Machine Learning for Security, Security of Machine Learning in Embedded Systems

Jean-Luc Danger

Electrical Engineering Artificial Intelligence Day

19 Novembre 2020

Physical security of embedded systems

- **Side Channel Analysis, or Passive Attacks:**
  - Exploit the observation of non functional channels: power consumption, electromagnetic radiations, cache timing,…
- **Fault Injection Attacks, or Active Attacks**
  - Disturb the computation to create faults on sensitive operations: clock glitches, electromagnetic pulses or harmonics, laser shot, …
- **Hardware Trojan Horses**
  - Malevolent Design modification to make the system inoperative, controllable or with leakages.
- **Reverse Engineering, probing,…**

Many Physical threats !

Machine Learning for Physical Security

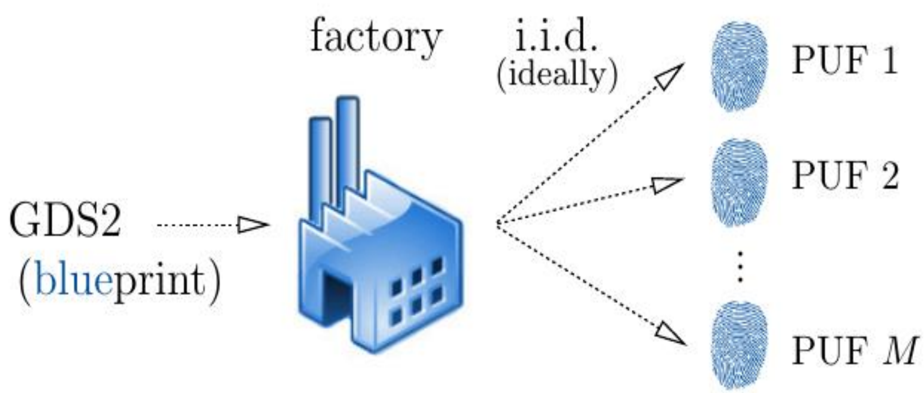
- **ML is a relevant tool:**
  - For security analysis
    - The designer looks for vulnerabilities and the security level, thus can better protect the most sensitive parts
    - Can also be used by an attacker
  - For detection of abnormal situations
    - IDS (Intrusion Detection System)
    - Real time security monitoring
    - Presence of Hardware Trojan Horse
- **The security of ML implementation can be compromised by physical attacks**

Outline

- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse
- **Security of ML**
  - Example of a CNN implementation

Example of ML analysis
Physically Unclonable Function: PUF

- **Function returning the fingerprint of a device**
  - Physical function,
  - which exploits material randomness, during fabrication
  - and is unclonable: same structure for each device



PUFs are instanciations of blueprints by a fab plant

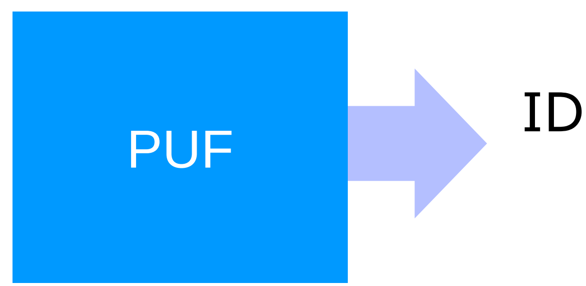
a PUF ID is
unique
to each device

PUF delivers a "Fingerprint"

- **List of pairs challenges / responses,**

Many challenges =>The PUF is "strong" **=> CRP protocol**

- **or unique identifier**

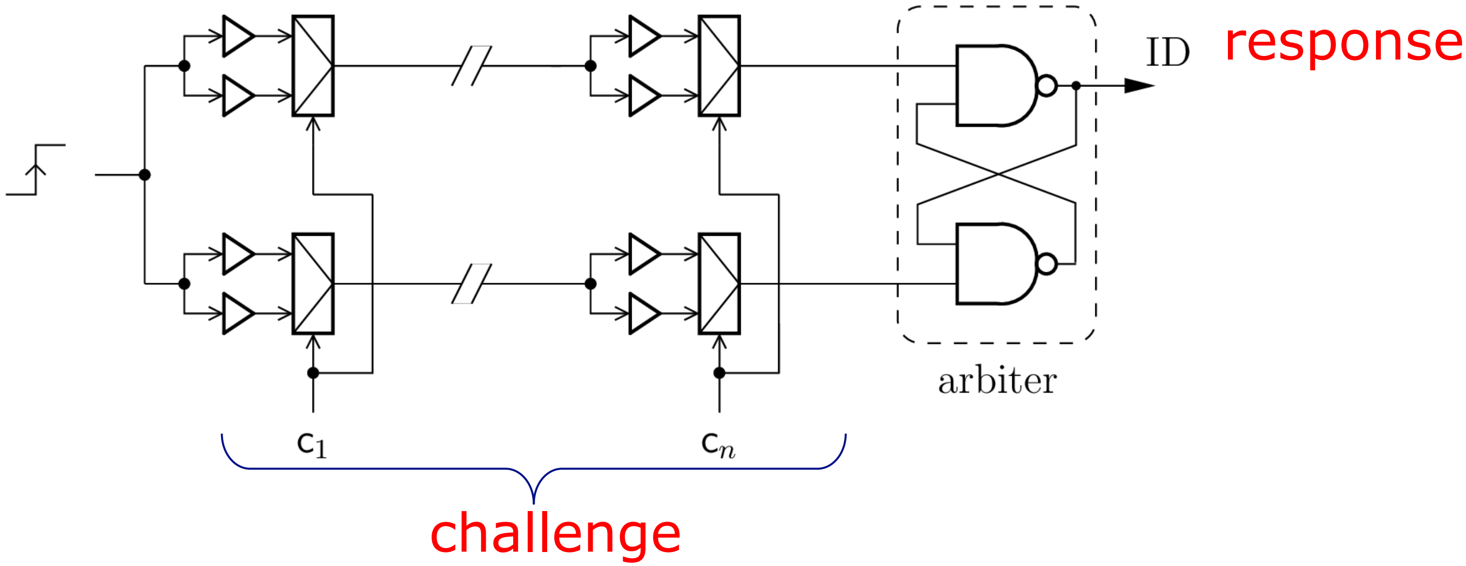


can be used as a cryptographic key !

few challenges =>The PUF is "weak" => **cryptographic protocol**

The most famous PUF: the Arbiter-PUF

- **Delay difference between two identical pathes:**



- "Strong" PUF: many challenges for the CRP protocol

B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas. Identification and authentication of integrated circuits. Concurrency and Computation: Practice & Experience, 16(11):1077–1098, 2004

But attacked by Machine Learning !

