Conference on **Cryptographic Hardware and Embedded Systems** (CHES) 2020

Strength in Numbers: Improving Generalization with Ensembles in Machine Learning-based Profiled Side-Channel Analysis

<u>Guilherme Perin</u>, Lukasz Chmielewski, Stjepan Picek



In collaboration with

riscure

Radboud University

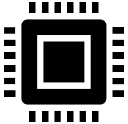
TU Delft

Contributions

- Analysis of output class probabilities (predictions)
- Using proper metrics for profiled SCA with deep learning
- Improving generalization in DL-based profiled SCA:
    - Ensembles: combining multiple NN models into a stronger model

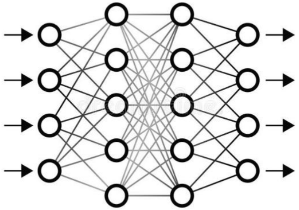
TUDelft

DL-based profiled SCA



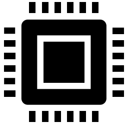
Device A (AES)

Profiling Traces
(*known* key)



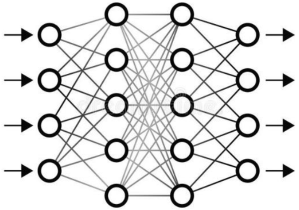
(learning algorithm - DNN)

Validation Traces
(*known* key)



Device B (AES)

Attack Traces
(*unknown* key)



(learning algorithm - DNN)

Recovered key

Good (enough) generalization

"... Improving Generalization ..."

- If (n-order) SCA leakages are there, we can improve generalization by:
  - Using a small NN model (implicitly regularized)
  - Using a large NN model and add (explicit) regularization (dropout, data augmentation, noise layers, batch normalization, weight decay, etc.)
  - Being precise in training time/epochs (early stopping)
  - Or, using **ensembles**.

TUDelft

DL-based SCA is (mostly) about hyperparameters

- No points of interest selection
- Less sensitive to trace desynchronization (CNN)
- Implement high-order profiled SCA
- Allow visualization techniques

More secure products

- Work in progress:
  - Create a good DL model is difficult: efficient and automated hyperparameters tuning not solved yet for SCA
  - SCA is already costly by itself: adding hyperparameters tuning can render the DL-based SCA impractical

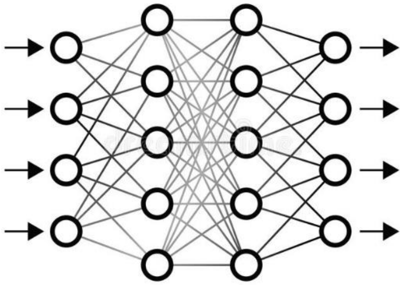
DL-based SCA is (also) about metrics

- Accuracy, Loss, Recall, Precision: not very consistent for SCA (multiple test traces)

