Deep Learning-based Side-channel Analysis

Exploiting Vulnerabilities and Explaining Neural Network Decisions

Trevor Yap Hong Eng

Supervisor: Prof.Thomas Peyrin
Co-supervisor: Dr.Shivam Bhasin

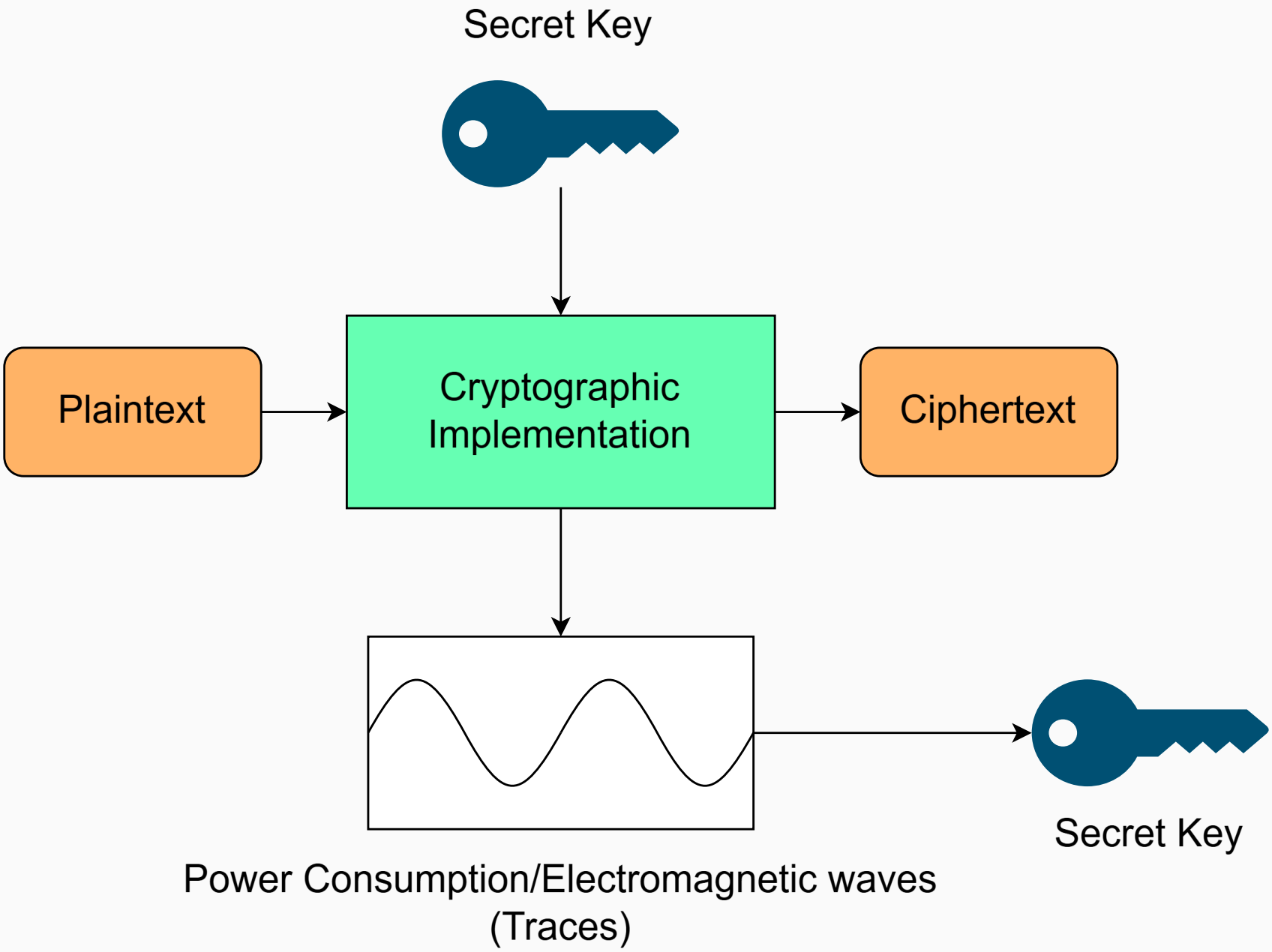
March 25, 2025

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

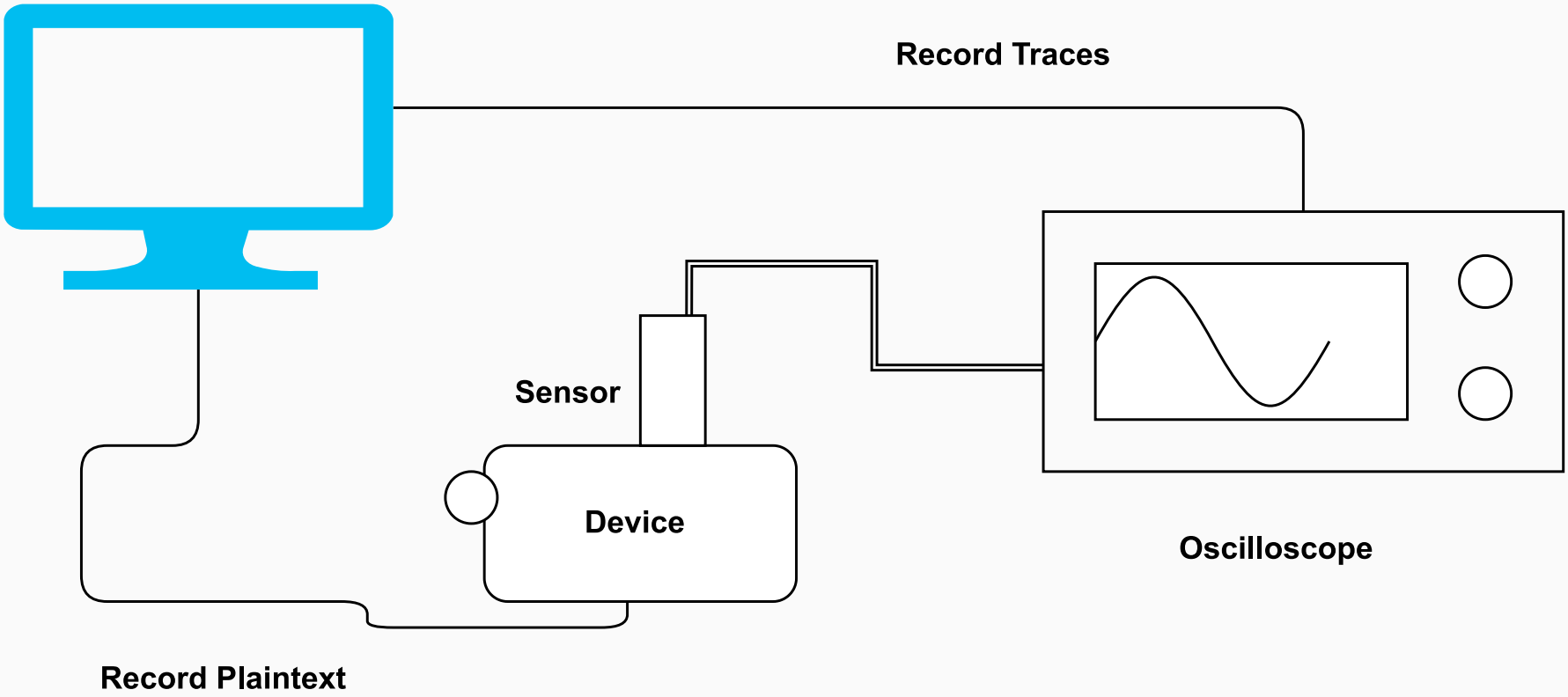
Table of contents

Table of contents

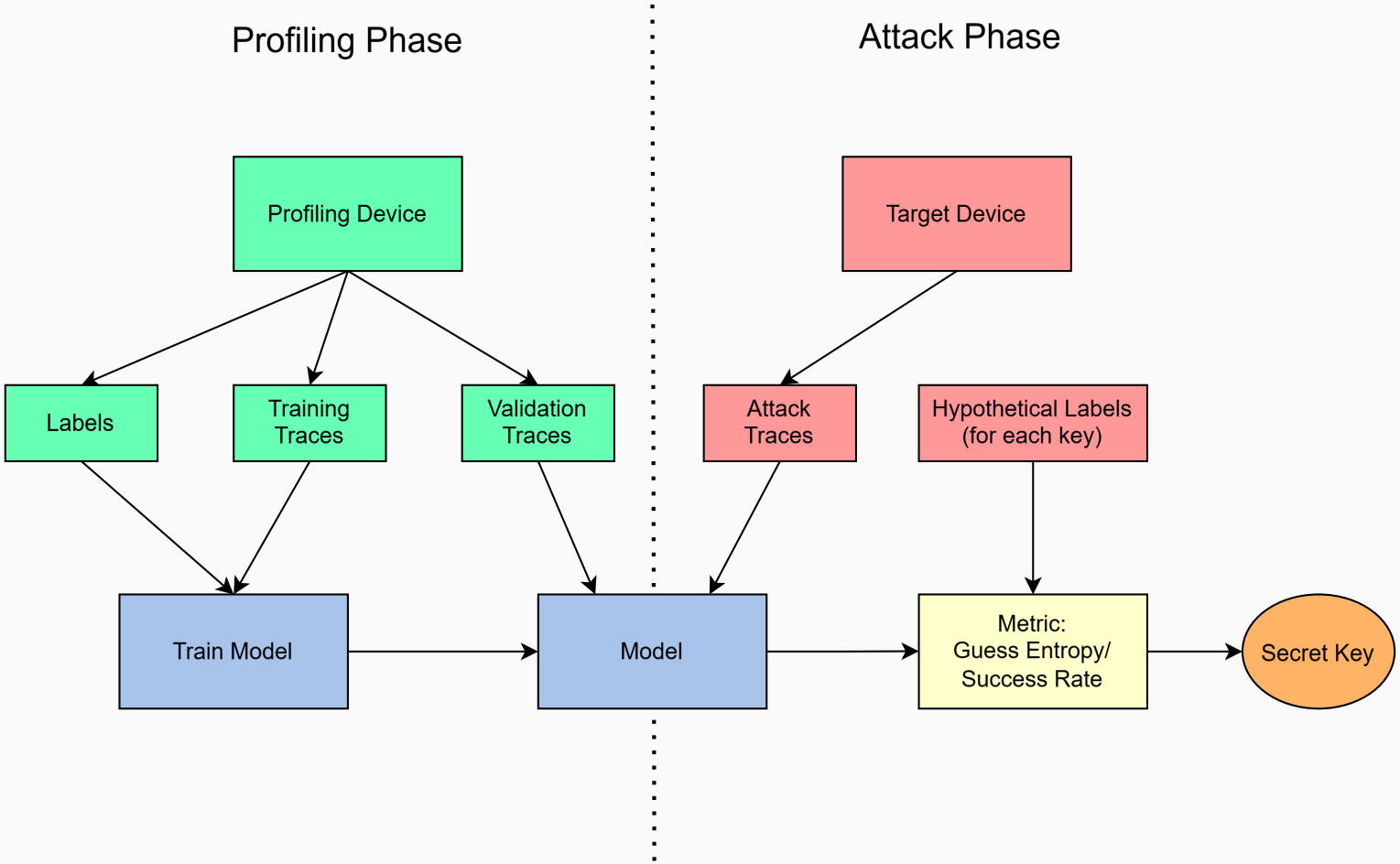
Overview of side channel analysis



Trace acquisition set up



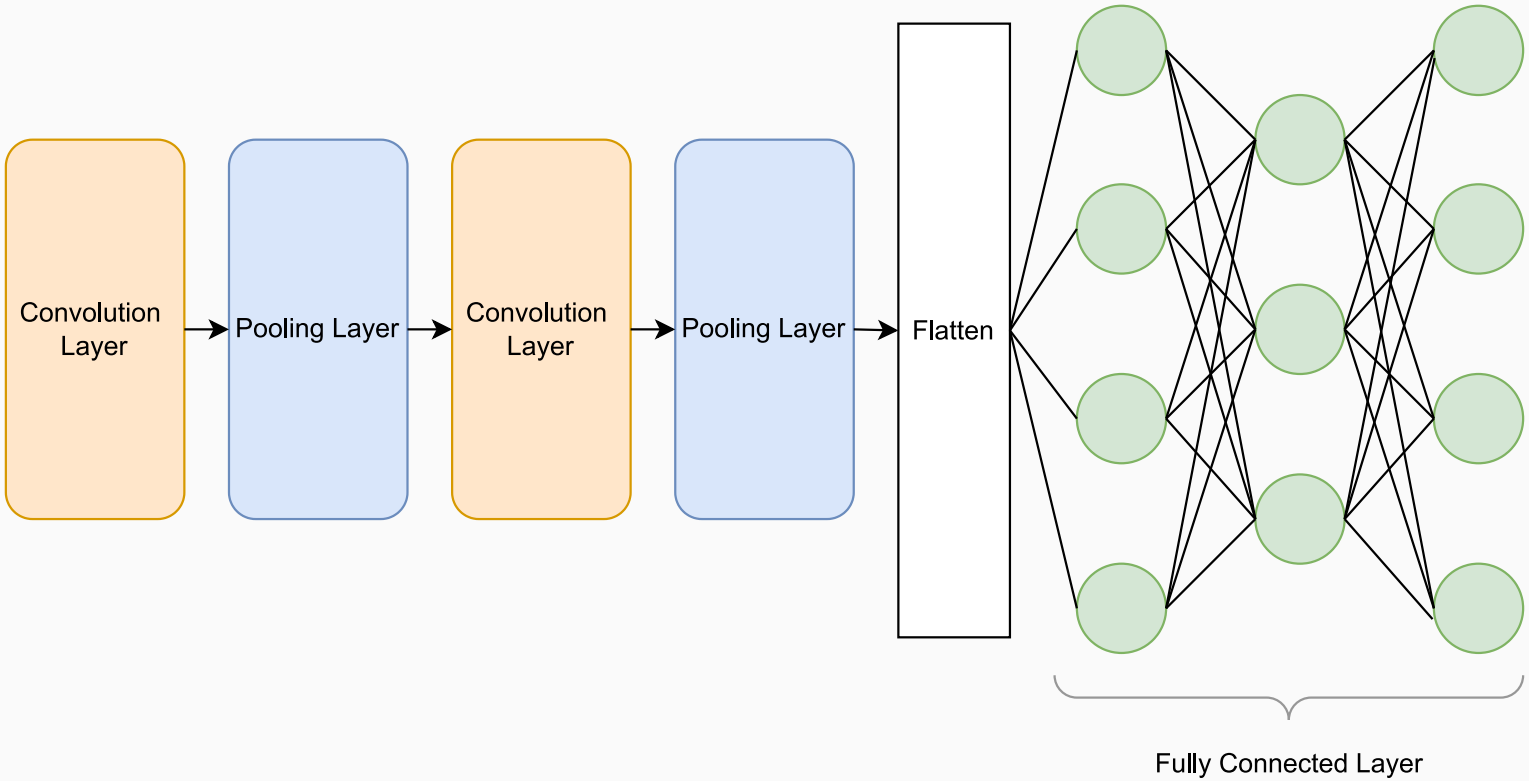
Profiling Attack



We denote $GE = 0$ if the attack is successful.

We define NTGE to be the least number of attack traces required for $GE = 0$.

