Grado en Física

Trabajo Fin de Grado

Ataques a protocolos QKD

Nerea Fernández Ubierna

Tutor: Carlos Sabín Lestayo

Universidad Autónoma de Madrid

Curso académico 2024-25

Resumen

Este proyecto estudia dos protocolos QKD, los ataques a los que pueden ser vulnerables y las posibles soluciones para mitigarlos: el protocolo BB84 frente a un ataque *Photon-Number Splitting*, detectado mediante el uso de estados señuelo; y el protocolo SARG04 ante un *Trojan Horse Attack*, para el cual se propone como solución una monitorización en tiempo real de los detectores. El principal objetivo es desarrollar dos programas interactivos que permitan al usuario establecer una serie de parámetros ajustados a sus necesidades. En el caso del protocolo BB84, se demuestra que, mediante el método de estados señuelo, y gracias a la variación del alcance de estos estados, Alice y Bob pueden detectar un posible ataque PNS. Para el protocolo SARG04, se simula tanto su funcionamiento como las consecuencias del robo de la clave final por parte de Eve, tras un ataque THA dirigido a los dispositivos de preparación de bases de medida de Bob.

1. Introducción

De la mano de los primeros ordenadores cuánticos, surge una nueva preocupación: los protocolos criptográficos actuales no son seguros frente a un posible futuro cuántico. La criptografía clásica se basa en problemas que, con la tecnología actual, son computacionalmente intratables; sin embargo, no existe una demostración formal de su seguridad. Esto abre la puerta a algoritmos cuánticos, como el de Shor, que podrían resolver estos problemas de forma eficiente y comprometer la confidencialidad de la información.

Una de las soluciones más prometedoras es la distribución cuántica de claves (QKD), que permite compartir claves privadas con una seguridad demostrable basada en las leyes de la física cuántica. No obstante, las limitaciones tecnológicas actuales hacen que estos protocolos no estén exentos de vulnerabilidades y puedan ser objeto de ataques.

El objetivo de este trabajo es analizar dos protocolos QKD: BB84 y SARG04, junto con sus respectivos ataques, PNS y THA, y proponer soluciones o medidas de detección. Todo ello se complementa con simulaciones interactivas que permitan experimentar con distintos parámetros del sistema.

2. QKD: Quantum Key Distribution

2.1. Definición

Quantum Key Distribution, o en español Distribución Cuántica de Claves, es un protocolo que se puede demostrar como completamente seguro, por lo menos en su aspecto teórico. Una clave privada se puede crear entre dos personas separadas espacialmente. La clave privada está conformada por bits clásicos, que servirán para la encriptación de información sensible, mientras que la parte cuántica es la propia distribución de la clave. En los sistemas QKD se emplean dos canales de comunicación: un canal cuántico, por el que viaja la clave codificada en estados cuánticos llamados *qubits*, donde un *qubit* corresponde a un bit de la clave privada, y un canal público, que se asume como inseguro. No obstante, la seguridad de la clave privada compartida tras la distribución está garantizada por las propiedades de la información cuántica, y por lo tanto está solamente condicionada por el hecho de que las leyes fundamentales de la física cuántica sean correctas [1].

La convención que se usa para explicar el esquema básico de un sistema QKD es la siguiente: Alice es la emisora de la clave privada, Bob el receptor y Eve es la espía que intenta conocer la clave privada compartida entre Alice y Bob. Alice enviará el mensaje codificado a Bob mediante el canal cuántico, y usarán el canal público para filtrar y quedarse solamente con la clave compartida (Fig. 1.). Este proceso se explicará detenidamente cuando se estudien los protocolos BB84 y SARG04 en detalle. Además, a lo largo de este proyecto, el canal cuántico empleado por Alice y Bob siempre será fibra óptica, que es muy común en los experimentos reales para intentar reusar la estructura de telecomunicaciones ya existente.

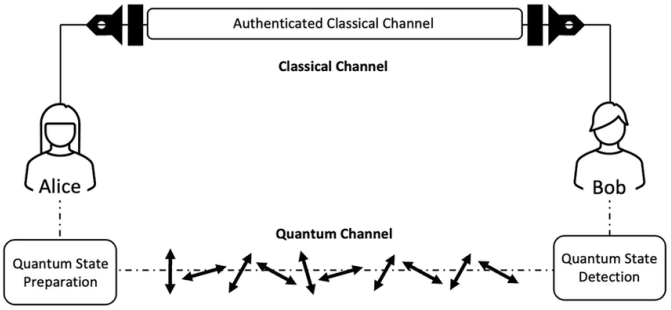

Fig. 1. Esquema básico de un protocolo QKD. Alice es la emisora del menaje y Bob el receptor. El mensaje codificado viaja por el canal cuántico y el canal clásico sirve para la destilación de la clave [2].

2.2. La física que garantiza la seguridad

El término seguridad se refiere a que terceras personas que no deberían conocer la clave secreta, como Eve, no sean capaces de robar esa información sin que Alice o Bob lo sepan. La idea que garantiza dicha seguridad es la siguiente: Eve no puede ganar ninguna información de los *qubits* transmitidos entre Alice y Bob sin generar perturbaciones en sus estados

cuánticos. Primero, por el teorema de no clonación, que declara que no se puede crear una copia idéntica de un estado cuántico desconocido arbitrario, Eve no puede crear una copia de los estados de Alice. Segundo, ganar información implica perturbar el sistema. En cualquier intento de distinguir entre dos estados cuánticos no ortogonales, ganar información acerca de estos estados sólo es posible introduciendo perturbaciones en la señal [1]. La no ortogonalidad es un aspecto esencial y se explicará detalladamente en la próxima sección.

3. El protocolo BB84

3.1. Definición

El protocolo BB84 es uno de los protocolos QKD más empleados, así como una fuente de inspiración para otros muchos que han ido surgiendo gracias a la constante evolución de la computación cuántica. Este protocolo recibe su nombre de sus creadores, Charles Bennet y Gilles Brassard, en el año 1984.

El funcionamiento del protocolo se puede resumir brevemente en que Alice, que quiere enviar una clave privada a Bob, va a codificar su información en 4 posibles estados de polarización del fotón de manera aleatoria. Estos estados de polarización de los fotones son estados cuánticos, y viajarán por el canal cuántico hasta llegar a Bob. Él será el encargado de detectar esos fotones, que llevan la información clásica codificada y Alice quiere transmitir de forma segura. Tras un proceso de verificación de que no ha habido espionaje por parte de terceras personas, el protocolo se declara como válido y Alice y Bob tienen cada uno una clave segura, privada y compartida con la que poder encriptar y desencriptar información.

3.2. Preparación y funcionamiento en detalle

El protocolo empieza por Alice. Ella elige una cadena aleatoria a de bits clásicos (0 o 1), que será el mensaje que quiere enviar de manera segura. Luego, elige otra cadena de bits clásicos, b, que se conoce como la cadena de bases. Si el bit de b es 0, la base de codificación será la Z, si es 1, la X. Además, cada base está formada por dos posibles estados ortogonales. Sabiendo la base en la que se va a codificar, se selecciona uno de los dos posibles estados según la combinación de un bit de a con un bit de b, en ese orden (por convención). Estas 4 combinaciones van a conformar los estados de codificación de los *qubits*, o también llamados los 4 estados de polarización del fotón [1]:

$$|\psi_{00}\rangle = |0\rangle$$
$$|\psi_{10}\rangle = |1\rangle$$
$$|\psi_{01}\rangle = |+\rangle = \tfrac{|0\rangle+|1\rangle}{\sqrt{2}}$$
$$|\psi_{11}\rangle = |-\rangle = \tfrac{|0\rangle-|1\rangle}{\sqrt{2}}$$

Donde la base Z la conforman los estados $\{|0\rangle, |1\rangle\}$ y la base X los estados $\{|+\rangle, |-\rangle\}$ y se recuerda que $|0\rangle = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ y $|1\rangle = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$.

Bob recibe estos *qubits*, y los mide o bien en la base X, o bien en la base Z, pero la elección la realiza de forma aleatoria. Bob crea otra cadena de bases, b', con la selección aleatoria entre 0 y 1. Si el valor del bit aleatorio es 0, Bob medirá el *qubit* en la base Z, mientras que si es 1, medirá en base X. Bob ahora tendrá una cadena de resultados, a', conformada por bits clásicos, conocida como la clave cruda o *raw key*.

