Da ricevitore, senza rumore, sono in grado di prendere sia i bit di autenticazione che quelli del messaggio (guardando solo i valori di picco). 4 valori possibili di picco: alto-alto(medio-alto/medio-basso/basso-basso.

La positività dà info sul messaggio; prendendo come soglia x, sopra x il bit del messaggio era 1, se sotto il bit messaggio è 0. Quando il valore assoluto del picco è minore del valore di picco del messaggio erano discordi – per decodificare il messaggio, guardo solo positivo(negativo, per decodificare l’autenticazione, guardo il valore assoluto del picco che ricevo.

Se minore del valore assoluto del bit del messaggio, discordi, altrimenti se maggiore sono concordi. L’approccio di sovrapposizione permette comunque all’attaccante di agire (sapendo a che frequenza trasmettiamo, i valori di potenza iniziali – sia di messaggio che autenticazione). L’unica cosa “che non sa” è la chiave di autenticazione e la vuole prendere per ricostruire il messaggio.

Occorre capire come aggiungere l’additive noise, ma anche come agisce il ricevitore (retrieval della chiave e calcolare “una sorta” di distanza di Hamming della chiave – se dista più di tot bit/un parametro dal messaggio originale, lo scarta).

Capire se fare lo studio direttamente sul segnale di somma oppure del segnale di additive noise. Si può fare che, assumendo di avere il canale in un certo modo (conoscendo lo stato del canale, che aggiunge un certo gain).

* Esempio: considerazione della varianza sul segnale ricevuto, non sulla potenza.
  + Segnale di autenticazione possibilmente corrotto, cause rumore/interferenze
    - False alarm
    - Misdetection = Chiave sbagliata ma, per qualche motivo, accetto un messaggio non legittimo

Cosa possiamo fare:

* Simuliamo di conoscere tutto il canale, sapendo perfettamente l’azione sul canale
* False alarm = Generare per n volte il messaggio con l’autenticazione che rimane la stessa, ma cambia il dato
* Proviamo diversi valori di picco possibili (sia per autenticazione che del messaggio)
  + Per ogni valore di picco del messaggio si prova ogni valore di picco dell’autenticazione (potenza)
* Ripetiamo n volte: generiamo il segnale al trasmettitore, il ricevitore riceve e decodifica la chiave e calcoliamo la distanza della chiave ricevuta da quella che si aspetta di ricevere
  + Questo per capire, per ogni coppia potenza-messaggio/potenza-autenticazione
  + Permette di trovare il valore di threshold per cui confrontare la distanza
    - 1/2/3/4… bit (non tanto a numeri specifici, ma a percentuale di lunghezza della chiave, scalando rispetto ad essa)
    - Valutando in questo modo 10%, etc.
* Per ogni tripletta di questi valori (potenza messaggio/potenza autenticazione/valore di threshold), calcoliamo false alarm e misdetection
  + Scegliamo se inviare messaggio autentico o messaggio forgiato con chiave randomica
    - Decidere per false alarm quanti messaggi vengono rifiutati
    - Decidere per misdetection quanti messaggi vengono accettati
  + Per il momento, lo scegliamo casualmente e poi ne discutiamo
* Questo ci permette di settare i vari possibili scenari di potenza conoscendo il canale rispetto alla threshold che conviene avere

Da capire se farlo sul messaggio puro (segnale + autenticazione) oppure quello del segnale con additive noise.

Primo approccio:

* Capire se “ingegnerizzare” l’additive noise
  + Riportando il segnale in potenza entro certi valori
  + Servono a dare dei boundaries piuttosto che concentrarsi sui singoli valori misurati
* In questo caso devo condividere il valore dell’additive noise aggiunto
  + Permette al ricevitore di tornare indietro

Secondo approccio (meno dispendioso energeticamente – no invio info rumore calcolato):

