Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

OTTIMIZZAZIONE DEI PARAMETRI DI FUNZIONAMENTO PER DISPOSITIVI DI DISTRIBUZIONE DI CHIAVE QUANTISTICA

Relatore:
Prof. Giuseppe Vallone
Correlatore:
Dott. Giulio Foletto

Laureando:
Tommaso Milanello

Anno Accademico 2022/2023

28 Settembre 2023

Abstract

In this thesis we will present one of the first direct applications of the fundamental laws quantum physics: Quantum Key Distribution, (QKD). It is an innovative method for exchanging secure keys using the properties of photons and exploiting well-tested physical theories to ensure their security. There will be a general introduction on the subject, after which the practical work done by the candidate will be discussed. Using python scripts, the generation of secure keys has been optimized by modifying the parameters with which a source device and a receiving device work. The quantitative results are specific for the devices used in DEI laboratories, but from them general conclusions can be inferred and the scripts are adaptable to machines with any type of characteristics.

Sommario

In questa tesi si esporrà una delle prime applicazioni dirette delle leggi fondamentali della fisica quantistica: la Quantum Key Distribution, (QKD). È un metodo innovativo per lo scambio di chiavi sicure utilizzando le proprietà dei fotoni e sfruttando teorie fisiche ben collaudate per garantirne la sicurezza. Ci sarà un'introduzione generale sull'argomento, al termine della quale verrà discusso il lavoro pratico svolto dal candidato. Utilizzando degli script python, la generazione delle chiavi sicure è stata ottimizzata modificando i parametri con cui lavorano un dispositivo sorgente e un dispositivo ricevente. I risultati quantitativi sono specifici per i dispositivi utilizzati nei laboratori DEI, ma da essi si possono inferire conclusioni generali e gli script sono adattabili a macchine con qualsiasi tipo di caratteristiche.

Indice

Capitolo 1

Introduzione

Ci troviamo in un'epoca in cui la crittografia è necessaria e fondamentale per lo svolgersi quotidiano di molte attività. Basta pensare a come tutti i giorni inviamo centinaia di messaggi e ci aspettiamo che solo il corretto destinatario li riceva e possa leggerli. Sia nella vita di ognuno che nelle attività produttive, è necessario che le informazioni vengano scambiate solo tra le persone prestabilite e che non ci sia qualche malintenzionato in ascolto. Si possono distinguere due tipi di metodi di crittografia: a chiave simmetrica e a chiave asimmetrica, entrambi con lati positivi e negativi. La crittografia simmetrica è generalmente più veloce ma c'è bisogno che sia il mittente che il destinatario conoscano a priori la stessa chiave segreta e non è sempre facile riuscire a scambiarsela in maniera sicura. D'altro canto, la crittografia asimmetrica non richiede la conoscenza a priori di una chiave comune, ma necessita di più assunzioni per garantire la sicurezza dei dati. Il metodo più usato generalmente è quello asimmetrico in quanto più semplice da realizzare. Gli algoritmi asimmetrici si basano sul fatto che non abbiamo ancora raggiunto la potenza di calcolo necessaria a decrittare una chiave in un tempo ragionevole. L'inesorabile avanzamento della tecnologia potrebbe arrivare troppo repentinamente e sovrastare la sicurezza dataci da questi algoritmi. I computer quantistici sarebbero una svolta in questo senso, in quanto i tempi necessari per decifrare un messaggio crittografato con un metodo asimmetrico diminuirebbero drasticamente. Una delle soluzioni a questo problema potrebbe quindi essere riuscire a creare un canale sicuro in modo da scambiarsi una chiave sicura segreta tra grandi distanze e poter ricorrere a metodi di crittaggio simmetrico. Qui entra in gioco l'argomento principale di questa tesi: la Quantum Key Distribution (QKD). Si tratta ancora di una tecnologia in fase evolutiva, tuttavia diverse aziende hanno già messo dei dispositivi in commercio. La QKD è una tecnica

innovativa che permette lo scambio di una chiave in maniera sicura sfruttando i principi e teoremi della meccanica quantistica.
Verranno introdotti nel capitolo 2 delle nozioni per capire meglio i concetti fisici principali ed il funzionamento di questa tecnologia. Nel terzo capitolo si discuterà degli strumenti utilizzati per compiere le simulazioni in modo efficente e successivamente nel capitolo 4 si esporranno le simulazioni atte a migliorare le performance delle macchine utilizzate per la QKD e anche per aumentare la resistenza ad attacchi per provare ad intercettare la chiave segreta. Queste simulazioni sono fatte tramite il linguaggio di programmazione Python e si basano sul modello matematico per stimare le performance della QKD sviluppato da Giulio Foletto, che mi ha seguito ed aiutato pazientemente con questa tesi, e dal team di QuantumFuture guidato dai professori Giuseppe Vallone e Paolo Villoresi. Infine nel capitolo finale si discuterà dei risultati ottenuti da questa tesi.

Capitolo 2

Quantum Key Distribution

Per poter parlare di QKD bisogna prima introdurre delle nozioni fondamentali, prima tra tutte il qubit. Nell'informazione classica la più piccola unità con cui si misura l'informazione si chiama bit, esso può assumere solo i valori 0 o 1. Il suo corrispettivo nell'informazione quantistica è il qubit (quantum bit), si tratta di uno stato quantistico bidimensionale. Si può affermare che prima di essere misurato, il valore di un qubit è una sovrapposizione di 0 e 1. Utilizzando la notazione di Dirac, un qubit si definisce:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

Dove α e β sono due numeri complessi. Questo stato è un vettore in uno spazio complesso bidimensionale: lo spazio di Hilbert H. I vettori che formano la base ortonormale di questo spazio sono $|0\rangle$ e $|1\rangle$. Misurando uno stato generico $|\psi\rangle$ si avrà probabilità $|\alpha|^2$ di ottenere il valore 0 e probabilità $|\beta|^2$ di ottenere il valore 1, quindi per la correttezza della probabilità è importante che:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

normalizzando quindi α e β e facendo sì che lo stato del qubit sia un vettore unitario. Si può quindi riscrivere come:

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle) \tag{2.3}$$

In cui γ, θ e ϕ sono angoli reali. Ignorando ora il termine di fase $e^{i\gamma}$ dato che non

comporta alcuna informazione aggiuntiva sullo stato si ottiene:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.4}$$

Dato che lo stato è descritto da soli due angoli reali θ, ϕ, lo si può associare a un punto sulla superficie di una sfera unitaria, chiamata in questo contesto sfera di Bloch (Figura 2.1).

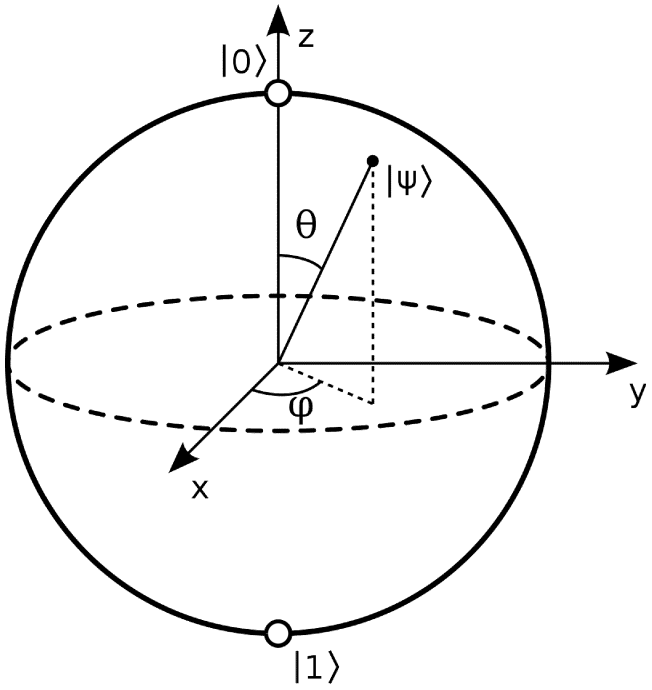


Figura 2.1: Sfera di Bloch (figura tratta da [1])

Nella QKD un qubit è spesso codificato nella polarizzazione di un fotone. Si usano i fotoni perché la luce ci permette di trasmettere stati quantistici attraverso lunghe distanze con errori ragionevoli. Sia un collegamento in fibra ottica che lo spazio libero sono ottimi metodi di connessione. [2]

2.1 Protocollo BB84

In uno scambio di chiave segreta tramite la QKD ci sarà un mittente, che chiameremo Alice, un destinatario, che chiameremo Bob, ed una terza persona malintenzionata che cerca di scoprire quale sia la chiave segreta, che chiameremo Eve. Consideriamo che Alice sia in possesso di una sorgente di singoli fotoni.

Lo scambio tra Alice e Bob deve avvenire seguendo un protocollo in modo tale che entrambi seguano gli stessi procedimenti standardizzati. Il protocollo che si andrà a descrivere è il primo protocollo per la QKD, ideato da C. Bennett e G. Brassard nel 1984, chiamato in loro onore BB84.

