CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Secure Connected Devices

PROFESSOR MÁIRE O’NEILL
Regius Chair in Electronics and Computer Engineering,
Director, CSIT
Director, Research Institute in Secure Hardware & Embedded Systems (RISE)

QUEEN’S UNIVERSITY BELFAST

# Secure Connected Devices

- **Trusted Hardware**
  - PUF-based authentication
  - Hardware Trojan Detection
  - Side Channel Analysis
  - Security & Approximate Computing
  - Deep Learning in HW Security
- **Advanced Crypto Architectures**
  - Post-quantum crypto architectures
  - Hybrid quantum/PQC designs
  - Homomorphic Encryption, IBE, ABE
  - Password Authenticated Key Exchange

# SCD Research Team

**Academic Staff**

Prof Máire O’Neill

Dr Ayesha Khalid (SL)

Dr Ciara Rafferty (SL)

Dr Chongyan Gu (SL)

Dr Arnab Kumar Biswas

Dr Indranil Ghosh Ray

Dr Anh-Tuan Hoang

**Research Staff**

Shichao Yu

Jack Miskelly

Malik Imran

Aditya Japa

Ziying Ni

Gor Piliposyan

**PhD students**

Ryan Bevin

An Troung To

Yuhang Hao

Mark Kennaway

Zain Shabbir

Niall Canavan

Adam Farren

Robert Moore

Ziying Ni

Tuan Dung Pham

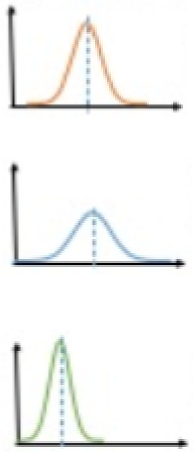
# Machine Learning and
# Physical Unclonable Functions (PUFs)

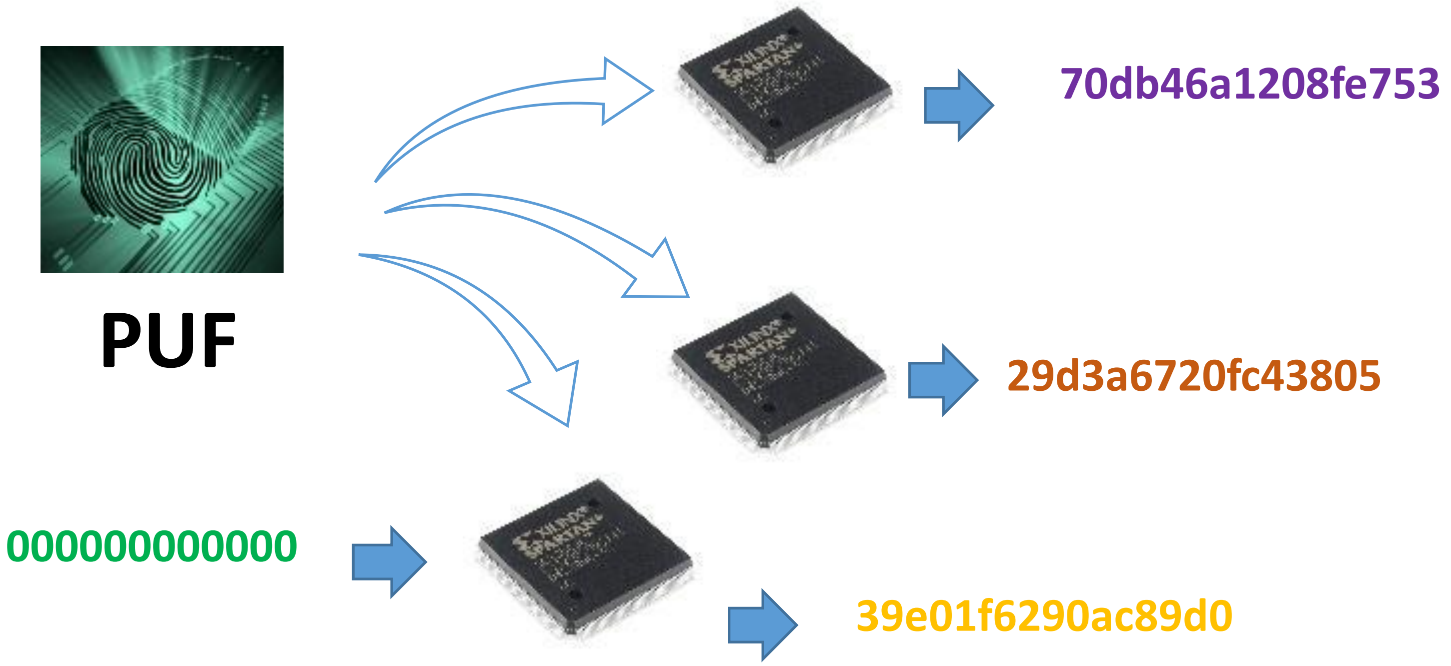
CENTRE
FOR SECURE
INFORMATION
TECHNOLOGIES

# What is a PUF?

A PUF (Physical Unclonable Function) is a digital circuit that uses manufacturing process variations to generate a unique digital fingerprint.

**Process Variations**





*No two chips should give the same response when supplied with the same challenge.*

CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

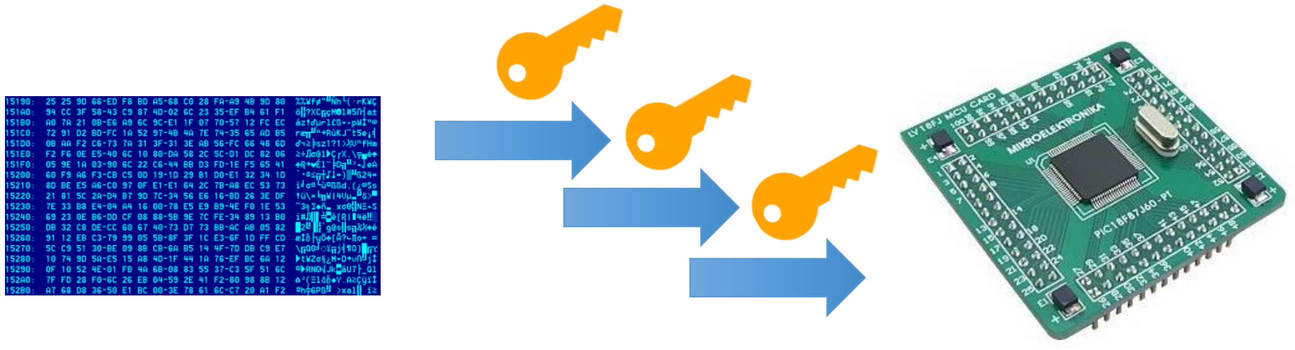
# PUF Applications

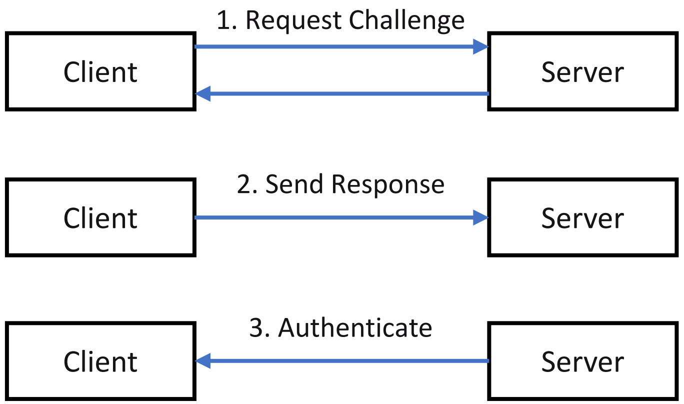
Anti-cloning/Anti-tamper

Key generation/Memoryless key storage

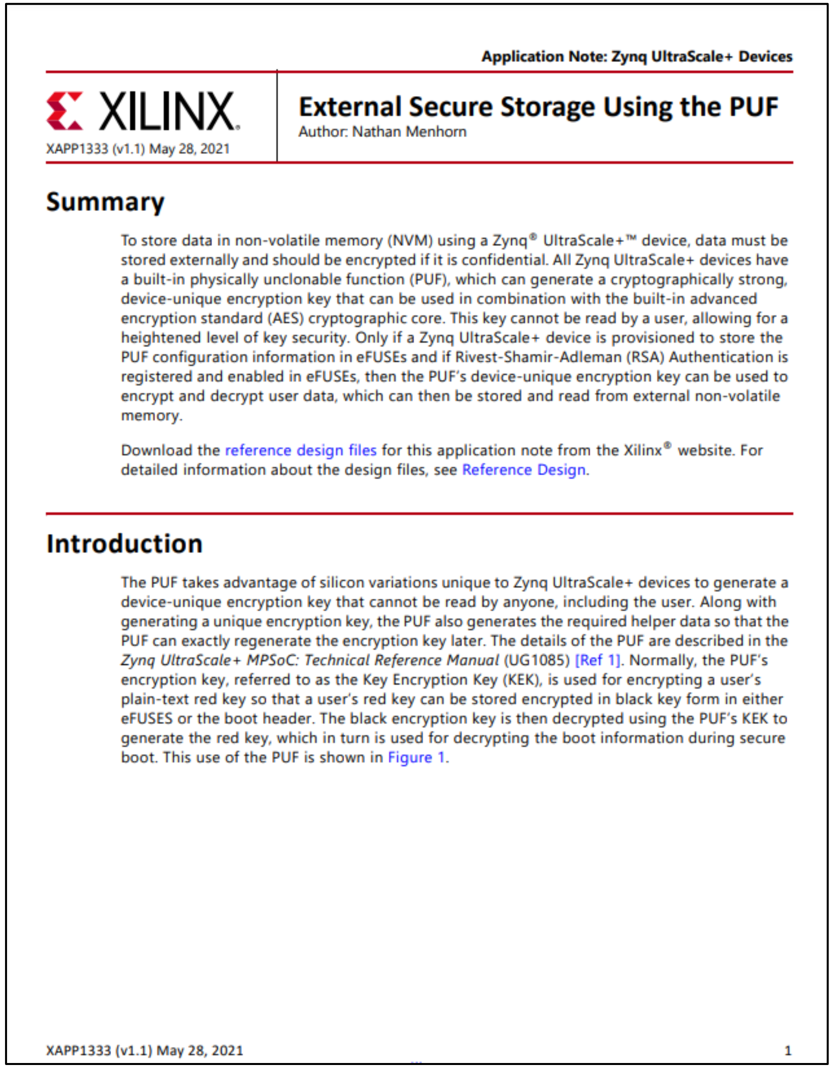
Lightweight device authentication and unique identification

