere clave de los dispositivos QKD de emisor y receptor, pero en lugar de ello hemos utilizado dispositivos cuánticos reales a los que accedemos a través del ETSI GS QKD 004 para hacer las peticiones y hacer la solución más escalable. En la siguiente subsección, 2.4, se hará una explicación detallada del funcionamiento de este estándar.

2.4. ETSI GS QKD

Existen diferentes agencias de estandarización cuyo objetivo es la adaptación de procesos a un modelo a considerar de referencia con el objetivo de perseguir la automatización de la resolución de problemas. Existen distintas agencias como la NIST o la ETSI.

En el mundo de las comunicaciones cuánticas, la agencia de estandarización ETSI ha estado trabajando en la creación de estándares de aplicación y control para la distribución de clave cuántica. En este trabajo se hará uso del estándar de la interfaz de control para redes definidas por software, definidas en el trabajo [39], ya que el escenario entero está pensado para ser ejecutado en redes que cumplan esta cualidad (para poder ser en un futuro todavía más escalables) y del estándar de la especificación de una API entre para recuperar las claves, definido en el trabajo [40].

Como se explica en [9], la distribución de clave cuántica se basa en la creación, transmisión y detección de señales a nivel cuántico, lo cuál es difícil de lograr si la red también está en uso con señal clásica. Por otro lado, la transmisión cuántica no se puede amplificar ni regenerar sin repetidores cuánticos (no viables con la tecnología actual). Para optimizar la transmisión de señales cuánticas junto con comunicaciones clásicas, es necesario integrar los sistemas QKD de tal manera que puedan comunicarse con el control de la red y recibir comandos de éste. Estos sistemas QKD conocedores de la red (*network-aware*) deben integrarse en la capa física, pero también de manera

lógica con la arquitectura de gestión, y para conseguirlo se deben describir las capacidades de los dispositivos QKD al controlador de la red, siendo YANG el principal lenguaje de modelado usado para describir elementos de red, tal y como describen los trabajos [37, 41].

Este trabajo hace uso de dos de los estándares del ETSI: el ETSI GS QKD 015 [39] y el ETSI GS QKD 004 [9]. En la Figura 2.3 podemos ver de manera ilustrada el alcance de éstos.

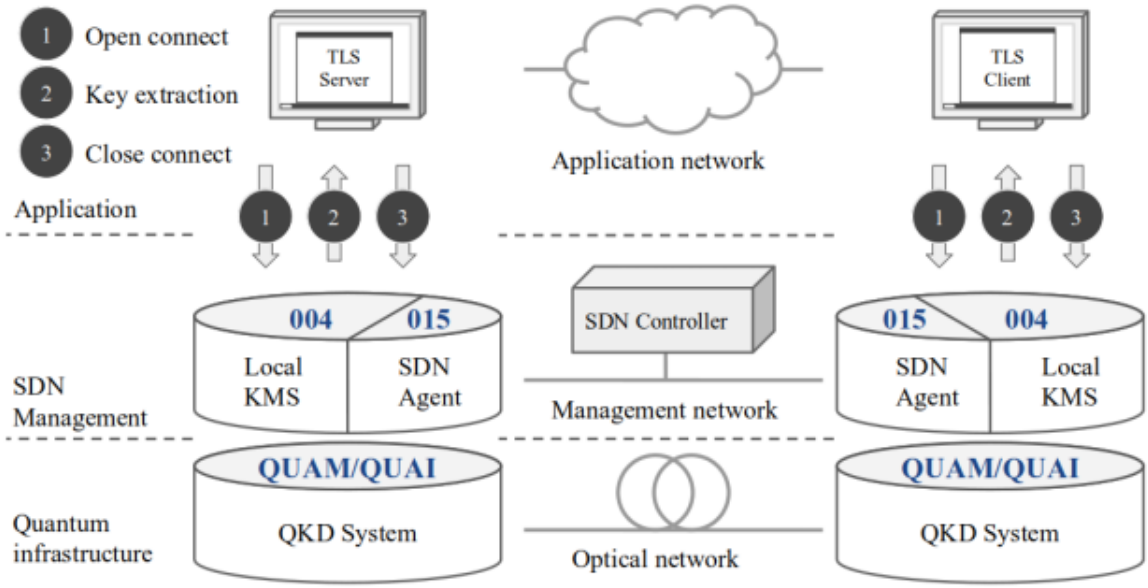

Figura 2.3: ETSI GS QKD. Esquema del alcance y funcionamiento de los estándares utilizados para la generación de la clave QKD.

A continuación se describirá una visión a alto nivel de los componentes y de los estos estándares mencionados.

El **controlador** QKD es el "cerebro" de la arquitectura SDN, encargándose de la gestión y estado de la red así como de la distribución de este estado. Suele contar con un sistema de cálculo de rutas extremadamente útil y necesario, pues debe ser capaz de establecer directamente un servicio a nivel de aplicación gracias al conocimiento de la topología de la red.

El **Nodo QKD** es la representación abstracta a alto nivel que componen todos los elementos que conforman un nodo en la Madrid Quantum Network (Agente, LKMS, QKDs y Aplicaciones). Toda la información de estos elementos es necesario gestionarlo de alguna manera y dado que se trabaja en un entorno SDN, lo más apropiado sería un modelo de red, el modelo ETSI GS QKD 015 en este caso. La figura 2.4 muestra la arquitectura de red SDN junto a la tecnología QKD que ha utilizado este proyecto.

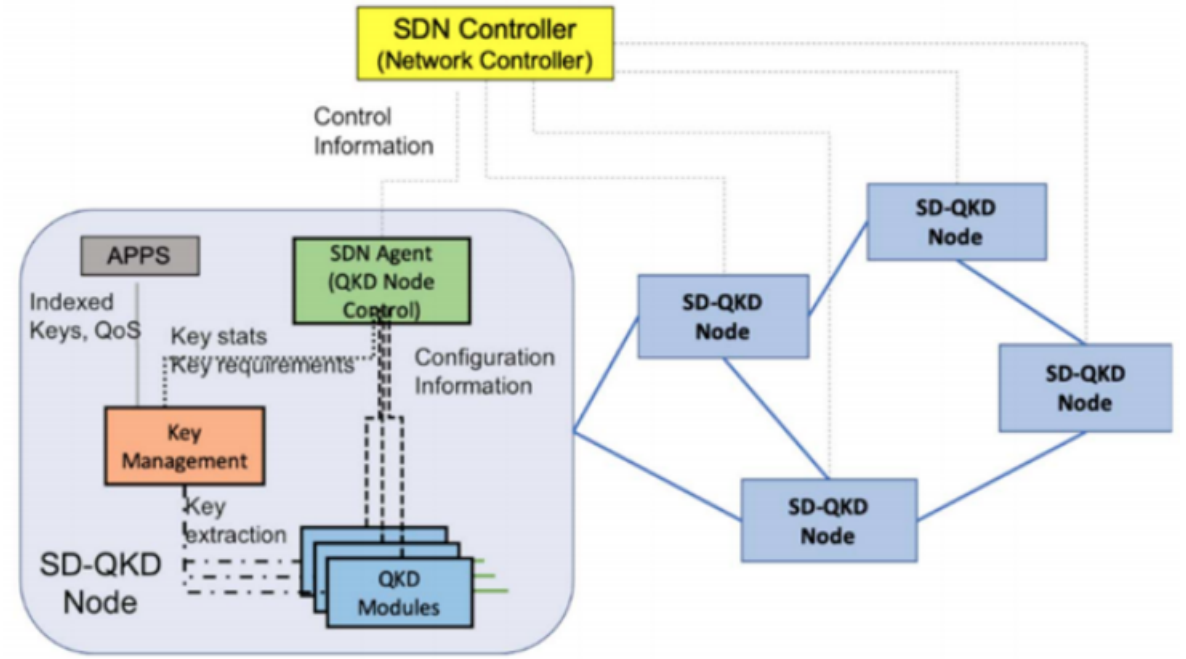

Figura 2.4: Estructura Nodo QKD con SDN.

Se van a resumir los campos del modelo **ETSI GS QKD 015** presentados en [39], cuyo objetivo es la simplificación de la gestión de los recursos QKD mediante una capa de abstracción a alto nivel descrita en YANG, optimizando la creación y uso de claves QKD. Hay 4 grupos bien diferenciados:

- Las **capacidades** del Nodo QKD. La primera capacidad es *link_stats_support* y permite extraer información estadística de los enlaces. La segunda capacidad *applications_stats_support* permite extraer información de las aplicaciones en ejecución en el Nodo QKD. La última es *key_relay_mode_enable*, y determina que el Nodo QKD es capaz de funcionar realizando retransmisión de clave permitiendo que dos Nodos QKD que no están directamente enlazados tuvieran al final exactamente la misma clave.
- Las **aplicaciones** en este modelo contienen la información relevante para que un par de apps interconectadas en cada extremo de la red comiencen un servicio de extracción de clave simetrica.
- Las **interfaces** representan a un alto nivel de abstracción los dispositivos QKD alojados en el Nodo QKD. Contienen la información crucial y justa para ser controlados a alto nivel desde el Controlador SDN.
- Los **enlaces** son un par de Interfaces interconectadas entre sí.

El **Agente** de una arquitectura SDN es el intermediario entre el Nodo QKD y el Controlador. Su función es facilitar alguna de las tareas del Controlador haciéndose cargo de la parte de la red que le corresponde. También se relaciona con el LKMS para las peticiones de clave.

