Hardware Security of Cryptography and Deep Learning

Jakub Breier

Silicon Austria Labs, Graz, Austria

Dealer Seminar, PARC

30 August 2022

Talk Outline

- Hardware attacks on cryptography
  - Classification
  - Fault injection attacks
  - Differential fault analysis
  - DEFAULT – design of a fault resistant cipher
- Hardware attacks on neural networks
  - Misclassification
  - Reverse engineering
  - Backdoor planting

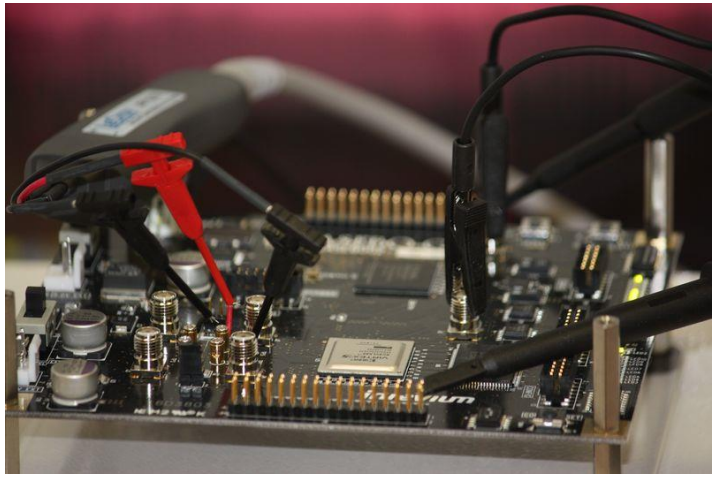
Hardware Attacks on Cryptography

Hardware Attacks on Cryptography

- Cryptography provides algorithms that enable secure communication in theory
- In real world, these algorithms have to be implemented on real devices:
  - software implementations - general-purpose devices
  - hardware implementations - dedicated secure hardware devices
- To evaluate security level of cryptographic implementations, it is necessary to include physical security assessment

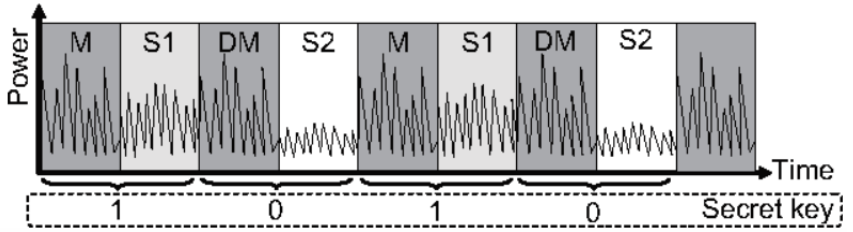
Classification

- Fault injection attacks
  - Optical fault injection
  - Electromagnetic fault injection
  - Clock/voltage glitching
- Side-channel attacks
  - Timing analysis
  - Power analysis
  - Electromagnetic analysis
- Hardware trojans
- Probing

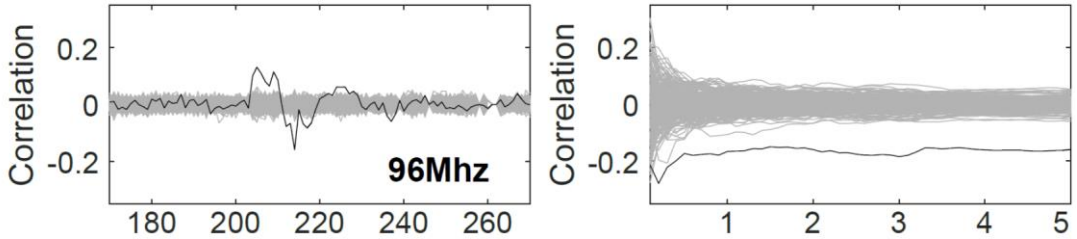


Side-Channel Attacks

- Exploit device leakage of various forms
  - Timing, power, electromagnetic emanation, acoustic signals, photon emission,…
- Can use different types of information
  - Hamming weight/distance (HW/HD), stochastic leakage, transition count, …
- Two general types:
  - **Simple attacks** – exploit operation dependent leakage (usually single trace)
  - **Differential attacks** – exploit data dependent leakage (many traces)



Simple power analysis[1]



Differential power analysis[2]

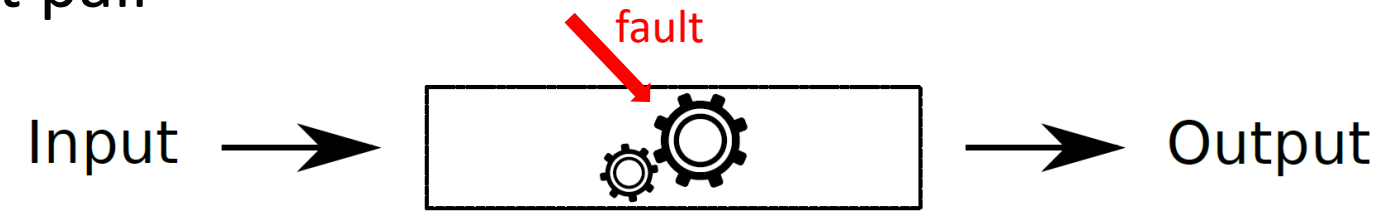
[1] A. Miyamoto, et al: Enhanced power analysis attack using chosen message against RSA hardware implementations, ISCS 2008.
[2] F. Schellenberg, et al.: An Inside Job: Remote Power Analysis Attacks on FPGAs, DATE 2018.

Fault Injection Attacks

- The best cryptanalysis of AES-128 needs complexity of $2^{126.1}$

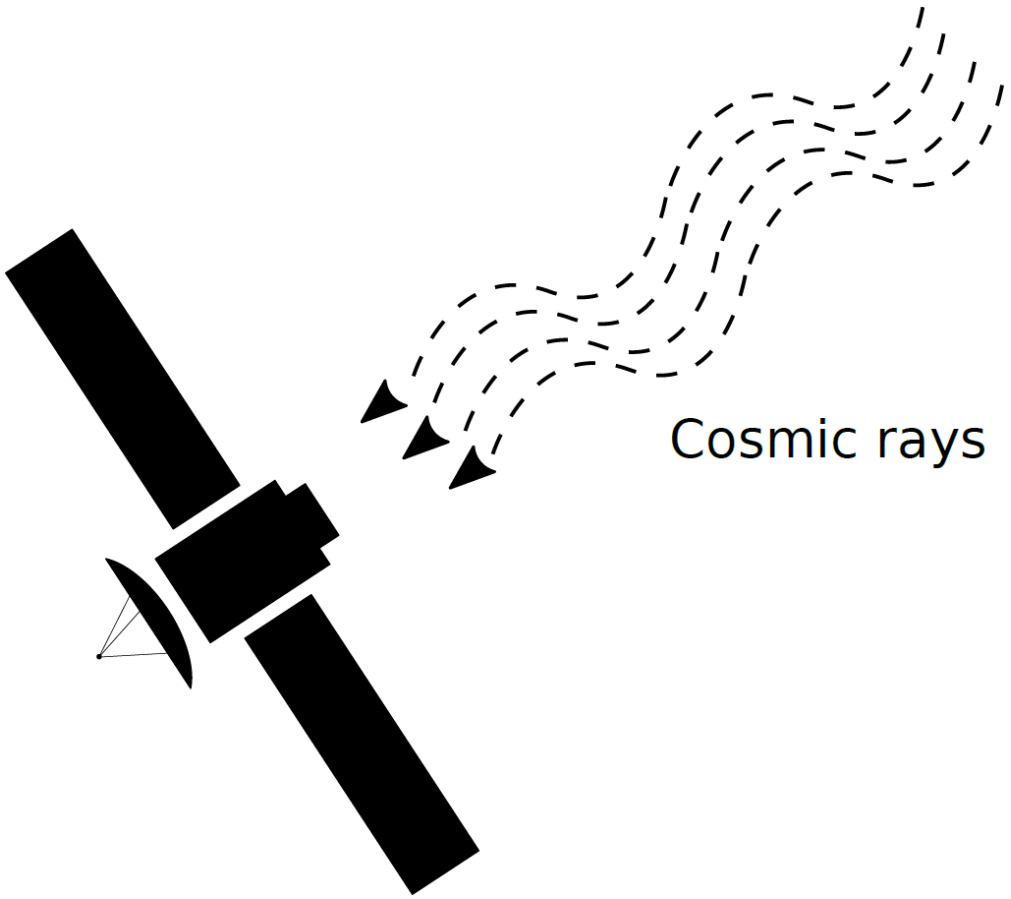
  - A. Bogdanov et al. Biclique cryptanalysis of the full AES, ASIACRYPT 2011.

- The best fault attack on AES-128 needs just one faulty and one correct ciphertext pair



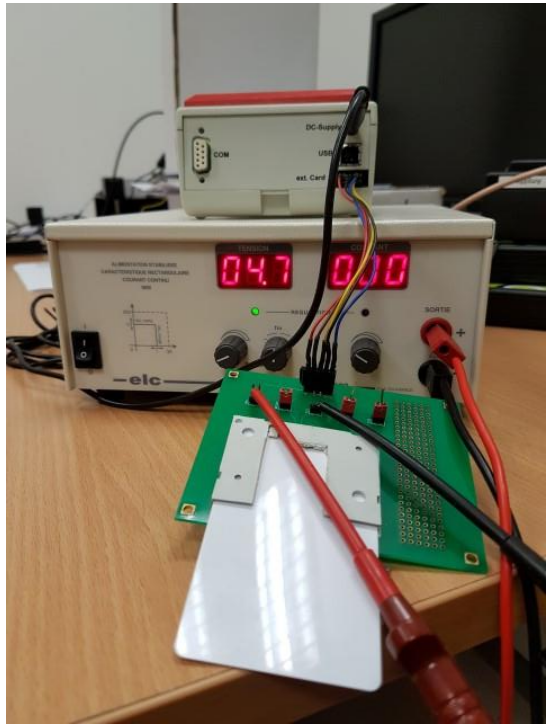
  - J. Breier et al. Laser Profiling for the Back-Side Fault Attacks: With a Practical Laser Skip Instruction Attack on AES, CPSS 2015.