Data Provenance/Incident Tracing





# PUF in Practice



Application Note: Zynq UltraScale+ Devices

XILINX

XAPP1333 (v1.1) May 28, 2021

## External Secure Storage Using the PUF

Author: Nathan Menhorn

## Summary

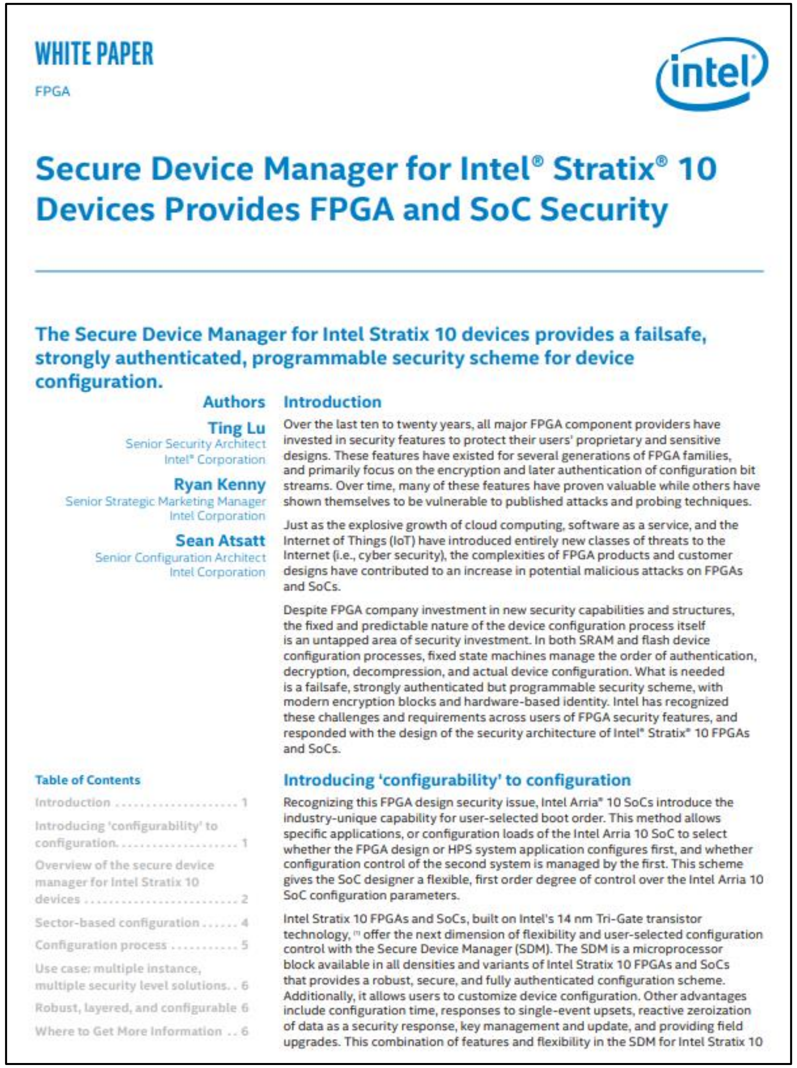
To store data in non-volatile memory (NVM) using a Zynq® UltraScale+™ device, data must be stored externally and should be encrypted if it is confidential. All Zynq UltraScale+ devices have a built-in physically unclonable function (PUF), which can generate a cryptographically strong, device-unique encryption key that can be used in combination with the built-in advanced encryption standard (AES) cryptographic core. This key cannot be read by a user, allowing for a heightened level of key security. Only if a Zynq UltraScale+ device is provisioned to store the PUF configuration information in eFUSEs and if Rivest-Shamir-Adleman (RSA) Authentication is registered and enabled in eFUSEs, then the PUF's device-unique encryption key can be used to encrypt and decrypt user data, which can then be stored and read from external non-volatile memory.

Download the reference design files for this application note from the Xilinx® website. For detailed information about the design files, see Reference Design.

## Introduction

The PUF takes advantage of silicon variations unique to Zynq UltraScale+ devices to generate a device-unique encryption key that cannot be read by anyone, including the user. Along with generating a unique encryption key, the PUF also generates the required helper data so that the PUF can exactly regenerate the encryption key later. The details of the PUF are described in the *Zynq UltraScale+ MPSoC: Technical Reference Manual* (UG1085) [Ref 1]. Normally, the PUF's encryption key, referred to as the Key Encryption Key (KEK), is used for encrypting a user's plain-text red key so that a user's red key can be stored encrypted in black key form in either eFUSES or the boot header. The black encryption key is then decrypted using the PUF's KEK to generate the red key, which in turn is used for decrypting the boot information during secure boot. This use of the PUF is shown in Figure 1.

XAPP1333 (v1.1) May 28, 2021 1



WHITE PAPER

FPGA

intel

## Secure Device Manager for Intel® Stratix® 10 Devices Provides FPGA and SoC Security

**The Secure Device Manager for Intel Stratix 10 devices provides a failsafe, strongly authenticated, programmable security scheme for device configuration.**

Authors

Ting Lu
Senior Security Architect
Intel® Corporation

Ryan Kenny
Senior Strategic Marketing Manager
Intel Corporation

Sean Atsatt
Senior Configuration Architect
Intel Corporation

### Introduction

Over the last ten to twenty years, all major FPGA component providers have invested in security features to protect their users' proprietary and sensitive designs. These features have existed for several generations of FPGA families, and primarily focus on the encryption and later authentication of configuration bit streams. Over time, many of these features have proven valuable while others have shown themselves to be vulnerable to published attacks and probing techniques.

Just as the explosive growth of cloud computing, software as a service, and the Internet of Things (IoT) have introduced entirely new classes of threats to the Internet (i.e., cyber security), the complexities of FPGA products and customer designs have contributed to an increase in potential malicious attacks on FPGAs and SoCs.

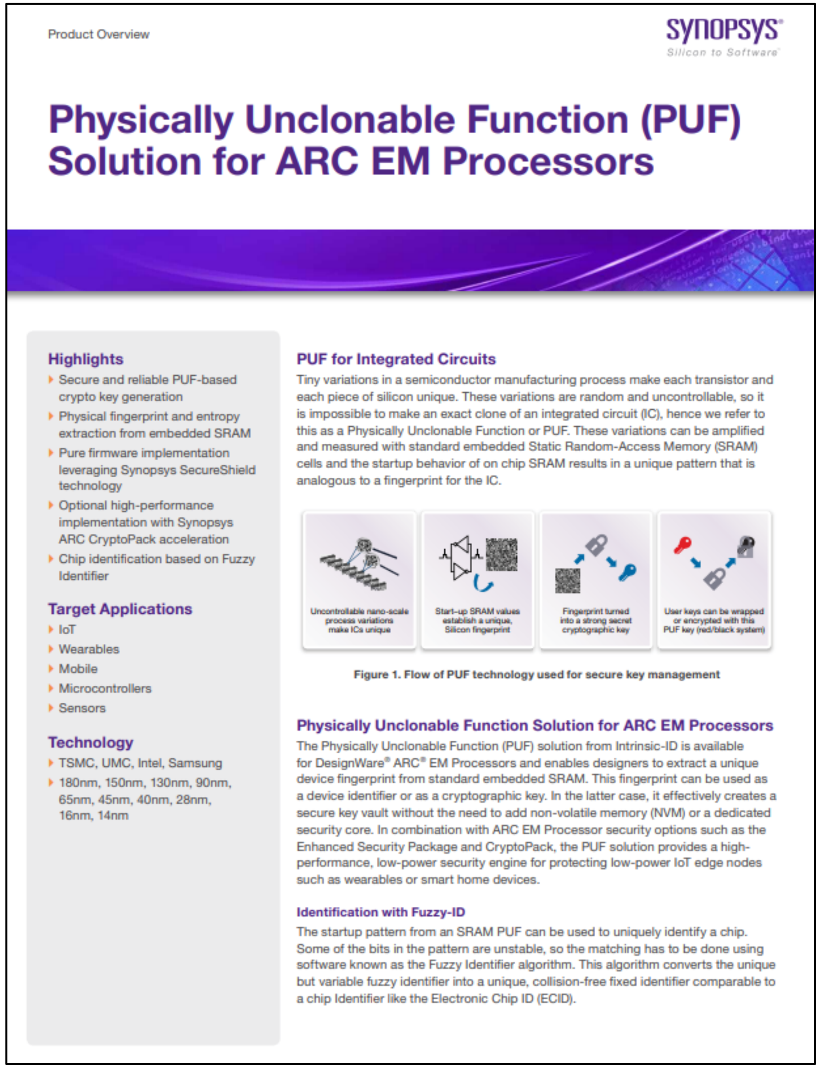
Despite FPGA company investment in new security capabilities and structures, the fixed and predictable nature of the device configuration process itself is an untapped area of security investment. In both SRAM and flash device configuration processes, fixed state machines manage the order of authentication, decryption, decompression, and actual device configuration. What is needed is a failsafe, strongly authenticated but programmable security scheme, with modern encryption blocks and hardware-based identity. Intel has recognized these challenges and requirements across users of FPGA security features, and responded with the design of the security architecture of Intel® Stratix® 10 FPGAs and SoCs.