El **LKMS** es el encargado de almacenar todas las claves generadas por los diferentes Módulos QKD alojados en los Nodos QKD. Es un componente crucial en la arqui-

tectura pues será el encargado de suministrar clave a todas las aplicaciones que lo soliciten. Su funcionalidad final es gestionar y manejar claves seguras generadas a través de un protocolo QKD y entregar bajo demanda conjuntos de claves idénticos a las aplicaciones que los solicitan en los extremos donde estén conectadas.

Los **módulos QKD** son dispositivos que funcionan de manera entrelazada, de forma que en un punto de la red se encuentra un QKD transmisor y en optro punto un QKD receptor. La principal labor de estros dispositivos es suministrar clave al LKMS que le esté solicitando clave de manera que quede clave preparada para el momento en que una aplicación lo solicite.

Las **Aplicaciones** serán las principales beneficiadas en toda esta arquitectura, puesto que extraerán clave simétrica en ambos puntos de la red para dar comienzo un servicio de aplicación tradicional. Par que estas aplicaciones extraigan clave del LKMS se suele hacer usando algún tipo de API de manera que la comunicación con el LKMS sea lo más estándar posible. Actualmente hay dos APIs posibles, y nosotros hemos elegido la ETSI GS QKD 004.

Por último, vamos a introducir el estándar **ETSI GS QKD 004** presentado en el trabajo [40]. Esta API pretende especificar cómo las aplicaciones deberían pedir clave a su gestor de clave local, por lo que es utilizada para recuperar la clave QKD final desde las aplicaciones que lo requieran, como esta versión de TLS. Básicamente podemos resumir esta API en tres llamadas principales, que son mostradas en la Figura 2.5.

```
Interface QKD(
    OPEN_CONNECT (in source, in destination, inout QOS, inout Key_stream_ID, out status);
    GET_KEY (in Key_stream_ID, inout index, out Key_buffer, inout Metadata, out status);
    CLOSE (in Key_stream_ID, out status);
)
```

Figura 2.5: Interfaz de Aplicación ETSI GS QKD 004. Llamadas de la interfaz de aplicación 004 para la obtención de la clave QKD.

1. ***open_connect***: reserva un identificador único de servicio para extraer clave a ambos lados de un enlace QKD. En esta llamada se pueden enviar ciertos parámetros a modo de calidad de servicio. Como vemos en la Figura 2.5, esta llamada requiere dos parámetros de entrada que son el identificador de la aplicación QKD cliente y servidor. Estos identificadores son requisito principal de la API y son muy importantes ya que sirven como identificadores únicos del enlace QKD. Esta llamada se encarga de transmitir al LKMS la necesidad de creación de un enlace QKD, que a su vez se transmite al Agente del nodo QKD y este lo reenvía al controlador de la red. En este momento, cuando el controlador ha recibido la petición de creación de aplicación por los dos extremos de la conexión, éste se encarga de evaluar el mejor camino entre ambos puntos de la conexión. Además, configura los nodos inicial y final, así como todos los intermedios para hacer el reenvío de clave (*key relay*) a través de ellos hasta que llega a ambos extremos, como se muestra en la figura 2.6.

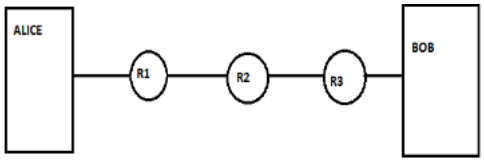

Figura 2.6: Canal QKD entre Alice y Bob con 3 nodos intermedios que efectúan el reenvío de clave.

2. ***get_key***: a través del identificador reservado en la llamada anterior (denominado Key Stream ID), comienza a extraer clave del LKMS local de manera sincronizada, de manera que se consigue una clave QKD simétrica totalmente segura a ambos extremos de la conexión.
3. ***close***: termina el servicio de extracción de clave, el LKMS dejará de entregar clave a aquellas aplicaciones que estén extrayendo clave a través del identificador único de las llamadas anteriores.

Por último, vamos a describir de manera breve el funcionamiento completo de un caso de uso en el cual una aplicación efectúa una petición de clave simétrica. En primer lugar, la aplicación efectúa una llamada *get_key* con el identificador previamente recuperado (a través de la llamada *open_connect*). El LKMS recibe la petición y se comunica con el Agente, que transmite la petición al Controlador QKD (y lo mismo ocurre en el otro extremo). Cuando ambas peticiones llegan al controlador, este evalua el mejor camino entre ellos, reconfigurando los nodos intermedioos si es necesario para efectuar el key relay, liberando la clave.

2.5. Mejoras de protocolos clásicos utilizando tecnología *Quantum Resistant*

La integración de QKD en protocolos de seguridad en Internet es una de las líneas más activas en los últimos años, existe una gran cantidad de trabajos que pretenden la completa integración de algoritmos de distribución de clave cuántica basados en el BB84 en el protocolo de seguridad SSL y su versión concluyente TLS [42, 43, 44, 45, 46, 47, 48, 49, 50]. En esta sección van a ser nombrados los principales; así como las principales aportaciones que han sumado a este trabajo o las principales diferencias que se han efectúado para una correcta integración.

El primero de ellos salió en 2007 [42], viéndose mejorado en una versión posterior de 2008 [43]. Este trabajo define de manera teórica una integración muy trabajada del protocolo QKD como un intercambio de clave para SSL, siendo posiblemente la versión más completa y práctica de todas (aún siendo la más antigua). Se define también un handshake *quantumSSL*, que hace uso de una implementación simple del protocolo BB84 para generar las claves de escritura y el *pre_master_secret*. Estos trabajos suponen un gran avance en esta integración, y han servido de guía para el trabajo descrito en este documento. Las principales diferencias que hemos planteado con respecto este trabajo son cuatro.

La primera es el uso del protocolo TLS en lugar de SSL, ya que éste es un protocolo más estable, moderno y maduro. También se decidió la definición de un nuevo

intercambio de clave en el TLS *handshake*, de manera que si se encuentra una extensión específica en los mensajes *Hello*, el intercambio de clave no se hará a través de algoritmos de clave pública si no a través de QKD; es decir, utilizando unas claves simétricas que han sido transmitidas de una manera totalmente segura. Estos dos trabajos no hacen uso de ningún estándar para la implementación de QKD (lo que lo hace difícilmente escalable), sino que el más sofisticado de ellos reescribe el protocolo SSL entero en C para empotrar una implementación del protocolo BB84 dentro. Por último, con el planteamiento de este trabajo no es necesaria la creación de un nuevo componente QKD y una decena de mensajes asociados a éste para el protocolo, como se ha hecho en estos trabajos [43, 44, 45], sino que añadiendo extensiones en los mensajes ya existentes se consigue un funcionamiento completo y corecto con menos latencia (ya que no hacen falta más message round-trips que los necesitados por TLS, incluso se necesitan menos) y menos almacenamiento y cómputo requerido. Esta es una mejora bastante importante, ya que se adapta con mínimos cambios ambas tecnologías, facilitando su adopción y escalada.

A la vez que este trabajo muy completo, surgen otros trabajos que introducen QKD y mencionan sin llevar a cabo una implementación que podría hacer uso de claves QKD en los protocolos TLS [43, 50] e Ipsec [51, 52], pero las integraciones descritas son puramente teóricas.

Después surgen trabajos teóricos como [44, 45, 46] que explican que la integración de la distribución de clave cuántica basada en el BB84 dentro de TLS en un escenario puramente teórico sería de gran utilidad, basándose en el problema de seguridad de los algoritmos de PKI sobre los que se sostenta TLS, pero ninguno define cómo debe darse la implementación ni mucho menos un escenario (ni siquiera teórico) para las pruebas.

Existen trabajos que integran un nuevo componente en TLS como [43, 44, 46], usado para aislar toda la operación de QKD (sin aportar seguridad extra), pero se ha considerado no decantarse por esta opción, puesto que se ha entendido posible y viable la correcta integración sin la creación de un compontente exclusivamente dedicado a QKD dentro de TLS, que supondría aumentos en la latencia (debido a las message round-trips que serían necesarias para la inicialización y operación de QKD) y en la complejidad y almacenamiento del protocolo, puesto que sería necesario el almacenamiento de este componente y de los mensajes en TLS (aumentando la latencia descrita en [47]).

Las propuestas definidas en los trabajos [44, 45, 46] son muy simples y no definen prácticamente la integración como los anteriores, y el punto fuerte de este trabajo es que la integración queda completamente definida y embedida, de manera que el uso de QKD dentro de TLS para otros servicios (como HTTPS) es trivial y transparente (como se va a demostrar más adelante). Una vez se han presentado todos los trabajos relacionados, vamos a dar paso a la descripción de la metodología para la integración de un nuevo intercambio de clave basado en QKD en TLS.

Capítulo 3

Integración de QKD en el protocolo TLS

En este capítulo se va a definir de manera detallada la correcta integración de un esquema que implementa algoritmos de distribución de clave cuántica en el protocolo de seguridad TLS en su versión 1.3. Se va a hacer especial hincapié en el funcionamiento del nuevo esquema TLS utilizando claves distribuidas de manera cuántica para el cifrado.

Si echamos un vistazo a los trabajos anteriores, varios de los trabajos vistos en la sección 2.5, como son [42, 43] giran al rededor de un mismo lenguaje de programación, C, para incluir el protocolo QKD directamente en la librería (en este caso de SSL, debido a la antigüedad del trabajo), pero desde un punto de vista práctico y experimental hemos optado por una tecnología más moderna, y no tan a bajo nivel, más conveniente para las pruebas de concepto iniciales y un rápido prototipado, y en un futuro se discutirá su migración a otras tecnologías más a bajo nivel.

