```
import numpy as np
from scipy.stats import multivariate_normal
import warnings
warnings.filterwarnings('ignore')
hamming = [bin(n).count('1') for n in range(256)]
sbox = (99, 124, 119, 123, 242, 107, 111, 197, 48, 1, 103, 43, 254, 215,
    171, 118, 202, 130, 201, 125, 250, 89, 71, 240, 173, 212, 162, 175, 156,
    164, 114, 192, 183, 253, 147, 38, 54, 63, 247, 204, 52, 165, 229, 241,
    113, 216, 49, 21, 4, 199, 35, 195, 24, 150, 5, 154, 7, 18, 128, 226,
    235, 39, 178, 117, 9, 131, 44, 26, 27, 110, 90, 160, 82, 59, 214, 179,
    41, 227, 47, 132, 83, 209, 0, 237, 32, 252, 177, 91, 106, 203, 190, 57,
    74, 76, 88, 207, 208, 239, 170, 251, 67, 77, 51, 133, 69, 249, 2, 127,
    80, 60, 159, 168, 81, 163, 64, 143, 146, 157, 56, 245, 188, 182, 218,
    33, 16, 255, 243, 210, 205, 12, 19, 236, 95, 151, 68, 23, 196, 167, 126,
    61, 100, 93, 25, 115, 96, 129, 79, 220, 34, 42, 144, 136, 70, 238, 184,
    20, 222, 94, 11, 219, 224, 50, 58, 10, 73, 6, 36, 92, 194, 211, 172, 98,
    145, 149, 228, 121, 231, 200, 55, 109, 141, 213, 78, 169, 108, 86, 244,
    234, 101, 122, 174, 8, 186, 120, 37, 46, 28, 166, 180, 198, 232, 221,
    116, 31, 75, 189, 139, 138, 112, 62, 181, 102, 72, 3, 246, 14, 97, 53,
    87, 185, 134, 193, 29, 158, 225, 248, 152, 17, 105, 217, 142, 148, 155,
    30, 135, 233, 206, 85, 40, 223, 140, 161, 137, 13, 191, 230, 66, 104,
    65, 153, 45, 15, 176, 84, 187, 22)
def cov(x, y):...
all_traces = np.load('./chipwhisperer/trac...')[:-1]
pt = np.load('./chipwhisperer/plai...')
knownkey = np.load('./chipwhisperer/key....')
numFeatures = 25
number_of_training = 5000
number_of_tesing = 9
number_of_additional = 5
```

calls

warnings.filterwarnings

bin.count

range

np.load

np.load

np.load

for subkey in range(16):

range(16)

```
print('subkey[%d]' % subkey)
tracesTrain = all_traces[0:number_of_training]
ptTrain = pt[0:number_of_training]
outputSbox = [sbox[ptTrain[i][subkey] ^ knownkey[i][subkey]] for i in range
    (len(ptTrain))]
outputSboxHW = [hamming[s] for s in outputSbox]
TracesHW = [[] for _ in range(9)]
```

range

range

print

len

for i in range(len(tracesTrain)):

range(len(tracesTrain))

```
TracesHW = [np.array(TracesHW[HW]) for HW in range(9)]
Means = np.zeros((9, len(tracesTrain[0])))
```

```
HW = outputSboxHW[i]
TracesHW[HW].append(tracesTrain[i])
```

range

len

TracesHW.append

range

np.zeros

np.array

len

for i in range(9):

range(9)

SumDiff = np.zeros(len(tracesTrain[0]))

Means[i] = np.average(TracesHW[i], 0)

range

np.zeros

len

np.average

for i in range(9):

range(9)

for j in range(i):

range(i)

```
SumDiff_old = SumDiff.copy()
features = []
featureSpacing = 5
```

range

SumDiff += np.abs(Means[i] - Means[j])

SumDiff.copy

np.abs

for i in range(numFeatures):

range(numFeatures)

```
nextFeature = SumDiff.argmax()
features.append(nextFeature)
featureMin = max(0, nextFeature - featureSpacing)
featureMax = min(nextFeature + featureSpacing, len(SumDiff))
```

```
meanMatrix = np.zeros((9, numFeatures))
covMatrix = np.zeros((9, numFeatures, numFeatures))
```

range

np.zeros

np.zeros

for HW in range(9):

range(9)

SumDiff.argmax

features.append

max

min

len

for j in range(featureMin, featureMax):

range(featureMin, featureMax)

range

range

SumDiff[j] = 0

```
pooled_covMatrix = covMatrix.mean(axis=0)
tracesTest = all_traces[9000:9000 + number_of_tesing + number_of_additional]
ptTest = pt[9000:9000 + number_of_tesing + number_of_additional]
key_rank = np.zeros(16)
P_k = np.zeros(256)
```

for i in range(numFeatures):

range(numFeatures)

range

meanMatrix[HW][i] = Means[HW][features[i]]

covMatrix.mean

np.zeros

np.zeros

for j in range(numFeatures):

range(numFeatures)

range

```
x = TracesHW[HW][:, (features[i])]
y = TracesHW[HW][:, (features[j])]
covMatrix[HW, i, j] = cov(x, y)
```

cov

for j in range(len(tracesTest)):

range

len

test_X = [tracesTest[j][features[i]] for i in range(len(features))]

for kguess in range(0, 256):

range(0, 256)

range

len

```
print('Top 10 guesses: ', P_k.argsort()[-10:...'t')
tarefs = np.argsort(P_k)[::-1]
key_rank[subkey] = list(tarefs).index(knownkey[j][subkey])
        print('Highest GE: ', key_rank[subkey], ...'

')
```

```
HW = hamming[sbox[ptTest[j][subkey] ^ kguess]]
rv = multivariate_normal(meanMatrix[HW], pooled_covMatrix)
p_kj = rv.pdf(test_X)
P_k[kguess] += np.log(p_kj)
```

print

np.argsort

list.index

print

P_k.argsort

rv.pdf

np.log

multivariate_normal

range

len

CFIonTA

cov

```
return np.cov(x, y)[0][1]
```

KEY

try

while

for

if

raise

return

input

default

call