Daniel Pantjuskin-Moos
108013248222
Vincent König
108011232630
Gruppe: D

Abgabe PHYSEC 4

1. Bit Error Rate

```
def ber(X, Y):
  hw = 0.0
  for x,y in len(X):
    hw += x^y

  BER = hw/len(X) # len(X) = len(Y)

  if 0 <= BER <= 0.5:
    return BER
  else:
    return 1 - BER
```

2. Implementierung Quantisierer

Implementierung Jana Multibit

```
def JanaMultibit(X):
  #(i)determine the Range of RSS measurements from the minimum and the maximum
                                        measured RSS values
  x_min = min(X)
  x_max = max(X)

  #(ii) find N, the number of bits that can be extracted per measurement
  # N is already implemented
  N = number_of_bits

  #(iii) divide the Range into m = 2^n equal sized intervals
  M = 2**N  # M intervals, len(Range) = M+1
  interval = abs((max(X)-min(X))/float(M)) # interval size

  Range =[]   # intervals

  if interval == 0: #min=max
    interval = 1

  while x_min <= x_max: # fill in
    Range.append(x_min)
    x_min += interval

  #(iv) choose an N bit assignment for each of the M intervals (for example
                                       use the Gray code sequence)
  bit_assignment = utils.gray_code(N)

  #(v) for each RSS measurement, extract N bits depending on the interval in
                                       which the RSS measurement lies.
  quantized=[]
  for x in X:
    for j in xrange(len(Range) - 1):
      if Range[j] <= x <= Range[j+1]: # including max and min values
        quantized.extend(bit_assignment[j])
        break
    if len(Range) == 1: # min=max - only one value
      quantized.extend(bit_assignment[0])
  return quantized
```

3. Attack Trees

Modifizierbare *draw.io*-Datei hier downloaden zur näheren Inspizierung.

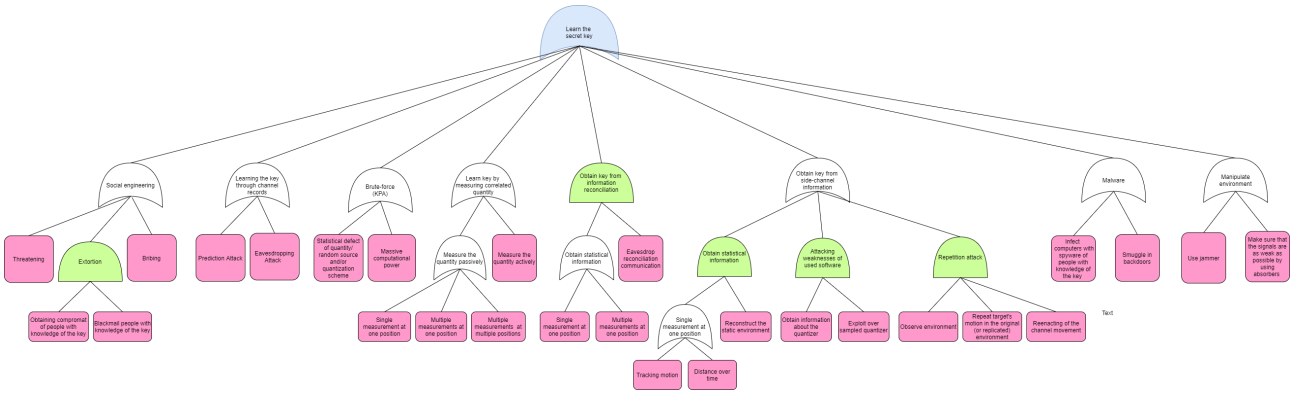

Abbildung 1: Attack tree

Weitere Angriffsszenarien neben

- *Repetition Attack*
- *Prediction Attack*
- *Eavesdropping Attack*
- *Angriff auf den Quantisierer*

sind unter anderem *social engineering* Attacken wie

- *Bestechen*
- *Drohen*
- *Erpressen mithilfe von Kompromat*

oder *Schadsoftwareattacken* wie

- Spyware
- Backdoors

sowie *Manipulation des Kanals* wie

- Jammen
- Signalabsorber
- Attacken wie das Alternieren zwischen Umklammern und Loslassen der Antenne