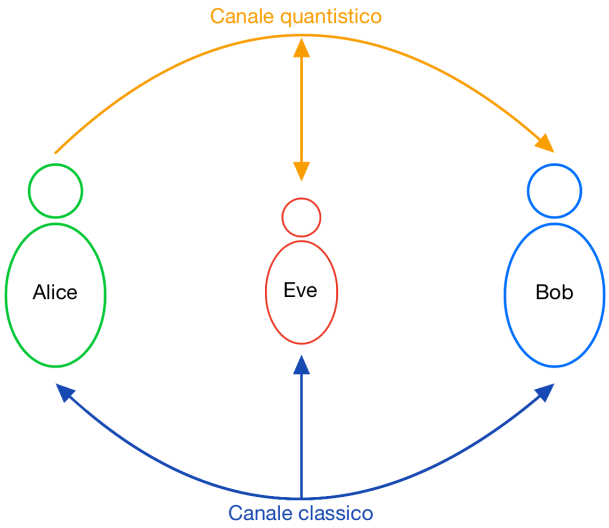


Figura 2.2: Schema dei partecipanti ad una QKD

Come si può vedere in Figura 2.2 Alice e Bob sono collegati tramite un canale quantistico, in cui Eve può intervenire ricevendo informazioni ed introducendone a sua volta, e da un canale classico autenticato, che può essere ad esempio il web, da cui Eve può solo ricevere. Il protocollo funziona nel modo seguente:

1. Alice prepara un fotone decidendone la sua polarizzazione. Può decidere di usare la base orizzontale/verticale (base Z) o la base diagonale/antidiagonale (base X) in maniera casuale. Utilizza poi gli stati all'interno della singola base per codificare i bit della chiave che vuole condividere con Bob, così da avere una codifica del tipo:

   - $|H\rangle$, codifica per 0_Z
   - $|V\rangle$, codifica per 1_Z
   - $|D\rangle$, codifica per 0_X
   - $|A\rangle$, codifica per 1_X

   Dopodichè invia il fotone e ripete il procedimento per N volte. Bob riceve i fotoni e li misura scegliendo casualmente una delle due basi. In un sistema

ideale senza perdite o intromissioni da parte di Eve, se Alice invia un fotone preparato nella base Z con valore $|H\rangle$ e Bob lo misura in base Z troverà il valore $|H\rangle$ il 100% delle volte, mentre se Bob dovesse misurare lo stesso fotone nella base X troverà il valore $|D\rangle$ il 50% delle volte e il valore $|A\rangle$ il restante 50%. Finito questo procedimento sia Alice che Bob avranno N coppie di bit e basi con cui hanno preparato e misurato ognuno di essi. Si assume che la trasmissione sia priva di perdite per semplificare questa discussione. A questo punto Alice e Bob hanno due raw key (chiavi grezze) che per il momento possono anche non essere identiche.

2. Ora avviene la fase di classical post-processing. Si inizia con il sifting: Alice e Bob comunicano attraverso il canale classico la base scelta per preparare e misurare ogni fotone. Scartano le misurazioni fatte con basi diverse in quanto avranno un risultato completamente casuale.

3. Successivamente, Alice e Bob, eseguono il parameter estimation (PE). Rivelano nel canale classico una parte della chiave ottenuta così da individuare eventuali errori. Se non ne riscontrano allora Eve non ha alcuna informazione sulla chiave. Altrimenti gli errori potrebbero essere causa dell'intervento di Eve nel canale quantistico e quindi sono considerati come informazione che Eve ha sulla chiave. La parte rivelata viene marcata come insicura così da essere eliminata successivamente.

4. Alice e Bob proseguono con le procedure di error correction (EC): correggono gli errori e alla fine di questo processo condividono la stessa identica raw key.

5. Infine, avviene la privacy amplification (PA) in cui la raw key viene ridotta, ad esempio con algoritmi di hashing, così da eliminare completamente qualsiasi informazione che possa essere trapelata ad Eve, sia in fase di comunicazione quantistica, che in fase di classical post-processing

Si noti che in fase di parameter estimation Alice e Bob potrebbero accorgersi che Eve ha ottenuto troppe informazioni riguardanti la chiave, non rendendola più sicura e segreta. In questi casi la soluzione è quella di interrompere il processo e ricominciare da capo. Non ci sono problemi di sicurezza nel farlo in quanto non si utilizzeranno mai queste informazioni trapelate a Eve per la futura chiave. [2] [3] [4]

Questo è solo uno di vari protocolli utilizzabili per la QKD, è quello su cui sono stati svolte le simulazioni di cui si parlerà in seguito.

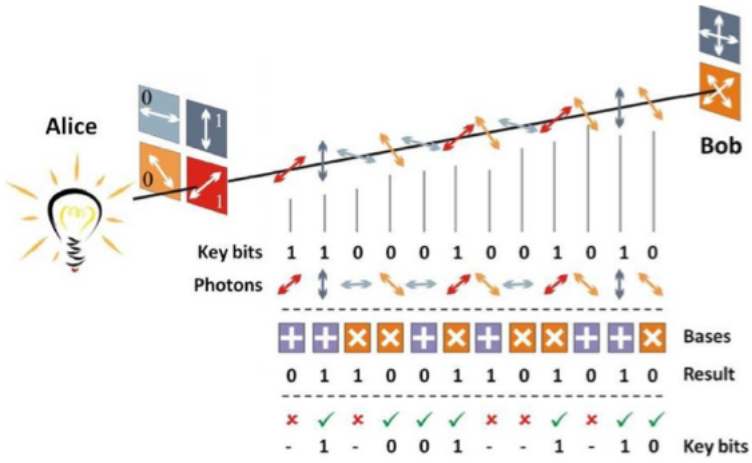


Figura 2.3: Schematizzazione del protocollo BB84(figura tratta da [5])

2.2 Sicurezza e attacchi

Eve ha molti modi diversi di compiere un attacco sulla QKD, come primo si cita il denial of service (DOS), in cui semplicemente interrompe la connessione nel canale quantistico tra Alice e Bob e bisognerà ripartire da capo con la QKD. Come per qualsiasi protocollo di comunicazione classico, la QKD non è di per sé equipaggiata per fronteggiare questo attacco, che va mitigato esternamente. Tuttavia va notato che il DOS non mina la sicurezza della chiave, che semplicemente non viene creata.

Fra gli attacchi che Eve può usare per carpire la chiave, quello più semplice, dato che Eve ha completo accesso al canale quantistico, è il "intercept and resend", "intercetta e rispedisci". In cui Eve intercetta i fotoni inviati da Alice, li misura e li rispedisce a Bob. La sicurezza relativa a questo attacco ci è data dal No-Cloning Theorem (Teorema del Non-Clonazione). Questo teorema dice che non è possibile creare una copia esatta di uno stato quantistico sconosciuto. In pratica se Eve misura il fotone intercettato nella base corretta riuscirà a conoscerlo e a rimandarlo a Bob, ma se sbaglia base invierà a Bob un fotone nella base diversa da quella scelta da Alice. Quindi invierà un errore nel 25% dei casi. Eve avrà più informazione sulla chiave ripetto a Bob e grazie al protocollo BB84 Alice e Bob riusciranno ad accorgersene in fase di classical postprocessing e a scartare la chiave.

Un altro attacco è il photon-number splitting (PNS). Il protocollo BB84 originalmente prevede che vengano inviati singoli fotoni da Alice a Bob, però questo è molto difficile da implementare e spesso per la QKD vengono usati laser attenuati che inviano impulsi a singoli fotoni con una probabilità fissabile ma non pari a 1. Con il PNS Eve può intercettare proprio gli impulsi che contengono più fotoni, tenere per sè alcuni fotoni, misurarli e mandare a Bob gli altri invariati, evitando che egli possa accorgersi dell'attacco. Un modo per fronteggiare questo attacco è quello dei decoy states.
Per effettuare una trasmissione con i decoy states Alice sceglie casualmente l'intensità di ogni impulso tra alcuni livelli prefissati. Maggiore è l'intensità e più fotoni verranno inviati con ogni impulso. Durante il classical post-processing Bob segnala ad Alice quali impulsi sono stati ricevuti. Incrociando i suoi dati con quelli di Bob, Alice conosce per ogni livello di intensità la probabilità di ricezione. Da queste probabilità Alice riesce a stimare quanti impulsi sono partiti con più di un fotone, e vanno quindi considerati insicuri. [3]

Capitolo 3

Strumenti per le simulazioni

In questo capitolo si esporranno gli strumenti utilizzati per le simulazioni. Per strumenti si intende tutto ciò che è stato utile ad arrivare ai risultati esposti nel Capitolo 4, sia software che hardware.

3.1 Modello di QKD e grapes

La struttura delle simulazioni si basa sulla libreria `grapes` [6] e su un modello delle performance della QKD costruito con essa. `Grapes` permette di descrivere una computazione tramite grafi diretti e aciclici. Ogni valore da calcolare è salvato in un nodo del grafo e ogni ramo rappresenta una funzione, la quale dato un input produce un output, senza alcuno stato globale che vada ad interferire con i calcoli. Utilizzare un grafo del genere ha come vantaggio che al momento della definizione della computazione non serve specificare quali variabili fungono da input e quali da output. Si scelgono a runtime i valori di input che descrivono l'ambiente da esaminare e l'obbiettivo a cui si vuole arrivare. Se l'obbiettivo è raggiungibile tramite gli input forniti, `grapes` seguirà il percorso tra i rami e i nodi fino a computare il valore desiderato.
Tramite questa libreria, il team di QuantumFuture del DEI ha sviluppato un modello che permette di calcolare le performance di un sistema di QKD partendo da una sua descrizione fisica. Grazie alla struttura di grafo, il modello permette di analizzare il ruolo individuale di molte variabili che influenzano la QKD.