Además, este trabajo utiliza la última versión del protocolo TLS, la versión 1.3. Esta versión fue lanzada en agosto de 2018 [27], por lo cual no existen muchas implementaciones para esta versión en las librerías adaptadas para los diversos lenguajes de programación. De hecho, cuando comenzó este trabajo no existía una librería oficial de TLS con la versión 1.3. Existe una librería en Python denominada *TLSlite-ng* [32], una biblioteca Open Source que implementa SSL y TLS, que no disponía de la versión TLSv1.3 al inicio de este estudio. Es por eso que se utiliza una versión no oficial de esta librería, ya que surge un trabajo en GitHub de una persona independiente que implementa la versión 1.3 de este handshake incluida en la librería TLSlite.

3.1. TLS

En primer lugar, se va a describir de manera detallada el funcionamiento del TLS *handshake* en su versión TLSv1.3 para una mejor comprensión de la integración. El TLS *handshake* es un protocolo que sirve para que dos extremos se verifiquen entre sí y puedan establecer un intercambio de claves para el tráfico cifrado. Es una negociación para el consenso acerca de los parámetros de la sesión, y se negocian variables

como la versión que se va a utilizar, los *ciphersuites* o conjuntos de cifrado (conjunto de algoritmos que aseguran una conexión de red), la necesidad de autenticar la identidad de ambas partes mediante algoritmos de PKI o la generación de claves de sesión para poder utilizar cifrado simétrico una vez finalizado el handshake.

Como se mencionó en el estado del arte de TLS, este protocolo de transporte se ha visto enormemente mejorado en la versión 1.3 de TLS [27], ya que se han reducido el número de *message round-trips* entre cliente y servidor disminuyendo el cómputo por ambas partes, además de eliminar ciphersuites inseguros y añadir extensiones en los mensajes TLS, lo cual resultará realmente útil para la integración de QKD en el *handshake*. En la siguiente figura tenemos un ejemplo de *handshake* TLS en la versión 1.3.

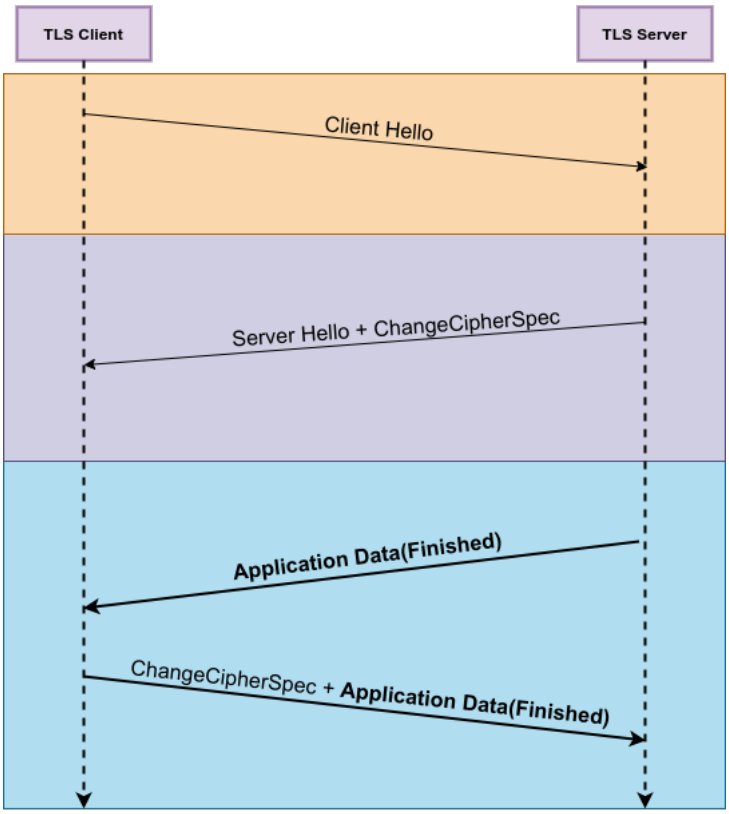

Figura 3.1: TLSv1.3 handshake. Protocolo de enlace TLS *handshake* cliente-servidor en la versión TLSv1.3.

Como ayuda en la explicación, vamos a dividir el protocolo de *handshake* de TLS en tres fases: la primera (resaltada en la Figura 3.1 en color salmón) corresponde con del envío del primer mensaje por parte del cliente al servidor, a continuación llega la segunda fase (resaltada en la Figura 3.1 en color gris) con el envío de la primera respuesta por parte del servidor y, para finalizar, se iniciaría la última fase (resaltada en la Figura 3.1 en color azul) en la que tiene lugar los envíos del mensaje final por ambas partes, indicando si el protocolo de iniciación ha sido exitoso o erróneo.

1. Envío del *ClientHello*

   En esta fase se va a definir el funcionamiento del protocolo en el envío del men-

saje *ClientHello* desde el cliente al servidor. Este mensaje es el primero de toda la conexión, y todos los TLS *handshake* deben comenzar de esta manera. El mensaje *ClientHello* almacena información necesaria para el correcto funcionamiento del enlace:

- Listado de *ciphersuites* soportados por el cliente.
- Listado de versiones de protocolo TLS soportadas por el cliente.
- Listado de claves públicas del cliente, indicando el algoritmo de intercambio de clave a usar.
- *Client random data*: una serie de bits aleatorios usados para derivar el secreto compartido.

2. Envío del *ServerHello*

   En esta fase se va a definir el funcionamiento del protocolo en el recibimiento del mensaje *ClientHello* por parte del servidor y en el consecuente envío del *ServerHello* junto al mensaje *ChangeCipherSpec* vacío. Este mensaje en versiones anteriores indicaba que la comunicación pasa de ser plana a cifrada, aunque en esta versión se utiliza para un modo específico que ayuda a disfrazar la sesión como TLSv1.2.

   Cuando un mensaje *ClientHello* llega a un servidor TLS, el servidor debe escoger la versión TLS, el algoritmo de intercambio de clave y el *ciphersuite* de entre los ofertados por el cliente. Si en alguno de los campos no le concuerda ninguna opción, el protocolo termina aquí de manera errónea puesto que el *handshake* podría ser inseguro. Si por el contrario sí es capaz de elegir un parámetro en todos los campos, el servidor envía el conjunto de cifrado seleccionado, el método de intercambio de clave deseado, su clave pública para el intercambio de clave, la versión de TLS concordada y el server random data empotrados dentro del mensaje *ServerHello*.

3. Envío de los *Finished*

   Aquí tiene lugar la mayoría de los cómputos realizados por el protocolo. Cuando el *ServerHello* llega al cliente, ambas partes ya han negociado todos los parámetros del protocolo de enlace y son capaces de inicializar los algoritmos de intercambio de clave basados en clave pública para conseguir una clave simétrica extremo a extremo. Para ello, llevan a cabo una serie de cálculos que resultan en la obtención de un secreto compartido.

   Cuando el servidor obtiene este secreto, envía un mensaje *Finished* con datos de verificación construidos a través de la generación de un hash de todos los mensajes del *handshake* y el secreto compartido. Este mensaje va envuelto dentro de un mensaje *ApplicationData*. Cuando el cliente obtiene este secreto, tan solo le queda preparar el *Finished* con los mismos datos de verificación y su secreto.

   El primero en enviar el *Finished* es el servidor, que el cliente verificará y sólo entonces se producirá el envío del *Finished* por parte del cliente. Cuando ambas partes han verificado correctamente el mensaje *Finished*, comienza la sesión y todos los datos a partir de aquí están cifrados. Cuando hemos llegado a este punto, TLS crea una sesión en cada parte con las claves simétricas derivadas

del secreto compartido para poder ser utilizadas posteriormente, y el protocolo de enlace ha acabado de manera satisfactoria.

3.2. Integración

Para el correcto funcionamiento de esta implementación, es necesario que cada una de las partes implicadas en el esquema propuesto pertenezcan a una red QKD. Para ello, tiene que existir en cada extremo de la conexión la estructura de un nodo QKD mencionada en el capítulo 2.3. Esto significa que tanto en el extremo cliente como en el servidor debe existir un módulo QKD, un LKMS, un Agente y, en alguna parte de la red, un controlador QKD capaz de orquestar la distribución de la clave cuántica.

La integración de QKD en el TLS *handshake* se ha pensado de manera exhaustiva. Había muchas decisiones de diseño por tomar, dado que el abanico de opciones a atacar para lograr la integración era muy amplio. Se ha estudiado cada uno de los trabajos anteriores relacionados para asegurar que la opción escogida era la más acorde a los requisitos que se debían cumplir.

La idea se sustenta en dos pilares principales. El primero es la creación de un nuevo método de intercambio de clave de entre los soportados por la versión TLSv1.3 del protocolo descrita en el trabajo [27] (entre los que están DH, ECDHE, ECDSA, etc.): QKD-TLS-AES-256-GCM-SHA384, que será el encargado del intercambio de la clave QKD entre ambas partes para obtener el mismo secreto a usar como clave simétrica. Se ha definido este conjunto de cifrado para implementar el nuevo intercambio de clave. Este conjunto de cifrado indica que se va a realizar un intercambio de clave QKD a través de TLS, utilizando el algoritmo de cifrado por bloques AES (descrito en el trabajo [53]) con una clave de 256bits en el modo de operación *Galois/Counter Mode* (un modo de operación para criptografía de clave simétrica descrito en [54]) y se utilizará SHA (una función hash descrita en [55] que produce una salida resumen única para cada entrada) con una clave de 384 bits como tecnología de *hashing*.

