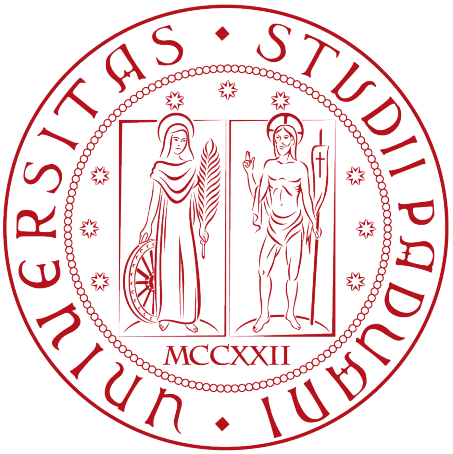
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Implementazione di un'Autenticazione dei Messaggi a Livello Fisico per OpenVLC

Tesi di laurea

Relatore

Dott. Stefano Cecconello

Laureando

Francesco Lapenna

Matricola 2072134

Anno Accademico 2024-2025

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Francesco Lapenna presso il Dipartimento di Matematica dell'Università degli Studi di Padova. Gli obbiettivi da raggiungere erano molteplici.

In primo luogo era richiesta la configurazione di due schede BeagleBone Black per consentire la comunicazione tramite luce visibile. In secondo luogo era richiesta l'implementazione di un modulo di autenticazione sulla piattaforma OpenVLC seguendo lo standard IEEE 802.15.7. Tale modulo permette di autenticare i messaggi con basso impatto sulla velocità di trasmissione e di individuare pacchetti non validi già al primo stadio della comunicazione. Infine era richiesto di testare il sistema in diverse condizioni ambientali.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Dott. Stefano Cecconello, relatore della mia tesi e al Prof. Alessandro Brighente, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto la mia famiglia per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Padova, Luglio 2025 Francesco Lapenna

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Contesto e motivazioni

La Visible Light Communication (VLC)[g]è una tecnologia wireless che sfrutta la luce visibile, tipicamente LED, per trasmettere dati. Questo è reso possibile grazie alla modulazione della luce, ovvero a uno "sfarfallio" impercettibile all'occhio umano.

I principali vantaggi che rendono questa tecnologia innovativa sono l'efficienza energetica, poiché VLC sfrutta fonti di luce già designate all'illuminazione al fine di trasmettere dati, e la sicurezza, in quanto la luce visibile, a differenza delle onde radio, non può penetrare superfici opache. Il tutto garantendo al contempo una comunicazione ad alta velocità.

Tra i principali ambiti di applicazione vi sono: Intelligent Transportation System (ITS)[g], permette ai veicoli di comunicare tra loro e con le infrastrutture circostanti in modo tale da ridurre il rischio di incidenti e ottimizzare il traffico; *Smart Cities* e *Smart Homes*, dove la comunicazione tramite luce visibile può essere sfruttata per la comunicazione sicura e veloce tra dispositivi Internet of Things (IoT)[g]; Ambienti ospedalieri, nei quali la VLC può offrire maggiore sicurezza riducendo eventuali rischi per la salute associati alle onde radio e minimizzando le interferenze con le apparecchiature mediche sensibili.

Sebbene la VLC, come menzionato in precedenza, presenti di per sé caratteristiche che la rendano sicura, è comunque vulnerabile ad attacchi di tipo *spoofing* e *packet-injection*, in cui l'attaccante impersona un *device* legittimo, *replay attacks*, in cui

l'attaccante intercetta e ritrasmette pacchetti legittimi, e Denial Of Service (DoS), in cui l'attaccante inonda la rete con pacchetti per esaurire le risorse del sistema.

La VLC è regolata dallo standard IEEE 802.15.7 che attualmente non definisce protocolli di sicurezza a livello fisico, in quanto assume che il canale di comunicazione sia sicuro e non accessibile a terzi e perciò delega la sicurezza ai livelli superiori. Tuttavia questa assunzione non è valida in tutti gli ambiti di applicazione, come ad esempio in ambienti aperti o in scenari di ITS, dove i dispositivi possono essere facilmente accessibili da parte di attaccanti esterni. In questi scenari, la sicurezza a livello fisico è decisamente vantaggiosa, in quanto permette al sistema di rilevare e scartare pacchetti sospetti prima che vengano elaborati dai livelli superiori, riducendo così il consumo di risorse computazionali ed energetiche.

1.2 Obiettivi della tesi

Questo progetto di tesi si propone di sviluppare e integrare nel *framework OpenVLC*. URL: `http://www.openvlc.org/`, un modulo di autenticazione a livello fisico che permetta di prevenire alcuni tra i possibili rischi citati in precedenza, come *replay attacks*, *spoofing* e *packet injection*, scartando pacchetti illeciti il prima possibile nella comunicazione e mantenendo al contempo le prestazioni del sistema.

1.3 Organizzazione del testo

Il secondo capitolo fornisce un background teoretico e tecnico necessario a comprendere il contesto e le tecnologie coinvolte nel progetto.

Il terzo capitolo approfondisce la struttura dell'ambiente OpenVLC, le eventuali minacce alla sicurezza e le possibili contromisure.

Il quarto capitolo descrive le scelte di progettazione ed i dettagli di implementazione del modulo di autenticazione sviluppato.

Il quinto capitolo descrive i test eseguiti e analizza le performance del sistema sviluppato.

Nel sesto capitolo si riassumono i risultati ottenuti e i possibili sviluppi futuri.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Background

In questo capitolo si introducono brevemente i concetti fondamentali necessari alla comprensione del contenuto della tesi.

2.1 Visible light communication

Nelle telecomunicazioni, la VLC, talvolta indicata anche come *"LiFi"*, prevede l'uso della luce visibile (cioè la luce avente frequenza di 400-800 THz e lunghezza d'onda di 780-375 nm) come mezzo di trasmissione. La VLC è un sottoinsieme delle tecnologie di comunicazione ottica wireless.

Questa tecnologia usa comuni lampade fluorescenti e LED per trasmettere segnali da 10 kbit/s fino a 500 Mbit/s su corte distanze.

Generalmente il segnale luminoso viene ricevuto da un dispositivo elettronico dotato di fotodiodo. Tuttavia in alcuni casi è sufficiente una fotocamera digitale, la quale, essendo un insieme di fotodiodi, potrebbe addirittura essere preferibile in quanto è in grado di ricevere segnali luminosi a diverse frequenze, permettendo così la trasmissione di più canali contemporaneamente.

2.1.1 Vantaggi e applicazioni

Una delle principali caratteristiche della VLC, nonché la principale ragione della sua sicurezza, è l'incapacità della luce visibile di attraversare superfici opache. Se la comunicazione avviene in un ambiente chiuso, tale caratteristica permette di

confinare la comunicazione a quell'ambiente. Di conseguenza, costringe un potenziale attaccante che vuole intercettare la comunicazione ad avere accesso fisico a tale ambiente.

Un'altra caratteristica della VLC è la possibilità di essere integrata a fonti di luce pre-esistenti, e di conseguenza essere usata al duplice scopo di fornire luce e trasmettere dati. Questa caratteristica apre le porte all'Ubiquitous Computing (ubicomp)[g], in quanto i dispositivi che emettono luce (come lampade, LED, schermi, semafori, ecc.) sono già presenti in abbondanza in ogni ambiente.

Un ulteriore ambito di applicazione promettente è l'Indoor Positioning System (IPS)[g]. Analogamente al GPS, permette di localizzare un dispositivo, ma è in grado di operare in spazi chiusi dove il GPS non è disponibile.

2.1.2 Standard IEEE 802.15.7

L'Institute of Electrical and Electronics Engineers (IEEE)[g]è un'organizzazione internazionale di ingegneri che si occupa di standardizzare le tecnologie di comunicazione.

Lo standard che regola la VLC è l'IEEE 802.15.7. Questo standard definisce le specifiche per la comunicazione ottica wireless a corto raggio. Si concentra, in particolare, sul livello fisico e sul livello di accesso al mezzo.

Questo standard è in grado di fornire velocità di trasmissione dati sufficienti a supportare servizi multimediali audio e video e considera anche la mobilità del collegamento ottico, la compatibilità con varie infrastrutture luminose, le limitazioni dovute al rumore e alle interferenze da fonti come la luce ambientale e un sottolivello MAC che soddisfa le esigenze uniche dei collegamenti visibili e delle altre lunghezze d'onda della luce.

Tuttavia, non definisce protocolli di sicurezza a livello fisico, in quanto assume che il canale di comunicazione sia sicuro e non accessibile a terzi. Perciò delega la sicurezza ai livelli superiori.

2.2 Modello OSI

L'Open Systems Interconnection Model (OSI)[g]è lo standard di riferimento per le reti di calcolatori. Sviluppato dalla International Organization for Standardization (ISO), definisce un'architettura sviluppata su sette livelli: *Physical*, *Data Link*, *Network*, *Transport*, *Session*, *Presentation*, ed *Application*.
I livelli più inerenti a quanto trattato in questa tesi sono i primi due: Physical Layer (PHY)[g]e Medium Access Control Layer (MAC)[g]il quale è un sottolivello del livello *Data Link*.

Il PHY è il primo livello del modello OSI. Si occupa della conversione di dati in segnali elettrici, radio o ottici e della trasmissione e ricezione di tali dati grezzi su un canale di comunicazione fisico.

il MAC è il livello che controlla l'*hardware* responsabile dell'interazione con il mezzo di trasmissione. Il livello MAC è responsabile della gestione dell'accesso al mezzo di trasmissione, della sincronizzazione dei dispositivi e della gestione degli errori. Insieme al *Logical Link Control layer (LLC)* costituisce il *Data Link layer* del modello OSI.

2.3 PhyAuth USENIX

Un progetto di ricerca recente, analogo a quello discusso in questa tesi, e da cui è stata presa ispirazione, è PhyAuth.
PhyAuth è un *framework* di autenticazione *hop-by-hop* dei messaggi a livello fisico. Il suo scopo è quello di difendere le reti ZigBee da attacchi packet-injection.

L'idea alla base di PhyAuth è quella di inglobare, nei segnali fisici di ogni messaggio trasmesso, una One Time Password (OTP)[g]calcolata sulla base di una chiave segreta *device*-specifica ed una funzione crittografica di hash.
La validità di tale chiave viene verificata dal dispositivo ricevente, il quale, se la verifica non va a buon fine, scarta l'intero pacchetto.

Questo *framework* permette di rilevare accuratamente pacchetti sospetti con poco impatto sulle prestazioni del sistema e con poche modifiche al protocollo ZigBee, garantendo inoltre compatibilità con i dispositivi su cui non è installato.

2.4 OpenVLC e BeagleBone

OpenVLC è una piattaforma *open source* per la comunicazione ottica wireless, sviluppata dal Pervasive Wireless Systems group del Dr. Giustiniano all'IMDEA Networks Institute (Madrid, Spagna). OpenVLC è progettata per essere flessibile e *low-cost*, permettendo, nella versione 1.4 attualmente in sviluppo, la trasmissione di dati ad una velocità di 400 kb/s e ad una distanza di quasi 20 metri.

OpenVLC è progettato per funzionare su BeagleBone Black, una piattaforma *hardware open source low-cost* per sviluppatori, dotata di un processore ARM Cortex-A8 e di una serie di interfacce di comunicazione, tra cui USB, Ethernet e GPIO. BeagleBone è particolarmente adatto per applicazioni *embedded* e IoT, grazie alla sua flessibilità e alle sue capacità di elaborazione.

Capitolo 3

Analisi

In questo capitolo si approfondisce in dettaglio il funzionamento del framework OpenVLC, le minacce alla sicurezza a cui è soggetto e le possibili contromisure. Inoltre si espone la soluzione proposta per attenuare alcune di queste problematiche.

3.1 Analisi della piattaforma OpenVLC

Introduzione. OpenVLC è una piattaforma *open source* per la comunicazione ottica wireless, sviluppata dal Pervasive Wireless Systems group del Dr. Giustiniano all'IMDEA Networks Institute (Madrid, Spagna). OpenVLC è progettata per consentire agli sviluppatori di sperimentare con la VLC, per essere flessibile e *low-cost*, permettendo, nella versione 1.4 attualmente in sviluppo, la trasmissione di dati ad una velocità di 400 kb/s e ad una distanza di quasi 20 metri.

Motivazioni. Si è scelto di utilizzare OpenVLC proprio a causa della sua accessibilità e popolarità. Infatti, nonostante sia un progetto di piccole dimensioni, possiede delle caratteristiche, come ad esempio l'essere *open source* e l'avere una *community* di discrete dimensioni, che lo rendono la soluzione ideale, e probabilmente l'unica, per la ricerca in ambito VLC.
È infatti importante evidenziare, che durante lo svolgimento del progetto, a causa di problematiche riscontrate nella configurazione delle due schede BeagleBone, è

stata valutata l'idea di trovare un'alternativa ad OpenVLC. Si è considerato, ad esempio, l'utilizzo di un *software* diverso o addirittura di un qualche programma che permettesse di simulare una comunicazione tramite luce visibile. Tuttavia a seguito di varie ricerche, si è concluso che OpenVLC fosse l'unica soluzione *open source* attualmente in circolazione.