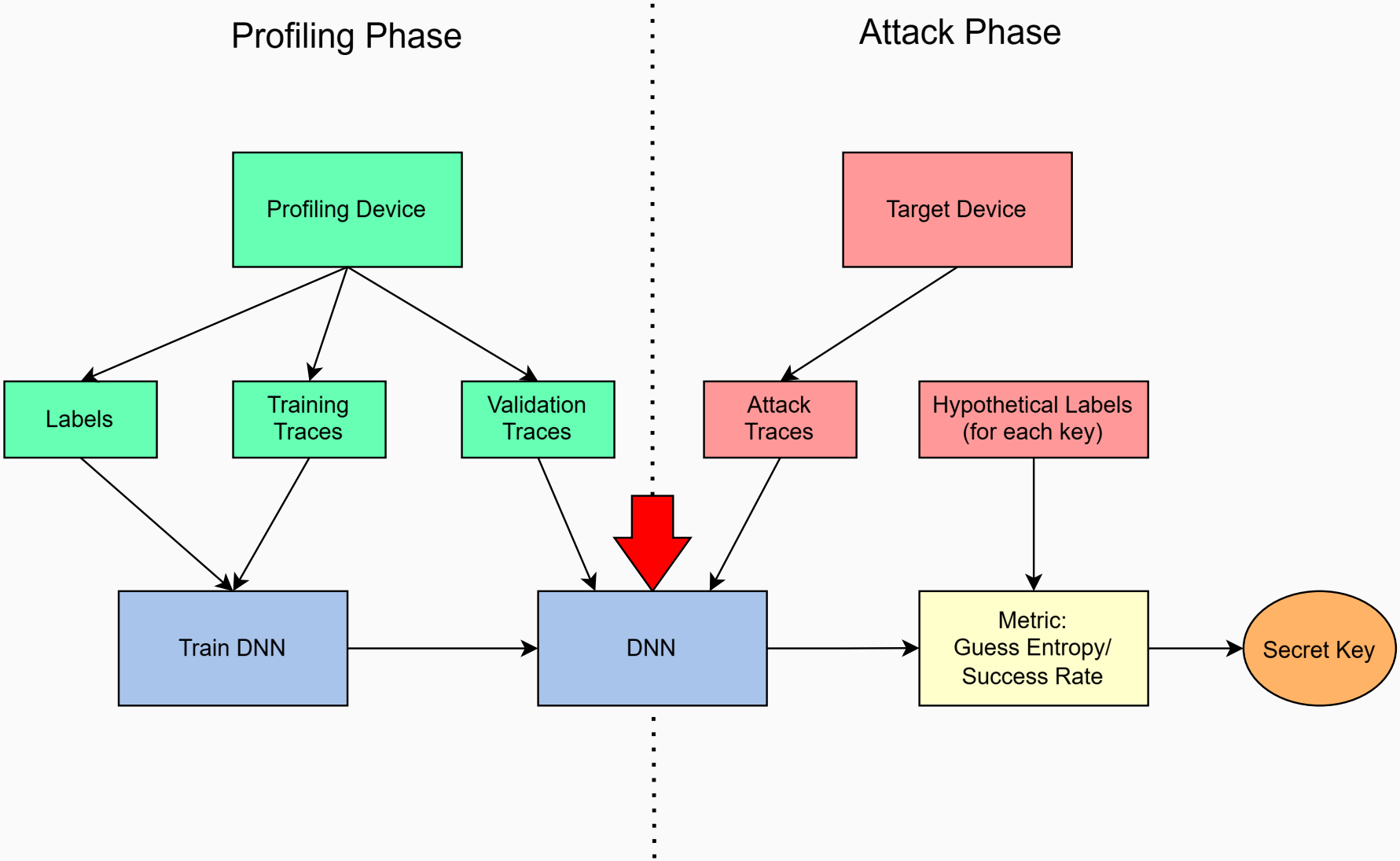
Deep Neural Network (DNN)



DNNs are layers of weights trained for various tasks.

Common DNNs are like Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs)

Profiling attack with DNN



Advantage of DNN

Classical SCA (i.e. Template Attack)	**DNN-based attack**
More traces to attack	Significantly lesser traces to attack
Preprocessing to remove any desynchronous in the traces	No preprocessing needed
Preprocessing to obtain Points of Interest (PoIs)	No preprocessing needed

Timeline of DLSCA

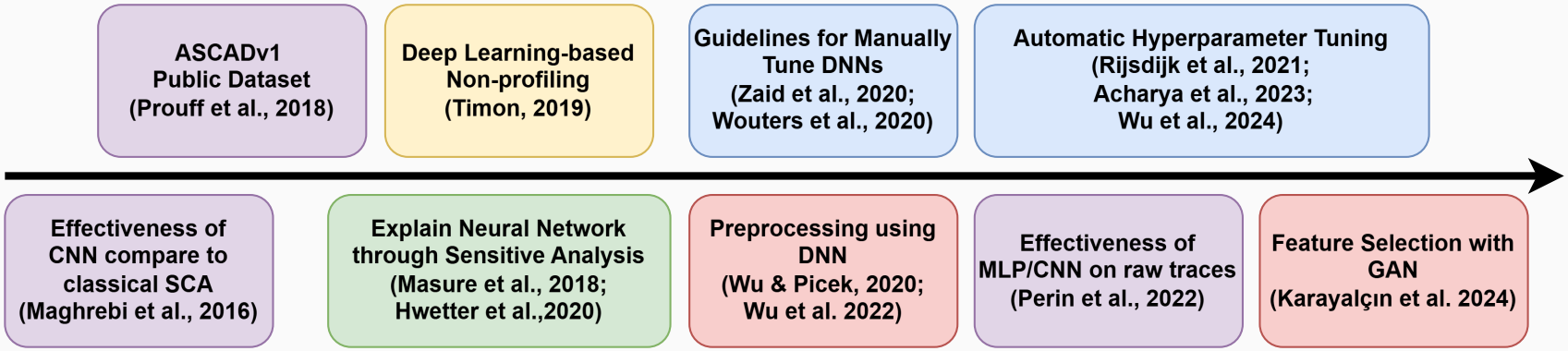
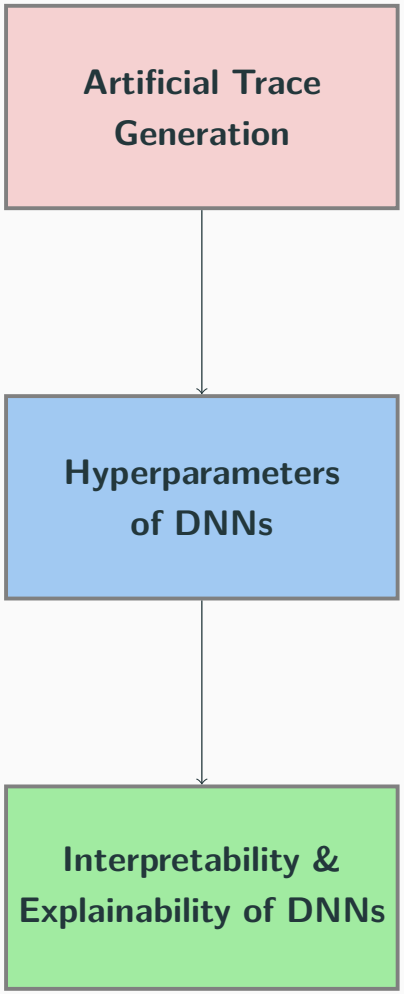


Figure 1: Timeline of DLSCA (Non-Exhaustive).

- There has been a notable rise in the number of published papers. Currently, there are over 250 papers being published.

Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

Chapter 5: Apply Bayesian Optimization Hyperband (BOHB) to efficiently allocate resources when searching for well-performing DNNs automatically. (SAC'24)

Chapter 6: Employ transparent models that provide SAT formulae to facilitate understanding of the network and pinpoint the Points of Interest used by the DNN. (TCHES'23)

Chapter 7: Develop an explainability technique for generic DNNs to locate the Points of Interest used by these networks. (Indocrypt'24)

Table of contents

Motivation: More Data!!!

- In a practical setting, there might be a **limitation on the number of traces** that can be **collected** by the adversary. (e.g. a typical use of credit cards is around 50000 times.)
- Performance of SCA could be affected.
- Therefore, there is a need for **more data/traces**.
- But **manually creating artificial traces** can be quite **complicated and tedious** as it needs to capture the leakage information and its characteristics properly.

Related Works

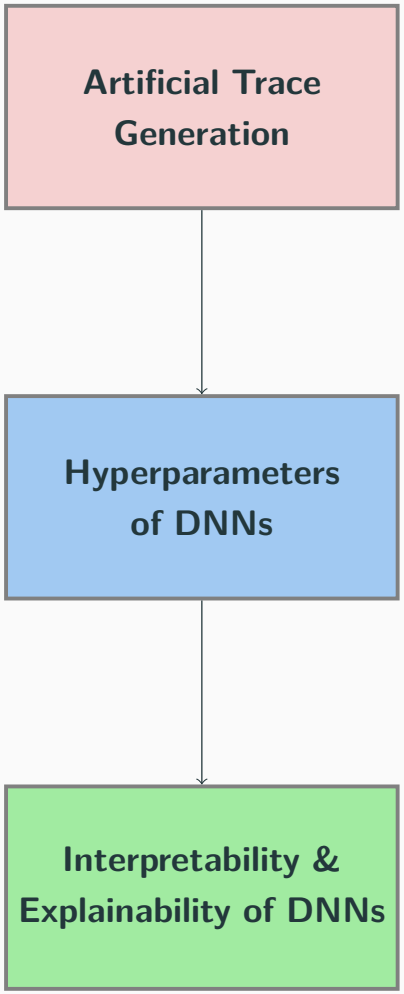
Manually creating data:

- Synthetic Minority Over-sampling Technique (SMOTE) (Picek et al., 2019): apply data augmentation to deal with data imbalance.
- Manually adding jitters into the original traces (Cagli, Dumas, and Prouff, 2017).

Automatically creating data:

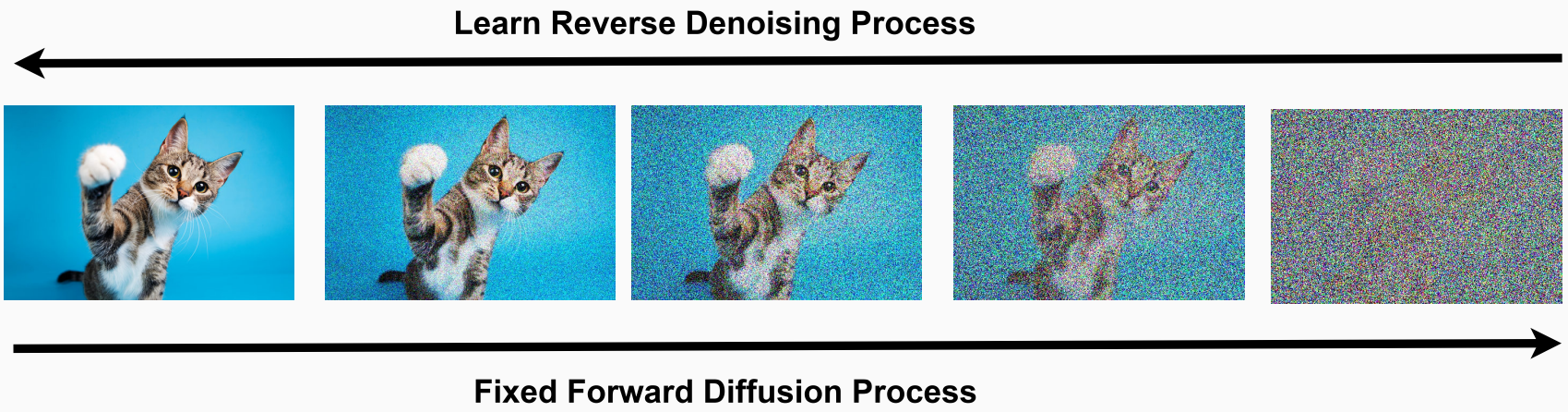
- Use of Conditional Generative Adversarial Network (CGAN) (Wang et al., 2020).
- Improvement of CGAN with Siamese network (Mukhtar et al., 2022).

Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Denoising Diffusion Probabilistic Model (DDPM)



General Idea:

- **Add the noise in the forward direction** at each time step according to a fix schedule until is data is simply noise (Theoretically, we can actual sample each for any time step directly from the original data).
- Use a **Deep Neural Network (DNN) to learn the noise** added at **each time step**. This will learn the reverse process.
- Generate new data from noise using the trained DNN.

Known Mask Setting: Diffusion Model Framework

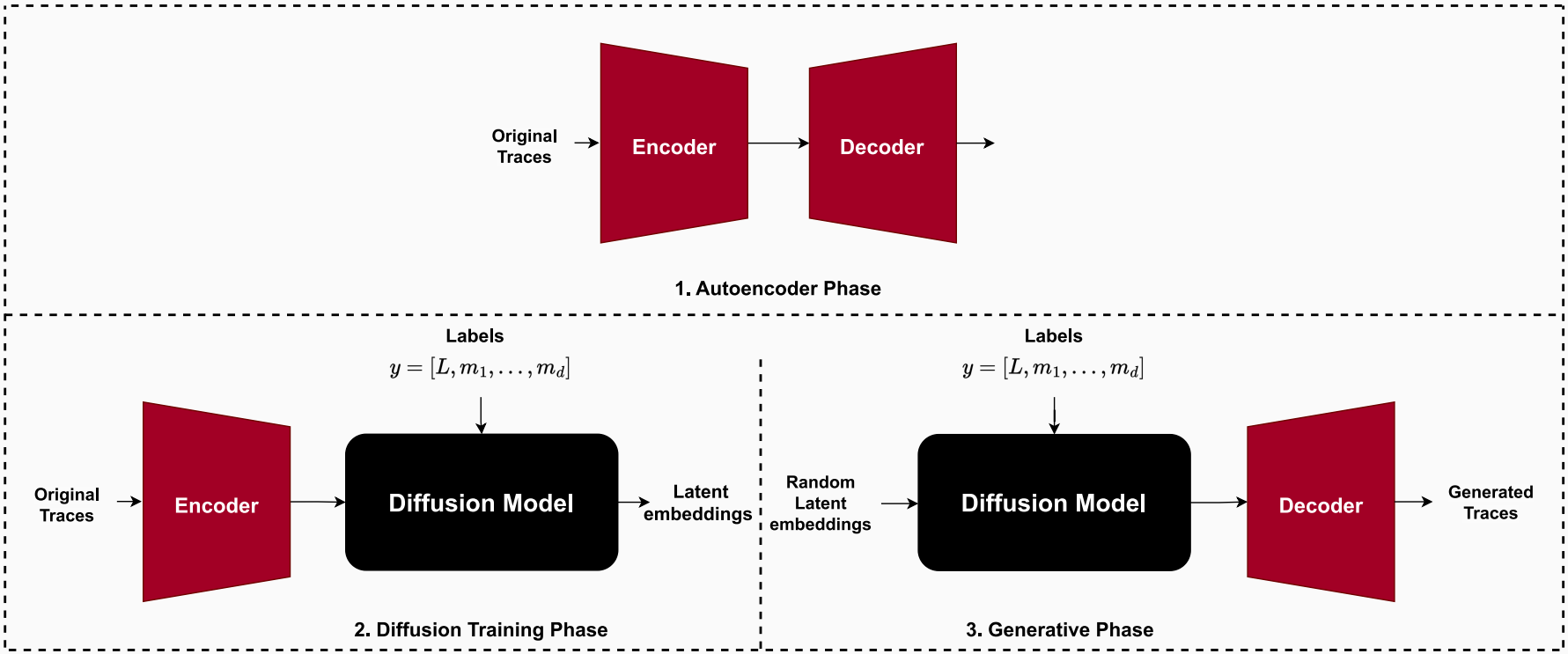
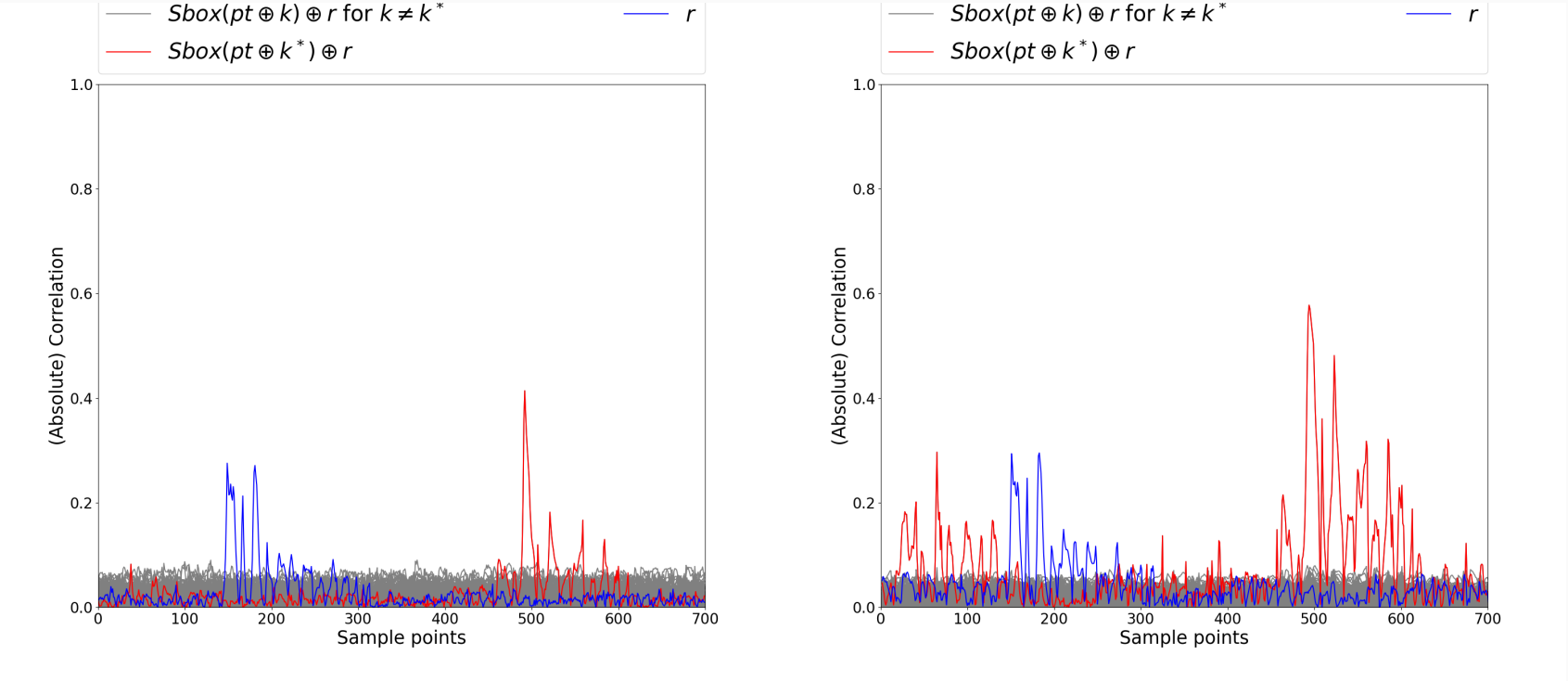