- Success Rate
- Guessing Entropy

Custom loss/error function in Keras/TensorFlow

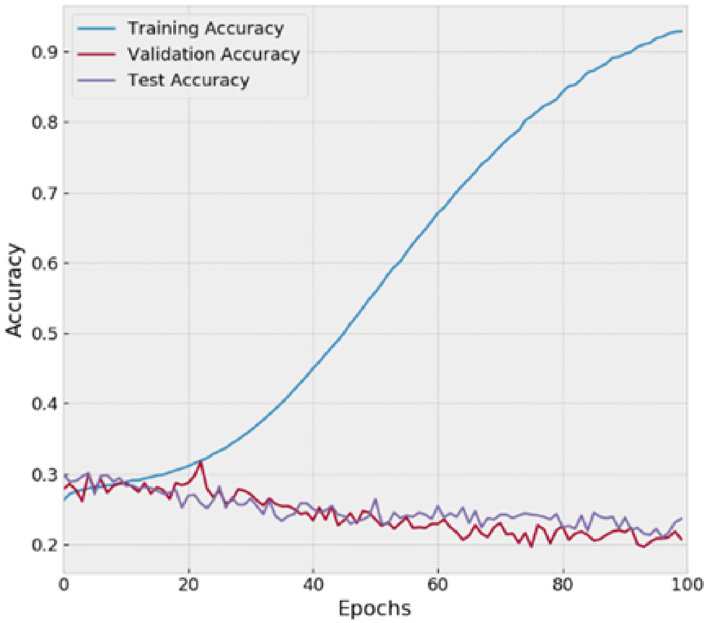
SCA Traces

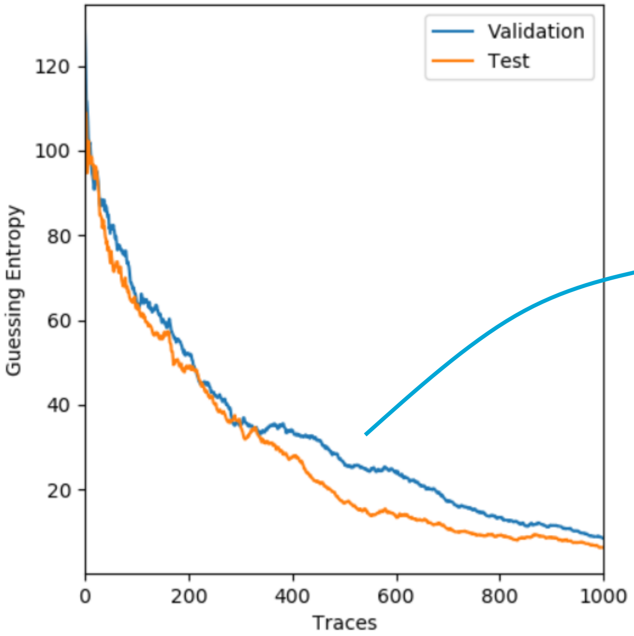


Predictions ➡ Key Rank (GE, SR)

What can we learn here?

Results on Masked AES (MLP)
Attacking 1 key byte with HW model





Predictions
or
Output Class Probabilities

Output Class Probabilities

Example: HW model of 1 byte on AES (S-box output)

Classes / Labels

Test Traces

$P =$

HW = 0	HW = 1	HW = 2	HW = 3	HW = 4	HW = 5	HW = 6	HW = 7	HW = 8
$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$p_{0,3}$	$p_{0,4}$	$p_{0,5}$	$p_{0,6}$	$p_{0,7}$	$p_{0,8}$
$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$	$p_{1,6}$	$p_{1,7}$	$p_{1,8}$
$p_{2,0}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$	$p_{2,6}$	$p_{2,7}$	$p_{2,8}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$p_{N-1,0}$	$p_{N-1,1}$	$p_{N-1,2}$	$p_{N-1,3}$	$p_{N-1,4}$	$p_{N-1,5}$	$p_{N-1,6}$	$p_{N-1,7}$	$p_{N-1,8}$

$p_{i,j} =$ probability that trace i contains label (HW) j

$j = S_{box}(pt_i \oplus k_i)$ (leakage or selection function)

TU Delft

Summation: Key Rank

Label according to key guess $\boldsymbol{k}$

Label(0) = Sbox($pt_0 \oplus k$) = 3
Label(1) = Sbox($pt_1 \oplus k$) = 6
Label(2) = Sbox($pt_2 \oplus k$) = 2
...
Label(N-1) = Sbox($pt_{N-1} \oplus k$) = 4

Classes / Labels

Test Traces

$P(\boldsymbol{k}) =$

HW = 0	HW = 1	HW = 2	HW = 3	HW = 4	HW = 5	HW = 6	HW = 7	HW = 8
$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$\boldsymbol{p_{0,3}}$	$p_{0,4}$	$p_{0,5}$	$p_{0,6}$	$p_{0,7}$	$p_{0,8}$
$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$	$\boldsymbol{p_{1,6}}$	$p_{1,7}$	$p_{1,8}$
$p_{2,0}$	$p_{2,1}$	$\boldsymbol{p_{2,2}}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$	$p_{2,6}$	$p_{2,7}$	$p_{2,8}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$p_{N-1,0}$	$p_{N-1,1}$	$p_{N-1,2}$	$p_{N-1,3}$	$\boldsymbol{p_{N-1,4}}$	$p_{N-1,5}$	$p_{N-1,6}$	$p_{N-1,7}$	$p_{N-1,8}$

$$P(\boldsymbol{k}) = \sum_{i=0}^{N-1} \log p_{i,j} = \log p_{0,3} + \log p_{1,6} + \log p_{2,2} + \cdots + \log p_{N-1,4}$$

Recovered key: $\underset{k}{\operatorname{argmax}}[P(0), P(1), \dots, P(255)]$

Summation: Key Rank

Test Accuracy is **100%**

Classes / Labels

Test Traces

$P(\boldsymbol{k}) =$

HW = 0	HW = 1	HW = 2	HW = 3	HW = 4	HW = 5	HW = 6	HW = 7	HW = 8
0,01	0,02	0,08	**0,40**	0,20	0,25	0,01	0,02	0,01
0,02	0,01	0,06	0,14	0,15	0,20	**0,35**	0,02	0,05
0,01	0,01	**0,53**	0,08	0,22	0,10	0,02	0,02	0,01
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0,01	0,01	0,02	0,25	**0,40**	0,08	0,20	0,02	0,01

$$P(\boldsymbol{k}) = \sum_{i=0}^{N-1} \log p_{i,j} = \log \mathbf{0,40} + \log \mathbf{0,35} + \log \mathbf{0,53} + \cdots + \log \mathbf{0,40}$$