* Aggiunta del rumore in modo casuale
  + Assumendo che venga generato secondo una distribuzione di probabilità con certa media e varianza (conosciuta anche dall’attaccante)
  + L’unica cosa ulteriormente segreta è il seed al PRNG
    - Se il seed è lo stesso e la funzione di sampling è la stessa
    - Il ricevitore non ha bisogno di sapere il valore specifico di additive noise
    - Capendo il seme, la sequenza è quella
    - Si può generare in locale il valore di additive noise, rimuoverlo e verificare
* In questo modo, il messaggio non rimane entro certi valori di potenza
* Capire come controbilanciare gli effetti del valore assoluto di decadimento tra distanza e potenza
* Proviamo a trovare dei modelli di canale semplici (e.g., Bluetooth)

In particolare:

1. Modello la potenza del segnale in base alla teoria
   1. Posizione ricevitore (x,y,z)
   2. Il trasmettitore invia il messaggio con una certa potenza
   3. Vediamo la potenza ottenuta dal ricevitore in base alla distanza
   4. Vediamo il valore assoluto dei picchi presi dal ricevitore
   5. Direttamente sul messaggio sovrapposto (senza additive noise)
2. Trovare un modello di canale ed implementare la sua azione

Giusto per capire nelle distanze del Bluetooth (entro 100 m.) e facciamo un test, se basandoci esclusivamente sui valori assoluti, riusciamo ad avere buone prestazioni di decodifica. Altrimenti, occorre trovare un diverso approccio.

Esempio libreria simulazione: Sionna (simula la trasmissione reale dei messaggi e il canale wireless – basta dirgli il canale e lui trova tutto).

Summary:

1. Studio delle specifiche del Bluetooth
   1. Potenza da utilizzare
      1. Non serve simulare frequency shifting serve solo la potenza
2. Simulazione trasmissione del messaggio (fading = caduta di potenza)
   1. Prova di varie distanze (fino a 150 m. max)
3. Vedere quanto il valore assoluto della potenza decade
   1. Proviamo a diverse distanze
4. Assumiamo comunicazione diretta
5. Proviamo a vedere se Matlab permette un toolbox di canale simulazione Bluetooth
   1. O un canale wireless con del fading
      1. Basta che sia wireless, la proprietà Bluetooth servono per frequenze/potenza e altri fattori
   2. Per capire l’interferenza che esiste sui messaggi
6. Generiamo solo situazioni autentiche (no controllo se va bene/va male)
7. Troviamo solo quanti bit sono sbagliati
   1. Capendo prestazionalmente se conviene

Questo ci serve per capire la parte di decodifica:

* per basarci sui valori assoluti
* o sui valori del ricevitore
  + capire in quali fasce (livelli) – intorni dove vengono ricevuti i segnali
  + l’importante è distinguere bene i valori

A noi sta l’idea di come raccogliere e presentare i vari risultati.

Potremmo provare a farlo:

* sia sul segnale sovrapposto
* che sull’approccio dell’additive noise
  + assumendo che condividano il seme
  + buttiamo del rumore casuale sia in potenza che in fase
  + decidiamo come aggiungere questo

Attenzione ai dispositivi su cui agiamo: potenza, segno, consumo dei dispositivi utilizzati.

* Vedere come il fading agisca sul ricostruire il segnale – se lo ricevo esattamente come generato, ritorna il messaggio originale.
* Con tanto fading, il messaggio con additive noise si riduce di potenza ma tolgo un valore di potenza maggiore del risultante
* Capire come i picchi aiutino ad evitare messaggi erroneamente decodificati
  + Se la decodifica va bene, il messaggio deve poi essere corretto
  + Se la decodifica dà dei problemi, cerchiamo di capire cosa fare
    - Accettazione del messaggio (capire come)
* Facendo test solo su fading, serve un parametro di riferimento per capire la chiave

In summa:

* Studio di come realizzare il decoding (valori assoluti o meno)
* Variando la posizione di ricevitore e trasmettitore usando la potenza normale del Bluetooth (proviamone diverse)
* Facciamolo sia con che senza additive noise
* Con additive noise usiamo secondo approccio (aka casuale)

Vedere se conviene questo approccio, altrimenti usiamo approccio teorico (Line-of-Sight) e uso delle formule di decadimento del segnale.

Paragrafo della discussione anche per Alessandro.