First IC Disturbances – Cosmic Rays and Satellites



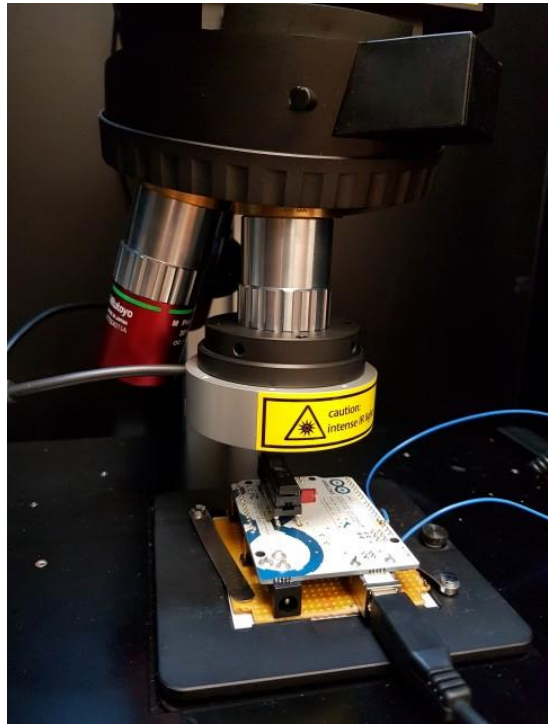
D. Binder et al. Satellite anomalies from galactic cosmic rays, IEEE Transactions on Nuclear Science, 1975.

Fault Injection Techniques in Practice



Voltage Glitching
$

EM Pulse Injection
$$



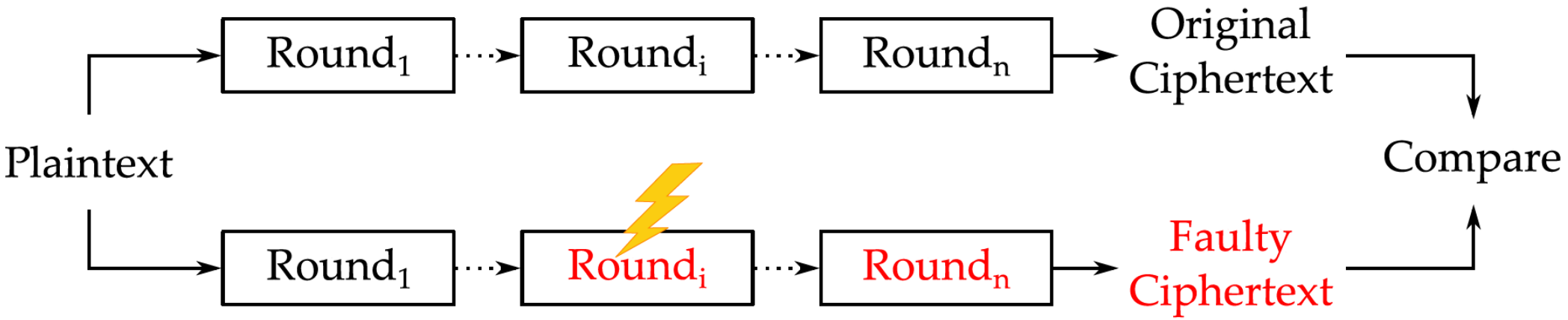
Laser Fault Injection
$$$

Fault Attacks in Cryptography

- Fault attacks exploit the possibility to insert a fault in the process of the algorithm execution in a way that could help to reveal the key
- The idea of fault attacks was introduced by Boneh, DeMillo and Lipton in 1996
  - D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults, EUROCRYPT’97.
- The first practical attack was implemented by Biham and Shamir, introducing a Differential Fault Analysis (DFA) on DES
  - E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems, CRYPTO'97.

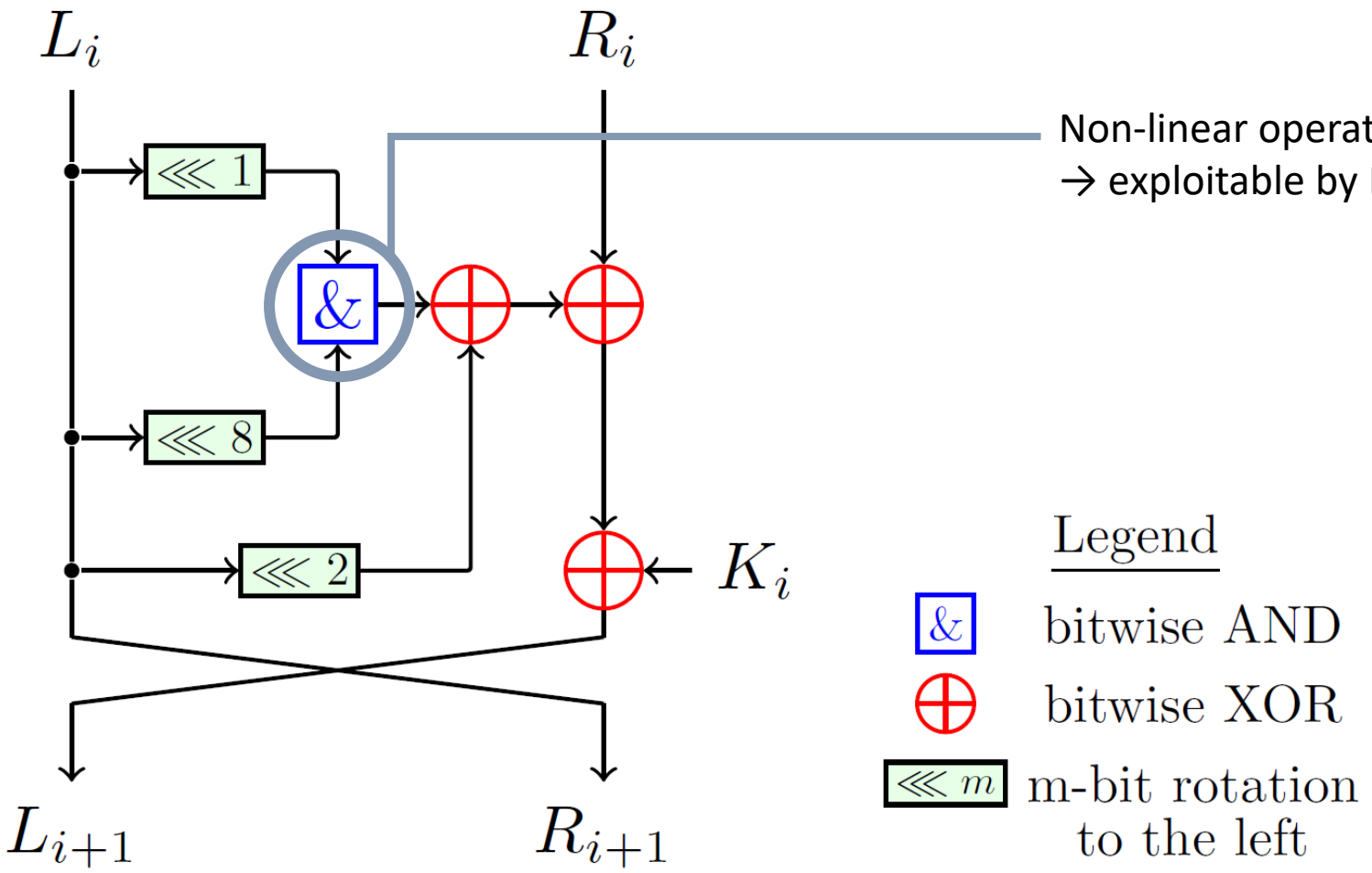
Working Principle of DFA

- Attacker injects a fault in a chosen round of the algorithm to get the desired fault propagation at the end of an encryption
- The secret key can then be determined by examining the differences between a correct and a faulty ciphertext



Example – SIMON Block Cipher[1]

- Typical block cipher alternates between linear and non-linear operations
- Linear function is defined as
  $$f(a \oplus \delta) = f(a) \oplus f(\delta)$$
- Examples of linear functions:
  - Logical XOR
  - Byte shift
  - Bit permutation
- Examples of non-linear functions:
  - Modular addition
  - Logical AND
  - SBox



[1]Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L., 2015, June. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd annual design automation conference* (pp. 1-6).

Exploiting AND Operation by DFA

- *Attacker's knowledge:* ability to observe the change in 'c'
- *Attacker's goal:* recover the value of 'b'

a	b	c = a & b
0	0	0
0	1	0
1	0	0
1	1	1

Flip bit 'a'

a'	b	c'= a'& b
1	0	0
1	1	1
0	0	0
0	1	0

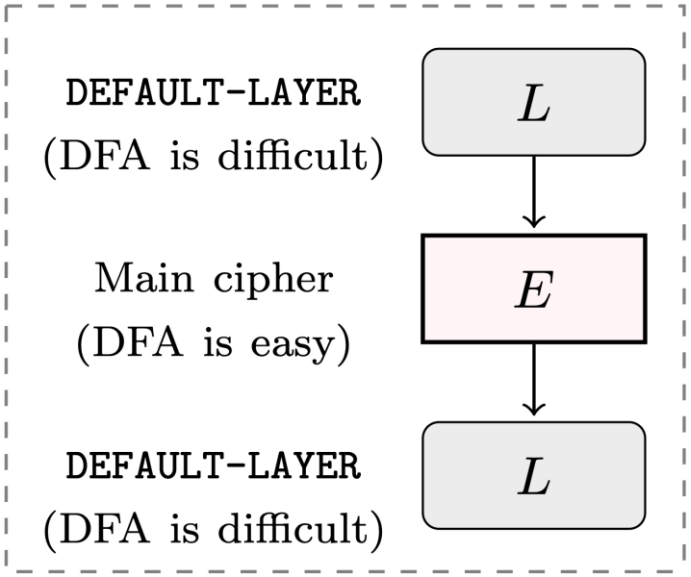
- If the result does not change → 'b' is 0
- If the result changes → 'b' is 1

Countermeasures

- Applied at the implementation level – either SW or HW
- 3 main approaches
  - Redundancy – duplication, error detection/correction codes
  - Sensors – light, EM field disturbance
  - Protocol-level – re-keying
- Out of scope for cryptography design
- Need to be verified and tested in evaluation lab
  - Common Criteria
  - EMVCo

DEFAULT[1]: A Cipher Design Resistant to DFA

- Main goals
  - Remove the dependency on the fault-secure implementation
  - Impose a lower bound on the search complexity of DFA
- Approach
  - Increase the linearity of the non-linear components
  - Design a sandwich construction
  - Use GIFT-128 as the base cipher



[1]Baksi, A., Bhasin, S., Breier, J., Khairallah, M., Peyrin, T., Sarkar, S. and Sim, S.M., 2021, December. DEFAULT: Cipher Level Resistance Against Differential Fault Attack. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 124-156). Springer, Cham.

