Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Mappe di canale radio per reti cellulari

Relatore:
Prof. Stefano Tomasin

Laureando:
Riccardo Trevisiol
matricola: 1216353

Anno Accademico 2021/2022

Abstract

Per il funzionamento della rete 5G i dispositivi si collegano a una stazione base, e il canale di comunicazione è quantizzato mediante un codebook. Questa versione quantizzata è trasmessa dal dispositivo alla stazione base e viene trasmesso in chiaro: un utente esterno alla rete può intercettare il segnale inviato da un terminale, e attraverso esso localizzarlo, sapendo quindi la posizione e gli spostamenti.
Questa tesi si propone di individuare un modo per ricostruire la mappa che associa ai canali quantizzati la posizione del dispositivo, partendo da delle sequenze di canali quantizzati intercettati. Innanzitutto, i segnali intercettati vengono utilizzati per creare una matrice di adiacenza, ovvero una matrice nella quale per ogni elemento ne vengono salvati quelli che risultano vicini. Quest'ultima poi viene usata per ricreare una mappa con un algoritmo che si muove a spirale, andando a interrogare la matrice di adiacenza per poter collocare gli elementi nel giusto ordine.

Indice

Introduzione **1**

1 Stima della posizione da segnali di controllo **3**

1.1 Introduzione al problema e prima analisi dell'attacco 3

1.1.1 Descrizione dell'attacco 5

1.2 Modello di Canale 5

1.3 Feedback sulle Informazioni del Precodificatore 9

1.4 Attacco di localizzazione dell'UE e mitigazione 11

1.5 Analisi delle prestazioni MSE 14

1.6 Conclusioni 18

2 Ricostruzione automatica della mappa **19**

2.1 Versione senza ripetizioni 20

2.2 Funzioni ausiliari 22

2.3 Spiegazione dell'algoritmo con codici univoci 23

2.4 Versione con ripetizioni 26

2.5 Spiegazione dell'algoritmo con ripetizioni 28

2.6 Risultati 31

Conclusioni **33**

Bibliografia **35**

Ringraziamenti **37**

Introduzione

Per il funzionamento della rete 5G i dispositivi si collegano a una stazione base, e il canale di comunicazione è quantizzato mediante un codebook. Questa versione quantizzata è trasmessa dal dispositivo alla stazione base e viene trasmesso in chiaro: un utente esterno alla rete può intercettare il segnale inviato da un terminale, e attraverso esso localizzarlo, sapendo quindi la posizione e gli spostamenti. Questa tesi si propone di individuare un modo per ricostruire la mappa che associa ai canali quantizzati la posizione del dispositivo, partendo da delle sequenze di canali quantizzati intercettati. Innanzitutto, i segnali intercettati vengono utilizzati per creare una matrice di adiacenza, ovvero una matrice nella quale per ogni elemento ne vengono salvati quelli che risultano vicini. Quest'ultima poi viene usata per ricreare una mappa con un algoritmo che si muove a spirale, andando a interrogare la matrice di adiacenza per poter collocare gli elementi nel giusto ordine. Per affrontare il problema, sono state assunte alcune semplificazioni: la prima è che per produrre la matrice di adiacenza è stata usata una mappa conosciuta a priori, supponendo che le strisce fossero già state raccolte. La seconda invece riguarda l'assunzione che le sequenze di codici fossero esaurienti per la completezza della mappa.

Nel primo capitolo della tesi viene tradotto l'articolo *Localization Attack by Precoder Feedback Overhearing in 5G Networks and Countermeasures* [1], nel quale viene individuato il problema, descritta la procedura di un attacco e proposte delle soluzioni di mitigazione. Uno dei punti focali di questo attacco è la ricostruzione di una mappa a partire dalle sequenze di codice, argomento che viene affrontato nel capitolo successivo.

Nel secondo capitolo della tesi viene affrontata la ricostruzione della mappa a partire dalle sequenze intercettate, supponendo che queste siano già state raccolte. In questo capitolo vengono considerati sia il caso in cui una mappa abbia solo sequenze distinte per ogni posizione, senza ripetizioni, che il problema più generale, dove in una mappa ci possono essere delle sequenze che si ripetono in

posizioni adiacenti.

Capitolo 1

Stima della posizione da segnali di controllo

Nelle reti cellulari di quinta generazione (5G), gli utenti inviano alla stazione base l'indice di un *precoder* scelto da un *codebook* da utilizzare per la trasmissione in *downlink* con un segnale di $feedback$ di livello 2. Il *precoder* è associato al canale dell'utente e alla posizione dello stesso all'interno della cella. In [1] viene descritto un metodo tramite il quale un attaccante è in grado di determinare la posizione di un utente captando passivamente il segnale di $feedback$ di livello 2 non criptato. L'attaccante deve prima ricostruire una mappa degli indici del *precoder* delle celle, per poi, una volta captato l'indice del *precoder*, individuare la posizione della vittima sulla mappa.

1.1 Introduzione al problema e prima analisi dell'attacco

Gli interessi politici ed economici presenti nella società fanno pressione per lo sviluppo di nuove tecnologie per la localizzazione degli utenti [2], a vantaggio di aziende e Stati, poiché molti servizi sfruttano la conoscenza della nostra posizione. Nei sistemi 5G è stata infatti posta molta enfasi sulla questione della localizzazione. Un esempio di questa sfrutta il fatto che la rete cellulare può ricevere i segnali radio trasmessi dall'utente attraverso più stazioni base, alle volte dotate di molteplici antenne [3]. Utilizzando questa tecnica, si è rilevata una riduzione dell'errore di localizzazione [4]. Pertanto, il passaggio dalle microonde nelle reti di quarta generazione (4G) alle $mmWaves$ nelle reti 5G, e successivamente alle

onde submillimetriche nelle reti di sesta generazione (6G), apre la strada ad una precisione di localizzazione nell'ordine dei centimetri.
Purtroppo anche i dispositivi ostili, cioè esterni alla rete, possono essere interessati a localizzare utenti specifici, costituendo un importante problema per la loro *privacy*. Questo è un argomento chiave ai nostri giorni, con impatto sia sociale che personale [5]. La *privacy* della posizione è uno dei problemi di sicurezza critici, in particolare per le nuove reti 5G [6]. Una soluzione per localizzare l'utente è quella di sfruttare nuovamente il segnale radio da lui utilizzato. Nella letteratura sono disponibili vari metodi di localizzazione basati sulle informazioni dello stato del canale. A partire dalle informazioni sullo stato del canale, è possibile stimare il tempo di arrivo (ToA) o l'angolo di arrivo (AoA) e successivamente applicare la trilaterazione o la triangolazione [7]. Un altro approccio è anche chiamato *Radio Fingerprint Pattern Matching* (RFPM) [8], [9]; questo prevede che l'indicatore della potenza del segnale ricevuto (in inglese RSSI) [10], [11], e la risposta all'impulso del canale multipercorso [12] possano essere usati in alternativa a questo scopo. In un approccio alternativo, l'attaccante intercetta il segnale demodulato proveniente dall'utente e dalle stazioni base legittime. Questo tipo di attacco potrebbe essere più facile da implementare rispetto a quello basato sulla stima del canale, poiché richiede solo la demodulazione e la decodifica dei segnali digitali. In particolare, non richiede più antenne di ricezione (necessarie per la localizzazione in base ai parametri del canale) e può utilizzare *chipset* standard. La sua minore complessità lo rende più pericoloso per la *privacy* della posizione dell'utente.

Tutti gli attacchi in letteratura che si concentrano su soluzioni passive forniscono solo le informazioni sulla cella in cui si trova l'utente, fornendo una localizzazione grossolana. Questi attacchi sono anche indicati come attacchi di identificatori di celle (cell id, CID). Nell'articolo già citato, [1], viene proposto un nuovo attacco di localizzazione, basato sull'intercettazione del segnale di controllo trasmesso dagli utenti e relativo alle informazioni sullo stato del loro canale, come previsto dalle specifiche dello standard *New Radio* (NR) 3GPP delle reti 5G.
In particolare, 3GPP fornisce all'utente i mezzi per stimare il canale di *downlink* e quindi permette di scegliere il miglior *precoder* da un *codebook* predefinito; infine, l'utente restituisce alla stazione base l'indice del *precoder* selezionato. Il *precoder*, infatti, è legato al canale sperimentato dall'utente, e quindi rivela, in parte, la sua posizione.

1.1.1 Descrizione dell'attacco

L'attaccante (probabilmente supportato da utenti collusi) costruisce prima una mappa degli indici del *precoder* restituiti da varie posizioni nella cella. Quindi, ascoltando l'indice del *precoder* restituito dall'utente vittima, l'attaccante trova la sua posizione sulla mappa. Ci concentriamo sul *codebook* di tipo I a pannello singolo, che è l'unica soluzione obbligatoria nello standard, ad oggi. Quando si opera a $mmWaves$, sia l'utente che la stazione base devono essere dotati di più antenne ed è necessario un preciso $beamforming$ per le comunicazioni di dati; quindi, verrà restituita una descrizione più precisa del *precoder*. Questo, a sua volta, fornisce una migliore localizzazione dell'utente, simile a quanto accade nelle tecniche di localizzazione radio.

