Appunti sul codice:

* chiave non ha senso inizializzarla a stringa, ma fare randi per generarla direttamente
* decidere se lavorare con la potenza in decibel o in volt
* transmission power va fatta in milliwatt
  + dopo trasformiamo in decibel chiedendo come fare
* poi applichiamo la formula
  + potenza ricevuta = potenza trasmessa / (quadrato della distanza)
* riduzione di potenza del segnale in base a lunghezza d’onda e distanza (ritrovare la formula di Bluetooth)

Prendiamo le tre distanze x,y,z e ne calcoli la distanza euclidea. Partiamo prima su solo “x” e poi runniamo simulazioni, quando funziona, inserendo anche “y” e “z”

In base al segnale, trovare la riduzione che permette di considerare la perdita e quindi ottenere il segnale ricevuto.

Quando codifico ho due valori di threshold semplici:

* threshold = potenza max dato
  + voglio un BER minore di una certa percentuale
  + il valore di threshold va adattato in base a quanto perdo; se perdo poco, non devo ingegnerizzarlo finché rimango sopra al valore di threshold
  + distanza massima per threshold e ottenere queste prestazioni
  + occorre trovare il valore di potenza che ci deve essere
* potenza chiave + potenza dato = un certo valore di soglia
* vedere come si modella il rumore
  + 1) dato dal canale
  + 2) dato dalla strumentazione

Immaginiamo di avere LOS (senza interferenze, dritto per dritto), l’effetto del rumore sul canale è dato solo dalla path loss. Dopo, va inserito il rumore della strumentazione, che viene definito sulla base di SNR. Serve inserire una gamma di SNR che va da -20 db a +20 db (intervallo, selezioniamo valori qui) – c’è più rumore che segnale.

Si va come “segnale misurato = awgn(segnale\_SNR)” che genera un segnale modificato rispetto a quello presente. Genera un segnale complesso (in teoria), noi ne prendiamo la parte reale.

Ogni segnale creato venga plottato, per vedere sia corretto. Definiamo noi un bit rate o durata del segnale (decidiamo sulla base del bitrate di Bluetooth). Con valori reali non ci sono problemi di sincronizzazione.

Lì facciamo la parte di decodifica 🡪 il dato ricevuto diventa 1 se il segnale ricevuto è maggiore del centro (centrarlo in 0 per semplicità, altrimenti per scalarlo), 0 altrimenti. Definiamo lì il parametro center

Troviamo bit corretti segnale e bit diversi chiave e plottiamo (per esempio, variando la distanza con quella threshold, che BER si ottiene). Questo serve per ottenere la distanza massima per ottenere u BER target. Va fatto sia per il dato che per la chiave.

(Prima parte: threshold e decodifica)

(Seconda parte: threshold legitimacy)

BER è calcolato con la distanza di Hamming. Ricavato il mio segnale di autenticazione, devo dire “quanti errori ammetto sulla chiave perché sia autentico”?

Occorre fare uno studio su false alarm e misdetection

1. FALSE ALARM (mando solo autentici)
   1. invio solo messaggi autentici (calcolo “threshold = …” (array di valori)) – ogni volta cambia la chiave, messaggio e la distanza (meglio fare così)
      1. quanti segnali autentici tra quelli ricevuti
      2. quanti segnali autentici tra quelli inviati
   2. due loop uno per distanza e uno per threshold. dopo calcoliamo false alarm per la coppia distanza-threshold
   3. Capire dove variare se dentro o fuori, considerando l’effetto awgn.
   4. Plot
2. MISSED DETECTION (mando solo non autentici)
   1. chiave reale
   2. chiave non autentica
   3. generiamo il segnale con chiave non corretta
   4. e ripetiamo lo stesso ragionamento
   5. plot

Dovremmo avere una curva per ogni valore di threshold (quindi, di ogni distanza). Facciamo un grafico per distanza minima – distanza massima – distanza centrale). Prendiamo 3/4 valori e ogni linea rappresenta un valore singolo di threshold. Ogni riga dovrebbe essere una threshold e ognuna rappresenta una combinazione distanza-valore