El siguiente paso es la declaración pública de las cadenas de bases que han usado tanto Alice como Bob de manera individual, es decir, publicar b y b' mediante el canal clásico público. Es interesante notar que anunciar públicamente b y b' no afecta a la seguridad de la clave privada final, ya que estas cadenas no contienen ninguna información acerca de a ni a' (recordar que de a y a' se sacará la clave final compartida). Alice y Bob se quedarán tan solo con los bits en las posiciones donde $b = b'$, y el resto de los bits se descartan. Después de este proceso normalmente ambos se quedan con cerca de la mitad de los bits que tenían [1]. Es fácil ver que si para una posición se cumple $b = b'$, entonces $a = a'$, ya que se ha codificado y descodificado con la misma base, y los estados dentro de una base son ortogonales.

Bit aleatorio de Alice	0	1	1	0	1	0	0	1
Base de envío aleatorio de Alice	+	+	×	+	×	×	×	+
Polarización de fotones enviados por Alice	↑	→	↘	↑	↘	↗	↗	→
Base de medición aleatoria de Bob	+	×	×	×	+	×	+	+
Mediciones de la polarización de fotos de Bob	↑	↗	↘	↗	→	↗	→	→
Discusión pública de las bases								
Clave secreta compartida	0		1			0		1

Fig. 2. Esquema completo del protocolo BB84 para generar una clave secreta compartida entre Alice y Bob, que están separados espacialmente.

En este punto ya se tiene la clave destilada. Para terminar el protocolo se deben hacer dos pasos más: la reconciliación de la información y la amplificación de la privacidad. La reconciliación de la información no

es más que la corrección de errores a través del canal clásico público [1]. Alice escoge n bits aleatoriamente de los bits que le quedan, conocidos como bits de muestra, y compara mediante el canal clásico con los bits de Bob en esas mismas posiciones. Teóricamente, los valores de bits deberían coincidir, pero debido a imperfecciones en los aparatos, ruido en el canal cuántico o incluso un intento de espionaje por Eve, se pueden producir errores en el protocolo. La medida que se emplea es el $QBER$ (*Quantum Bit Error Rate*):

$$QBER = \frac{n_{bits,erroneos}}{n_{bits,totales}} \tag{1}$$

Donde $n_{bits,erroneos}$ es la cantidad de bits que no coinciden entre Bob y Alice de los bits de muestra y $n_{bits,totales}$ es la cantidad total de bits de muestra que se usa.

Si el $QBER$ no supera un cierto límite, calculado según diferentes factores como el protocolo empleado o las pérdidas del canal, el protocolo se toma como válido y se puede continuar. Si no, se desechan todos los bits y se vuelve a empezar.

El último paso es la amplificación de la privacidad. La idea es destilar de nuevo la clave que se obtiene tras la reconciliación de información para mantener la correlación de la nueva clave por debajo de un cierto límite. Por ejemplo, empleando funciones *hash* universales [3]. Una función *hash* H es una función computable mediante un algoritmo tal que [19]:

$$\begin{aligned} H : U &\longrightarrow M \\ x &\longrightarrow h(x) \end{aligned} \tag{2}$$

Normalmente el conjunto U tiene un número elevado de elementos y el conjunto M un número acotado. La idea básica de un valor *hash* es que sirva como una representación compacta de la cadena de entrada [3]. Por otra parte, un conjunto de funciones *hash* se define como universal si, al elegir una función $h \in H$ de manera uniformemente aleatoria, la probabilidad de que dos claves distintas, $x \neq y$; $x, y \in U$ coincidan, es decir, $h(x) = h(y)$, es baja, siendo como mucho $1/|M|$ [1].

3.3. La importancia de la no ortogonalidad

El simple hecho de que se empleen bases no ortogonales en el protocolo garantiza que, en caso de intervención, Eve no pueda saber con seguridad si el resultado de su medida es correcto o no.

Partiendo de las dos bases Z y X, se recuerda que los estados pertenecientes a Z eran $\{|0\rangle, |1\rangle\}$, mientras que los estados de la base X eran $\{|+\rangle, |-\rangle\}$. Es trivial que los dos estados dentro de una misma base son ortogonales entre ellos.

$$|\langle 0|0\rangle|^2 = |\langle 1|1\rangle|^2 = |\langle +|+\rangle|^2 = |\langle -|-\rangle|^2 = 1$$
$$|\langle 1|0\rangle|^2 = |\langle 0|1\rangle|^2 = |\langle -|+\rangle|^2 = |\langle +|-\rangle|^2 = 0$$

Si solo se codificase con la base Z y, por ejemplo, Alice envía el estado $|0\rangle$ y Eve mide en base Z, el resultado que obtiene de la medida es el bit 0 con una probabilidad del 100 %, por lo que emplear tan solo estados ortogonales sería un problema. Ahora bien, los estados de una misma base no son ortogonales con los de la otra.

$$|\langle 0|+\rangle|^2 = |\langle 1|+\rangle|^2 = |\langle 0|-\rangle|^2 = |\langle 1|-\rangle|^2 = \tfrac{1}{2}$$
$$|\langle +|0\rangle|^2 = |\langle +|1\rangle|^2 = |\langle -|0\rangle|^2 = |\langle -|1\rangle|^2 = \tfrac{1}{2}$$

Esto permite que Eve no pueda acertar los estados iniciales siempre, porque no hay una probabilidad 0 de obtener algún resultado al escoger la base incorrecta. Además, si se mide en la base incorrecta se proyecta el estado, de manera que es distinto tras la medida y la intervención es detectable.

Claro está que, aún así, Eve sigue teniendo una probabilidad del 50 % de elegir la base correcta para su medida, pero en la siguiente sección se verá que a pesar de ello la intervención de Eve sería detectable.

4. Ataque IR: Intercept-Resend

4.1. Definición

El ataque de Intercepción y Reenvío es uno de los primeros ataques que se comenzaron a valorar con la creación de los protocolos QKD [4]. El ataque es sencillo: Eve desvía el canal cuántico de Alice y Bob, de forma que cuando Alice envíe los estados, Eve los recibirá y los medirá haciendo una selección aleatoria de las bases. Según los resultados que obtenga volverá a codificar los *qubits*, debido a que Bob debe recibir un mensaje a pesar de que Eve esté en medio, y por las leyes de la física cuántica, Eve no podrá copiar los estados cuánticos de Alice (teorema de no clonación). Además, como ya se ha explicado, al tener dos bases no ortogonales, Eve no podrá estar segura de qué estados provenían sus resultados, por lo que va a generar una perturbación detectable en el sistema. En el momento de intercambio de muestras, Bob y Alice se darán cuenta de la intervención mediante el QBER generado por Eve.

4.2. Análisis de riesgo

Para este tipo de ataque, donde Eve mide todos los *qubits*, hay cierta probabilidad de que aún así las muestras de Alice y Bob coincidan.

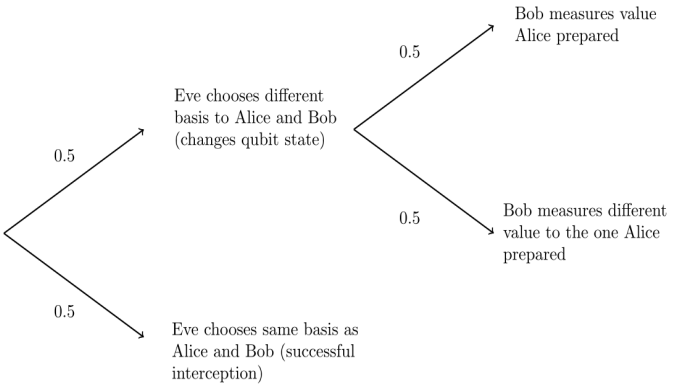

Fig. 3. Probabilidad de que Eve, habiendo interceptado y medido un *qubit* enviado por Alice, no sea detectada. Tiene una probabilidad de 0,5 de acertar la base, pero incluso fallando, hay una probabilidad de 0,5 de que Bob mida el bit que debería, debido a que, por ejemplo, $\langle 0|+\rangle$=0,5 [5].

Si Alice y Bob solo comparan 1 bit de su clave, hay una probabilidad de 0,75 de que los bits coincidan (Fig. 3.), y por tanto no se detecte el espionaje de Eve. En cambio, al comparar 2 bits, la probabilidad disminuye a $0,75^2 = 0,5625$. Por consiguiente, se puede expresar de forma general que [5]:

$$P(x) = 0,75^x \tag{3}$$

Donde $P(x)$ es la probabilidad de que Eve no introduzca errores y x es el número de bits intercambiados entre Bob y Alice a comparar. Si se contrastasen 15 bits, la probabilidad de que Eve no introduzca errores es del 1,3 %. Si a pesar de ello, Alice y Bob piensan que aún es demasiado arriesgado, comparando 50 bits hay una probabilidad del 0,00006 % de no introducir errores, es decir, que de esos 50 bits coincidan todos tras un ataque I-R [5].