3.2 Dual Annealing

Il cuore delle simulazioni fatte è l'algoritmo di ottimizzazione sfruttato per trovare i massimi valori di SKR per ogni valore dei parametri controllato. L'algoritmo in questione è il dual annealing. Si tratta di un algoritmo stocastico di ottimizzazione globale, quindi che sfrutta la casualità per trovare il minimo globale per una funzione anche non lineare. Combina un'ottimizzazione locale di tipo hill climbing, ossia che scala le "montagne" delle funzioni per trovarne gli estremi, con un simulated annealing, che si ispira alla tecnica usata nella metallurgia in cui un materiale viene scaldato e raffreddato lentamente per raggiungere una struttura cristallina ottimale. Compiendo dei salti casuali all'interno della funzione, la cui intensità andrà via via a diminuire in maniera dettata da un parametro assimilabile a una temperatura, l'algoritmo andrà a trovare un minimo che sarà molto probabilmente quello globale. Per capire come funziona si può pensare ad uno scatolone chiuso il cui fondo è fatto come una catena montuosa, con le varie valli e i vari picchi. Al suo interno inseriamo una pallina ed iniziamo a scuotere lo scatolone, registrando le "valli" in cui finisce la pallina (i minimi locali). Più volte scuotiamo lo scatolone e più diminuiamo l'intensità delle scosse fino a decretare un ultimo punto di arrivo, quello sarà il minimo ritornato dall'algoritmo di ricerca. Nelle simulazioni fatte servirà sempre il massimo, ma il dual annealing è progettato per ritornare un minimo; il problema si risolve effettuando la ricerca sulla funzione cambiata di segno, così da trovare i massimi. [7]

3.3 Singularity: Container e Job Scheduler

Singularity è un software che permette di costruire ed utilizzare dei containers. Essendo le simulazioni progettate per cercare dei massimi globali c'è bisogno di una discreta potenza di calcolo e quindi di discreto tempo affinché i risultati siano affidabili. Grazie alla costruzione di un apposito container è stato possibile far girare i programmi sul cluster "Blade" del Dipartimento di Ingegneria dell'Informazione dell'Università di Padova.
Un cluster High Performance Computing (HPC) è una combinazione di hardware specializzato, che comprende numerosi computer ad alta capacità computazionale e un framework software di elaborazione distribuita configurato per gestire enormi quantità di dati ad alta velocità con prestazioni parallele ed elevata disponibilità. Grazie ad un container è possibile isolare diverse istanze, tutte basate sullo stesso

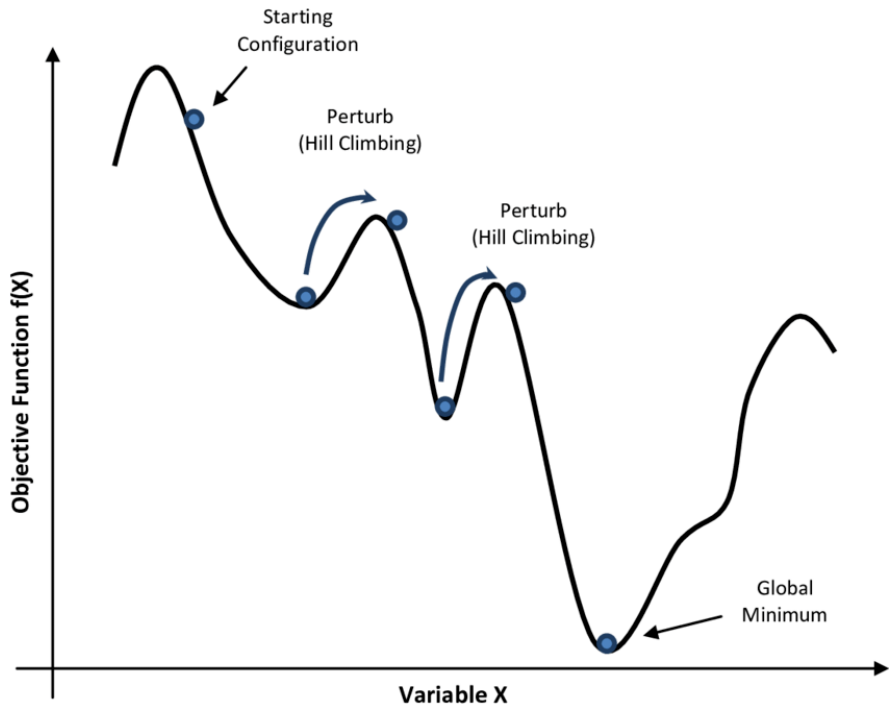


Figura 3.1: Semplice esempio di funzionamento del simulated annealing (figura presa da [8]).

sistema operativo del computer su cui è installato il container, ma tutte indipendenti tra loro. Ogni istanza avrà i suoi pacchetti, librerie, programmi e dati personalizzati dal creatore del container, atti a far funzionare l'applicazione per cui quel container è stato creato. Quindi grazie ai container è possibile sfruttare un cluster HPC per l'esecuzione di programmi pesanti e l'acquisizione di dati senza la paura che programmi esterni di altri utenti influiscano sull'esecuzione della propria applicazione. Grazie a questa tecnologia sono state svolte le simulazioni sul cluster HPC Blade.

Un ultimo elemento importante da accennare è il Job Scheduler, ossia ciò che permette l'utilizzo in contemporanea da parte di più utenti del cluster. Per poter utilizzare il cluster tramite i container è necessario mandare una richiesta caricando un file in cui vengono indicate le risorse di cui avrà bisogno la computazione e le istruzioni da linea di comando che andranno ad essere eseguite. Alcune delle risorse da dichiarare sono il tempo di utilizzo necessario, la memoria necessaria e il numero di core o cpu necessari all'esecuzione dell'applicazione. Il Job Scheduler, chiamato Slurm, alloca le risorse richieste e schedula il periodo di esecuzione in modo tale che non ci siano interruzioni e tutte le risorse rimangano disponibili per il tempo indicato. Così facendo non avverranno problemi in caso di numerose richieste simultanee. [9] [10] [11] [12] [13]

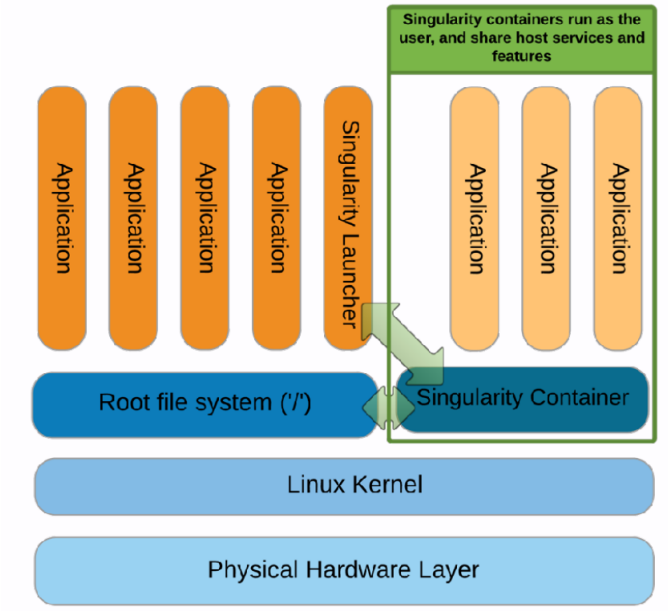


Figura 3.2: Architettura di un container (immagine presa da [14]).

Capitolo 4

Simulazioni

Le simulazioni fatte in questa tesi sono servite a capire come vari parametri e diversi dispositivi per la QKD possono influenzare l'efficienza della generazione di una chiave sicura. Si è partito da esperimenti reali in corso all'Università di Padova e al variare di parametri su cui si ha poco controllo si sono ottimizzati parametri regolabili al fine di rendere più efficente la QKD negli specifici scenari. In tutte le simulazioni fatte, l'obbiettivo, ossia il valore che ci interessa andare ad esaminare, è il Secret Key Rate (SKR):

$$\mathrm{SKR} = \frac{\text{bit sicuri}}{\text{secondi}} \tag{4.1}$$

che consiste nel capire quanti bit sicuri vengono generati al secondo ed è utile per confrontare le varie impostazioni possibili di un dispositivo per la QKD.

4.1 Descrizione degli scenari

I risultati che verranno esposti riguarderanno ottimizzazioni fatte su due scenari relativi a due esperimenti attualmente in corso all'interno dell'università di Padova.

- Legnaro: chiamato così perché si trova ai laboratori nazionali di Legnaro (in provincia di Padova), una delle sedi dell'Istituto Nazionale di Fisica Nucleare. Alice è situata lì ed è collegata con un canale in fibra fino ad un ricevitore Bob, situato nei laboratori DEI di via Trasea. Le perdite di canale sono 14 dB, contando anche le perdite al ricevitore si arriva circa a 16 dB. Questo esperimento è fatto per testare un link di lunghezza extraurbana

tra due istituzioni di ricerca, molto importante per valutare l'efficacia a distanze importanti della QKD.