Differential Properties of SBoxes

- An n-bit S-box S is a look-up table

$$S(x) = y$$

  where $x, y \in \mathbb{F}_2^n$

- For an input and output difference $\delta, \Delta \in \mathbb{F}_2^n \setminus \{0\}$, x is a solution if

$$S(x) \oplus S(x \oplus \delta) = \Delta$$

  - If x is a solution, so is $x \oplus \delta$. Solutions come in pairs

- Standard cipher designs minimize the number of solutions to protect against differential cryptanalysis[1]

[1]Biham, E. and Shamir, A., 1991. Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, *4*(1), pp.3-72.

Difference Distribution Table (DDT)

3-bit Sbox:

x	0	1	2	3	4	5	6	7
$S(x)$	4	7	0	5	2	6	3	1

DDT:

Δ \ δ	1	2	3	4	5	6	7
1		67	01	45	23		
2	46	13		02			57
3	03		47		56		12
4	15		26	37		04	
5	27	05			14	36	
6		24	35			17	06
7				16	07	25	34

Entries are Sbox inputs

Linear Structures in SBoxes

- An element $a \in \mathbb{F}_2^n$ is a *linear structure* of S if for some constant $b \in \mathbb{F}_2^n$,

$$S(z) \oplus S(z \oplus a) = b$$

  holds for all $z \in \mathbb{F}_2^n$.

- If x is a solution for input-output difference (δ, Δ),

  then $x \oplus a$ is always a solution as well

$$\begin{aligned} S(x \oplus a) \oplus S(x \oplus a \oplus \delta) &= (S(x) \oplus b) \oplus (S(x \oplus \delta) \oplus b) \\ &= S(x) \oplus S(x \oplus \delta) \\ &= \Delta \end{aligned}$$

DDT for SBox with Linear Structures

x	0	1	2	3	4	5	6	7
$S(x)$	0	7	2	5	4	6	3	1

Δ \ δ	1	2	3	4	5	6	7
1		4567					0123
2		0123					4567
3					01234567		
4	1256			0347			
5			1247			0356	
6			0356			1247	
7	0347			1256			

DFA Security for Substitution-Permutation Networks

♯ LS	State Size	DFA Security
2	128	2^{32}
	256	2^{64}
4	128	2^{64}
	256	2^{128}

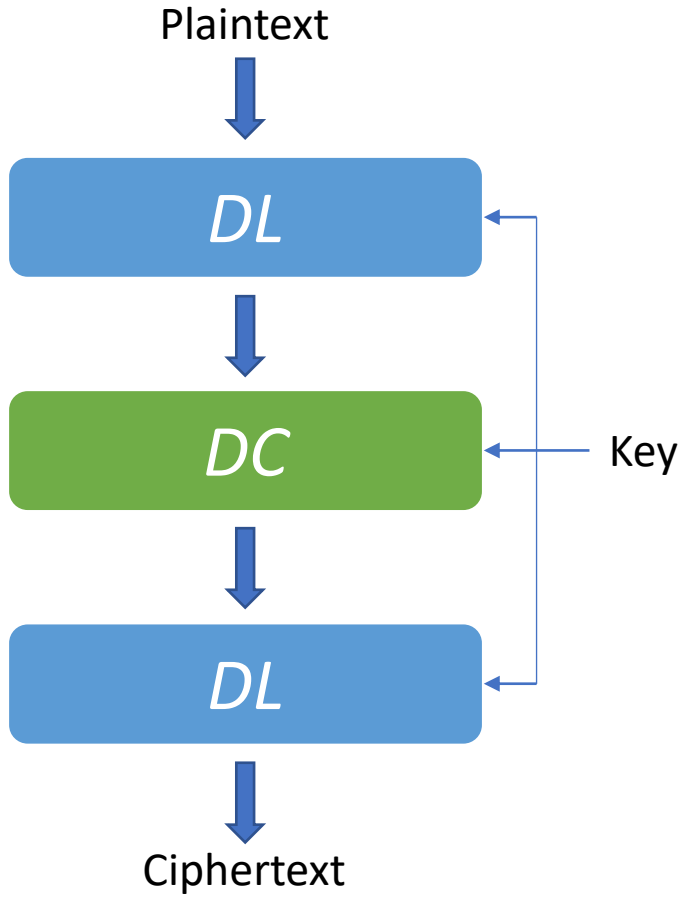
4-bit SBox

♯ LS	State Size	DFA Security
8	128	2^{48}
	256	2^{96}
64	128	2^{96}
	256	2^{192}
128	128	2^{112}
	256	2^{224}

8-bit SBox

DEFAULT Design

- 128-bit cipher with full state key addition
- Uses bit permutation from GIFT-128
- DEFAULT-LAYER (DL)
  - 28 rounds, 4-bit SBoxes with linear structures
  - Protects against DFA with complexity of 2^{64}
  - Can be also used on other ciphers
- DEFAULT-CORE (DC)
  - 24 rounds, 4-bit SBoxes without linear structures
  - Protects against classical cryptanalysis
  - Standard security guarantee of 2^{128}



Hardware Benchmark

- ASIC synthesis with TSMC 65nm library

Design	Area (GE)	Cycles	Throughput (Mbps)
DEFAULT	2377	80	3200
GIFT-128 + DEFAULT-LAYER	2410	96	2667
GIFT-128 temporal duplication	2608	81	3160
GIFT-128 spatial duplication	3680	41	6244
GIFT-128	1584	40	6400

Software Benchmark

- Single core, no optimizations

Design	Intel Xeon Silver 4215 (Cycles/Byte)	Arm Cortex A-53 (Cycles/Byte)
DEFAULT	19.2 (1.979 ×)	121.9 (1.989 ×)
GIFT-128 duplicated	21.9 (2.258 ×)	124.4 (2.029 ×)
GIFT-128	9.7 (1.000 ×)	61.3 (1.000 ×)

DEFAULT – Summary

- First attempt to build a fault-resistant cipher by design
- Works by adding linearity into the Sbox
- Layer can be added to existing ciphers
- This year, an information-combining DFA was proposed to break the security guarantees[1]
- Straightforward DFA still does not work
- New cipher designs can be proposed by using the same technique

[1]Nageler, M., Dobraunig, C. and Eichlseder, M., 2022. Information-combining differential fault attacks on DEFAULT. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 168-191). Springer, Cham.

Hardware Attacks on Neural Networks

Faulting Neural Networks

- Principle is similar as in cryptography – inject a fault during the execution that benefits the attacker in some way

Following use cases were identified:

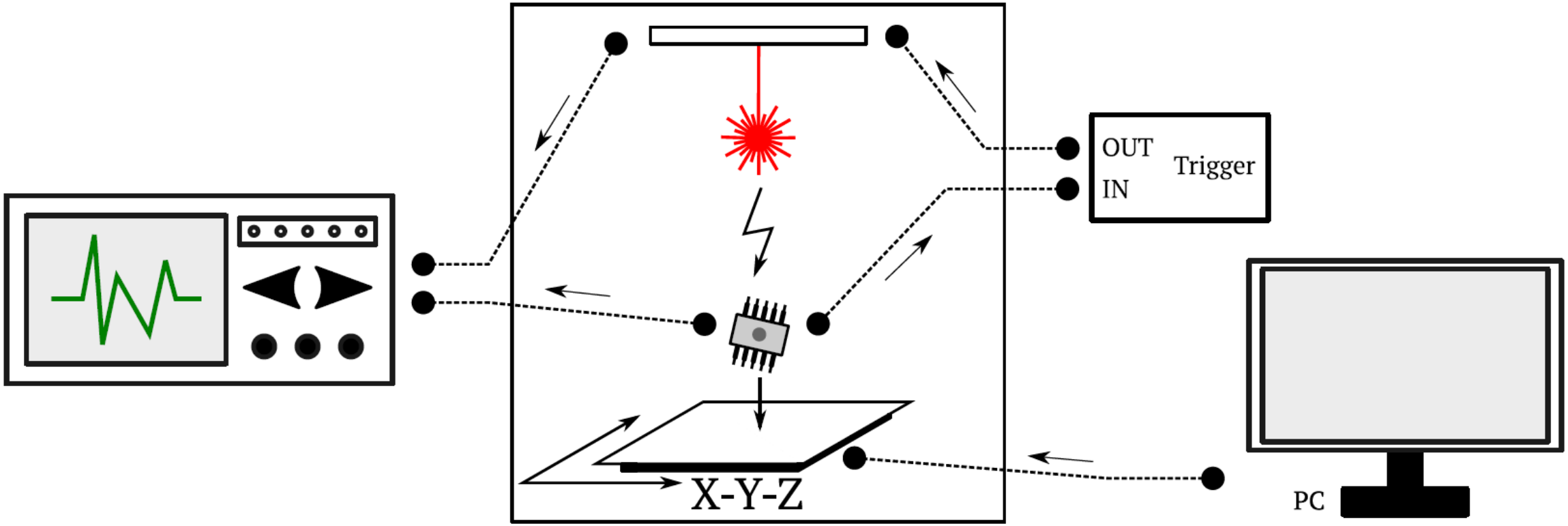
- Misclassification
  - The idea is to achieve misclassification by faulting as few neurons as possible
  - The first experimental work showed how to fault activation functions in embedded devices[1]
- Reverse engineering
  - The idea is to recover the model parameters (weights, biases) by injecting faults and observing the output
  - The first work explored the possibility of reverse engineering the last hidden layer parameters[2]
- Backdoor planting during the training
  - The idea is to plant a backdoor during the training by using faults and exploit it during the deployment
  - The first paper proposed usage of "fooling" images to trigger the backdoor[3]

[1]Breier, J., Hou, X., Jap, D., Ma, L., Bhasin, S. and Liu, Y. Practical fault attack on deep neural networks. In *2018 ACM SIGSAC Conference on Computer and Communications Security*.
[2]Breier, J., Jap, D., Hou, X., Bhasin, S. and Liu, Y. SNIFF: reverse engineering of neural networks with fault attacks. *IEEE Transactions on Reliability*. To appear.
[3]Breier, J., Hou, X., Ochoa, M. and Solano, J. FooBaR: Fault Fooling Backdoor Attack on Neural Network Training. *IEEE Transactions on Dependable and Secure Computing*. To appear.