Figure 2: Framework for using diffusion model in CPA. We define Z to be the corresponding mask data (e.g., $L = Z \oplus m_1 \oplus \cdots \oplus m_d$).

Results: ASCADf



(a) Original Traces.

(b) Generated Traces.

We observed that the generated traces by diffusion models **preserve the leakage characteristic** from the original traces in known mask setting.

Unknown Mask Setting: Framework for Profiling Setting

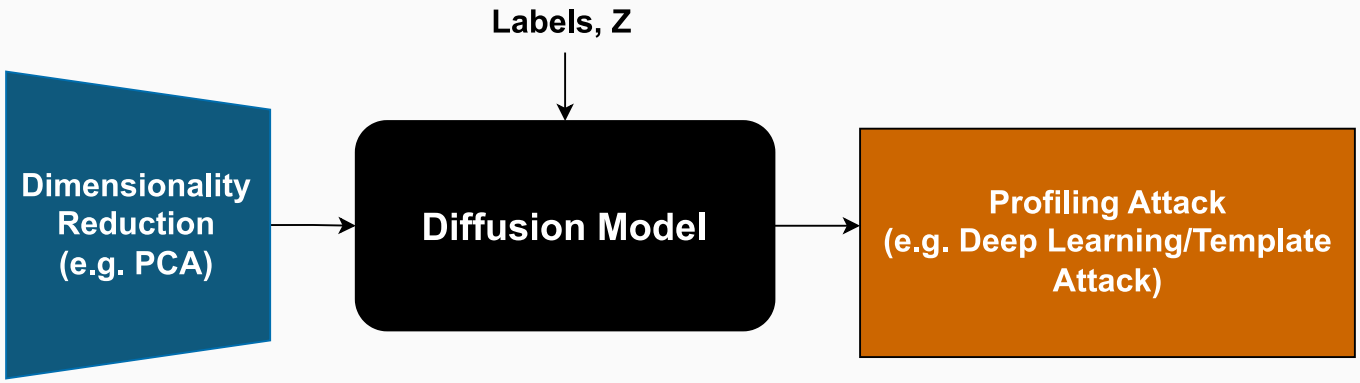


Figure 4: Framework for using diffusion model in Profiling attack. Z is defined to be the hypothesis sensitive variable (e.g., $Z = Sbox(pt \oplus k^*)$)

Results: ASCADf desynchronization 50

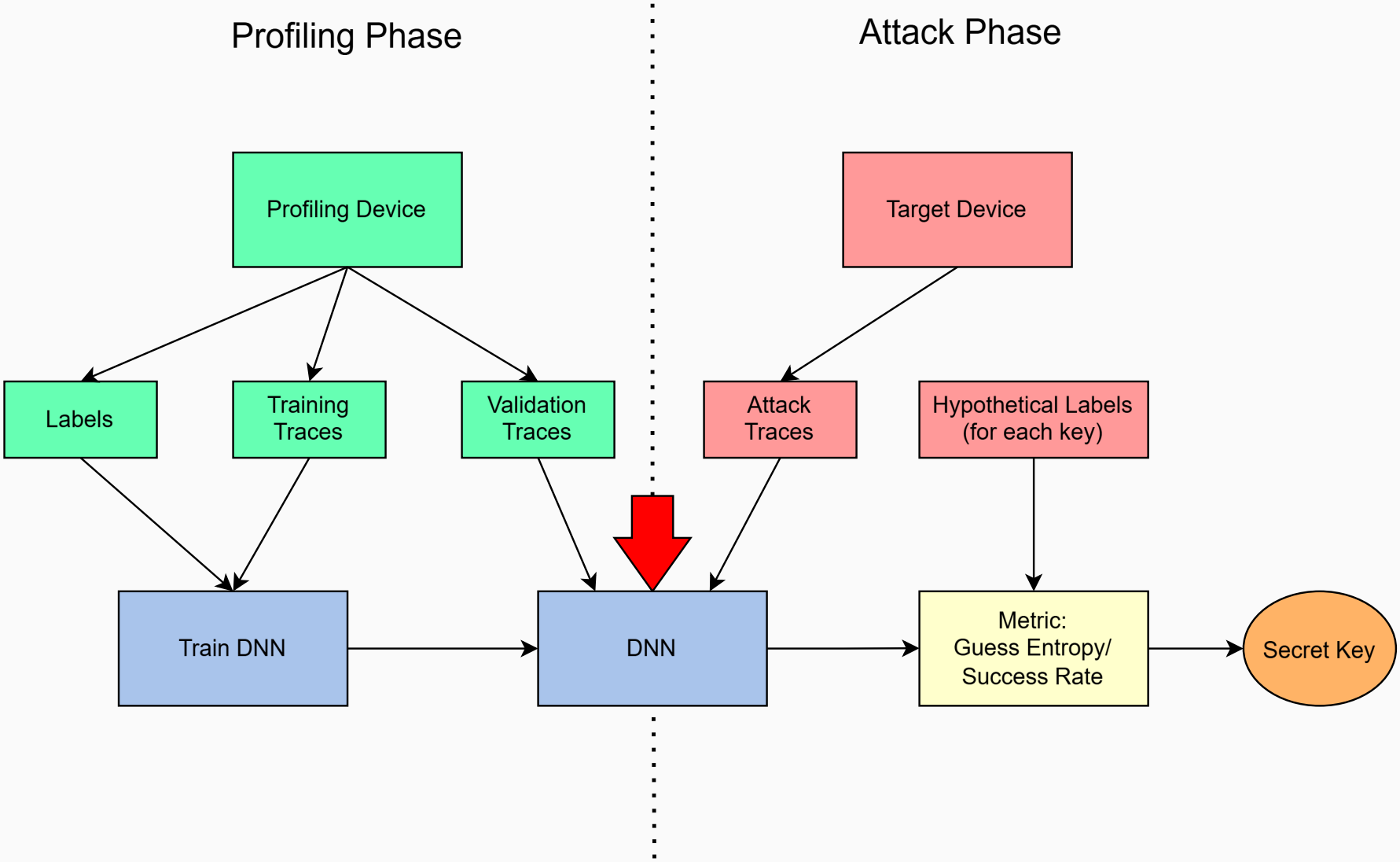
	Original	Downsampled	Downsampled+Generated Latent
NTGE	$9,606$	$9,017$	$5,730$
Dataset Size	$45,000$	$35,584$	$71,168$

Table 1: *NTGE* for ASCADf_desync50 when applying TA in the various setting.

We also validated the **effectiveness of diffusion model** in an unknown mask setting that the **artificial traces can improve performance** of Template Attack especially when the underlying primitive is **protected by hiding countermeasures** (desynchronization).

Table of contents

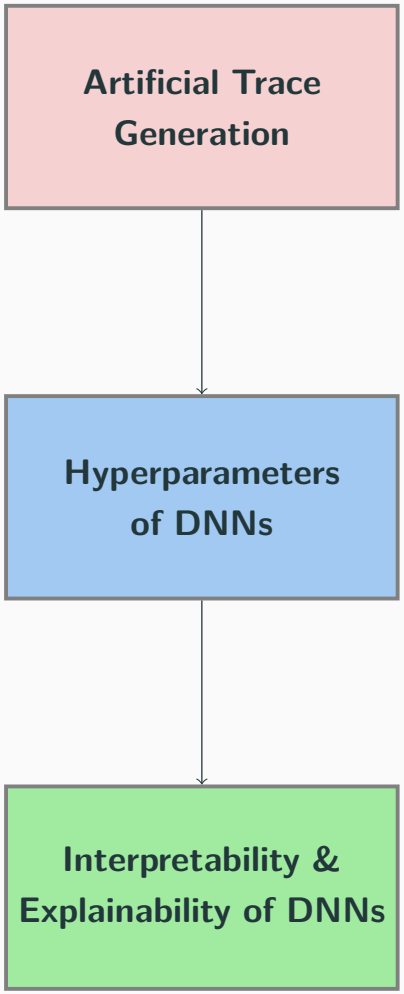
Profiling attack with DNN



Motivation for Hyperparameter tuning

- DNNs were shown to **outperform classical SCA** even in the presence of countermeasures.
- Introduced a **large number of hyperparameters** to tune (e.g., the number of layers, kernel size, type of activation functions, etc.) compared to other machine learning or classical SCA.
- Previous work have pointed out that the **performance** of DNNs is **greatly influenced by their hyperparameters** (Maghrebi, Portigliatti, and Prouff, 2016). This pushes for the need for methodologies to find good hyperparameters in the domain of SCA.

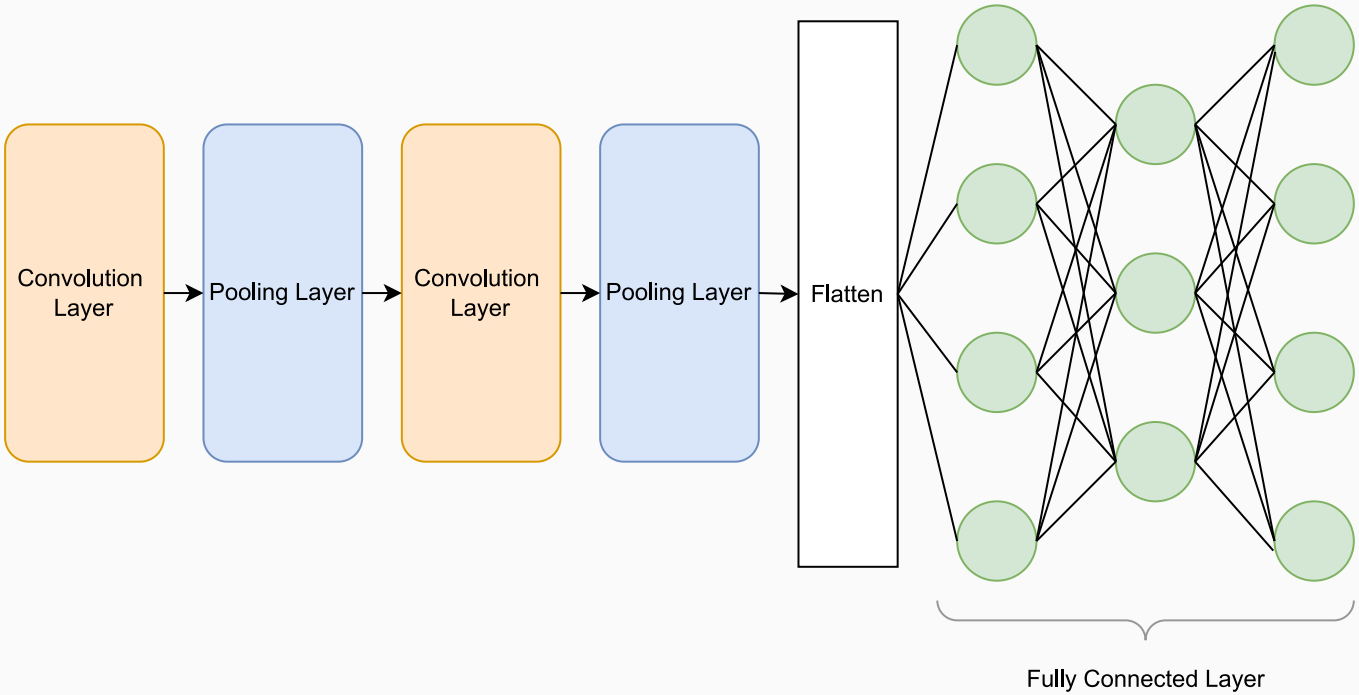
Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

What are Loss functions & Deep features?



- Loss function is the objective that is optimized when training the neural network.
- Deep features are the features/embeddings acquired from the output of each intermediate layer within the DNN.

Deep Features Loss Functions

Baseline Loss Function

- Common loss functions in SCA: Categorical Cross Entropy (CCE) and Focal Loss Ratio (FLR).
- These baseline loss functions consider only the inter-class distances.

Deep Features Loss Functions

- SoftNN compares both **inter-class and intra-class distances** between the data points.
- Center loss updates the center of a deep feature for each class to minimize the **intra-class distances** and leave the inter-class to the baseline loss function.

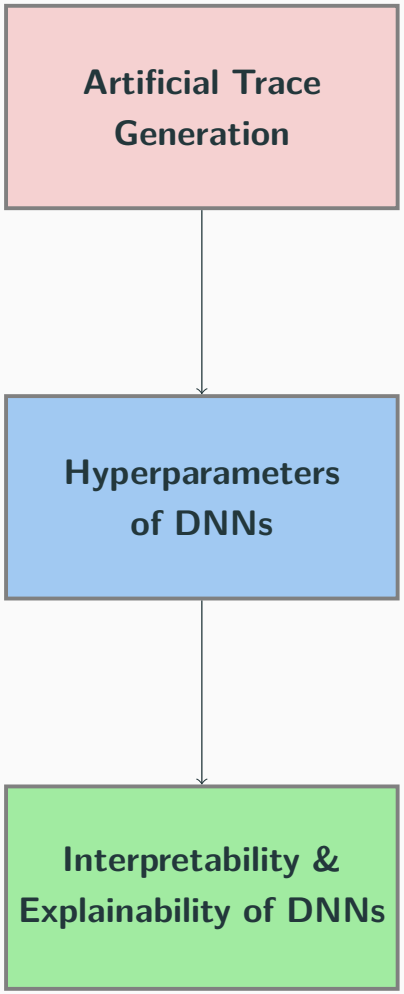
Experiment Results: ASCADf

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	1461/1783.5	1531/1963	1136/1802	411/603	505/635	488/789.5
CNN (HW)	1787/ >2000	1614/1970.5	1610/ >2000	556/772	570/831.5	531/790
MLP (ID)	245/474	294/ >2000	349/544.5	328/612	248/386.5	236/344
CNN (ID)	457/962.5	633/994	403/ >2000	470/ >2000	377/716	297/727.5

Table 2: *NTGE* on the ASCADf (best/median).