Hardware. OpenVLC si appoggia su schede BeagleBone Black, una piattaforma *hardware open source low-cost* per sviluppatori, dotata di un processore ARM Cortex-A8 e di una serie di interfacce di comunicazione, tra cui USB, Ethernet e GPIO. BeagleBone è particolarmente adatto per applicazioni *embedded* e IoT, grazie alla sua flessibilità e alle sue capacità di elaborazione.
La seconda componente *hardware* fondamentale per il funzionamento di OpenVLC è l'*OpenVLC1.3 cape*, una scheda di espansione progettata per la BeagleBone Black. Questo *cape* integra i circuiti necessari per la trasmissione e la ricezione di segnali luminosi, includendo un LED per l'emissione e un fotodiodo per la ricezione. Inoltre, il *cape* dispone di circuiti di amplificazione e filtraggio per garantire una comunicazione affidabile e ridurre il rumore di fondo. L'interfaccia tra la BeagleBone e il *cape* avviene tramite i pin GPIO, permettendo una gestione diretta dei segnali ottici tramite *software*.

Driver e Sistema Operativo. Il codice sorgente di OpenVLC è formato da due componenti: il *software*, scritto in linguaggio C, necessario a gestire i livelli PHY e MAC e ad interfacciarli con i livelli superiori del modello OSI; il firmware necessario a controllare i Programmable Real-Time Unit (PRU) delle schede BeagleBone.
Nella versione 1.3 è stata introdotta la possibilità di utilizzare anche un LED infrarossi per trasmettere dati, il quale può affiancare il classico LED oppure essere usato singolarmente. Di conseguenza, nel caso del trasmettitore, il *framework* può essere selezionato tra varie possibilità a seconda del livello di *dimming* desiderato. Ovvero in funzione di quanto si desidera usare il LED VL piuttosto che il LED IR. Di seguito le possibili opzioni fornite in OpenVLC:

- 0% Dimming (solo luce visibile)

- 25% Dimming
- 50% Dimming
- 75% Dimming
- 100% Dimming (solo infrarossi)

Nell'ambito del progetto qui presentato, trattandosi di VLC, si è deciso di usare la prima opzione, ovvero solo luce visibile.

In figura 3.1 si possono osservare una scheda BeagleBone Black (a sinistra) ed un *OpenVLC1.3 cape*. Sul *cape*, *IR LED* indica il LED infrarossi, *VL LED* indica il LED visible light e *PD* indica il fotodiodo ricevitore.

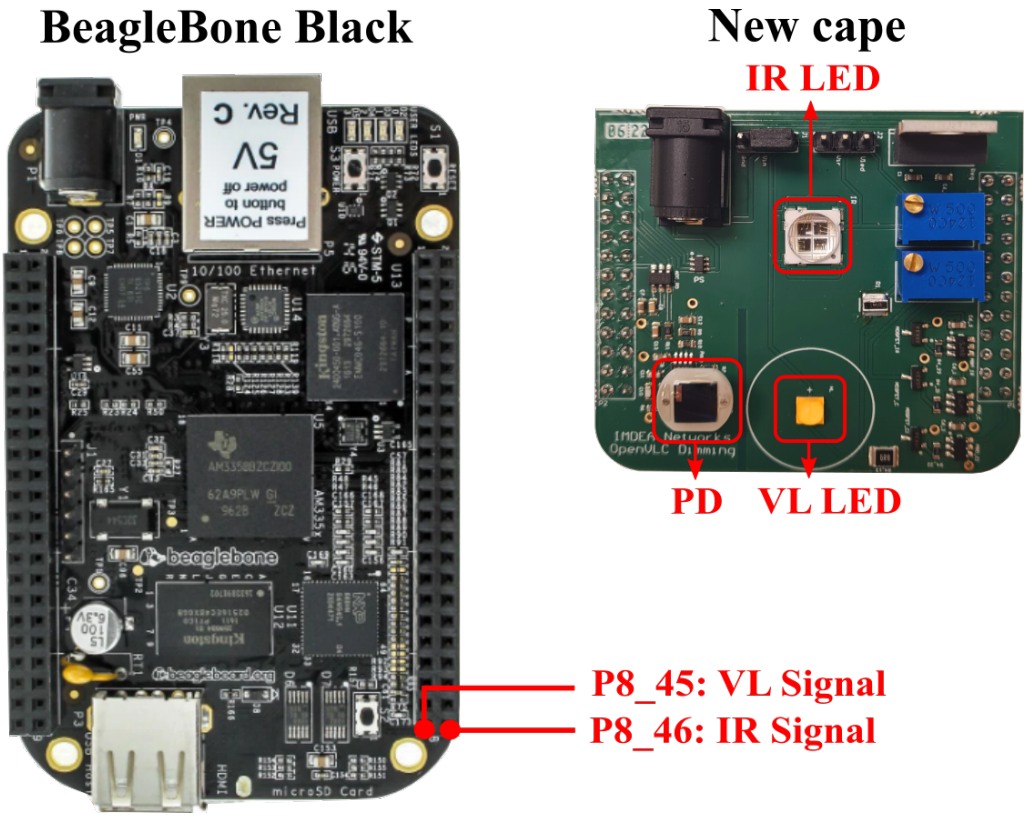


Figura 3.1: a sinistra una scheda BeagleBone Black, a destra un *OpenVLC1.3 cape* (fonte: documentazione OpenVLC)

Durante la configurazione, sulle schede BeagleBone, viene installato il sistema operativo *Debian 8 "jessie"*. Il driver di OpenVLC è di fatto un modulo del *kernel Linux* che agisce da ponte tra lo *stack* di rete del sistema operativo e l'*hardware* BeagleBone.

Reti e Sicurezza. Una volta configurate correttamente due schede, la prima come trasmettitore e la seconda come ricevitore, è quindi possibile usarle come una *common network interface*. Ovvero il sistema operativo, in questo caso *Debian 8*, può usare OpenVLC allo stesso modo in cui userebbe una normale interfaccia di rete, come ad esempio inviare/ricevere pacchetti IP e comunicare con altri dispositivi su una rete locale.
Come già anticipato, ognuna delle due schede ha uno ed un solo ruolo, non è infatti possibile attualmente trasmettere e ricevere dalla stessa scheda contemporaneamente. Questo tipo di canale di comunicazione viene denominato *simplex* (unidirezionale), in contrapposizione ad esempio ad un canale *duplex* in cui un *device* può cambiare ruolo dinamicamente o ad un canale *full-duplex* in cui un *device* può trasmettere e ricevere simultaneamente.
In OpenVLC, per cambiare il ruolo di un *device*, bisogna di fatto ricompilare il driver.

OpenVLC, essendo un progetto di piccole dimensioni ed avendo come obbiettivo quello di sperimentare con la VLC, piuttosto che usarla per comunicazioni realistiche, non implementa delle vere e proprie reti o un'infrastruttura *hardware/software* che permetta di implementare delle reti di calcolatori.

Per quanto riguarda invece la sicurezza, il driver di OpenVLC, gestendo i livelli PHY e MAC, non implementa alcun meccanismo di sicurezza. Lascia invece che questa sia gestita dai livelli superiori. Tuttavia definire protocolli di sicurezza a livello fisico è decisamente utile, in particolar modo se si considera che la VLC viene utilizzata in contesti in cui la sicurezza delle reti di comunicazione è critica, come ad esempio ospedali e convogli di veicoli.

In figura 3.2 si riporta uno schema riassuntivo dell'architettura di OpenVLC in cui si può osservare quanto descritto precedentemente.

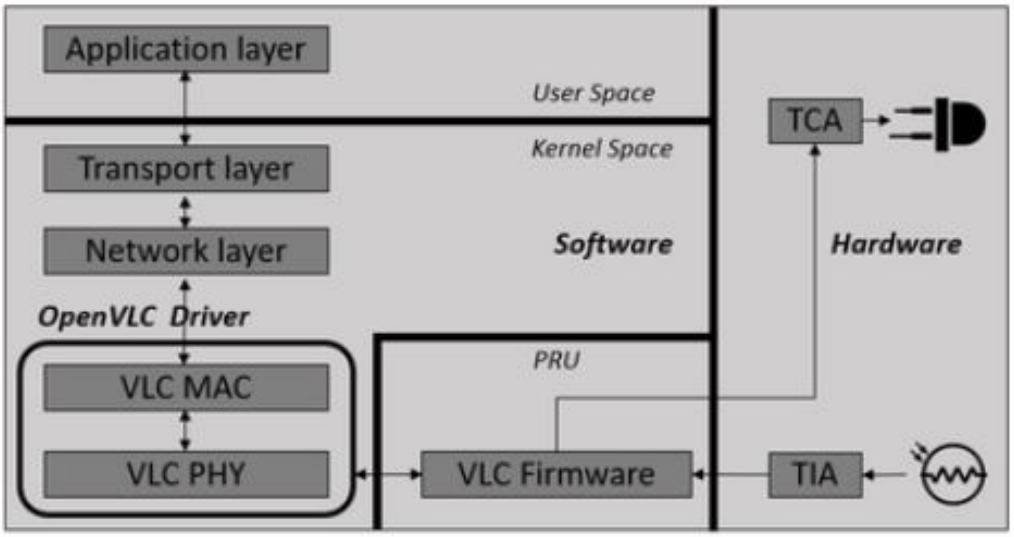


Figura 3.2: Architettura di OpenVLC (fonte: documentazione OpenVLC)

PHY Frame. Come mostrato in figura 3.3, il PHY *frame*, cioè la struttura del pacchetto dati a livello fisico, si suddivide in diversi campi.

Ogni pacchetto inizia con un preambolo, lungo quattro byte, necessario alla sincronizzazione dei *device*, a cui segue lo *Start of Frame Delimiter (SFD)* che indica l'inizio del *frame*.

Successivamente vengono inviati quattro campi da due byte ciascuno: la lunghezza in byte del *frame* (*Frame Length*), l'indirizzo del *device* a cui il pacchetto è destinato (*Destination*), l'indirizzo del *device* da cui proviene (*Source*) e il protocollo di comunicazione (*Protocol*).

A questo punto sono presenti i dati veri e propri, contenuti nel *Payload*, il quale ha una lunghezza variabile da 0 a 255 byte.

Infine, nel campo *Error Correction Code*, si hanno 16 byte generati da Reed-Solomon necessari alla correzione degli errori.

Preamble	Start of Frame Delimiter (SFD)	Frame Length	Destination	Source	Protocol	Payload	Error Correction Code
Bytes: 4	1	2	2	2	2	0 . . . 255	16

Figura 3.3: Struttura del PHY frame in OpenVLC

Modulazione e Codifica. Come definito dallo standard IEEE 802.15.7, OpenVLC utilizza la modulazione On–off keying (OOK)[g]nella trasmissione dei dati. In VLC,

la modulazione dei dati riguarda il modo in cui le informazioni digitali vengono convertite in un segnale analogico tramite la variazione dell'intensità luminosa. In pratica i dispositivi trasmettono bit mediante uno "sfarfallio" del LED impercettibile all'occhio umano. Si ricorda infatti che lo scopo primario del LED è l'illuminazione, di conseguenza è necessario che rimanga costantemente acceso e che questo "sfarfallio" sia impercettibile in modo tale da non essere fastidioso per gli occhi.
In VLC, la modulazione OOK, è la forma più semplice di modulazione Amplitude Shift Keying (ASK), la quale rappresenta i dati digitali come presenza o assenza di un segnale luminoso che vengono interpretate rispettivamente come bit-1 e bit-0. Questa modulazione è facile da implementare ed ha un basso costo computazionale, tuttavia richiede una sincronizzazione molto precisa tra trasmettitore e ricevitore.

Al fianco della modulazione OOK, viene usata la codifica *Manchester*, una forma di comunicazione dati nella quale ogni punto viene segnalato da una transizione. Ad esempio, la transizione del segnale, in questo caso luminoso, da basso ad alto viene interpretato dalla scheda ricevente come bit-1; viceversa da alto a basso viene interpretato come un bit-0.
Nella VLC ha il vantaggio di mantenere il bilanciamento DC, riducendo il rischio di *flickering* ("sfarfallio") visibile e migliora la sincronizzazione tra trasmettitore e ricevitore. In OpenVLC l'unico campo del PHY *frame* in cui non viene applicata la codifica Manchester è il preambolo, che di conseguenza viene trasmesso in semplice OOK.

Correzione degli Errori. In OpenVLC, la correzione degli errori è gestita tramite i codici Reed-Solomon, ovvero codici di correzione degli errori applicati a livello fisico nella VLC per migliorare l'affidabilità della comunicazione e compensare gli errori introdotti da disturbi ambientali.
Sia k il numero di byte informativi ed n il numero di byte totali trasmessi, Reed-Solomon RS(n,k) è in grado di correggere fino a (n-k)/2 byte errati.