El segundo pilar consiste en la creación de una extensión de mensaje TLS *QkdApplicationId* a incluir dentro de los mensajes *ClientHello* y *ServerHello*, que servirá como identificador de aplicación QKD para que TLS sea capaz de hacer uso de la interfaz que implementa el estándar de aplicación ETSI GS QKD 004 para hacer la petición de clave.

Para la mejor comprensión de la integración, vamos a apoyarnos en la división del protocolo TLS que ha surgido del apartado anterior, explicando detalladamente en cada fase qué cambios han sido necesarios para la correcta operación de la implementación final.

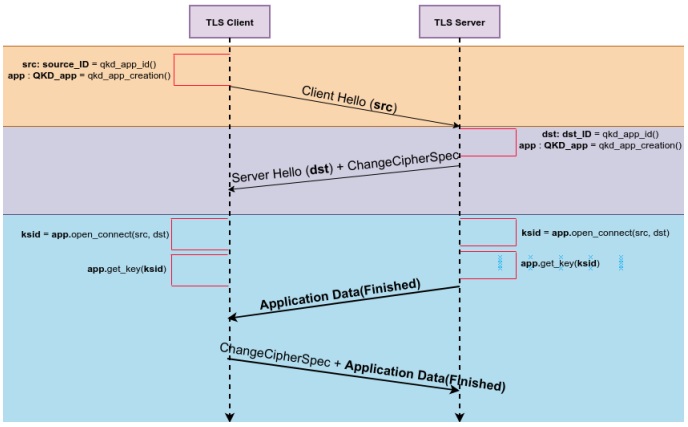

Figura 3.2: QKD-TLS handshake. Operación del QKD-TLS *handshake* propuesto.

Ahora, se va a describir de manera exhaustiva cada una de las fases del protocolo TLS-QKD, que se han hecho coincidir con las del protocolo TLSv1.3, como se ilustra en la Figura 3.2.

3.2.1. Envío del ClientHello

Cuando el cliente está negociando una conexión TLS con un servidor, se deben establecer unos pasos de configuración previos para el funcionamiento adecuado del protocolo de enlace.

En primer lugar, el cliente debe generar un identificador de la aplicación QKD que será ejecutada en su lado, ya que este ID será necesario por ambas partes para realizar la conexión entre las aplicaciones. Este identificador va empotrado dentro del mensaje ClientHello a través de la creación de la extensión de mensaje TLS *QkdApplicationId*, como podemos observar en la Figura 3.3.

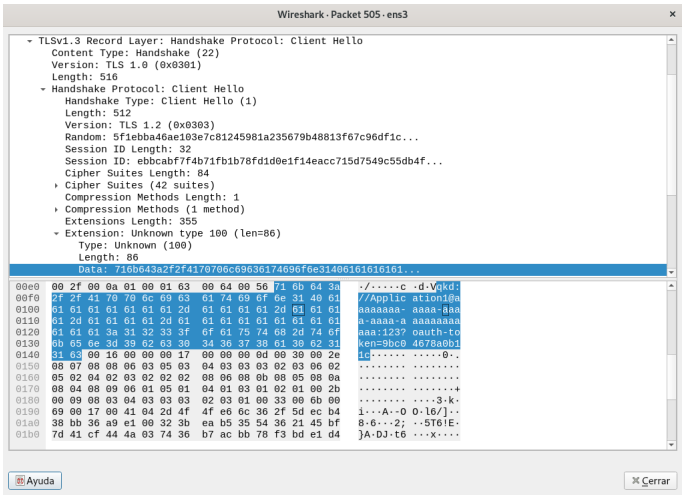

Figura 3.3: Mensaje *ClientHello*. Captura *Wireshark* del mensaje *ClientHello*, destacando el identificador de aplicación QKD del cliente.

La figura 3.3 representa el envío del mensaje *ClientHello* de cliente a servidor, y podemos apreciar destacada en azul cómo la extensión de mensaje TLS *QkdApplicationId* es empotrada en el mensaje con el identificador de la aplicación QKD cliente. Dado que el núcleo de TLS no se ha cambiado, el protocolo no reconoce el tipo de extensión como tal sino como *Unknown*.

El siguiente paso consiste en crear la aplicación QKD cliente para realizar la conexión y pedir clave QKD. El último paso de esta fase es la creación del mensaje *ClientHello* con la versión TLS soportada a 1.3, los *ciphersuites* soportados con QKD-TLS-AES-256-GCM-SHA384 como principal. Este es el método de intercambio de clave que se ha propuesto, que utiliza TLS con QKD, usando AES 256 para el cifrado en el *Galois-Counter Mode*, usando un algoritmo de SHA384 para *hashing*. El algoritmo de cifrado por bloques AES en el modo de operación GCM cifra con la clave cuántica cada mensaje de la conexión. Además, el algoritmo SHA384 se encarga de crear una salida resumen para cada uno de los mensajes, asegurando la integridad de los mensajes de la conexión. En este caso no es necesario el envío de ninguna clave pública y ningún *client random data*. Por último se envía al servidor junto con la extensión contenedora del ID de la aplicación QKD cliente.

3.2.2. Envío del ServerHello

Cuando un mensaje *ClientHello* llega a un servidor TLS y el primer método de intercambio de clave es QKD-TLS-AES-256-GCM-SHA384, ambas partes son conscientes de que el intercambio de clave será realizado a través de QKD, y ambos se encargan de configurar cada parámetro de conexión para realizar un QKD-TLS *handshake* correcto. Como resultado de ello, el servidor genera un identificador de aplicación para su aplicación QKD, y lo envía empotrado dentro del mensaje *ServerHello* (en la extensión de mensaje TLS *QkdApplicationId*) al cliente.

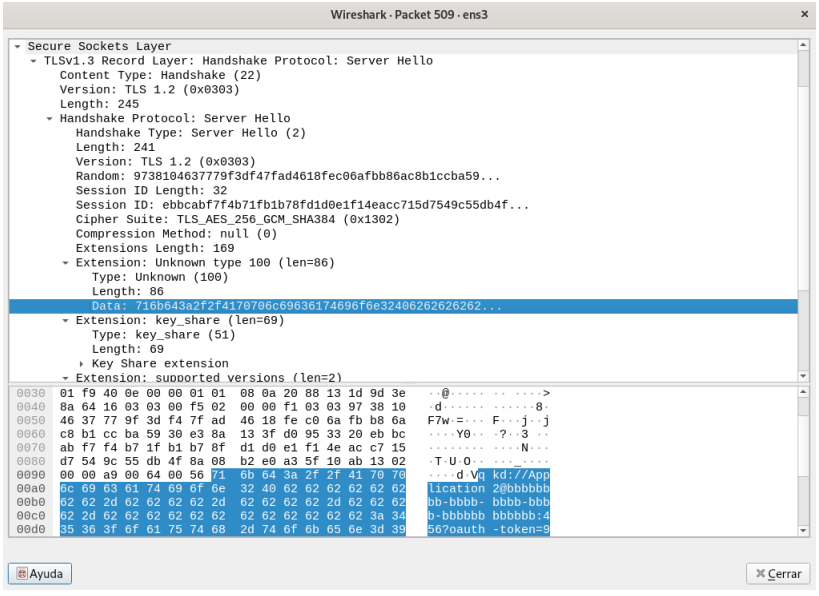

Figura 3.4: *ServerHello* message. Captura *Wireshark* del mensaje *ServerHello*, destacando el identificador de aplicación QKD del servidor.

Como podemos ver en la Figura 3.4 marcado en azul, la extensión de mensaje TLS contenedora del identificador de aplicación QKD del servidor es enviada al cliente además del intercambio de clave basado en QKD, como confirmación.

Después, el servidor efectúa la creación de la aplicación QKD servidor para realizar el intercambio de clave. El último paso de esta fase es la generación del mensaje *ServerHello* con la versión TLS negociada a 1.3, el ciphersuite negociado a QKD-TLS-AES-256-GCM-SHA384, ninguna clave pública y ningún *client random data*, y el envío al cliente junto con la extensión contenedora del ID de aplicación QKD servidor.

3.2.3. Envío de los Finished

Esta fase es la más compleja de las tres y en la que ha sido necesario añadir más operación. En el momento en que ambos extremos han recibido el mensaje *Hello* de la otra parte, se comienza el QKD-TLS *handshake*. El primer paso que realizan ambas partes es conectar sus respectivas aplicaciones a traves de la función *open_connect* de la interfaz ETSI GS QKD 004 [40], encargada de solicitar al LKMS que reenvíe la petición a su Agente para finalmente ser reenviada al controlador QKD de la red la reserva de una asociación (*Key Stream ID*) para pedir en un futuro un conjunto de claves a ambos extremos del enlace QKD, como se ha explicado en 2.4.

Cuando se ha registrado la aplicación en el controlador, con los dos extremos; es decir, cuando ambos extremos de la conexión han realizado su respectiva llamada *open_connect*, el controlador QKD evalúa el mejor camino entre ambos extremos. Además, se encarga de configurar los nodos inicial, final y todos los nodos intermedios entre ambos extremos, de manera que la clave es reenviada por todos ellos (*key relay*) hasta que llega a ambos extremos.