5. Ataque PNS: Photon-Number Splitting

5.1. Definición

El protocolo BB84, así como otros protocolos de variable discreta, se basa en una suposición teórica clave: uso de fotones individuales. Como concepto teórico es tremendamente elegante, pero llevado a la práctica ha resultado ser un reto tecnológico. Para solventar este problema se propuso la idea de emplear pulsos coherentes débiles (*weak coherent states*), de manera que una fuente láser lo suficientemente atenuada con un número promedio de fotones bajo puede jugar el papel de fotón único. Es importante tener en cuenta que todos los fotones pertenecientes a un pulso están codificados en el mismo estado. Esto abrió las puertas a un ataque hacia un protocolo que se había planteado inicialmente como totalmente seguro.

El ataque PNS, en español división del número de fotones, es un tipo de ataque en el que Eve realiza una medida QND (*Quantum Non-Demolition*) del número de fotones contenidos en cada pulso, y de cada pulso multifotónico (pulso que contenga dos o más fotones), se quedará con un fotón y el resto se los enviará a Bob. Si el pulso contiene un solo fotón será bloqueado por Eve y no llegará a Bob. De esta manera, si Eve espera al intercambio de bases para medir los fotones que ha robado, podrá conocer al completo la clave privada generada sin introducir errores detectables, es decir, sin introducir QBER [6]. Se asume que Eve está restringida únicamente por las leyes de la física.

5.2. Medidas QND: Quantum Non-Demolition

Las medidas QND, o de no demolición cuántica, son aquellas medidas diseñadas para circunvalar las restricciones impuestas por el principio de incertidumbre de Heisenberg al realizar medidas repetidas de un estado cuántico. La principal idea de este tipo de medidas es monitorizar un solo observable que puede ser medido múltiples veces sin que su resultado cambie, idéntico al primer resultado obtenido sin perturbaciones exteriores del sistema. Para el caso del ataque PNS, este observable corresponderá al número de fotones dentro de un pulso coherente débil (WCS). Para lograr estas medidas se propuso la idea de emparejar el sistema medido a otro sistema "metro" mediante una interacción adecuada. De este modo, el estado cuántico de la partícula medida está determinado por la medida directa destructiva en la partícula "metro". Este tipo de medidas ya han sido probadas experimentalmente ([7], [8]), aunque el verdadero reto es lograr la resolución para medir fuentes de un solo fotón o pulsos coherentes débiles. Para esto se necesita un emparejamiento entre la fuente primaria y la fuente metro extremadamente fuerte, que, a pesar de ser muy difíciles de obtener, están al alcance de las técnicas de electrodinámica de cavidad cuántica [9], [10], [11].

5.3. Pulsos Coherentes Débiles

Los pulsos coherentes débiles, o WCSs, son la alternativa eficiente y de bajo coste que se encontró para sustituir a las fuentes de un solo fotón. Se crean me-

diante un pulso láser coherente atenuado lo suficiente como para conseguir un número de fotones promedio, μ, menor que 1. [12]

En el contexto de física cuántica, los denominados estados coherentes o cuasi-clásicos, también conocidos como estados Glauber, se consideran el análogo más cercano a una onda clásica, y se identifican con los autoestados del operador destrucción del oscilador armónico cuántico ($\hat{\alpha}\,|\alpha\rangle = \alpha\,|\alpha\rangle$), en los que las fluctuaciones del campo electromagnético son mínimas. Son estados del oscilador armónico cuántico caracterizados por ser paquetes gaussianos de mínima indeterminación ($\Delta x \Delta p = \frac{\hbar}{2}$). [13]

Los autoestados $|\alpha\rangle$ se conocen como estados de Fock, que son estados cuánticos con un número bien definido de partículas o cuantos, en este caso fotones [14]. Un pulso coherente también se define como la superposición de n fotones en estos estados de Fock [12]:

$$|\alpha\rangle = exp(-\frac{|\alpha|^2}{2}) \sum_n \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{4}$$

Donde $|\alpha|^2 = \mu$ es el número promedio de fotones en un pulso. Además, la probabilidad de encontrar n fotones en un pulso sigue una distribución de Poisson [12]:

$$P(n|\mu) = \langle n|\, \alpha^* \alpha\, |n\rangle = e^{-\mu} \frac{\mu^n}{n!} \tag{5}$$

En la práctica no hay forma de crear pulsos de un solo fotón sin crear a su vez pulsos de vacío ($n = 0$) o pulsos multifotónicos ($n \geq 2$) [12]. Los pulsos peligrosos y que permiten realizar un ataque PNS son los pulsos multifotónicos, por ello se trata de atenuar lo máximo posible los pulsos. Sin embargo, esto no es tan sencillo, porque el precio a pagar de atenuar demasiado los pulsos débiles es encontrarse con mayoritariamente pulsos vacíos, de modo que hay que encontrar un punto medio. Este equilibrio hace que realmente en la práctica sí que haya pulsos multifotónicos, de los cuales casi todos son pulsos con 2 fotones y en menor medida pulsos con más fotones.

5.4. Ataque a un protocolo BB84

Ya se ha explicado al principio de esta sección que la idea de un ataque PNS es quedarse con un fotón de los pulsos que sean multifotónicos y dejar pasar el resto de fotones. En caso de estar ante un pulso unifotónico, el pulso se bloquea y no llega a Bob. Para conseguirlo, Eve tendrá que desviar el canal cuántico entre Alice y Bob antes del ataque y hacer que todos los *qubits* pasen por uno alternativo controlado por ella.

Uno podría pensar, ¿no se da cuenta Bob de que no le llegan todos los estados? Bien, hay que recordar que uno de los canales cuánticos más empleados para experimentos es la fibra óptica. El principal motivo es reusar la fibra óptica ya existente, que es un modo de reducir drásticamente el potencial coste de realizar una infrastructura a gran escala para protocolos QKD.

En la implementación del protocolo BB84 con pulsos débiles, se conoce como tasa cruda de detección a la probabilidad de que Bob detecte un fotón por cada pulso enviado por Alice. Sin intervencion de Eve, esta tasa viene dada por [15]:

$$R_{raw}(\delta) = \sum_{n\geq 1} p_n(1 - (1 - \eta_{det}\eta_\delta)^n) \sim \eta_{det}\eta_\delta\mu \tag{6}$$

Donde η_{det} es la eficiencia del detector (normalmente 10 % para detectores como los SPAD), μ es el número promedio de fotones en el pulso, p_n es la probabilidad de encontrar n fotones en el pulso (dada por la ecuación (5)) y η_δ es la atenuación debida a las pérdidas en una fibra de longitud l [15]:

$$\eta_\delta = 10^{-\delta/10}, \delta = \alpha l[dB] \tag{7}$$

Donde α es el factor de atenuación de la fibra óptica. Normalmente $\alpha = 0{,}25$ dB/km [15].

Se asume que Eve tiene un canal cuántico sin pérdidas, es decir, $\eta_\delta = 1$, por lo que su tasa cruda después de un ataque PNS sería [15]:

$$R_{raw}(\delta) = \sum_{n\geq 2} p_n(1 - (1 - \eta_{det})^{n-1}) \sim \eta_{det}p_2 \tag{8}$$

Siendo p_2 la probabilidad de encontrar 2 o más fotones en el pulso. Por lo tanto, para que un ataque PNS no sea detectable y Bob reciba la cantidad de estados que esperaría tras las pérdidas por el canal, la tasa cruda después de un ataque PNS tiene que ser igual a la tasa cruda de detección de Bob. La longitud mínima que debe tener una fibra se puede despejar al igualar ambas tasas.

$$\eta_{det}\eta_\delta\mu = \eta_{det}p_2 \Longrightarrow l_{min} = \frac{10}{\alpha} log_{10}(\frac{\mu}{p_2}) \tag{9}$$

También se podría plantear la siguiente pregunta: ¿Bob no puede detectar si le llega más de un fotón en

un mismo pulso? Por lo general, no. Los detectores más comunes son los SPAD (*Single-Photon Avalanche Diode*), que tan solo detectan si al menos un fotón ha llegado al detector, pero no pueden contar la cantidad de fotones que reciben de un mismo pulso coherente débil.