L'attacco verrà analizzato valutando l'accuratezza della localizzazione ottenuta rispetto a vari parametri, come ad esempio la modalità di $feedback$, il numero di sottobande e la dimensione della cella, e tramite l'analisi dell'errore di localizzazione asintotica di un modello di $feedback$ del *precoder* semplificato, poiché il numero di sottobande e *cluster* di canali diventa grande. Viene proposta anche una strategia di mitigazione contro l'attacco proposto, in cui l'utente seleziona casualmente il *precoder* tra quelli che forniscono la velocità di trasmissione più alta. Varie simulazioni su modelli di canale di riferimento confermano che l'attacco può raggiungere un'elevata accuratezza di localizzazione, che viene invece notevolmente ridotta quando viene adottata la soluzione di mitigazione, a scapito di una riduzione di velocità di trasmissione trascurabile. Inoltre, viene confrontata la tecnica con le tre soluzioni sopra menzionate disponibili in letteratura: gli attacchi CID, ToA e RFPM. Il metodo proposto ha una precisione di localizzazione significativamente migliore rispetto all'attacco CID, che aumenta con l'aumentare del numero di sottobande, superando ToA nella maggior parte dei casi, ma richiedendo anche $hardware$ e $software$ più semplici. Al contrario, la sua precisione è inferiore a quella di RFPM, che richiede però $hardware$ e $software$ più complessi.

1.2 Modello di Canale

Si consideri una cella di raggio R, come mostrato in 1.1. La base del nodo di nuova generazione (gNB) si trova al centro del cerchio, che coincide con l'origine delle coordinate (0,0), mentre la posizione dell' apparecchiatura della vittima (UE) ha coordinate $p = (p_x, p_y)$. Il gNB è dotato di un $array$ lineare di N

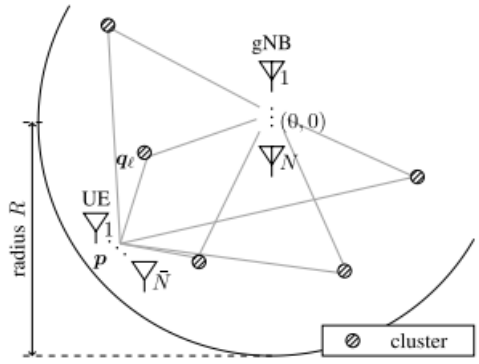

Figura 1.1: Descrizione schematica del modello di canale per una cella circolare di raggio R, e un generico *cluster* di coordinate q.

antenne a polarizzazione incrociata ($2N$ antenne in totale), mentre l'UE è dotato di un *array* lineare di $\bar{N}$ antenne. Secondo la versione 16 dello standard di rete cellulare 3GPP, la modulazione a divisione di frequenza ortogonale (OFDM) viene utilizzata nel *downlink* e le sottoportanti sono raggruppate in sottobande. Si assuma che K sottobande siano assegnate alla vittima in posizione p e che il canale sia lo stesso per tutte le sottoportanti all'interno della stessa sottobanda. Sia $H(p,k)$ il canale di *downlink* sulla sottobanda k per una UE in posizione p. Il gNB fornisce un flusso all'UE in $mmWave$, e trasmette un simbolo modulato digitalmente su $s_\kappa(k)$ su κ sottoportanti della sottobanda k, con una colonna di dimensione N precodificata dal vettore $w(k)$ in modo che il segnale ricevuto dalla UE sia

$$r_\kappa(k) = H(p,k)w(k)s_\kappa(k) + w_\kappa(k), \tag{1.1}$$

dove $k = 1, ..., K$, e $v(k)$ rappresenta l'$AdditiveWhiteGaussianNoise(AWGN)$, un vettore colonna di dimensione $\bar{N}$ a media zero e varianza σ_v^2.

In [1], per lo sviluppo del canale, non viene adottato il modello definito dallo standard 3GPP, poiché in ciò le realizzazioni del canale non dipendono dalla posizione della UE. In realtà però l'angolo di partenza dal gNB e quello di arrivo all'UE di ciascun percorso di canale sono correlati la posizione del gNB, dell'UE e dei riflettori che definiscono il percorso. Poiché queste informazioni vengono sfruttate per localizzare l'UE, si necessita di un modello di canale coerente con lo spazio e con la posizione, descritti di seguito:

- **Modello di canale coerente con la posizione**
  Prima di procedere, si ricorda che tutte le sottoportanti della stessa sotto-

banda sperimentano lo stesso canale. Il canale tra una UE e il suo servizio di gNB è determinato dalle riflessioni su un *cluster* L. La posizione del *cluster* $\ell \in \mathbb{L} = \{1, ..., L\}$ ha coordinate $q_\ell = (q_{x,\ell}, q_{y,\ell})$, come mostrato in 1.1 il ritardo di trasmissione del percorso attraverso il *cluster* è

$$\tau_l(p) = \frac{1}{c_0}(||p - q_l||_2 + ||q_l||), \tag{1.2}$$

dove c_0 è la velocità della luce. Da questa si calcola il guadagno complessivo del canale per tutte le sopportanti $k \in K = \{1, ..., K\}$, relativo al cluster ℓ è

$$\gamma_l(p, k) = \sqrt{N}\frac{A_P L * g_l(k)}{\tau_l(p)}, \tag{1.3}$$

dove A_{PL} è il fattore di normalizzazione della perdita di percorso, T_S il periodo di campionamento e $g(k)$ è una gaussiana complessa a media nulla e a varianza unitaria, indipendente per ogni sottobanda di ogni *cluster*. Si assuma inoltre un guadagno invariante nel tempo. L'angolo tra il *cluster* ℓ e il gNB è

$$\phi'(q_\ell) = \arctan 2(q_{y,\ell}, q_{x,\ell}) \tag{1.4}$$

mentre l'angolo tra il *cluster* ℓ e l'UE è

$$\bar{\phi}'(q_\ell) = \arctan 2(q_{y,\ell} - p_y, q_{x,\ell} - p_x). \tag{1.5}$$

Per la sottobanda k, l'angolo di partenza dal gNB è

$$\phi(p, q_\ell, k) = \phi'(q_\ell) + c_{ASD}\alpha_\ell(p, k), \tag{1.6}$$

mentre l'angolo di arrivo è

$$\bar{\phi}(p, q_\ell, k) = \bar{\phi}'(q_\ell) + c_{ASD}\bar{\alpha}_\ell(p, k) + v_O, \tag{1.7}$$

dove $\alpha_\ell(p, k) \sim U([-1, 1])$ e $\bar{\alpha}_\ell(p, k) \sim U([-1, 1])$ sono spazialmente correlati e indipendenti per ogni sottobanda e c_{ASD} e c_{ASA} sono costanti; infine, $v_O \sim U([0, 2\pi))$ rappresenta l'orientamento del dispositivo nel piano $x - y$. Per un segnale in partenza da gNB nella direzione del *cluster* ℓ alla sottobanda k, si definisce il vettore di guida, che rappresenta lo sfasamento

relativo tra le antenne secondo

$$a_\ell(p,k) = \frac{1}{N}\begin{pmatrix} e^{j\frac{2\pi d}{\lambda_k}(-\frac{N-1}{2})sin\phi(p,q_\ell,k)} \\ \vdots \\ e^{j\frac{2\pi d}{\lambda:\kappa}(\frac{N-1}{2})sin\phi(p,q_\ell,k)} \end{pmatrix}, \tag{1.8}$$

dove d è la distanza dell'antenna dal gNB,

$$\lambda_k = \frac{c_0}{f_c + \delta_f(k-(K-1)/2} \tag{1.9}$$

è la lunghezza d'onda del della sottobanda k, f_c è la frequenza di trasporto centrale e δ_f è il passo delle sottobande. Similarmente, per il segnale ricevuto dall'UE,

$$\bar{a}_\ell(p,k) = \frac{1}{N}\begin{pmatrix} e^{j\frac{2\pi \bar{d}}{\lambda_k}(-\frac{N-1}{2})sin\bar{\phi}(p,q_\ell,k)} \\ \vdots \\ e^{j\frac{2\pi \bar{d}}{\lambda:\kappa}(\frac{N-1}{2})sin\bar{\phi}(p,q_\ell,k)} \end{pmatrix}, \tag{1.10}$$

dove $\bar{d}$ è la distanza tra le antenne. Infine, il canale a banda stretta alla sottobanda k è modellato come una matrice $\bar{N} \times 2N$

$$H(p,k) = \sum_{\ell \in L} \gamma_\ell(p,k)\bar{\alpha}_\ell(p,k)\left[\begin{pmatrix}\sin(\mu)\\ \cos(\mu)\end{pmatrix} \otimes \alpha_\ell(p,k)\right], \tag{1.11}$$

dove μ è l'angolo di cofasatura legato all'UE rispetto al gNB.
Questo modello è coerente con la posizione, poiché qualsiasi UE in posizione p sperimenterà delle riflessioni da dei *cluster*, avendo così gli stessi $\gamma_\ell(p,k)$ e $a_\ell(p,k)$. Solo l'angolo di cofasatura μ è indipendente per ogni UE nella stessa posizione, siccome l'orientamento di ogni dispositivo non dipende da questa.