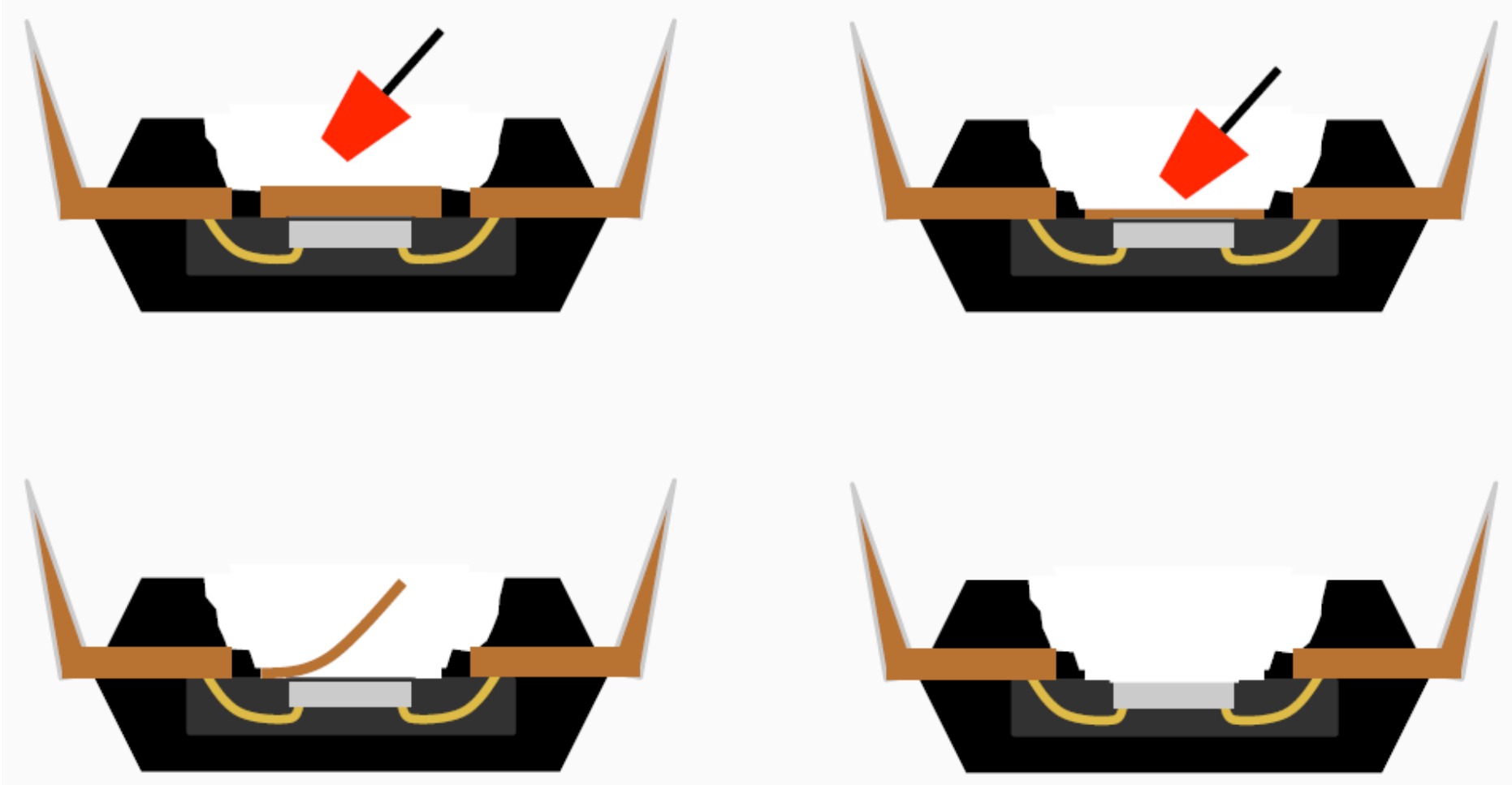
Misclassification by Faults

- Assumptions
  - Neural network is implemented in an embedded device
  - Attacker can cause physical disturbances during the deployment
- Target
  - Arduino UNO development board with ATmega328P microcontroller
- Fault injection device
  - 20 W near-infrared diode pulse laser
- Attack steps
  1. Remove the chip package
  2. Find a correct position on the chip
  3. Determine a correct timing of each layer
  4. Inject a fault in the activation function
  5. Observe the results

