1 Details on DNN Architectures

Throughout this section, we denote $classes = 9$ for the HW leakage model and $classes = 256$ for the ID leakage model, and "padding = same" is padding evenly left and right such that the output has the same dimension as the input. We use the same DNN architecture for the CW dataset for both HW and ID leakage models. The architecture is explained in Table 1.

Table 1: DNN architecture used for the CW dataset for both HW and ID leakage models.

Layers	**Hyperparameters**
Conv1D_1	Number of filters/channels out = 8, kernel size = 11, stride = 1, padding = same, activation = ReLU
Average Pooling	kernel size = 2, stride = 2
Linear Regression 1	features out = = 128, activation = ReLU
Linear Regression 2	features out = = 128,activation = ReLU
Linear Regression 3	features out = *classes*, activation = Softmax

Table 2: DNN architecture used for the ASCADf and ASCADr datasets.

Layers	**Hyperparameters**
Conv1D_1	Number of filters/channels out = 128, kernel size = 25, stride = 1, padding = same, activation = SeLU
Batch Normalization	
Average Pooling	kernel size = 25, stride = 25
Linear Regression 1	features out = 20, activation = SeLU
Linear Regression 2	features out = 15, activation = SeLU
Linear Regression 3	features out = *classes*, activation = Softmax

Table 3: DNN architecture used for the AES_HD dataset.

Layers	**Hyperparameters**
Conv1D_1	Number of filters/channels out = 2, kernel size = 1, stride = 1, padding = same, activation = SeLU
Average Pooling	kernel size = 4, stride = 4
Linear Regression 1	features out = 15, activation = SeLU
Linear Regression 2	features out = 10, activation = SeLU
Linear Regression 3	features out = 4, activation = SeLU
Linear Regression 4	features out = *classes*, activation = Softmax

We used the same architecture for ASCADf and ASCADr (Table 2). For the AES_HD dataset, we use the same DNN architecture as in Zaid et al. [1]. Details are in Table 3.

As for DNNs trained on the desynchronized datasets, we show the architectures in Tables 4 and 5 for ASCADf_desync50 and ASCADf_desync100, respectively.

Table 4: DNN architecture used for the ASCADf_desync50 dataset.

Layers	Hyperparameters
Conv1D_1	Number of filters/channels out = 8, kernel size = 34, stride = 17, padding = same, activation = SeLU
Max Pooling	kernel size = 2, stride = 2
Batch Normalization	
Linear Regression 1	features out = 400, activation = SeLU
Linear Regression 2	features out = 400, activation = SeLU
Linear Regression 3	features out = 400, activation = SeLU
Linear Regression 4	features out = 400, activation = SeLU
Linear Regression 5	features out = *classes*, activation = Softmax

Table 5: DNN architecture used for the ASCADf_desync100 dataset.

Layers	Hyperparameters
Conv1D_1	Number of filters/channels out = 12, kernel size = 30, stride = 15, padding = same, activation = SeLU
Max Pooling	kernel size = 2, stride = 2
Batch Normalization	
Linear Regression 1	features out = 300, activation = SeLU
Dropout rate = 0.05	
Linear Regression 2	features out = 300, activation = SeLU
Dropout rate = 0.05	
Linear Regression 3	features out = *classes*, activation = Softmax

References

[1] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for Efficient CNN Architectures in Profiling Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, p. 1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391