Nel caso di OpenVLC viene usato RS(216,200), di conseguenza vengono aggiunti 16 byte di correzione degli errori per 200 byte informativi. RS(216,200) non possiede una grandissima capacità di correzione degli errori, ma essendo OpenVLC orientato

alla velocità piuttosto che all'affidabilità, riduce sicuramente latenza ed overhead rispetto a varianti di Reed-Solomon più performanti.

Configurazione. I dettagli su come è stato configurato il sistema in questo progetto, sulle problematiche riscontrate e su come sono state risolte, si possono trovare nell'appendice A.

3.2 Vulnerabilità

Di seguito vengono elencate le principali vulnerabilità individuate della VLC, ed in particolare di OpenVLC, che come menzionato precedentemente non presenta alcun meccanismo di sicurezza, essendo una piattaforma sperimentale ed orientata alla velocità.
Queste vulnerabilità sono particolarmente rilevanti in contesti in cui la sicurezza della rete è critica, come ambienti industriali, ospedalieri o automotive.

- **Injection di pacchetti.** In assenza di autenticazione, un dispositivo non autorizzato può trasmettere pacchetti arbitrari sulla rete VLC, simulando il comportamento di un nodo legittimo e causando malfunzionamenti o attacchi di *spoofing*;
- **Replay di messaggi.** Senza meccanismi di autenticazione e protezione, è possibile catturare e ritrasmettere pacchetti validi, inducendo il ricevitore ad accettare messaggi duplicati o obsoleti;
- **Man in the Middle (MitM)[g].** Un attaccante può inserirsi tra trasmettitore e ricevitore, modificando i pacchetti in transito o iniettando messaggi malevoli, compromettendo l'integrità e l'autenticità della comunicazione;
- **Intercettazione del segnale ottico.** Un attaccante posizionato in modo opportuno può intercettare il fascio luminoso trasmesso tra i dispositivi, acquisendo i pacchetti scambiati e potenzialmente informazioni sensibili come password o dati privati. Come si osserverà dai risultati dei test nel capitolo 5,

considerando che la luce visibile non si propaga in maniera totalmente unidirezionale, questo tipo di attacco non è una possibilità remota, nel momento in cui l'attaccante ha accesso all'ambiente in cui avviene la comunicazione;

- **Distruzione o alterazione dei pacchetti.** Un attaccante può disturbare la comunicazione inviando segnali ottici che interferiscono con la trasmissione, causando la perdita o la corruzione dei dati.

3.2.1 Principi di crittografia e limitazioni

Prima di poter parlare della soluzione proposta è necessario specificare che l'autenticazione è solo uno dei pilastri della sicurezza dell'informazione. Infatti, per garantire la robustezza del sistema da tutti questi possibili attacchi, è indispensabile affiancare all'autenticazione la confidenzialità, l'integrità e la correzione degli errori.

L'autenticazione infatti garantisce l'identità del mittente di un messaggio, ma non impedisce a terzi di leggere il contenuto del messaggio. Questo si può ottenere però dalla confidenzialità, grazie alla quale solo il destinatario è in grado di decifrare e leggere il messaggio.
Allo stesso modo, non garantisce che il messaggio non sia manipolato in modo tale da risultare proveniente dallo stesso mittente, ma con un contenuto diverso dall'originale. Implementare dei meccanismi di verifica dell'integrità assicura al destinatario che il messaggio non sia stato manipolato in alcun modo.
Parallelamente, la correzione degli errori garantisce una maggior resilienza nel caso in cui l'attaccante voglia solo disturbare la comunicazione.

Nella pratica, la soluzione più comune è la crittografia simmetrica. Secondo questa tecnica di cifratura dei messaggi, prima della comunicazione i due *device* si scambiano una chiave condivisa attraverso un canale sicuro, e usano questa chiave per cifrare i messaggi. In questo modo la crittografia simmetrica garantisce simultaneamente autenticazione e confidenzialità.

Ovviamente ad un aumento della sicurezza, corrispondono inevitabilmente rallentamenti nella comunicazione, in quanto è necessaria una maggiore elaborazione dei

dati. Pertanto, bisogna contestualizzare e comprendere le reali necessità del sistema per bilanciare questo *trade-off*.

3.3 Soluzione proposta

Per far fronte a tali vulnerabilità, la soluzione che si propone è l'autenticazione a livello fisico dei messaggi. L'idea di fondo è di fare in modo che il *device* trasmettitore incorpori nei suoi segnali fisici una OTP derivata da una Pre-Shared Key (PSK)[g]. Il *device* ricevitore rileva tale OTP, la estrae dai segnali fisici e la verifica, autenticando, se risulta valida, il *device* trasmettitore. Se invece non viene rilevata una OTP valida, il pacchetto viene scartato prima di ulteriori elaborazioni.

La PSK è una chiave segreta condivisa tra i due dispositivi prima dell'inizio della comunicazione. Può essere condivisa attraverso un canale di comunicazione sicuro, oppure, come nel caso di questo progetto, può essere *"hard-coded"*, cioè codificata direttamente nel codice sorgente dei dispositivi.

Vulnerabilità Contrastate dall'Autenticazione. L'autenticazione sviluppata in questo progetto è volta a contrastare un sottoinsieme delle vulnerabilità sopra elencate, in particolare *injection* di pacchetti e *replay* dei messaggi.

Nell'*injection* di pacchetti, l'attaccante si finge un nodo legittimo della rete e mette in circolazione pacchetti aventi un contenuto malevolo o semplicemente volti a congestionarla, deteriorando così la qualità della comunicazione e consumando le risorse degli altri dispositivi. Questo tipo di attacco può essere prevenuto dall'autenticazione a livello fisico, in quanto ogni pacchetto che non possiede una OTP valida, come ad esempio quelli forgiati dall'attaccante (che si suppone non abbia la PSK necessaria a generare una OTP valida), viene prontamente scartato, evitando ulteriori elaborazioni che possano depletare le risorse del *device* o addirittura eseguire codice malevolo.

Anche nel caso in cui l'attaccante abbia il pieno controllo di un nodo legittimo (e di conseguenza anche della PSK), e quindi sia in grado di generare OTP valide che superino i controlli a livello fisico, il coordinatore della rete, dato che la OTP è

device-specifica, può risalire al nodo corrotto leggendo la OTP dei pacchetti malevoli e contrassegnarlo come illegittimo.

Nel *replay* dei messaggi, l'attaccante cattura messaggi legittimi inviati da un nodo della rete e, senza modificarli, li ritrasmette in modo tale da congestionare la rete e depletare le risorse del sistema. Anche questo tipo di attacco può essere prevenuto dall'autenticazione sviluppata, poiché, per come è stato progettato l'algoritmo di generazione della OTP, ogni OTP (e di riflesso ogni pacchetto) è valida per un unico utilizzo. Quindi la seconda volta che un pacchetto replicato viene trasmesso, questo verrà scartato dal *device* ricevitore in quanto la OTP in esso contenuta risulterà scaduta.

Possibili Contromisure alle altre Vulnerabilità. Una possibile contromisura all'attacco MitM si può implementare semplicemente aggiungendo un Message Integrity Code (MIC)[g]al PHY *frame*. Ovvero un codice di pochi byte generato da una funzione di hash, calcolato sia sulla OTP che sul payload, che possa essere verificato dal ricevitore, in modo tale da assicurare che il messaggio non abbia subito modifiche non autorizzate durante la trasmissione.

L'intercettazione del segnale ottico si può mitigare attraverso la crittografia dei messaggi, ad esempio cifrando il messaggio con la chiave pubblica del *device* destinatario, in modo tale che esso sia l'unico in grado di decifrarlo e leggerne il contenuto. Oppure usando una chiave privata condivisa tra i due *device* in modo tale da garantire al contempo anche l'autenticazione.

L'ultima vulnerabilità, cioè la distruzione o alterazione dei pacchetti tramite segnali ottici che interferiscono con la comunicazione, è in realtà già mitigata attraverso i codici di correzione degli errori presenti in OpenVLC (Reed-Solomon). Tuttavia si potrebbe pensare di implementare delle varianti capaci di correggere un maggior numero di errori, anche se al costo di rallentare la comunicazione.

Capitolo 4

Progettazione ed implementazione

In questo capitolo si descrive il workflow, le scelte di progettazione ed i dettagli di implementazione del modulo di autenticazione sviluppato.

4.1 Progettazione

Premessa e Workflow. Essendo una prima versione sperimentale, questo metodo di autenticazione, come del resto OpenVLC, è stato progettato per una comunicazione Point-to-point (P2P)[g]e assume che i due nodi abbiano già stabilito una connessione. Il workflow consiste in due passi: il trasmettitore, in fase di invio dei dati, calcola una OTP e la incorpora nei segnali fisici del pacchetto in una posizione nota; il ricevitore, in fase di ricezione, estrae la OTP dai segnali fisici del pacchetto e ne verifica la validità.
I pacchetti che non presentano una valida OTP vengono scartati, altrimenti possono procedere ad ulteriori elaborazioni da parte di OpenVLC.

4.1.1 Generazione e verifica dell'OTP

Una OTP è un codice di autenticazione temporaneo, specifico per dispositivo in quanto coinvolge una chiave segreta e valido per una sola operazione o comunque per un breve lasso di tempo.

Generazione. La formula utilizzata dal trasmettitore per la generazione della OTP è la seguente:

$$\text{POTP}(K, T, SN, \text{src-addr}) = \text{Truncate}\left(\text{HMAC}(K, T, SN, \text{src-addr})\right) \tag{4.1}$$

in cui K denota la PSK, cioè la chiave segreta precondivisa tra le due schede; T è un numero intero che rappresenta l'intervallo di tempo nel quale la OTP è valida; SN è un *sequence number* incrementale; *src-addr* rappresenta l'indirizzo MAC del dispositivo trasmettitore.

La sicurezza di questo tipo di OTP risiede proprio nella sua dipendenza da diversi parametri. Infatti si richiede ai dispositivi la conoscenza della chiave K, la OTP scade una volta concluso il periodo di tempo T, può essere utilizzata un'unica volta a causa del SN ed è strettamente legata al dispositivo che la genera grazie al *src-addr*.

Lunghezza. Per essere resistente ad attacchi *brute force*, una OTP deve avere una lunghezza di almeno 31 bit; naturalmente maggiore è la dimensione della OTP e maggiore è la latenza introdotta nella comunicazione.
Considerato ciò, si è deciso di utilizzare una OTP di lunghezza 32 bit, in modo tale che fosse abbastanza resistente agli attacchi ma senza introdurre rallentamenti percepibili nella comunicazione.
La funzione `Truncate`, come è intuibile, troncherà l'output della funzione HMAC a tale lunghezza.

HMAC. La funzione Hash-based Message Authentication Code (HMAC)[g], è un meccanismo crittografico che consente di verificare l'integrità e l'autenticità di un messaggio. In questo contesto, HMAC viene utilizzato per generare la OTP combinando la chiave segreta K con i parametri T, SN e *src-addr* tramite una funzione di hash sicura. Il risultato è un codice che può essere calcolato solo da chi possiede la chiave K e che varia in base ai parametri forniti, garantendo così che ogni OTP sia unica e valida per un unico utilizzo ed intervallo temporale.

Time-step T. Il *time-step* T è un numero intero che rappresenta il numero di intervalli di tempo (*time-step*) di durata X trascorsi tra T_0 e l'attuale tempo unix. Viene calcolato tramite la seguente formula:

$$T = \frac{\text{current Unix time} - T_0}{X} \tag{4.2}$$

in cui *current Unix time* indica i secondi trascorsi a partire dall'*epoch* Unix, T_0 è un riferimento ad un tempo iniziale arbitrario ed X indica l'intervallo di tempo di validità della OTP. T_0 e X devono avere lo stesso valore in entrambi i dispositivi.
Nel caso specifico di questo progetto, X è stato impostato a 3, quindi ogni OTP è valida per un massimo di tre secondi.

Chiave Segreta K. Come già anticipato, K è una chiave segreta precondivisa tra trasmettitore e ricevitore. In questo progetto ha lunghezza di 128 bit.

Sequence Number SN. SN è un numero intero che incrementa ad ogni OTP generata in un determinato periodo di tempo. Trascorso quel periodo di tempo, infatti, viene azzerato nuovamente.
Il SN è un componente fondamentale in quanto è solo grazie ad esso che la OTP è valida per un solo utilizzo. Infatti il ricevitore, in un determinato intervallo di tempo, salva in memoria il SN più alto ricevuto e non accetta pacchetti con SN minore o uguale ad esso.
Se tale componente non fosse presente, ogni OTP sarebbe valida per un intero intervallo di tempo, in questo caso tre secondi, e la protezione contro *replay attacks* sarebbe notevolmente inficiata.

Verifica. Il ricevitore, nel processo di verifica della OTP, genera una OTP attesa tramite la medesima funzione di generazione utilizzata dal trasmettitore (equazione 4.1) e la confronta con quella ricevuta.
Come parametri per la funzione, vengono usati dei valori attesi, ad esempio, il ricevitore si aspetta che la OTP ricevuta sia stata generata con il SN successivo a quello dell'ultimo pacchetto ricevuto.