- **Modello di canale coerente nello spazio**
  I canali di UE in diverse posizioni sono spazialmente correlati, come confermato da una vasta letteratura. Pertanto, tutti i canali dell'UE sono determinati dagli stessi *cluster* mentre altri parametri variano nello spazio. Per ottenere un modello spazialmente consistente, consideriamo $\alpha_l(p,k)$

spazialmente correlato in diverse posizioni p, quindi

$$E[\alpha(p_1,k),\alpha(p_2,k)] = e^{(-\frac{||p_1-p_2||_2}{2d_S})}, \tag{1.12}$$

dove d_S è la distanza di correlazione spaziale. $\alpha_\ell(p,k)$ sono spazialmente correlati in p, e sono indipendenti in ℓ e k.

1.3 Feedback sulle Informazioni del Precodificatore

Il *precoder* $w(k)$ da utilizzare sulla sottobanda k è determinato dalla UE, in base alle sue condizioni di canale. A tale scopo, il gNB trasmette periodicamente un segnale pilota alla UE. Da questo, la UE ottiene la stima del canale alla sottobanda k, modellato come

$$\hat{H}(p,k) = H(p,k) + V, \tag{1.13}$$

dove V rappresenta una matrice con i vettori di errore AWG, con elementi gaussiani complessi $zero-mean$ e indipendenti, quindi $(V)_{\bar{n},n} \sim CN(0,\sigma^2), \bar{n} = 1,...,\bar{N}, n = 1,...,2N$. Basandosi su $\hat{H}(p,k)$, l'UE seleziona il *precoder* $w(k)$ da un *codebook* condiviso col gNB. Assumendo degli schemi di modulazione e codifica perfetti e che l'UE miri a massimizzare il *throughput* di *downlink* ottenuto, l'efficienza spettrale di ciascuna sottoportante della sottobanda k può essere stimata come

$$R[w(k)] = \sum_{k=1}^{K} \log_2 * \Big\{ \det\Big[I_{\bar{N}} + \frac{\hat{H}(p,k)w(k)w(k)^H\hat{H}(p,k)^H}{\sigma_v^2} \Big] \Big\}. \tag{1.14}$$

Questa è una stima dell'efficienza spettrale: per calcolare quella effettiva, $\hat{H}(p,k)$ è sostituito da $H(p,k)$. Una volta che l'UE ha selezionato i vettori di precodifica $w(k)$, trasmette al gNB una sequenza di bit, identificandoli all'interno del *codebook*. Lo standard 3GPP definisce un grande *codebook* e quindi specifica due procedure per la selezione del vettore e il $feedback$ dei loro indici. Queste due procedure prendono il nome di modalità di $feedback$.

- **Costruzione del codebook**
  Nella versione 16 dello standard 3GPP sono definiti vari tipi di *codebook*. Poiché l'unico tipo di *codebook* che le UE saranno obbligate a rispettare è il *codebook* di tipo I, l'articolo si ferma all'analisi di quello. Ciascun vettore

di precodifica è identificato dalla coppia (m, n), dove $m \in M$ e $n(k) \in N$ e O è il fattore di sovracampionamento. L'indice m identifica il vettore

$$\tilde{w}_m = (1, \mathrm{e}^{(\frac{j2\pi m}{NO})}, ..., \mathrm{e}^{(\frac{j2\pi(N-1)m}{NO})}). \tag{1.15}$$

L'indice n è invece associato all'angolo di cofasamento tra le polarizzazioni delle coppie di antenne $\psi_n = \mathrm{e}^{(j2\pi n/4)}$. Successivamente, il vettore di precodifica associato alla coppia (m, n) è definito come:

$$w_{m,n}c = \frac{1}{\sqrt{N}} \begin{pmatrix} \tilde{w}_m \\ \psi_m \tilde{w}_m \end{pmatrix}. \tag{1.16}$$

La costruzione delle tuple (m, n) varia tra le differenti modalità di $feedback$.

- **Modalità di feedback 1**
  Nella modalità di $feedback$ 1, viene utilizzato lo stesso vettore di precodifica per tutte le sottobande, mentre viene applicata una cofasatura dipendente dalla sottobanda delle due polarizzazioni. L'UE seleziona il *precoder* come soluzione del seguente problema di massimizzazione della velocità:

$$(w^*, \{n^*(k)\}) = arg \max R_{m,n(k)}[\{w(k) = w_{m,n(k)}\}], \tag{1.17}$$

  dove $m \in M$ e $n(k) \in N$, per ogni sottobanda di k. Il $feedback$ del gNB comprende due indici, che per la prima modalità sono definiti come segue:

  - $i_{1,1}$ è la rappresentazione intera senza segno $\log 2(NO)$-bit di $m^* \in M$.
  - $i_2(k)$, $k = 1, ..., K$, è un pacchetto a 2 bit $n^*(k)$. Questo richiede in tutto 2000 bit.

- **Modalità di feedback 2**
  La scelta del vettore di precodifica e cofasatura dipende dalla sottobanda. La cofasatura viene scelta come per la modalità precedente, e l'UE sceglie il *precoder* tramite il problema di massimizzazione del *troughput*

$$(m^*, \{\delta^*(k)\}, \{n^*(k)\}) = argmax_{m,\{\delta(k)\},\{n(k)\}} R \times [\{w(k) = w_{2m+\delta(k),n(k)}\}], \tag{1.18}$$

  dove $m \in \{0, ..., (NO)/2 - 1\}$ e $\delta(k) \in \{0, ...3\}$ per ogni sottobanda.
  Per quanto riguarda il feedback del gNB abbiamo:

  - $i_{1,1}$ è la rappresentazione intera senza segno $\log 2(NO)$-bit di $m^* \in M$.
  - $i_2(k)$, $k = 1, ..., K$ è un insieme di 4 bit, di cui 2 più significativi e due meno, che trasportano il valore intero e senza segno di $\delta^*(k)$ e $n^*(k)$. Questo richiede un totale di 4000 bit, il doppio del caso precedente.

- **Modalità di feedback 3**
  Nella terza modalità di $feedback$, si assegna ad ogni sottobanda un vettore di precodifica differente, quindi con entrambi i valori di $m(k)$ e $n(k)$ differenti. La coppia di indici per ogni sottobanda viene trovata grazie a

$$(m^*(k), \{n^*(k)\}) = argmax_{\{m(k)\},\{n(k)\}} R \times [\{w(k) = w_{m(k),n(k)}\}]. \quad (1.19)$$

  Per quanto riguarda il feedback del gNB abbiamo:

  - $i_2(k)$, $k = 1, ..., K$ è un intero senza segno, del quale $\log_2(NO)$bit rappresentano $m(k)$, e i due bit meno significativi $n(k)$. Ciò richiede un numero complessivo di $K(\log_2(NO) + 2$ bit.

1.4 Attacco di localizzazione dell'UE e mitigazione

Nella seguente sezione, si vogliono descrivere le varie parti che compongono l'attacco per poi descrivere una strategia di mitigazione per l'attacco.

- **Modello dell'attaccante**
  L'attaccante è dotato di un dispositivo in grado di captare lo scambio di segnali tra il gNB e l'UE. Si assuma che l'attaccante intercetti il $feedback$ del *precoder* della UE, quindi $i_{1,1}$ e $\{i_2(k)\}$. Come già detto in precedenza, le UE nella stessa posizione sperimentano polarizzazioni e rumore differenti, producendo valori di $i_{1,1}$ e $\{i_2(k)\}$ diversi. Dato che l'attaccante è interessato solo alla posizione della UE, trascurerà le informazioni riguardanti la polarizzazione, ovvero $\{i_2(k)\}$ nella modalità di $feedback$ 1 e nelle modalità 2 e 3 i due bit meno significativi di $\{i_2(k)\}$. Da qui in avanti, $b(p) = [b_1, ..., b_K]$ indicherà il vettore di bit utile alla localizzazione restituito in posizione p per ogni sottobanda k, e B indicherà l'insieme dei possibili valori di b_k. Il processo di quantizzazione è ancora influenzato dal rumore, producendo diversi vettori di $feedback$ $b(p)$ dalla stessa posizione p.