Always the **highest** value per row

Summation: Key Rank

Test Accuracy is **27%**

Classes / Labels

$P(\boldsymbol{k}) =$

HW = 0	HW = 1	HW = 2	HW = 3	HW = 4	HW = 5	HW = 6	HW = 7	HW = 8
0,01	0,02	$\mathbf{0,25}$	0,08	0,20	$\mathbf{0,40}$	0,01	0,02	0,01
0,02	0,01	0,06	0,14	$\mathbf{0,35}$	0,20	0,02	$\mathbf{0,15}$	0,05
0,01	0,01	0,08	$\mathbf{0,53}$	0,22	0,10	0,02	0,02	0,01
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0,01	0,01	0,02	$\mathbf{0,40}$	0,08	$\mathbf{0,25}$	0,20	0,02	0,01

Test Traces

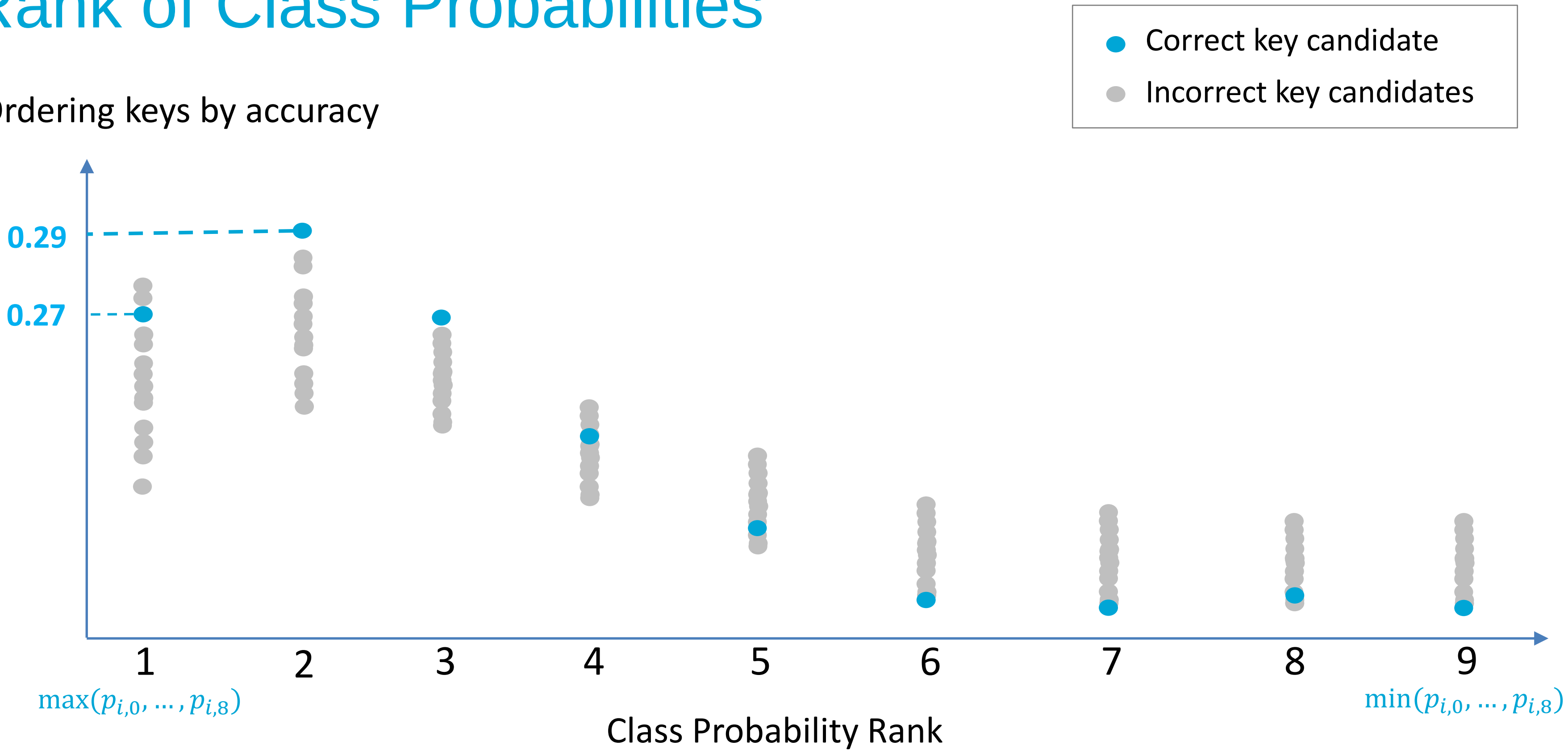
$$P(\boldsymbol{k}) = \sum_{i=0}^{N-1} \log p_{i,j} = \log \mathbf{0,25} + \log \mathbf{0,15} + \log \mathbf{0,53} + \cdots + \log \mathbf{0,25}$$