- Garr: GARR è l'istituzione che si occupa della connettività di rete di tutte le istituzioni di ricerca e educazione italiane, e l'università di Padova sta collaborando con GARR per esperimenti di QKD. L'esperimento è stato fatto collegando una facility chiamata VSIX, che è un internet exchange, ossia un'istituzione universitaria per gestire i servizi di connettività e di interfaccia del territorio dell'ateneo di Padova, al centro di calcolo dell'ateneo. È quindi un collegamento urbano tra galleria Spagna e via San Francesco, in centro a Padova.

La differenza importante tra i due è che l'esperimento Garr utilizza una lunghezza d'onda di 1310 nanometri, mentre l'esperimento Legnaro di 1550 nanometri. Sono due lunghezze standard usate molto spesso nelle comunicazioni. Quella a 1550 nanometri è più ottimizzata per canali a lunga distanza, offre la migliore trasmissività in fibra, quindi meno perdite, inoltre utilizza una connessione esclusiva per l'esperimento. Invece, quella a 1310 nanometri, è in sovrapposizione sulla stessa fibra con un collegamento classico che ha una diversa frequenza d'onda, questo quindi prova che è possibile sfruttare collegamenti già esistenti per attuare una QKD.
Nella presentazione dei risultati si riferirà agli esperimenti chiamandoli solo Garr e Legnaro.

4.2 Esercizio esempio

Per iniziare è stato svolto un esercizio su cui si basano le simulazioni vere e proprie.
L'esercizio inizia caricando le informazioni dei contesti degli esperimenti in un grafo. Bisogna indicare quale parametro sia l'obbiettivo, ossia il parametro che effettivamente andrà ottimizzato, i parametri rispetto ai quali si ottimizza, quelli che si fanno variare per studiare un set ampio di scenari (variabili di scenario) e quelli che rimangono fissati (costanti di scenario). Dopodichè per poter usare il grafo con un'ottimizzazione dual annealing lo si trasforma in una funzione. Si definiscono i limiti che influiranno sia sulla precisione dell'ottimizzazione, che sui valori massimo e minimo dello scenario. Infine, si ottimizza il valore del parametro obbiettivo per ogni valore delle variabili di scenario.

I risultati di questo esempio vengono esposti in Figura 4.2 e in **??**. È stato utilizzato un set arbitrario di costanti di scenario come partenza, il secret key rate come obiettivo, e la lunghezza di blocco come variabile di scenario. Il significato di tutti questi parametri verrà meglio spiegato nelle sezioni successive.
Tutte le simulazioni sono fatte sulla falsa riga di questo esercizio, con miglioramenti nella raccolta dei dati.
Il risultato finale di tutte le ottimizzazioni, oltre ai dati raccolti, è stata la creazione di un programma in python completamente adattabile a qualsiasi contesto di QKD si voglia e che esegue simulazioni su qualsiasi parametro sia di interesse.

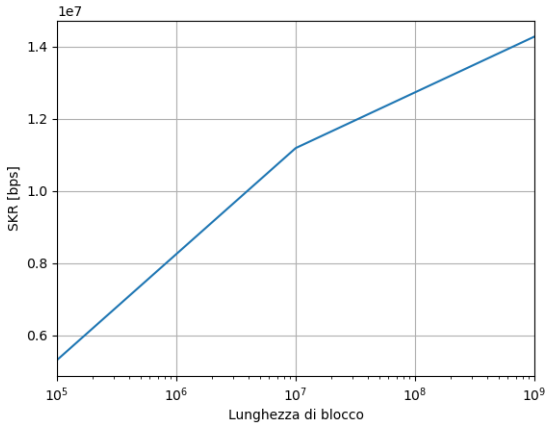


Figura 4.1: Risultati dell'esercizio esempio

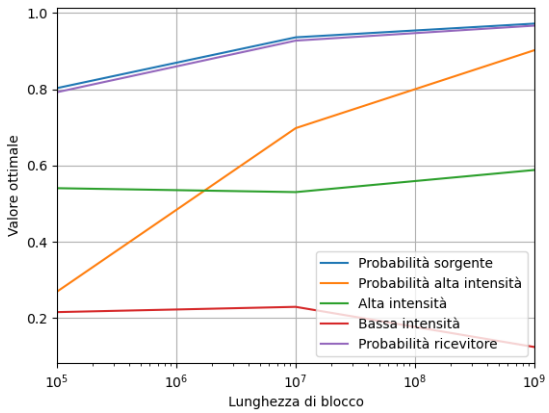


Figura 4.2: Valori ottimizzati dei parametri dell'esercizio esempio

4.3 Prima simulazione: efficienza di canale

Per la prima simulazione si fa variare come parametro di scenario l'efficienza di canale. Su questo parametro si ha poco controllo ed è quindi utile capire come può influenzare la QKD.
Il significato dei parametri ottimizzati è il seguente:

- Probabilità della sorgente: è la probabilità che il dispositivo sorgente utilizzi come base per la preparazione dell'impulso la base di chiave.
- Probabilità del ricevitore: è la probabilità che il dispositivo ricevitore utilizzi come base per la misurazione dell'impulso la base di chiave.
- Probabilità di alta intensità: è la probabilità che il dispositivo sorgente scelga di mandare impulsi con il livello di intensità alto.
- Intensità alta: è il valore di intensità degli impulsi con livello alto.
- Intensità bassa: è il valore di intensità degli impulsi con livello basso.

In tutte le simulazioni sono state usati cinque diversi metodi di ottimizzazione:

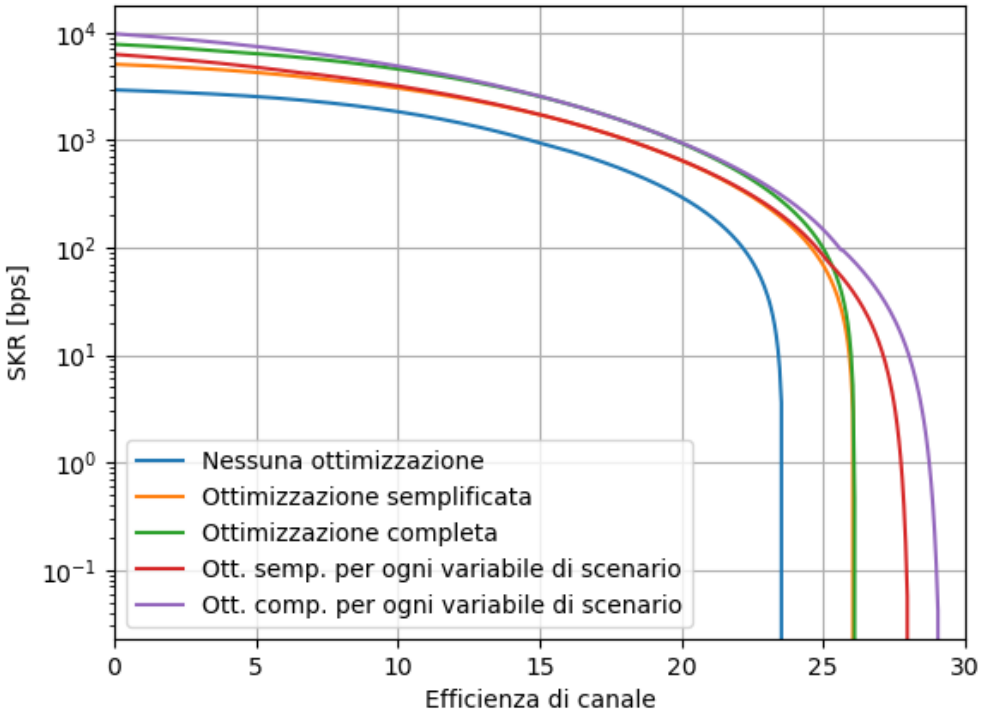
- Nessuna ottimizzazione: l'SKR è stato calcolato per ogni scenario tenendo i parametri utilizzati nel rispettivo esperimento, quindi non è stato ottimizzato nulla e si usa questa curva come punto di partenza per il confronto con le altre ottimizzazioni.
- Ottimizzazione semplificata: si sono ottimizzati i parametri facilmente regolabili nei dispositivi per la QKD. Si è ottimizzato solo per il valore di partenza del contesto e dopodichè si è calcolato l'SKR per ogni valore della variabile di scenario senza ottimizzare nuovamente ogni volta. Nel dettaglio si è ottimizzato rispetto ai parametri:
  - Probabilità della sorgente
  - Probabilità di alta intensità
  - Intensità alta

  La probabilità del ricevitore viene tenuta fissata e uguale a quella della sorgente, mentre l'intesità bassa viene fissata a seguito dell'ottimizzazione di quella alta tenendo fissato il rapporto fra le due al valore assunto nell'esperimento reale.