5.5. Una posible solución: método de estados señuelo

Ante este nuevo problema surge una solución: el método de estados señuelo o *decoy state method*. Este nuevo método tiene como objetivo detectar un ataque PNS hacia un protocolo QKD en un canal con altas pérdidas, como es la fibra óptica. El método de estados señuelo se basa en que una de las personas legítimas que tratan establecer una clave segura envíe estados con diferentes valores de μ (número de fotones promedio en un pulso), llamados estados señuelo. Estos se intercalan aleatoriamente con los estados señal, que son los que conformarán la clave privada. Para identificar un posible ataque PNS se comprueba el alcance de los estados señuelo. Para las simulaciones se tomarán solo dos valores de μ, el $\mu_{señuelo}$ y el $\mu_{señal}$, y se puede enfocar de dos maneras [16]:

La primera sería usar $\mu_{señal} < \mu_{señuelo} < 1$. En este caso, como $\mu_{señuelo}$ es mayor, los pulsos señuelo contendrán más cantidad de pulsos multifotónicos, y según el esquema de ataque PNS, llegarán a Bob estadísticamente más estados señuelo que estados señal. En otras palabras, el alcance de los estados señuelo será sospechosamente alto, lo que viene a indicar que Eve ha realizado un ataque PNS.

El caso contrario, $\mu_{señuelo} < \mu_{señal} < 1$, es análogo. Como ahora $\mu_{señuelo}$ es pequeño, la mayoría de los pulsos de estados señuelo serán bloqueados ya que contendrán un fotón, y a Bob le llegarán mayoritariamente pulsos de estados señal. Ahora el alcance de los estados señuelo será llamativamente bajo.

6. El protocolo SARG04

6.1. Definición

El protocolo SARG04 fue propuesto por Scarani, Acín, Ribordy y Gisin en el año 2004, de ahí su nombre. El tratamiento de los estados cuánticos es completamente idéntico al BB84, al igual que su preparación y medición. La gran diferencia surge en la parte clásica del protocolo. Por una parte, cambiará la manera de codificar y descodificar los bits clásicos. Por otra parte, en vez de intercambiar las bases empleadas por Alice y Bob durante el protocolo (como sucede en el BB84), Alice enviará por el canal clásico conjuntos de estados que Bob usará para destilar la clave compartida privada. Tras este proceso realizarán la corrección de errores y amplificación de la privacidad de la misma manera que en el protocolo BB84. Este es un protocolo específicamente diseñado como una mejora del BB84 frente a ataques PNS gracias a la diferencia en la destilación de la clave.

6.2. Preparación y funcionamiento en detalle

La parte del protocolo relacionada con el tratamiento de los estados cuánticos es idéntica al protocolo BB84. El momento en el que SARG04 y BB84 se bifurcan es a la hora de codificar y descodificar los bits clásicos. El bit 0 se codifica como $|0\rangle$ o $|1\rangle$, es decir, con los estados que conforman la base Z, donde se escoge uno de ellos de forma aleatoria, y el bit 1 se codifica con los estados de las base X, $|+\rangle$ y $|-\rangle$ [17].

Una vez que Alice ha codificado la clave privada y la ha enviado por el canal cuántico, Bob mide esos estados que recibe de la misma manera que en el protocolo BB84, obteniendo la clave cruda. Sin embargo, Bob ahora tendrá que "adivinar" los estados que envió inicialmente Alice para luego poder destilar la clave correctamente. Bob asume que siempre acierta la base con la que debería medir. De esta manera, si Bob ha medido un estado en la base X y ha obtenido como resultado el bit 0, sabe que si ha acertado la base, el estado que envió Alice tuvo que ser $|+\rangle$. Del mismo modo, si mide con la base Z y obtiene el resultado 1, deduce que Alice ha mandado el estado $|1\rangle$.

Para el proceso de destilación, Alice tendrá que enviar unos conjuntos de estados para que Bob pueda decidir con qué bits quedarse de sus resultados y así crear una clave compartida. Alice va a enviar a Bob a través del canal público un conjunto de estados conformados por dos posibilidades: un estado será con el que realmente ha codificado el bit de la clave privada, mientras que el otro será uno de la base contraria. Por ejemplo, si Alice codifica el bit con $|+\rangle$, el segundo será o $|0\rangle$ o $|1\rangle$. En total, Alice enviará uno de los siguiente cuatro conjuntos por cada *qubit* enviado: $S_1 : \{|0\rangle, |+\rangle\}$; $S_2 : \{|0\rangle, |-\rangle\}$; $S_3 : \{|1\rangle, |+\rangle\}$; $S_4 : \{|1\rangle, |-\rangle\}$. Esto explica por qué SARG04 no es vulnerable a un ataque PNS. Como los conjuntos enviados por Alice contienen estados no ortogonales, a Eve ya no le resulta útil robar un fotón para conocer la clave final, ya que en muchos casos no puede estar segura del estado inicial preparado por Alice.

Con los conjuntos enviados por Alice y los estados que Bob ha tratado de adivinar, Bob podrá destilar

su clave. Si el estado deducido por Bob había sido el $|0\rangle$ y el conjunto enviado por Alice es el $\{|0\rangle, |+\rangle\}$, Bob no podrá estar seguro si se ha equivocado o no, debido a que ambos estados podrían encajar con su medida. Ahora bien, si el estado deducido es $|1\rangle$ y Alice envía el conjunto $\{|0\rangle, |+\rangle\}$, Bob sabrá con certeza que se ha equivocado en su selección de bases, porque $|\langle 0|1\rangle|^2 = 0$, y añadirá a su clave destilada el bit 1 correspondiente a la base X, que sería la base correcta.

7. THA: Trojan Horse Attack

7.1. Definición

Como bien se sabe, la mayoría de los problemas de seguridad relacionados con los protocolos QKD se deben fundamentalmente a las limitaciones de la tecnología disponible. El ataque del caballo Troyano, o *Trojan Horse Attack* es un ataque hacia los elementos ópticos relacionados con los estados cuánticos para llevar a cabo un protocolo QKD. Eve envía un pulso brillante a alguno de los aparatos de preparación y/o medida empleados por Alice y Bob a través de la fibra ótpica, pero como una gran proporción del pulso se acabará perdiendo al viajar por la fibra, tendrá que ser lo suficientemente brillante para hacer el viaje de ida y vuelta y que queden fotones. Un pulso brillante es un pulso con un μ grande, que en experimentos con este ataque toma valores del orden de $\mu \sim 10^6$ [18].

Según las transformaciones que haya sufrido el pulso, Eve puede deducir información acerca de la clave final, como la base empleada por Alice para codificarla o la selección de bases realizada por Bob. Además, el ataque tendrá que efectuarse en una ventana de tiempo en la que Bob no esté activamente midiendo ningún estado.

7.2. Ataque a un protocolo SARG04

Cuando se habla sobre un ataque de caballo Troyano, normalmente no se valora inferir sobre los aparatos de preparación de los estados de Alice. Esto se debe a que Alice tiene mayor facilidad para detectar un THA: simplemente tendrá que instalar un detector de monitorización pasivo. Esto se logra con un detector apropiado (o un conjunto de ellos) que vaya midiendo la luz que llega al subsistema de Alice y salte una alarma en caso de que sobrepase un cierto límite preestablecido. Por el contrario, esta medida no puede ser adoptada de una manera sencilla por Bob. Introducir un sistema de monitorización pasivo puede introducir atenuación no deseada en los estados que recibe por el canal cuántico, que de por sí son bastante débiles [18]. Esto haría que la tasa de creación de clave secreta se reduzca incluso más (recordar que estas tasas son bajas debido a la atenuación del canal y el proceso de destilación).

Esta es la principal razón por la que se dice que el protoclo BB84 es "resistente" frente al THA. Si se tiene en cuenta que el ataque se realiza sobre los aparatos de Bob, será de interés saber las bases de medida de los estados cuánticos que elige. Como en el protocolo BB84 las bases se anuncian públicamente, esa información no es ningún secreto. En cambio, se ha visto que para el protocolo SARG04 la selección de bases codifica el bit de la clave privada. De modo que, si Eve sabe la base de medida de Bob, conoce también el valor del bit de la clave final. Además, sabrá qué bits no se usan finalmente tras el proceso de destilación de la clave porque Bob le tiene que comunicar esta información a Alice a través del canal público.

