Task4

DPA on DES S-boxes

I ran a Differential Power Analysis (DPA) on the first round of DES using power traces captured from a ChipWhisperer board. For each of the eight S-boxes the script tested all 64 possible 6-bit subkey guesses. For every guess it split the traces into two groups (LSB of the S-box output = 0 or 1), averaged each group, and measured the difference. The script reports the five key guesses with largest difference peaks for each S-box and the sample index where the peak occurs. Example output (partial) showed keys such as `0x27`, `0x31`, `0x04` for S-box 1 and similar top-5 lists for S-boxes 2..8. These top guesses indicate which 6-bit values are most likely for each S-box.

How the attack works (simple)

1. Generate N random 64-bit plaintexts.
2. For each plaintext capture one power trace while the device encrypts it. Save all traces in memory and reuse them.
3. For each S-box (1..8) and each 6-bit key guess (0..63):
    - Use `sbox_out(sbox, plaintext, guess)` to compute the S-box output for every plaintext.
    - Group traces by the least-significant bit (LSB) of that output into `zero` and `one`.
    - Compute the average trace of each group (sample-wise).
    - Compute the absolute difference of these two average traces.
    - Find the maximum peak in the difference and its sample index. This peak value measures how strongly the guessed key explains power differences.
4. For each S-box sort the 64 guesses by peak and report the top five guesses and their sample indices.

This method finds small data-dependent leakage in the power traces; correct key guesses tend to produce larger peaks at the time samples where the device processes the S-box.

How to run the script (exact)

1. Requirements: Python 3, `numpy`, `chipwhisperer` Python package, and `sbox_out.py` in the same folder. Make sure ChipWhisperer hardware is connected.

2. Make the script executable and run with the number of traces you want. Example:

```
chmod +x dpa.py
./dpa.py 2000
# or
python3 dpa.py 2000
```

This will generate 200 random plaintexts in memory, capture 2000 traces, then run DPA and print the top 5 key guesses per S-box. The script saves traces to `traces/` and also to `traces_all.npy` / `plaintexts_all.npy`.

```
=== DPA top 5 keys per S-box (reuse traces) ===

S-box 1:
  #1: key= (0x27), max_peak=0.004486, sample=3853 (original ≈ 3853)
  #2: key= (0x31), max_peak=0.004429, sample=3893 (original ≈ 3893)
  #3: key= (0x04), max_peak=0.004363, sample=3853 (original ≈ 3853)
  #4: key= (0x10), max_peak=0.004010, sample=3893 (original ≈ 3893)
  #5: key= (0x19), max_peak=0.003898, sample=3829 (original ≈ 3829)

S-box 2:
  #1: key= (0x2F), max_peak=0.004184, sample=3829 (original ≈ 3829)
  #2: key= (0x1A), max_peak=0.003911, sample=3829 (original ≈ 3829)
  #3: key= (0x07), max_peak=0.003668, sample=3801 (original ≈ 3801)
  #4: key= (0x0E), max_peak=0.003474, sample=3829 (original ≈ 3829)
  #5: key= (0x04), max_peak=0.003029, sample=3765 (original ≈ 3765)

S-box 3:
  #1: key= (0x19), max_peak=0.005610, sample=3841 (original ≈ 3841)
  #2: key= (0x18), max_peak=0.004555, sample=3841 (original ≈ 3841)
  #3: key= (0x2C), max_peak=0.004093, sample=3841 (original ≈ 3841)
  #4: key= (0x09), max_peak=0.004092, sample=3865 (original ≈ 3865)
  #5: key= (0x08), max_peak=0.003659, sample=3841 (original ≈ 3841)

S-box 4:
  #1: key= (0x0E), max_peak=0.005261, sample=3961 (original ≈ 3961)
  #2: key= (0x21), max_peak=0.004470, sample=3829 (original ≈ 3829)
  #3: key= (0x36), max_peak=0.004063, sample=3829 (original ≈ 3829)
  #4: key= (0x23), max_peak=0.003676, sample=3765 (original ≈ 3765)
  #5: key= (0x1E), max_peak=0.003477, sample=3829 (original ≈ 3829)

S-box 5:
  #1: key= (0x13), max_peak=0.004476, sample=3841 (original ≈ 3841)
  #2: key= (0x0A), max_peak=0.003871, sample=4097 (original ≈ 4097)
```

```
  #3: key= (0x3B), max_peak=0.003777, sample=4097 (original ≈ 4097)
  #4: key= (0x07), max_peak=0.003605, sample=4097 (original ≈ 4097)
  #5: key= (0x2F), max_peak=0.003398, sample=3269 (original ≈ 3269)

S-box 6:
  #1: key= (0x2E), max_peak=0.005319, sample=3829 (original ≈ 3829)
  #2: key= (0x03), max_peak=0.004487, sample=3801 (original ≈ 3801)
  #3: key= (0x13), max_peak=0.003974, sample=3801 (original ≈ 3801)
  #4: key= (0x31), max_peak=0.003672, sample=3829 (original ≈ 3829)
  #5: key= (0x12), max_peak=0.003596, sample=3801 (original ≈ 3801)

S-box 7:
  #1: key= (0x21), max_peak=0.004604, sample=3829 (original ≈ 3829)
  #2: key= (0x01), max_peak=0.003844, sample=3829 (original ≈ 3829)
  #3: key= (0x05), max_peak=0.003239, sample=3829 (original ≈ 3829)
  #4: key= (0x03), max_peak=0.003016, sample=3829 (original ≈ 3829)
  #5: key= (0x00), max_peak=0.002896, sample=3829 (original ≈ 3829)

S-box 8:
  #1: key= (0x03), max_peak=0.004860, sample=3917 (original ≈ 3917)
  #2: key= (0x1D), max_peak=0.003704, sample=3829 (original ≈ 3829)
  #3: key= (0x04), max_peak=0.003440, sample=2689 (original ≈ 2689)
  #4: key= (0x08), max_peak=0.003340, sample=3829 (original ≈ 3829)
  #5: key= (0x28), max_peak=0.003251, sample=3801 (original ≈ 3801)
```