1 Hyperparameters for DNNs Used.

Throughout the paper, we use the following hyperparameters for our DNNs. Using these hyperparameters and the architectures stated in our repository, the trained DNNs successfully recovered the key.

Table 1: Hyperparameters used when training DNNs for each dataset.

	CW	**ASCADf**	**ASCADr**	**AES_HD**	**ASCADf_desync50**	**ASCADf_desync100**
Batch Size	128	50	200	256	500	300
Learning Rate	0.0001	0.005	0.005	0.005	0.001	0.001
Epochs	20	50	100	20	100	100
Weight Initialization	Glorot Uniform	He Uniform	He Uniform	He Uniform	Random Uniform	Glorot Uniform
Optimizer	RMSprop	Adam	Adam	Adam	RMSprop	RMSprop
Regularizer	None	None	None	None	l_2	Dropout
Regularizer Strength	-	-	-	-	0.0001	-

2 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 2.

Table 2: 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 3: 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

We used the same architecture for ASCADf and ASCADr (Table 3). For the AES_HD dataset, we use the same DNN architecture as in Zaid et al. [1]. Details are in Table 4.

Table 4: 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

Table 5: 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 6: 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

As for DNNs trained on the desynchronized datasets, we show the architectures in Tables 5 and 6 for ASCADf_desync50 and ASCADf_desync100, respectively.

3 Hyperparameter Search Space to Find More DNNs

In various cases, we would like to find more DNNs for our experiments. The hyperparameter search space to find the 10 DNNs for Section 7 are stated in Table 7.

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.

Hyperparameter	Options
CNN	
Convolution layers	1 to 4 in step of 1
Convolution filters	4 to 16 in step of 4
Kernel size	26 to 52 in step of 2
Pooling type	Average or Max
Pooling size	2 to 10 in step of 2
Number of Dense Layers	1 to 4 in a step of 1
Neurons per layer	$10, 20, 50, 100, 200, 300, 400, 500$
Others	
Batch size	300 to 1100 in a step of 100
Activation function	$ReLU$ or $SeLU$
Optimizer	Adam or RMSprop
Learning Rate	$0.0005, 0.0001, 1e-4, 5e-4$
Weight Initializer	Random Uniform or Glorot Uniform or He Uniform

Table 7: Hyperparameter search space.

1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391