- Employing Deep Features Loss Functions can improve the performance of the DLSCA attack.

Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

Chapter 5: Apply Bayesian Optimization Hyperband (BOHB) to efficiently allocate resources when searching for well-performing DNNs automatically. (SAC'24)

Related Works

Manual Hyperparameter Tuning:

- (Zaid et al., 2020) and (Wouters et al., 2020) provided guidelines and offer a more precise methodology that helps to generate smaller and well-performing DNNs manually.

Automatic Hyperparameter Tuning:

Techniques	Reference	Time Taken
Bayesian Optimization	(Wu, Perin, and Picek, 2024)	$\sim$ 10 hrs
Reinforcement Learning	(Rijsdijk et al., 2021)	$\sim$ 4 days
Evolutionary Algorithm	(Acharya, Ganji, and Forte, 2023)	$\sim$ 2 days

- Most techniques are slow and could run for days.

Motivation

- Due to the large number of IT products to be evaluated, evaluating the security of these products in evaluation labs becomes very **time-sensitive.**
- **Resources such as time are valuable assets** to determine a device's security.
- An evaluator will naturally **set a budget for any resources** like the time needed to quantify the security of the primitive tested.

*Are there automated tools available can produce comparable results while **allocating resources more efficiently**?*

Bayesian Optimization HyperBand (BOHB)

Bayesian Optimization HyperBand (BOHB) is a multifidelity optimization method, which allow speed up in the optimization process by:

- **allocating more resources to promising configurations**
- **stopping evaluations of poorly performing ones early**.

Budget

- In evaluation lab, they look into the **time taken like manpower or server hours.**
- The budget for BOHB could be any resource, like the **amount of time taken** or the number of epochs to train a neural network.
- The number of epochs is essentially the same as the time taken for training. As a DNN is trained with more epochs, the more time it will take to finish training.
- *Therefore, we consider **the number of epochs as the budget** as the parameters for BOHB.*

Prior Objective functions

Prior Objective functions: Val_Loss

It was shown that **minimizing the categorical cross-entropy loss** is equivalent to **maximizing the generalization of the mutual information** between the leakage model and the traces (also known as perceived information).

Prior Objective functions: L_m

L_m as an objective function based on

$$LDD(k, k^*) = \sum_{i=0}^{Q} ||LM(p_i, k^*) - LM(p_i, k))||^2, k \in \mathcal{K},$$

where LM is the leakage model, p_i is the public data and k is the corresponding key. Then is L_m define as the correlation between the key guessing vector $\boldsymbol{G}$ and LDD:

$$L_m(\mathbf{LDD}, \boldsymbol{G}) = corr(argsort(LDD), \boldsymbol{G}).$$

(Wu, Perin, and Picek, 2024) have found that L_m is the best objective function compared to key rank and validation loss.

New Objective functions: ge_{+ntge}

New Objective functions: ge_{+ntge}

$$ge_{+ntge}(\theta) = \begin{cases} NTGE & \text{if } GE = 0, \\ GE + N_a + c & \text{otherwise} \end{cases}$$

where c is a small positive constant and N_a maximum number of attack traces.

- Consider the attack phase into the objective directly (i.e. GE and $NTGE$).

Experimental Results: Compare to Prior Works

	Dataset	Epochs	No. of parameters	$NTGE_{best}$
[3]	ASCADf (ID)	50	16,960	191
	AES_HD	20	3,282	1,050
	ASCADf_desync50 (ID)	50	87,279	244
[7]	ASCADf (ID)	8	15,107	130
	ASCADr (ID)	8	317,408	120
	AES_HD	33	102,757	170
[5]	ASCADf (HW)	10	1,388,457	447
	ASCADf (ID)	10	1,544,776	120
	ASCADr (HW)	10	1,314,009	496
	ASCADr (ID)	50	1,539,320	1,568
	CTF2018 (HW)	50	2,418,085	618
[6]	ASCADf (HW)	50	8,480	1,246
	ASCADf (ID)	50	79,439	202
	ASCADr (HW)	50	15,241	911
	ASCADr (ID)	50	70,492	490
	CTF2018 (HW)	50	33,788	122
	ASCADf_desync50 (HW)	50	516,361	1,592
	ASCADf_desync50 (ID)	50	41,321	443
Ours	ASCADf (HW)	56	845,109	849
	ASCADf (ID)	200	10,596	201
	ASCADr (HW)	34	659,409	879
	ASCADr (ID)	17	1,465,056	1,568
	AES_HD	34	1,725,856	1,030
	CTF2018 (HW)	500	3,645	82
	CTF2018 (ID)	18	25,596	2,523
	ASCADf_desync50 (HW)	500	10,401	2,4698
	ASCADf_desync50 (ID)	500	91,976	1,311

- ge_{+ntge} obtain the best results in 8 out of 14 scenarios.
- We show that BOHB is able to recover the secret key for all the datasets tested.
- Attain the best result for CTF2018 (HW) compare to prior works.
- We successfully recovered the secret key for the CTF2018 (ID) dataset using BOHB, a first in the field.

Table of contents

Lack of transparency of DNN

We have seen that DNN can be used to enhance the performance of key recovery.

- However, DNNs are seen as a **black-box tool**.
- The weights of the DNNs are not human interpretable with respect to the underlying task.
- Unable to tell which areas of the traces that the DNNs are using to recover the secret key.
- The evaluator wants to help the developer **localize and understand where the vulnerability** comes from in order to remove or at least reduce it.

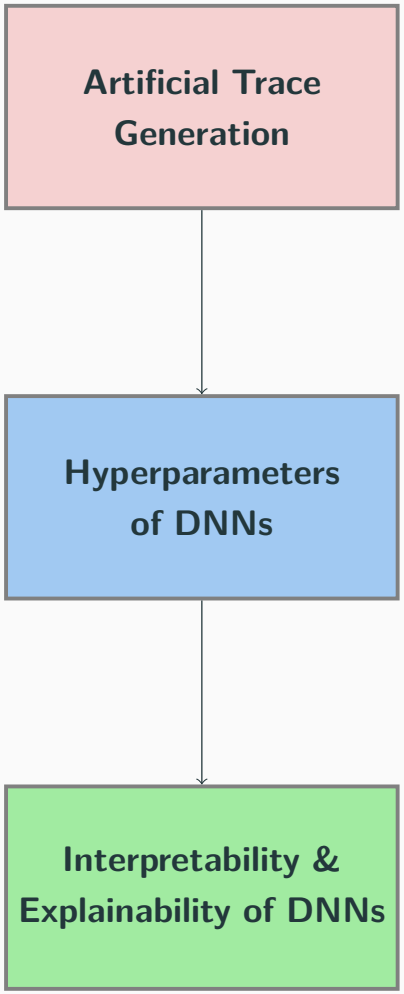
What did DNN learn when it successfully recovered the secret key?

Interpretability of DNN

Goal:

Use a neural network that is **easy to interpret**.

Main Contributions



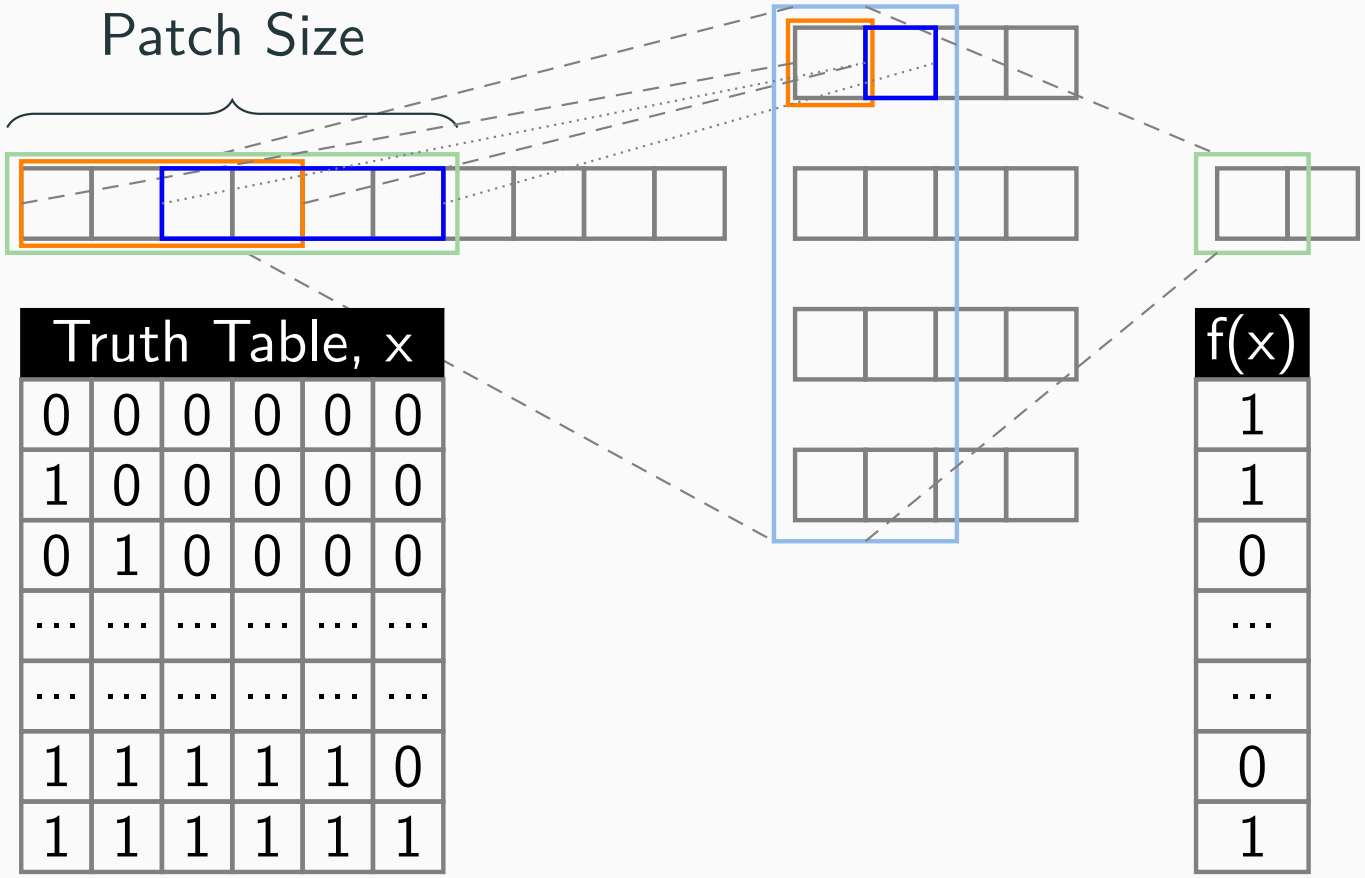
Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

Chapter 5: Apply Bayesian Optimization Hyperband (BOHB) to efficiently allocate resources when searching for well-performing DNNs automatically. (SAC'24)

Chapter 6: Employ transparent models that provide SAT formulae to facilitate understanding of the network and pinpoint the Points of Interest used by the DNN. (TCHES'23)

Truth Table Deep Convolutional Neural Network (TT-DCNN)



Binarized input and output.

- Can create a truth table by enumerating all possible input.
- This truth table can be converted into **SAT equations for interpretation**, namely **Disjunctive Normal Form (DNF)** (e.g. $(x_0 \wedge x_1) \vee (x_2 \wedge \neg x_1)$).

Heuristic used to analyse the SAT equations

We want to know which are the **important disjuncts/literals**, as most of the disjuncts are unnecessary for key recovery. We proposed three types of heuristic:

Heuristic used to analyse the SAT equations

We want to know which are the **important disjuncts/literals**, as most of the disjuncts are unnecessary for key recovery. We proposed three types of heuristic:

1. **Sieving** disjuncts based on their size
   (e.g. $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ is a disjunct of size 4),

Heuristic used to analyse the SAT equations

We want to know which are the **important disjuncts/literals**, as most of the disjuncts are unnecessary for key recovery. We proposed three types of heuristic:

1. **Sieving** disjuncts based on their size (e.g. $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ is a disjunct of size 4),
2. **Separating** disjuncts based on their combinations of literals (CoLs) (e.g. CoL of x_1, x_3, x_5, x_6, examples of disjuncts of these combination are $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$),

Heuristic used to analyse the SAT equations

We want to know which are the **important disjuncts/literals**, as most of the disjuncts are unnecessary for key recovery. We proposed three types of heuristic:

1. **Sieving** disjuncts based on their size
   (e.g. $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ is a disjunct of size 4),
2. **Separating** disjuncts based on their combinations of literals (CoLs) (e.g. CoL of x_1, x_3, x_5, x_6, examples of disjuncts of these combination are $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$),
3. **Trimming** disjuncts based on the literals.
   (e.g. trimming the literal (x_3) then the $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ become $(x_1 \wedge x_5 \wedge \neg x_6)$.)

Heuristic used to analyse the SAT equations

We want to know which are the **important disjuncts/literals**, as most of the disjuncts are unnecessary for key recovery. We proposed three types of heuristic:

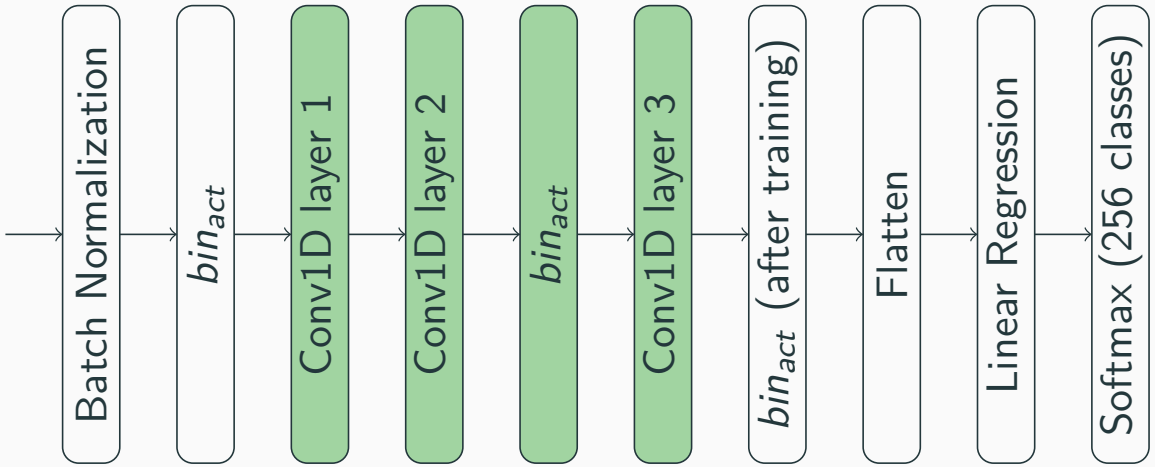
1. **Sieving** disjuncts based on their size (e.g. $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ is a disjunct of size 4),
2. **Separating** disjuncts based on their combinations of literals (CoLs) (e.g. CoL of x_1, x_3, x_5, x_6, examples of disjuncts of these combination are $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$),
3. **Trimming** disjuncts based on the literals. (e.g. trimming the literal (x_3) then the $(x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_6)$ become $(x_1 \wedge x_5 \wedge \neg x_6)$.)