Table of Contents

### Introducing 'configurability' to configuration

Recognizing this FPGA design security issue, Intel Arria® 10 SoCs introduce the industry-unique capability for user-selected boot order. This method allows specific applications, or configuration loads of the Intel Arria 10 SoC to select whether the FPGA design or HPS system application configures first, and whether configuration control of the second system is managed by the first. This scheme gives the SoC designer a flexible, first order degree of control over the Intel Arria 10 SoC configuration parameters.

Intel Stratix 10 FPGAs and SoCs, built on Intel's 14 nm Tri-Gate transistor technology, [m] offer the next dimension of flexibility and user-selected configuration control with the Secure Device Manager (SDM). The SDM is a microprocessor block available in all densities and variants of Intel Stratix 10 FPGAs and SoCs that provides a robust, secure, and fully authenticated configuration scheme. Additionally, it allows users to customize device configuration. Other advantages include configuration time, responses to single-event upsets, reactive zeroization of data as a security response, key management and update, and providing field upgrades. This combination of features and flexibility in the SDM for Intel Stratix 10



Product Overview

SYNOPSYS
Silicon to Software

## Physically Unclonable Function (PUF) Solution for ARC EM Processors

### Highlights

- Secure and reliable PUF-based crypto key generation
- Physical fingerprint and entropy extraction from embedded SRAM
- Pure firmware implementation leveraging Synopsys SecureShield technology
- Optional high-performance implementation with Synopsys ARC CryptoPack acceleration
- Chip identification based on Fuzzy Identifier

### Target Applications

- IoT
- Wearables
- Mobile
- Microcontrollers
- Sensors

### Technology

- TSMC, UMC, Intel, Samsung
- 180nm, 150nm, 130nm, 90nm, 65nm, 45nm, 40nm, 28nm, 16nm, 14nm

### PUF for Integrated Circuits

Tiny variations in a semiconductor manufacturing process make each transistor and each piece of silicon unique. These variations are random and uncontrollable, so it is impossible to make an exact clone of an integrated circuit (IC), hence we refer to this as a Physically Unclonable Function or PUF. These variations can be amplified and measured with standard embedded Static Random-Access Memory (SRAM) cells and the startup behavior of on chip SRAM results in a unique pattern that is analogous to a fingerprint for the IC.

Uncontrollable nano-scale process variations make ICs unique
Start-up SRAM values establish a unique, Silicon fingerprint
Fingerprint turned into a strong secret cryptographic key
User keys can be wrapped or encrypted with this PUF key (red/black system)

Figure 1. Flow of PUF technology used for secure key management

### Physically Unclonable Function Solution for ARC EM Processors

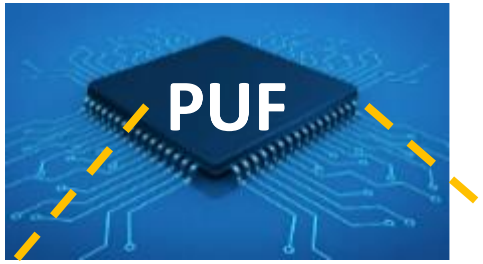
The Physically Unclonable Function (PUF) solution from Intrinsic-ID is available for DesignWare® ARC® EM Processors and enables designers to extract a unique device fingerprint from standard embedded SRAM. This fingerprint can be used as a device identifier or as a cryptographic key. In the latter case, it effectively creates a secure key vault without the need to add non-volatile memory (NVM) or a dedicated security core. In combination with ARC EM Processor security options such as the Enhanced Security Package and CryptoPack, the PUF solution provides a high-performance, low-power security engine for protecting low-power IoT edge nodes such as wearables or smart home devices.

#### Identification with Fuzzy-ID

The startup pattern from an SRAM PUF can be used to uniquely identify a chip. Some of the bits in the pattern are unstable, so the matching has to be done using software known as the Fuzzy Identifier algorithm. This algorithm converts the unique but variable fuzzy identifier into a unique, collision-free fixed identifier comparable to a chip Identifier like the Electronic Chip ID (ECID).

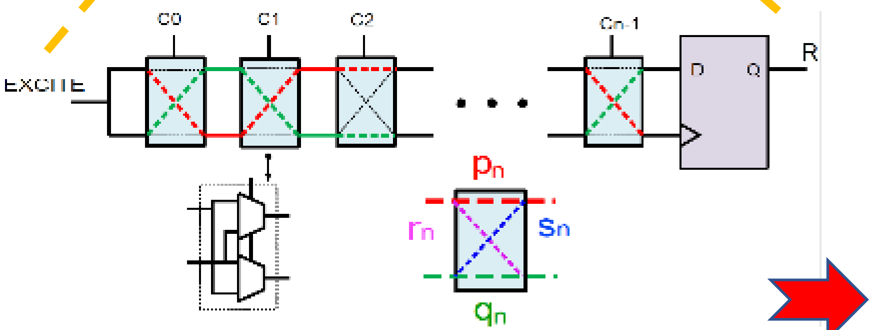
# Challenge-Response PUFS are vulnerable to ML attacks

## Device



Challenge

Response



Arbiter PUF

## PUF CRPs

| Challenge | Response |
|---|---|
| 10011…110 | 10110…101 |
| 00110…101 | 00101…010 |
| 10110…001 | 10101…000 |
| … | |
| 01101…101 | 10001…111 |

### Linear Function

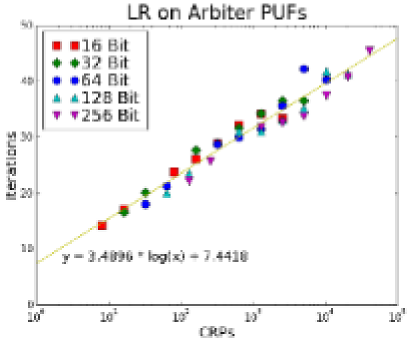
$$\boxed{\Delta(n) = P \cdot \boldsymbol{\omega}^{\boldsymbol{T}}}$$

$$\omega_i = \alpha_i + \beta_{i-1}, 2 \le i \le n$$

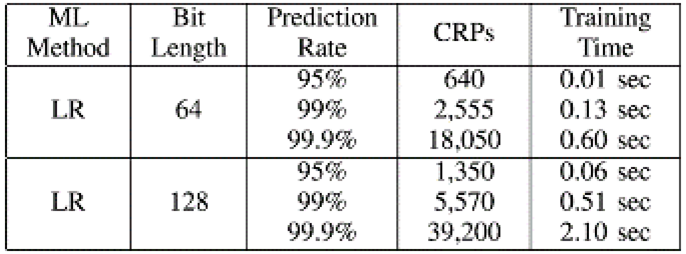
$$\alpha_i = \frac{p_i - q_i - r_i + s_i}{2}$$

$$\beta_i = \frac{p_i - q_i + r_i - s_i}{2}$$

## Machine Learning Algorithm



Logistic Regression (LR) on Arbiter PUFs



| ML Method | Bit Length | Prediction Rate | CRPs | Training Time |
|---|---|---|---|---|
| LR | 64 | 95%<br>99%<br>99.9% | 640<br>2,555<br>18,050 | 0.01 sec<br>0.13 sec<br>0.60 sec |
| LR | 128 | 95%<br>99%<br>99.9% | 1,350<br>5,570<br>39,200 | 0.06 sec<br>0.51 sec<br>2.10 sec |

LR Prediction Results

U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas, “PUF modelling attacks on simulated and silicon data,” IEEE Trans Information Forensics and Security, vol. 8(11), pp.1876–1891, 2013.