7.3. Una posible solución

Si Eve no tiene mucho cuidado, un ataque THA con un pulso brillante puede generar *afterpulsing*, o pulsos residuales, en los detectores SPAD de Bob. Este efecto eleva el ruido en los detectores, es decir, se registran *clicks* como si hubiese llegado un pulso pero realmente no ha llegado nada. Si la tasa de ruido es sospechosamente elevada, esto le indicaría a Bob que Eve está intentando intervenir en el sistema.

Suponiendo que Eve consigue generar pulsos brillantes que no eleven demasiado el ruido de los SPAD, una posible solución que podría emplear Bob sería monitorizar en tiempo real los detectores SPAD. Este sistema de monitorización se basa en la acumulación de estadísticas de los tiempos entre eventos de detección consecutivos, así como en la extracción en tiempo real de la pospulsación y de la eficiencia general de los detectores, mediante modelos matemáticos ajustados a los datos medidos [19].

8. Simulación en Qiskit de los protocolos, ataques y soluciones

8.1. ¿Qué es Qiskit?

Qiskit es un kit de desarrollo de software creado por IBM para trabajar con ordenadores cuánticos a nivel de circuitos, pulsos y algoritmos. Qiskit proporciona herramientas para crear y manipular programas cuánticos y ejecutarlos en dispositivos cuánticos prototipo en IBM Quantum Experience. También existe la opción de simular el comportamiento de los ordenadores cuánticos a nivel local. La versión principal

de Qiskit usa el lenguaje de programación Python [20]. Para las simulaciones se ha empleado la versión 1.2.4 de Qiskit.

8.2. Protocolo BB84, ataque PNS y método de estados señuelo

8.2.1. Protocolo BB84

Para simular el protocolo BB84 se han empleado las funciones que aparecen en el libro de Qiskit [5]. Se ha creado una función para codificar el mensaje de Alice mediante estados cuánticos; una función para que Bob mida esos estados cuánticos, primero utilizando *AerSimulator()*, que simula el comportamiento real de un ordenador cuántico, y luego con *generate_preset_pass_manager()*, para la simulación en un ordenador cuántico real; una función para realizar la destilación de la clave compartida entre Bob y Alice; una función para tomar una muestra de la clave destilada y eliminarla de dicha clave; una función para calcular el QBER del protocolo; y, por último, una función para calcular la probabilidad de encontrar n fotones en un pulso coherente débil, según la ecuación correspondiente (5).

El código de las funciones es trivial y basado en la teoría que ya se ha explicado acerca del protocolo BB84, excepto quizás la codificación del mensaje y su medida. Para esta parte se han empleado funciones específicas de la librería Qiskit, mediante la importación del módulo llamado *QuantumCircuit* y vale la pena explicarlo en detalle.

```
62
63  #Función que usa Alice para codificar el mensaje que quiere enviar [1]
64  def encode_message(bits,bases):
65      message = []
66      #longitud del mensaje de Alice
67      n = len(bases)
68
69      for i in range(n):
70          #Crear un qubit. Por defecto está en estado |0>
71          qc = QuantumCircuit(1,1)
72          #codificar en estado |0>
73          if (bases[i] == bits[i] == 0):
74              pass
75          #codificar en estado |1>
76          elif (bases[i] == 0 and bits[i] == 1 ):
77              qc.x(0)
78          #codificar en estado |+>
79          elif (bases[i] == 1 and bits[i] == 0):
80              qc.h(0)
81          #codificar en estado |->
82          elif (bases[i] == bits[i] == 1):
83              qc.x(0)
84              qc.h(0)
85          qc.barrier()
86          message.append(qc)
87
88      return message
89
```

Fig. 4. Extracto de código con la función para codificar el mensaje en los estados cuánticos que Alice enviará a través del canal cuántico. Si la pareja de bit aleatorio y base aleatoria es 00, se codifica en estado $|0\rangle$, si es 10 en $|1\rangle$, si es 01 en $|+\rangle$ y si es 11 en $|-\rangle$, usando las puertas lógicas X y H para conseguir los estados.

En la Fig. 4 se observa que Alice emplea la función *qc=QuantumCircuit(1,1)*. Esto se usa para crear un solo *qubit* y su correspondiente registro clásico para guardar el resultado de la medida de ese *qubit*. Según la combinación de bits y bases de Alice, ambas cadenas creadas con un *randint(2, size=n)*, se codificará en un estado u otro. Si la combinación es 00, se codifica en estado $|0\rangle$, pero como este es el estado por defecto de los *qubits* creados con el *QuantumCircuit()*, no hace falta hacer nada "extra". Si la combinación es 10, se codifica en estado $|1\rangle$. Este estado se puede obtener mediante una puerta lógica, que no es más que una matriz unitaria para la transformación de estados de los *qubits*.

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

Si la combinación es 01 se codifica en estado $|+\rangle$. Este estado se consigue con una puerta Hadamard, también denominada puerta H.

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Si la combinación es 11 se codifica en estado $|-\rangle$. Se logra con una puerta X seguida de una puerta H.

$$HX|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

```
#Función que usa Bob para medir los qubits que le llegan por el canal cuántico [2]
def measure_message(message, base):
    #longitud del mensaje a medir
    n = len(base)
    measurements=[]

    for i in range(n):
        #Si el bit aleatorio de la cadena de bases es 0, se mide en base Z,
        #que es la base computacional por defecto
        if(base[i]==0):
            message[i].measure(0,0)
        #Si el bit es 1, se mide en base X. Se logra pasar desde la base Z
        #hasta la base X mediante una puerta de Hadamard
        else:
            message[i].h(0)
            message[i].measure(0,0)
        #Simulador en un ordenador clásico del comportamiento de un ordenador cuántico
        aer_sim = AerSimulator()
        result = aer_sim.run(message[i], shots=1, memory=True).result()
        measured_bit = int(result.get_memory()[0])
        measurements.append(measured_bit)

    return measurements
```

Fig. 5. Extracto de código con la función que emplea Bob para medir los estados que le llegan por el canal cuántico. Si en su cadena de bases aleatoria tiene el valor 0, medirá en base Z, y si tiene el 1, en base X. Usará la puerta H si debe medir en base X.

Ya conociendo el mecanismo de la función de codificación de Alice, la función de medida de Bob es sencilla de entender (Fig. 5.). Si Bob decide de forma aleatoria medir en la base Z, no tiene que aplicar ninguna puerta, porque de por sí la base Z ya es la base computacional y un ordenador cuántico mide en ella. Los estados $|0\rangle$ y $|1\rangle$ dan como resultado el bit 0

y 1, respectivamente. Si Bob decide medir en la base X, tendrá que aplicar una puerta de Hadamard antes de medir porque, si ha escogido la base correcta, la medida de $|+\rangle$ y $|-\rangle$ da como resultado los bits 0 y 1, y si no se aplicase la puerta antes, sería como si Bob fallase constantemente la selección de bases y midiese en la base Z siempre. La puerta H aplicada sobre $|+\rangle$,

$$H|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix}1\\1\end{bmatrix}+\frac{1}{2}\begin{bmatrix}1\\-1\end{bmatrix} = \frac{1}{2}\left(\begin{bmatrix}1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}\right)+\frac{1}{2}\left(\begin{bmatrix}1\\0\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix} = |0\rangle$$

Y de manera análoga,

$$H|-\rangle = |1\rangle$$

8.2.2. Ataque PNS

Un ataque PNS es fácil de simular. La intervención de Eve hace que tan solo lleguen a Bob los pulsos con dos o más fotones. Por ello, se calcula las probabilidades de tener 0 fotones en un pulso, 1 fotón y 2 o más fotones. Se normalizan las probabilidades de tener 1 y 2 o más, ya que con los pulsos vacíos no se transmite información. Estas probabilidades normalizadas se multiplican al número inicial de bits que pretendía enviar Alice. Se seleccionan tan solo los bits que venían de pulso multifotónico, como consecuencia del ataque (Fig. 6.).

```
#Probabilidad de encontrar 0 fotones
P_0 = probability_photons(0, mu)
#Probabilidad de encontrar 1 fotón
P_1 = probability_photons(1, mu)
#Probabilidad de encontrar 2 o más fotones
P_2_or_more = 1 - P_0 - P_1
#Normalizar la probabilidad de encontrar pulso con 1 fotón ya que
# los pulsos sin fotones no se emplean
P_1_nor = (P_1)/(P_1 + P_2_or_more)
#Normalizar la probabilidad de encontrar pulso multifotónico ya que
#los pulsos sin fotones no se emplean
P_2_or_more_nor = (P_2_or_more)/(P_1 + P_2_or_more)
#Cantidad de bits que provendrán de un pulso multifotónico
n_pns = int(round(P_2_or_more_nor*n,0))
```

Fig. 6. Extracto de código donde se realiza el cálculo de las probabilidades de tener un pulso con 0 fotones, 1 fotón y 2 o más, para un determinado μ. También se calcula la cantidad de fotones que quedará tras un ataque PNS, que será los bits iniciales multiplicados por la probabilidad de tener un pulso multifotónico.