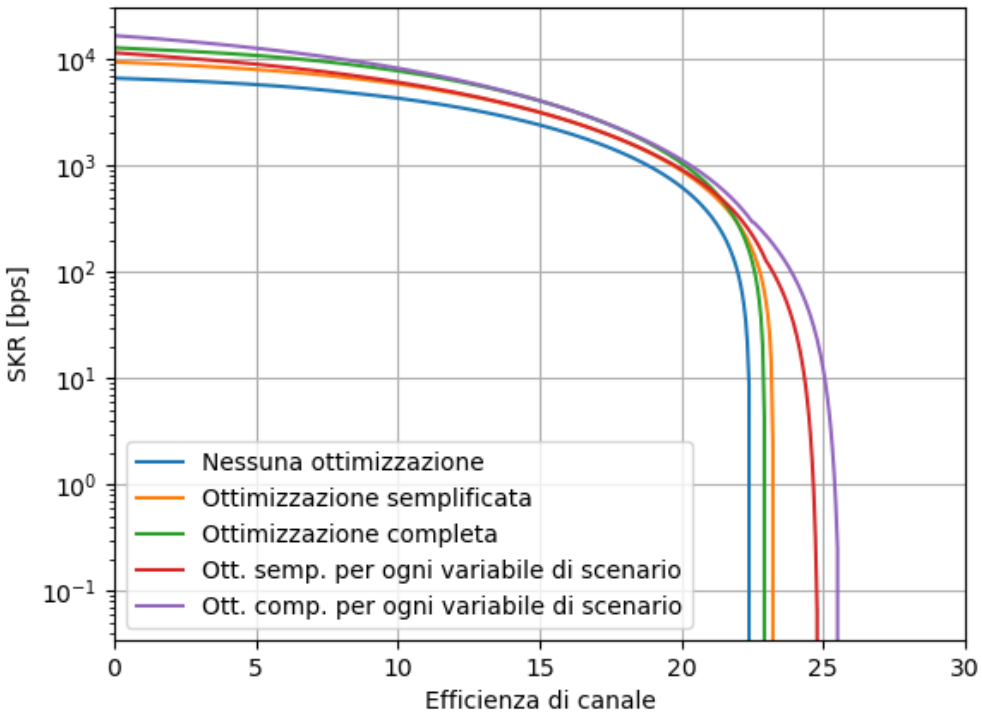
- Ottimizzazione completa: si sono ottimizzati sia parametri facilmente regolabili via software, che parametri per cui bisognerebbe modificare l'hardware del dispositivo, è per l'appunto un'ottimizzazione più completa rispetto alla precedente, ma sicuramente più difficile. Nel dettaglio si è ottimizzato rispetto ai parametri:
  - Probabilità della sorgente
  - Probabilità del ricevitore
  - Probabilità di alta intensità
  - Intensità alta
  - Intensità bassa
- Ottimizzazione semplificata per ogni scenario: si sono ottimizzati gli stessi parametri dell'ottimizzazione semplificata, ma per ogni valore della variabile di scenario.
- Ottimizzazione completa per ogni scenario: si sono ottimizzati gli stessi parametri dell'ottimizzazione completa, ma per ogni valore della variabile di scenario.

Si possono valutare le prestazioni di ogni ottimizzazione in Figura 4.3a e in Figura 4.3b. Si nota che le ottimizzazioni garantiscono un miglioramento di prestazioni di circa un fattore 3 in caso di canali molto efficienti e soprattutto permettono l'utilizzo di canali più disturbati di quasi 5 dB per entrambi gli esperimenti.
Nella Figura 4.4 viene graficata la variazione dei valori dei parametri nell'ottimizzazione completa per ogni scenario. Si mostrano i risultati dei parametri solo di questa ottimizzazione in quanto più interessanti di quella semplificata.
Si nota che per far fronte ad attenuazioni maggiori, il sistema tende ad utilizzare un livello di intensità alto sempre maggiore, lo stesso per la probabilità di intensità alta, aumenta la probabilità del livello di intensità maggiore in quanto con molte perdite un segnale debole tende a non arrivare a destinazione.
Guardando la linea di probabilità della sorgente, si può vedere che con poca attenuazione, il sistema si focalizza maggiormente sulla base di chiave, mentre all'aumentare delle perdite tende al valore di 0.5, quindi a bilanciare la scelta delle basi. Questa scelta è fatta per la necessità di avere abbastanza dati per la statistica che considera quanti fotoni con base di controllo sono arrivati al destinatario, in modo da poter eseguire correttamente il protocollo BB84 con decoy

states. La probabilità del ricevitore invece tende a rimanere stabile sopra lo 0.5 per cercare di misurare correttamente più fotoni possibili preparati in base di chiave. Questo andamento sarà standard per tutte le simulazioni successive.
I salti improvvisi nelle linee del grafo a circa 26 decibel avvengono perché, come approfondito nella sezione 3.2, l'algoritmo dual annealing trova un massimo all'interno di una funzione. La forma di questa funzione si modifica ad ogni valore dell'efficienza di canale e ad un certo punto potrebbe succedere che un "picco" della funzione diventi più alto di quello che prima era il massimo. Quando questo accade, la posizione del massimo globale cambia in maniera discontinua.

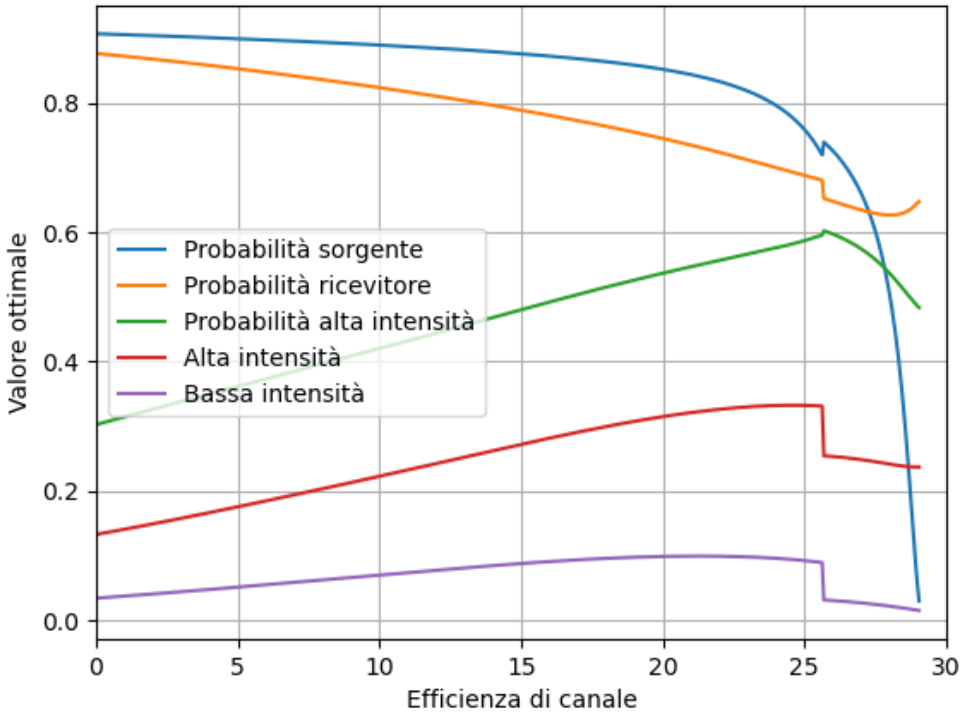


(a) Confronto tra i divesri tipi di ottimizzazioni per Garr

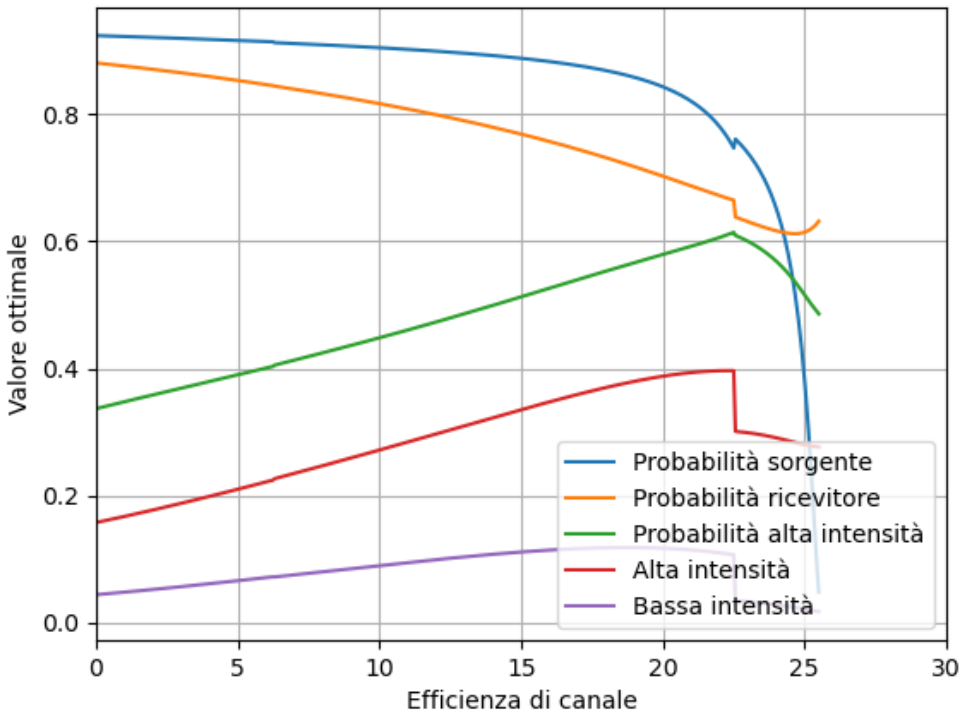


(b) Confronto tra i divesri tipi di ottimizzazioni per Legnaro

Figura 4.3: Confronti delle ottimizzazioni rispetto all'efficenza di canale nei due esperimenti