NOT Always the **highest** value per row

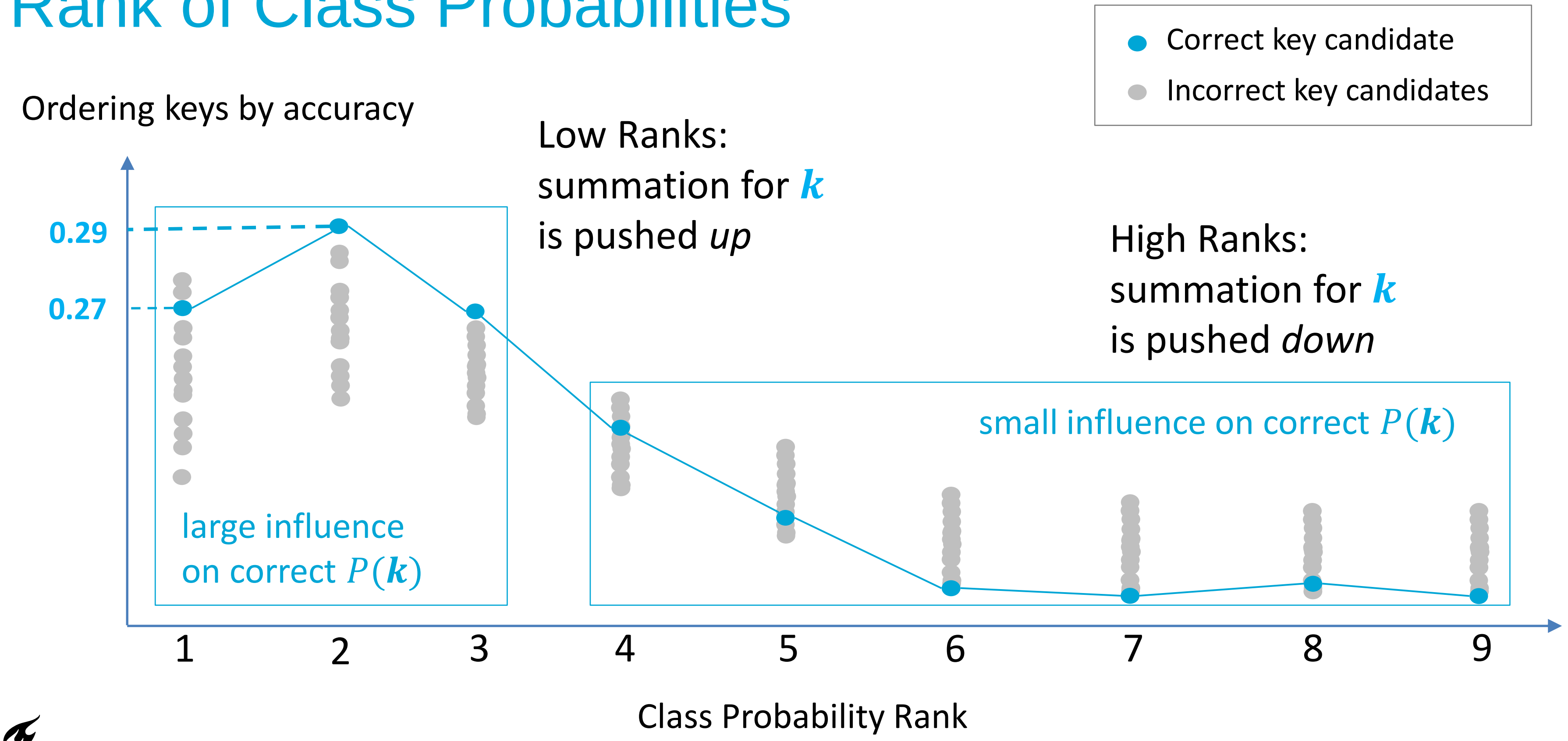
TU Delft

Rank of Class Probabilities

Ordering keys by accuracy

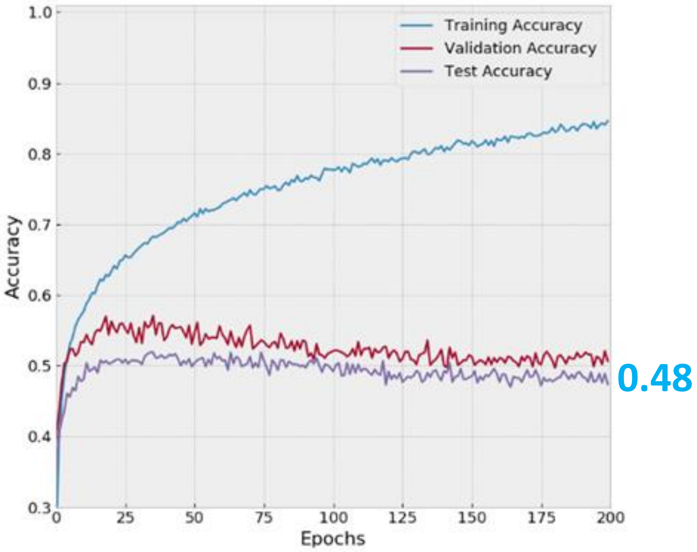


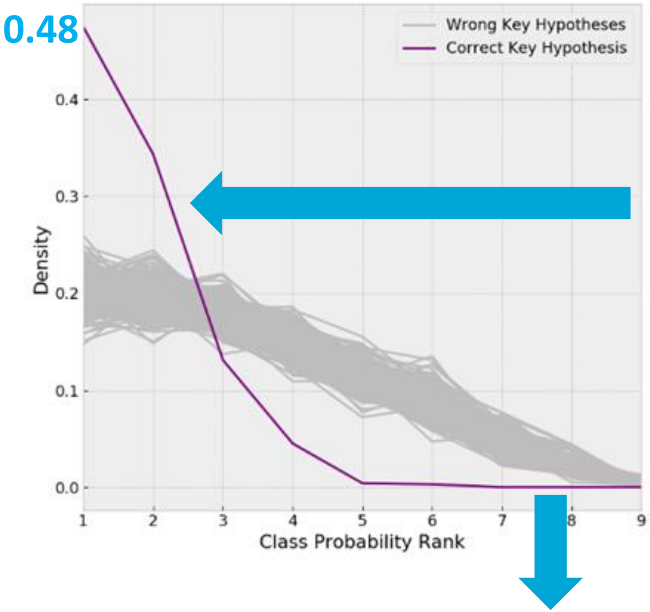
Rank of Class Probabilities



Results on Leaky AES (MLP)