Goal:

Find the most miniature set of rules that the neural network needs for key recovery.

TT-DCNN architecture, $TTSCA_{small}$

We found out that the neural network overfits very fast due to it being a perfect world. Therefore, we proposed the following architecture called $TTSCA_{small}$.



$TTSCA_{small}$ architecture

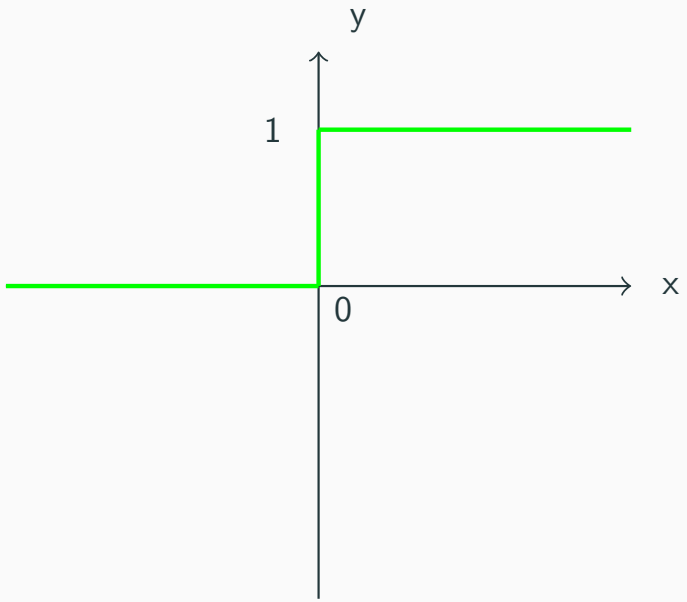
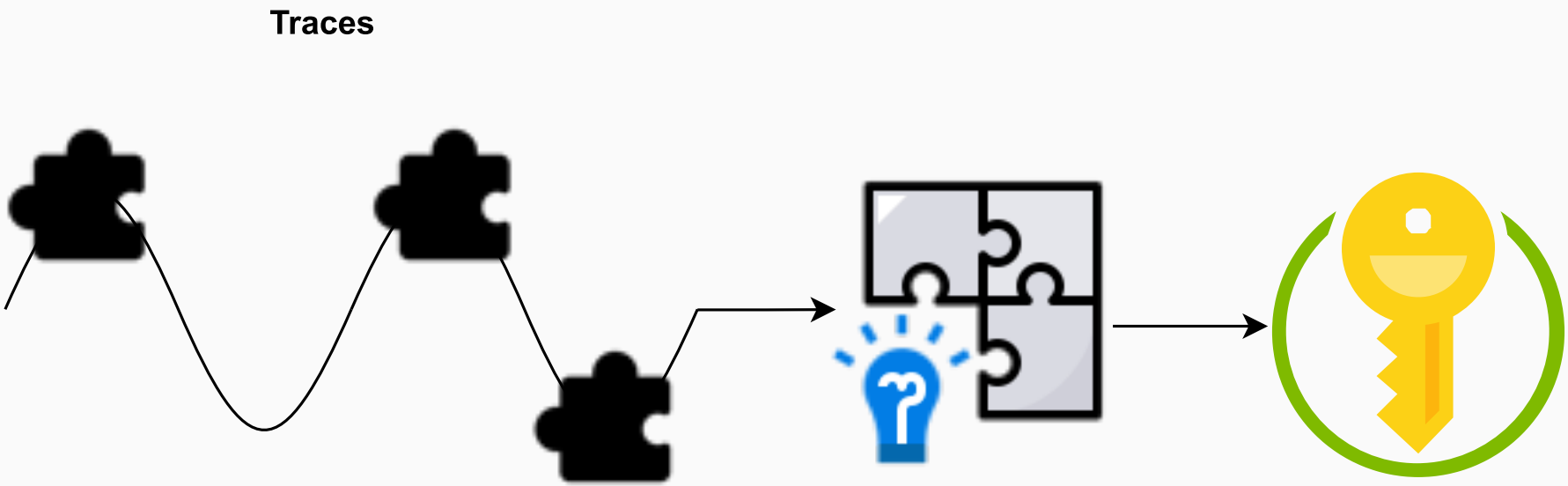


Fig: bin_{act}

Masking



Secret information is split into different shares.

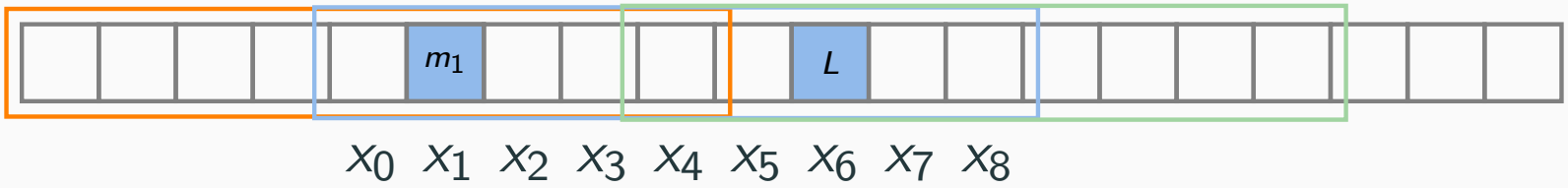
Masking

Supposed there are $d+1$ leakages points, $L_1, \ldots L_{d+1}$, then the secret variable, Z, is

$$Z = g(L_1, L_2, \ldots, L_{d+1}).$$

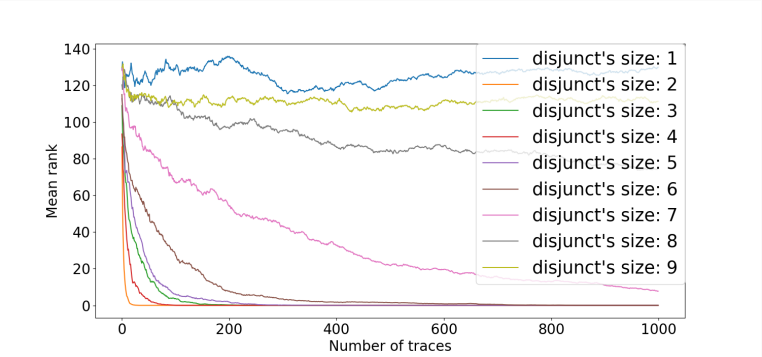
A common function g is the Boolean XOR of each leakage points (aka Boolean Masking).

A visualization of the simulated traces is shown below and we observed that the neural network sees the following:

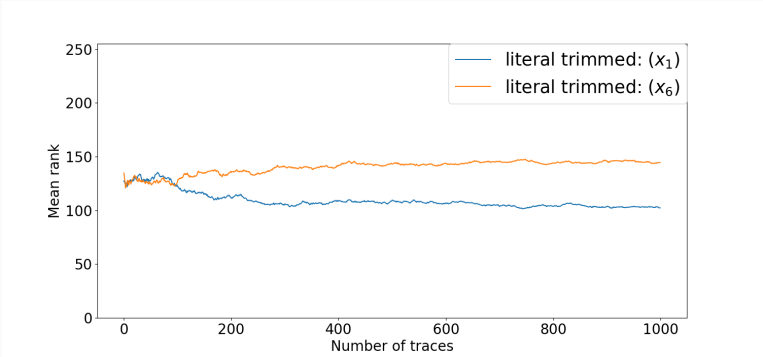


where $L = Z \oplus m_1$.

Results for simulated data with masking order 1

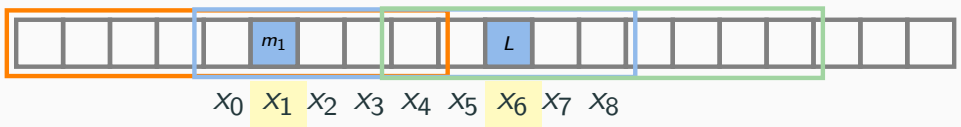


(a) Sieving based on size.



(b) Trimming.

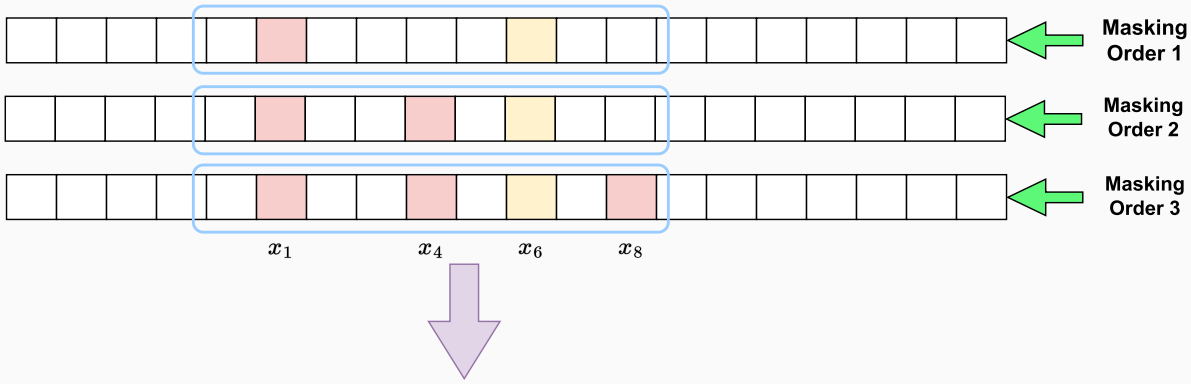
- **Sieving based on size**: The disjuncts of size 2 is the smallest size with $GE = 0$ (see orange line of Figure (a)).
- **Seperating based on CoLs:** There is only one CoL; the CoL for (x_1, x_6).
- **Trimming:** We trim based on (x_1) and (x_6) individually. (see Figure (b))



The most important literals are x_1 and x_6.

Results for simulated data

The $TTSCA_{small}$ can pinpoint the leakage's position and use it to retrieve the key.



Masking Order	Recovered Disjuncts
1	$(x_1 \wedge \neg x_6)$
2	$(x_1 \wedge \neg x_4 \wedge \neg x_6)$
3	$(x_1 \wedge x_4 \wedge x_6 \wedge x_8)$

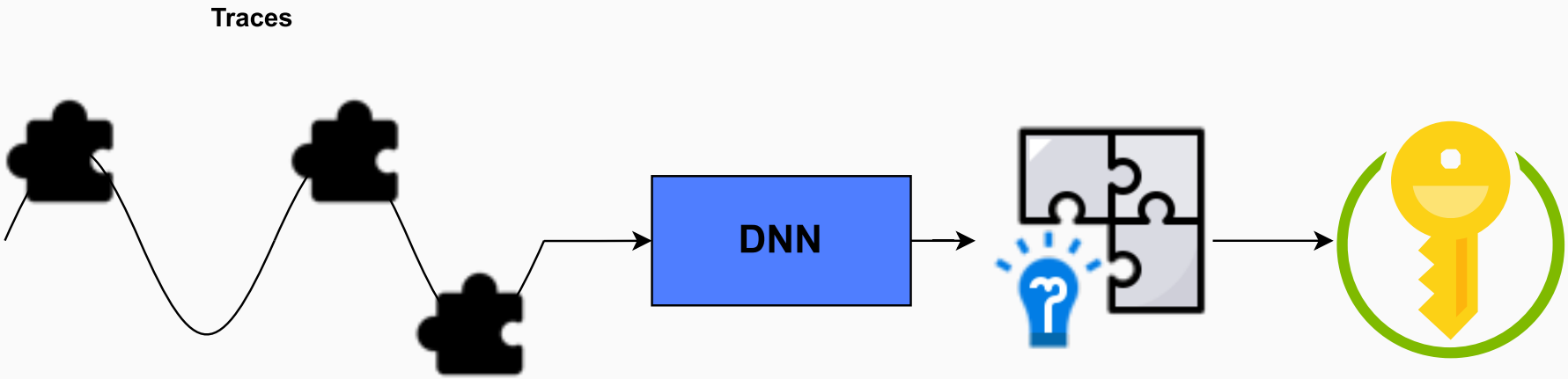
- We obtained similar results for masking orders of $0, 1, 2$ and 3.
- Validated that through the SAT equations from the network we are able to pinpoint the leakage position of the traces.

Further contributions

- We further **extend this work to real traces** by proposing an architecture call $TTSCA_{big}$.
- Able to locate windows of PoIs that correspond to the underlying leakage that the network uses.
- In the best case, we manage to find the underlying masking formula through the SAT equations.

Explainability of DNN

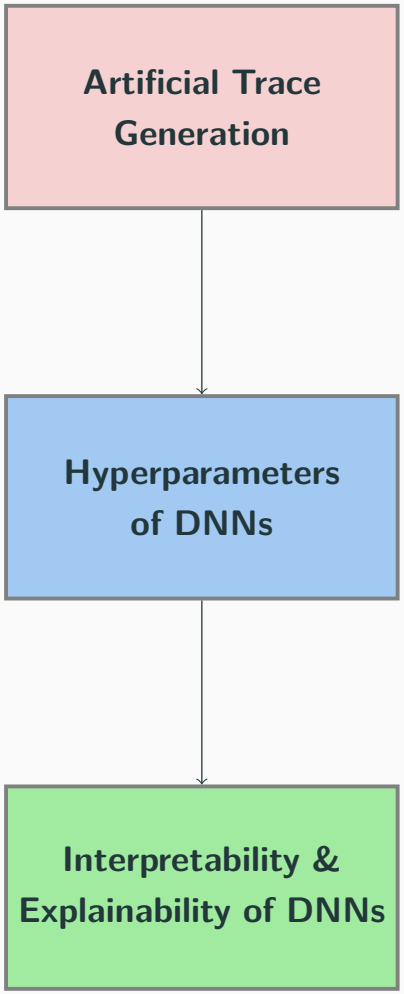
What about a generic neural network?



Since the DNN can **combine these shares implicitly** to retrieve the secret key of the target device and **traces contain relevant and redundant sample points**, an inherent question arises:

What relevant features/sample points that a trained DNN use to obtain the secret key?

Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

Chapter 5: Apply Bayesian Optimization Hyperband (BOHB) to efficiently allocate resources when searching for well-performing DNNs automatically. (SAC'24)

Chapter 6: Employ transparent models that provide SAT formulae to facilitate understanding of the network and pinpoint the Points of Interest used by the DNN. (TCHES'23)

Chapter 7: Develop an explainability technique for generic DNNs to locate the Points of Interest used by these networks. (Indocrypt'24)

Explainability of DNN

Assumption:

We assume that the DNN is trained and able to recover the secret key successfully.