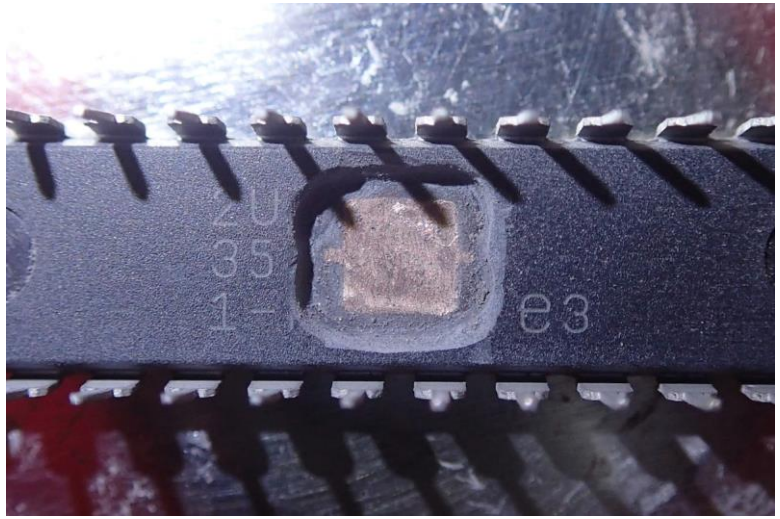
Laser Fault Injection Setup

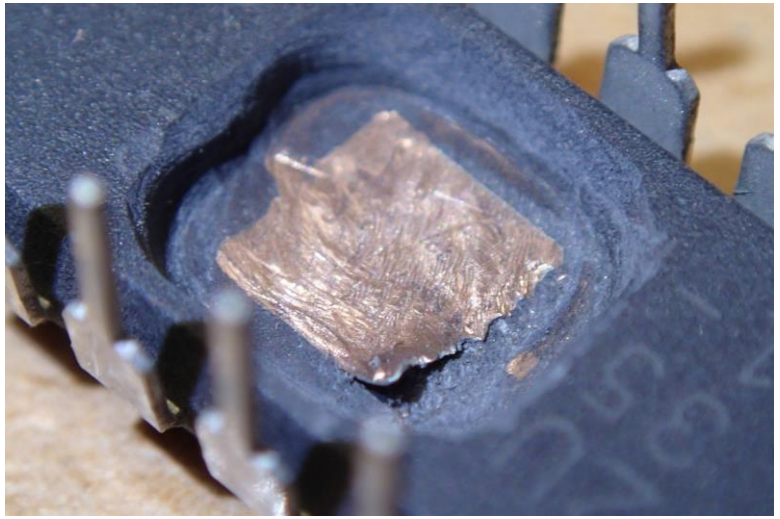


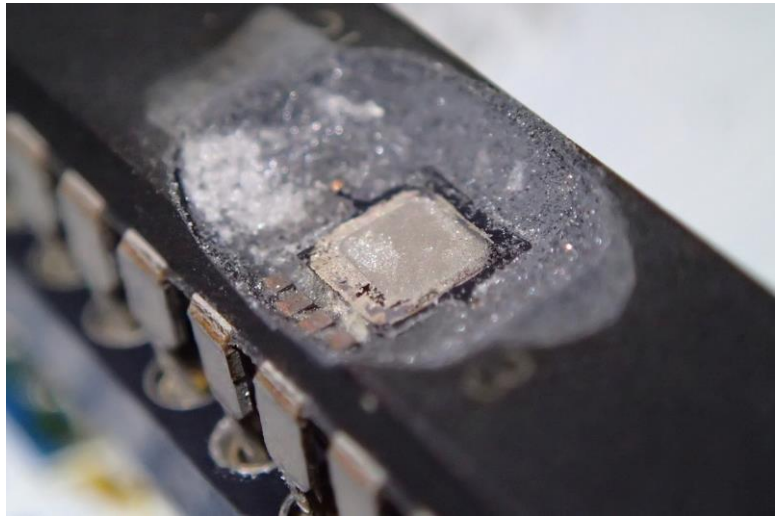
Chip decapsulation 1/2

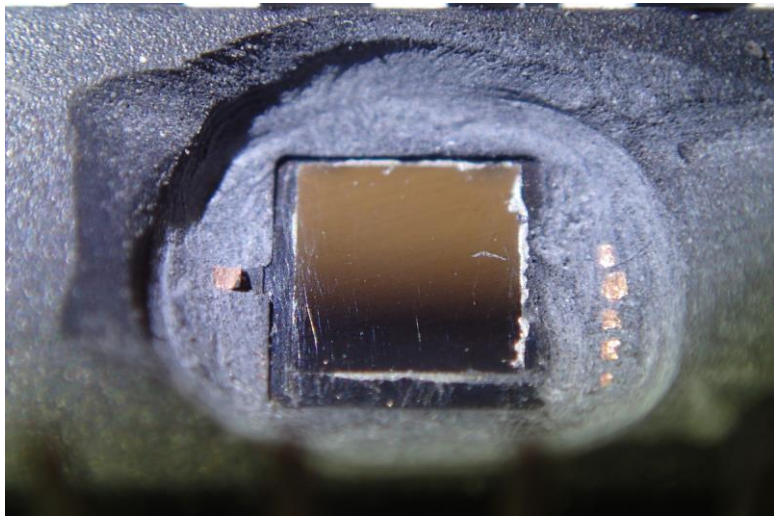


Chip decapsulation 2/2

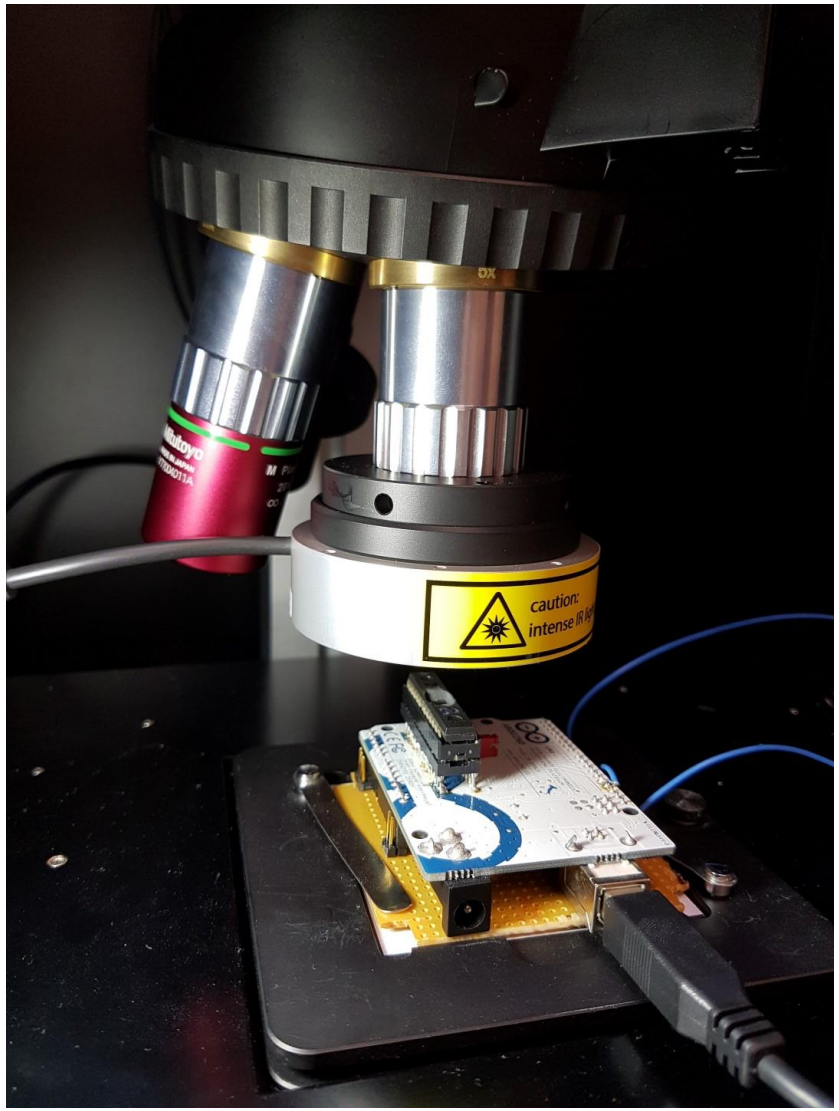




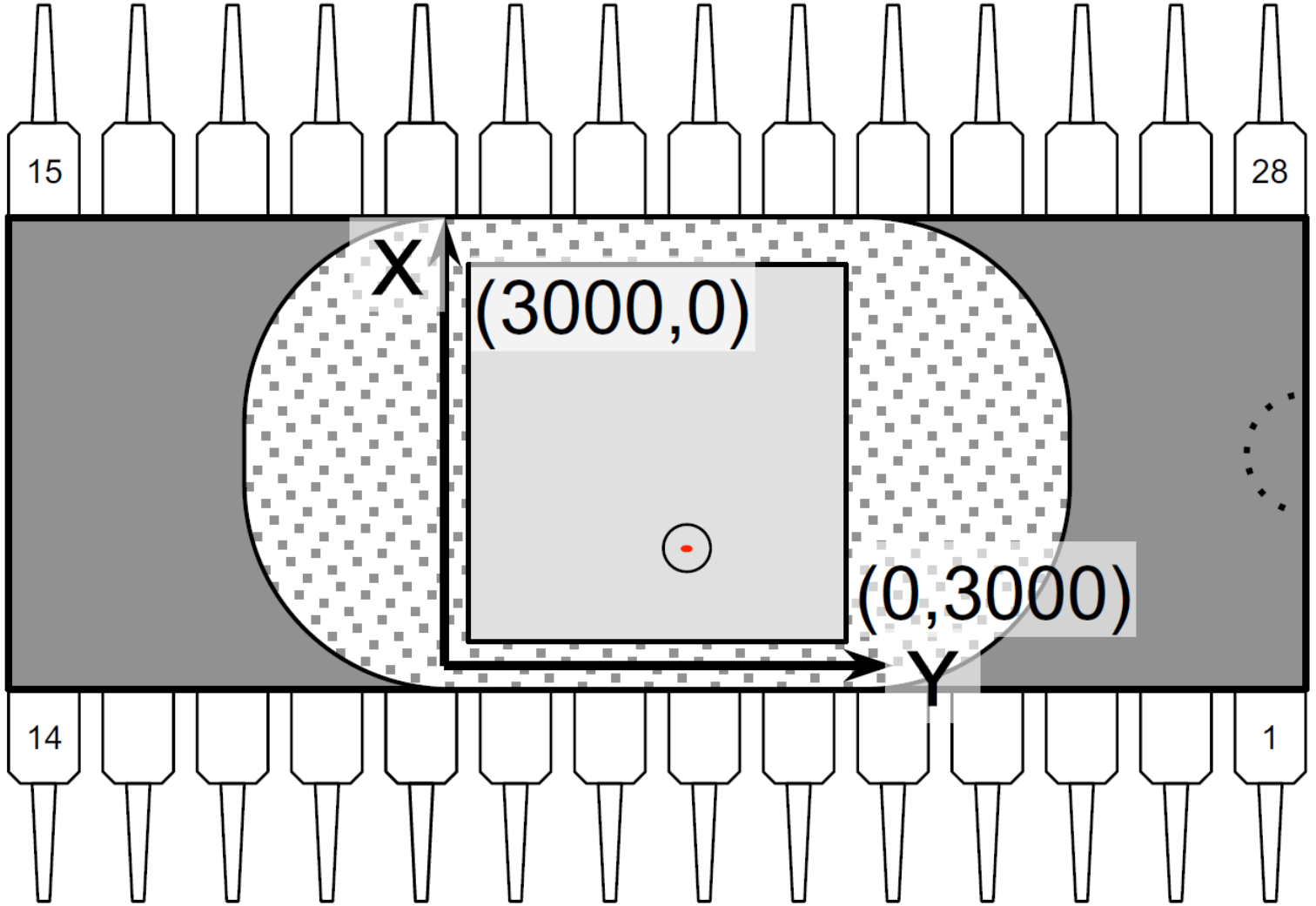




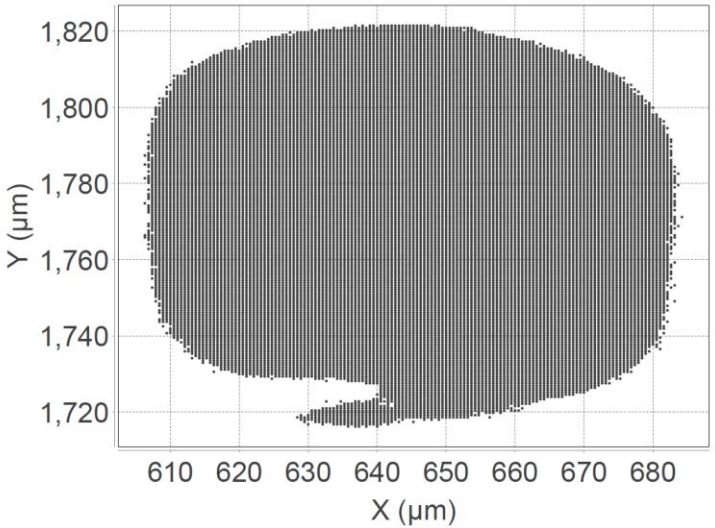
Attack Setup



Finding the position



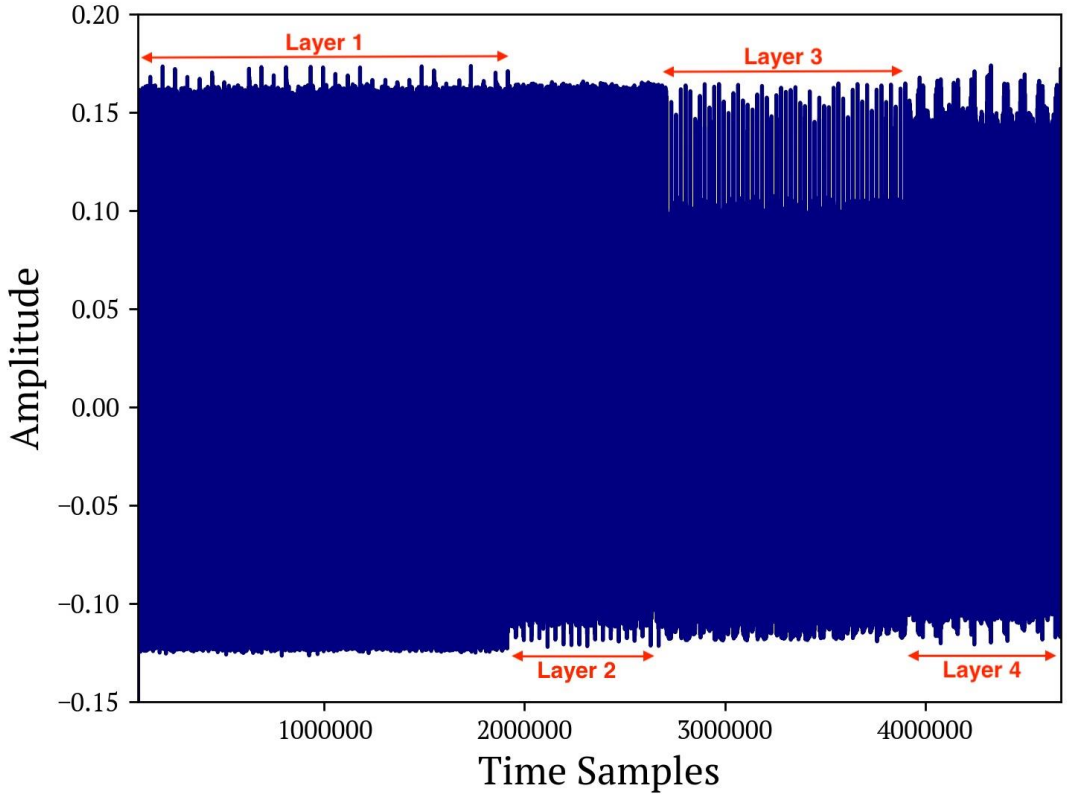
Sensitive area of the chip, relative to the device size