- **Attacco di localizzazione proposto**
  L'attacco prevede due fasi, la *fase preliminare* e la *fase di attacco*. Durante

la prima, l'attaccante trae vantaggio dalla collaborazione di UE complici che si muovono all'interno della cella e raccolgono i bit di $feedback$ del $precoder$. Sia S l'insieme delle posizioni esplorate durante questa fase. Si definisca la probabilità di occorrenza del vettore $b(s)$ come

$$p_{map}(\beta, s) = P[b(s) = \beta]. \tag{1.20}$$

Questa probabilità fornisce una mappa probabilistica dei vettori di $feedback$ alle posizioni nella cella e sarà sfruttata dall'attaccante per localizzare l'UE vittima nella fase successiva. Viene definita anche la probabilità di occorrenza del vettore β come

$$p(\beta) = E[P[b(p) = \beta]], \tag{1.21}$$

dove il valore atteso è assunto rispetto alla posizione p. Nella fase di attacco, l'attaccante intercetta il vettore di $feedback$ b^* inviato dalla UE da una posizione sconosciuta. Quindi, l'attaccante trova la posizione dell'UE calcolando la media di tutti i punti aventi come $feedback$ il bit b^*, cioè,

$$\hat{p}(b^*) = (\hat{p}_x, \hat{p}_y) = \sum_{s \in S} \frac{p_{map}(b^*, s)}{p(b^*)} s. \tag{1.22}$$

Una stima della probabilità $p_{map}(\beta, s)$ e $p(\beta)$ può essere ottenuta dai dati raccolti durante la fase preliminare.

- **Potenziale implementazione dell'attacco proposto**
  Si rendono necessari dei commenti sull'attacco. Questo è infatti diverso dalle soluzioni di localizzazione basate su una trilaterazione. L'elaborazione del segnale trascurabile è necessaria o eseguita dall'attaccante sul vettore di bit restituito, poiché è richiesta solo una semplice mappatura. Per fare ciò, l'attaccante sarà dotato di una sola antenna, mentre per una triangolazione accurata sono necessarie più antenne. Per l'attacco descritto, possono essere utilizzati dei *chipset* per la decodifica dei segnali di controllo senza un $hardware$ dedicato; la parte $software$ si rivela essere anch'essa elementare, essendo solo un *database* di posizioni e di vettori di $feedback$. Per quanto riguarda la complessità computazionale implicata nella costruzione della mappa, quando i bit di $feedback$ sono raccolti da UE colluse in una posizione nota, è richiesta una minore elaborazione dei dati. Tuttavia, esplorare la zona nella sua totalità richiede tempo e la disponibilità di utenti collusi.

La precisione di localizzazione è dettata anche dalla densità di punti della mappa per i quali è disponibile una corrispondenza tra la posizione e i bit di $feedback$. Inoltre, la mappa dovrebbe essere aggiornata ogni volta che cambia l'ambiente di propagazione *wireless*; in alternativa ad un metodo così dispendioso, l'attaccante può infettare le UE vicine con un'app contente un virus in grado di inviare la posizione della UE all'attaccante, così che questo possa associarla ai bit di $feedback$. Un altro approccio può includere l'uso di tecniche di *machine learning*, come mappe auto-organizzate, per la costruzione automatica della mappa anche senza conoscere la posizione dell'utente. La corrispondenza tra posizioni e bit di $feedback$ cambia nel tempo a causa del movimento degli oggetti; pertanto, la mappa dovrebbe essere periodicamente aggiornata. Entrambe le soluzioni, iniezione di virus e l'uso di mappe auto-organizzate, possono essere eseguiti continuamente, fornendo un costante aggiornamento della mappa.

- **Soluzioni di mitigazione**
  Per tentare di mitigare l'attacco, una possibile soluzione consiste nell'utilizzare un algoritmo di crittografia asimmetrica durante la trasmissione del vettore di feedback $b(p)$. Grazie a ciò, è possibile intercettare solo un testo cifrato, in modo che posizioni diverse siano indistinguibili dall'attaccante. Tuttavia, questa soluzione prevede una crittografia a priori, e ciò comporta una modifica della trasmissione standard. Un'altra opzione di mitigazione, conforme con le specifiche, prevede che l'UE selezioni precedentemente i *precoder* U nel *codebook* che forniscono i tassi d'errore più alti. Tra questi, l'UE sceglierà casualmente e ne restituirà l'indice al gNB. Quindi alla stessa posizione verranno associati più vettori di precodifica, riducendo l'accuratezza dell'attacco. Prendendo in esame la modalità di $feedback$ 1, e chiamando $m^* = 1, 2, ..., U$ dei valori di $feedback$ distinti, quindi

$$m_{u_1} \neq m_{u_2} \quad \forall \quad u_1 \neq u_2. \tag{1.23}$$

  Ciò comporta che l'efficienza sulla sottobanda k diminuiscono, quindi

$$R[\{w(k) = w_{m_1^*, n_1^*(k)}] \geq ... \geq R[\{w(k) = w_{m_U^*, n_U^*(k)}]. \tag{1.24}$$

  Ora l'UE genera indici u equiprobabili all'interno dell'insieme U, restituendo $m_u^*, \{n^*(k)\}_u$. Ciò favorisce delle imprecisioni nella localizzazione, ma comporta una riduzione dei dati trasmessi in *downlink*, dovuta ad una

scelta subottimale del vettore di precodifica.

1.5 Analisi delle prestazioni MSE

L'efficacia dell'attacco può essere stimata basandosi sull'errore quadratico medio (in inglese MSE) della posizione stimata, secondo

$$MSE = E[||\hat{p} - p||^2] \tag{1.25}$$

o come la sua radice $RMSE = \sqrt{MSE}$. Per calcolare l'MSE della posizione, dobbiamo partire dalle seguenti ipotesi su UE e gNB:

1. l'uso della modalità di feedback 3;
2. una stima perfetta del canale dell'UE;
3. una polarizzazione fissa dell'UE con $\mu = \pi/4$.

La prima ipotesi è utile per capire l'impatto del $feedback$ per più sottobande. Per le altre due ipotesi, invece, vi saranno più vettori di $feedback$ $b(p)$ deterministici per ogni posizione p. Per quanto riguarda il modello del canale, si assuma

4. $\alpha_\ell(p,k)$ randomico ed indipendente ad ogni posizione ($d_S = 0$);
5. $\alpha_\ell(p,k)$ randomico ed indipendente ad ogni intervallo di controllo del canale;
6. un'antenna ricevente $\bar{N} = 1$;
7. un canale sulla sottobanda k, determinato solo dal *cluster* $\ell^*(k)$, avente il guadagno di canale più alto;
8. una distribuzione uniforme delle UE nell'area della cella.

L'ipotesi 4) rimuove la consistenza spaziale di $\alpha_\ell(p,k)$. Con la 5) si assume invece che $\alpha_\ell(p,k)$ cambia anche quando l'UE rimane ferma nella stessa posizione. Quindi, quando l'attaccante raccoglie esempi per costruire la mappa probabilistica $p_{map}(,s)$, il valore di $\alpha_\ell(p,k)$ cambia. Nell'ipotesi 6), viene scelto il $beamformer$ di trasmissione con il rapporto più alto che è

$$w(k) = \begin{bmatrix} a_{\ell^*(k)}(p,k) \\ a_{\ell^*(k)}(p,k) \end{bmatrix}. \tag{1.26}$$

Le ipotesi 6) e 7) sono connesse: difatti, quando si utilizzano più antenne sul ricevitore, potrebbero essere sfruttati due *cluster*. Di conseguenza, il $beamformer$ scelto non verrà connesso con nessuno dei due, ma dividerà tra i due la potenza del segnale, anche in base ai guadagni del *cluster*. Dall'ipotesi 7), usando le formule da (1.3) a (1.11), si è in grado di calcolare l'indice del cluster avente il più alto guadagno di canale. In particolare, indicando la distanza tra il gNB e il *cluster* ℓ come $d_\ell = ||q_{\ell 2}||$ e la distanza tra la posizione p e il *cluster* ℓ come $\bar{d}_\ell(p) = ||p - q_\ell||_2$, si definisce

$$\zeta_\ell(p,k) = \frac{\xi_\ell(k)}{(\bar{d}_\ell(p) + d_\ell)^2}, \tag{1.27}$$

dove

$$\xi_\ell = |g_\ell(k)|^2. \tag{1.28}$$

L'indice del cluster scelto dalla sottobanda k è ottenuto al massimo di $\gamma_\ell(p,k)$, considerando

$$\ell^*(k) = argmax_{\ell \in L}(\zeta_\ell(p,k)). \tag{1.29}$$

In questo caso, diremo che l'UE è collegata al *cluster* $\ell^*(k)$.