También se tendrá que calcular la distancia mínima a partir de la cual se puede aplicar un ataque PNS sin ser detectable, como muestra la ecuación (9). Esta distancia, que se traduce en longitud de la fibra óptica, será la que se emplee para simular el ataque. Finalmente, se multiplica la cantidad de bits que han quedado que provenían de pulsos multifotónicos por la tasa después del ataque, que es lo que le llega realmente a Bob.

8.2.3. Método de estados señuelo

Los estados señuelo se identificarán con los números 2 y 3, creando una analogía con los bits 0 y 1. Las cadenas de bits y bases de Alice y Bob estarán ahora conformadas por bits que identifican los pulsos señal (0 y 1) y los pulsos señuelo (2 y 3). Esto no tiene ninguna diferencia importante con respecto a la simulación del protocolo BB84.

```
#Bits que Alice quiere enviar que acabarán siendo estados señuelo
bits_decoy_Alice = randint(low=2, high=4, size=n_2_decoy_fibra)
bits_decoy_Alice = bits_decoy_Alice.tolist()
#Bits que Alice quiere enviar que acabarán siendo estados señal.
#Estos bits son los que conformará la clave privada compartida
bits_signal_Alice = randint(2, size=n_2_signal_fibra)
bits_signal_Alice = bits_signal_Alice.tolist()
#Bits totales que Alice codifica
bits_method_Alice = bits_signal_Alice + bits_decoy_Alice
#Se mezclan los bits de estados señuelo y señal de manera aleatoria
random.shuffle(bits_method_Alice)
bits_method_Alice = np.array(bits_method_Alice)
```

Fig. 7. Extracto de código para crear las cadenas de bits, representando los estados señal con 0 y 1, y los estados señuelo con un 2 o un 3, por analogía.

Alice también se guarda las posiciones de los estados señuelo. Una vez sepa estas posiciones, vuelve a cambiar las cadenas de bases y bits para que contengan 0 y 1 únicamente, transformando 2 a 0 y 3 a 1. Esto se hace para reusar las funciones de codificación del mensaje de Alice y la de medida de Bob de la parte cuántica del protocolo. Además, los resultados de Bob tras la medida solo pueden ser bits 0 o 1, como era de esperar. Una vez que se tengan los resultados de Bob, se cambian los bits que estén en las posiciones que Alice se había guardado. El 0 se transforma en un 2 y el 3 en un 1, de nuevo, simplemente por analogía. Con este fin se ha creado una función especial para la destilación de la clave, donde se hace el cambio de bits y se destila la clave a la vez. Toda la parte de estados señuelo se podría hacer sin necesidad de cambiarlos, saber sus posiciones es suficiente, pero de esta manera es más visual y fácil de seguir.

Por otro lado, se crea una función para calcular el alcance de los estados señuelo para poder observar su estadística. En caso de haber ataque PNS, este valor será el que haga saltar las alarmas de Alice y Bob.

```
#Función para calcular alcance de los estados señuelo
def yield_decoy_method(Bob_key):
    n = len(Bob_key)
    yield_decoy = 0

    for i in range(n):
        #Los estados señuelo toman valores 2 y 3 para diferenciarse de
        #los estados de la señal, por lo que se cuenta cuántos hay en la
        #clave de Bob
        if(Bob_key[i]==2 or Bob_key[i]==3):
            yield_decoy+=1
    #El alcance de los estados señal será la resta de los estados totales
    #que le llegan a Bob menos el alcance de estados señuelo
    yield_signal = n - yield_decoy
    #Los alcances se calculan como la cantidad de estados que recibe Bob
    #de cada tipo entre la cantidad total de estados que recibe
    return yield_decoy/n, yield_signal/n
```

Fig. 8. Extracto de código con la función que se usa para calcular la cantidad de estados señuelo y de estados señal que llegan a Bob, contando la cantidad de doses y treses que se encuentran en su clave.

8.3. Protocolo SARG04 y ataque THA

8.3.1. Protocolo SARG04

La parte cuántica del protocolo SARG04 es idéntica al protocolo BB84, por lo que las funciones para codificar el mensaje de Alice y para medir los estados cuánticos son muy parecidas, con la diferencia de que Alice ya no necesita una cadena de bases aleatoria. Ahora, si el bit que desea enviar es 0, se codifica en base Z, mientras que si el bit es 1, se codifica en base X.

También habrá una función independiente para la selección aleatoria de bases de Bob, donde esta vez se elige entre los caracteres "X" o "Z", que es análogo a elegir 1 o 0 de manera aleatoria. La función de medida de Bob es igual que para el protocolo BB84, teniendo en cuenta que ahora la base es una cadena de caracteres.

Para el proceso de destilación se han creado varias funciones. Una será con la que Bob trata de adivinar qué estado ha enviado Alice.

```python
#Función con la que Bob intenta adivinar qué estado ha enviado Alice.
#Siempre parte de la misma idea: si Bob hubiese acertado la base,
#el estado enviado por Alice lo puede deducir sabiendo la base que ha
# escogido y el resultado que ha obtenido tras la medida
def states_guess(bases, results):
    #Longitud bases de Bob
    n = len(bases)
    states = []

    for i in range(n):
        #Si ha medido con la base Z y ha obtenido el bit 0,
        #el estado de Alice tendría que ser el |0>
        if(bases[i]=="Z" and results[i]==0):
            states.append("0")
        #Si ha medido con la base Z y ha obtenido el bit 1,
        #el estado de Alice tendría que ser el |1>
        elif(bases[i]=="Z" and results[i]==1):
            states.append("1")
        #Si ha medido con la base X y ha obtenido el bit 0,
        #el estado de Alice tendría que ser el |+>
        elif(bases[i]=="X" and results[i]==0):
            states.append("+")
        #Si ha medido con la base X y ha obtenido el bit 1,
        #el estado de Alice tendría que ser el |->
        else:
            states.append("-")

    return states
```

Fig. 9. Extracto de código con la función que Bob usa para tratar de adivinar qué estado ha enviado Alice, en función del resultado que él ha obtenido y suponiendo que siempre acerta la base. Por ejemplo, el resultado 0 medido en base Z correspondería al estado $|0\rangle$.

Otra función creará los conjuntos de estados que Alice manda por el canal público, con el procedimiento que se ha comentado en la sección del protocolo SARG04.

Para finalizar el proceso de destilación, se crea una función que con los estados que ha tratado de adivinar Bob y los conjuntos que ha mandado Alice por el canal público, Bob construya su clave cruda y le comunique a Alice las posiciones de los bits con los que se ha quedado.

El resto del proceso es completamente idéntico al protocolo BB84, incluyendo la reducción de estados que recibe Bob por las pérdidas de la fibra óptica. Como Eve no realiza un ataque PNS, la tasa de clave cruda viene dada solo por la ecuación (6).

8.3.2. Ataque THA

Dado que el THA es un ataque que se centra en los dispositivos de Alice y Bob, en lugar de simular el ataque en sí, se ha optado por simular sus consecuencias. En la sección de ataque de caballo Troyano se ha justificado que el objetivo va a ser Bob, de modo que, si Eve logra realizar un ataque THA sobre el preparador de bases, el protocolo SARG04 se vuelve inseguro. Teniendo las bases de Bob se consigue la clave final. Esto se simula con una función que crea la clave a partir de la selección de bases de Bob.

```python
#Función con la que, si Eve realiza un ataque THA al selector
#de bases de Bob, puede acabar sabiendo la clave final compartida
def key_Eve(bases_Bob, positions):
    n = len(bases_Bob)
    key = []
    final_key=[]

    for i in range(n):
        #Si las base elegida por Bob ha sido Z, Eve se guarda el bit 1,
        # ya que Bob se queda los bits cuando se ha equivocado de
        #base al medir
        if(bases_Bob[i]=="Z":
            key.append(1)
        #Si las base elegida por Bob ha sido X, Eve se guarda el bit 0,
        #ya que Bob se queda los bits cuando que se ha equivocado de
        #base al medir
        else:
            key.append(0)
    #Eve puede saber con qué posiciones quedarse ya que Bob le tiene que
    #proporcionar esta información a Alice a través del canal público
    for i in positions:
        final_key.append(key[i])

    return final_key
```

Fig. 10. Extracto de código donde se muestra la función que emplea Eve para conocer la clave final, sabiendo las bases elegidas por Bob mediante un ataque THA. Bob guarda los bits cuando se equivoca de base, por lo que Eve se guarda el bit contrario a la base que haya robado.