4. Angriffe

Eavsdropping Attack

Mit folgendem Befehl wurden die Messdaten konkateniert (ohne LF):

```
python konk_data.py -A 1_ohneBwg_A_B.csv 2_mBwg_A_B.csv 3.a_mBwg_A_B.csv
3.a_ohneBwg_A_B.csv 3.b_mBwg_A_B.csv 3.b_ohneBwg_A_B.csv 4_mBwg_A_B.csv
5_mBwg_A_B.csv -B 1_ohneBwg_B_A.csv 2_mBwg_B_A.csv 3.a_mBwg_B_A.csv
3.a_ohneBwg_B_A.csv 3.b_mBwg_B_A.csv 3.b_ohneBwg_B_A.csv 4_mBwg_B_A.csv
5_mBwg_B_A.csv -E 1_ohneBwg_E_A.csv 2_mBwg_E_A.csv 3.a_mBwg_E_A.csv
3.a_ohneBwg_E_A.csv 3.b_mBwg_E_A.csv 3.b_ohneBwg_E_A.csv 4_mBwg_E_A.csv
5_mBwg_E_A.csv
```

Die Ausgaben namens *A.csv*, *B.csv* und *E.csv* können hier runtergeladen werden.

Folgende Befehle wurden zum Testen des Frameworks benutzt.

```
python physec_praktikum.py -A A.csv -B B.csv -E E.csv -X 2 -Q 0
python physec_praktikum.py -A A.csv -B B.csv -E E.csv -X 2 -Q 1
```

Die Ausgaben des Frameworks können hier untergeladen werden.

Quantisierer im direkten Vergleich

Schaut man sich Abbildungen 2 und 3 an, so erkennt man merkbare Unterschiede zwischen beiden *Quantisierern*.