(a) Valori dei parametri ottimizzati per Garr

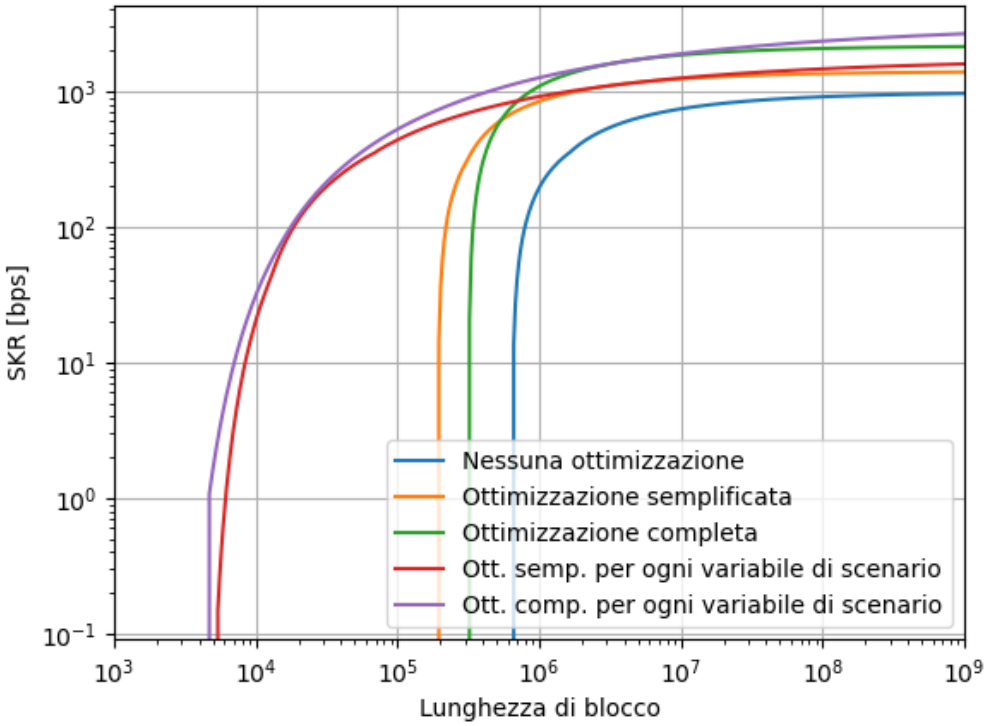


(b) Valori dei parametri ottimizzati per Legnaro

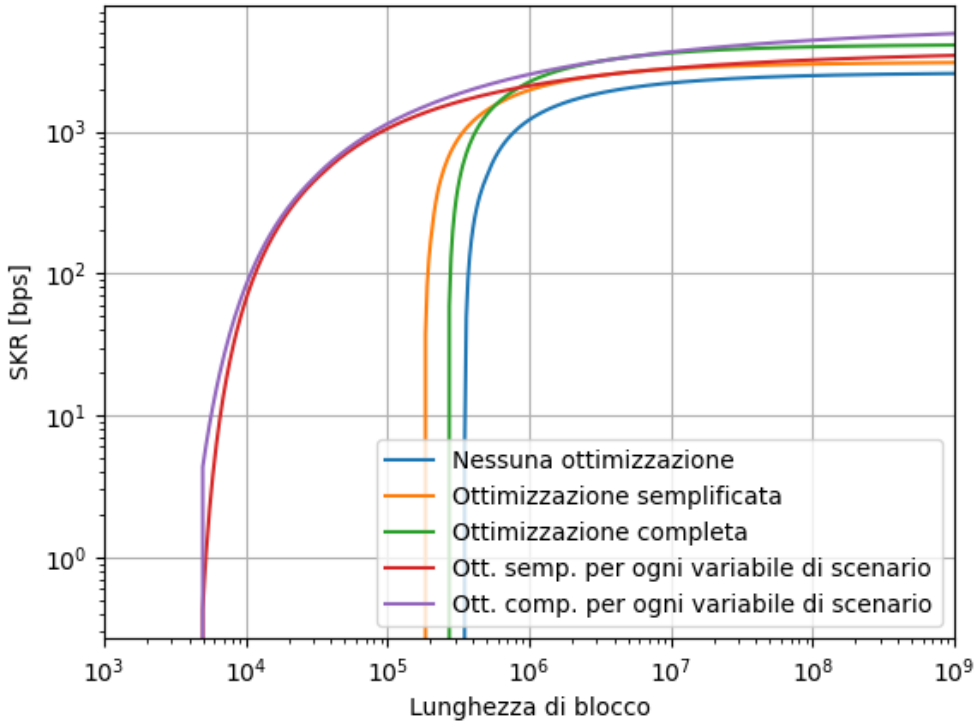
Figura 4.4: Valori dei parametri ottimizzati rispetto all'efficenza di canale nei due esperimenti

4.4 Seconda simulazione: lunghezza di blocco

La lunghezza di blocco è la lunghezza di chiave grezza che viene imposta al sistema come soglia per far partire il post-processing. È un parametro su cui si ha pieno controllo, però bisogna considerare diversi aspetti.
Se si imposta una lunghezza di blocco grande ci vorrà molto più tempo per arrivare ad una trasmissione di bit sicuri. Oltre al tempo necessario, servirà anche più fatica computazionale e si avrà bisogno di molto spazio per tenere memorizzata la chiave intera. Un altro problema potrebbe essere che il canale per la QKD non sia disponibile per abbastanza tempo da permettere di raggiungere il valore desiderato. Questo avviene ad esempio nella QKD satellitare, in cui il satellite è in vista solo per un periodo di tempo limitato per ogni orbita intorno alla Terra.
Riferendosi alla Figura 4.5 si osserva che le ottimizzazioni permettono di sostenere lunghezze di blocco molto più piccole. Si nota inoltre che non c'è molta differenza tra un'ottimizzazione semplificata ed una completa. Per valori bassi addirittura si sovrappongono.
Un altro elemento interessante è che effettuando un'ottimizzazione completa per un singolo valore di lunghezza di blocco si va a peggiorare maggiormente l'efficienza con valori bassi di scenario. Questo avviene perché ottimizzare molti parametri per un solo valore di lunghezza di blocco rende il sistema troppo specializzato per quel valore e lo rende meno adattabile ad altri valori. Nel grafico lo si vede notando che la linea verde crolla a zero per valori più alti della lunghezza di blocco rispetto alla linea arancione.
Nella Figura 4.6 si nota che all'aumentare della lunghezza di blocco aumenta la probabilità della sorgente di mandare fotoni preparati in base di chiave. Questo avviene perché avendo a disposizione una grande lunghezza di blocco, il sistema non ha bisogno di favorire la base di controllo per raccogliere un numero sufficiente di dati. Per lo stesso motivo si ha un aumento della probabilità di usare il livello di intensità alto. I valori dei livelli di intensità restano pressappoco invariati in quanto abbastanza ottimizzati per il corretto funzionamento.