Key Guessing Occlusion (KGO)

$index_{lst}$

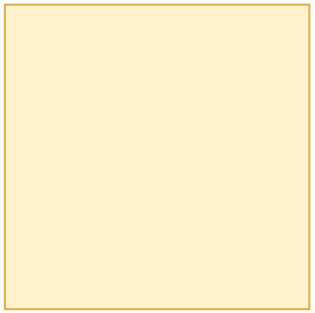
x_0	x_1	x_2
...	...	...
x_{N-3}	x_{N-2}	x_{N-1}

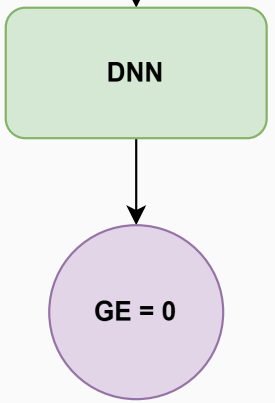
Attack Traces

x_0	x_1	x_2	x_3	x_4	...	...	...	x_{N-3}	x_{N-2}	x_{N-1}
9.04	23.8	31.5	15.1	21.4	...	...	...	61.7	29.8	44.6
14.4	41.0	23.9	16.0	8.9	...	...	...	31.0	29.8	24
...	...	...	...	...	...	...	...	...	...	...
7.52	16.5	31.5	15.1	21.4	...	...	...	61.7	29.8	44.6

Important sample points,

$index_{rev}$





Key Guessing Occlusion (KGO)

$index_{lst}$

x_3	x_0	x_{N-3}
...	...	...
x_{13}	x_{59}	x_1

Attack Traces

x_0	x_1	x_2	x_3	x_4	...	...	...	x_{N-3}	x_{N-2}	x_{N-1}
9.04	23.8	31.5	15.1	21.4	...	...	...	61.7	29.8	44.6
14.4	41.0	23.9	16.0	8.9	...	...	...	31.0	29.8	24
...	...	...	...	...	...	...	...	...	...	...
7.52	16.5	31.5	15.1	21.4	...	...	...	61.7	29.8	44.6

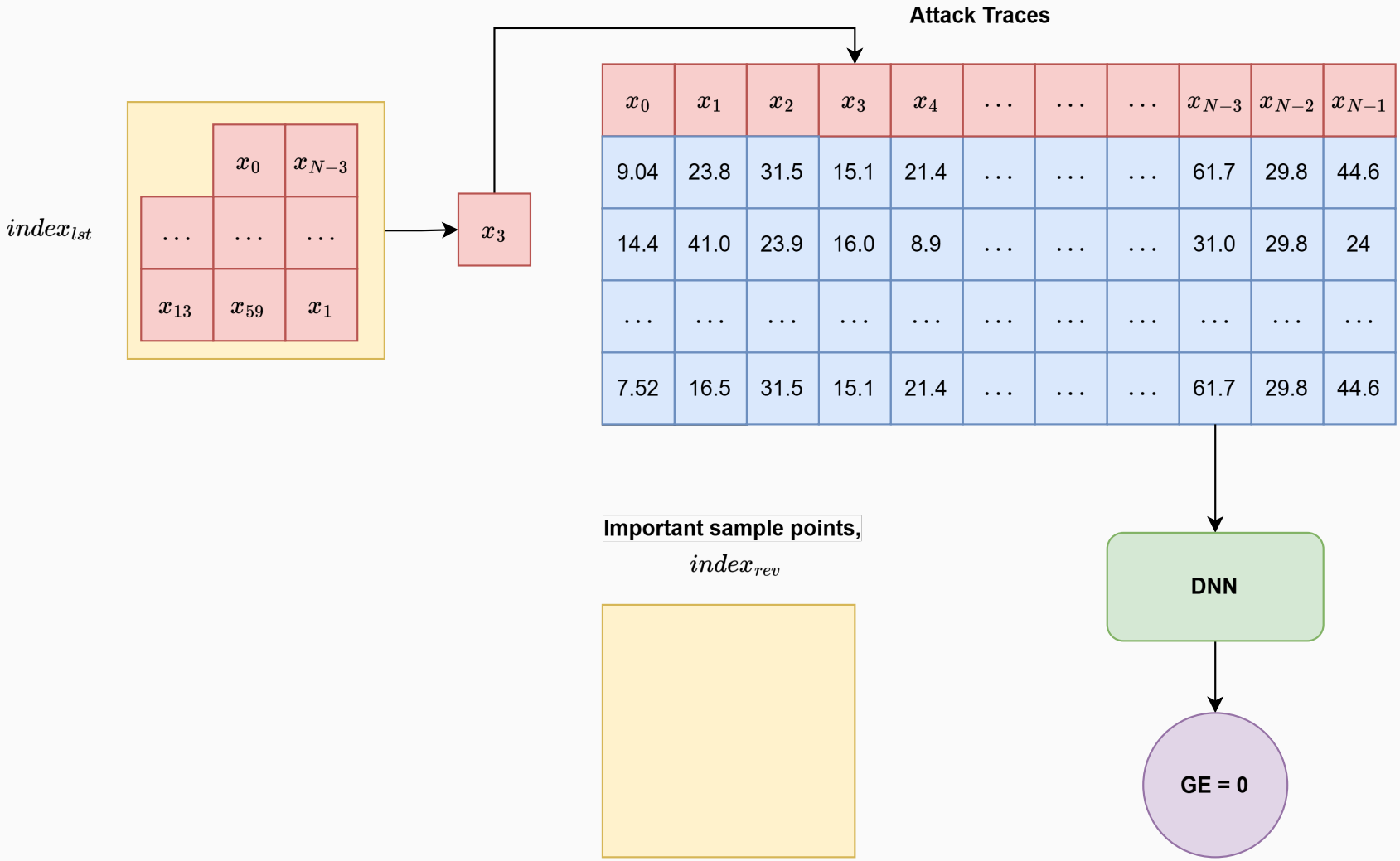
Important sample points,

$index_{rev}$

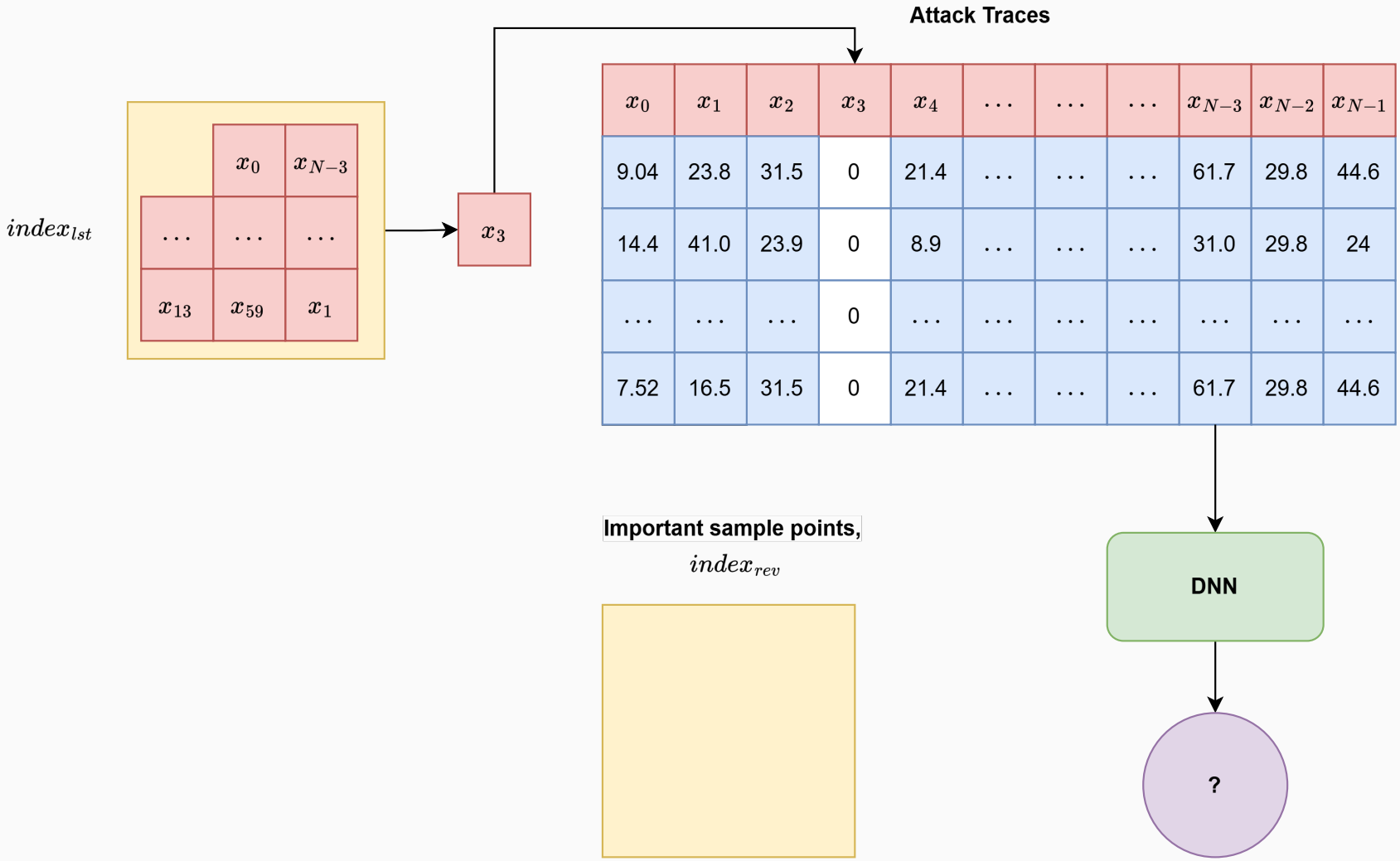
DNN

GE = 0

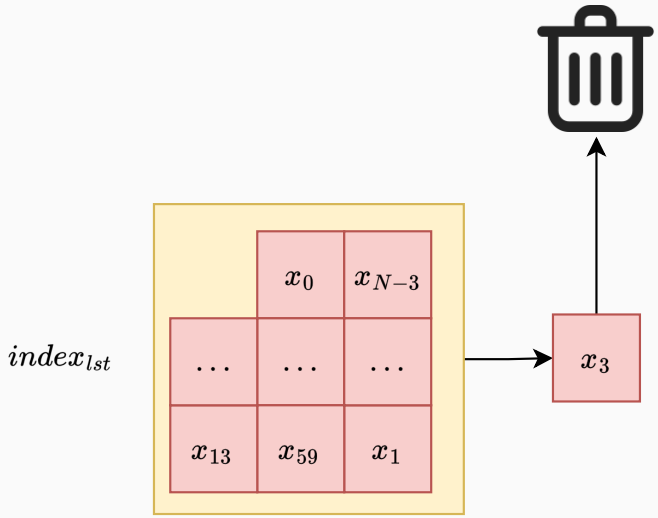
Key Guessing Occlusion (KGO)



Key Guessing Occlusion (KGO)



Key Guessing Occlusion (KGO)

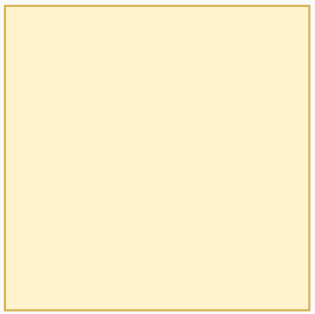


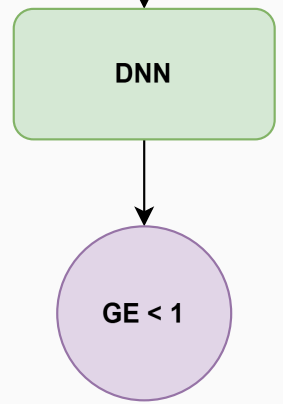
Attack Traces

x_0	x_1	x_2	x_3	x_4	...	...	...	x_{N-3}	x_{N-2}	x_{N-1}
9.04	23.8	31.5	0	21.4	...	...	...	61.7	29.8	44.6
14.4	41.0	23.9	0	8.9	...	...	...	31.0	29.8	24
...	...	...	0	...	...	...	...	...	...	...
7.52	16.5	31.5	0	21.4	...	...	...	61.7	29.8	44.6