In realtà, non viene generata una singola OTP, bensì un insieme di OTP da considerare valide, in quanto si è voluto introdurre un certo grado di flessibilità, in modo tale da migliorare la resilienza del sistema ad eventuali ritardi nella comunicazione o perdite di pacchetti.
Siano T_c e SN_c rispettivamente l'intervallo di tempo corrente e il prossimo SN atteso, siano x ed y rispettivamente il grado di flessibilità per T ed il grado di flessibilità per SN; vengono considerate valide tutte le OTP generate con l'equazione 4.1 in cui $T \in [T_c - x, T_c]$ e $SN \in [SN_c, SN_c + y]$.
Se non fosse stata introdotta questa flessibilità, si supponga che un pacchetto venga trasmesso con un ritardo superiore al *time-step*, quel pacchetto, sebbene legittimo, verrebbe considerato dal ricevitore come illegittimo. Allo stesso modo, supponendo che vengano persi alcuni pacchetti, tutti i successivi (fino al successivo *time-step*) verrebbero considerati erroneamente illegittimi.

4.1.2 Codifica e decodifica dell'OTP

Embedding ed Estrazione. Per inglobare la OTP all'interno dei segnali fisici di ogni pacchetto, si è deciso di usare l'operatore logico Exclusive or (XOR)[g], indicato dal simbolo $\oplus$.
La particolarità dello XOR è che consente di "fondere" l'informazione dell'OTP con i segnali fisici del pacchetto in modo reversibile, grazie alla proprietà di auto-invertibilità dello XOR. Secondo questa proprietà, dato un messaggio M ed una chiave K, se $C = M \oplus K$, allora risulta sempre valido che $M = C \oplus K$.

Questa proprietà è stata sfruttata per nascondere la OTP all'interno di una porzione nota del *frame header*, cioè i campi *Destination* e *Source* (figura 3.3) contenenti gli indirizzi di trasmettitore e ricevitore.
Il trasmettitore, infatti, applica l'operazione di XOR tra la sequenza di byte dell'OTP ed i quattro byte di questi due campi, ottenendo così una sequenza modificata che viene inserita nel pacchetto al posto di tali campi. Al ricevitore, l'operazione inversa di XOR tra la sequenza ricevuta e la porzione nota permette di recuperare l'OTP originale.

Questa tecnica garantisce che l'OTP sia trasmessa in modo non direttamente

leggibile, aumentando la sicurezza contro attacchi di intercettazione. Inoltre, l'operazione di XOR è computazionalmente leggera e non introduce latenza significativa nella comunicazione.

La posizione e la lunghezza della porzione di *frame* utilizzata per l'operazione di XOR sono concordate tra trasmettitore e ricevitore, inoltre in una comunicazione P2P gli indirizzi *Destination* e *Source* sono noti ad entrambi. Ciò permette una corretta estrazione e verifica dell'OTP.

4.2 Implementazione

Il codice implementato, come del resto il codice sorgente di OpenVLC, è stato scritto interamente in linguaggio C, con aderenza allo standard C99. È stato realizzato come un modulo esterno isolato dal codice OpenVLC, al fine di perseguire modularità e minimizzare i punti di contatto con il codice esistente.
Il modulo è organizzato, secondo le best practice del linguaggio C, in due file:

- `PhyAuthP2P.h`, che definisce l'interfaccia dichiarando le funzioni e le costanti necessarie;
- `PhyAuthP2P.c`, che definisce tali funzioni implementando la generazione di OTP.

Funzionalità principali. Il modulo PhyAuthP2P implementa le seguenti tre funzioni fondamentali per la generazione e gestione dell'OTP.

`hmac_sha256()`: questa funzione si occupa di calcolare il codice HMAC utilizzando l'algoritmo SHA-256. Sfrutta le Application Program Interface (API) crittografiche messe a disposizione dal *kernel* Linux, garantendo così efficienza e sicurezza nell'esecuzione. I parametri di input sono la chiave segreta precondivisa (PSK), l'intervallo di tempo corrente, il *sequence number* e l'indirizzo MAC del trasmettitore (*src-addr*). Il risultato prodotto è un digest di 32 byte, che rappresenta il valore hash autenticato e costituisce la base per la generazione dell'OTP.

`truncate()`: questa funzione applica la tecnica di *"dynamic truncation"* descritta nello standard RFC 4226. In pratica, estrae 4 byte dal digest ottenuto dalla funzione

`hmac_sha256`, riducendo così la dimensione dell'output a 32 bit, che corrisponde alla lunghezza scelta per l'OTP. Il processo di *truncation* garantisce che l'OTP sia compatta ma sufficientemente sicura contro attacchi di forza bruta.

`generate_potp()`: rappresenta la funzione principale del modulo. Riceve in ingresso tutti i parametri necessari (chiave segreta, intervallo di tempo, *sequence number*, indirizzo MAC) e richiama in sequenza le funzioni `hmac_sha256` e `truncate` per produrre l'OTP finale. Questa funzione incapsula l'intero processo di generazione, semplificando l'integrazione con il resto del sistema e permettendo sia al trasmettitore che al ricevitore di ottenere rapidamente il codice OTP da utilizzare o verificare.

Queste funzioni sono state progettate per essere modulari e facilmente integrabili nel flusso di OpenVLC, consentendo una gestione sicura e flessibile dell'autenticazione a livello fisico. L'utilizzo delle API del *kernel* assicura inoltre che le operazioni crittografiche siano eseguite in modo ottimale e conforme agli standard di sicurezza.

Interazione con OpenVLC. Quest'ultima funzione di generazione della OTP viene utilizzata, nel codice OpenVLC, sia dal trasmettitore che dal ricevitore. In particolare, il trasmettitore la chiama all'interno di `construct_frame_header()` per generare la OTP e successivamente inglobarla nei segnali fisici. Il ricevitore invece, la chiama all'interno di `phy_decoding()`, dopo aver estratto la OTP contenuta nel pacchetto, utilizzando diversi valori di T ed SN al fine di garantire la flessibilità discussa nella sezione precedente.

Dipendenze Il modulo PhyAuthP2P essendo stato sviluppato come parte del *kernel* Linux richiede le seguenti dipendenze per la corretta compilazione e funzionamento e per implementare le funzionalità crittografiche:

- `<crypto/hash.h>`: fornisce le API per la gestione delle funzioni di hash e HMAC;
- `<linux/crypto.h>`: include le primitive crittografiche del *kernel*;
- `<linux/string.h>`: funzioni di manipolazione delle stringhe;
- `<linux/slab.h>`: gestione della memoria dinamica nel *kernel*;

- `<linux/err.h>`: gestione degli errori;
- `<linux/types.h>`: definizioni dei tipi di dato.

Il codice sviluppato è tracciato nella *repository* del progetto nel *branch* PhyAuthP2P *OpenVLC-PA - GitHub*. URL: https://github.com/francesco-lapenna/OpenVLC-PA.

Capitolo 5

Test e validazione

In questo capitolo si espongono e discutono i test svolti su OpenVLC e sul modulo di autenticazione sviluppato. Il capitolo viene diviso in due sezioni: la prima riguarda i test svolti sul modulo di autenticazione sviluppato; la seconda riguarda test generici sul framework OpenVLC.

5.1 Test sul modulo di autenticazione

In questa sezione si riportano i risultati dei test svolti sul modulo di autenticazione sviluppato. L'obbiettivo di questi test è quello di analizzare le prestazioni di OpenVLC con l'utilizzo del modulo, confrontandole con quelle del *framework* OpenVLC originale. In più, si testa la qualità della comunicazione in diverse condizioni di luminosità ambientale: ambiente luminoso, ambiente buio e ambiente luminoso con disturbo indotto.

Sono stati condotti quattro test, ognuno della durata di circa 30 secondi. In tutti i test le schede sono state posizionate una di fronte all'altra ad una distanza di 100 centimetri. La comunicazione è stata testata utilizzando il comando `iperf`, il quale è un programma di misurazione delle prestazioni di rete. Si è scelto di utilizzare questo strumento anche per essere paragonabile ai test riportati nella documentazione di OpenVLC, anch'essi svolti con `iperf`.

In particolare, i comandi utilizzati per i test sono gli stessi riportati nella documen-

tazione ufficiale di OpenVLC: per il trasmettitore "`sudo iperf -c 192.168.0.2 -u -b 400k -l 800 -p 10001 -t 100`"; mentre per il ricevitore "`sudo iperf -u -l 800 -s -i3 -B 192.168.0.2 -p 10001`".

5.1.1 Data collection

In figura 5.1, si riporta il test iperf della documentazione di OpenVLC.

```
------------------------------------------------------------
Server listening on UDP port 10001
Binding to local address 192.168.0.2
Receiving 800 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 10001 connected with 192.168.0.1 port 44727
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 2.0 sec  97.7 KBytes   400 Kbits/sec   0.034 ms    0/  125 (0%)
[  3]  2.0- 4.0 sec  93.8 KBytes   384 Kbits/sec   0.120 ms    5/  125 (4%)
[  3]  4.0- 6.0 sec  97.7 KBytes   400 Kbits/sec   0.051 ms    0/  125 (0%)
[  3]  6.0- 8.0 sec  97.7 KBytes   400 Kbits/sec   0.096 ms    0/  125 (0%)
[  3]  8.0-10.0 sec  97.7 KBytes   400 Kbits/sec   0.049 ms    0/  125 (0%)
[  3] 10.0-12.0 sec  95.3 KBytes   390 Kbits/sec   0.044 ms    3/  125 (2.4%)
[  3] 12.0-14.0 sec  96.1 KBytes   394 Kbits/sec   0.134 ms    2/  125 (1.6%)
[  3] 14.0-16.0 sec  97.7 KBytes   400 Kbits/sec   0.047 ms    0/  125 (0%)
[  3] 16.0-18.0 sec  97.7 KBytes   400 Kbits/sec   0.072 ms    0/  125 (0%)
[  3] 18.0-20.0 sec  97.7 KBytes   400 Kbits/sec   0.048 ms    0/  125 (0%)
[  3]  0.0-20.0 sec   970 KBytes   397 Kbits/sec   0.049 ms   10/ 1251 (0.8%)
```

Figura 5.1: Test iperf riportato nella documentazione di OpenVLC

In figura 5.2, si riporta il test iperf svolto in ambiente luminoso con OpenVLC originale. Si osserva che le differenze rispetto al test illustrato precedentemente, non sono statisticamente significative.

```
debian@beaglebone:~/OpenVLC-PA/OpenVLC_VL_IR/utils$ sudo ./iperf_RX.sh
------------------------------------------------------------
Server listening on UDP port 10001
Binding to local address 192.168.0.2
Receiving 800 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 10001 connected with 192.168.0.1 port 37912
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 3.0 sec   145 KBytes   397 Kbits/sec   0.094 ms    1/  187 (0.53%)
[  3]  3.0- 6.0 sec   146 KBytes   399 Kbits/sec   0.172 ms    0/  187 (0%)
[  3]  6.0- 9.0 sec   146 KBytes   399 Kbits/sec   0.085 ms    1/  188 (0.53%)
[  3]  9.0-12.0 sec   147 KBytes   401 Kbits/sec   0.094 ms    0/  188 (0%)
[  3] 12.0-15.0 sec   144 KBytes   393 Kbits/sec   0.099 ms    3/  187 (1.6%)
[  3] 15.0-18.0 sec   143 KBytes   390 Kbits/sec   0.303 ms    4/  187 (2.1%)
[  3] 18.0-21.0 sec   145 KBytes   397 Kbits/sec   0.065 ms    2/  188 (1.1%)
[  3] 21.0-24.0 sec   146 KBytes   399 Kbits/sec   0.068 ms    1/  188 (0.53%)
[  3] 24.0-27.0 sec   145 KBytes   397 Kbits/sec   0.071 ms    1/  187 (0.53%)
[  3] 27.0-30.0 sec   141 KBytes   386 Kbits/sec   0.136 ms    6/  187 (3.2%)
[  3]  0.0-30.4 sec  1.43 MBytes   396 Kbits/sec   0.126 ms   19/ 1898 (1%)
```