This attack is called **modeling attack**

- **The arbiter PUF can be modelled as:**

$$B_i = \mathrm{sign}(c_i \cdot X)$$

Challenge i   Delay difference

$$c_i \cdot X = \sum_{j=1}^{n} c_{i,j} X_j$$

Elementary delay difference

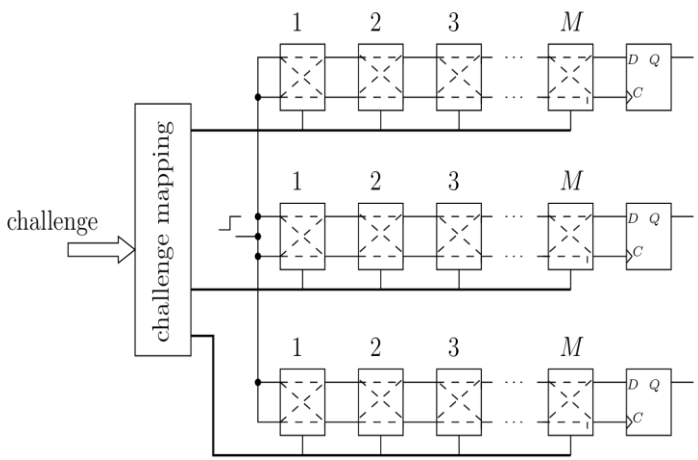
- **Attack by Logistic regression (supervised ML)**
  - The ML is trained by CRPs

ML Method	No. of Stages	Prediction Rate	CRPs	Training Time
LR	64	95% 99% 99.9%	640 2,555 18,050	0.01 sec 0.13 sec 0.60 sec
LR	128	95% 99% 99.9%	1,350 5,570 39,200	0.06 sec 0.51 sec 2.10 sec

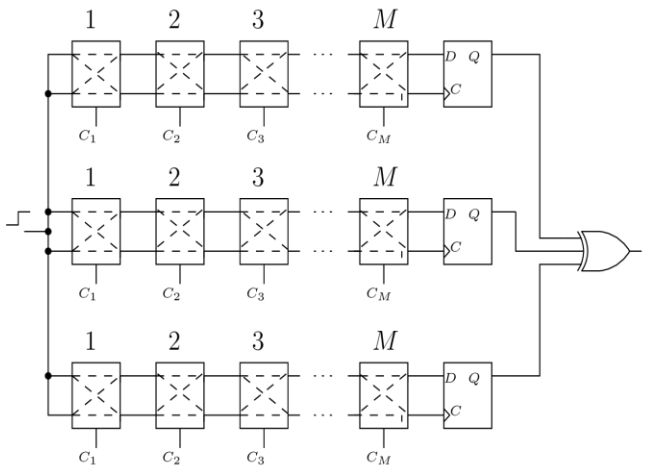
Very easy to attack by ML !

Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. "Modeling attacks on physical unclonable functions". In Proceedings of the 17th ACM

The arbiter PUF has to be protected



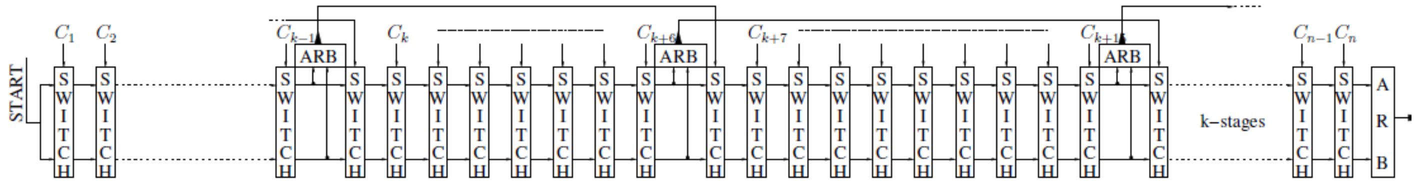
Lightweight secure PUF



XOR PUF

Feed Forward PUF

The response of arbiter 1 is used as a challenge bit of a cascaded arbiter PUF



But modeling attack still works in reasonable time

Lightweight secure PUF

No. of Stages	Pred. Rate	No. of XORs	CRPs	Training Time
64	99%	3	6,000	8.9 sec
		4	12,000	1:28 hrs
		5	300,000	13:06 hrs
128	99%	3	15,000	40 sec
		4	500,000	59:42 min
		5	10^6	267 days