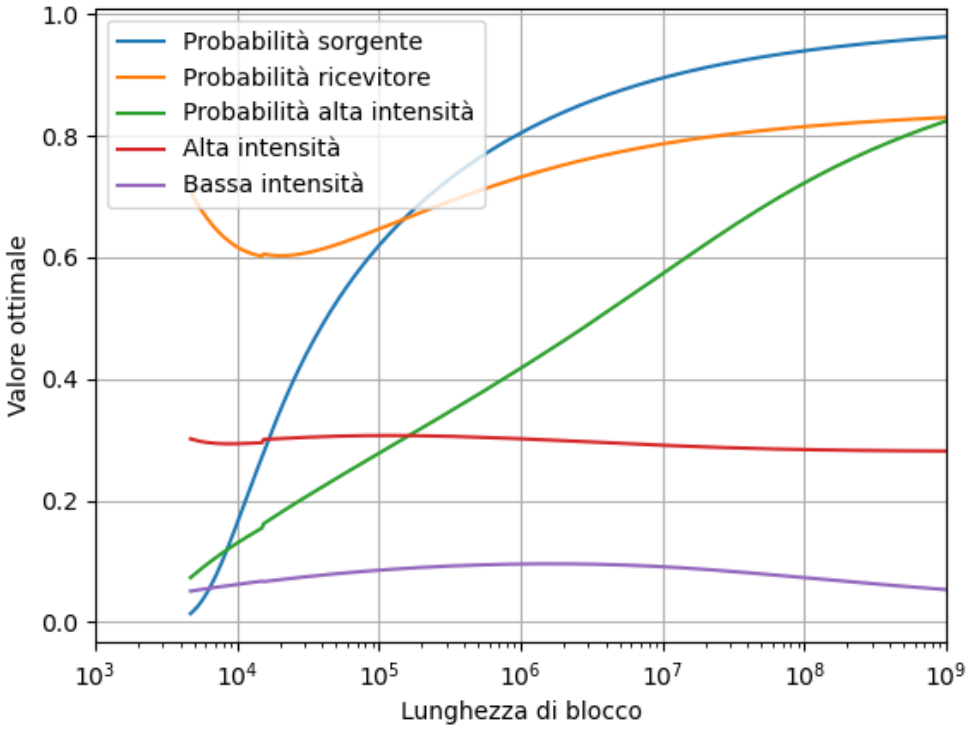


(a) Confronto tra i divesri tipi di ottimizzazioni per Garr

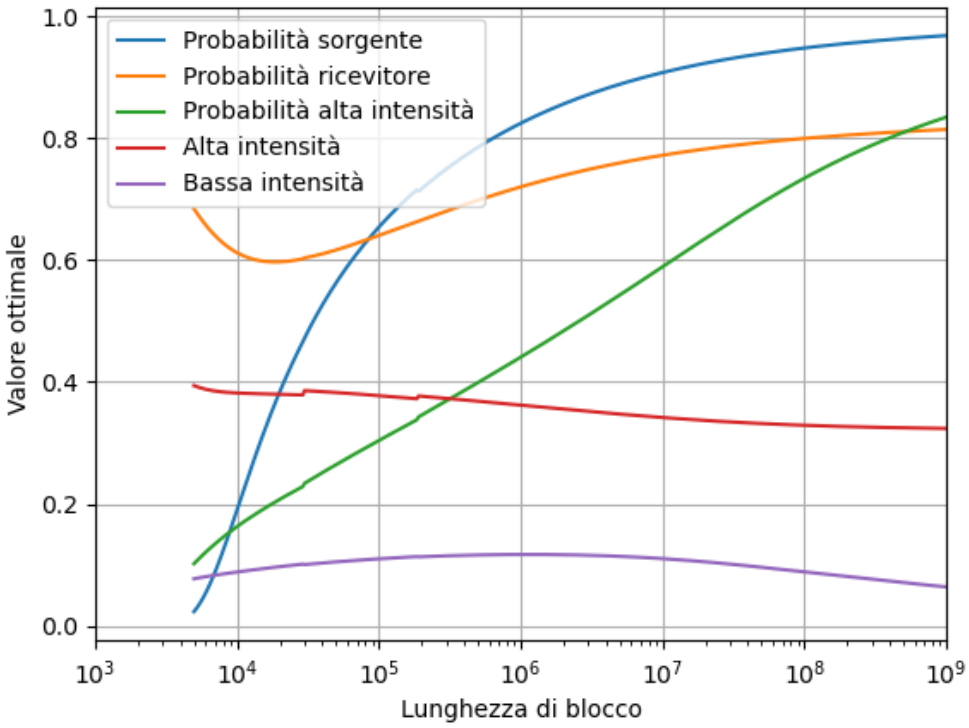


(b) Confronto tra i divesri tipi di ottimizzazioni per Legnaro

Figura 4.5: Confronti delle ottimizzazioni rispetto alla lunghezza di blocco nei due esperimenti



(a) Valori dei parametri ottimizzati per Garr

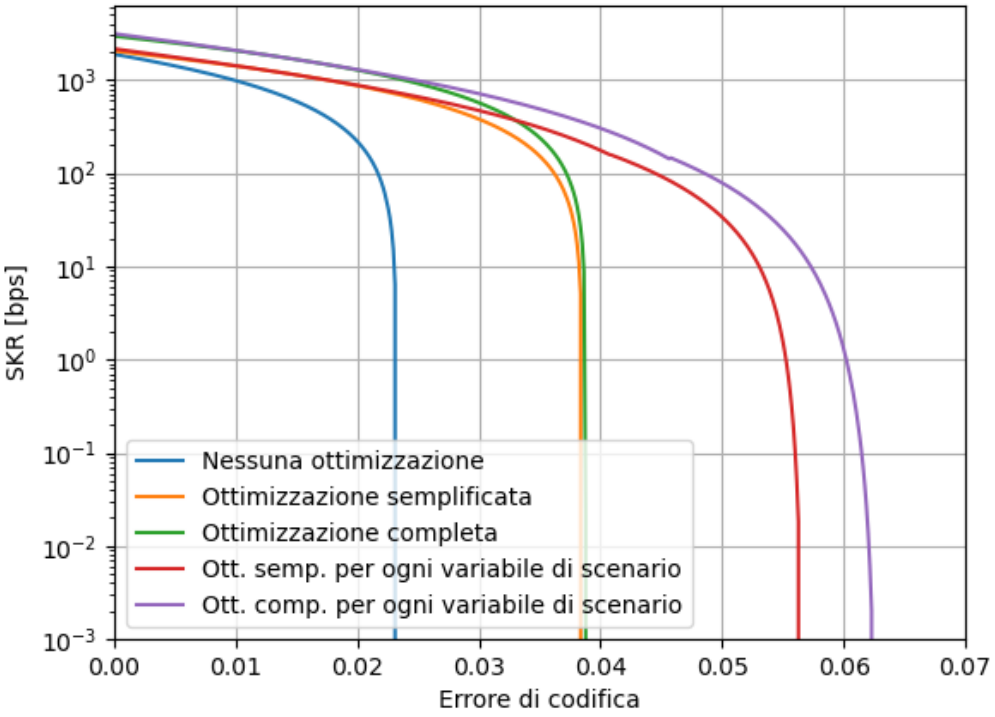


(b) Valori dei parametri ottimizzati per Legnaro

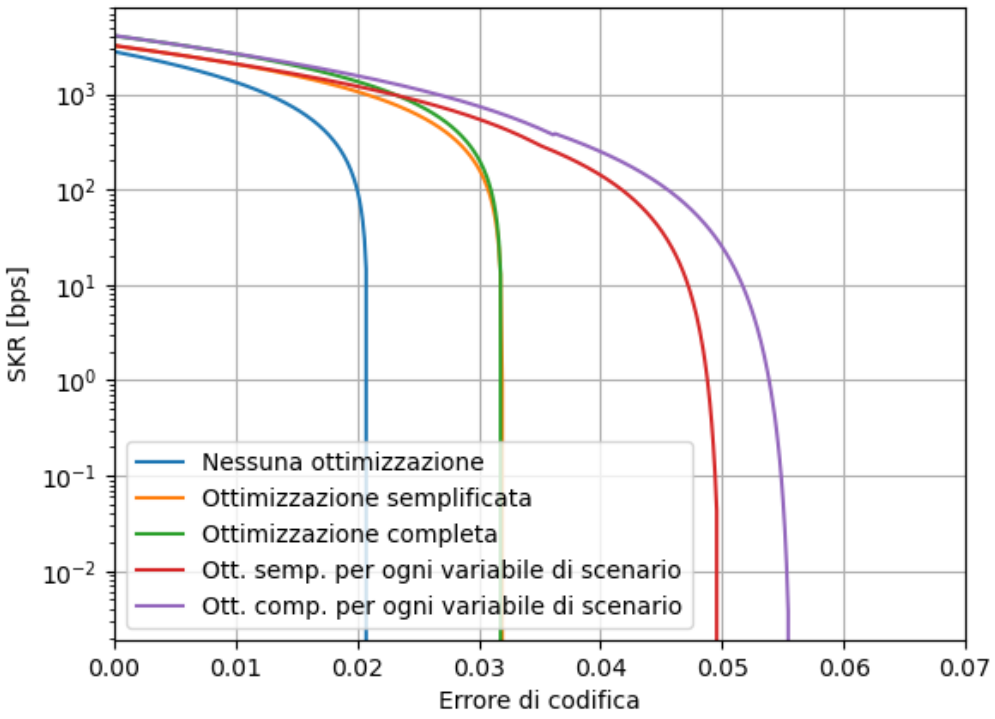
Figura 4.6: Valori dei parametri ottimizzati rispetto alla lunghezza di blocco nei due esperimenti

4.5 Terza simulazione: errore di codifica

L'errore di codifica è un parametro su cui si può agire poco, dipende dai componenti utilizzati per la costruzione dei dispositivi. Questi aspetti determinano quanto sono allineati gli stati prodotti dalla sorgente rispetto a quelli misurati al ricevitore.
Come si può vedere nella Figura 4.7, partendo dai parametri reali degli esperimenti, con un errore di codifica poco sopra al 2%, non si può più produrre una chiave sicura. Ottimizzando, non si ottiene un grande miglioramento a valori bassi, il vantaggio delle ottimizzazioni è riuscire ad arrivare a valori di SKR bassi ma comunque positivi anche con errori molto più alti. Posso quindi sostenere un errore di codifica fino al 5%.
Il parametro più interessante nella Figura 4.8 è la probabilità di sorgente che tende a seguire l'andamento dell'efficienza del sistema perché prova a mitigare i danni di un errore di codifica alto mandando più impulsi in base di controllo.