Figura 5.2: Test iperf con OpenVLC in ambiente luminoso

In figura 5.3, si riporta il test iperf svolto in ambiente luminoso con OpenVLC e con l'utilizzo del modulo di autenticazione sviluppato.

```
debian@beaglebone:~/OpenVLC-PA/OpenVLC_VL_IR/utils$ sudo ./iperf_RX.sh
------------------------------------------------------------
Server listening on UDP port 10001
Binding to local address 192.168.0.2
Receiving 800 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 10001 connected with 192.168.0.1 port 49930
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 3.0 sec   146 KBytes   399 Kbits/sec   0.046 ms    0/  187 (0%)
[  3]  3.0- 6.0 sec   147 KBytes   401 Kbits/sec   0.042 ms    0/  188 (0%)
[  3]  6.0- 9.0 sec   142 KBytes   388 Kbits/sec   0.045 ms    5/  187 (2.7%)
[  3]  9.0-12.0 sec   147 KBytes   401 Kbits/sec   0.037 ms    0/  188 (0%)
[  3] 12.0-15.0 sec   142 KBytes   388 Kbits/sec   0.069 ms    5/  187 (2.7%)
[  3] 15.0-18.0 sec   147 KBytes   401 Kbits/sec   0.044 ms    0/  188 (0%)
[  3] 18.0-21.0 sec   146 KBytes   399 Kbits/sec   0.042 ms    0/  187 (0%)
[  3] 21.0-24.0 sec   145 KBytes   395 Kbits/sec   0.048 ms    3/  188 (1.6%)
[  3] 24.0-27.0 sec   142 KBytes   388 Kbits/sec   0.038 ms    5/  187 (2.7%)
[  3] 27.0-30.0 sec   147 KBytes   401 Kbits/sec   0.048 ms    0/  188 (0%)
[  3]  0.0-30.8 sec  1.45 MBytes   396 Kbits/sec   0.034 ms   18/ 1923 (0.94%)
```

Figura 5.3: Test iperf con OpenVLC e modulo di autenticazione in ambiente luminoso

In figura 5.4, si riporta il test iperf svolto in ambiente buio con OpenVLC e con l'utilizzo del modulo di autenticazione sviluppato.

```
debian@beaglebone:~/OpenVLC-PA/OpenVLC_VL_IR/utils$ sudo ./iperf_RX.sh
------------------------------------------------------------
Server listening on UDP port 10001
Binding to local address 192.168.0.2
Receiving 800 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 10001 connected with 192.168.0.1 port 39105
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 3.0 sec   146 KBytes   399 Kbits/sec   0.052 ms    0/  187 (0%)
[  3]  3.0- 6.0 sec   147 KBytes   401 Kbits/sec   0.054 ms    0/  188 (0%)
[  3]  6.0- 9.0 sec   146 KBytes   399 Kbits/sec   0.051 ms    0/  187 (0%)
[  3]  9.0-12.0 sec   147 KBytes   401 Kbits/sec   0.060 ms    0/  188 (0%)
[  3] 12.0-15.0 sec   146 KBytes   399 Kbits/sec   0.043 ms    0/  187 (0%)
[  3] 15.0-18.0 sec   147 KBytes   401 Kbits/sec   0.046 ms    0/  188 (0%)
[  3] 18.0-21.0 sec   146 KBytes   399 Kbits/sec   0.045 ms    0/  187 (0%)
[  3] 21.0-24.0 sec   147 KBytes   401 Kbits/sec   0.045 ms    0/  188 (0%)
[  3] 24.0-27.0 sec   146 KBytes   399 Kbits/sec   0.055 ms    0/  187 (0%)
[  3] 27.0-30.0 sec   147 KBytes   401 Kbits/sec   0.037 ms    0/  188 (0%)
[  3]  0.0-30.5 sec  1.46 MBytes   400 Kbits/sec   0.050 ms    0/ 1908 (0%)
```

Figura 5.4: Test iperf con OpenVLC e modulo di autenticazione in ambiente buio

In figura 5.5, si riporta il test iperf svolto in ambiente luminoso con OpenVLC e con l'utilizzo del modulo di autenticazione sviluppato. In più, in questo test è stato introdotto un disturbo luminoso, posizionando una lampada diretta verso il fotodiodo

ricevitore ad una distanza di circa 20 centimetri. Tale disturbo è stato indotto per simulare un attacco di disturbo alla comunicazione, in modo tale da testare la robustezza della comunicazione.

```
debian@beaglebone:~/OpenVLC-PA/OpenVLC_VL_IR/utils$ sudo ./iperf_RX.sh
------------------------------------------------------------
Server listening on UDP port 10001
Binding to local address 192.168.0.2
Receiving 800 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 10001 connected with 192.168.0.1 port 48188
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 3.0 sec   130 KBytes   354 Kbits/sec   0.041 ms   21/  187 (11%)
[  3]  3.0- 6.0 sec   143 KBytes   390 Kbits/sec   0.055 ms    5/  188 (2.7%)
[  3]  6.0- 9.0 sec   139 KBytes   380 Kbits/sec   0.046 ms    9/  187 (4.8%)
[  3]  9.0-12.0 sec   133 KBytes   363 Kbits/sec   0.041 ms   17/  187 (9.1%)
[  3] 12.0-15.0 sec   138 KBytes   375 Kbits/sec   0.039 ms   12/  188 (6.4%)
[  3] 15.0-18.0 sec   142 KBytes   388 Kbits/sec   0.048 ms    6/  188 (3.2%)
[  3] 18.0-21.0 sec   145 KBytes   397 Kbits/sec   0.034 ms    1/  187 (0.53%)
[  3] 21.0-24.0 sec   140 KBytes   382 Kbits/sec   0.050 ms    9/  188 (4.8%)
[  3] 24.0-27.0 sec   127 KBytes   348 Kbits/sec   0.056 ms   24/  187 (13%)
[  3] 27.0-30.0 sec   136 KBytes   371 Kbits/sec   0.062 ms   14/  188 (7.4%)
[  3]  0.0-31.6 sec  1.41 MBytes   375 Kbits/sec   0.052 ms  122/ 1973 (6.2%)
```

Figura 5.5: Test iperf con OpenVLC e modulo di autenticazione in ambiente luminoso con disturbo indotto

5.1.2 Analisi dei risultati

Nella seguente tabella si riassumono i risultati dei test descritti nella precedente sezione. I dati sono ricavati dall'ultima riga di ogni test, che rappresenta il valore medio delle metriche calcolate nell'intervallo di tempo in cui si è svolta la comunicazione.

Tabella 5.1: Tabella riassuntiva dei risultati dei test sul modulo di autenticazione

Test	Interval	Transfer	Bandwidth	Jitter	Lost/Total
OpenVLC in ambiente luminoso	30.4 sec	1.43 MBytes	396 Kbits/sec	0.126 ms	19/1898 (1%)
OpenVLC + modulo in ambiente luminoso	30.8 sec	1.45 MBytes	396 Kbits/sec	0.034 ms	18/1923 (0.94%)
OpenVLC + modulo in ambiente buio	30.5 sec	1.46 MBytes	400 Kbits/sec	0.050 ms	0/1988 (0%)
OpenVLC + modulo in ambiente luminoso con disturbo	31.6 sec	1.41 MBytes	375 Kbits/sec	0.052 ms	122/1973 (6.2%)

Come si può osservare dalle prime due righe della tabella 5.1, a parità di condizioni ambientali, non c'è alcuna differenza sistematicamente significativa tra le prestazioni del sistema con e senza l'utilizzo del modulo di autenticazione. Le lievi differenze presenti sono infatti da interpretarsi come rumore. Di conseguenza si può affermare che la sua introduzione non ha un impatto rilevante sulle prestazioni del sistema.

Si possono invece notare differenze significative tra i test svolti in ambiente luminoso e quelli svolti in diverse condizioni di luminosità.
In particolare, il test svolto in ambiente buio ha mostrato un'ampiezza di banda (bandwidth) e un tasso di perdita migliori rispetto a quelli svolti in ambiente luminoso. Questo suggerisce che la comunicazione tramite luce visibile, seppure in maniera minima, trattandosi di una variazione dell'1%, è sicuramente influenzata dalla presenza di luce ambientale.
Inoltre, il test svolto in ambiente luminoso con disturbo indotto ha mostrato un tasso di perdita notevolmente più alto rispetto agli altri test, pari al 6.2%. Si deduce che la presenza di fonti luminose dirette in direzione del fotodiodo ricevitore, sia in grado di disturbare notevolmente la comunicazione. Tuttavia è importante evidenziare che la situazione in cui è stato testato il sistema, e cioè posizionando una fonte luminosa a distanza molto ravvicinata al fotodiodo, è estrema. Di conseguenza, resta

particolarmente difficoltoso, per un potenziale attaccante che volesse disturbare la ricezione dei dati, farlo in questa maniera.

5.2 Test su OpenVLC

In questa sezione si riportano i risultati dei test svolti sul *framework* OpenVLC originale, cioè senza l'utilizzo del modulo di autenticazione sviluppato. L'obbiettivo di questi test è quello di analizzare le prestazioni di OpenVLC a diverse distanze ed inclinazioni.

Sono stati condotti nove test, in ognuno dei quali sono stati trasmessi 1000 pacchetti di dati tramite il comando `nc` (netcat), ognuno della dimensione di 1480 byte. In tutti i test il *payload* era formato da byte "FF", ovvero da 8 bit impostati a "1", in modo tale da poter individuare efficacemente eventuali bit corrotti. L'unica eccezione è il test denominato "50 cm bit-0", in cui il *payload* era formato da byte "00", ovvero da 8 bit impostati a "0", per poter osservare eventuali bias a favore di bit-0 piuttosto che bit-1.

Tutti i test sono stati svolti in ambiente luminoso e senza l'utilizzo di Reed-Solomon, in modo tale da poter osservare le prestazioni del canale fisico senza l'intervento di alcun protocollo di correzione degli errori.

Inoltre, c'è da precisare che le schede BeagleBone non dispongono di una capacità computazionale elevata, di conseguenza il solo fatto di osservare la comunicazione e salvarne i dati, rende le schede maggiormente soggette ad errori. Questo implica che le metriche di errore calcolate durante i test sono sovrastimate rispetto a quelle che il sistema raggiunge in condizioni normali.

I test vengono valutati utilizzando le seguenti metriche:

- BER[g]: rapporto tra il numero di bit ricevuti in errore e il numero totale di bit trasmessi, indicativo dell'affidabilità a livello fisico del canale;
- Percentuale di byte corrotti: percentuale di byte che contengono almeno un bit errato rispetto al totale dei byte trasmessi, utile per valutare la possibilità di correggere errori con Reed-Solomon;

- PRR[g]: frazione di pacchetti correttamente ricevuti sul numero di pacchetti inviati, misura l'efficacia complessiva del link;
- Packet Loss Rate (PLR)[g]: percentuale di pacchetti persi rispetto al totale inviato; è il complementare di PRR;
- Percentuale di *payload* ricevuto: rapporto tra il numero di byte di *payload* validi ricevuti e il numero di byte di *payload* trasmessi, espressa in percentuale, per quantificare l'efficienza utile della trasmissione.

5.2.1 Data collection

Come accennato in precedenza, sono stati condotti nove test, tutti in ambiente luminoso.

L'unico test in cui il *payload* era formato da byte "00" è il test denominato "50 cm bit-0", in cui le schede sono state posizionate una di fronte all'altra ad una distanza di 50 centimetri. In tutti gli altri il *payload* era formato da byte "FF".

Nei test denominati "50 cm bit-1", "100 cm", "150 cm", "200 cm", le schede sono state posizionate perfettamente allineate una di fronte all'altra ad una distanza di 50, 100, 150 e 200 centimetri rispettivamente.

Nei test denominati "50 cm 10°", "50 cm 45°", "50 cm 60°", "50 cm 90°", le schede sono state posizionate ad una distanza di 50 centimetri, ma con un'inclinazione di 10°, 45°, 60° e 90° rispettivamente. Ovvero per ogni inclinazione, la scheda trasmittente è stata ruotata, rispetto alla scheda ricevente, di un angolo corrispondente, mantenendo la stessa distanza.

5.2.2 Analisi dei risultati

Bias a favore di bit-0

Per prima cosa, è necessario confrontare i test "50 cm bit-0" e "50 cm bit-1", in modo tale da osservare se vi è un bias a favore di bit-0 rispetto a bit-1. In tabella 5.2 sono riportate le metriche di errore calcolate per i due test.

Nonostante, a prima vista, la differenza in tutte le metriche sembrerebbe minima, è importante considerare che la numerosità dei campioni è di diversi milioni di bit. Di conseguenza, anche una piccola differenza può essere considerata statisticamente significativa.

Si osserva che le uniche metriche che non presentano una differenza significativa sono il PRR e il PLR. Il che denota, come ci si aspetterebbe date le pari condizioni ambientali in cui sono stati svolti i test, che in entrambi i casi le schede riescono a sincronizzarsi correttamente.

Tuttavia, si osserva che il BER è più basso per i bit-0 rispetto ai bit-1, con un valore di 1.10% per i bit-0 e di 1.25% per i bit-1. Inoltre, la percentuale di byte corrotti è più alta per i bit-1 (2.56%) rispetto ai bit-0 (1.93%). Infine, la percentuale di *payload* ricevuto è più alta per i bit-0 (99.24%) rispetto ai bit-1 (95.83%). Questo indica che i bit-1 tendono a essere corrotti più frequentemente rispetto ai bit-0, il che suggerisce un bias a favore dei bit-0.

La presenza di questo bias comporta che le metriche di errore calcolate per i successivi test, in quanto basati su bit-1, sono da considerarsi sovrastimate. In una normale comunicazione, ci si aspetta un bilanciamento tra i bit-0 e i bit-1, e di conseguenza che le metriche di errore siano comprese tra i valori di errore dei bit-0 e dei bit-1 qui presentati.