Attacking 1 key byte with HW model

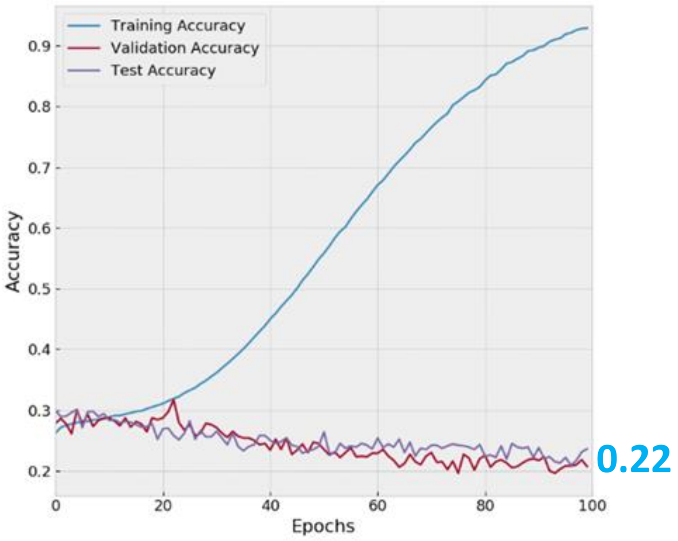


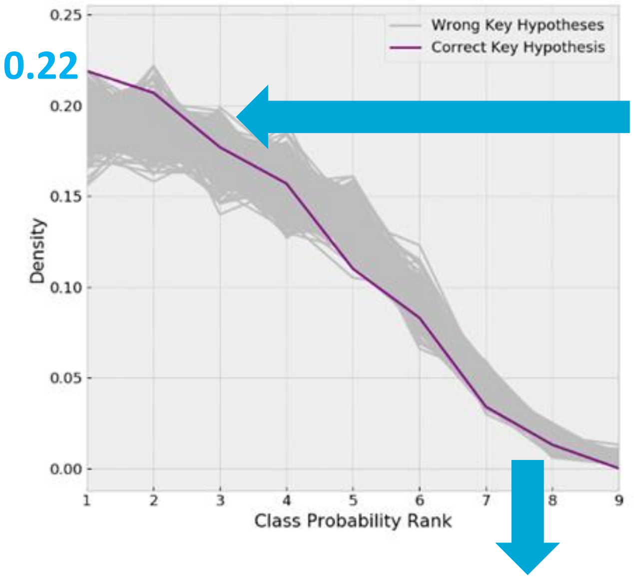


Output class probabilities are pushed towards ranks 1 and 2

No test traces with high ranked probabilities

Results on Masked AES (MLP)
Attacking 1 key byte with HW model





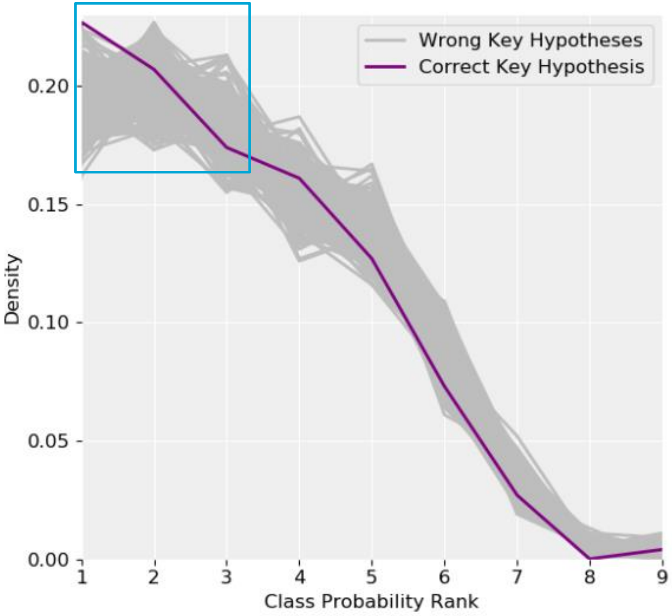
Output class probabilities are pushed towards ranks 1 and 2

Successful key recovery

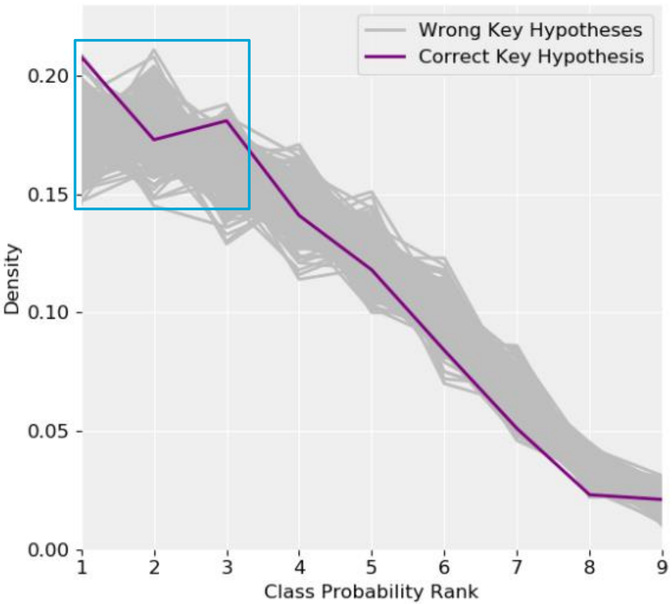
Few test traces with high ranked probabilities

TU Delft

Two CNN models on masked AES



CNN with 4 hidden layers



CNN with 3 hidden layers

TU Delft

Common story

- Deep learning analysis requires a large amount of hyperparameters experiments:

$$h_{best} = \underset{m \in M}{\operatorname{argmin}} Loss(\lambda_m, t_{train}, t_{val})$$

Select a proper metric

$$h_{best} = \underset{m \in M}{\operatorname{argmin}} GE(\lambda_m, t_{train}, t_{val})$$

- From multiple models, we elect a best one. Why not benefit from multiple models instead of a best single model?

TUDelft

Ensembles

- Boosting
- Stacking
- Bootstrap Aggregating (Bagging)

$$P(\boldsymbol{k}) = \sum_{m=0}^{M-1} \sum_{i=0}^{N-1} \log p_{i,j,m}$$

Select best models based on GE: $M_{best} < M$

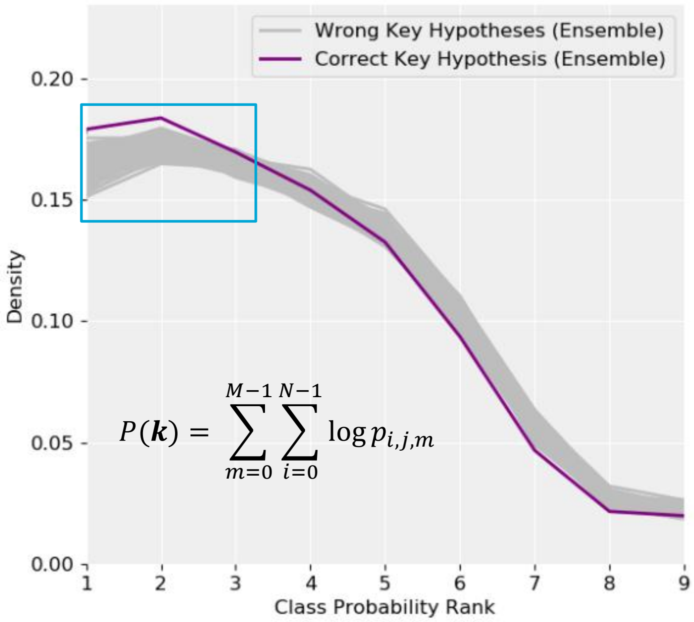
$$\boxed{\underset{m}{\operatorname{argmin}}\, GE(\lambda_m, t_{train}, t_{val})}$$