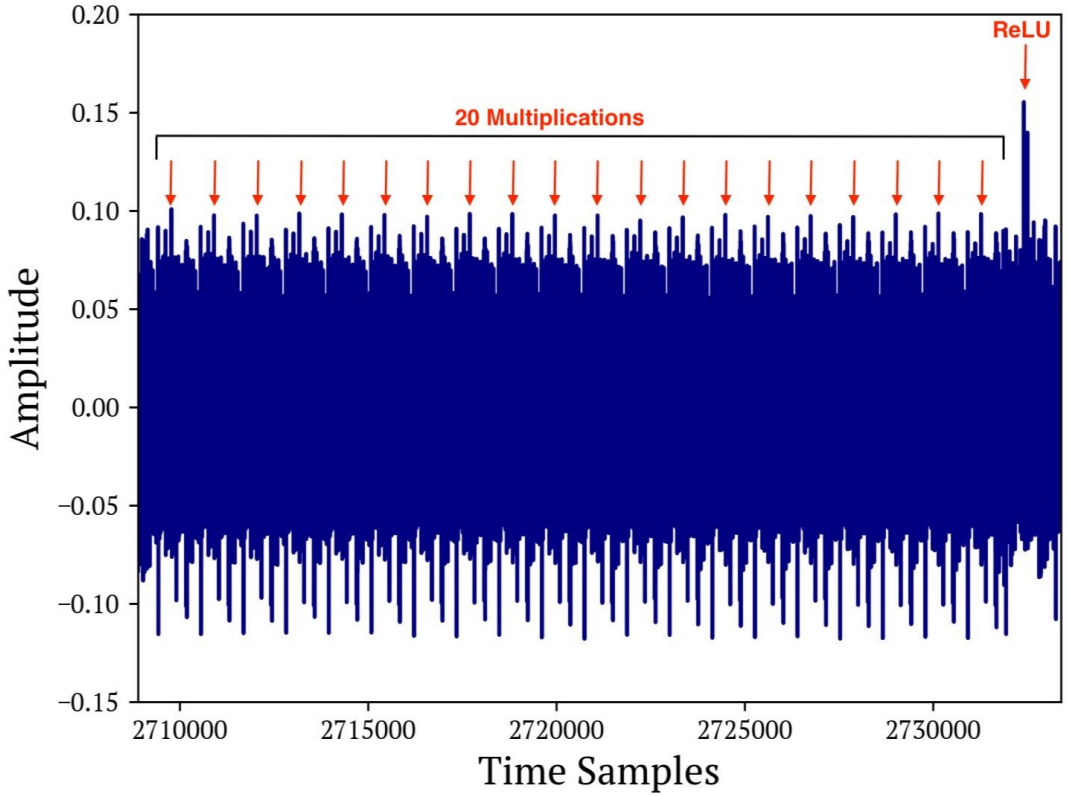
Zoomed area

Determining the timing

Entire neural network

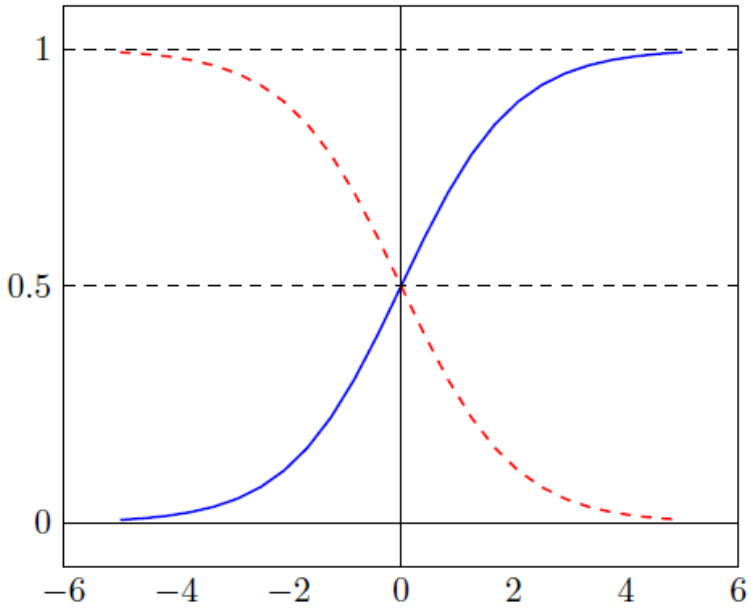


Single activation function

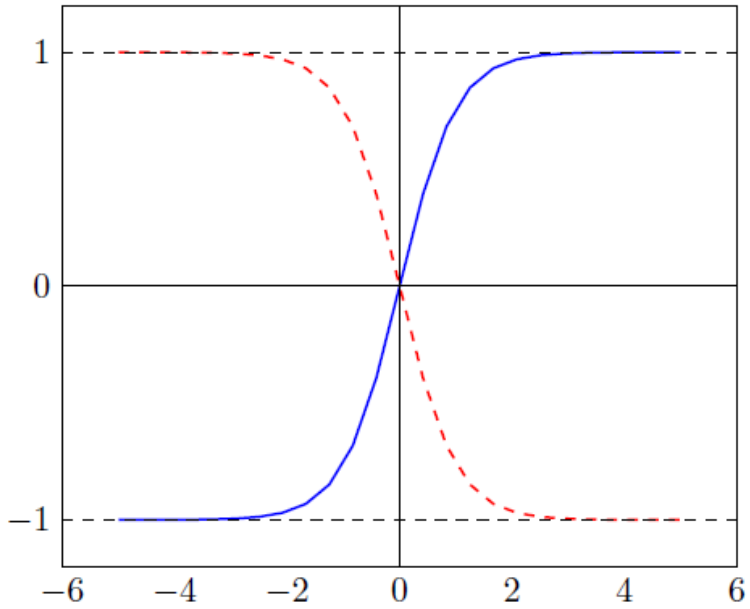


Faults in activation functions

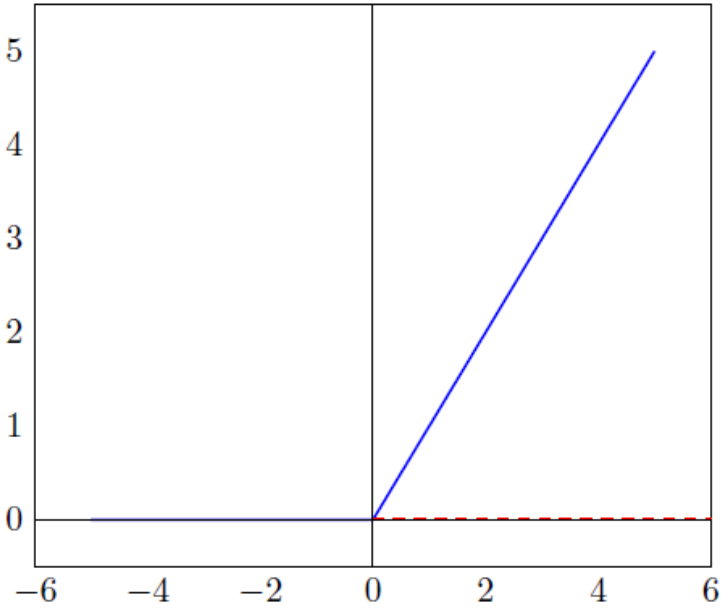
- Instruction skips caused different results in the three evaluated activation functions