Tabella 5.2: Confronto delle metriche di errore di bit-0 e bit-1: bias a favore di bit-0

Metrica	**50 cm bit-0**	**50 cm bit-1**
Bit Error Ratio (BER)	1.10%	1.25%
% Byte corrotti	2.56%	1.93%
Packet Reception Rate (PRR)	93.80%	93.90%
Packet Loss Rate (PLR)	6.20%	6.10%
% Payload Ricevuto	99.24%	95.83%

Test in diverse posizioni

Testando la comunicazione in diverse posizioni, si è osservato che, come ci si aspetterebbe, all'aumentare della distanza e dell'inclinazione tra le schede, tutte le metriche di errore tendono a peggiorare.

In figura 5.6 si riporta il BER in funzione della distanza tra le schede. Si osserva una progressiva degradazione della qualità della comunicazione ed un'accelerazione nel tasso di errore a partire dai 100 centimetri.

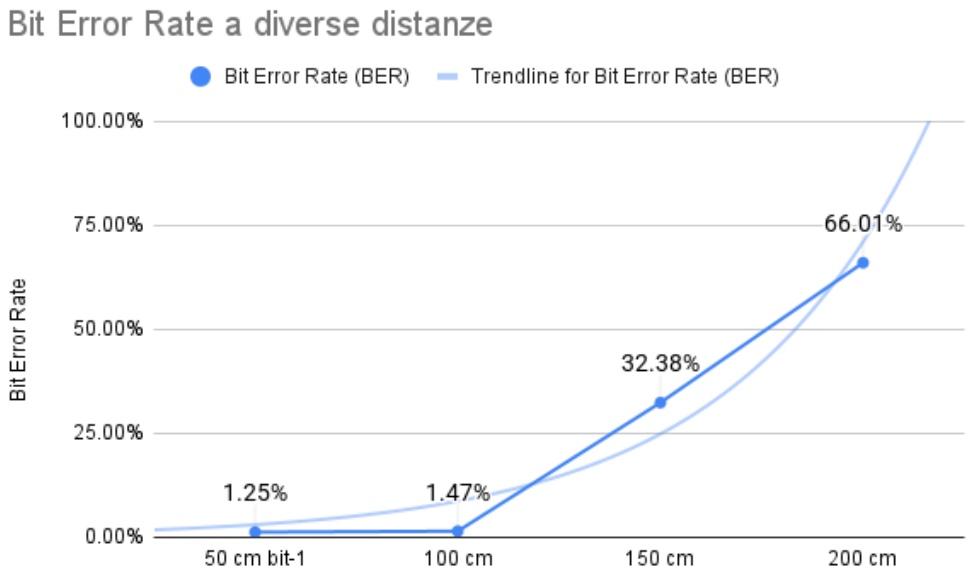


Figura 5.6: Test Bit Error Ratio (BER) in funzione della distanza

Analogamente, in figura 5.7 si riporta il BER in funzione dell'inclinazione tra le schede. Anche in questo caso si osserva una degradazione della comunicazione all'aumentare dell'inclinazione, con un tasso di errore che cresce in maniera esponenziale a partire dai 60°.

È interessante notare che, anche ad un'inclinazione ampia come 60°, il BER è comunque limitato. Il che suggerisce che, contrariamente a quanto si potrebbe pensare, la comunicazione tramite luce visibile non sia strettamente unidirezionale, rendendo possibile l'intercettazione del segnale anche da posizioni angolate.
Tuttavia, bisogna considerare che i test sono stati svolti ad una distanza limitata, e che combinata a distanze maggiori, l'inclinazione ha sicuramente un impatto negativo sulla comunicazione. Inoltre, come dimostrano i test successivi, all'aumentare dell'inclinazione, corrisponde un aumento del PLR, il che renderebbe difficoltosa un'intercettazione esaustiva del segnale ad alte inclinazioni.

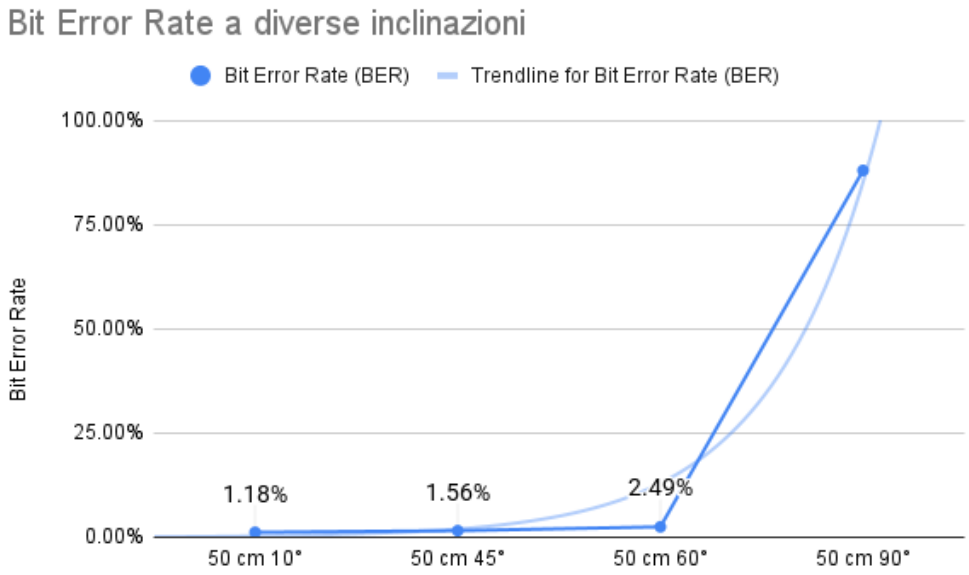


Figura 5.7: Test Bit Error Ratio (BER) in funzione dell'inclinazione

In figura 5.8 si riporta il PRR e la percentuale di *payload* ricevuto in funzione della distanza tra le schede. Si osserva un graduale peggioramento in entrambe le metriche. Si deduce che, all'aumentare della distanza, aumenta sia la perdita di interi pacchetti, sia la perdita di dati all'interno dei pacchetti stessi.

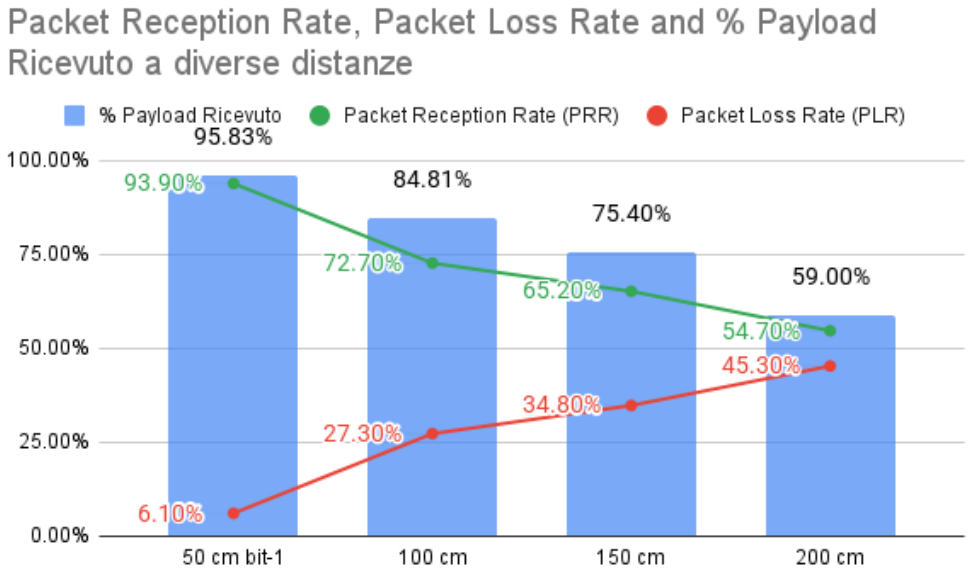


Figura 5.8: Test Packet Reception Rate (PRR) in funzione della distanza

Analogamente, in figura 5.9 si riporta il PRR e la percentuale di *payload* ricevuto

in funzione dell'inclinazione tra le schede. Anche in questo caso si possono fare le medesime considerazioni fatte per i test in funzione della distanza.

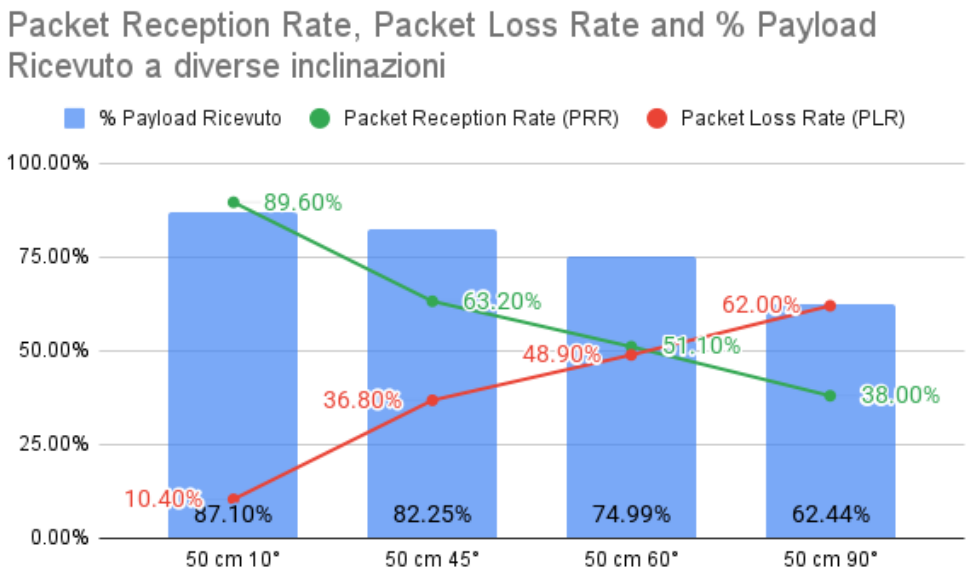


Figura 5.9: Test Packet Reception Rate (PRR) in funzione dell'inclinazione

In figura 5.10 si riporta la percentuale di byte corrotti in funzione della distanza tra le schede. Inoltre si riportano diversi valori di Reed-Solomon, in modo tale da poter individuare, per ogni distanza, la presenza di un valore di Reed-Solomon che permetta di correggere gli errori a quella distanza.

Ogni valore di Reed-Solomon è indicato da RS(n,k), dove n è il numero di byte totali (informativi + codici di correzione) e k è il numero di byte informativi. Ad esempio RS(15,2) indica un codice Reed-Solomon che per ogni 2 byte di informazione trasmessi, aggiunge 13 byte di codici di correzione, per un totale di 15 byte.
Ogni valore di Reed-Solomon è graficamente rappresentato da una *stepped area*, in cui la parte inferiore rappresenta la percentuale di byte che è in grado di correggere. In particolare, sono riportati cinque possibili valori di Reed-Solomon (in ordine di capacità di correzione): RS(15,2), RS(15,4), RS(15,7), RS(15,11) e RS(216,200).
Il valore supportato nativamente da OpenVLC è RS(216,200). Gli altri valori sono stati scelti in quanto sono i più diffusi, nonché quelli indicati dallo standard IEEE 802.15.7 per questo tipo di comunicazione
Naturalmente, ad una maggiore capacità di correzione corrisponderebbe inevitabil-

mente un rallentamento nella comunicazione, in quanto si dovrebbero trasmettere più byte di codici di correzione.

Come si può vedere, la percentuale di byte corrotti rispecchia il BER e, per distanze fino a 100 centimetri, rientra nelle capacità di correzione di RS(216,200). Questo significa che, come si può osservare empiricamente, a quelle distanze c'è una perdita di informazione molto bassa o addirittura nulla.