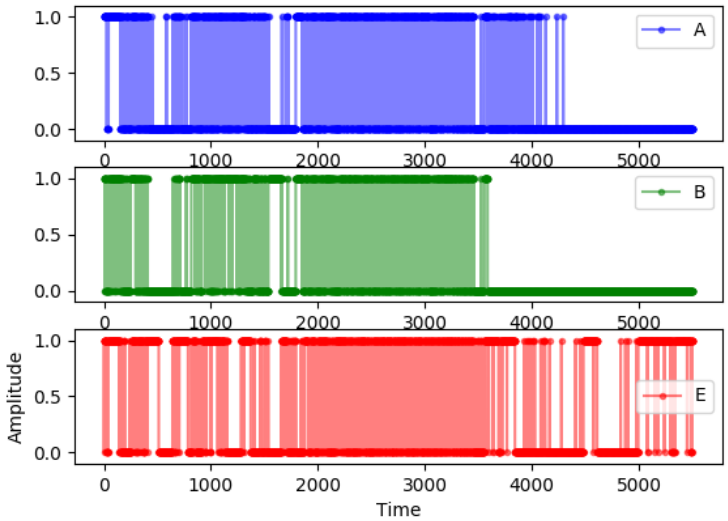

Abbildung 2: *quant0*

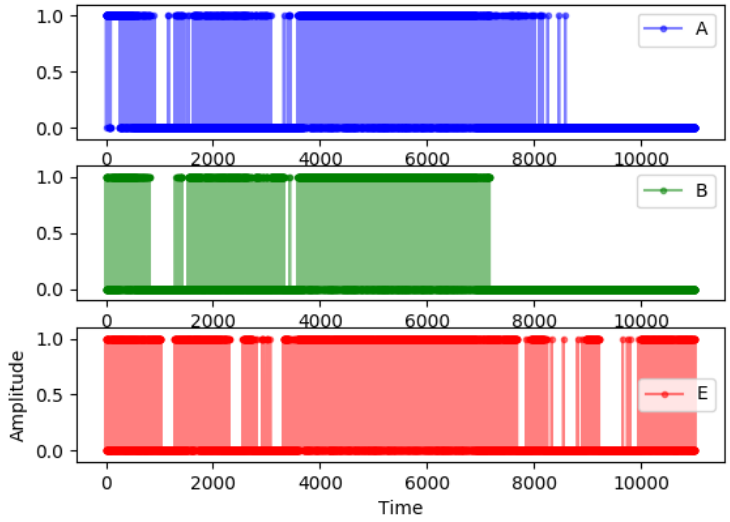

Abbildung 3: *quant1*

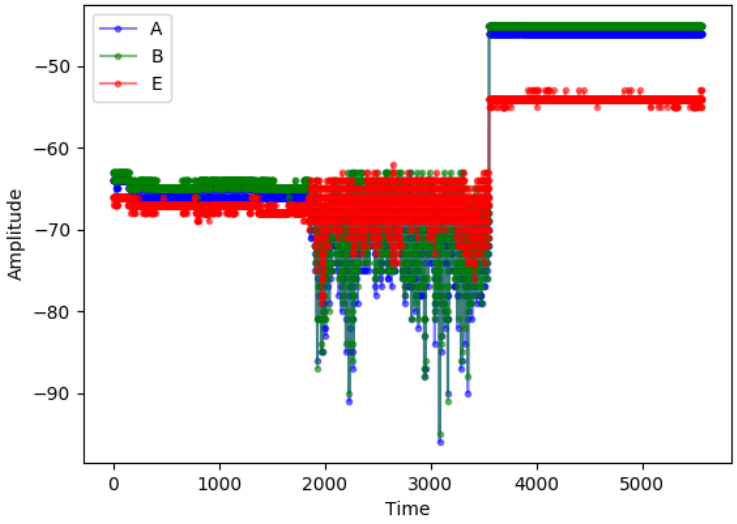

Abbildung 4: *Signalstärkenverlauf*

Wenn man Abbildung 2 betrachtet, so erkennt man, dass Einflüsse wie Bewegung und Distanz eine große Rolle spielen.
Analysiert man die Unterschiede so wird klar, dass Bewegung und größere Distanz einen Angreifer benachteiligen.

5. Reading Assignment

5..1 Nennen Sie 5 verschiedene Quellen für Entropie

Physikalische Phänomene

- Radioaktiver Zerfall
- Transmission von Photonen
- Thermisches Rauschen (Johnson-Nyquist-Rauschen)
- Geräusche der Atmosphäre
- Jitter

Mensch-Gerät Interaktion

- Dauer von Tastendrücken und andere Zeitmessungen
- Tasten-und Mausbewegungen
- Bewegungs- und Beschleunigungssensoren

5..2 Wie viele Samples aus einem Smartphone Accelerator benötigen wir um einen 128 bit key zu erstellen?

„Since the recommended security level for almost random bits is $\epsilon = 2^{-80}$. According to the heuristic (1) we need roughly 128/0.125 = 1024 samples to extract a 128-bit key. However taking into account the true error in our Theorem 1 we see that we need at least n $\approx$ 2214 bits!“

Laut dem Paper, welches sie auf Theoreme stützt, braucht man mindestens 1024 Samples. Hinzu kommt allerdings, dass empfohlen wird, ein *n* mit mehr als 2214 Bitlänge zu wählen.

5..3 Wie groß ist der Unterschied an tatsächlich extrahierbarer Entropie und Shannon Entropie?

Corollary 1 (A Signifficant Entropy Loss in the AEP Heuristic Estimate). *In the above setting, the gap between the Shannon entropy and the number of extractable bits ϵ-close to uniform equals at least $\Theta(\sqrt{\log(1/\epsilon)kn})$. In particular, for the recommended security level ($\epsilon = 2^{-80}$) we obtain the loss of $kn - N \approx \sqrt{80kn}$ bits, no matter what an extractor we use.*

Abbildung 5: Auszug aus dem Paper

„In information theory most widely used is Shannon entropy, which quantifies the encoding length of a given distribution. In turn, cryptographers use the more conservative notion called min-entropy, which quantifies unpredictability. In general, there is a large gap between these two measures: the min-entropy of an n-bit string might be only $O(1)$ whereas its Shannon entropy as big as $\Omega(n)$."

Die min-Entropie eines n-bit langen stringes kann konstant sein, wobei gleichzeitig seine Shannon Entropie linear oder größer zu n sein kann.

Von den 2214 bits können nur etwa 128 bits mit nahezu zufällig gleichverteilter Qualität extrahiert werden. Das ist ein Verhältnis von $\frac{128}{2214} \approx 0.0578$ oder anders gesagt: Der Schlüssel hat hier nur die Länge von 5,78% von der Bitlänge des Samples.