# ML- Attack Resistant PUF Design Approaches

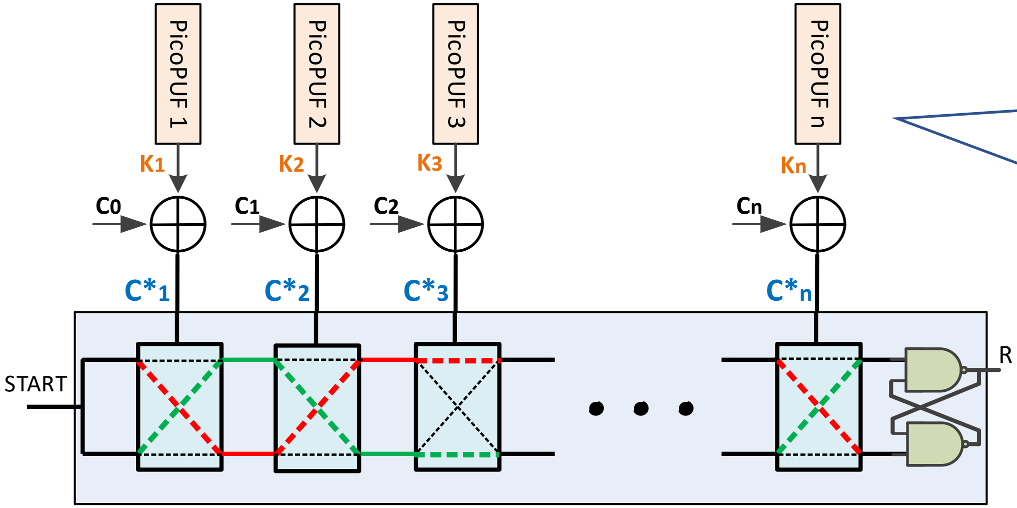
- Increase complexity of the PUF design
  - if too complex, PUF design is no longer a lightweight primitive

- Obfuscate the challenge/response (e.g. XOR Arbiter PUF)
  - Use a weak PUF
  - All XOR APUF shown to be susceptible to reliability based CMA-ES attacks (based on challenge-reliability pairs)

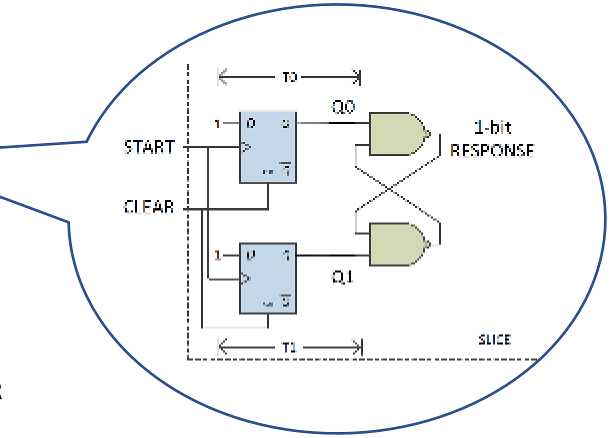
- Deception techniques

# ML- Attack Resistant PUF - Challenge obfuscation

Arbiter-based multi-PUF (MPUF) design - utilises a Weak PUF to obfuscate the challenges to a Strong PUF
=> harder to model than the conventional Arbiter PUF using ML attacks.



*Proposed 1-bit MPUF Design*



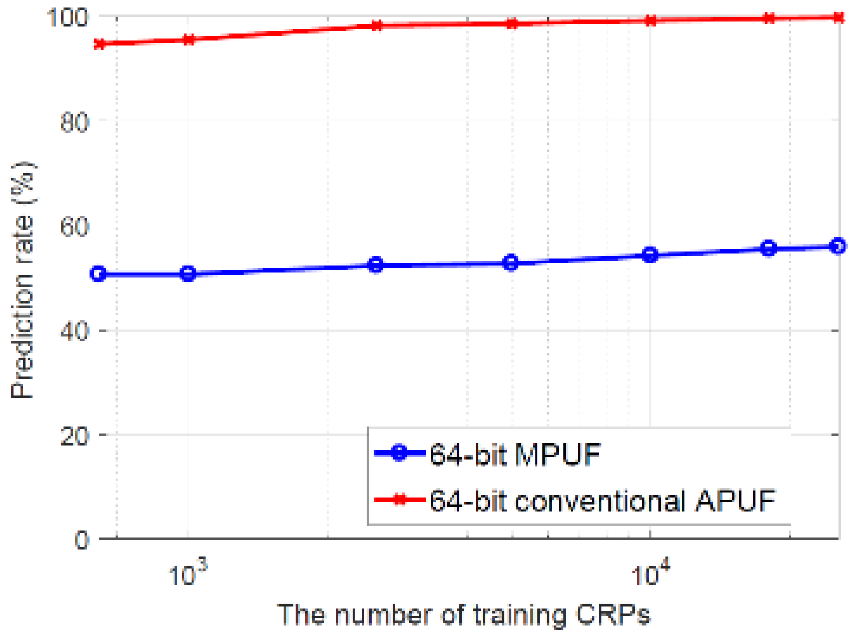
**1-bit PicoPUF Circuit Design**

The responses of the PicoPUFs are used to mask the original challenges $C_i$

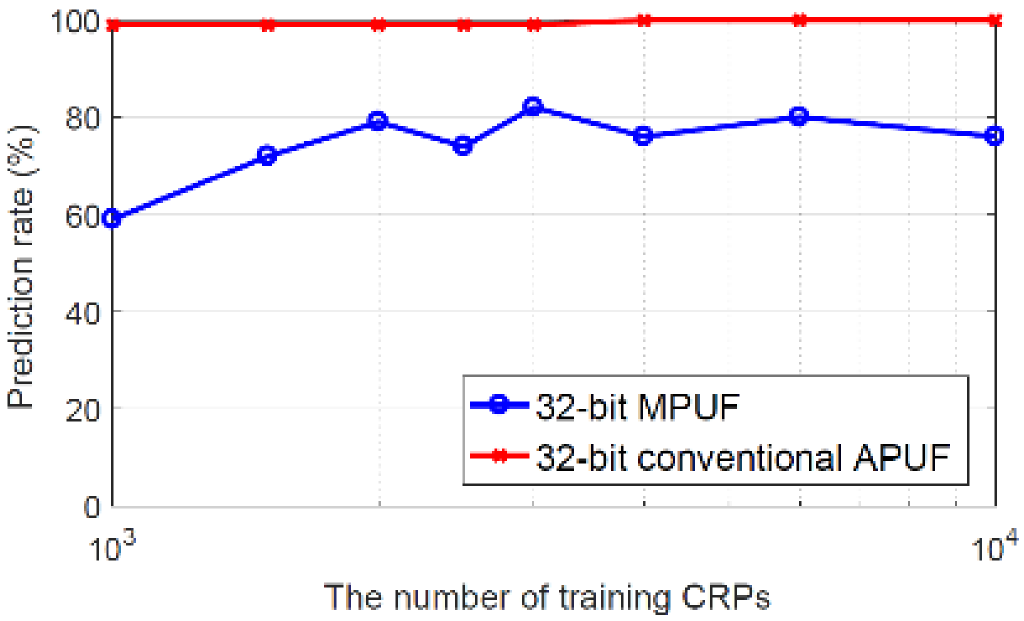
Q. Ma, C. Gu, N. Hanley, C. Wang, W. Liu and M. O'Neill, "A machine learning attack resistant multi-PUF design on FPGA," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jeju, 2018, pp. 97-104.

# ML- Attack Resistant PUF - Challenge obfuscation

Most common ML-based attacks applied to PUF:
- Logistic regression (LR)
- Covariance matrix adaptation evolution strategies (CMA-ES)



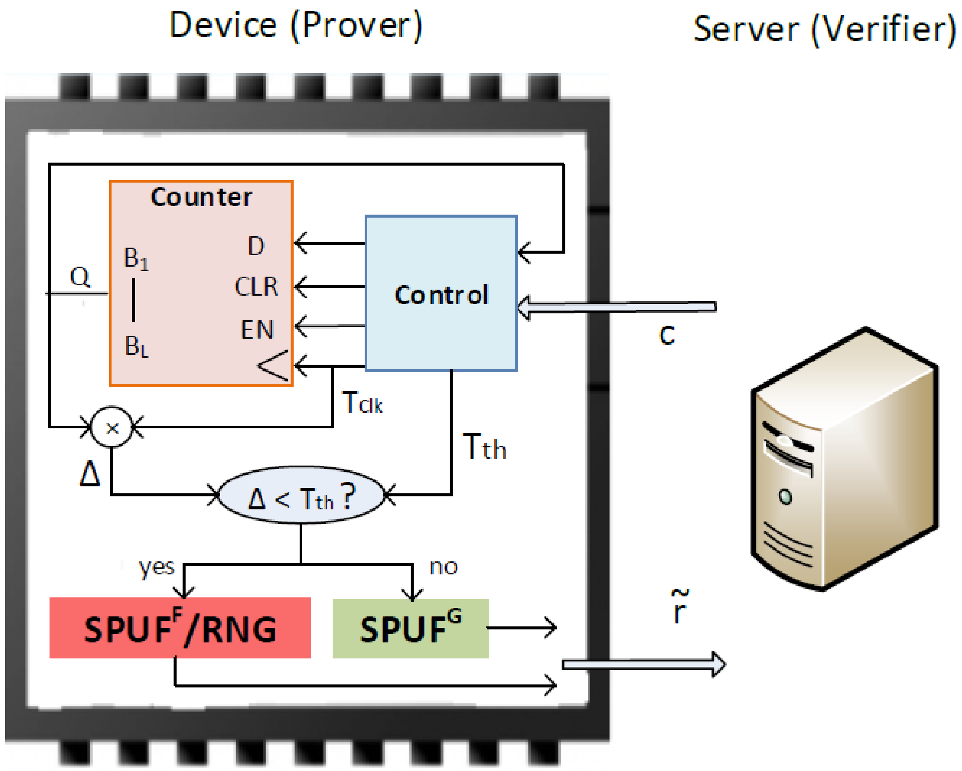
***Prediction rates for conventional Arbiter- PUF and proposed MPUF designs using LR***