A. MSE condizionale per determinate posizioni e guadagni del *cluster*
Per prima cosa, vengono calcolate le statistiche di $feedback$ per le posizioni q_l e i guadagni del *cluster* considerato. Per le posizioni, la distanza tra il *cluster* e il gNB è deterministica, come anche la distanza tra il *cluster* e l'UE lo è per una data posizione p di una vittima. Chiamando $X(p, \{q_\ell\}, \{\xi_\ell\}, k)$ un vettore di dimensione NO, i cui valori $X_i(p, \{q_\ell\}, \{\xi_\ell\}, k)$ sono la probabilità che l'UE in una sottobanda k restituisca $i_2(k) = i$. Dalle ipotesi 4) e 5) sappiamo che il $beamformer$ è il rapporto massimo del $beamformer$ combinato ad $a_{\ell(k)}(p,k)$, ottenendo

$$X(p, \{q_\ell\}, \{\xi_\ell\}, k) = P\Big[\frac{d}{\lambda_k} sin\phi(p, q_{\ell^*(k)}, k) \in \Big(\frac{i - 0.5}{NO} - \bar{o}, \frac{i + 0.5}{NO} - \bar{o}\Big)\Big], \tag{1.30}$$

dove $\bar{o} \in \{0, 1\}$. Grazie alla formula (30) si può definire

$$\theta_1(i,k,\tau,\bar{o}) = \frac{1}{c_{ASD}}\Big\{arcsin\Big[\frac{\lambda_k(i - 0.5 - \bar{o}NO)}{dNO}\Big]_{-1}^{1} - (-1)^\tau \phi'(q_{\ell^*(k)}) - \tau\pi\Big\}, \tag{1.31}$$

$$\theta_2(i,k,\tau,\bar{o}) = \frac{1}{c_{ASD}}\Big\{arcsin\Big[\frac{\lambda_k(i+0.5-\bar{o}NO)}{dNO}\Big]_{-1}^{1} - (-1)^{\tau}\phi'(q_{\ell^*(k)}) - \tau\pi\Big\}, \tag{1.32}$$

dove $[x]_{a1}^{a2} = min(max(x,a_1),a_2)$. Da 1.6, e richiamando la formula $\alpha_{\ell^*(k)}(p,k) \sim U([-1,1])$, possiamo ottenere

$$X_i(p,\{q_\ell\},\{\xi_\ell\},k) = \frac{1}{2}\sum_{r\in\mathbb{Z}}\sum_{\bar{o}=0}|[[\theta_1(i,k,\tau)]_{-1}^{1},[\theta_1(i,k,\tau)]_{-1}^{1}]|. \tag{1.33}$$

Per un vettore di $feedback$ b, sia

$$B(b) = \sum_{k=1}^{K}(NO)^{k-1}b_k. \tag{1.34}$$

Poiché i guadagni di canale sono indipendenti per sottobanda, $b(k)$ sarà indipendente per k, per delle posizioni note. Quindi, il vettore colonna $Y(p,\{q_\ell\})$ di lunghezza $(NO)^K$ della funzione di massa di probabilità congiunta (PMF) del vettore di bit di $feedback$ ha come valori

$$Y_{B(b)}(p,\{q_e ll\},\{\xi_\ell\}) = \prod_{k=1}^{K}X_{i_2(k)}(p,\{q_e ll\},\{\xi_\ell\},k) \tag{1.35}$$

mentre il valore medio di PMF tra tutte le posizioni dell'UE è

$$\bar{Y}_{B(b)}(\{q_e ll\},\{\xi_\ell\}) = \frac{1}{A}\int Y_{B(b)}(p,\{q_e ll\},\{\xi_\ell\})dp, \tag{1.36}$$

dove $A = \int dp = \pi R^2$ è la cella e $b \in B$. La ricostruzione MSE delle posizioni del cluster note e dei guadagni è calcolata come

$$MSE((\{q_\ell\},\{\xi_\ell\}) = \frac{1}{A}\sum_{b}\int ||p-\hat{p}(b)||^2 Y_{B(b)}(p,\{q_\ell\},\{\xi_\ell\})dp. \tag{1.37}$$

B. Valore Atteso

Assumendo che l'Ue abbia raccolto i bit di $feedback$ da ogni posizione, la (1.22) diventa

$$\hat{p}(b) = E[p] = \int p\frac{Y_{B(b)}(p,\{q_e ll\},\{\xi_\ell\})}{A\bar{Y}_{B(b)}(\{q_e ll\},\{\xi_\ell\})}dp, \tag{1.38}$$

dove A al denominatore deriva dalla distribuzione uniforme della posizione p della UE nella cella. L'MSE medio rispetto al *cluster* e alla posizione del gNB è ottenuto integrando (1.37) sul PDF delle posizioni del *cluster* e dei guadagni è:

$$MSE = \int\int ... \int\int \frac{MSE(\{q_\ell\}, \{\xi_\ell\})}{A^L} (\prod_{\ell=1}^{L} \prod_{k=1}^{K} (\mathrm{e}^{(\xi_l(k))}) \tag{1.39}$$

Se i *cluster* si trovano all'interno della cella, gli integrali si calcolano sull'intera zona della stessa. Questi possono essere calcolati numericamente.

C. Analisi asintotica

Dal momento che il calcolo esplicito dell'MSE a dell'RMSE in 1.39 richiede una risoluzione numerica degli integrali, nel prossimo paragrafo si affronta il comportamento asintotico dell'RMSE, considerando il numero di sottobande, i *cluster* del gNB, e così via. Per prima cosa, consideriamo che nessun $feedback$ sia disponibile. Ora, l'attaccante non ha informazioni specifiche sulla posizione della vittima dell'UE, se non la consapevolezza che questa è uniformemente distribuita nella cella. Data la forma circolare della lista, l'MSE minimo viene considerato il centro della cella. Nel caso estremo in cui $K = 0$, l'RMSE ($RMSE_0$) risulta

$$RMSE_0 = \sqrt{\frac{1}{A} 2\pi \int_0^R R^2 R dR} = \sqrt{\frac{1}{A} \frac{2\pi}{4} R^4} = \frac{1}{\sqrt{2}} R. \tag{1.40}$$

Questo valore descrive l'accuratezza di una tecnica di localizzazione basata sull'identità di rete temporanea dell'utente quando non implica il rilevamento. Con l'approccio proposto, ci saranno comunque delle informazioni disponibili rispetto a quelle considerate. Quindi 1.40 è un limite superiore dell'RMSE dell'attacco.

Considerando invece che l'attaccante intercetti per caso una risposta dell'UE che utilizza la modalità di $feedback$ 3 e un gNB dotato di molteplici antenne, cioè $N \to \infty$. Tenendo conto $\xi_\ell(k) \approx E[\xi_\ell(k)]$, da 1.27 e 1.29, abbiamo che l'UE è sempre legata al *cluster* con il cammino più breve e ogni *cluster* porta a un indice di $feedback$ differente. Assumendo un gran numero di *cluster* $L \to \infty$ e una distribuzione uniforme dei *cluster* nella cella, si possono approssimare le regioni assegnate ai singoli *precoder* come regioni di Voronoi attorno ai *cluster* di raggio $r \approx R/\sqrt{L}$. Quando il nume-

ro di antenne tende ad infinito, localizziamo l'UE nelle regioni di Voronoi del suo *cluster* più vicino, e possiamo approssimare l'RMSE come

$$RMSE_{\infty} = \lim_{L,N\to\infty} RMSE(K) \approx \frac{R}{\sqrt{2L}}. \tag{1.41}$$

Per valori intermedi di $N > 0$, posizioniamo l'UE sovrapponendo le K mappe, ognuna con NO regioni. Nonostante le mappe siano tutte diverse tra loro, sono tutte ottenute dalle stesse posizioni dei *cluster*, che sono fortemente collegati tra loro. Pertanto, l'RMSE diminuisce secondo il fattore

$$\lim_{K\to\infty} \frac{logRMSE(K)}{K} = -\frac{1}{\eta} logNO, \tag{1.42}$$

Infine, da un'elaborazione di 1.40, 1.41 e 1.42, otteniamo

$$RMSE \approx \frac{R}{\sqrt{2L}} + \big(1 - \frac{1}{\sqrt{L}}\big)\frac{R}{\sqrt{2}}(NO)^{\frac{K}{\eta}}. \tag{1.43}$$

Questo risultato non si basa sulle ipotesi iniziali, e quest'analisi può essere applicata alle altre modalità di $feedback$, quando viene regolato adeguatamente η.

1.6 Conclusioni

In [1], è stato proposto un nuovo tipo di attacco, secondo il quale un attaccante localizza una UE intercettandone il segnale di $feedback$ di precodifica, trasmesso in chiaro secondo l'attuale standard 3GPP. Sono state dettagliate le operazioni che l'attaccante deve fare e analizzata la localizzazione RMSE ottenuta usando varie modalità di $feedback$. È stata anche proposta una tecnica di mitigazione, dove l'UE seleziona casualmente uno dei *precoder* tra quelli che forniscono il tasso più alto. Questa tecnica si è rivelata efficace aumentando l'RMSE della localizzazione a più contatori, con una riduzione trascurabile del *downlink*.