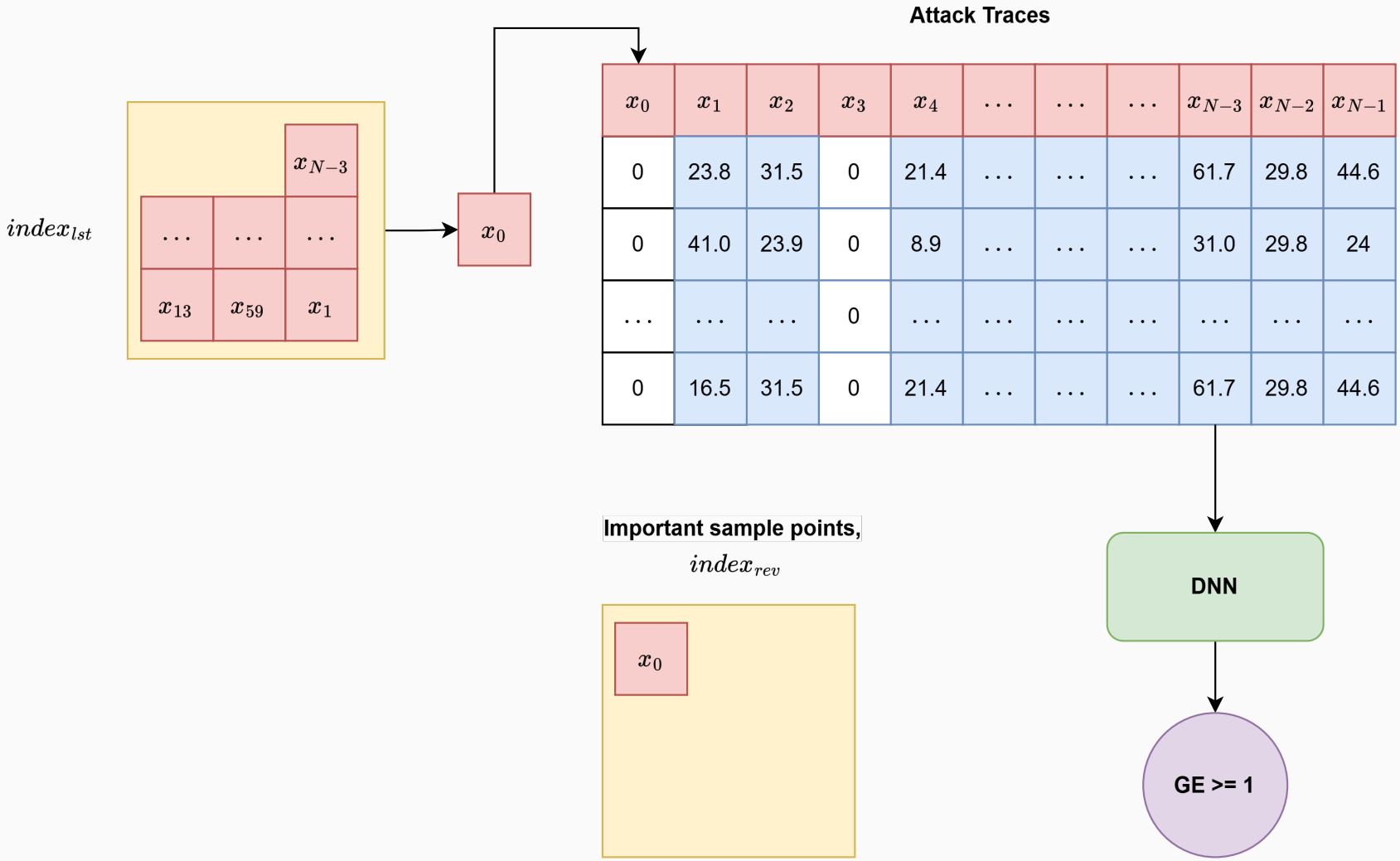
Important sample points,

$index_{rev}$

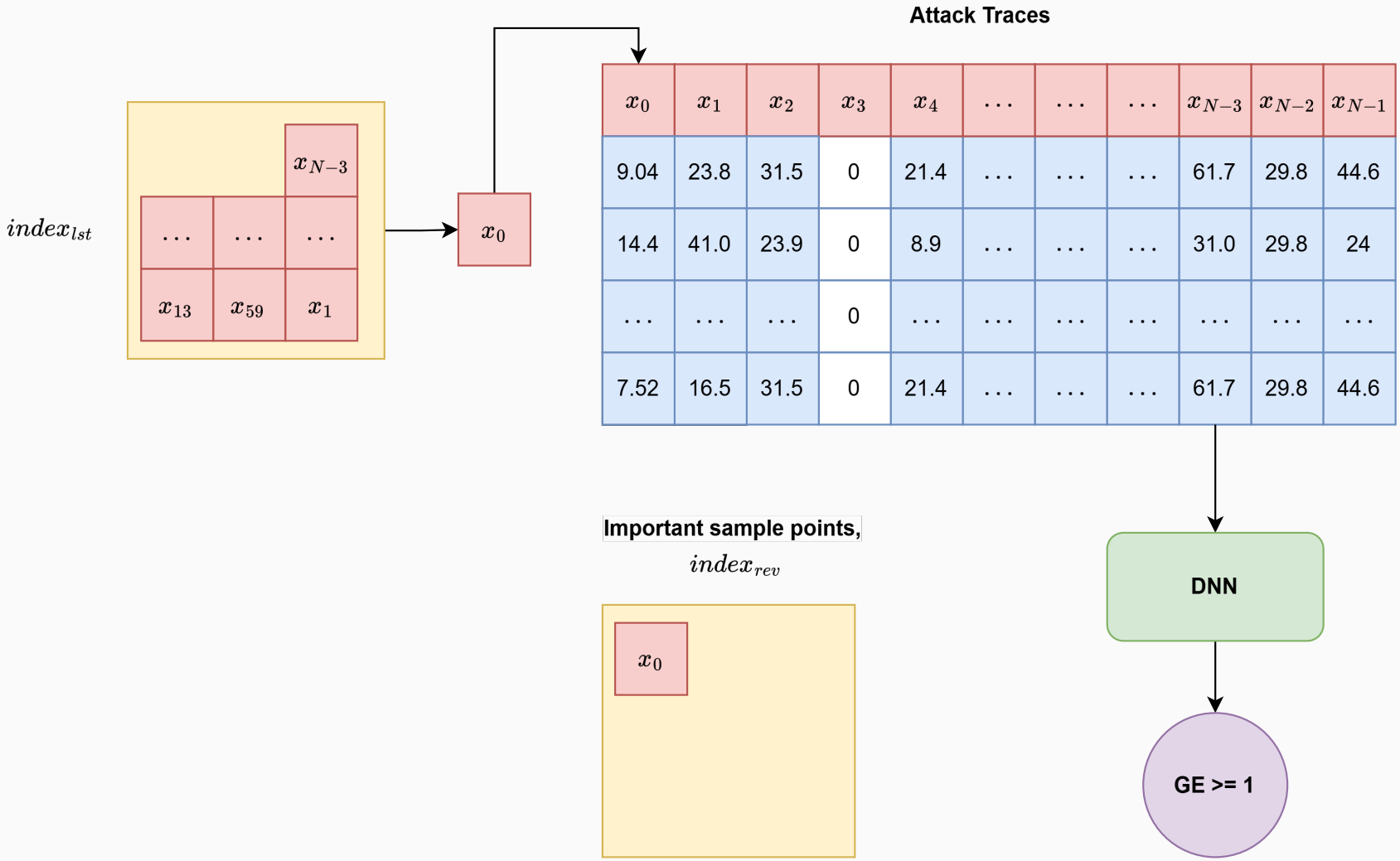




Key Guessing Occlusion (KGO)

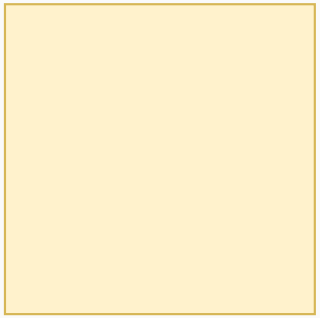


Key Guessing Occlusion (KGO)



Key Guessing Occlusion (KGO)

$index_{lst}$

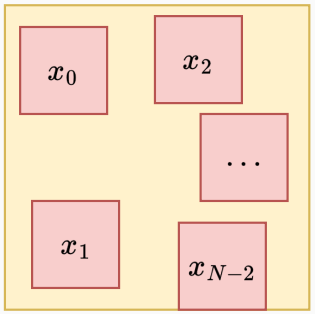


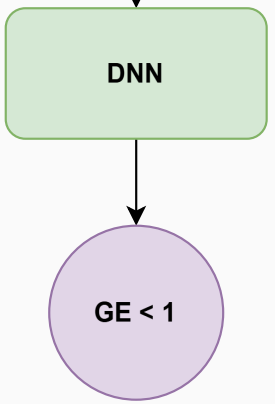
Attack Traces

x_0	x_1	x_2	x_3	x_4	...	...	...	x_{N-3}	x_{N-2}	x_{N-1}
9.04	23.8	31.5	0	0	...	...	...	0	29.8	0
14.4	41.0	23.9	0	0	...	...	...	0	29.8	0
...	...	...	...	...	...	...	...	...	...	...
7.52	16.5	31.5	0	0	...	...	...	0	29.8	0

Important sample points,

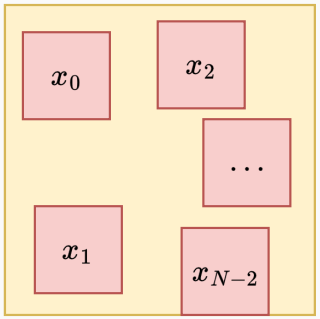
$index_{rev}$





Key Guessing Occlusion (KGO)

$index_{lst}$

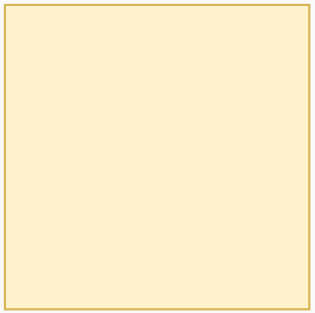


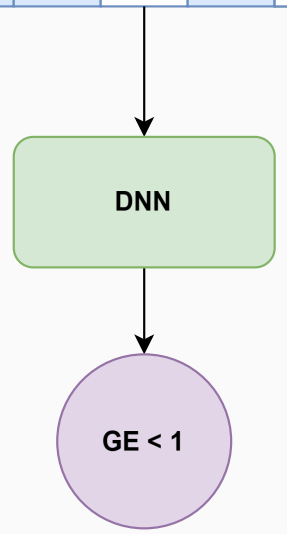
Attack Traces

x_0	x_1	x_2	x_3	x_4	...	...	...	x_{N-3}	x_{N-2}	x_{N-1}
9.04	23.8	31.5	0	0	...	...	...	0	29.8	0
14.4	41.0	23.9	0	0	...	...	...	0	29.8	0
...	...	...	...	...	...	...	...	...	...	...
7.52	16.5	31.5	0	0	...	...	...	0	29.8	0

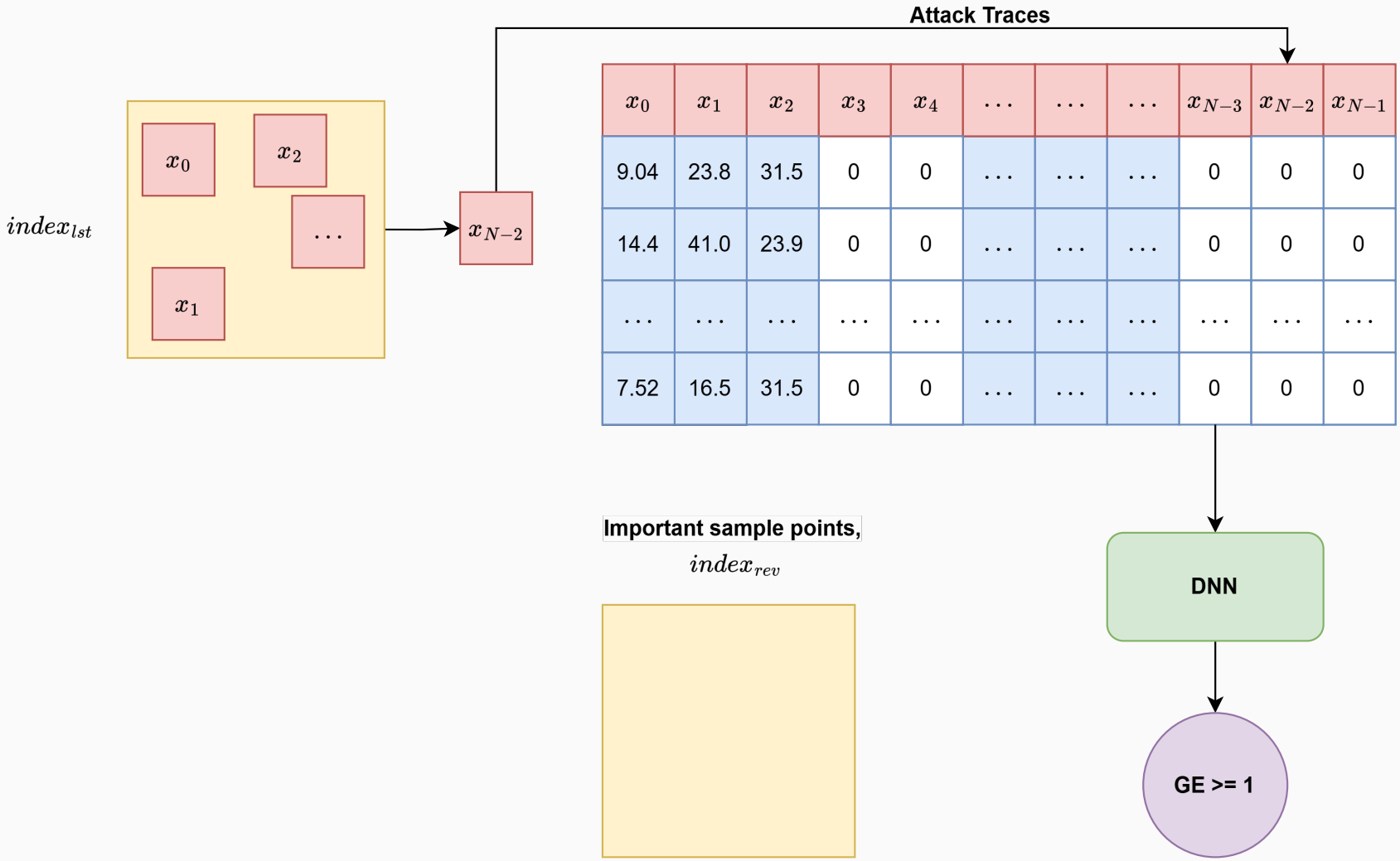
Important sample points,

$index_{rev}$



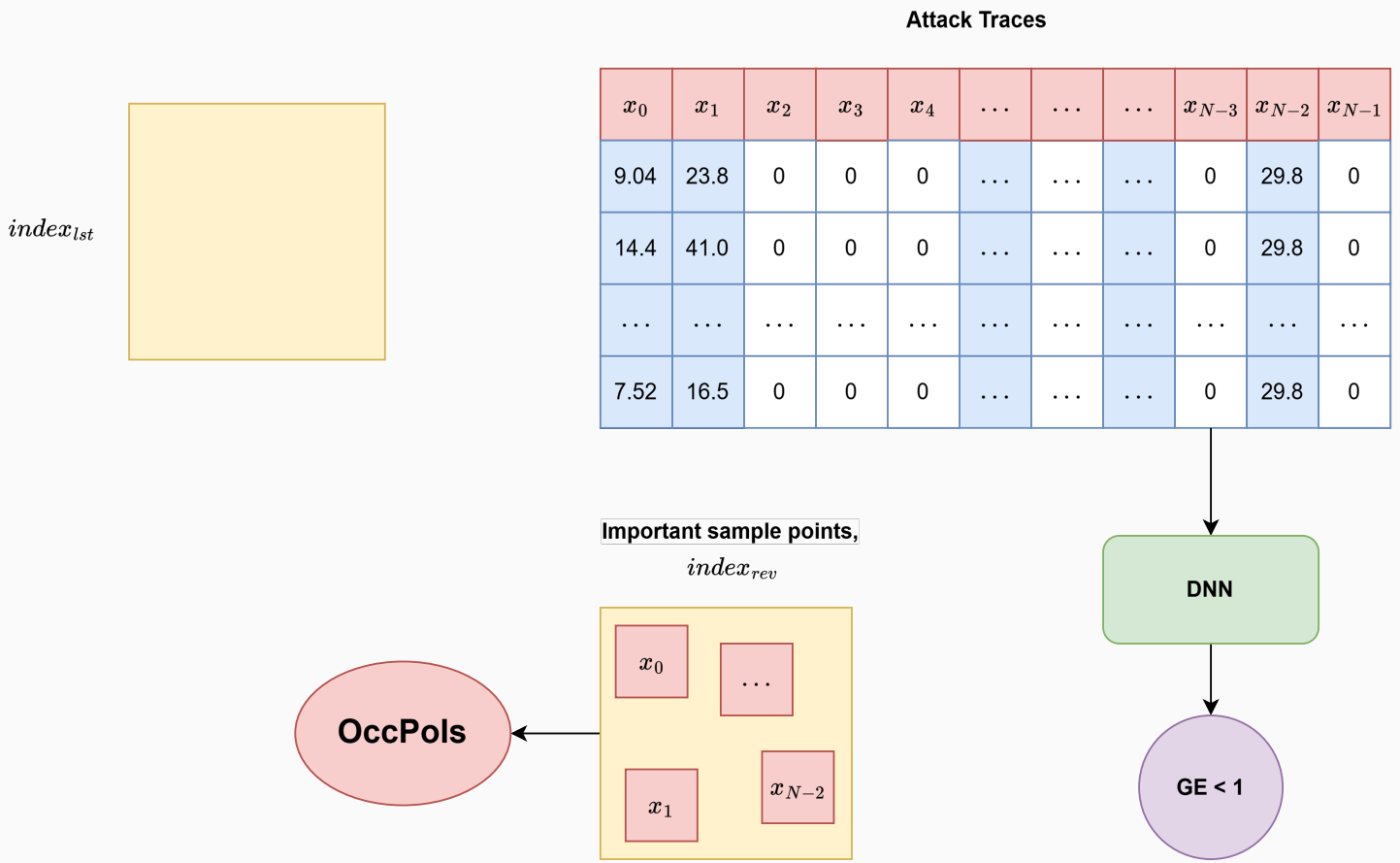


Key Guessing Occlusion (KGO)



Key Guessing Occlusion

We repeat the same until once a sample point is removed the GE will increase.



We call these important sample points as **OccPoIs**.

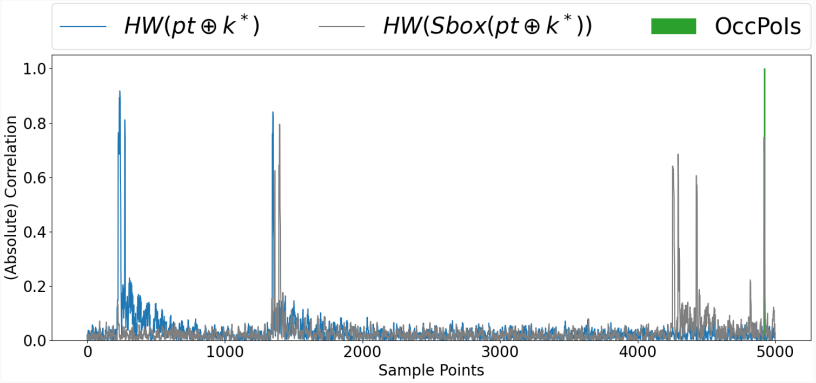
Exploring the number of OccPoIs

	CW	ASCADf	ASCADr	AES_HD
Total number of sample points	5000	700	1400	1250
ω	1	5	6	1

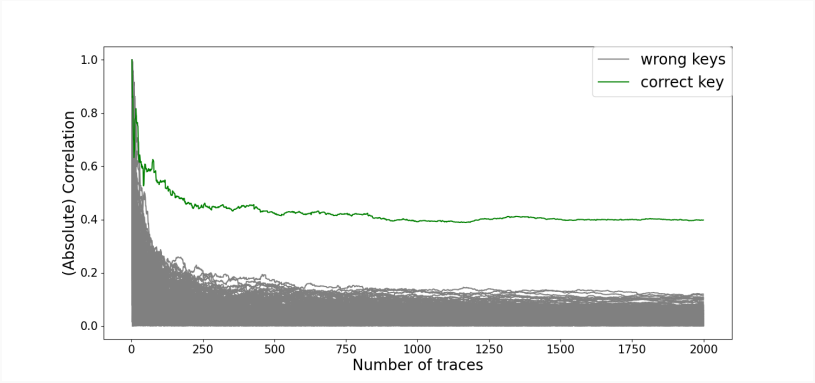
Table 3: The number of OccPoIs ω obtained by KGO.