***Prediction rates for conventional Arbiter- PUF and proposed MPUF designs using CMA-ES***

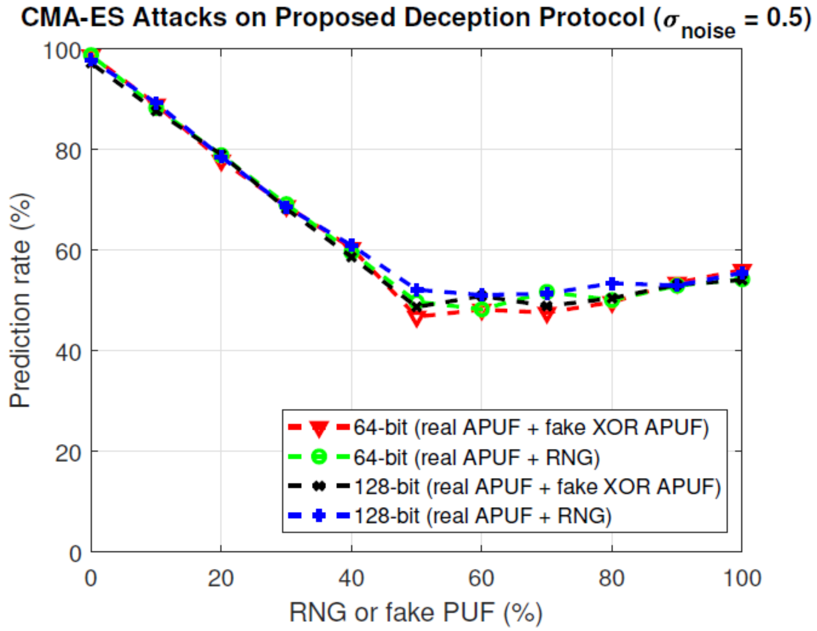
# ML- Attack Resistant PUF – Deception Protocols

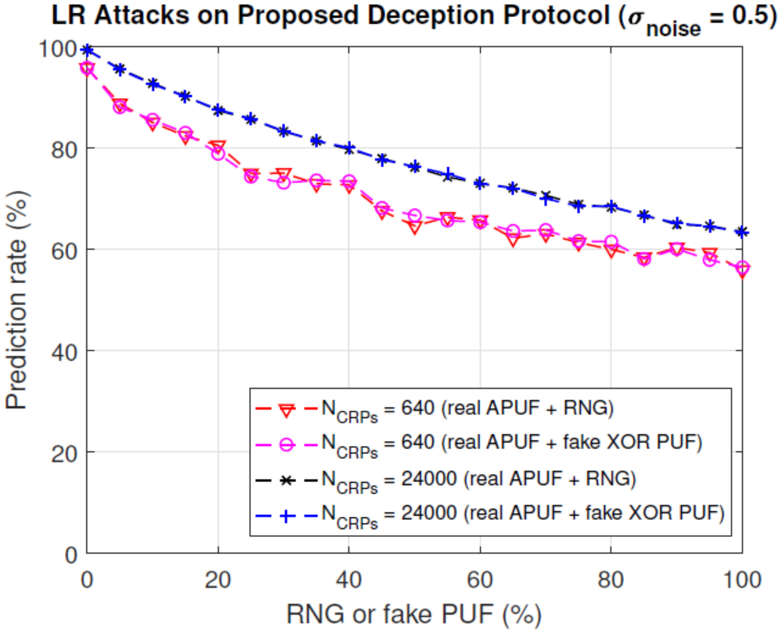
- Device detects an adversary sending continuous authentication requests
- Generate some responses from a deceptive PUF design and others generated from real PUF.
- Adversary will be deceived into deriving a fake PUF model from the collected data.
- Lightweight and do not require error-correction or sophisticated cryptographic algorithms



C Gu, C.H. Chang, W. Liu, S. Yu, Q. Ma, M. O'Neill, A Modeling Attack Resistant Deception Technique for Securing PUF based Authentication , 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)

# ML- Attack Resistant PUF – Deception Protocol





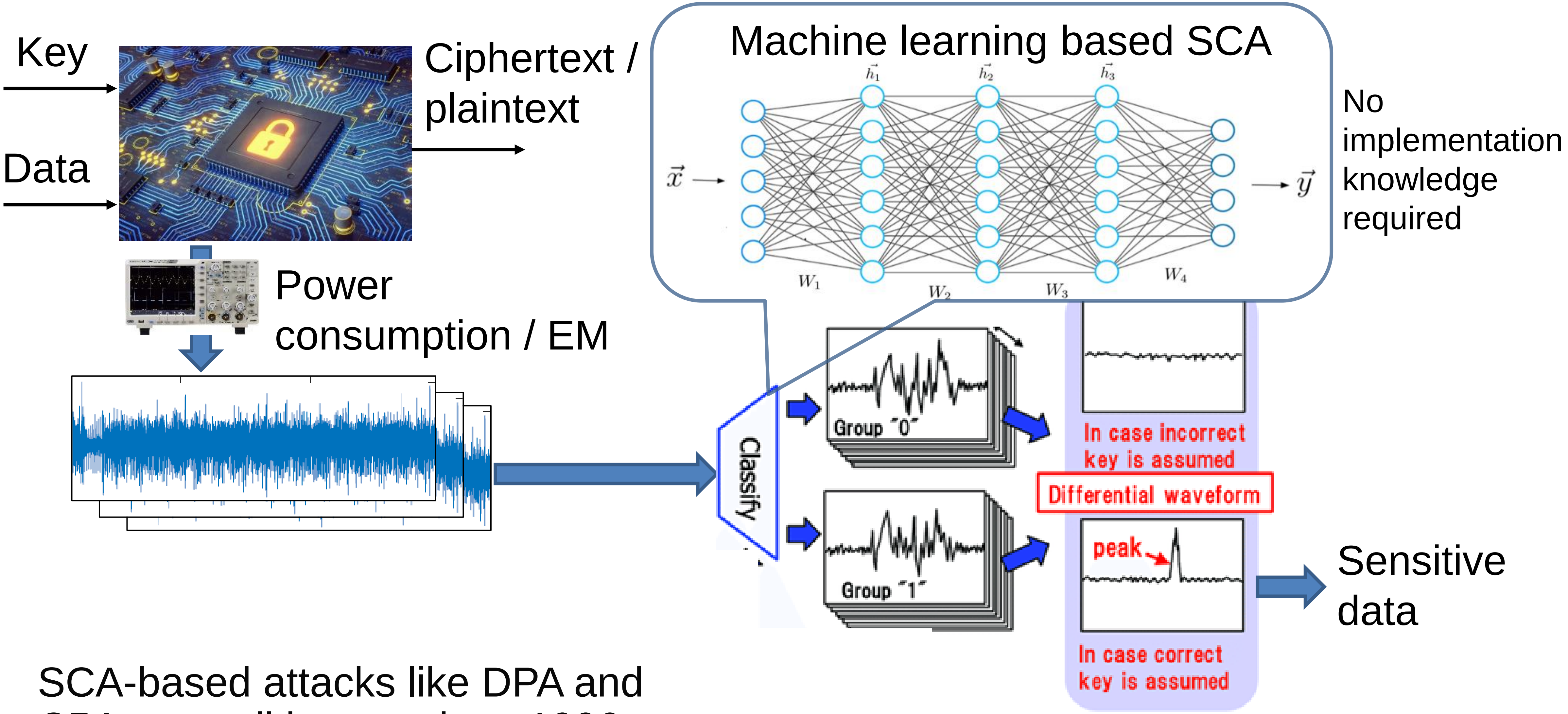
The CMA-ES attack results for the proposed deception protocol by applying different challenge bit lengths, 64-bit and 128-bit, as well as utilizing different strategies (RNG and fake PUF).

The LR attack results for the proposed deception protocol utilizing different strategies, RNG and fake PUF. The y-axis shows the achieved prediction rate of the LR attacks based on different percentages of fake information mixed with the training responses.