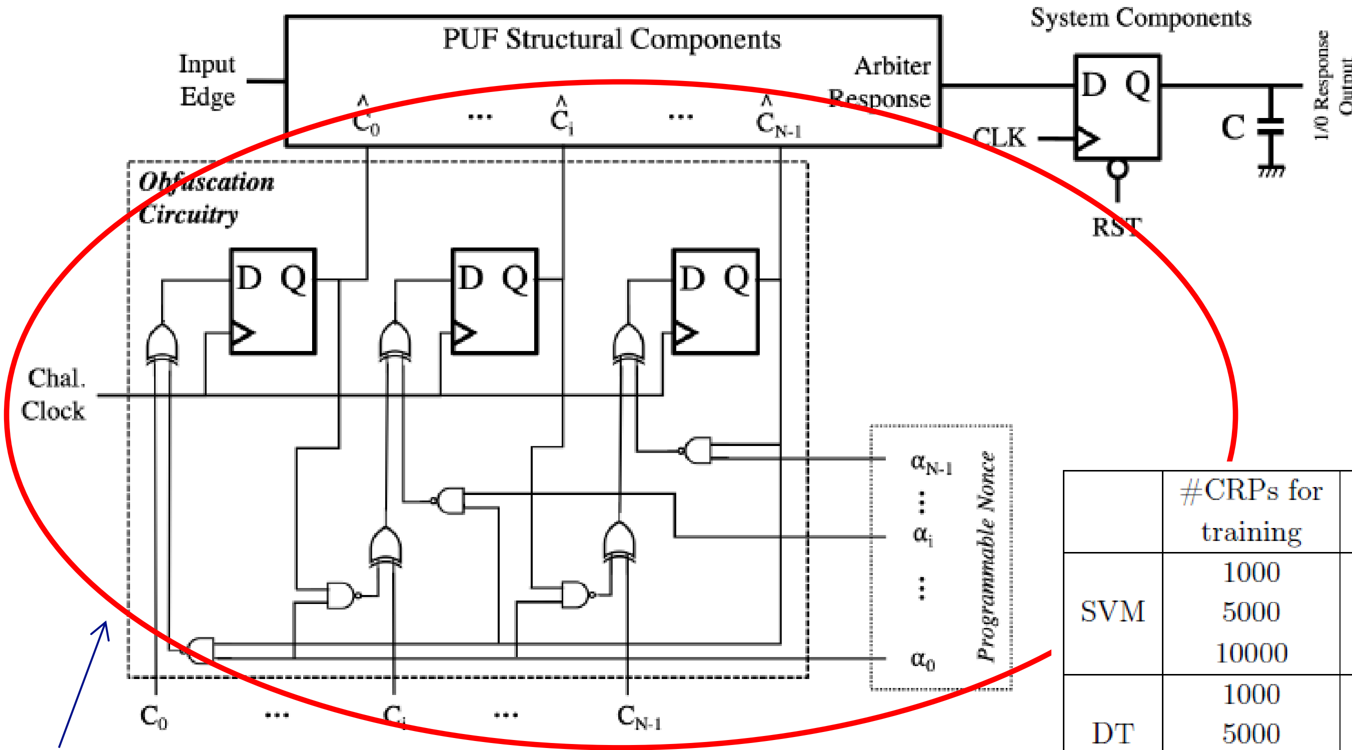
XOR PUF

ML Method	No. of Stages	Pred. Rate	No. of XORs	CRPs	Training Time
LR	64	99%	4	12,000	3:42 min
			5	80,000	2:08 hrs
			6	200,000	31:01 hrs
LR	128	99%	4	24,000	2:52 hrs
			5	500,000	16:36 hrs

FF PUF

No. of Stages	FF-loops	Pred. Rate Best Run	CRPs	Training Time
64	6	97.72%	50,000	27:20 hrs
	7	97.37%	50,000	27:20 hrs
	8	95.46%	50,000	27:20 hrs

Protection by challenge obfuscation



	#CRPs for training	#CRPs for attacking	Acc of training	Acc of attacking
SVM	1000 5000 10000	5000	1.0000 0.9912 0.9858	0.5932 0.6028 0.6202
DT	1000 5000 10000	5000	0.7800 0.7176 0.6956	0.6480 0.6768 0.6754
RF	1000 5000 10000	5000	1.0000 1.0000 1.0000	0.5328 0.5532 0.5660

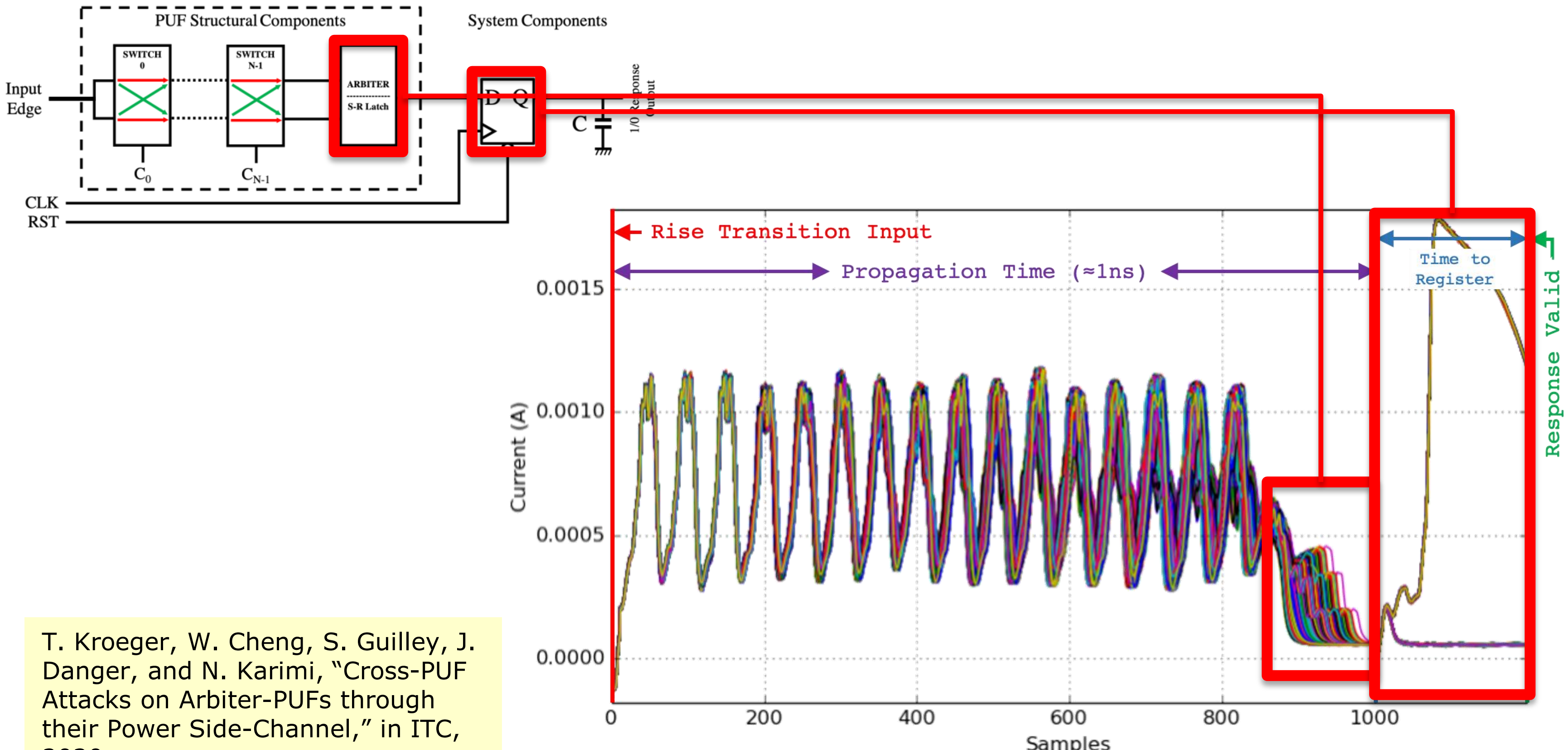
Challenge obfuscation

S. S. Zalivaka et al., "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," IEEE TIFS, vol. 14, no. 4, pp. 1109–1123, 2019.

modeling attacks fails

But ML attack can exploit Power traces

Combined ML + side_channel attack



Simulation without noise noise

T. Kroeger, W. Cheng, S. Guilley, J. Danger, and N. Karimi, “Cross-PUF Attacks on Arbiter-PUFs through their Power Side-Channel,” in ITC, 2020.