A continuación, ambas partes comienzan el intercambio de clave a través de la función de la interfaz ETSI GS QKD 004 [40] *get_key* asociada con el KSID recuperado anteriormente, pidiendo al LKMS una clave QKD para construir el secreto compartido. Esta función se encarga de solicitar al LKMS la obtención de una clave segura y que, previamente, se ha asegurado ser simétrica a ambos extremos del enlace QKD, Así, el secreto compartido es derivado en cada parte para obtener una clave QKD simétrica.

Por último, ambos extremos de la conexión ejecutan la función *close* de la interfaz ETSI GS QKD 004 [40], encargada de solicitar al controlador QKD de la red el cierre del enlace QKD asociado con el *Key Stream ID* dado.

A partir de aquí, el *handshake* TLS es exactamente igual al no modificado, con el detalle de que el secreto compartido ha sido obtenido a través de QKD en lugar de con algoritmos de PKI, convirtiendo la comunicación en *quantum-resistant*, y por lo tanto indescifrable.

En este momento, ambas partes generan un hash de todos los mensajes compartidos durante el handshake y el secreto compartido, utilizado como método de verificación de la integridad de éstos mensajes, para descartar que haya nadie interceptando y replicando éstos mensajes.

Cuando el servidor ha terminado, envía un mensaje *Finished* (empotrado dentro de

un mensaje *ApplicationData*) con el hash que el cliente deberá verificar, que es el resultado de aplicar la función de hashing con la clave QKD. En el momento en el que lo hace, envía un mensaje Finished con sus datos de verificación, que el servidor ahora deberá chequear. Cuando ambas partes han verificado correctamente el hash contenido en el mensaje Finished, comienza la sesión y todos los datos a partir de aquí están cifrados.

Ahora, la versión adaptada del protocolo TLS descrita en este trabajo crea una sesión en cada parte con las claves simétricas derivadas del secreto compartido para poder ser utilizadas posteriormente, y el protocolo de enlace ha acabado de manera satisfactoria.

De la manera descrita es como se ha conseguido la completa integración de tecnologías de distribución de clave cuántica en un protocolo criptográfico clásico como es TLS de manera transparente para el usuario.

3.3. Casos de Uso: HTTPS

En esta sección se va a dar una descripción detallada del caso de uso de HTTP basado en QKD-TLS. En la Figura 3.5 se muestra el diagrama de secuencia completo del caso de uso con todas las partes implicadas.

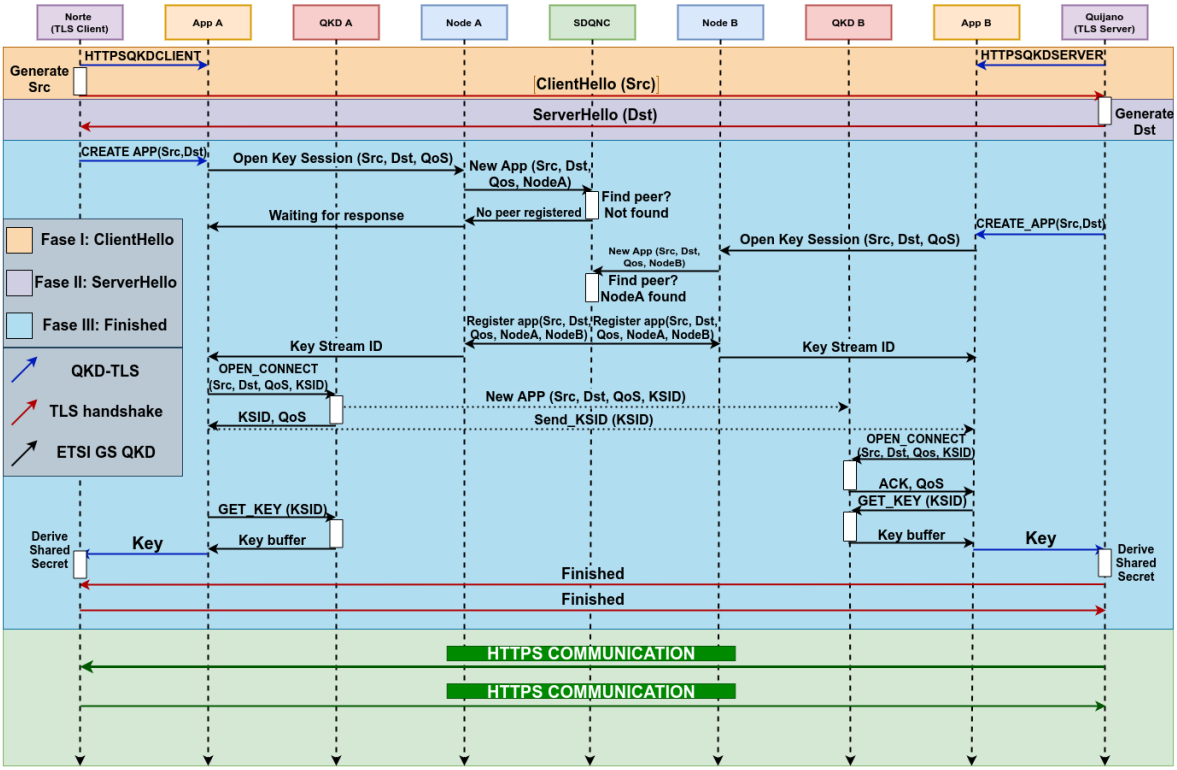

Figura 3.5: Diagrama de Secuencia del Caso de Uso HTTPS. Diagrama de secuencia que muestra el funcionamiento del caso de uso de HTTPS.

En primer lugar, la aplicación que ejecuta el servidor TLS en el nodo de Quinto (RM) comienza su ejecución, escuchando conexiones TLS en el puerto 4443. Después de un tiempo *t*, la aplicación que ejecuta el cliente TLS en el nodo Norte (TID) comienza su ejecución. En este momento, tiene lugar el handshake QKD-TLS implementado y explicado en la versión anterior. Cuando termina el *handshake*, ambas aplicaciones disponen de una clave TLS simétrica basada en QKD para el cifrado de la conexión. En este momento, el servidor aloja, a través de HTTPS, un fichero html donde el cifrado de la conexión HTTPS se soporta en TLS con la clave QKD recién recuperada. Es ahora cuando el cliente efectúa una conexión HTTPS con su clave QKD recién recuperada para alojar el fichero en su máquina. De esta manera, se ha ejecutado correctamente una conexión HTTPS cuya clave de cifrado TLS es una clave QKD recuperada directamente de dispositivos QKD reales. En la figura 3.6 podemos ver el código que ejecuta el cliente para efectuar la conexión HTTPS.

```
from tlslite import HTTPTLSConnection, HandshakeSettings
from tlslite_heritage import *

settings = HandshakeSettings()
settings.useExperimentalTackExtension = True
settings.qkd = True

h = HTTPTLSConnection(sys.argv[1], 4443, settings=settings)
h.request("GET", "/index.html")
r = h.getresponse()
print(r.read())
```

Figura 3.6: Código que ejecuta el cliente HTTPS.

Capítulo 4

Despliegue en escenario real

En este capítulo vamos a discutir los principales aspectos del despliegue completo del trabajo sobre un escenario real, así como las dificultades que han aparecido y las soluciones de diseño propuestas.

4.1. Escenario y despliegue

Como se ha explicado en la introducción, existen dos motivos para considerar innovador este trabajo. El primero consiste en la completa integración de un intercambio de clave QKD que hace uso de estándares reales en el protocolo TLS, y el segundo consiste en el correcto despliegue en un escenario real de redes definidas por software donde la clave QKD se obtiene de dispositivos cuánticos reales.

Se ha elegido un escenario con dos nodos que disponen de dispositivos QKD. El primero ejecutará una aplicación que funcionará como servidor TLS y el segundo uno que actuará como cliente TLS. El escenario propuesto es el mínimo viable para la correcta demostración de la operación, pero una vez se ha implementado el intercambio de clave QKD en el protocolo TLS, la escalabilidad a distintos nodos de la red se realiza de manera trivial.

Como se mencionó en la Introducción, la red cuántica de Madrid constituye el banco de pruebas de comunicaciones cuánticas más grande de toda Europa. Está constituida por 4 redes distintas que componen 11 nodos SDN con más de 280 kilómetros de fibra con enlaces de hasta 55 kilómetros de distancia. La red es *multitenant*, dado que los nodos están alojados en instalaciones en producción de Telefónica y RediMadrid. También hay un nodo especial, Norte, que conecta ambas redes. Este nodo es el *border node*.

Telefónica es la empresa de telecomunicaciones por excelencia de España, contribuyendo a la red de Madrid con lo que denominamos el anillo cuántico de TID, que actualmente cuenta con 3 nodos y enlaces de hasta 12 km y 7dB de pérdidas entre ellos.

RediMadrid es un proveedor de red que conecta diversas instituciones públicas, universidades y centros de investigación de Madrid, ofreciendo conectividad clásica 365 días al año.

La red de pruebas de Madrid dispone ahora de 26 dispositivos QKD, incluyendo emisores y receptores (20 ya están desplegados y 6 vienen este mes). Algunos de ellos funcionan en la banda C (1550 nm) y otros en la O (1300 nm) para lograr un aislamiento mejor del canal cuántico en los enlaces más difíciles. Algunos de ellos utilizan Variables Contínuas y otros Variables Discretas. Además, cada tipo de dispositivos QKD provienen de un vendedor (de los ya desplegados en la red, 6 son de IDquantique, 4 de Toshiba y 10 de Huawei). También se dispone de equipos de transporte óptico de ADVA y de Rohde and Schwarz.