Capitolo 2

Ricostruzione automatica della mappa

Il lavoro spiegato in questa tesi tenta di risolvere un problema derivante da ciò che viene spiegato nel primo capitolo.
In esso si assume infatti di avere la mappa che fa corrispondere ad ogni posizione il valore di $feedback$, e tramite l'intercettazione dei codici di $feedback$ inviati da un dispositivo, l'attaccante riesca a localizzare l'UE. Nei prossimi paragrafi si spiega come questo problema sia stato affrontato, partendo da una mappa casuale e tentando di ricostruirla tramite un algoritmo.
Si suppone che la cella raccolga sequenze di segnali di $feedback$ nel tempo e che il terminale si muova a velocità costante, ma in posizioni ignote nella rete. Inoltre, le sequenze raccolte sono parziali, ovvero non coprono tutta la mappa, e discontinue, cioè sono raccolte in generale in posizioni diverse della mappa.

2.1 Versione senza ripetizioni

Affrontando il problema, si è preferito implementare prima una versione che prevede una semplificazione, ovvero che i codici siano univoci all'interno della mappa, avendo un rapporto uno ad uno tra il numero di elementi e il numero di celle. Per ricostruire la mappa, si parte individuando i 4 potenziali angoli, e posizionandone uno nella prima cella in alto a sinistra. Da questo, l'algoritmo si muove verso il basso cercando nella matrice di adiacenza gli elementi confinanti a quello esaminato precedentemente. Nel caso si trovasse una corrispondenza, l'elemento viene posto nella casella corretta. La procedura viene reiterata fino alla fine della colonna, ovvero quando non viene trovato un altro angolo. Una volta terminata la prima colonna, si ripete il processo per la riga inferiore e successivamente fino al completamento della cornice esterna. Una volta arrivati al termine della prima procedura si cerca un altro angolo, aggiungendo ai controlli fatti in precedenza che sia adiacente all'angolo posizionato nel ciclo precedente. Questo procedimento viene iterato fino a che la mappa ricostruita non è completamente riempita, come mostrato in figura 2.1. In questo caso, la mappa (sopra) è stata ricostruita subendo una rotazione di 180° nella rebuild (sotto), dovuta al fatto che l'algoritmo posiziona per primo l'angolo col codice inferiore.

Nella prima versione del codice viene creata una matrice chiamata *mappa* in maniera casuale, della quale le caselle sono le celle e i numeri rappresentano i codici delle stesse. Da questa, tramite una funzione scritta appositamente, si crea una matrice d'adiacenza chiamata *adj*, le cui righe e colonne indicano i codici della mappa originale; tramite un'ispezione della mappa, quando due codici risultano adiacenti in orizzontale, in diagonale o in verticale, il valore della cella nella matrice *adj* viene incrementato di uno.

Per prima cosa, dopo aver creato *adj*, il programma memorizza nel vettore S la somma delle colonne di *adj* e crea una matrice di dimensioni $R * C$ chiamata *rebuild*, che sarà la nuova mappa. Tramite un ciclo for riesce poi ad individuare gli angoli, in quanto saranno quelle celle del vettore S il cui valore corrisponde a tre. Durante il primo ciclo while, l'angolo scelto è il primo del vettore *Angles*, di conseguenza quello con il codice "minore" dei quattro.
Una volta collocato il primo angolo nella prima cella della nuova mappa *rebuild*, il programma completa la prima colonna grazie ad un altro ciclo while. All'interno di questo, vi sono più controlli: il primo appura se il programma si trova al primo ciclo o meno, gli altri invece verificano se l'elemento in esame sia un lato o un angolo, se è adiacente o meno all'elemento precedente *prev* e se è già presente in

rebuild tramite la funzione *is_present*. Se tutti questi controlli sono soddisfatti, l'elemento viene aggiunto al vettore *next*, all'interno del quale vi saranno i candidati per occupare la prossima cella disponibile. Questo vettore è gestito tramite una variabile intera k che viene resettata alla fine di ogni riempimento di una cella; se l'elemento trovato è un lato viene posizionato alla cella $k - esima$, se invece è un angolo viene inserito nella prima posizione di questo vettore, cosicché sia posto come prossimo elemento.

	1	2	3	4	5	6	7	8	9	10
1	99	3	60	38	47	84	21	50	36	24
2	32	96	74	100	98	90	25	87	82	95
3	40	68	78	58	80	97	52	26	71	39
4	22	16	72	76	14	83	37	43	23	13
5	34	69	62	81	46	59	64	94	65	9
6	92	11	70	89	56	79	31	19	2	66
7	91	54	51	42	63	5	49	44	4	20
8	35	30	33	28	93	48	27	15	18	57
9	6	45	7	17	8	53	61	73	85	10
10	55	77	86	41	67	29	88	1	75	12

	1	2	3	4	5	6	7	8	9	10
1	12	10	57	20	66	9	13	39	95	24
2	75	85	18	4	2	65	23	71	82	36
3	1	73	15	44	19	94	43	26	87	50
4	88	61	27	49	31	64	37	52	25	21
5	29	53	48	5	79	59	83	97	90	84
6	67	8	93	63	56	46	14	80	98	47
7	41	17	28	42	89	81	76	58	100	38
8	86	7	33	51	70	62	72	78	74	60
9	77	45	30	54	11	69	16	68	96	3
10	55	6	35	91	92	34	22	40	32	99

Figura 2.1

2.2 Funzioni ausiliari

Nella scrittura dell'algoritmo sono state usate delle funzioni per gestire la ricostruzione della mappa, delle quali di seguito ne viene data una breve spiegazione:

1. **adjacency(R, C, mappa):** i parametri che vengono passati alla funzione sono una matrice, detta mappa, e le dimensioni della stessa, così da costruire una matrice di adiacenza. La mappa è riempita con dei numeri che vanno da uno al prodotto delle dimensioni: questi rappresentano i codici associati al precoder. La funzione restituisce una matrice che ha per dimensioni il prodotto delle dimensioni della mappa, dove ogni volta che un numero nella mappa risulta vicino ad un altro, viene incrementato il valore contenuto nella casella della matrice di adiacenza di uno.

2. **azzera(next):** la funzione prende come parametro un vettore di interi, e lo restituisce dopo aver sostituito ogni elemento con il numero zero.

3. **following(next, i, k):** la funzione prende come parametri il vettore nel quale vanno inseriti gli elementi vicini ad un codice già inserito nella mappa ricostruita, l'elemento analizzato e la variabile k, che è un contatore per posizionare l'elemento nella giusta cella del vettore. La funzione inserisce l'elemento i alla $k - esima$ posizione.

4. **is_present(e, rebuild):** la funzione prende come parametri un elemento e e la mappa ricostruita: ispezionando quest'ultima, restituisce il valore uno se l'elemento è presente, zero altrimenti.

2.3 Spiegazione dell'algoritmo con codici univoci

La mappa viene ricostruita grazie ad un algoritmo a spirale, che è stato scritto per funzionare su delle mappe con numero di righe e colonne pari. Di seguito, vi saranno una breve introduzione di ogni matrice, vettore e variabile necessari alla scrittura dell'algoritmo e la descrizione dell'algoritmo stesso.

- **R e C:** queste due variabili, inizializzate all'inizio del codice, rappresentano il numero di righe e colonne della mappa.
- **cont:** questa viene inizializzata a zero, e incrementata al completamento di ogni "cornice"; serve per sapere se il ciclo è al primo giro o meno.
- **conta_elementi:** viene incrementata ad ogni inserimento di un elemento nella mappa ricostruita, così da tenere il conto di elementi inseriti.
- **r e c:** queste due variabili vengono utilizzate per identificare la cella da riempire nella mappa ricostruita.
- **mr e mc:** vengono decrementate, permettendo all'algoritmo di arrestarsi al momento opportuno.
- **prev:** questa variabile viene utilizzata per salvare l'ultimo elemento inserito nella mappa ricostruita.
- **k:** è usata nella funzione following per tenere traccia della posizione in cui inserire gli elementi. Viene resettata ad uno ad ogni ciclo while.
- **S:** che contiene la somma di ogni colonna della matrice di adiacenza. Ciò comporta che gli indici del vettore corrispondono agli elementi della mappa.
- **next:** contiene i plausibili elementi da inserire nella mappa.
- **mappa:** la matrice mappa ha un numero di elementi pari ad R*C e viene inizialmente riempita con la funzione randperm(R*C) per poi essere riarrangiata in forma di mappa.
- **adj:** la matrice viene dichiarata ed inizializzata grazie alla funzione adjacency, già spiegata in precedenza.
- **adj_copy:** questa è una copia della matrice *adj*, che verrà modificata nel corso dell'algoritmo mantenendo intatta la versione di *adj* originaria.

- **rebuild:** questa è una matrice di zeri di dimensione R*C nella quale attraverso delle manipolazioni verrà ricreata la mappa.

All'interno di un ciclo *while*, le colonne e le righe vengono riempite seguendo l'andamento a spirale.
La prima parte del ciclo si concentra sull'individuare un angolo libero, 2.2. Le variabili c e r vengono incrementate, il vettore S viene nuovamente inizializzato su *adj_copy* e viene riempito il vettore *Angles*, come mostrato in 2.3: per riempire questo vettore, viene iterato un ciclo for sul vettore S e ogni qualvolta un elemento di S risulta uguale a tre, l'indice relativo viene inserito nel vettore.