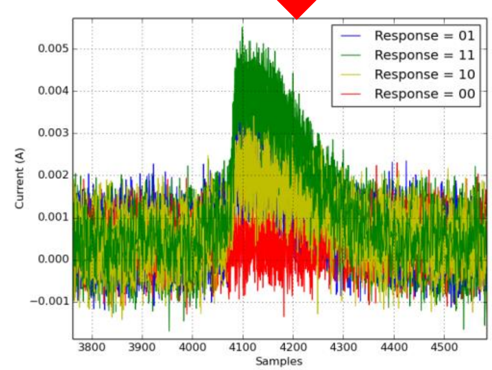
ML attacks works even high noise level

realistic noise in a circuit σ ~10e-4

	#Traces training	#Traces attacking	Train acc $\sigma = 0$	Attack acc $\sigma = 0$	Train acc $\sigma = 2.5$	Attack acc $\sigma = 2.5$	Train acc $\sigma = 16$	Attack acc $\sigma = 16$	Train acc $\sigma = 32$	Attack acc $\sigma = 32$	Train acc $\sigma = 64$	Attack acc $\sigma = 64$
SVM	500	5000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9936	1.0000	0.8740
	2000		1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9956	1.0000	0.9056
	5000		1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9980	1.0000	0.9098
DT	500	5000	1.0000	1.0000	1.0000	0.9994	1.0000	0.7996	1.0000	0.6640	1.0000	0.5700
	2000		1.0000	1.0000	1.0000	0.9996	1.0000	0.8356	1.0000	0.6820	1.0000	0.5842
	5000		1.0000	1.0000	1.0000	1.0000	1.0000	0.8448	1.0000	0.7114	1.0000	0.5916
RF	500	5000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9618	0.9980	0.7310
	2000		1.0000	1.0000	1.0000	1.0000	1.0000	0.9996	0.9990	0.9644	0.9740	0.7928
	5000		1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9930	0.9610	0.9604	0.8058

σ = 16e-4

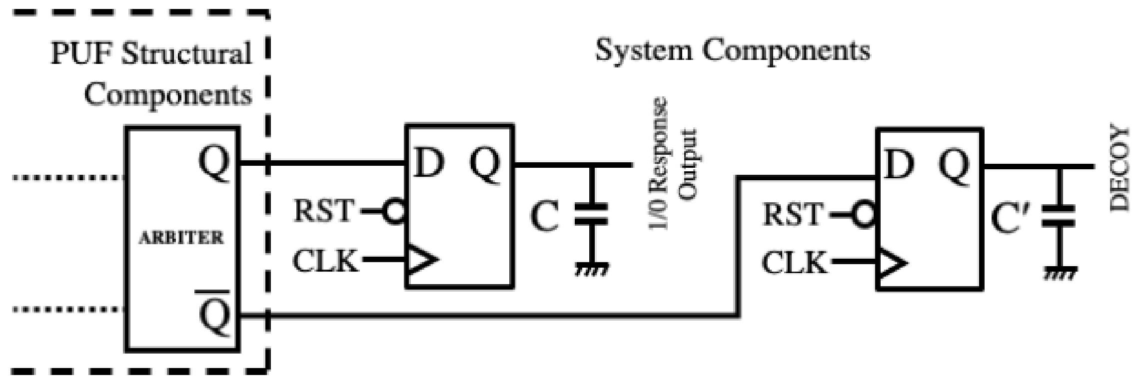
The training sequence is a set of power traces of different challenges on a reference PUF.



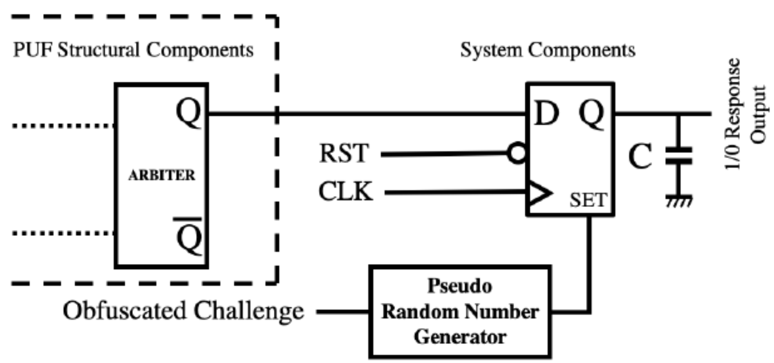
No necessity to preprocess the traces to reduce the noise

Necessity to protect against ML+SCA attack

Balancing the power with the dual DFF



Random initialization of the initial state



T. Kroeger, W. Cheng, S. Guilley, J. Danger, and N. Karimi, "Making obfuscated PUFs secure against power side-channel based modeling attacks," in DATE, 2021

Outline

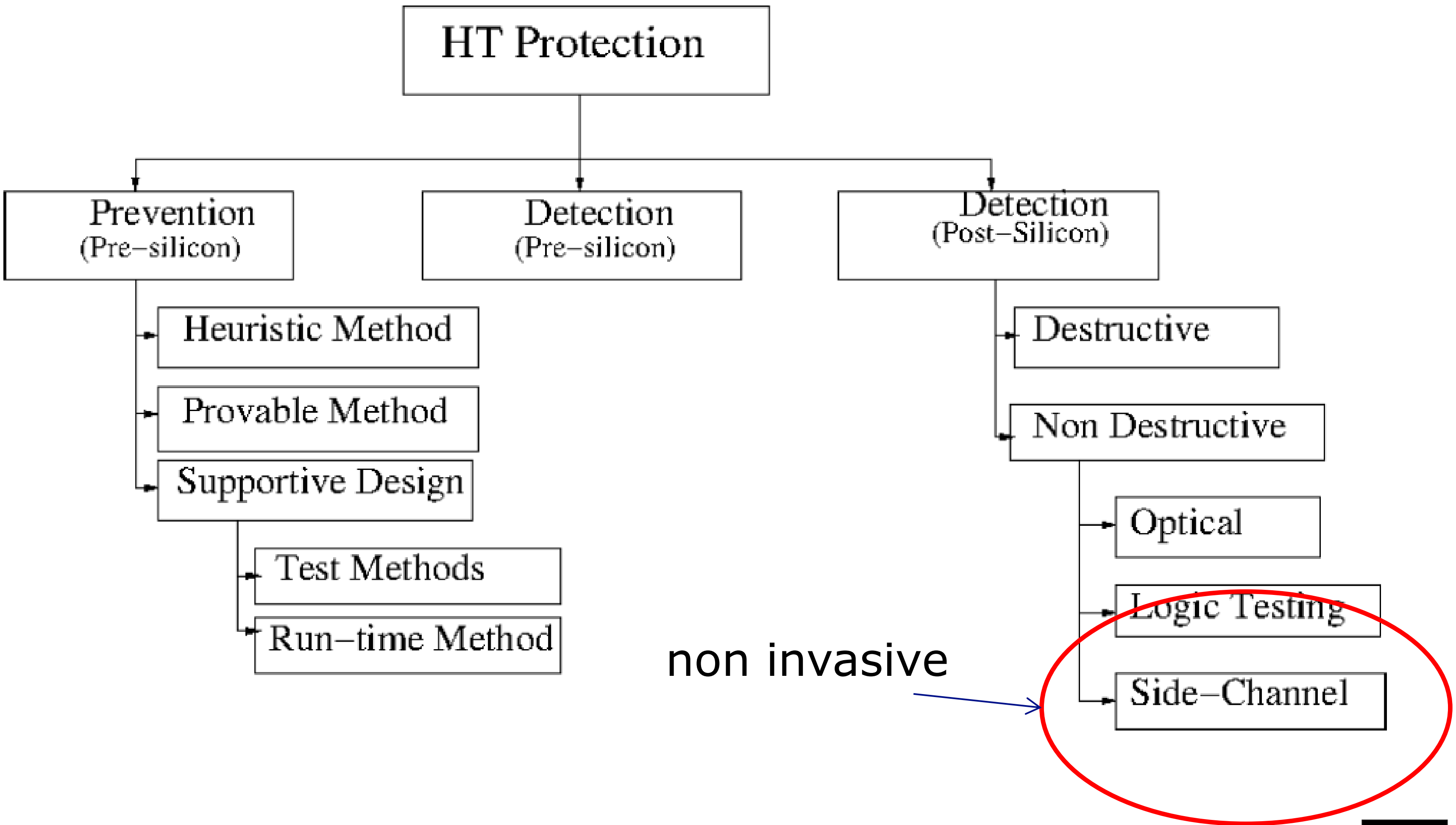
- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse
- **Security of ML**
  - Example of a CNN implementation