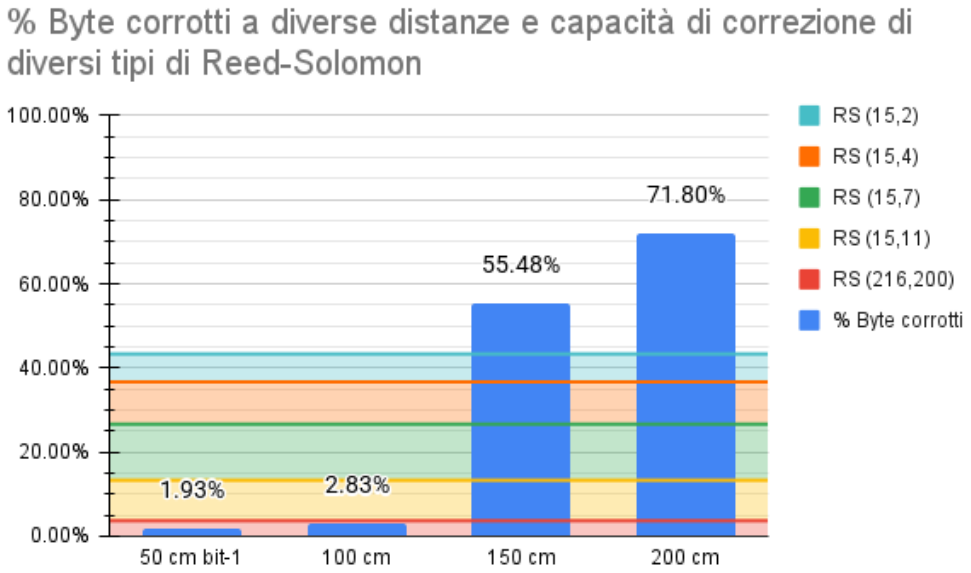


Figura 5.10: Test sulla percentuale di byte corrotti in funzione della distanza

Analogamente, in figura 5.11 si riporta la percentuale di byte corrotti in funzione dell'inclinazione tra le schede. Anche in questo caso si osserva che, fino ad un'inclinazione di 45°, la percentuale di byte corrotti rientra nelle capacità di correzione di RS(216,200). Tuttavia, si può notare che ad una distanza di 60°, per poter garantire la piena correzione degli errori, sarebbe necessario utilizzare un valore di Reed-Solomon più robusto, ad esempio RS(15,11). All'inclinazione di 90°, invece, la percentuale di byte corrotti è tale da non poter essere corretta da alcun valore di Reed-Solomon qui riportato.

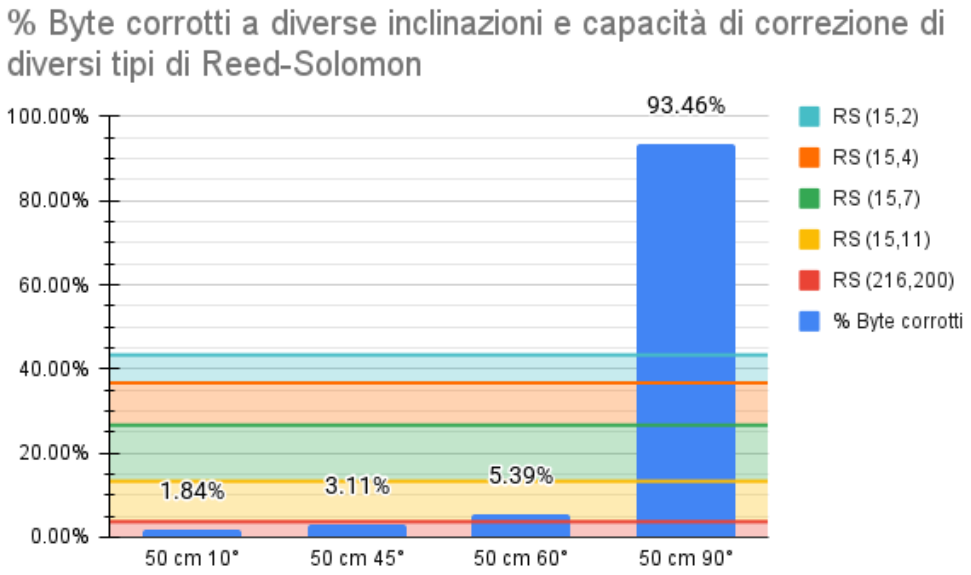


Figura 5.11: Test sulla percentuale di byte corrotti in funzione dell'inclinazione

Capitolo 6

Conclusioni

6.1 Discussione

In questa tesi è stato innanzitutto condotto uno studio accurato sulla piattaforma OpenVLC, sulle sue componenti e sul suo funzionamento, esponendo nel capitolo 3 tutti i dettagli *hardware* e *software* riguardanti questo *framework*. Includendo *testbed*, problematiche riscontrate e come risolverle, in modo tale che il progetto qui discusso sia facilmente riproducibile.
Sono state poi introdotte le principali vulnerabilità di OpenVLC e, per ognuna, delle possibili contromisure adatte a mitigarla.

Successivamente, nel capitolo 4, è stato presentato l'obbiettivo principale di questo progetto, ovvero un modulo di autenticazione dei messaggi a livello fisico per OpenVLC. È stato spiegato dettagliatamente il funzionamento del meccanismo della OTP implementato e motivate le scelte di *design* e progettazione.

Sono infine stati condotti numerosi test, riportati nel capitolo 5, al fine di testare il sistema in diverse condizioni di distanza, inclinazione, luce ambientale e con e senza l'utilizzo del modulo.
I test sul modulo hanno dimostrato che esso non introduce una differenza statisticamente significativa sulle prestazioni del sistema. Dai test in ambiente luminoso risulta infatti una *datagram loss* del 1% non utilizzando il modulo e dello 0.94% utilizzandolo. La *datagram loss* si azzera in ambiente buio e sale invece al 6.2% in ambiente luminoso con disturbo luminoso indotto. Il che dimostra quanto la VLC

sia sensibile all'ambiente in cui viene utilizzata.
Sono stati riportati in seguito diversi test sul *framework* OpenVLC originale. Tali test avevano come obbiettivo l'osservazione di quanto si degrada la qualità della comunicazione in funzione di distanza ed inclinazione. Tutti hanno mostrato, come ipotizzato, un progressivo peggioramento dell'affidabilità della trasmissione all'aumentare di tali variabili; passando, ad esempio, da un BER dell'1.25% ad una distanza di 50 centimetri ad un BER del 66.01% ad una distanza di 200 centimetri. Nonostante tale degradazione, degne di nota sono le prestazioni del sistema a 45 e 60 gradi, le quali dimostrano che la luce visibile non è totalmente unidirezionale, e di conseguenza può essere intercettata anche ad alte angolazioni.

In conclusione, il modulo sviluppato, in fase di test, ha dimostrato di poter garantire maggior sicurezza alle comunicazioni VLC non introducendo differenze significative nelle prestazioni della trasmissione.

6.2 Raggiungimento degli obiettivi

L'obiettivo primario di questo progetto era progettare e implementare un modulo di autenticazione a livello fisico per la piattaforma OpenVLC, in grado di prevenire attacchi di *replay* e *packet injection* nei primi stadi della comunicazione. Come definito nel piano di lavoro, gli obiettivi obbligatori erano:

O01 Configurare le schede BeagleBone Black per consentire la comunicazione tramite luce visibile;

O02 Studiare lo standard IEEE 802.15.7 per comprendere i requisiti tecnici e le specifiche;

O03 Progettare e implementare il modulo di autenticazione sulla piattaforma OpenVLC;

O04 Testare e validare il modulo sviluppato in ambiente buio;

O05 Documentare dettagliatamente il lavoro svolto e i risultati ottenuti.

Tutti questi punti sono stati soddisfatti: le schede sono state configurate con *Debian 8* e *driver OpenVLC_VL_IR* nella versione 1.3, si è approfondito lo standard IEEE 802.15.7, e il modulo `PhyAuthP2P` è stato integrato nel driver OpenVLC e testato in diverse condizioni, confermando che l'introduzione del meccanismo di autenticazione permette di scartare pacchetti illegittimi non degradando le prestazioni.

L'obiettivo desiderabile D01 (Testare e validare il modulo sviluppato in ambiente con luce ambiente) è stato esplorato: i test non hanno evidenziato alcuna differenza rispetto al *framework* OpenVLC originale.

L'obbiettivo facoltativo F02 (Testare il sistema in condizioni più rumorose (con disturbo indotto)) è stato soddisfatto. L'obbiettivo facoltativo F01 (Effettuare il *fine tuning* delle resistenze per ottimizzare la ricezione del segnale) non è stato esplorato.

6.3 Conoscenze acquisite

Nel corso di questo progetto ho avuto l'opportunità di approfondire competenze in:

- Visible Light Communication e piattaforma OpenVLC, incluso l'uso di *hardware* e PRU su BeagleBone Black;
- Standard IEEE 802.15.7, con focus su modulazione e codifica dei segnali;
- Sicurezza a livello fisico e crittografia, mediante studio di *framework* esistenti e l'implementazione di un meccanismo di generazione di OTP;
- Progettazione *software* in C e Linux *kernel-space*;
- Metodologie di test e validazione, calcolo di metriche di valutazione del sistema VLC e analisi di tali metriche.

Queste competenze rappresentano sia un ampliamento delle conoscenze teoriche sia uno sviluppo di abilità pratiche.

6.4 Valutazione personale e sviluppi futuri

L'esperienza di stage e tesi è stata altamente formativa. Le principali difficoltà incontrate sono state la configurazione iniziale dei *cape* OpenVLC e la gestione del sistema operativo *Debian "jessie"* in quanto obsoleto.

La soddisfazione maggiore deriva dall'aver realizzato un prototipo funzionante, con impatto minimo sulle prestazioni e potenziale estendibilità. Come possibili sviluppi futuri di questo progetto si segnalano:

- Integrazione di un *Message Integrity Code* per contrastare attacchi MitM;
- Valutazione di codici di correzione più potenti per ambienti ad alto rumore luminoso;
- Cifratura dei dati al fine di introdurre confidenzialità;
- Estensione del modulo a comunicazioni *multipoint*.

Appendice A

Appendice A

In questo capitolo si descrive la configurazione del sistema OpenVLC all'interno di questo progetto, si elencano alcune problematiche riscontrate durante la configurazione e il modo in cui sono state risolte.

A.1 Configurazione dell'ambiente OpenVLC

Hardware. Come indicato nella documentazione OpenVLC, l'*hardware* necessario comprende unicamente una scheda BeagleBone Black ed un *OpenVLC1.3 RevA cape*. Il primo ostacolo incontrato ha riguardato il corretto funzionamento del *cape*, la cui configurazione iniziale ha richiesto un certo tempo. Durante la fase in cui i tutor si occupavano della risoluzione di tale problematica, ho proseguito parallelamente con lo studio teorico dello standard IEEE 802.15.7 e dell'implementazione OpenVLC. È stato necessario apportare alcune modifiche al *cape*, in particolare collegando manualmente due pin e fornendo un'alimentazione esterna a 12V. Tale intervento si è reso indispensabile a causa della documentazione incompleta fornita dal progetto OpenVLC, che ha richiesto un'attività di *troubleshooting* autonoma per individuare la soluzione.

Software. La versione di OpenVLC che si è scelto di utilizzare è *OpenVLC 1.3*, in particolare *OpenVLC_VL_IR*, in quanto è l'ultima rilasciata ed è compatibile con

l'*hardware* a nostra disposizione.

Configurazione. Per quanto riguarda la configurazione vera e propria, sono state seguite le istruzioni indicate nella *repository*, ufficiale GitHub di Openvlc: *OpenVLC - GitHub*. URL: `https://github.com/openvlc/OpenVLC`. Tutte le modifiche alla configurazione originale sono tracciate nella *repository*, del progetto: *OpenVLC-PA - GitHub* nel file `README.md`.

Per prima cosa si è scritta un'immagine ISO di *Debian 8*, su una scheda micro SD. Questa immagine contiene una versione di *Debian* apposita per funzionare su schede BeagleBone.

In seguito è necessario installare il sistema operativo sulla scheda BeagleBone. Si è inserita la micro SD nell'apposito slot sulla scheda BeagleBone, e una volta connessa al PC tramite cavo USB e avviata una connessione SSH, si può procedere ad eseguire uno script di installazione.

Installato il sistema operativo e riavviata la scheda, è necessario disabilitare l'HDMI in quanto usa alcuni PRU necessari ad OpenVLC, aggiornare il sistema ed installare gli headers.

A questo punto si è incorsi in un errore, in quanto *Debian 8*,, non essendo più supportato, ha archiviato alcune *repository*,. Di conseguenza è stato necessario aggiornarle in modo tale che venissero ricercate nell'archivio.

Si deve poi eseguire lo script *load_test.sh* che compila il *driver*, e permette di cambiare l'indirizzo IP della scheda, è infatti indispensabile che le schede abbiano due indirizzi diversi.

L'ultimo passo è quello di configurare il ruolo della scheda (trasmettitore o ricevitore) selezionando lo script corretto all'interno della cartella 'PRU'. In questo progetto, per il trasmettitore, è stato usato *TX_VL_IR_Dimming_0/deploy.sh* in modo tale da non usare il LED infrarossi, ma solo la luce visibile.

Gli ultimi due passi della configurazione devono essere eseguiti ad ogni avvio della scheda. Per facilitare l'installazione del sistema operativo ed automatizzare l'inizializzazione delle schede sono stati creati alcuni script che si possono trovare nella *repository*, del progetto al percorso `OpenVLC-PA/OpenVLC_VL_IR/utils/`.