	1	2	3	4	5	6	7	8	9	10
1	12	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Figura 2.2: Il posizionamento del primo angolo nella matrice ricostruita

L'algoritmo, grazie alla variabile *cont*, riesce a determinare se si trova al primo giro oppure al secondo. Nel caso in cui la ricostruzione della cornice esterna fosse già stata compiuta e ci si trovasse quindi al secondo giro, viene controllato anche che l'elemento dentro il vettore *Angles* sia anche adiacente all'angolo del giro precedente. Alla fine della procedura, le variabili *prev*, *conta_elementi* e r vengono aggiornate.

	1	2	3	4
1	12	24	55	99

Figura 2.3: Il vettore Angles alla prima implementazione

Dopo aver scelto l'angolo, si procede compilando la colonna a sinistra, come si vede in 2.4: un ciclo for cerca gli elementi adiacenti a quello salvato in *prev* che nel vettore S hanno un valore assegnato di 5 o di 3, prendendo quindi in considerazione solo gli elementi che sono sul lato o su un angolo e inserendoli nel vettore *next* se e solo se non sono già stati inseriti in *rebuild*, eliminando problemi di ridondanza. Alla fine del ciclo for, il primo elemento del vettore *next* viene inserito in *rebuild* e le variabili *prev*, r, e c vengono aggiornate.

	1	2	3	4	5	6	7	8	9	10
1	12	0	0	0	0	0	0	0	0	0
2	75	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0
4	88	0	0	0	0	0	0	0	0	0
5	29	0	0	0	0	0	0	0	0	0
6	67	0	0	0	0	0	0	0	0	0
7	41	0	0	0	0	0	0	0	0	0
8	86	0	0	0	0	0	0	0	0	0
9	77	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Figura 2.4: Riempimento della colonna a sinistra

```
for i = 1:R*C
   if (((adj_copy(i, prev) == 1)  && is_present(i, rebuild) == 0))
       if(S(i) == 3)
           next(1) = following(next, i, k);
           break;
       elseif (S(i) == 5)
           if(conta_elementi >= 2)
               if(adj(i, rebuild(1, 1)) == 0)
                   next = following(next, i, k);
               end
           else
               next = following(next, i, k);
           end
       end
   end
end
```

Figura 2.5: Codice per l'inserimento

Nel codice in 2.5, è riportata l'operazione di selezione di un elemento della colonna sinistra al primo giro. Negli altri casi vengono modificati i controlli nel primo costrutto if.

Anche in questo caso, l'algoritmo separa il caso in cui si trova al primo giro dagli altri sempre grazie alla variabile *cont*. Questa procedura viene ripetuta per altre tre volte, riempiendo la colonna in 2.6 e le righe restanti, come mostrato in 2.7 e 2.8.

Alla fine dei quattro cicli *while* le variabili *mr* e *mc* vengono decrementate, permettendo all'algoritmo di arrestarsi quando il loro prodotto è minore di 4 (in tal caso, si sarebbe raggiunto il centro della matrice), viene incrementato il valore di *cont* e r e c vengono settate a *cont*, così da ricominciare il ciclo *while* successivo dalla posizione del prossimo angolo, posto in diagonale rispetto al precedente: se il primo si trova in posizione $(1, 1)$, il secondo si troverà in posizione $(2, 2)$.

	1	2	3	4	5	6	7	8	9	10
1	12	0	0	0	0	0	0	0	0	24
2	75	0	0	0	0	0	0	0	0	36
3	1	0	0	0	0	0	0	0	0	50
4	88	0	0	0	0	0	0	0	0	21
5	29	0	0	0	0	0	0	0	0	84
6	67	0	0	0	0	0	0	0	0	47
7	41	0	0	0	0	0	0	0	0	38
8	86	0	0	0	0	0	0	0	0	60
9	77	0	0	0	0	0	0	0	0	3
10	55	6	35	91	92	34	22	40	32	99

Figura 2.6: Riempimento della colonna a destra

	1	2	3	4	5	6	7	8	9	10
1	12	10	57	20	66	9	13	39	95	24
2	75	85	0	0	0	0	0	0	0	36
3	1	0	0	0	0	0	0	0	0	50
4	88	0	0	0	0	0	0	0	0	21
5	29	0	0	0	0	0	0	0	0	84
6	67	0	0	0	0	0	0	0	0	47
7	41	0	0	0	0	0	0	0	0	38
8	86	0	0	0	0	0	0	0	0	60
9	77	0	0	0	0	0	0	0	0	3
10	55	6	35	91	92	34	22	40	32	99

Figura 2.7: Riempimento della riga in alto

	1	2	3	4	5	6	7	8	9	10
1	12	0	0	0	0	0	0	0	0	0
2	75	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0
4	88	0	0	0	0	0	0	0	0	0
5	29	0	0	0	0	0	0	0	0	0
6	67	0	0	0	0	0	0	0	0	0
7	41	0	0	0	0	0	0	0	0	0
8	86	0	0	0	0	0	0	0	0	0
9	77	0	0	0	0	0	0	0	0	0
10	55	6	35	91	92	34	22	40	32	0

Figura 2.8: Riempimento della riga in basso

2.4 Versione con ripetizioni

Nei paragrafi precedenti è stata affrontata una versione semplificata del problema, nella quale i codici all'interno delle celle sono distinti fra loro. Il capitolo successivo si sofferma sul caso in cui i codici nella mappa possono ripetersi in celle adiacenti dando luogo a delle "macchie" con lo stesso codice, in rappresentanza di celle che coprono un'area più vasta. Questo nuovo approccio ha portato ad una nuova versione della mappa, in grado di funzionare sia in delle mappe

che contengono delle macchie di codici, sia nella versione ristretta affrontata nei paragrafi precedenti.
Nell'implementare questo codice, è stato supposto che i codici ripetuti non potessero essere nella cornice; questo ha dato la possibilità di riutilizzare una parte del codice precedente, assieme alle varie funzioni ausiliari. Nella figura 2.9 vengono riportate la mappa originale e la mappa ricostruita.

	1	2	3	4	5	6	7	8	9	10
1	99	1	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	52	71	26	39
5	22	16	72	76	14	14	37	43	23	13
6	34	69	62	81	14	14	64	94	65	9
7	92	11	70	89	56	79	31	19	2	66
8	91	54	51	63	42	5	49	44	44	20
9	35	30	33	28	93	48	27	44	44	57
10	45	6	7	17	8	53	61	73	85	10

	1	2	3	4	5	6	7	8	9	10
1	10	85	73	61	53	8	17	7	6	45
2	57	44	44	27	48	93	28	33	30	35
3	20	44	44	49	5	42	63	51	54	91
4	66	2	19	31	79	56	89	70	11	92
5	9	65	94	64	14	14	81	62	69	34
6	13	23	43	37	14	14	76	72	16	22
7	39	26	71	52	97	80	58	78	68	40
8	95	82	87	25	90	98	100	74	96	32
9	24	36	50	21	84	47	38	60	3	77
10	55	75	12	88	29	67	41	86	1	99

Figura 2.9: In questo caso, la mappa originale, riportata sopra, è stata ricostruita subendo una rotazione di 180° nella rebuild, riportata sotto, dovuto al fatto che l'algoritmo posiziona per primo l'angolo col codice inferiore.

2.5 Spiegazione dell'algoritmo con ripetizioni

L'algoritmo per risolvere il problema spiegato nel paragrafo precedente sfrutta molte funzioni, vettori, matrici e variabili della versione che non prevedeva delle ripetizioni della mappa. In aggiunta, utilizza:

- **noa:** un vettore chiamato *noa*, Number Of Appearances, di lunghezza R*C, che per ogni elemento della mappa, rappresentato dall'indice delle sue celle, ha salvato il numero di volte che questo deve comparire nella mappa. Questo è ottenuto attraverso l'uso del vettore S, come riportato nel codice riportato in 2.10

```
noa = zeros(R*C, 1);    %vettore che tiene il Number Of Apparition (NOA) degli elementi
for i = 1:R*C
    if(S(i) == 3 || S(i) == 5 || S(i) == 8)
        noa(i) = 1;
    else
        noa(i) = S(i)/8; %per ogni comparsa del numero, entra in contatto con 8 elementi
    end
end
```

Figura 2.10

- **diag_sx, left, above, diag_dx:** queste quattro variabili memorizzano ad ogni ciclo gli elementi attorno alla cella che sta per essere riempita, con gli elementi già posizionati. Il nome è esplicativo, ma prendono gli elementi delle celle nelle due diagonali in alto, nella cella a sinistra e in quella sopra. Nel caso i 4 valori corrispondessero, si andrà a prendere i valori salvati nelle celle distanti di una posizione nella stessa direzione, come spiegato in 2.11.