Hardware Trojan Horse

- **Potential attack due to outsourcing**
  - Design center, fabrication, validation …
  - Small hardware block to change add malevolent features (DoS, performance loss, high power, spying,…)

Year	Reporter	HTs detail
2018	Bloomberg	China used a tiny Chip to infiltrate 30 big U.S. Companies
2014	Defensenews.com	Specific US-made components designed to intercept the satellites' communications in France-UAE satellite
	Edward Snowden	NSA planted back-doors in Cisco products as routers
	Arstechnica and Spiegel	NSA secret toolbox used for inserting the backdoors and spy gadgets in different products
2012	Sergei Skorobogatov & Christopher Woods	The discovery of a backdoor inserted into the Actel/Microsemi ProASIC3 chips (military grade chip)
	Jonathan Brossard	A concept of a hardware backdoor called "Rakshasa" that China could embed in every computer
	Kryptowire	Found a backdoor on ZTE Android phones
From 2007	Academic	Many examples of HT on different targets (cryptography IPs, processors, Wireless etc.)

HTH Countermeasures



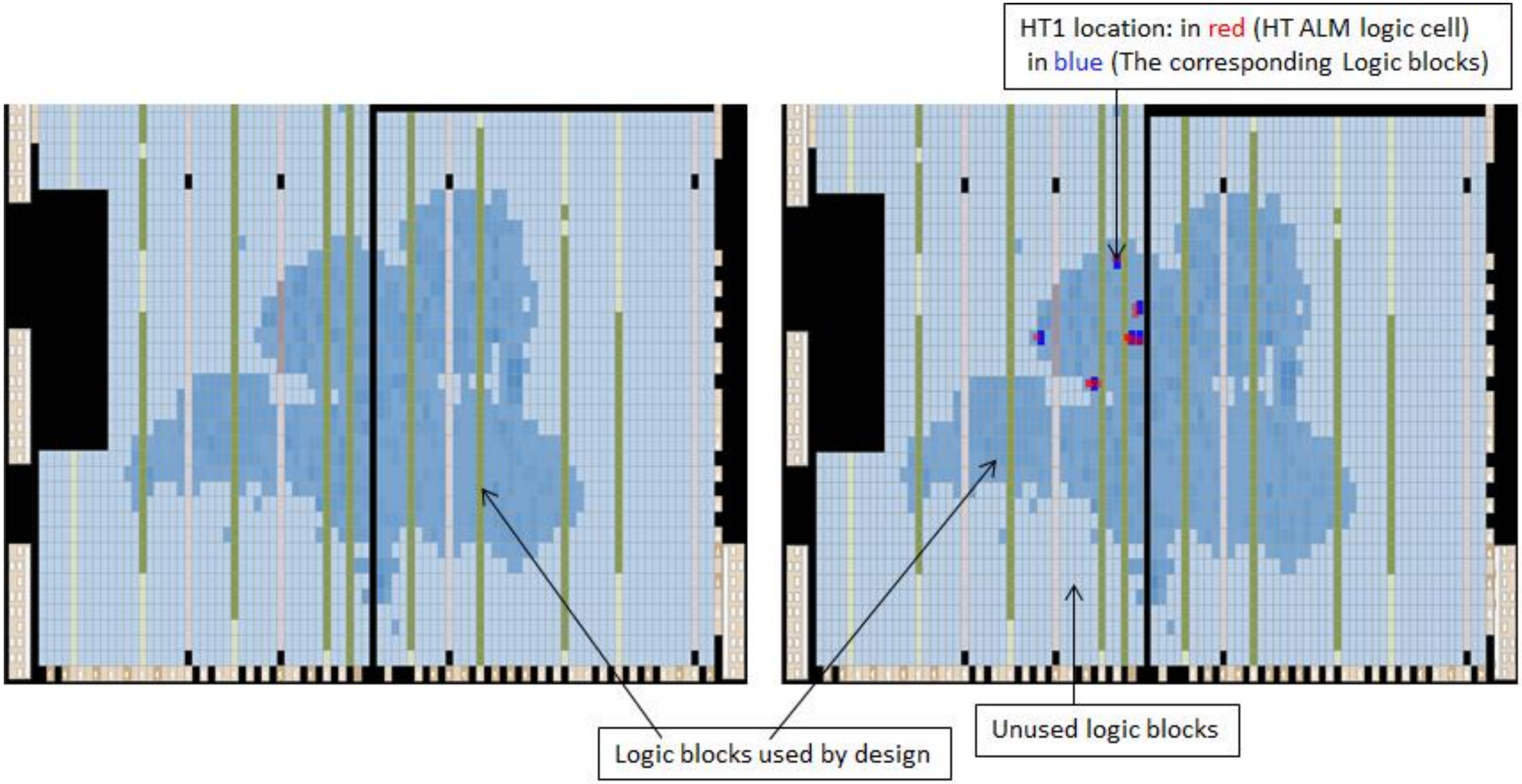
HTH detection by ML

- **State of the art of HTH detection**
  - Statistical tests (T-Test) to compare the equality of population according to the null hypothesis.
- **Test example**
  - 3 HTHs of different sizes in RISC-V CPU running in FPGA:
    - 2 HTHs (HT1 & HT2) are inserted PicoRV32 target
    - 1 HTH (HT3) is inserted in Freedom E300 target