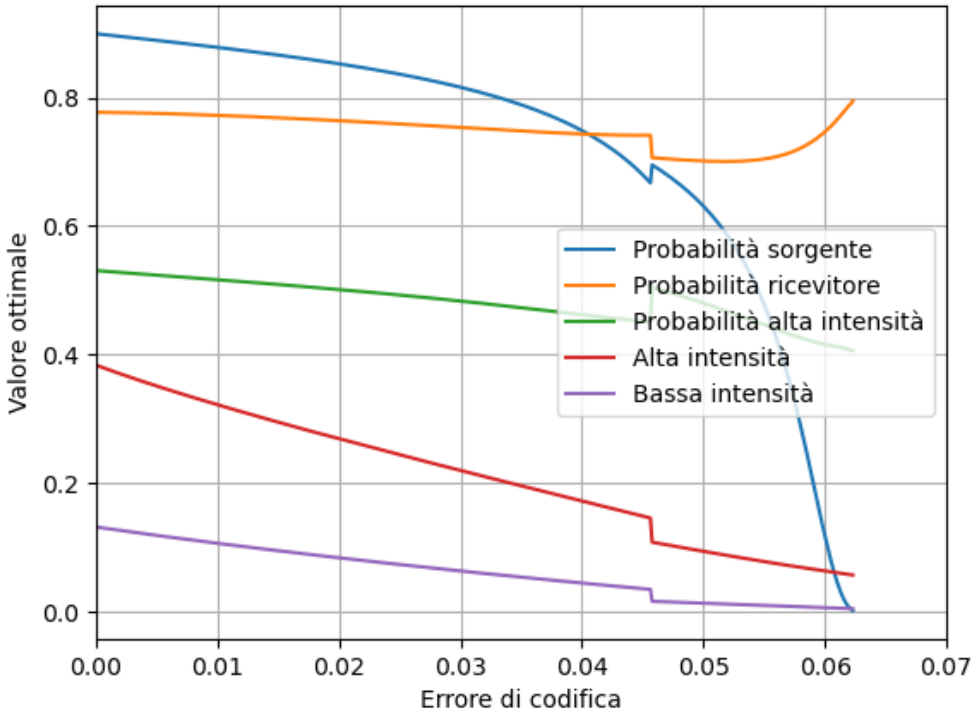


(a) Confronto tra i divesri tipi di ottimizzazioni per Garr

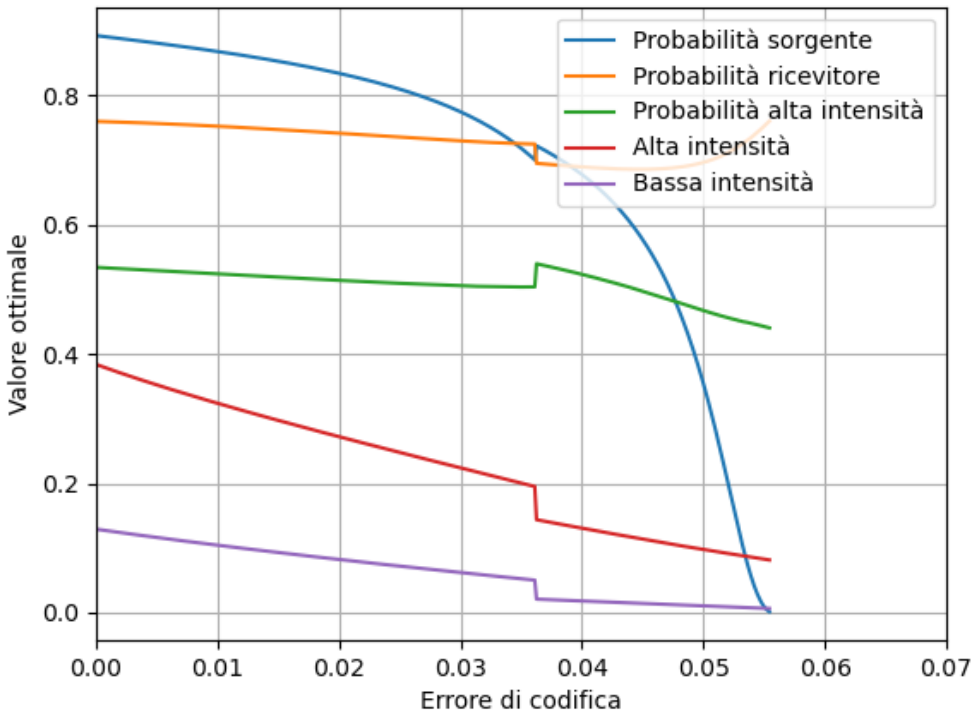


(b) Confronto tra i divesri tipi di ottimizzazioni per Legnaro

Figura 4.7: Confronti delle ottimizzazioni rispetto alla lunghezza di blocco nei due esperimenti



(a) Valori dei parametri ottimizzati per Garr



(b) Valori dei parametri ottimizzati per Legnaro

Figura 4.8: Valori dei parametri ottimizzati rispetto alla lunghezza di blocco nei due esperimenti

Capitolo 5

Conclusioni

Nel Capitolo 1 si è data un'introduzione sulla crittografia e su come sia da molti anni una necessità per la società odierna. Si è parlato di come l'avanzamento tecnologico, oltre che un beneficio, potrebbe diventare una minaccia e si è introdotto come l'entrata in gioco della fisica quantistica applicata come possibile soluzione a nuove minacce.
Nel Capitolo 2 sono state introdotte le nozioni basilari per comprendere il funzionamento di una QKD, come i qubit ed il loro significato. Si è poi spiegato il protocollo BB84 così da rendere più chiaro come può essere implementata una QKD ed infine si sono presentati alcuni semplici attacchi da parte di malintenzionati e di come riuscire a contrastarli.
Nel Capitolo 3 si è iniziato ad andare nella pratica del lavoro svolto per questa tesi, introducendo gli strumenti sfruttati per eseguire le simulazioni: la libreria `grapes`, il modello QKD, l'algoritmo dual annealing, ed il software singularity utilizzato per sfruttare il server HPC "Blade".
Nel Capitolo 4 si è parlato del cuore di questa tesi: le simulazioni. Si sono prima introdotti gli scenari utilizzati considerando due esperimenti realmente in corso ed un semplice esempio fatto per prendere familiarità con il codice utilizzato nelle simulazioni. Infine, si sono presentati i risultati delle ottimizzazioni descrivendo e spiegando le motivazioni dei dati raccolti.
Tramite le simulazioni è stato possibile capire che i parametri e i dispositivi utilizzati per gli esperimenti potrebbero essere ottimizzati ulteriormente, ma comunque nelle condizioni attuali il livello è ottimo e non si avrebbe un miglioramento di prestazioni importante. Se però si volessero utilizzare gli stessi dispositivi in uno spettro di condizioni più ampio, sarebbe meglio ritoccare i vari parametri per il nuovo contesto, in quanto si potrebbe far sì che i dispositivi funzionino anche in

scenari peggiori.
Concludo dicendo che questa tesi potrà avere anche un impatto utile alla valutazione futura dispositivi per QKD in quanto il programma utilizzato per le simulazioni è stato fatto in modo tale da essere versatile ed adattabile a qualsiasi esperimento e contesto.

Bibliografia

[1] Marco Avesani. Security of quantum protocols certified by the dimension of the hilbert space, 2016.

[2] Valerio Scarani, Helle Bechmann-Pasquinucci, Nicolas J. Cerf, Miloslav Duš ek, Norbert Lütkenhaus, and Momtchil Peev. The security of practical quantum key distribution. *Reviews of Modern Physics*, 81(3):1301–1350, sep 2009.

[3] Giulio Foletto. *General Measurements and Advanced Tools for Quantum Information Protocols*. PhD thesis, Università degli Studi di Padova, 2022.

[4] Vasileios Mavroeidis, Kamer Vishi, Mateusz D., and Audun Jøsang. The impact of quantum computing on present cryptography. *International Journal of Advanced Computer Science and Applications*, 9(3), 2018.

[5] Alberto Carrasco-Casado, Veronica Marmol, and Natalia Denisenko. *Free-Space Quantum Key Distribution*, pages 589–607. 08 2016.

[6] Grapes: A simple library for dataflow programming in python, `https://github.com/giuliofoletto/grapes`.

[7] Dual annealing optimization with python. `https://machinelearningmastery.com/dual-annealing-optimization-with-python/`. Visitato il: 20/08/2023.

[8] Omid Ghasemalizadeh, Meysam Khaleghian, and Saied Taheri. A review of optimization techniques in artificial networks. *International Journal of Advanced Research*, 4:1668–1686, 09 2016.

[9] Che cosa sono i container? `https://cloud.google.com/learn/what-are-containers?hl=it`. Visitato il: 21/08/2023.

[10] Cosa sono i container. https://www.ibm.com/it-it/topics/containers. Visitato il: 21/08/2023.

[11] Singularity containers. https://hpc.math.unipd.it/singularity-containers/. Visitato il: 22/08/2023.

[12] 'Blade' computing cluster. https://www.dei.unipd.it/bladecluster. Visitato il: 22/08/2023.

[13] Containers on the hpc clusters. https://researchcomputing.princeton.edu/support/knowledge-base/singularity. Visitato il: 23/08/2023.

[14] Introduction to singularity. https://kks32-courses.gitbook.io/hpc-containers/singularity. Visitato il: 23/08/2023.

Ringraziamenti

Grazie mamma, papà e Edo per aver perseverato e creduto in me tra spronate e rassicurazioni.
Grazie Sofia per dare al mio cuore ciò di cui ha bisogno e darmi una felicità che non sapevo di poter provare.
Grazie nonna Linda per tutti i manicaretti e per il bene incodizionato che sai dare.
Grazie al resto della mia famiglia per essere sempre insieme pronti a divertirci e scannarci.
Grazie Frighesh, Zarro, Deno, Grecc, Pola, Benny, Venada, Skitto e Sugo per tutti gli anni di compagnia e pazzia, nonchè per essere il miglior gruppo di amici che si potesse chiedere.
Grazie Giulio, per il tempo che hai impiegato e per avermi guidato in questo mondo quasi fantascientifico che tanto mi incuriosiva.