Acronimi e abbreviazioni

API Application Program Interface. 22, 23, 46

BER Bit Error Ratio. vi, 30, 32–34, 36, 39, 46

HMAC Hash-based Message Authentication Code. 19, 46

IEEE Institute of Electrical and Electronics Engineers. 5, 12, 39, 40, 46

IoT Internet of Things. 1, 7, 9, 46

IPS Indoor Positioning System. 5, 47

ITS Intelligent Transportation System. 1, 2, 47

MAC Medium Access Control Layer. 6, 9, 11, 47

MIC Message Integrity Code. 17, 47

MitM Man in the Middle. 14, 17, 41, 47

OOK On–off keying. 12, 13, 47

OSI Open Systems Interconnection Model. 6, 9, 47

OTP One Time Password. 6, 16–23, 38, 40, 47

P2P Point-to-point. 18, 22, 48

PHY Physical Layer. 6, 9, 11–13, 17, 48

PLR Packet Loss Rate. 31–33, 48

PRR Packet Reception Rate. vi, 31, 32, 34, 35, 48

PRU Programmable Real-Time Unit. 9, 40, 43, 48

PSK Pre-Shared Key. 16, 19, 22, 48

ubicomp Ubiquitous Computing. 5, 48

VLC Visible Light Communication. 1, 2, 4, 5, 8, 10–14, 38–40, 48

XOR Exclusive or. 21, 22, 49

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 44

BER Il *Bit Error Ratio BER* (ing. tasso di errore sui bit), è il numero di errore sui bit diviso per il numero totale dei bit trasmessi durante un dato intervallo di tempo. 44

HMAC L'*Hash-based Message Authentication Code HMAC*, è una modalità per l'autenticazione di messaggi (message authentication code) basata su una funzione di hash, utilizzata in diverse applicazioni legate alla sicurezza informatica. 44

IEEE L'*Institute of Electrical and Electronics Engineers IEEE* è un'associazione internazionale di scienziati professionisti con l'obiettivo della promozione delle scienze tecnologiche. 44

IoT l'*Internet of Things IoT* (ing. Internet delle cose (IdCa)) è un neologismo utilizzato nel mondo delle telecomunicazioni e dell'informatica che fa riferimento all'estensione di internet al mondo degli oggetti e dei luoghi concreti, che acquisiscono una propria identità digitale in modo da poter comunicare con altri oggetti nella rete stessa. 44

IPS l'*Indoor Positioning System IPS* (ing. Sistema di Posizionamento Indoor) prevede l'uso di dispositivi di rete al fine di localizzare persone o oggetti in luoghi in cui il GPS mancano di precisione o non sono disponibili, come ad esempio all'interno di edifici e strutture sotterranee. 44

ITS con il termine *Intelligent Transportation System ITS* (ing. sistemi di trasporto intelligenti) si intende: l'integrazione delle conoscenze nel campo delle telecomunicazioni, elettronica, informatica - in breve, la "telematica" - con l'ingegneria dei trasporti, per la pianificazione, progettazione, esercizio, manutenzione e gestione dei sistemi di trasporto. 44

MAC Il'*Medium Access Control Layer MAC*, è il livello che controlla l'hardware responsabile dell'interazione con il mezzo di trasmissione. Insieme al logical link control layer (LLC) costituisce il data link layer del modello OSI. 44

MIC In crittografia, un *Message Integrity Code MIC* (ing. codice di integrità del messaggio), è un codice, generato da una funzione di hash, usato per autenticare e verificare l'integrità di un messaggio. 44

MitM Il *Man in the Middle MitM*, è è una terminologia impiegata nella crittografia e nella sicurezza informatica per indicare un attacco informatico in cui qualcuno segretamente ritrasmette o altera la comunicazione tra due parti che credono di comunicare direttamente tra di loro. 44

OOK *On–off keying OOK* denota la più semplice forma di modulazione ASK (amplitude-shift keying), la quale rappresenta i dati digitali come presenza o assenza di una portante. 44

OSI L'*Open Systems Interconnection Model* (ing. Modello OSI), sviluppato dalla International Organization for Standardization (ISO), è lo standard architetturale di riferimento per le reti di calcolatori. 44

OTP Una *One Time Password OTP* (ing. password monouso), è una password valida per una singola sessione o transazione, e spesso solo per un breve periodo di tempo. 44

P2P Nelle telecomunicazioni, una connessione *Point-to-point P2P* (ing. punto a punto), si riferisce ad una comunicazione diretta tra due nodi. Si contrappone alla comunicazione point-to-multipoint o broadcast in cui diversi nodi possono ricevere le informazioni trasmesse da un nodo. 44

PHY Il'*Physical Layer PHY* (ing. Livello fisico), è il primo livello del modello OSI. Si occupa della trasmissione e ricezione dei dati grezzi su un canale di comunicazione fisico. 44

PLR La *Packet Loss Rate PLR* (ing. tasso di perdita dei pacchetti), è definita come la percentuale di pacchetti trasmessi che non vengono correttamente ricevuti dal destinatario. 44

PRR La *Packet Reception Rate PRR* (ing. tasso di ricezione dei pacchetti), è definita come la percentuale di pacchetti correttamente ricevuti rispetto al totale dei pacchetti trasmessi. 45

PRU Le *Programmable Real-time Unit PRU*, sono microcontroller integrati all'interno di alcuni System-on-Chip (SoC), progettati per eseguire compiti a bassa latenza in tempo reale, in parallelo con il processore principale. Ogni PRU è una piccola CPU indipendente, programmabile direttamente, e dotata di accesso diretto alla memoria, a periferiche e a linee di I/O, il che consente di ottenere un controllo preciso sul timing delle operazioni. 45

PSK In crittografia, una *Pre-Shared Key PSK* (ing. chiave pre condivisa), è una chiave segreta che è stata condivisa tra due dispositivi prima dell'uso tramite un canale sicuro. 45

ubicomp l'*Ubiquitous Computing (o "ubicomp")* (ing. Computazione ubiqua) è un paradigma di calcolo in cui i computer sono integrati in oggetti di uso quotidiano e nell'ambiente circostante. 45

VLC nelle telecomunicazioni con il termine *Visible Light Communication VLC* (ing. comunicazione attraverso luce visibile) si indica l'uso di luce visibile (luce con una frequenza di 400-800 THz/wavelength di 780-375 nm) come mezzo di

trasmissione. VLC è un sottoinsieme delle tecnologie wireless di comunicazione ottica. 45

XOR L'*Exclusive or XOR* (ing. o esclusivo), è un operatore logico che produce in uscita VERO (V) se e solo se gli ingressi sono diversi tra di loro. Se gli ingressi sono uguali (V-V oppure F-F) restituisce FALSO (F). 45

Bibliografia

Riferimenti bibliografici

Armstrong, John e Arthur J. Lowery. *Power Efficient Optical OFDM*. Vol. 42. 6. 2006, pp. 370–372. DOI: 10.1049/el:20060075.

Cao, Robert e Jenny J. Zhang. *Handbook of Visible Light Communication: Ten Years of Research*. Cham, Switzerland: Springer, 2019. ISBN: 978-3-030-05022-2.

Chow, Cheng Cong, Chun Qing Chan e Robin D. Murch. *Survey of Visible Light Communication Systems*. 2016, pp. 1–6. DOI: 10.1109/WINCOM.2016.7777839.

Demirors, Erol, Gheorghe Ghinea e David Walker. *Smart Lighting for Smart Cities: Networked Illumination Systems and Services*. Vol. 56. 9. 2018, pp. 64–71. DOI: 10.1109/MCOM.2018.1700710.

Dimitrov, Serge e Harald Haas. *Principles of LED Light Communications: Towards Networked Li-Fi*. Cambridge, UK: Cambridge University Press, 2015. ISBN: 978-1-107-11331-6.

Fonseca, D. F. et al. *Modulating LiFi for dual operation in the visible and infrared spectra*. Vol. 216. 2024, pp. 251–259. DOI: 10.1016/j.comcom.2024.01.005.

Fonseca, D. F. et al. *Visible light or infrared? Modulating LiFi for dual operation in the visible and infrared spectra*. Madonna di Campiglio, Italy, 2023, pp. 47–50. DOI: 10.23919/WONS57325.2023.10061923.

Ghassemlooy, Zabih et al. *Visible Light Communications: Theory and Applications*. Boca Raton, FL: CRC Press, 2017. ISBN: 978-1-4987-2906-0.

Haas, Harald. *Li-Fi: Visible Light Communication, 2nd Edition*. Cambridge, UK: Cambridge University Press, 2018. ISBN: 978-1-108-44108-5.

Hranilovic, Steve. *Wireless Optical Communication Systems*. Vol. 28. Telecommunication Networks and Series. 2007. ISBN: 978-0-387-34571-7.

IEEE Standard for Local and metropolitan area networks–Part 15.7: Short-Range Optical Wireless Communications. 2018.

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Kahn, Joseph M. e John R. Barry. *Wireless Infrared Communications*. Vol. 85. 2. 1997, pp. 265–298. DOI: 10.1109/5.554262.

Khalid, Adeel, Dermot O'Brien e Jan L.A. Dubbeldam. *A Hybrid RF/Optical Wireless System for Vehicular Networks*. Vol. 24. 5. 2017, pp. 100–106. DOI: 10.1109/MWC.2017.1600217WC.

Komine, Takashi e Masao Nakagawa. *Fundamental Analysis for Visible-Light Communication System using LED Lights*. 2004, pp. 457–461. DOI: 10.1109/ICC.2004.1313179.

Li, Ang et al. *PhyAuth: Physical-Layer Message Authentication for ZigBee Networks*. Anaheim, CA: USENIX Association, ago. 2023, pp. 1–18. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/li-ang.

Liu, Cheng et al. *Underwater Visible Light Communication Using Blue LEDs*. Vol. 32. 10. 2020, pp. 605–608. DOI: 10.1109/LPT.2020.2982123.

Rajagopal, Srinivasan, Kirk Roberts e Shree K. Lim. *IEEE 802.15.7 Visible Light Communication: Modulation Schemes and Dimming Support*. 2012, pp. 1685–1690. DOI: 10.1109/GLOCOMW.2012.6477809.

Rehman, S. U. et al. *Visible Light Communication: A System Perspective-Overview and Challenges*. Vol. 19. 5. Mar. 2019, p. 1153. DOI: 10.3390/s19051153. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6427442/.

Tsonev, Dobroslav, Harold Chun e Harald Haas. *A 3-Gbps Single-LED OFDM-Based Wireless Visible Light Communication Link*. Vol. 25. 2. 2013, pp. 171–174. DOI: 10.1109/LPT.2012.2235128.

Tsonev, Dobroslav, Stefan Videv e Harald Haas. *Towards a 100 Gb/s Visible Light Wireless Access Network*. 2015, Th1I.2.

Wu, Xuan et al. *High-Speed Visible Light Communications Based on Off-the-Shelf LEDs*. Vol. 7. 6. 2015, pp. 1–14. DOI: 10.1109/JPHOT.2015.2483043.

Zhang, Rui, Lei Sun e Shina Fu. *Recent Advances in Visible Light Communication and Its Applications*. A cura di Harald Haas, Hakan Bulow e Giulio Cossu. Wiley, 2019, pp. 201–228. ISBN: 978-1-118-80091-8.

Siti web consultati

BeagleBone. URL: https://www.beaglebone.org/.

Bit Error Rate - Wikipedia. URL: https://en.wikipedia.org/wiki/Bit_error_rate.

Codifica Manchester - Wikipedia. URL: https://it.wikipedia.org/wiki/Codifica_Manchester.

HMAC - Wikipedia. URL: https://it.wikipedia.org/wiki/HMAC.

Indoor Positioning System - Wikipedia. URL: https://en.wikipedia.org/wiki/Indoor_positioning_system.

Institute of Electrical and Electronics Engineers - Wikipedia. URL: https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers.

Intelligent transportation system - Wikipedia. URL: https://it.wikipedia.org/wiki/Intelligent_transportation_system.

Internet of Things - Wikipedia. URL: https://it.wikipedia.org/wiki/Internet_delle_cose.

Man in the Middle - Wikipedia. URL: https://it.wikipedia.org/wiki/Attacco_man_in_the_middle.

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/.

On-off Keying - Wikipedia. URL: https://it.wikipedia.org/wiki/On-off_keying.

One Time Password - Wikipedia. URL: https://en.wikipedia.org/wiki/One-time_password.

OpenVLC. URL: http://www.openvlc.org/ (cit. a p. 2).

OpenVLC - GitHub. URL: https://github.com/openvlc/OpenVLC (cit. a p. 43).

OpenVLC-PA - GitHub. URL: https://github.com/francesco-lapenna/OpenVLC-PA (cit. alle pp. 24, 43).

OSI Model - Wikipedia. URL: https://en.wikipedia.org/wiki/OSI_model#Layer_2:_Data_link_layer.

Point-to-point - Wikipedia. URL: https://en.wikipedia.org/wiki/Point-to-point_(telecommunications).

Practical Networking - YouTube. URL: https://www.youtube.com/@PracticalNetworking.

Pre-Shared Key - Wikipedia. URL: https://en.wikipedia.org/wiki/Pre-shared_key.

Ubiquitous computing - Wikipedia. URL: https://en.wikipedia.org/wiki/Ubiquitous_computing.

Visible light communication - Wikipedia. URL: https://en.wikipedia.org/wiki/Visible_light_communication.

XOR - Wikipedia. URL: https://it.wikipedia.org/wiki/Disgiunzione_esclusiva.