	Target design	Insertion phase	Overhead
HT1	PicoRV32	P&R	0.53%
HT2	PicoRV32	P&R	0.27%
HT3	Freedom	RTL	0.1%

Junko Takahashi, Keiichi Okabe, Hiroki Itoh, Xuan-Thuy Ngo,Sylvain Guilley, Ritu-Ranjan Shrivastwa, Mushir Ahmed, PatrickLejoly, "Machine Learning based Hardware Trojan Detection using Electromagnetic Emanation", ICICS 2020

HT1 insertion



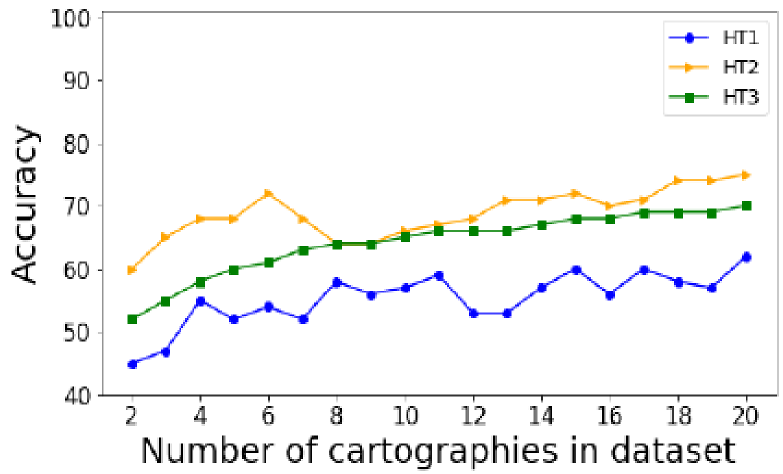
PicoRV32 without HT

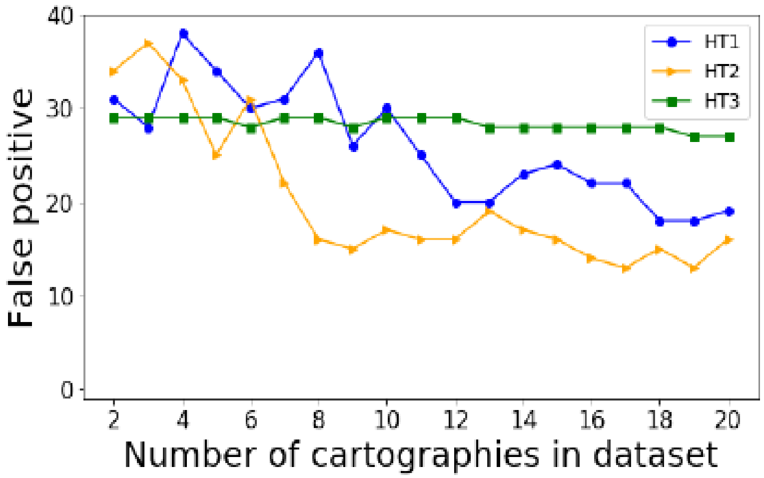
PicoRV32 with HT1

ML Detection Methodology

- **Acquire data for training**
  - 2 FPGAs are used: Reference and HT
  - The dataset comes from N cartographies of the device.
  - Each cartography is a matrix of 13 * 13 points having each EM traces of 5000 samples
- **Train with supervised ML algorithms**
  - SVM, Multi-Layer Perceptron, Decision Tree, KNN
- **Acquire data on target FPGA**
- **Apply the trained models to decide if there is a HTH in the target FPGA**