hyperparameters

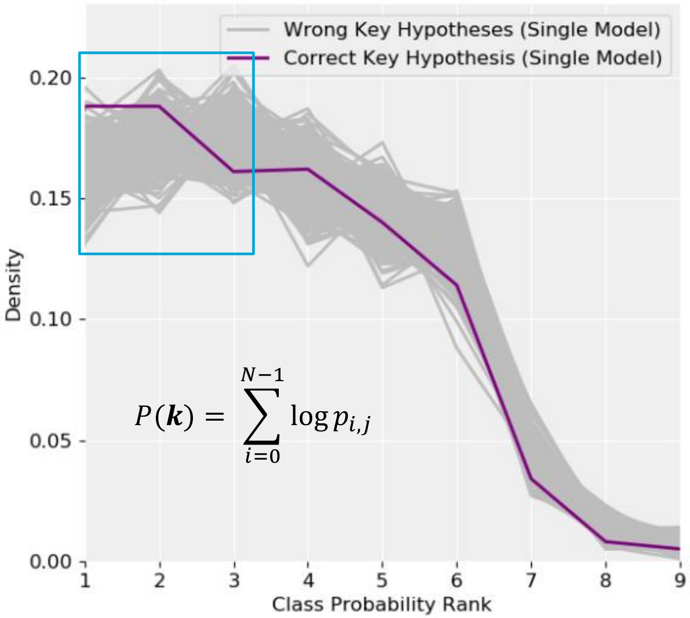
train traces

validation traces

Ensembles



Ensemble ($M_{best} = 10, M = 50$)



Single Best Model = $\underset{m}{\operatorname{argmin}}\, GE(\lambda_m, t_{train}, t_{val})$

Datasets

Dataset	Training	Validation	Test	Features	Countermeasures
Pinata SW AES	6,000 (fixed key)	1,000	1,000	400	No
DPAv4	34,000 (fixed key)	1,000	1,000	2,000	RSM
ASCAD	200,000 (random keys)	500	500	1,400	Masking
CHES CTF 2018	43,000 (fixed key)	1,000	1,000	2,000	Masking

Range of Hyperparameters

MLP

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini-batch	100	1000	100
Dense Layers	2	8	1
Neurons	100	1000	100
Activation Function	Tanh, ReLU, ELU or SELU		

CNN

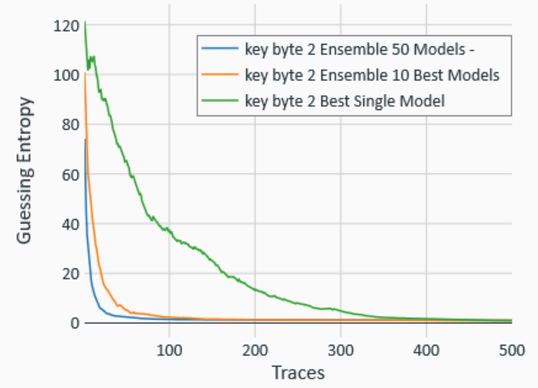
Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini-batch	100	1000	100
Convolution Layers (i)	1	2	1
Filters	8*i	32*i	4
Kernel Size	10	20	2
Stride	5	10	1
Dense Layers	2	8	1
Neurons	100	1000	100
Activation Function	Tanh, ReLU, ELU or SELU		

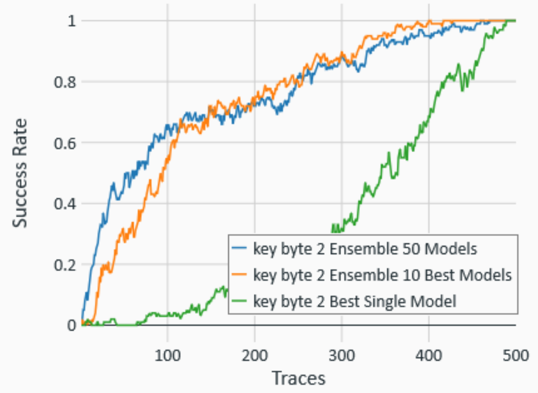
*optimal ranges based on literature

TU Delft

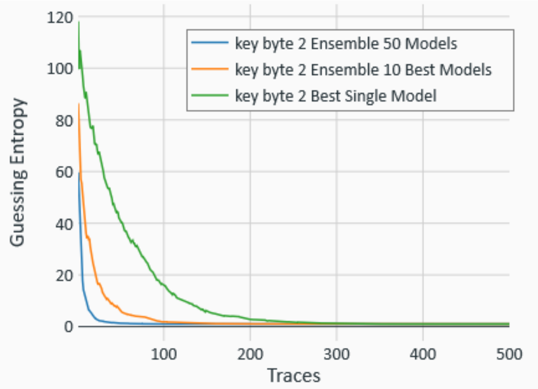
Results on ASCAD (Hamming Weight)