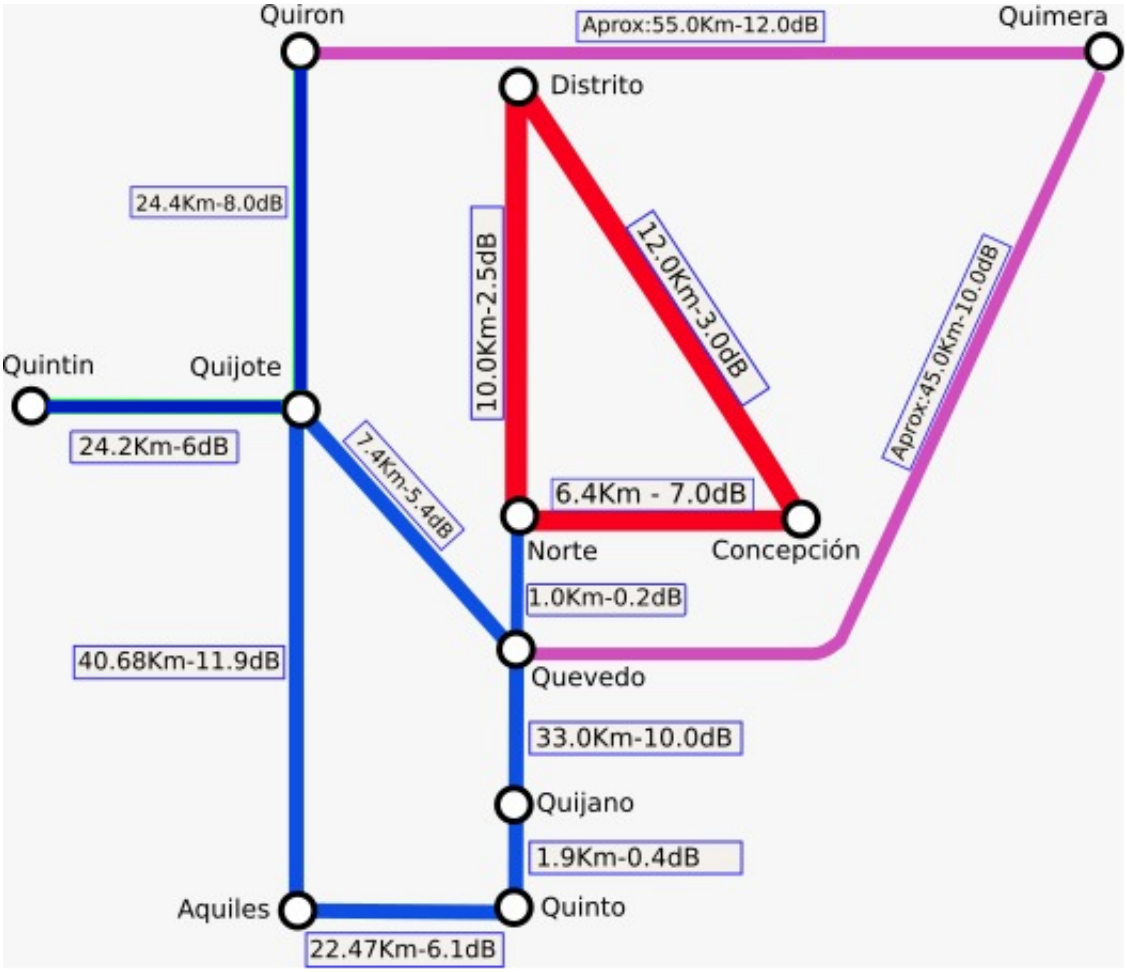

Figura 4.1: Red Cuántica Definida por Software de Madrid. Existen distintas redes marcadas por distintos colores, siendo la de TID la roja y la de RM la azul. En enlace morado está aún por licitar. El enlace usado es el que une Norte con Quinto.

El nodo Norte constituye un *border node* puesto que separa las redes de RM y TID. Este enlace es un elemento clave para la red de Madrid, puesto que permite hacer pruebas de QKD en una red *multitenant*. A través de este enlace somos capaces de generar una clave entre cualquier pareja de nodos de toda la red, sin importar si pertenecen a TID o RM.

Este caso de uso podría darse entre cualquiera de los nodos que aparecen en la Figura 4.1, pero el escenario propuesto se ha desplegado entre los nodos Norte (perteneciente a la red de TID) y Quinto (perteneciente a la red de RM), separados a 36 km de

distancia con un ruido máximo entre nodos de 10dB. Estos nodos disponen de un dispositivo QKD emisor y receptor, respectivamente, además de un encriptador en cada parte para poder desplegar de manera correcta el escenario propuesto.

4.2. Evaluación de resultados

En esta sección del capítulo se van a comentar los resultados obtenidos por la implementación descrita en la sección 4.1. Cabe destacar que estos resultados sostienen la parte innovadora de este trabajo mencionada anteriormente y el despliegue en una red real.

El primer paso ha sido el diseño y la posterior implementación de una adaptación del protocolo TLS *handshake* en un *framework* existente, y la prueba del correcto funcionamiento. Esta adaptación se ha integrado correctamente dentro de una implementación independiente de la librería *TLSlite-ng* de manera que para el usuario es transparente: un usuario que quiera hacer uso del intercambio de clave cuántica integrado en la librería no tiene que realizar ningún cálculo extra. Tan sólo tiene que incluir la extensión *QKDApplicationId* en la generación del mensaje *ClientHello* así como elegir el tipo de handshake QKD-TLS-AES-256-GCM-SHA384 como el primer intercambio de clave deseado por el cliente. A partir de ahí, el TLS *handshake* es transparente para el usuario.

La Figura 4.2 representa un ejemplo de ejecución correcta del *handshake* QKD-TLS, en el que vemos que el tiempo de handshake asciende a 9.8 segundos (notablemente mayor a los 14msec que tarda el handshake de TLS para una clave RSA de 2048 bits [56]). Aún así, consideramos que el tiempo añadido es aceptable considerando el aumento de seguridad conseguido.

```
Handshake time: 9.800 seconds
Version: TLS 1.3
Cipher: aes256gcm python
Ciphersuite: QKD_TLS_AES_256_GCM_SHA384
No client certificate provided by peer
Server X.509 SHA1 fingerprint: c4365fbd94328a333738035c12d2459bdb323a29
Group used for key exchange: secp256r1
SNI: server
Next-Protocol Negotiated: None
Encrypt-then-MAC: False
Extended Master Secret: True
```

Figura 4.2: QKD-TLS *handshake* realizado de manera exitosa.

En segundo lugar, la arquitectura implementada ha sido desplegada en un entorno real de distribución de clave cuántica, usando dispositivos QKD reales para obtener clave QKD. Este despliegue ha sido realizado en la red cuántica de Madrid, la red que constituye el banco de pruebas de comunicaciones cuánticas más grande de toda Europa. El caso de uso se ha lanzado entre dos nodos que no pertenecen a la misma red (uno pertenece al anillo de TID y el otro a la red de RM), por lo que ha sido necesario hacer *key relay* en 2 nodos intermedios (Quevedo y Quijano, de la figura 4.1). Recordemos que el escenario estudiado en este trabajo podría ser desplegado entre

cualquiera de los nodos descritos en la figura 4.1. Este es uno de los resultados más destacables de este proyecto, puesto que la prueba se ha realizado utilizando dispositivos QKD reales conectados a través de fibras ópticas de RediMadrid y Telefónica.

En tercer y último lugar, la arquitectura implementada ha sido probada con aplicaciones de alto nivel como es el protocolo HTTP. Además de ser uno de los más utilizados para la securización del transporte, el hacer uso de esta infraestructura QKD para ejecutar servicios de alto nivel de manera "transparente" es un gran avance para la comunidad científica puesto que la integración de por sí en un *framework* que utiliza TLS no es suficiente, se necesita el despliegue real en un escenario de tecnologías de alto nivel para asegurar que la implementación propuesta es capaz de desplegar casos de uso reales y prácticos haciendo uso de clave QKD de manera transparente para el usuario.

Capítulo 5

Conclusiones y trabajo futuro

Dar una predicción acertada de los avances que puede suponer el uso de la computación cuántica y su capacidad en la resolución de complejos problemas es muy complicado. Debido a la llegada de la computación cuántica, los sistemas tradicionales de seguridad y las comunicaciones hoy en día están seriamente comprometidos. Para enfrentar este ataque, las grandes tecnológicas como Huawei o Toshiba y las empresas de telecomunicaciones y proveedores de conexión como Telefónica o RediMadrid buscan formas que permitan luchar contra esta problemática.

Una de las alternativas más prometedoras para enfrentarse a este problema es la Criptografía Cuántica, que utiliza principios de la mecánica cuántica (y no de técnicas de computación clásica como puede ser la teoría de números), prometiendo ser una solución idónea para convertirse en la siguiente capa de seguridad en las comunicaciones pues es inmune (ITS) a ataques tanto clásicos como cuánticos.

Esta tecnología se basa en la generación de números aleatorios reales, de naturaleza cuántica, de manera síncrona en dos extremos distintos de una red, gracias al continuo envío de fotones de luz a través de un canal de fibra óptica o un enlace satelital.

