Bit-By-Bit: The Side Channel Hackathon

Challenge -3

Side-Channel Key Recovery With Neural Networks

Methodology and Justification

This project focuses on recovering the first byte of the AES encryption key using power traces and deep learning. Our approach involves a profiling attack, where a deep neural network is trained on a labeled dataset (Dataset B) and then used to predict probabilities for each possible key byte on a target dataset (Dataset A) without known keys.

We target the intermediate value z = SBox[Plaintext $\oplus$ Key], as it is a common leakage point in AES and has been shown to correlate well with power consumption. The label for training is derived from z, and we used Hamming Weight (HW) of z as the label (set by USE_HW_LABEL=True) because it is easier to learn and generalizes better across devices.

The power traces were pre-processed using the following steps:

1. Standardization : to normalize the signal.
2. Bandpass filtering : to remove low- and high-frequency noise.
3. Derivative computation : to highlight signal transitions.
4. Feature combination : original + derivative signals were concatenated to enhance temporal information.

We split the profiling dataset into training and validation sets using stratified sampling to maintain class balance.

CNN Architecture Description

We used 1D Convolutional Neural Network (CNN) called Enhanced 1D CNN, tailored for time-series data like power traces. Here's a high-level description:

- Input: Single-channel power trace.
- Initial Conv Layer: Extracts low-level features.
- 3 Residual Blocks: Each with increasing filters (64 → 128 → 256 → 512) to capture complex patterns.
- Attention Mechanism: Learns which time regions are important in the trace.
- Global Pooling: Reduces temporal dimension.

- Classifier: A fully connected network with dropout and ReLU activation, outputting class probabilities (either 256 classes for key bytes or 9 for HW values).

CNNs are a standard choice for side-channel analysis because:

CNNs effectively capture local patterns and temporal dependencies in power traces. They are robust to misalignment and noise, which are common in side-channel signals.

Key Guess Results

The trained model is evaluated on Dataset A. The model outputs probability distributions over possible labels for each trace, which we use to compute log-likelihood scores for all 256 key byte values.

These scores are then sorted to produce a key ranking is shown in the picture below:

```
Epoch 1: train_loss=1.8774, train_acc=0.256, val_loss=1.8684, val_acc=0.280, lr=0.000293
Epoch 2: train_loss=1.8239, train_acc=0.274, val_loss=1.8070, val_acc=0.297, lr=0.000271
Epoch 3: train_loss=1.7666, train_acc=0.312, val_loss=1.7386, val_acc=0.318, lr=0.000238
Epoch 4: train_loss=1.7077, train_acc=0.338, val_loss=1.7138, val_acc=0.330, lr=0.000197
Epoch 5: train_loss=1.6742, train_acc=0.355, val_loss=1.6767, val_acc=0.351, lr=0.000150
Epoch 6: train_loss=1.6303, train_acc=0.376, val_loss=1.7524, val_acc=0.320, lr=0.000104
Epoch 7: train_loss=1.5908, train_acc=0.406, val_loss=1.6397, val_acc=0.375, lr=0.000063
Epoch 8: train_loss=1.5417, train_acc=0.432, val_loss=1.6451, val_acc=0.375, lr=0.000030
Epoch 9: train_loss=1.4952, train_acc=0.455, val_loss=1.6336, val_acc=0.380, lr=0.000008
Epoch 10: train_loss=1.4615, train_acc=0.481, val_loss=1.6261, val_acc=0.385, lr=0.000300
Epoch 11: train_loss=1.6045, train_acc=0.392, val_loss=1.6672, val_acc=0.350, lr=0.000298
Epoch 12: train_loss=1.5724, train_acc=0.410, val_loss=1.7644, val_acc=0.339, lr=0.000293
Epoch 13: train_loss=1.5508, train_acc=0.429, val_loss=1.8247, val_acc=0.321, lr=0.000284
Epoch 14: train_loss=1.5749, train_acc=0.412, val_loss=1.8855, val_acc=0.341, lr=0.000271
Epoch 15: train_loss=1.4929, train_acc=0.458, val_loss=1.7627, val_acc=0.339, lr=0.000256
Epoch 16: train_loss=1.4411, train_acc=0.485, val_loss=1.8249, val_acc=0.353, lr=0.000238
Epoch 17: train_loss=1.3974, train_acc=0.518, val_loss=1.7663, val_acc=0.353, lr=0.000218
Epoch 18: train_loss=1.3359, train_acc=0.553, val_loss=2.9638, val_acc=0.206, lr=0.000197
Epoch 19: train_loss=1.3814, train_acc=0.527, val_loss=2.3514, val_acc=0.265, lr=0.000174
Epoch 20: train_loss=1.2435, train_acc=0.612, val_loss=2.3860, val_acc=0.239, lr=0.000150
Epoch 21: train_loss=1.2042, train_acc=0.646, val_loss=1.9979, val_acc=0.373, lr=0.000127
Epoch 22: train_loss=1.0930, train_acc=0.714, val_loss=2.0184, val_acc=0.370, lr=0.000104
Epoch 23: train_loss=1.0187, train_acc=0.765, val_loss=2.1283, val_acc=0.375, lr=0.000083
Epoch 24: train_loss=0.9549, train_acc=0.809, val_loss=1.9675, val_acc=0.375, lr=0.000063
Epoch 25: train_loss=0.8907, train_acc=0.850, val_loss=2.0795, val_acc=0.378, lr=0.000045
Early stopping at epoch 25

Top 10 key guesses:
#1 Key=3F, Score=-44.40
#2 Key=AD, Score=-46.41
#3 Key=50, Score=-47.92
#4 Key=AF, Score=-52.29
#5 Key=26, Score=-52.79
#6 Key=CA, Score=-52.97
#7 Key=CF, Score=-53.76
#8 Key=49, Score=-54.33
#9 Key=4B, Score=-55.09
#10 Key=9E, Score=-56.44
```

Conclusion

In this project, we have applied a deep learning-based profiling attack on AES using power traces. By targeting the Hamming Weight of the SBox output and leveraging a tailored CNN architecture, we have ranked key byte candidates based on their computed log-likelihood scores derived from the model's predicted probabilities.

The scripts are available at https://github.com/suryaarasu11/bitbybit/ .