# Machine Learning and Side Channel Analysis

CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Side Channel Analysis (SCA)



Key

Data

Ciphertext / plaintext

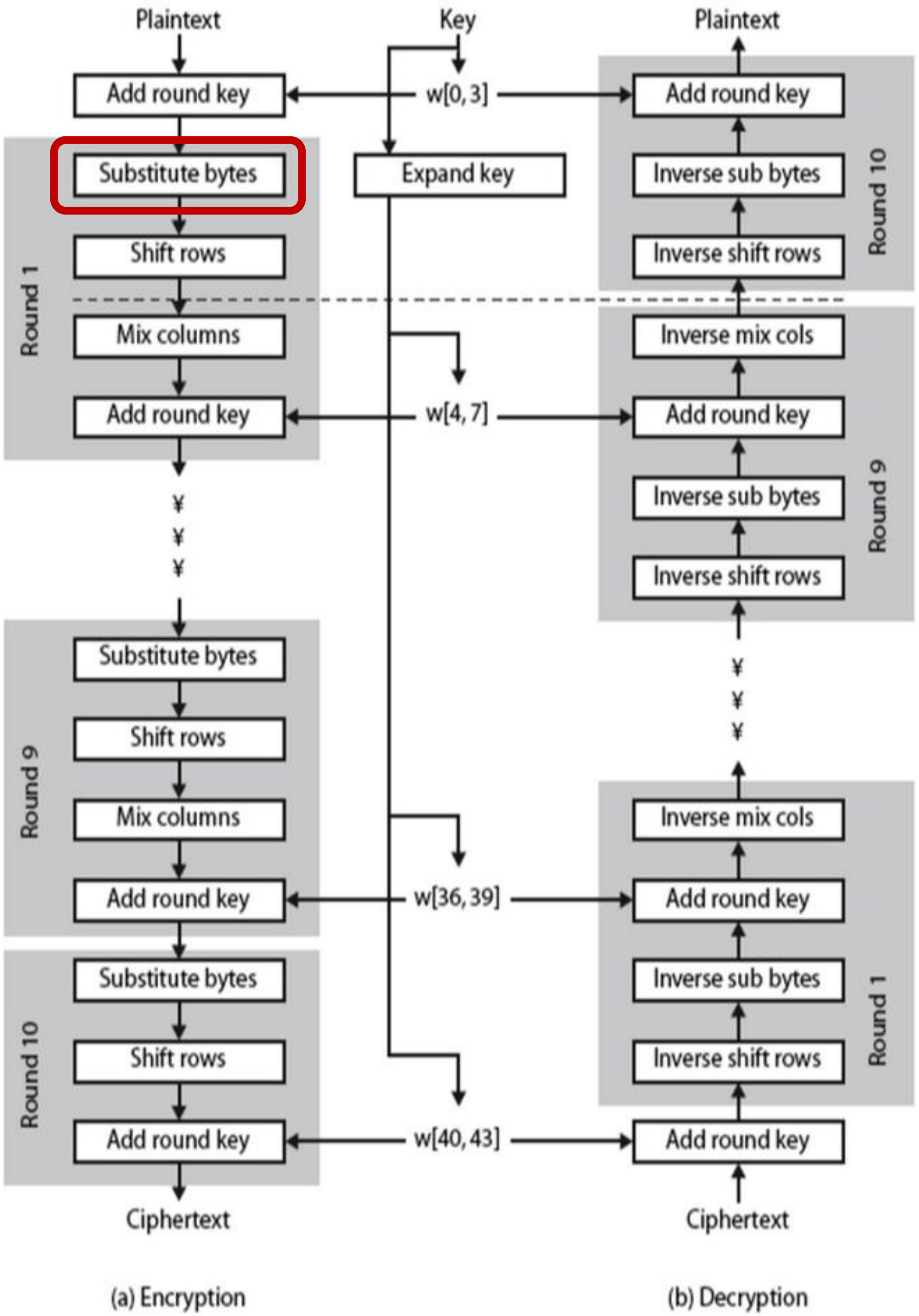
Power consumption / EM

Machine learning based SCA

No implementation knowledge required

Sensitive data

SCA-based attacks like DPA and CPA are well known since 1996



# Leakage of AES implementation

- Advanced Encryption Standard (AES) is safe in theory, but it is vulnerable under SCA.
- Non-linear and sensitive operation Sbox works with 8-bit sub-byte key.
- Sbox leaks hypothesis key via the relationship between the output value or its hamming weight, and side-channel information.
- The leakage can be trained using machine learning

# SCA Countermeasures against DPA/CPA

**Masking**

- a random mask is generated to conceal intermediate values, removing the correlation between the measurements and the secret data

**Hiding**

- aim is to make measurements look random or constant
- decreases the SNR only
- Timing (insert dummy operations, shuffling …)
- Amplitude (filters, pipelining …)

# Evaluated AES implementation with SCA countermeasure (from ASCAD Database)

- AES implementation on 8-bit AVR ATMega 8515 microprocessor
- Two **masks** are used for
  - Plaintext $\overline{p_i} = p_i \oplus m_i$
  - SBox $\overline{SBox(x)} = SBox(x \oplus m_{i.in}) \oplus m_{i.out}$

A-T. Hoang, N. Hanley, M.O'Neill, **Plaintext: A Missing Feature for Enhancing the Power of Deep Learning in Side-Channel Analysis?** IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), 2020(4), 49-85

A-T. Hoang, N. Hanley, A. Khalid, D. Kundi, M.O'Neill, **Stacked Ensemble Model for Enhancing the DL based SCA.** 19th International Conference on Security and Cryptography, SECRYPT 2022, Lisbon, Portugal, July 11-13, 2022, pages 59–68, 2022

# Attack model

- Attack on the output of the 3rd SBox in the 1st round of AES

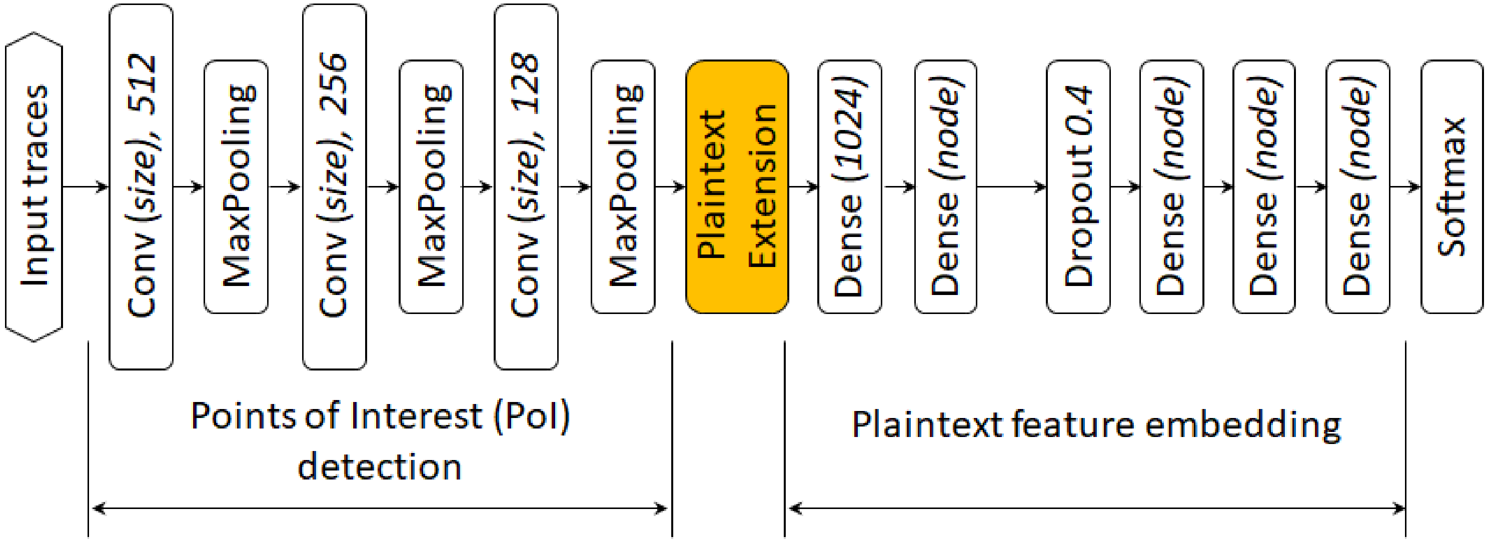
- Classification uses the output value of SBox (256 classes)

$$SBox(p_2 \oplus k_2)$$

# CNN with Plaintext extension (CNNP)

- Three convolutional layers
- The number of convolutional filters reduces from 512 to 128
- Maxpooling is used for feature finding
- Finding features are extended with Plaintext

CNNP with single convolutional filter kernel (size) version 1
(Simplified version of multiple PoI sizes combination)



- Five fully-connected layers are used to compile the features extracted from the previous layers
- Over-fitting is prevented by using dropout