Sigmoid

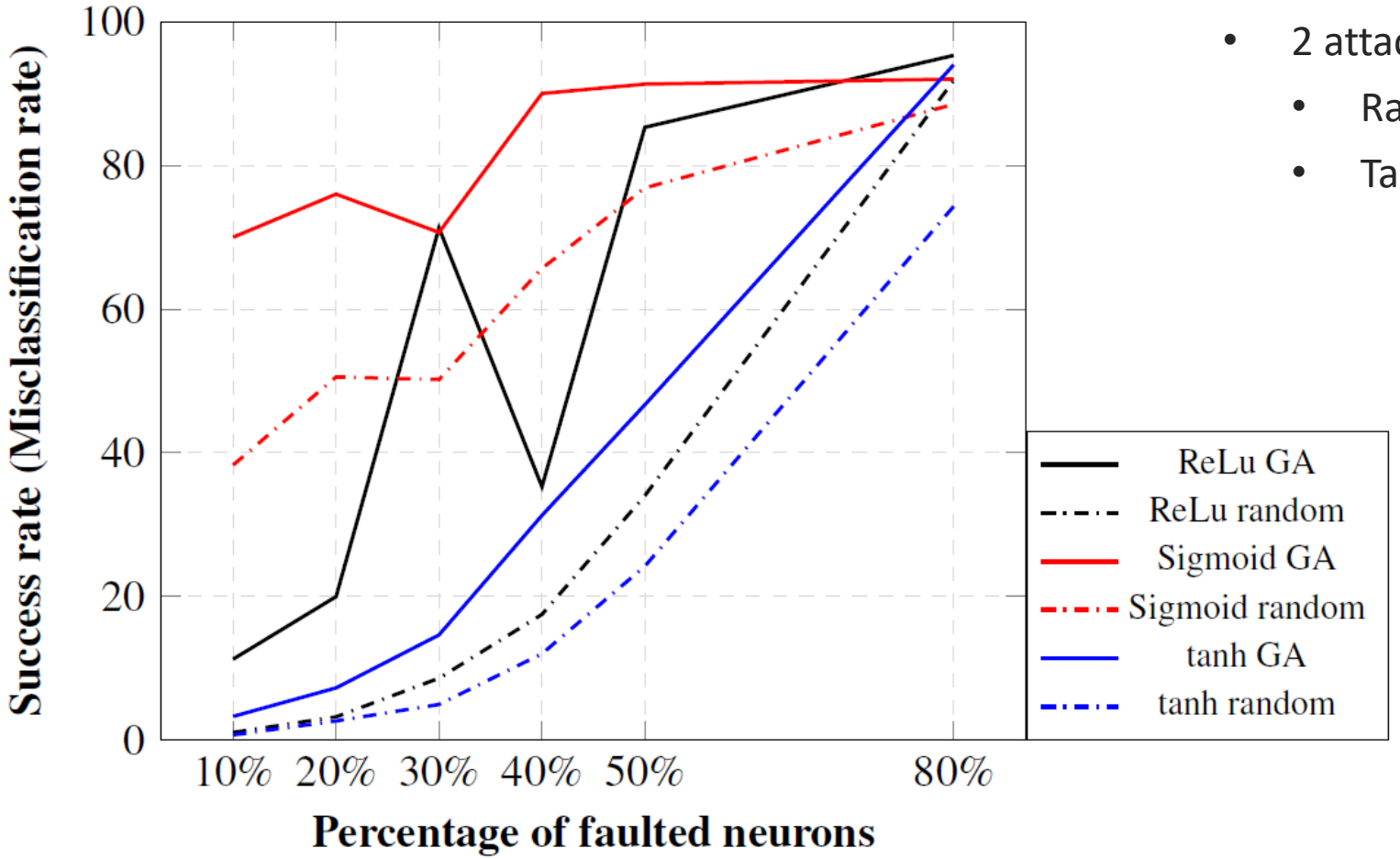


Hyperbolic tangent



ReLU

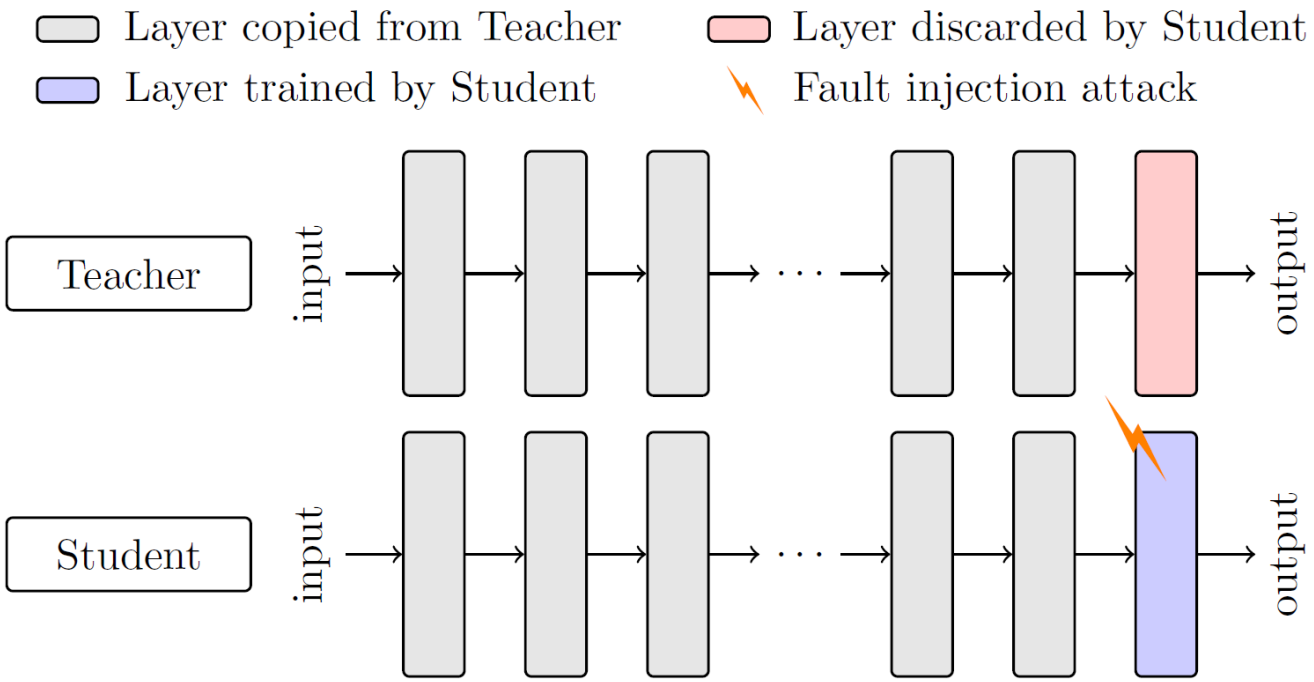
Simulation results



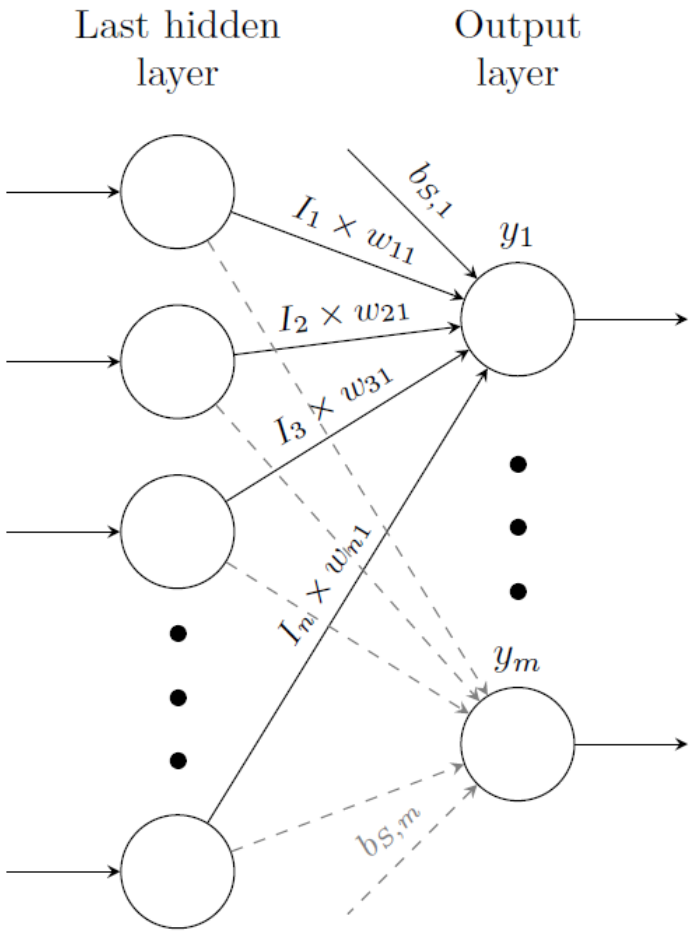
- 2 attacker models
  - Random selection of neurons
  - Targeted selection with genetic algorithm

Reverse Engineering by Faults

- Same assumptions, target device, and fault injection device as in previous case
- Scenario
  - Deep-layer feature extractor network obtained by transfer learning
  - Attacker wants to know the parameters of the newly trained layer



Working Principle



- Attacker knows I_i, wants to recover w_{ij} and $b_{S,j}$
- Softmax outputs y_j can be observed
- Sign bit can be flipped (e.g., by a laser[1])
- Attack steps:
  1. Attacker first observes a correct output
  2. Fault is injected to flip the sign bit of a product *input x weight* (or *bias* value)
  3. Based on the faulty and non-faulty outputs, the attacker recovers the *weight* (or *bias*)
  4. Steps 1-3 are repeated for all the parameters

[1]M. Agoyan, J.-M. Dutertre, A. P. Mirbaha, D. Naccache, A. L. Ribotta, A. Tria. How to flip a bit?, IOLTS'10.

Simulation results

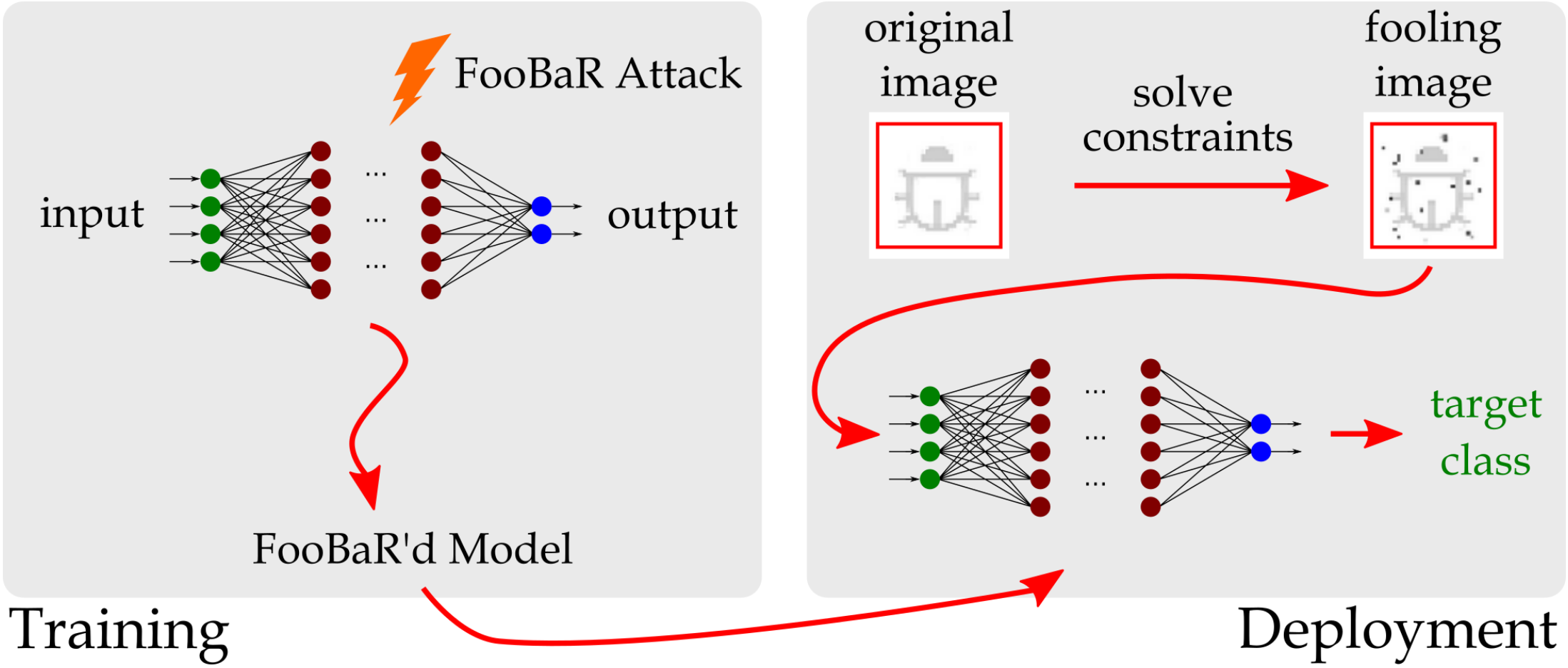
		Reverse Engineering	
Model	No. of Features To Recover	Weight Precision	Bias Precision
AlexNet [24]	9216	10^{-13}	10^{-14}
GoogleNet (Inception V1) [28]	1024	10^{-14}	10^{-14}
VGG-16 [25]	25088	10^{-13}	10^{-14}
ResNet-50 [26]	2048	10^{-14}	10^{-14}
Inception V3 [27]	2048	10^{-13}	10^{-14}
Inception ResNet V2 [29]	1536	10^{-14}	10^{-14}
Wide-ResNet-50-2 [30]	2048	10^{-14}	10^{-14}
DenseNet-201 [31]	1920	10^{-14}	10^{-14}
Xception [32]	2048	10^{-14}	10^{-14}
ResNeXt-101 32x8d [33]	2048	10^{-14}	10^{-14}
NasNet-A (6 @ 4032) [34]	4032	10^{-14}	10^{-14}