Una vez existe una tecnología capaz de proteger las comunicaciones actuales contra los posibles ataques de la computación cuántica, la Criptografía Cuántica, es necesaria su integración en las tecnologías que son usadas día a día para la securización de las conexiones punto a punto. Este ha sido el punto de partida de este trabajo, teniendo como objetivo principal la integración de tecnologías QKD empotradas en el protocolo de transporte más utilizado hoy en día, TLS. Algunas de las aportaciones que se pueden obtener son las siguientes:

- Utilización de un estándar para la obtención de clave cuántica que procede de dispositivos QKD reales. Ninguno de los trabajos anteriores hacen uso de ellos sino que utilizan una implementación propia del BB84, asistiendo al protocolo con clave cuántica simulada. En nuestro caso, esta negociación ya ha sido realizada y el usuario recibe la clave independientemente del protocolo usado. La utilización de estándares hace que la solución propuesta sea más escalable.
- Definición de un escenario real de despliegue que hace uso de dispositivos QKD. Los trabajos previos no definen un escenario real de uso, tan sólo teóricos. En [43] no llega a ser un escenario completo ya que en ningún momento se define

el canal cuántico ni las pruebas realizadas. En [45] se explican las distancias máximas para que el canal cuántico funcione correctamente en cada uno de los escenarios teóricos. En este trabajo se han efectúado las pruebas del nodo Norte de la empresa de telecomunicaciones Telefónica al nodo Quinto, del proveedor de conexión RediMadrid. El enlace entre ellas está formado por 36 km de fibra óptica con un ruido máximo de 10dB y dos nodos intermedios, Quevedo y Quijano, encargados de efectuar el *key relay*.

- La arquitectura implementada ha sido probada en la Madrid Quantum Network con aplicaciones de alto nivel como es el protocolo HTTPS. Esto proporciona una utilización de estos servicios de alto nivel con infraestructura QKD de manera "transparente". Además, se ha implementado igualmente de manera transparente para los usuarios.
- En lugar de una implementación ad-hoc como en los trabajos [43, 44, 45, 46], la implementación descrita en este trabajo está basada en estándares, lo que facilita su transporte y aplicabilidad con diferentes dispositivos QKD.
- Está implementado y desplegado en una red real con un novedoso principio en redes como es SDN, que las empresas de telecomunicaciones están incorporando cada vez más en sus infraestructuras.

Para resumir, la novedad de este trabajo radica en la definición de un escenario real de integración de QKD en TLS, implementado con un estándar de uso QKD en auge, y su aplicación sobre una red definida por software de computadores físicos que hacen uso de dispositivos QKD reales para obtener la clave cuántica. Además, esta implementación es independiente; es decir, no está ligada al protocolo de negociación de la clave QKD, pudiendo funcionar tanto con dispositivos de Variable Contínua como de Variable Discreta. Es más, es indiferente al protocolo usado a nivel QKD [57], ya que esta aproximación funciona con cualquier dispositivo, como se ha visto en su despliegue la Madrid Quantum Network.

Se considera necesario hablar de la latencia y el *overhead* añadidos por la operación de QKD en el *handshake* de TLS. El único punto crítico que se considera necesario destacar es el *overhead* introducido en la fase 3 (envío de los *Finished*). El protocolo TLS es un protocolo muy rápido (como se comenta en [47]) y las tecnologías de distribución de clave cuántica actuales no lo son tanto. La latencia se ha incrementado en la fase 1, pero el *overhead* introducido es completamente despreciable: tan sólo se añade el cómputo necesario para el cálculo de una cadena de bytes que sirva como identificador de aplicación QKD, pudiendo considerarse nulo. Lo mismo ocurre con la latencia incrementada en la fase 2: tan sólo se añade el cómputo necesario para el cálculo del identificador de la aplicación QKD, por lo que es considerado también despreciable. La mayor parte de la latencia introducida se da en la fase 3, como se ve claramente en el diagrama de secuencia anterior. Es en esta fase donde ocurre la solicitud de clave cuántica, por lo que dependiendo del tamaño de la clave el tiempo de recuperación de ésta será mayor o menor.

Finalmente, a nivel personal como investigador y como alumno, la integración de un protocolo tan puntero en tecnologías de alto nivel ha sido un gran desafío y una experiencia enriquecedora. Este trabajo sienta las bases de la integración completa de un esquema de seguridad muy novedoso en un protocolo de comunicaciones muy utilizado, pudiendo suponer el inicio para una futura contribución en el mundo de la

securización de las telecomunicaciones.

5.1. Trabajo Futuro

1. **Estandarización del intercambio de clave cuántica como un nuevo intercambio de clave empotrado en el protocolo TLS.**

   La implementación desarrollada en este trabajo requiere de una serie de condiciones necesarias que dificultan el despliegue y, sobre todo, la escalabilidad de la solución. Consideramos que una línea futura de investigación sería la integración completa del intercambio de claves cuánticas en el protocolo TLS. Para ello, sería necesaria la escritura de un RFC para TLS que incluya un nuevo intercambio de clave cuántico con los parámetros necesarios (como se ha desarrollado en el capítulo 3) para forzar que todas las implementaciones de este protocolo usen clave cuántica de manera que un usuario de TLS solo necesite incluir una extensión en el mensaje *ClientHello* para el uso del esquema propuesto.

2. **Implementación de nuevos casos de uso.**

   Los casos de uso son la forma idónea de demostración del correcto funcionamiento de la tecnología. Al fin y al cabo, no es suficiente con la integración del protocolo QKD en el TLS *handshake*, sino que podría usarse en entornos más complejos. El primero y más obvio es el de HTTPS, ya que la mayoría de las conexiones a través de Internet utilizan este protocolo. Una línea futura lógica es el desarrollo de otros casos de uso que validen esta integración. Actualmente, se está trabajando en el caso de uso de SMTP con el objetivo de securizar las comunicaciones del correo electrónico.

3. **Actualización a clave TLS *on-the-way*.**

   La implementación ha sido desarrollada a través de la librería *TLSlite-ng* de python, que contiene implementación sobre la versión 1.3. Creo que una línea futura lógica sería la actualización de la clave TLS on-the-way, actualizando las claves de sesión cada un cierto tiempo determinado.

Bibliografía

[1] H. Delfs & H. Knebl, "Introduction to cryptography: Principles and Applications", *Textbook*, 2015.

[2] History of the Enigma. https://www.cryptomuseum.com/crypto/enigma/hist.htm. Accessed: 2022-06-19.

[3] C. E. Shannon, "A mathematical theory of communication", *The Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, July 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x*, July 1948.

[4] W. Diffie & M. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, November 1976, doi: 10.1109/TIT.1976.1055638*, November 1976.

[5] J.Johnson *et al.*, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1",*IETF RFC 3447*, February 2003.

[6] D. E. Denning, "Is quantum computing a cybersecurity threat? although quantum computers currently don't have enough processing power to break encryption keys, future versions might", *American Scientist, vol. 107, no. 2, pp. 83-86*, 2019.

[7] V. Bhatia & K. Ramkumar, "An efficient quantum computing technique for cracking rsa using shor's algorithm", *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), pp. 89-94*, 2020.

[8] M. Elboukhari *et al.*, "Implementation of secure key distribution based on quantum cryptography", *2009 International Conference on Multimedia Computing and Systems, pp.361-365, doi:10.1109/MMCS.2009.5256673*,, 2009.

[9] M. A. Nielsen & I. L. Chuang, "Quantum Computation and Quantum Information: 10th Anniversary Edition", *ambridge University Press*, 2011.

[10] OpenQKD. https://openqkd.eu/. Accessed: 2022-06-19.

[11] CiViQ. https://civiquantum.eu. Accessed: 2022-06-19.

[12] Quantum FLagship. https://qt.eu/. Accessed: 2022-06-19.

[13] EuroQCI. https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci. Accessed: 2022-06-19.

[14] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks", *ACM SIGCOMM computer communication review 38.2 (2008), págs. 69-74.*, Abril 2008.

[15] IDQUANTIQUE. https://www.idquantique.com/xg-series-qkd/. Accessed: 2022-06-19.

[16] D. LANCHO *et al.*, "QKD in Standard Optical Telecommunications Networks", *Quantum Communication and Quantum Networking. QuantumComm 2009. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 36. Springer, Berlin, Heidelberg.*, 2009.

[17] A. CIURANA *et al.*, "Entanglement Distribution in Optical Networks", *IEEE JOURNAL OF SELECTED TOPICS IN QUANTUM ELECTRONICS, VOL. 21, NO. 3, MAY/-JUNE 2015*, Mayo 2015.

[18] ADVA. https://www.adva.com. Accessed: 2022-06-19.

[19] ROHDE AND SCHARZ. https://www.rohde-schwarz.com. Accessed: 2022-06-19.

[20] A. AGUADO *et al.*, "Secure Critical Infrastructures via QKD: the Madrid QKD Network", *QCrypt 2018, 8th International Conference on Quantum Cryptography*, Agosto 2018.

[21] D. BUNANDAR *et al.*, "Numerical finite-key analysis of quantum key distribution", *npj Quantum Inf 6, 104 (2020)*, Octubre 2020.

[22] T. ELGAMAL, "The Secure Sockets Layer Protocol", *Danvers IETF Meeting*, Abril 1995.

[23] A. FREIER *et al.*, "The Secure Sockets Layer (SSL) Protocol Version 3.0" *IETF RFC 6101*, August 2011.

[24] T. DIERKS *et al.*, "The TLS Protocol Version 1.0", *IETF RFC 2246*, Enero 1999.

[25] T. DIERKS *et al.*, "The TLS Protocol Version 1.1", *IETF RFC 4346*, Abril 2006.

[26] T. DIERKS *et al.*, "The TLS Protocol Version 1.2", *IETF RFC 5246*, Agosto 2008.