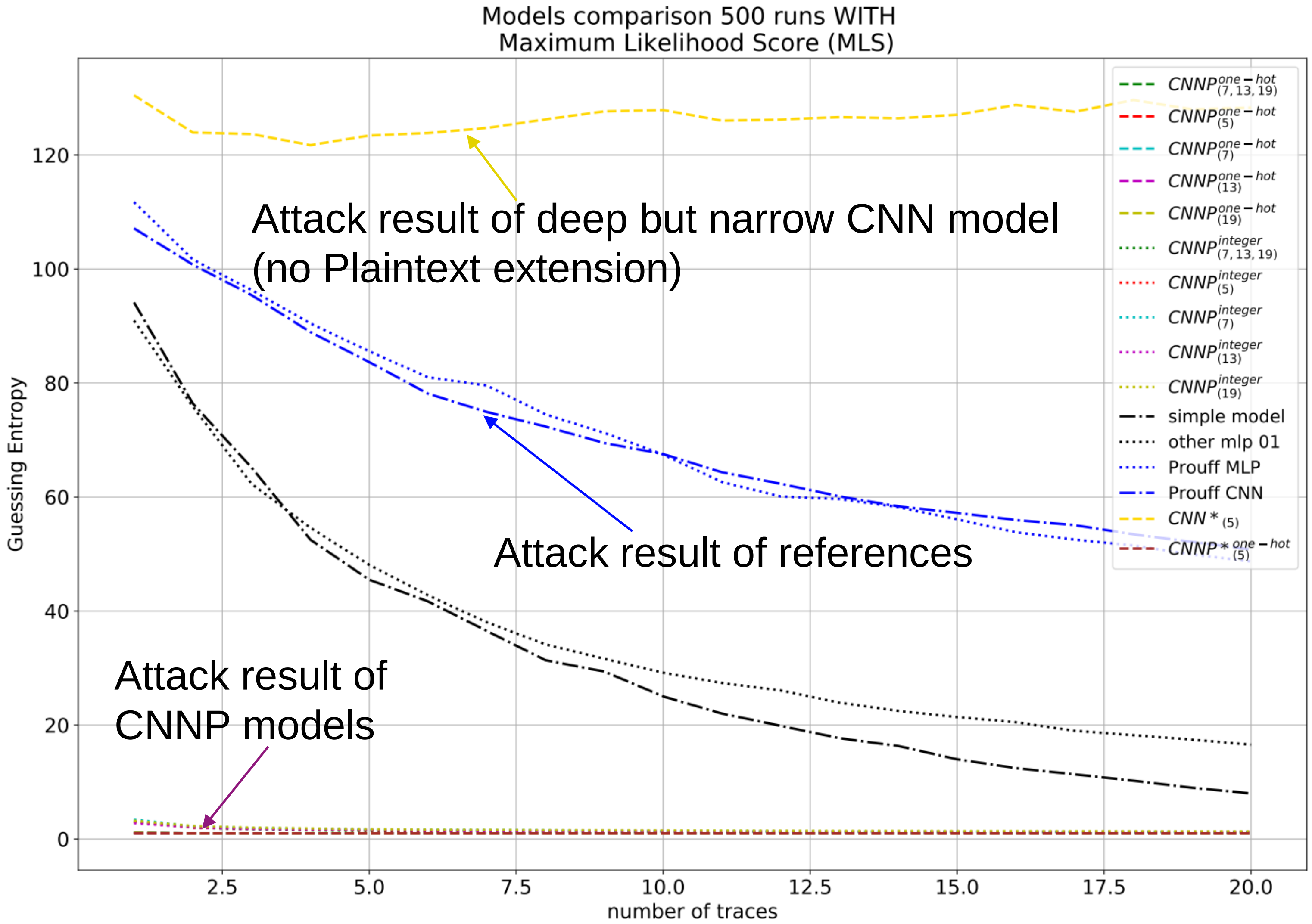
# Attackers knowledge & experimental conditions

- Assumption about attacker:
  - Knows plaintext / ciphertext
  - Aware of SCA countermeasure but not aware of the detailed design and random mask value
  - Can profile keys on the implementation

- Hypothesis keys are ranked using Maximum likelihood score

- Training is performed on VMware hosted Ubuntu with access to virtual NVIDIA GRID M60-8Q and M40-4Q GPUs.

# Evaluation of CNNP models on ASCAD fixed key dataset



Attack result of deep but narrow CNN model (no Plaintext extension)

Attack result of references

Attack result of CNNP models

- CNNP model can reveal the secret key within 2 traces
- CNNP models relies on the bijection S[(.) ⊕ *K*] to reveal *K* without using traces

# ML-based Countermeasures against ML-based SCA

- Countermeasures based on adversarial attacks
  - add adversarial perturbations to the crypto implementation

- Reinforcement learning approach to construct low-cost hiding countermeasure combinations
  - finds the best combination of countermeasures within a specific budget

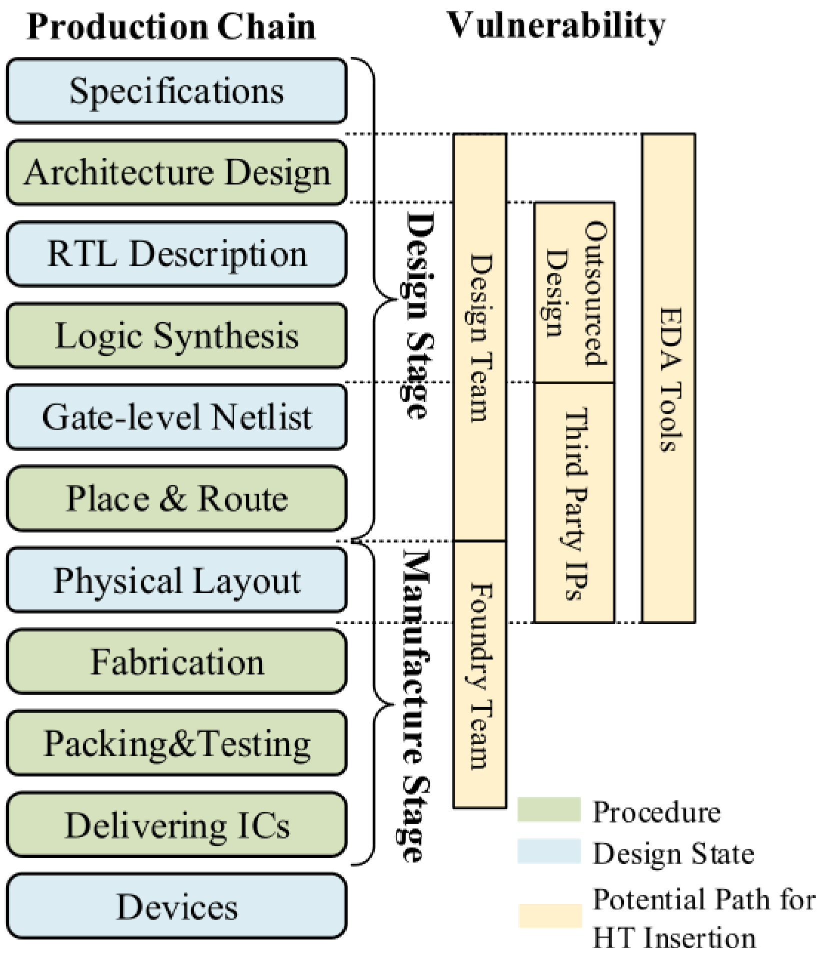
S.Picek, D.Jap, S.Bhasin. Poster: When adversary becomes the guardian–towards side-channel security with adversarial attacks. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2673–2675, 2019

J. Rijsdijk, L Wu, G Perin, Reinforcement Learning-Based Design of Side-Channel Countermeasures, International Conference on Security, Privacy, and Applied Cryptography Engineering, LNCS 13162, pp 168-187, 2021

# Machine Learning and Hardware Trojan Detection

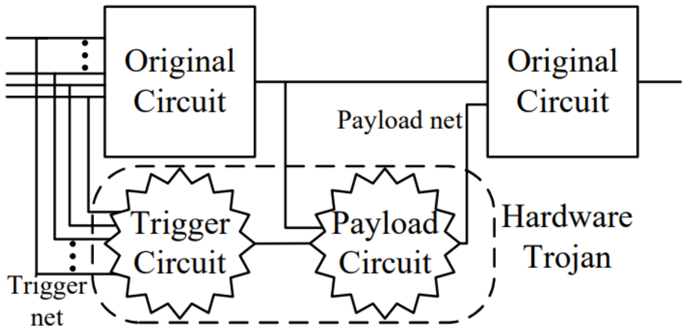
CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Hardware Trojan Detection