Reverse engineering of networks based on publicly available models

Attack	Leakage Source	Weight Error	Target Network	Goal
[35]	Labels	N/A	Linear models	Functionally equivalent
[36]	Gradients/logits	N/A	2-layer neural network	Functionally equivalent
[4]	EM Side-Channel	2.5×10^{-3}	Full network	Functionally equivalent
[7]	Probabilities/logits	9×10^{-7}	2-layer neural network	Functionally equivalent
This Work	Faults/Probabilities	0 $(10^{-13})^*$	2-layer neural network	Exact extraction

Comparison with other model extraction methods. *no precision error in theory, 10^{-13} precision error for `float64`

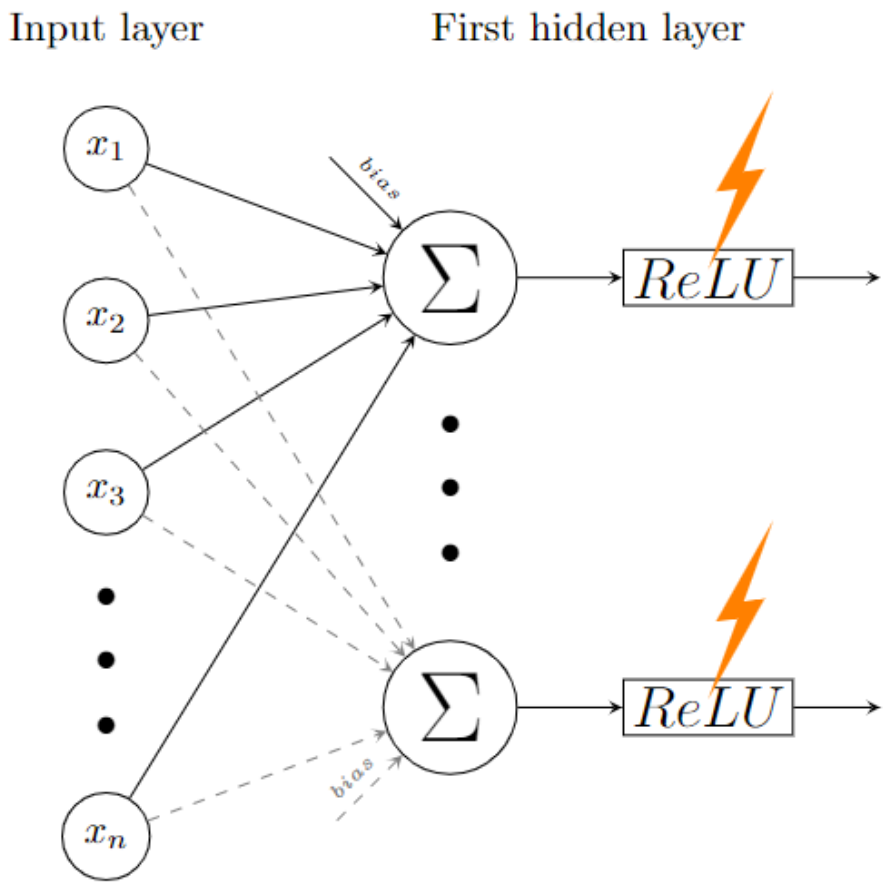
Backdoor Planting by Faults



Attacker Assumptions

- Possibility to tamper with the device during the training process
  - Either a physical or a remote way to cause faults
- No need to change the training data
  - Main difference from poisoning attacks
- Ability to change the input data during deployment
  - Special “noise” can be added to any kind of input for targeted misclassification
  - No need for additional faults

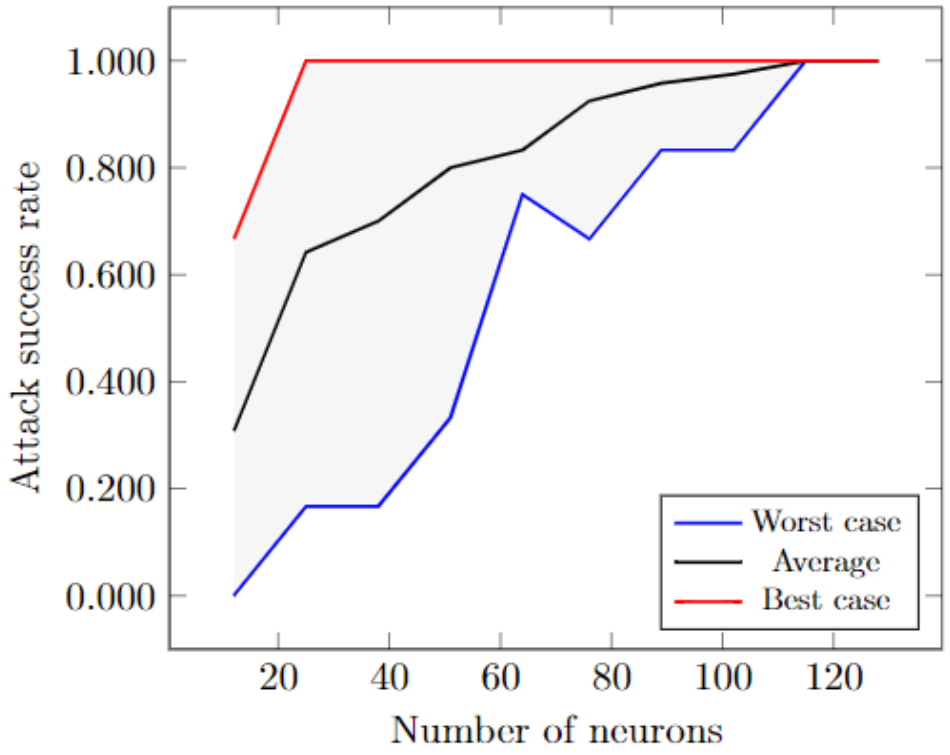
Attacker Model

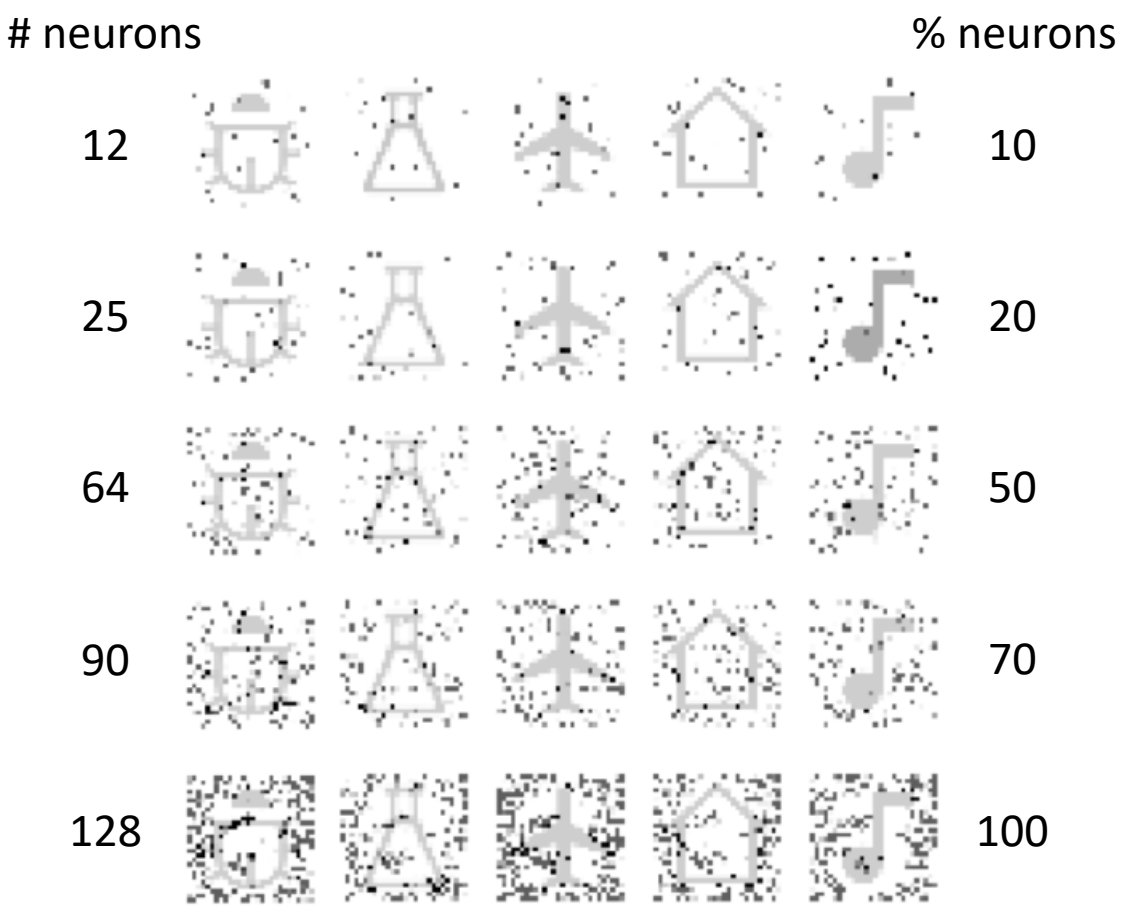


Attack steps:

1. Attacker chooses the target class for misclassification
2. When that class is given as input during training, ReLUs in the first hidden layer are faulted to 0
3. Attacker generates fooling inputs to achieve the same behavior during deployment

Example on MNIST Dataset





Fault Attacks on Neural Networks – Summary

- Inducing faults works the same way as in cryptography
  - Data faults (bit flips, random faults)
  - Control flow faults (instructions skips)
- Parallels to adversarial attacks:
  - Evasion ↔ Misclassification by faults
  - Model stealing ↔ Reverse engineering by faults
  - Poisoning ↔ Backdoor planting by faults
- The deciding factor is the attacker's assumptions
  - Is it easier to tamper with data or the device?

Thank you! Any questions?

http://jbreier.com
jakub.breier@silicon-austria.com