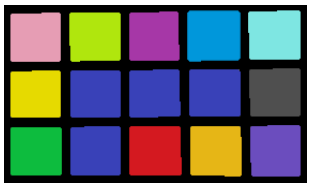

Figura 2.11: In questo caso, per posizionare l'elemento rosso, le caselle attorno a lui sono dello stesso colore. Il codice prevede quindi che non si andranno ad interpellare le caselle a lui adiacenti, ma quelle di distanza uno, quindi gli altri colori.

L'algoritmo funziona tramite l'ancoraggio agli elementi circostanti. Una volta ricostruita la cornice, l'algoritmo interroga per ogni cella le quattro celle adiacenti che sono già state riempite per poi posizionare l'elemento che risulta adiacente a tutte e quattro. Ogni volta che l'indicatore delle colonne c arriva alla casella adiacente alla cornice e l'indicatore delle righe r è minore del numero di righe R, c viene reimpostato a due in modo da partire dalla seconda colonna, considerato che la prima è già occupata dalla cornice, e l'indice r viene incrementato di uno, così da spostarsi alla riga successiva. In 2.12 e 2.13 si vede come la mappa viene riempita, con il relativo codice Tramite una ripetizione del ciclo esposto prece-

```
indice = 1;
while (conta_elementi < R*C)
    %[...]nella parte di codice mancante, v'è la manipolazione delle quattro variabili descritte
    for indice = 1:R*C
        if(adj(diag_sx, indice) > 0 && adj(left, indice) > 0 && adj(above, indice) > 0 && adj(diag_dx, indice) > 0 && noa(indice)>0)
            rebuild(r, c) = indice;
            c = c+1;
            if (c == C && r < R)
                c = 2;
                r = r+1;
            end
            conta_elementi = conta_elementi + 1;
            noa(indice) = noa(indice)-1;
            break;
        end
    end
end
```

Figura 2.12

	1	2	3	4	5	6	7	8	9	10
1	10	85	73	61	53	8	17	7	6	45
2	57	44	44	27	48	93	28	33	30	35
3	20	0	0	0	0	0	0	0	0	91
4	66	0	0	0	0	0	0	0	0	92
5	9	0	0	0	0	0	0	0	0	34
6	13	0	0	0	0	0	0	0	0	22
7	39	0	0	0	0	0	0	0	0	40
8	95	0	0	0	0	0	0	0	0	32
9	24	0	0	0	0	0	0	0	0	77
10	55	75	12	88	29	67	41	86	1	99

Figura 2.13

dentemente, si arriva al completamento della mappa, raggiungendo il risultato in 2.9

Questo codice può presentare degli errori nella ricostruzione della mappa. Questa discrepanza è dovuta al fatto che il codice analizza le sequenze in ordine crescente: ciò comporta che nel ricostruire una macchia che si sviluppa su più righe, può incorre in delle incongruenze, nel caso in cui un codice della seconda riga della macchia fosse maggiore del codice nella cella adiacente come in 2.14; quest'ultimo codice sarebbe posizionato per primo, come mostrato in 2.15.

99	86	41	67
3	100	100	47
96	100	100	98
68	78	58	80

Figura 2.14

99	86	41	67
3	100	100	47
96	100	98	0
0	0	0	0

Figura 2.15

Quest'errore comporta un arresto della ricostruzione e l'ingresso dell'algoritmo in un ciclo infinito.

2.6 Risultati

Alla fine delle due soluzioni, per verificare la correttezza delle stesse, vengono paragonate le matrici di adiacenza della mappa originale e della mappa ricostruita: nel caso fossero uguali, vuol dire che la ricostruzione è avvenuta correttamente. Il primo algoritmo è stato testato tramite il codice sorgente "script.m", che ripete il codice per 100 volte calcolando la percentuale di correttezza e lancia un messaggio d'errore nel caso le due matrici di adiacenza non fossero uguali. L'algoritmo scritto ha ottenuto una percentuale di correttezza del 100%, rispettando le aspettative che erano state poste al principio.

Per il testing della seconda soluzione è stato utilizzato un metodo manuale, scrivendo volta per volta diverse matrici con diverse macchie di diverse forme in posizioni diverse. La percentuale di correttezza del codice in questo caso è stata dell'84%. Il motivo della discrepanza è spiegato nel paragrafo 2.5

Conclusioni

La tesi si proponeva di trovare un metodo per ricostruire la mappa che associa ai canali quantizzati la posizione del dispositivo, partendo da delle sequenze di canali quantizzati intercettati.
Il problema è stato affrontato con due differenti approcci. Il primo approccio al problema prevede di ricreare una mappa che non presentasse delle ripetizioni al suo interno. L'algoritmo ha soddisfatto le aspettative, ricostruendo la mappa originale nel 100% dei casi.

Nella seconda versione proposta invece si è cercato un metodo per ricreare una mappa che potesse presentare delle ripetizioni nelle celle; in questo caso, a causa di alcuni limiti del codice già descritti nel paragrafo 2.5, la percentuale di correttezza del codice si ferma all'84%.

Bibliografia

[1] *Localization Attack by Precoder Feedback Overhearing in 5G Networks and Countermeasures*
`https://arxiv.org/abs/2012.07727`

[2] *5G; Location Management Services; Stage 3, document TS 29.572 Version 15.1.0 Release 15, 3GPP, Jul. 2018.*

[3] *A. Shahmansoori, G. E. Garcia, G. Destino, G. Seco-Granados, and H. Wymeersch, "Position and orientation estimation through millimeter wave MIMO in 5G systems," IEEE Trans. Wireless Commun., vol. 17, no. 3, pp. 1822–1835, Mar. 2018.*

[4] *R. H. Clarke, "A statistical theory of mobile-radio reception," Bell Syst. Tech. J., vol. 47, no. 6, pp. 957–1000, Jul./Aug. 1968.*

[5] *K. Michael and M. G. Michael, "The social and behavioural implications of location-based services," J. Location Based Services, vol. 5, nos. 3–4, pp. 121–137, Sep. 2011, doi: 10.1080/17489725.2011.642820.*

[6] *R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions," IEEE Commun. Surveys Tuts., vol. 22, no. 1, pp. 196–248, 1st Quart., 2020.*

[7] *Y. Wang, "Linear least squares localization in sensor networks," EURASIP J. Wireless Commun. Netw., vol. 2015, no. 1, pp. 1–7, Mar. 2015.*

[8] *K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. M. Ni, "CSI-based indoor localization," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 7, pp. 1300–1309, Jul. 2013.*

[9] *S. Tewes, A. A. Ahmad, J. Kakar, U. M. Thanthrige, S. Roth, and A. Sezgin, "Ensemble-based learning in indoor localization: A hybrid approach," in Proc. IEEE 90th VTC-Fall, Honolulu, HI, USA, Sep. 2019, pp. 1–5.*

[10] *N. Pirzada, M. Y. Nayan, M. F. Hassan, F. Subhan, and H. Sakidin, "WLAN location fingerprinting technique for device-free indoor localization system," in Proc. 3rd ICCOINS, Kuala Lumpur, Malaysia, Aug. 2016, pp. 650–655*

[11] *A. Chakraborty, L. E. Ortiz, and S. R. Das, "Network-side positioning of cellular-band devices with minimal effort," in Proc. IEEE Conf. INFOCOM, Kowloon, China, Apr./May 2015, pp. 2767–2775.*

[12] *F. Wen, J. Kulmer, K. Witrisal, and H. Wymeersch, "5G positioning and mapping with diffuse multipath," IEEE Trans. Wireless Commun., early access, Oct. 21, 2020, doi: 10.1109/TWC.2020.3031180.*

[13] *D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking LTE on layer two," in Proc. IEEE Symp. Secur. Privacy (SP), San Francisco, CA, USA, May 2019, pp. 1121–1136.*

Ringraziamenti

Alla fine di questo percorso, sono sentiti i ringraziamenti a coloro che mi sono stati vicini in questo viaggio.
Il primo va ai miei genitori, per l'amore di cui sono stati capaci. Grazie per la pazienza e il sostegno che non mi avete fatto mancare in questi anni, stando al mio fianco nei momenti belli e in quelli più bui.
Grazie a mia sorella, per i consigli e l'esperienza che mi ha donato, e per avermi sempre saputo offrire un punto di vista valido alle situazioni della vita.
Un ringraziamento ai miei parenti, il cui affetto negli anni non ha mai tardato a farsi sentire, festeggiando nei successi e dandomi sostegno nei momenti di sconforto.
A Luca, mio cugino e caro amico, che con le sue parole è sempre stato in grado di spronarmi a fare di più e a dare del mio meglio.
Ai miei amici, per essere stati sempre presenti anche durante questa ultima fase del mio percorso di studi. Grazie per tutti i momenti di spensieratezza.
Un ultimo ringraziamento ad Anna, per il sostegno e la comprensione che mi hai dato in questo periodo, e per avermi tranquillizzato quando ne avevo bisogno.
Un grazie infinite, a tutti voi.

Riccardo