9. Resultados y discusión

Se han realizado dos programas interactivos donde se permite al usuario introducir por pantalla algunos parámetros según sus necesidades.

Para el programa de protocolo BB84, ataque PNS y estados señuelo, al ejecutar el programa se pide introducir el número promedio de fotones del pulso débil

μ, la eficiencia del detector η_{det}, el factor de atenuación de la fibra óptica α y el número de bits que desea enviar Alice inicialmente n, teniendo en cuenta que debido a la atenuación del canal este último parámetro debe ser grande. La longitud de la fibra no es introducida por el usuario, ya que se calcula automáticamente. Este valor corresponde a la longitud mínima a partir de la cual puede llevarse a cabo un ataque PNS, en función de los valores de α y μ introducidos. Para el método de estados señuelo se solicita introducir $\mu_{señuelo}$, ya que debe ser diferente que el $\mu_{señal}$, y para el $\mu_{señal}$ se usa el parámetro introducido al inicio del programa. También se debe especificar el porcentaje de estados señuelo y estados señal que se desea tener.

Por otra parte, el programa para el protocolo SARG04 y ataque THA, permite al usuario introducir el número promedio de fotones del pulso débil μ, la eficiencia del detector η_{det}, el factor de atenuación de la fibra óptica α, el número de bits que desea enviar Alice inicialmente n y la longitud de la fibra l.

9.1. Protocolo BB84, ataque PNS y método de estados señuelo

Se ha comenzado usando el simulador *AerSimulator()* de un ordenador cuántico para comprobar que los resultados son correctos y se corresponden con la teoría explicada. Al introducir los parámetros , el programa devuelve un mensaje de validación así como la longitud de la fibra óptica a la que se puede empezar a realizar ataques PNS. En esta simulación se han usado los valores: $\alpha = 0{,}25$ dB/km; $\mu = 0{,}1$; $\eta_{det} = 0{,}1$; $n = 10^6$

Alice y Bob piensan que el protocolo BB84 es seguro, pero realmente Eve realiza un ataque PNS sin introducir QBER y conoce la clave final compartida, tal y como se esperaba.

```
¡Ataque PNS exitoso!

Clave final de Alice = [1, 1, 1, 0, 0, 0, 1, 0]
Clave final de Bob  = [1, 1, 1, 0, 0, 0, 1, 0]
Clave robada por Eve = [1, 1, 1, 0, 0, 0, 1, 0]

Longitud de la clave final = 8

QBER = 0.0 %
```

Fig. 11. Resultados mostrados por pantalla tras ejecutar la simulación, donde se ve que Eve ha logrado robar la clave de Alice y Bob sin introducir QBER en el sistema. Parámetros empleados: $\alpha = 0{,}25$ dB/km; $\mu = 0{,}1$; $\eta_{det} = 0{,}1$; $n = 10^6$.

El protocolo es inseguro para esas características, por lo que se pregunta al usuario si desea usar el método de estados señuelo. Si responde que sí, se pide especificar las características del pulso señuelo a emplear. En este caso los valores introducidos han sido: $\mu_{señuelo} = 0,5$; $\%_{señal} = 70$; $\%_{señuelo} = 30$.

Por último, se realiza el método de estados señuelo y se obtienen los siguientes resultados:

```
¡Ataque PNS detectado con método de estados señuelo!

Clave final de Alice = [0, 1, 1, 0, 0, 1]
Clave final de Bob = [0, 1, 1, 0, 0, 1]
Clave robada por Eve = [0, 1, 1, 0, 0, 1]

Longitud de la clave sin estados señuelo= 6
Longitud de la clave con estados señuelo= 218

QBER = 0.0 %

¡El alcance de los estados señuelo es sospechoso!

Alcance estados señuelo = 97.86 %
Alcance estados señal = 2.14 %

Este es el alcance que se espera:

Alcance estados señuelo sin ataque PNS = 67.92 %
Alcance estados señal sin ataque PNS  = 32.08 %
```

Fig. 12. Resultados de la simulación del método de señuelos, donde se observa un alcance de los estados señuelo sospechosamente elevado, indicando que Eve ha realizado un ataque PNS. Parámetros empleados: $\mu_{señal} = 0,1$; $\mu_{señuelo} = 0,5$; $\%_{señal} = 70$; $\%_{señuelo} = 30$.

Tal y como muestra la Fig. 12., gracias al método de estados señuelo se ha conseguido detectar un ataque PNS. Se ve claramente que el alcance de los estados señuelo es demasiado alto en comparación con el alcance que se esperaría sin ataque PNS. En este caso se ha empleado que $\mu_{señuelo} = 0{,}5$, mientras que $\mu_{señal} = 0{,}1$. Como Eve solo deja pasar pulsos multifotónicos en un ataque PNS, esto modifica el alcance de los estados señuelo a causa de que cuanto mayor μ, mayor cantidad de pulsos multifotónicos. Eve acaba bloqueando casi todos los pulsos señal, puesto que no es capaz de diferenciar entre estados señal y estados señuelo, y esto hará que casi no lleguen pulsos señal a Bob, modificando por completo la estadística del alcance.

Normalmente es más práctico, a la hora de usar el método de estados señuelo, hacer que $\mu_{señuelo} < \mu_{señal} < 1$, porque como los estados señuelo no se usan para crear la clave final, de esta manera se asegura la seguridad frente a este tipo de ataques sin perder demasiada tasa de creación de la clave final. Para otra simulación, esta vez con $\mu_{señal} = 0{,}5$, $\mu_{señuelo} = 0{,}1$, $n = 10^6$ y manteniendo el resto de parámetros iguales, se obtiene que:

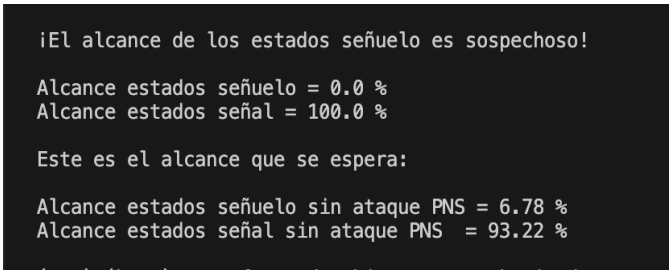

Fig. 13. Resultados de la simulación del método de señuelos, donde se observa un alcance de los estados señuelo sospechosamente pequeño, indicando que Eve ha realizado un ataque PNS. Parámetros empleados: $\mu_{señal} = 0,5$; $\mu_{señuelo} = 0,1$; $\%_{señal} = 70$; $\%_{señuelo} = 30$.

Esta vez pasa lo contrario. Como los pulsos señuelo son menos intensos que los pulsos señal, es decir, su μ es más pequeño, casi todos serán bloqueados por Eve en un ataque PNS, alterando la estadística de los alcances. Con ataque PNS no llega ningún estado señuelo pues, teniendo en cuenta que casi todos los bloquea Eve y en este caso solo hay un 30 % de ellos, Bob no los recibe.

Se ha usado un 70 % de estados señal y un 30 % de estados señuelo. No hay un porcentaje establecido, puesto que depende en gran medida de las características del protocolo, canal y detectores. No obstante, estos parámetros resultan razonables para equilibrar seguridad y eficiencia a la hora de generar la clave final.

Por último, se ha realizado la simulación completa en un ordenador cuántico real de IBM y concuerda con lo que se esperaba, de nuevo obteniendo un alcance sospechoso en los estados señuelo. En el ordenador real se han usado muchos menos *qubits* a consecuencia del tiempo que lleva generar cada estado y medirlo. Hay que tener en cuenta que cada estado individual del mensaje es un circuito cuántico en sí mismo. Un circuito cuántico es un cojunto de operaciones lógicas que se realizan sobre un *qubit*.