[27] E. RESCORLA, "The Transport Layer Security (TLS) Protocol Version 1.3", *IETF RFC 8446*, August 2018.

[28] GOOGLE, "Google Transparency Report - HTTPS encryption on the web", 2020.

[29] V. MAVROEIDIS *et al.*, "The Impact of Quantum Computing on Present Cryptography", *International Journal of Advanced Computer Science and Applications, Vol.9, No.3* , 2018.

[30] OPENSSL. https://www.openssl.com. Accessed: 2022-06-19.

[31] PARAMIKO. https://www.paramiko.org/. Accessed: 2022-06-19.

[32] H. KARIO, "tlslite-ng: A pure python implementation", *PyPi*.

[33] H.K. LO, *et al.*, "Secure Quantum Key Distribution", *Nature Photonics 8, 595-604*, 2014.

[34] H. WONYOUNG, "Quantum Key Distribution with High Loss: Toward Global Secure Communication", *Phys.Rev.Lett 91-057901*, August 2003.

[35] C. BENNET & *G. Brassard*, "Quantum cryptography: Pyblic key distribution and coin tossing", *International Conference on Computers, Systems & Signal Processing, pp.175-179*, December 1984.

[36] F. JAZAERI *et al.*, "A Review on Quantum Computing: From Qubits to Front-end Electronics and Cryogenic MOSFET Physics", *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems", 2019, pp. 15-25*, 2019.

[37] L. LHOTKA, "JSON Encoding of Data Modeled with YANG", *IETF RFC 7951*, August 2016.

[38] P. WINIARCZYK *et al.*, "BB84 analysis of operation and practical considerations and implementations of quantum key distribution systems", *2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), pp. 23-26* , 2011.

[39] "ETSI: Quantum Key Distribution: Control Interface for Software Defined Networks", *ETSI GS QKD 015 V1.1.1*, March 2021.

[40] "ETSI: Quantum Key Distribution: Application Interface", *ETSI GS QKD 015 V2.1.1*, August 2020.

[41] H. ZHENG *et al.* "A YANG Data Model for Wavelength Switched Optical Networks (WSONs)", *IETF RFC 9094*, August 2021.

[42] S. AL-JANABI *et al.*, "Extension of SSL/TLS for Quantum Cryptography", *manager's Journal on Software Engineering 2. 64-76. 10.26634/jse.2.2.669*, 2007.

[43] S. AL-JANABI *et al.*, "A novel extension of SSL/TLS based on Quantum Cryptography", *919 - 922. 10.1109/ICCCE.2008.4580740*, 2008.

[44] M. ELBOUKHARI *et al.*, "Integration of Quantum Key Distribution in the TLS Protocol", *IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.12*, 2009.

[45] M. PIVK *et al.*, "SSL/TLS with quantum cryptography", *Proceedings of the 3rd International Conference on Quantum, Nano and Micro Technologies, ICQNM 2009.96 - 101. 10.1109/ICQNM.2009.29*, 2009.

[46] M. ELBOUKHARI· *et al.* "Improving TLS Security By Quantum Cryptography", *International Journal of Network Security & Its Applications. 2.10.5121/ijnsa.2010.2306*, 2010.

[47] Y. TANIZAWA *et al.*, "An approach to integrate quantum key distribution technology into standard secure communication applications", *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), 2017, pp. 880-886, doi:10.1109/ICUFN.2017.7993926*, 2017.

[48] B.A. HUBERMANN *et al.*, "Quantum Secured Internet Transport", *Transport, Inf Syst Front 22, 1561-1567*, 2020.

[49] M. ELBOUKHARI *et al.*, "Integration of Quantum Key Distribution in EAP-TLS protocol used for wireless LAN authentication", *2010 5th International Symposium On I/V Communications and Mobile Network, 2010, pp. 1-4, doi: 10.1109/ISVC.2010.5656266.*, 2010.

[50] A. MINK *et al.*, "Quantum Key Distribution (QKD) and Commodity Security Protocols: Introduction and Integration", *International Journal of Network Security & Its Applications (IJNSA), vol. 1, no. 2, pp101-111*, July 2009.

[51] M.A. SFAXI *et al.*, "Using Quantum Key Distribution within IPSEC to secure MAN communications", *MAN 2005 Conference*, 2005.

[52] S. MARKSTEINER *et al.*, "On the Resilience of a QKD Key Synchronization Protocol for IPsec", *International Journal on Advances in Security, vol. 9, no. 3 & 4*, 2016.

[53] U. BLUMENTHAL *et al.*, "Advanced Encryption Standard (AES)", *NIST FIPS 197* , Noviembre 2001.

[54] D. A. MCGREW *et al.*, "The Galois/Counter Mode of Operation (GCM)", *NIST*, 2005.

[55] "Secure Hash Standards (SHS)", *NIST FIPS 180-4*, Agosto 2015.

[56] B. DOWLING *et al.*, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol", *J Cryptol 34, 37*, 2021.

[57] S. WANG *et al.*, "Field and long-term demonstration of a wide area quantum key distribution network", *Opt. Express 22, 21739-21756*, 2014.

[58] S. WOLF, "Unconditional security in cryptography", *Lectures on Data Security: Modern Cryptology in Theory and Practice, Springer, 1999, pp. 217-250*, 1999.

[59] D.MERMIN, "Breaking rsa encryption with a quantum computer: Shor's factoring algorithm", *Lecture notes on Quantum computation, pp. 481-681*, 2006.

[60] C. H. BENNETT *et al.*, "Experimental quantum cryptography", *Journal of Cryptology, vol. 5, no. 1, pp. 3-28*, 1992.

[61] C. H. BENNETT *et al.*, "Quantum cryptography", *Scientific American, vol. 267, no. 4, pp. 50-57*, 1992.

[62] D. MAYERS, "Unconditional security in quantum cryptography", *Journal of the ACM (JACM), vol. 48, no. 3, pp. 351-406*, 2001.

[63] M. PEEV *et al.*, "The SECOQC quantum key distribution network in Vienna", *New Journal of Physics, vol. 11, no. 7, p. 075 001*, 2009.

[64] M. KOROLOV & D. DRINKWATER "What is quantum cryptography? It's no silver but could improve security", *CSO* March 2019.

[65] A. R. DIXON *et al.*, "Continuous operation of high bit rate quantum key distribution", *Applied Physics Letter 96, 161102*, 2010.

[66] M. DIANATI & R. ALLEAUME, "Transport Layer Protocols for the Secoqc Quantum Key Distribution (QKD) Network", *32nd IEEE Conference on Local Computer Networks, pp.1025-1034*, October 2007.

Anexo A: Aspectos éticos, económicos, sociales y ambientales

.1. Introducción

Este proyecto nace de un problema real en la securización de las comunicaciones actuales, que utilizan la teoría de números para impedir que un atacante pueda recuperar la información de la comunicación y la computación cuántica es capaz de romper esta seguridad. Socialmente es un problema de privacidad, puesto que las comunicaciones serían interceptables por cualquier atacante poseedor de un procesador cuántico.

Este proyecto pretende resolver el problema ético, social y económico que esto supone.

.2. Descripción de impactos relevantes relacionados con el proyecto

Este proyecto provoca un impacto en los tres ámbitos: social, ético y económico, puesto que el hecho de que no sea posible mantener una comunicación segura a través de la red hace que nuestra forma de vivir e interactuar con el mundo (conexiones HTTP, transferencias bancarias, etc.) se vería cambiada de una manera radical, puesto que la privacidad dejaría de existir.

.3. Análisis detallado de alguno de los principales impactos

El principal impacto de este trabajo reside en la securización de las comunicaciones que están en peligro hoy en día.

Este proyecto tiene un gran valor desde un punto de vista social, ético y económico, puesto que pretende que cualquier persona sea capaz de mantener una comunicación punto a punto segura a lo largo de Internet.

Desde un punto de vista social y ético, es muy importante que todo el mundo conserve su privacidad en la red, puesto que una falta de ello significaría que no nos sentimos seguros en Internet, provocando que nuestra interacción sea más cuidadosa y, en definitiva, falsificada por la falta de privacidad.

Desde un punto de vista económico, sería desastroso un mundo en el que es posible romper toda la criptografía clásica, en la que se sustentan todas las comunicaciones y cifrado de los datos hoy en día. Por ello, este proyecto tiene un gran valor económico puesto que propone una solución a este problema, consiguiendo que las conexiones a través de Internet sigan siendo seguras.

.4. Conclusiones

Podemos concluir que este proyecto aporta un gran valor desde un punto de vista social, ético y económico para conseguir la securización de las conexiones a través de Internet, tal y como se ha descrito en la sección .3.

Anexo B: Presupuesto Económico

COSTE DE MANO DE OBRA (coste directo)			Horas	Precio/hora	Total
			300	15 €	**4.500 €**
COSTE DE RECURSOS MATERIALES (coste directo)		**Precio de compra**	**Uso en meses**	**Amortización (en años)**	**Total**
Ordenador personal (Software incluido).....		2.500,00 €	6	5	150,00 €
Otro equipamiento					
COSTE TOTAL DE RECURSOS MATERIALES					**200,00 €**
GASTOS GENERALES (costes indirectos)	15%	sobre CD			**705,00 €**
BENEFICIO INDUSTRIAL	6%	sobre CD+CI			**324,30 €**
SUBTOTAL PRESUPUESTO					**6.129,30 €**
IVA APLICABLE				21%	**1.287,15 €**
TOTAL PRESUPUESTO					**7.416,45 €**