Results with T-test

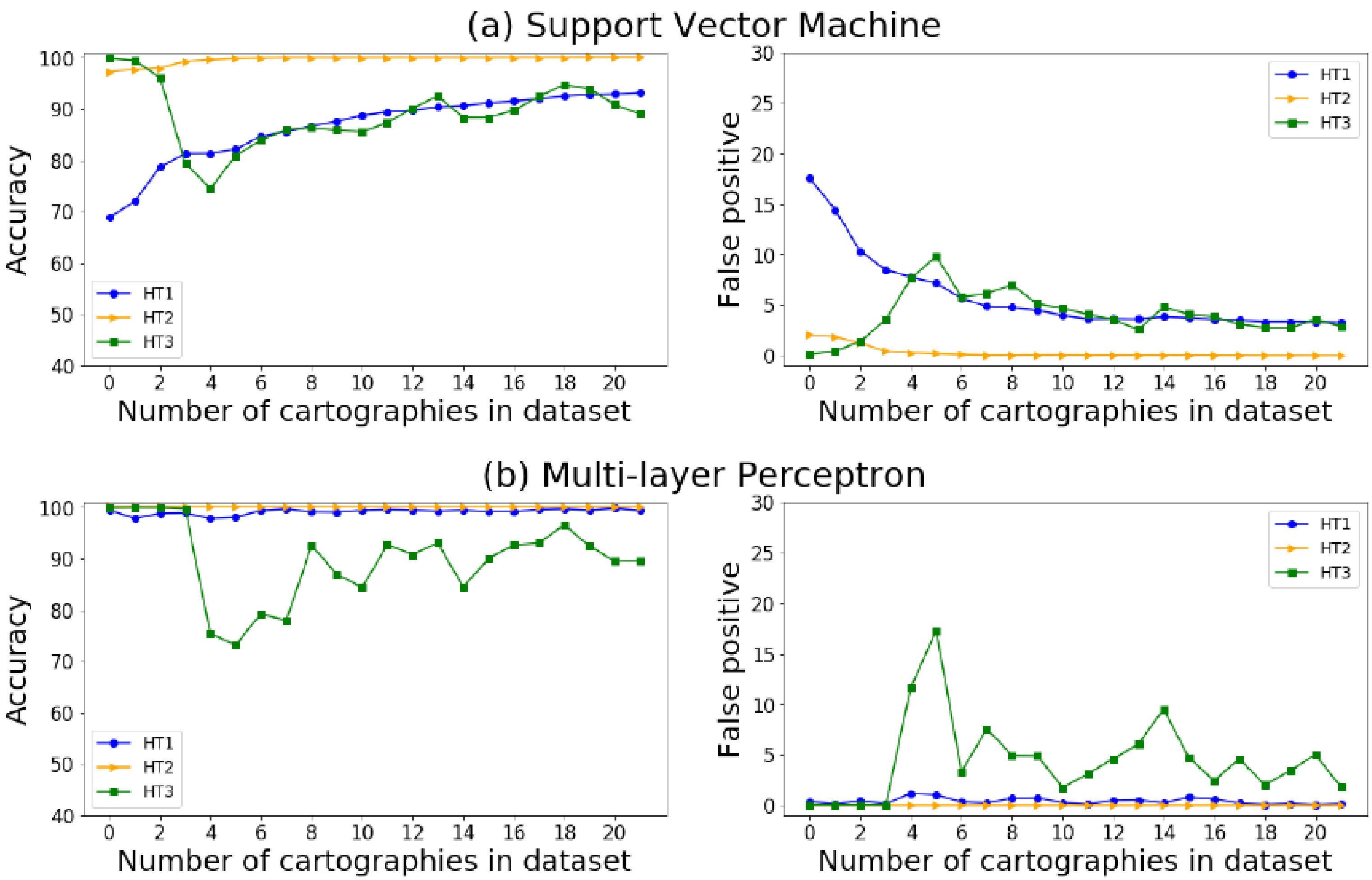




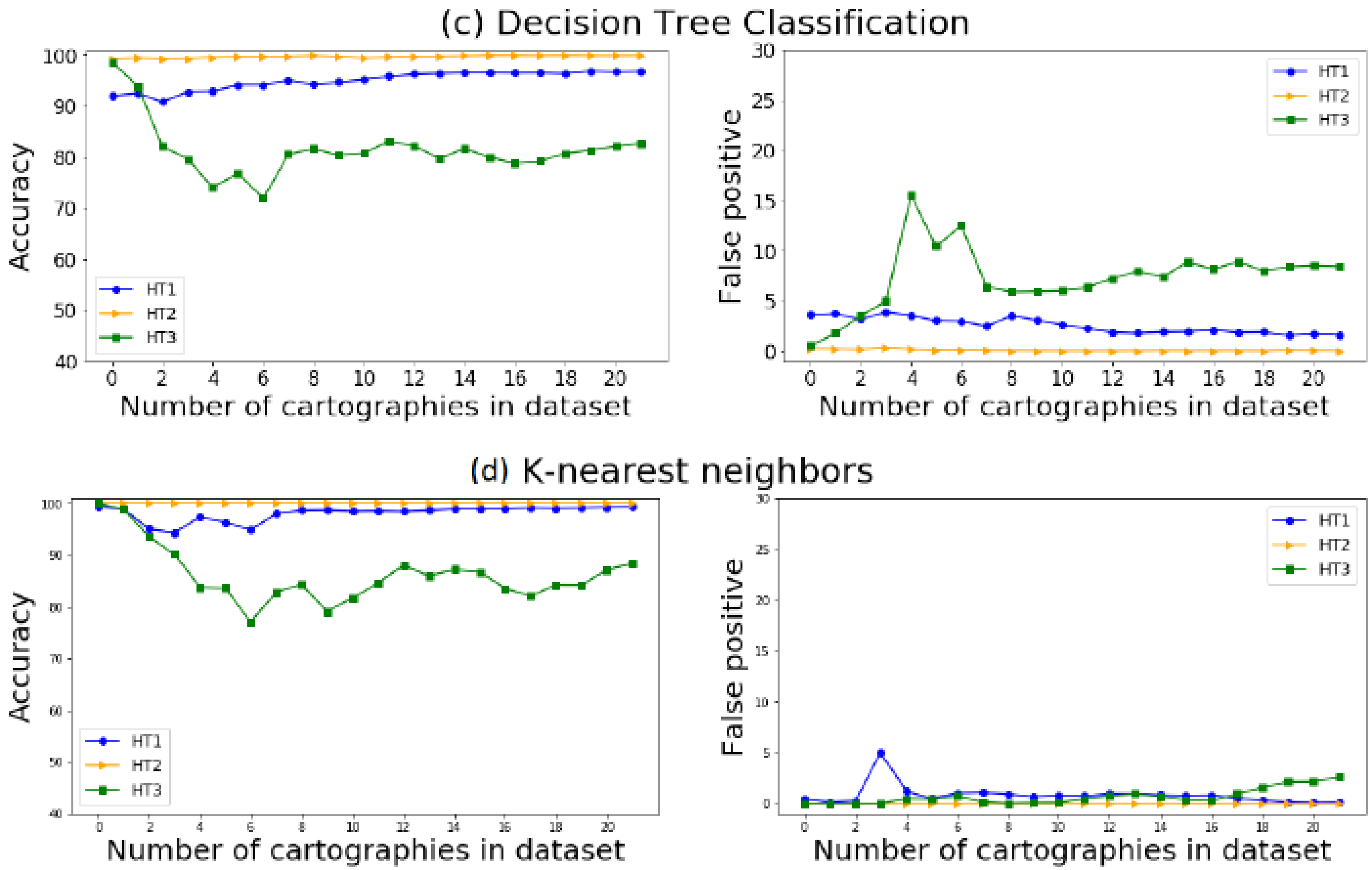
Accuracy < 80%

Many false positives

Results in ML 1/2



Results in ML 2/2



Accuracy >80% even for a tiny HTH

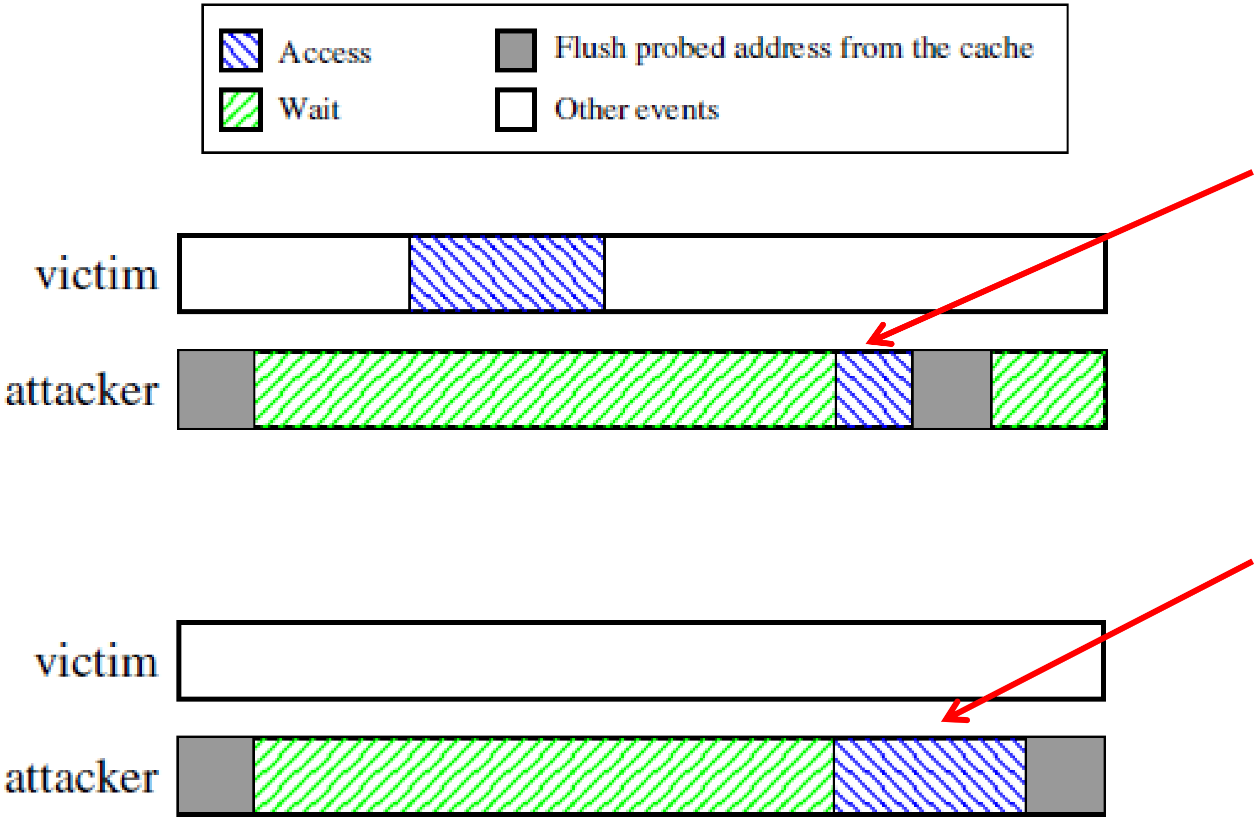
Outline

- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse
- **Security of ML**
  - Example of a CNN implementation

Attack of CNN implementation

- **The CNN security requires:**
  - Protection of the trained model which is often patented
  - Protection of the user privacy, when personal input data are computed with CNN
  - Protection of the output to prevent adversarial attacks
- **But the implementation can be attacked by side-channel: the cache timing attack**

Cache Timing attack example: Flush and Reload

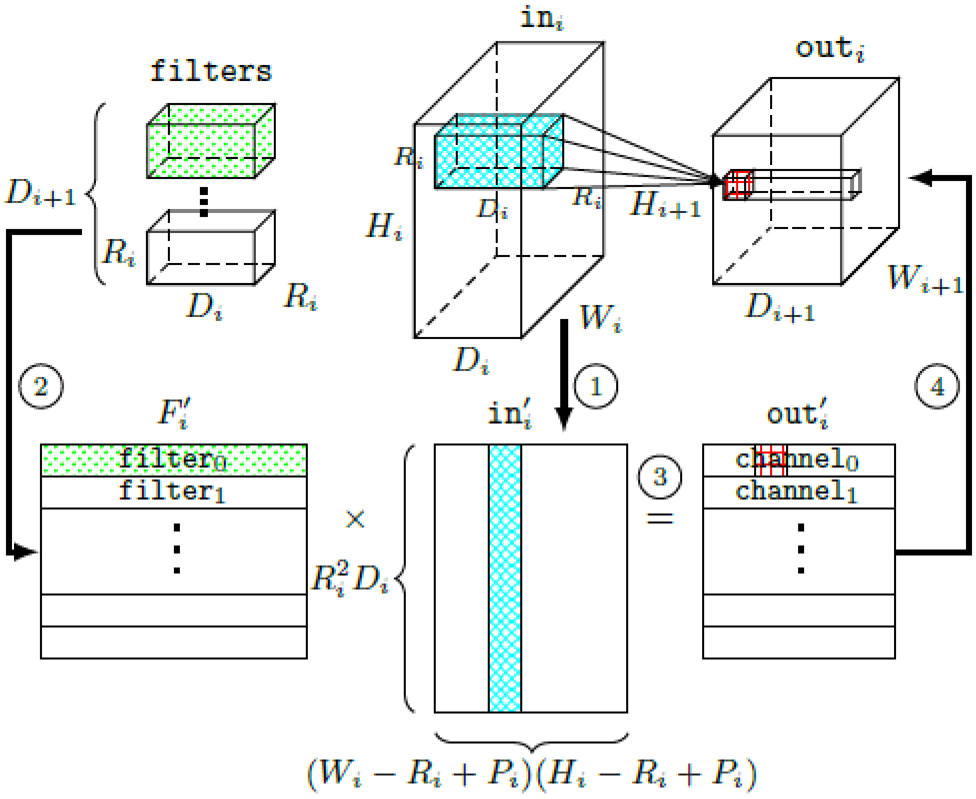


Cache HIT:
The victim used the probed address

Cache MISS:
The victim did not use the probed address

Example: Cache Telepathy Attack

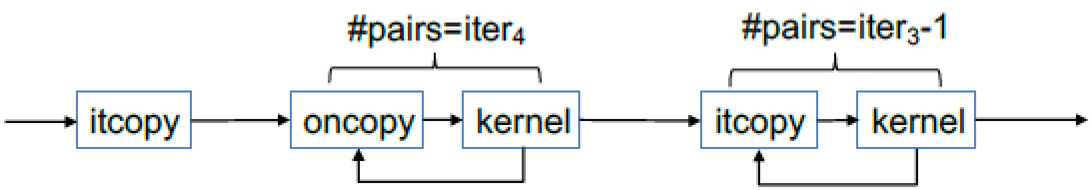
- Computation of convolutionnal layers are transformed into single matrix multiplications by using GEMM:



Yan, M., Fletcher, C.W., Torrellas, J. 'Cache telepathy: Leveraging shared resource attacks to learn DNN architectures'. In: Capkun, S., Roesner, F., editors. 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020. (USENIX Association, 2020. pp. 2003–2020. Available from: https://www:usenix:org/conference /usenixsecurity20/presentation/yan

Side channel leakage when using Gemm

- **3 functions are repeteadly used**
  - Kernel, itcopy , oncopy
  - They form specific patterns according to the iteration type and length.



- **The cache attack allows to count the function calls and determine the number of layers, the input, output output and filter size**
- **Protections**
  - Active research*

* TP: Linda Guiga CIFRE PhD with Idemia

Conclusion

- **ML algorithms provide powerful tools for the security of embedded systems:**
  - Point out design weaknesses , as modeling and cloning unclonable physical functions.
  - Efficient leakage analysis by profiling and combining with side-channels traces.
  - No necessity of preprocessing to reduce noise
  - Detection of abnormal behavior as those coming from stealthy Hardware Trojan Horses
  - Active research for IDS in connected cars*

- **But its implementation can be vulnerable to physical attacks**

* TP: Natasha AlKhatib PhD in the C3S chair

Thank you for your attention