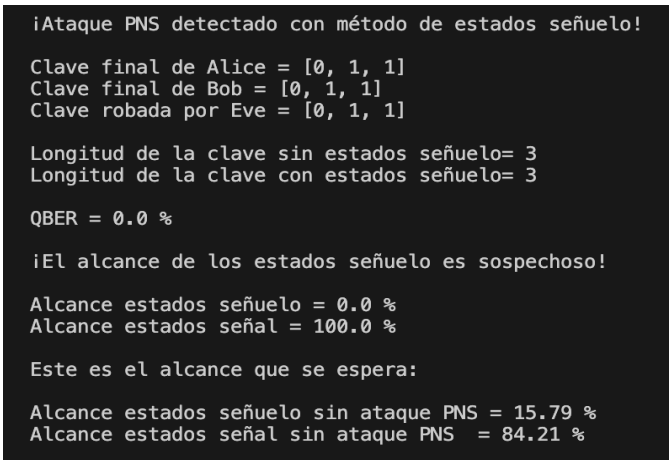

Fig. 14. Resultados de la simulación del método de señuelos en un ordenador cuántico real, donde se observa un alcance de los estados señuelo sospechosamente elevado, indicando que Eve ha realizado un ataque PNS. Parámetros empleados: $\mu_{señal} = 0,5$; $\mu_{señuelo} = 0,1$; $\%_{señal} = 70$; $\%_{señuelo} = 30$; $\alpha = 0{,}25$ dB/km; $\eta_{det} = 0{,}1$; $n = 5 \cdot 10^3$

9.2. Protocolo SARG04 y ataque THA

Una vez más, se empieza con el simulador de ordenadores cuánticos *AerSimulator()*. Se puede probar la simulación con diferentes combinaciones y parámetros. En este caso, escogiendo $\alpha = 0{,}25$ dB/km, $\mu = 0{,}1$, $\eta_{det} = 0{,}1$, $n = 10^6$ y $l = 80$ km se consiguen los siguientes resultados:

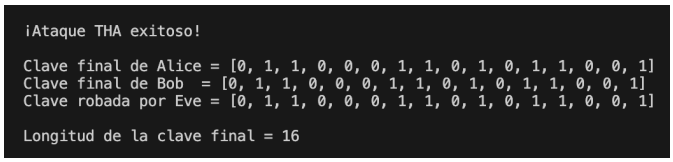

Fig. 15. Resultados de una simulación del protocolo SARG04, donde Eve ha robado la selección de bases de Bob y ha logrado conocer la clave final. Parámetros empleados: $\alpha = 0{,}25$ dB/km; $\mu = 0{,}1$; $\eta_{det} = 0{,}1$; $n = 10^6$; $l = 80$ km.

Se ha verificado que, efectivamente, el código para el protocolo SARG04 funciona y las claves de Alice y Bob coinciden. Además, si Eve consigue robar las bases de medida de Bob, se demuestra explícitamente que tal y como ya se había propuesto, Eve consigue conocer la clave final y la seguridad del protocolo está comprometida.

Por último, se ha simulado en un ordenador cuántico real de IBM para verificar que las simulaciones son correctas y, de nuevo, se obtienen los resultados esperados. Los parámetros empleados han sido:

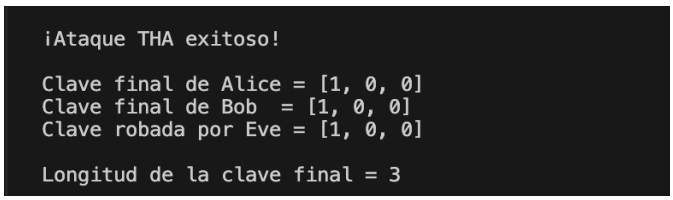

Fig. 16. Resultados de una simulación del protocolo SARG04 en un ordenador cuántico real, donde Eve ha robado la selección de bases de Bob y ha logrado conocer la clave final. Parámetros empleados: $\alpha = 0{,}25$ dB/km; $\mu = 0{,}1$; $\eta_{det} = 0{,}1$; $n = 5 \cdot 10^3$; $l = 20$ km.

10. Conclusiones

A lo largo de este trabajo se ha demostrado que, aunque los protocolos de Distribución Cuántica de Claves (QKD) como BB84 y SARG04 ofrecen seguridad

teórica basada en principios fundamentales de la mecánica cuántica, su implementación práctica puede ser vulnerable a diversos ataques. El protocolo BB84 resulta susceptible al ataque PNS , pero esta debilidad puede ser mitigada mediante el uso del método de estados señuelo, permitiendo detectar el ataque incluso en condiciones de canal con altas pérdidas. El protocolo SARG04, diseñado específicamente para mejorar la resistencia frente a ataques PNS, puede ser comprometido por un ataque de tipo THA. En este contexto, se ha propuesto la monitorización en tiempo real de los detectores como una posible solución para identificar anomalías generadas por estos ataques.

Finalmente, las simulaciones realizadas con Qiskit han permitido validar los resultados teóricos, evidenciando la eficacia de las estrategias de defensa planteadas y resaltando la importancia de considerar las limitaciones tecnológicas reales para una implementación segura de los protocolos QKD.

11. Referencias

[1] Nielsen, M. A., & Chuang, I. L. (2000). *Quantum Computing and Quantum Information* (pp. 582–588). Cambridge University Press.

[2] Rozenman, G., Kundu, N. K., Liu, R., Zhang, L., Maslennikov, A., Reches, Y., & Youm, H. (2023). *The quantum internet: A synergy of quantum information technologies and 6G networks. IET Quantum Communication, 4.*

[3] Función hash. Wikipedia. Recuperado de: `https://es.wikipedia.org/wiki/Funci%C3%B3n_hash`

[4] Bennett, C., & Brassard, G. (1984). *Quantum cryptography: Public key distribution and coin tossing.* In Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing (pp. 175–179). IEEE.

[5] Qiskit Textbook – Quantum Key Distribution. Recuperado de: `https://github.com/Qiskit/textbook/blob/main/notebooks/ch-algorithms/quantum-key-distribution.ipynb`

[6] A. Ashkenazy, Y. Idan, D. Korn, D. Fixler, B. Dayan, and E. Cohen, *Photon Number Splitting Attack – Proposal and Analysis of an Experimental Scheme,* Adv. Quantum Technol., vol. 7, no. 7, p. 2300437, 2024, doi: 10.1002/qute.202300437.

[7] Bencheikh, K., Levenson, J. A., Grangier, P. & Lopez, O. *Quantum nondemolition demonstration via repeated backaction evading measurements.* Phys. Rev. Lett.

[8] Bruckmeier, R., Hansen, H. & Schiller, S. *Repeated quantum nondemolition measurements of continuous optical waves.*Phys. Rev. Lett.79,1463 - 1466 (1997).

[9] P. Grangier, J. A. Levenson, and J.-P. Poizat, *Quantum non-demolition measurements in optics,* Nature, vol. 396, no. 6710, pp. 537–542, Dec. 1998

[10] Brune, M.et al. *Quantum Rabi oscillation: a direct test of field quantization in a cavity.* Phys. Rev. Lett. 76,1800 - 1803 (1996).

[11] Turchette, Q. A., Hood, C. J., Lange, W., Mabuchi, H. & Kimble, H. J. *Measurement of conditional.*

[12] T. F. da Silva, G. C. do Amaral, D. Vitoreti, G. P. Temporão, and J. P. von der Weid, *Spectral characterization of weak coherent state sources based on two-photon interference,* Phys. Rev. A.

[13] Luz cuasiclásica: estados coherentes de Glauber de radiación. Fisicacuantica.es

[14] Fock state. Wikipedia. Recuperado de: `https://en.wikipedia.org/wiki/Fock_state`

[15] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, *Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulses implementations,* Phys. Rev. Lett.

[16] W.-Y. Hwang, *Quantum key distribution with high loss: Toward global secure communication,* Phys. Rev. Lett.

[17] C. Branciard, N. Gisin, B. Kraus, and V. Scarani, *Security of two quantum cryptography protocols using the same four qubit states,* Physical Review A.

[18] N. Jain, E. Anisimova, I. Khan, V. Makarov, C. Marquardt, and G. Leuchs, *Trojan-horse attacks threaten the security of practical quantum cryptography,* New J. Phys., vol. 16, no. 12, p. 123030, Dec. 2014. doi: 10.1088/1367-2630/16/12/123030.

[19] da Silva, T. F., Xavier, G. B., Temporão, G. P., & von der Weid, J. P. *Real-time monitoring of single-photon detectors against eavesdropping in quantum key distribution systems.*

[20] Qiskit. Wikipedia. Recuperado de: `https://es.wikipedia.org/wiki/Qiskit`