- Relative small number of OccPoIs compare to the total number of sample points.
- In some cases, obtain the least number of sample points required to retrieve the secret key.

Exploring Leakages within OccPoIs: Chipwhisperer dataset



(a) CPA with respect to sample points.

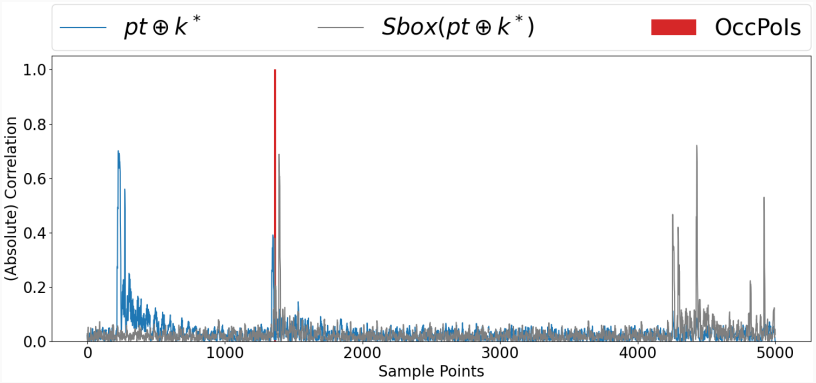


(b) CPA on the OccPoI with respect to the number of traces.

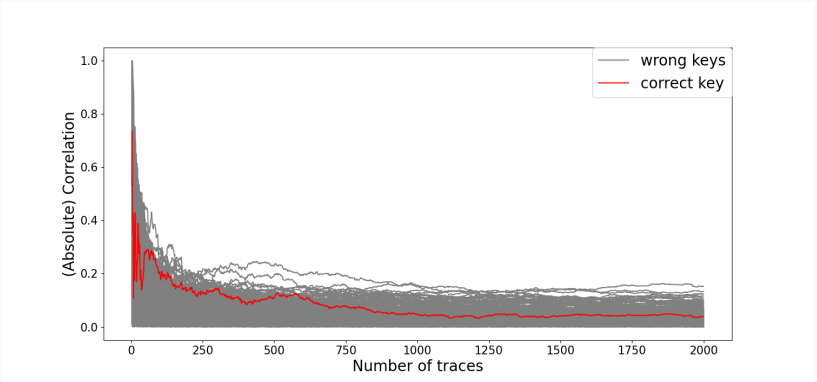
Figure 6: CPA in CW (HW).

- CPA measures the linear relationship between the sample point and a given leakage model.
- The DNN found sample points that are inline with classical technique, CPA.

Exploring Leakages within OccPoIs: Chipwhisperer dataset



(a) CPA with respect to sample points.

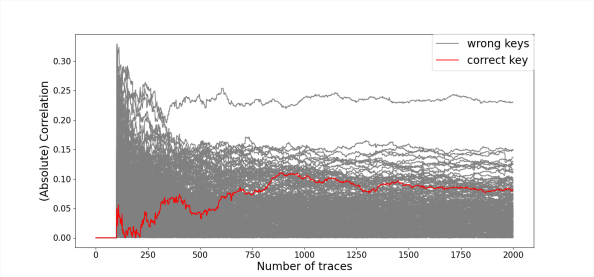


(b) CPA on the OccPoI with respect to the number of traces.

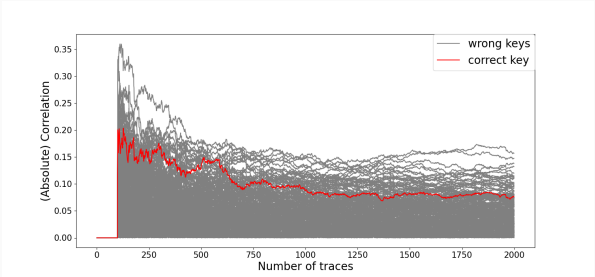
Figure 7: CPA in CW (ID).

- The DNN found sample points that the classical technique, CPA, missed.

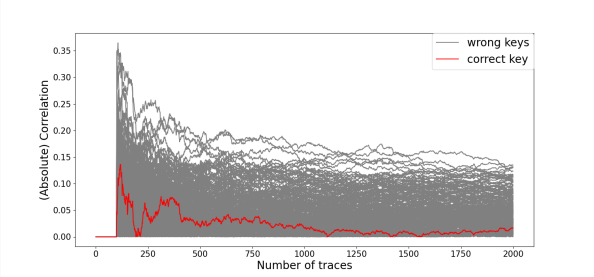
Exploring Leakages within OccPoIs: Chipwhisperer dataset



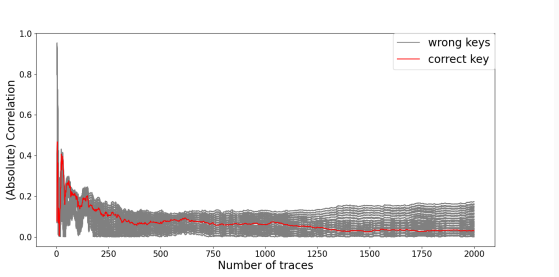
(a) $HW(Sbox(pt \oplus k^*))$.



(b) $MSB(Sbox(pt \oplus k^*))$.



(c) $LSB(Sbox(pt \oplus k^*))$.



(d) $pt \oplus k^*$.

- We tried different known leakage model and could not obtain the underlying leakage using classical techniques.
- This means that DNN is learning a more complex function to exploit the underlying leakage.

Feature selection tool for TA

- Since the number of OccPoIs is **much smaller** than the total number of sample points and
- DNN **could recover the secret key** from these sample points,
- We use KGO as a **feature selection tool**.

Exploitation of OccPoIs as feature selection tool

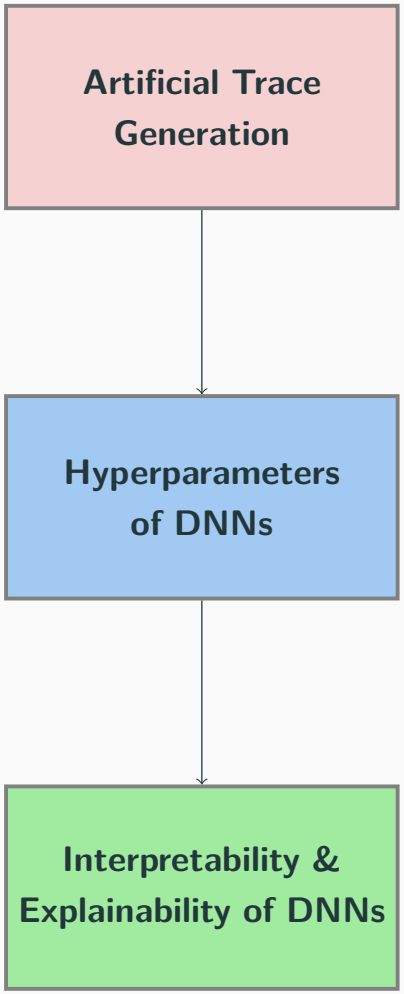
We compared with the classical features selection and other explainability techniques for Template Attack (TA).

	CW	ASCADf	ASCADr	AES_HD
SOSD	**3**	$> 10k$ ($GE = 2$)	$> 100k$ ($GE = 103$)	2623
SOST	9	$> 10k$ ($GE = 68$)	$> 100k$ ($GE = 51$)	2718
CPA (first-order)	11	$> 10k$ ($GE = 67$)	$> 100k$ ($GE = 162$)	2809
CPA (multi.)	-	367	**7184**	-
Saliency	9	$> 10k$ ($GE = 248$)	$> 100k$ ($GE = 210$)	3515
LRP	8	$> 10k$ ($GE = 100$)	50061	**2265**
1-Occlusion	18	$> 10k$ ($GE = 162$)	$> 100k$ ($GE = 7$)	3381
KGO	10	**313**	42991	6176

Table 4: NT_{GE} of the TA when using various feature selection techniques.

- KGO able to obtain $GE = 0$ for all datasets.
- KGO could obtain better results than second-order CPA for ASCADf.
- KGO is stable compare to other techniques.

Main Contributions



Chapter 3: Utilizes Diffusion Model to automatically generate artificial traces that capture leakage information. (AI-HWS'24)

Chapter 4: Investigate the effectiveness of two Deep Features Loss Functions: SoftNN & Center Loss (ASHES'23)

Chapter 5: Apply Bayesian Optimization Hyperband (BOHB) to efficiently allocate resources when searching for well-performing DNNs automatically. (SAC'24)

Chapter 6: Employ transparent models that provide SAT formulae to facilitate understanding of the network and pinpoint the Points of Interest used by the DNN. (TCHES'23)

Chapter 7: Develop an explainability technique for generic DNNs to locate the Points of Interest used by these networks. (Indocrypt'24)

Future Works

- Interpretability/Explainability of DLSCA,
- Other Ciphers/Primitives (e.g., Ascon or Dilithium),
- Deep Learning-based Non-Profiling Attack,
- Blind Side Channel Analysis.

Journal/Conference Papers

- **Chapter 3** is published as Trevor Yap and Dirmanto Jap. Creating from Noise: Trace Generations Using Diffusion Model for Side-Channel Attack. In: Andreoni, M. (eds) Applied Cryptography and Network Security Workshops. ACNS 2024. Lecture Notes in Computer Science, vol 14586. Springer, Cham. https://doi.org/10.1007/978-3-031-61486-6_7.
- **Chapter 4** is published as Trevor Yap, Stjepan Picek, and Shivam Bhasin. 2023. Beyond the Last Layer: Deep Feature Loss Functions in Side-channel Analysis. In Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security (ASHES '23). Association for Computing Machinery, New York, NY, USA, 73–82. https://doi.org/10.1145/3605769.3623996.
- **Chapter 5** is published as Trevor Yap, Shivam Bhasin and Léo Weissbart. Train Wisely: Multifidelity Bayesian Optimization Hyperparameter Tuning in Side-Channel Analysis. Cryptology ePrint Archive (2024). Available at https://eprint.iacr.org/2024/170. Accepted in Selected Areas in Cryptography (SAC'24).
- **Chapter 6** is published as Trevor Yap, Adrien Benamira, Shivam Bhasin, & Thomas Peyrin (2023). Peek into the Black-Box: Interpretable Neural Network using SAT Equations in Side-Channel Analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023(2), 24–53. https://doi.org/10.46586/tches.v2023.i2.24-53.
- **Chapter 7** is based on Trevor Yap, Shivam Bhasin and Stjepan Picek. OccPoIs: Points of Interest based on Neural Network's Key Recovery in Side-Channel Analysis through Occlusion (2023). Available at https://eprint.iacr.org/2023/1055. Accepted at the International Conference on Cryptology in India 2024 (Indocrypt'24)

Additional Papers

- Kai Hu, Thomas Peyrin, Quan Quan Tan, Trevor Yap (2023). Revisiting Higher-Order Differential-Linear Attacks from an Algebraic Perspective. In: Guo, J., Steinfeld, R. (eds) Advances in Cryptology – ASIACRYPT 2023. ASIACRYPT 2023. Lecture Notes in Computer Science, vol 14440. Springer, Singapore. https://doi.org/10.1007/978-981-99-8727-6_14
- Adrien Benamira, Thomas Peyrin, Trevor Yap, Tristan Guérand and Bryan Hooi, "Truth Table Net: Scalable, Compact & Verifiable Neural Networks with a Dual Convolutional Small Boolean Circuit Networks Form in International Joint Conference on Artificial Intelligence.https://doi.org/10.24963/ijcai.2024/2
- Kai Hu, Trevor Yap (2024). Perfect Monomial Prediction for Modular Addition. IACR Transactions on Symmetric Cryptology, 2024(3), 177-199. https://doi.org/10.46586/tosc.v2024.i3.177-199
- Minghui, Z., & Yap, T. (2024). Avenger Ensemble: Genetic Algorithm-Driven Ensemble Selection for Deep Learning-based Side-Channel Analysis. Retrieved from https://eprint.iacr.org/2024/1949 Accepted in CASCADE'2025.
- Rezaeezade, A., Yap, T., Jap, D., Bhasin, S., & Picek, S. (2025). Breaking the Blindfold: Deep Learning-based Blind Side-channel Analysis. Retrieved from https://eprint.iacr.org/2025/157

References


Acharya, Rabin Yu, Fatemeh Ganji, and Domenic Forte (2023). **"Information Theory-based Evolution of Neural Networks for Side-channel Analysis".** In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1, pp. 401–437. DOI: 10.46586/TCHES.V2023.I1.401-437. URL: https://doi.org/10.46586/tches.v2023.i1.401-437.


Cagli, Eleonora, Cécile Dumas, and Emmanuel Prouff (2017). **"Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing".** In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, pp. 45–68. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4%5C_3.

References

Maghrebi, Houssem, Thibault Portigliatti, and Emmanuel Prouff (2016). **"Breaking Cryptographic Implementations Using Deep Learning Techniques".** In: *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*. Ed. by Claude Carlet, M. Anwar Hasan, and Vishal Saraswat. Vol. 10076. Lecture Notes in Computer Science. Springer, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1. URL: https://doi.org/10.1007/978-3-319-49445-6%5C_1.

Mukhtar, Naila et al. (2022). **"Fake It Till You Make It: Data Augmentation Using Generative Adversarial Networks for All the Crypto You Need on Small Devices".** In: *Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings*. Ed. by Steven D. Galbraith. Vol. 13161. Lecture Notes in Computer Science. Springer, pp. 297–321. DOI: 10.1007/978-3-030-95312-6_13. URL: https://doi.org/10.1007/978-3-030-95312-6%5C_13.

References

- Picek, Stjepan et al. (2019). **"The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations".** In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1, pp. 209–237. DOI: 10.13154/TCHES.V2019.I1.209-237. URL: https://doi.org/10.13154/tches.v2019.i1.209-237.
- Rijsdijk, Jorai et al. (2021). **"Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis".** In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3, pp. 677–707. DOI: 10.46586/TCHES.V2021.I3.677-707. URL: https://doi.org/10.46586/tches.v2021.i3.677-707.
- Wang, Ping et al. (2020). **"Enhancing the Performance of Practical Profiling Side-Channel Attacks Using Conditional Generative Adversarial Networks".** In: *CoRR* abs/2007.05285. arXiv: 2007.05285. URL: https://arxiv.org/abs/2007.05285.

References

- Wouters, Lennert et al. (2020). **"Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks".** In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3, pp. 147–168. DOI: 10.13154/TCHES.V2020.I3.147-168. URL: https://doi.org/10.13154/tches.v2020.i3.147-168.
- Wu, Lichao, Guilherme Perin, and Stjepan Picek (2024). **"I Choose You: Automated Hyperparameter Tuning for Deep Learning-Based Side-Channel Analysis".** In: *IEEE Trans. Emerg. Top. Comput.* 12.2, pp. 546–557. DOI: 10.1109/TETC.2022.3218372. URL: https://doi.org/10.1109/TETC.2022.3218372.
- Zaid, Gabriel et al. (2020). **"Methodology for Efficient CNN Architectures in Profiling Attacks".** In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.1, pp. 1–36. DOI: 10.13154/TCHES.V2020.I1.1-36. URL: https://doi.org/10.13154/tches.v2020.i1.1-36.

Appreciation

Appreciation

Thomas Peyrin

Shivam Bhasin

Appreciation

Thomas Peyrin

Shivam Bhasin

Stjepan Picek

Appreciation

Thomas Peyrin

Shivam Bhasin

Stjepan Picek

- colleagues and researchers

Appreciation

Thomas Peyrin

Shivam Bhasin

Stjepan Picek

- colleagues and researchers
- family and friends

Appreciation

Thomas Peyrin

Shivam Bhasin

Stjepan Picek

Emily

- colleagues and researchers
- family and friends

Thank You!