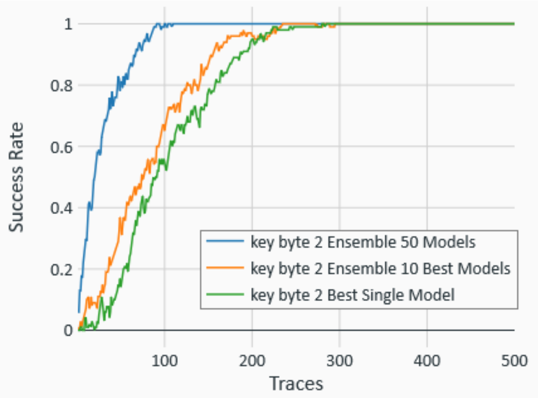
MLP





CNN

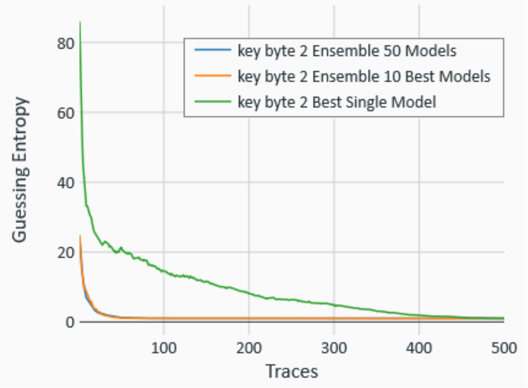


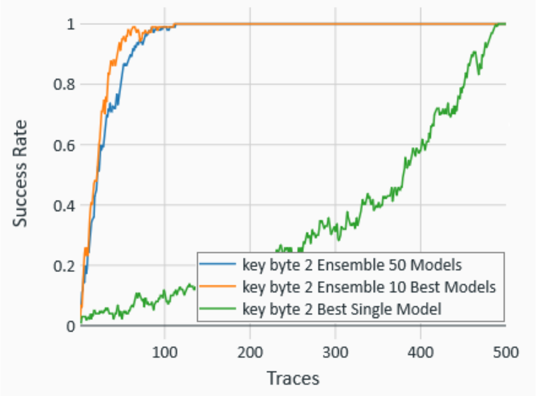


TU Delft

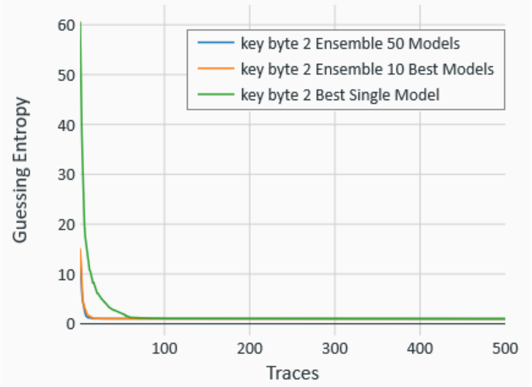
Results on ASCAD (Identity)

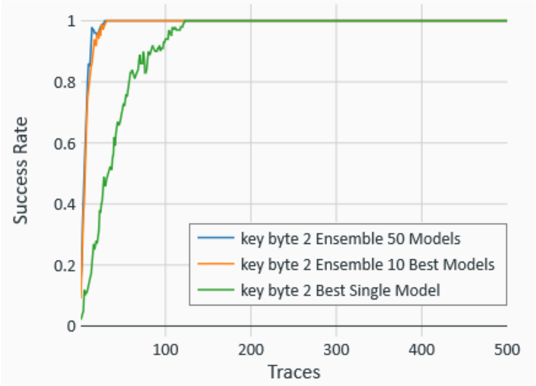
MLP





CNN





TUDelft

Conclusions

- Output class probabilities are a valid distinguisher for side-channel analysis.
- Output class probabilities are sensitive to small changes in hyperparameteres: ensembles remove the effect of small variations, improving generalization results.
- Ensembles *do not* replace hyperparameters search. Ensembles *relax* the fine tuning of hyperparameters: GE or SR of ensemble tends to be superior to GE or SR of a single best model.
- Ensembles *do not* improve learnability: they *improve* what single models already learn.
- Limited amount of models can be enough to build a strong ensemble.

As future works:

- Explore another ensemble methods (e.g., stacking) .
- Verify how ensembles work in combination with other regularization methods and other metrics (SR, MI).
- Formalize the density distribution of output class probabilities (a new metric).

Thank you!

- Our code is available at: **https://github.com/AISyLab/EnsembleSCA**

TU Delft