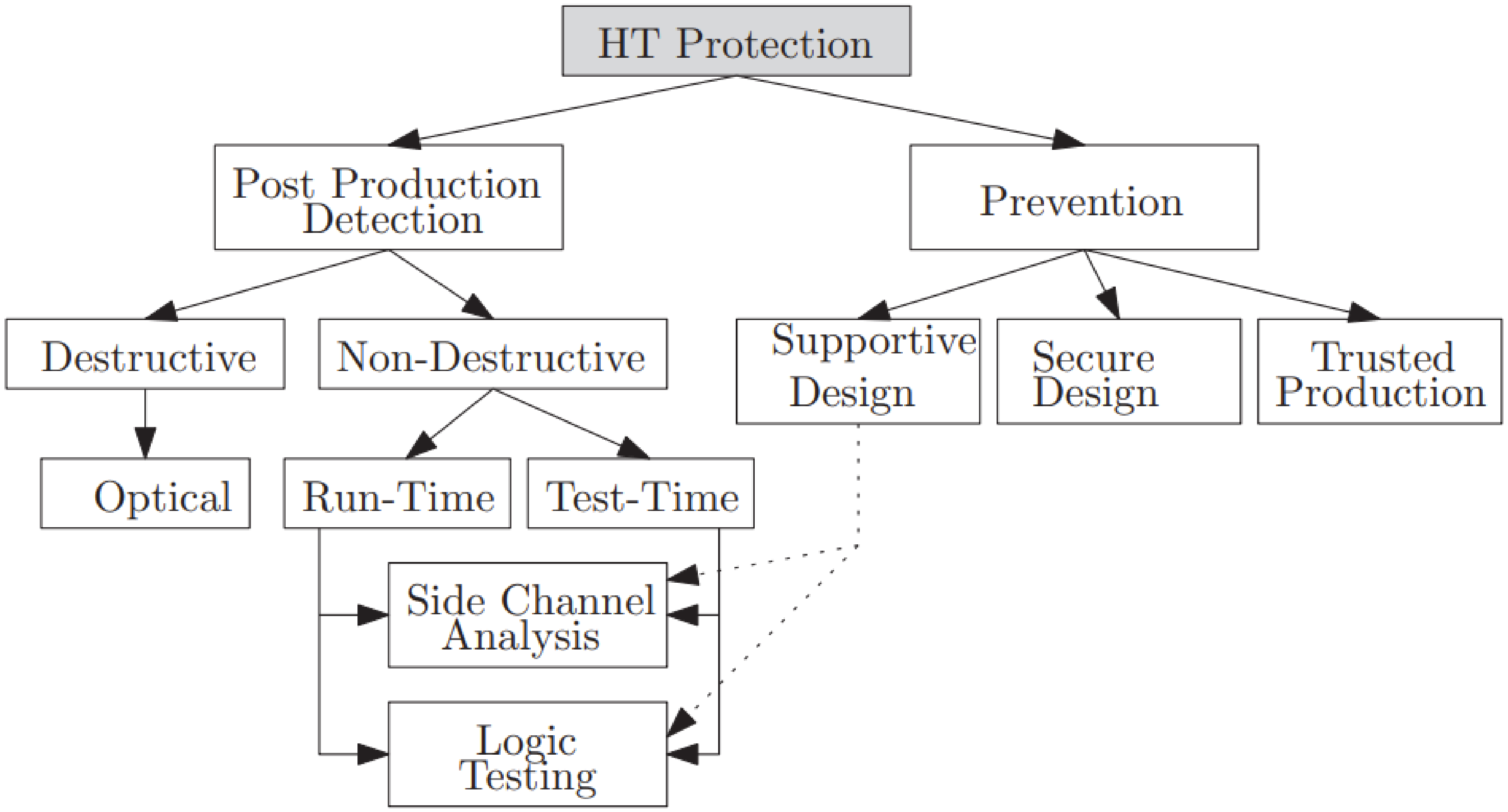


## Hardware Trojan

- Additional circuit inserted into an IC design at RTL or gate level for malicious purposes;
- Malicious modification of a circuit.
- Usually stealthy to escape verification and manufacturing test processes
- Detection is very difficult – there may be no Trojan-free reference for comparison



# Hardware Trojan Detection Approaches

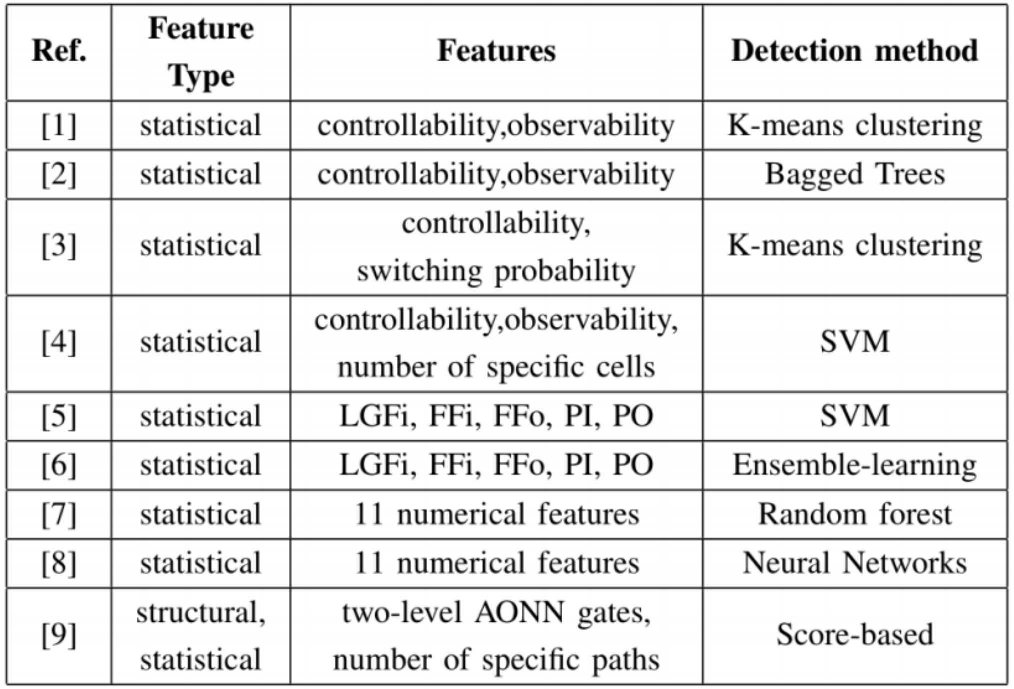


J. Francq, F. Frick, Introduction to hardware Trojan detection methods, Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015.

# Previous Work –HT Feature Extraction

- Most of the previous research extract HT features by statistical analysis of netlist information.
- Most of them need knowledge driven approaches for the features selection and weight adjustment.

A SUMMARIZATION OF GATE-LEVEL HT FEATURES EXTRACTED IN PREVIOUS RESEARCH



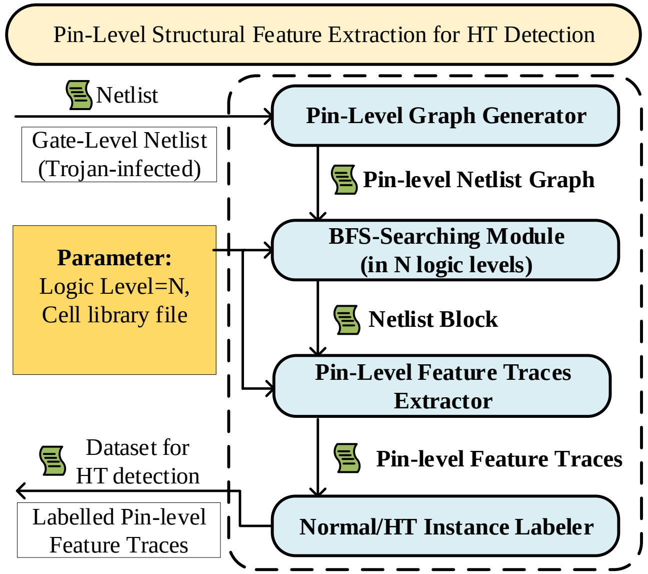
| Ref. | Feature Type | Features | Detection method |
|---|---|---|---|
| [1] | statistical | controllability,observability | K-means clustering |
| [2] | statistical | controllability,observability | Bagged Trees |
| [3] | statistical | controllability, switching probability | K-means clustering |
| [4] | statistical | controllability,observability, number of specific cells | SVM |
| [5] | statistical | LGFi, FFi, FFo, PI, PO | SVM |
| [6] | statistical | LGFi, FFi, FFo, PI, PO | Ensemble-learning |
| [7] | statistical | 11 numerical features | Random forest |
| [8] | statistical | 11 numerical features | Neural Networks |
| [9] | structural, statistical | two-level AONN gates, number of specific paths | Score-based |

[1] H. Salmani, "Cotd: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist "
[2] C. H. Kok, "Classification of Trojan nets based on scoap values using supervised learning"
[3] Y. He "Trigger identification using difference-amplified controllability and dynamic transition probability for hardware Trojan detection"
[4] X. Xie, "Hardware Trojans classification based on controllability and observability in gate-level netlist"
[5] K. Hasegawa, "Hardware Trojans classification for gate-level netlists based on machine learning"
[6] Y. Wang, "Ensemble-learning-based hardware Trojans detection method by detecting the trigger nets"
[7] K. Hasegawa "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier"
[8] K. Hasegawa, "Hardware Trojans classification for gate-level netlists using multilayer neural networks"
[9] Q. Liu, "A hardware Trojan detection method based on structural features of Trojan and host circuits"

# DL-based HT Detection Methods (Data-driven)

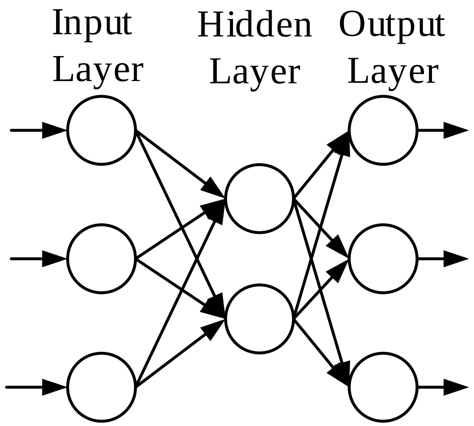
- ❑ Data-driven HT detection that can effectively detect HTs without any prior knowledge of the circuits
- ❑ Natural language processing (NLP) technique for information encoding;
- ❑ DL-based classification models for HT detection (tested using LSTM and CNN models).

Netlists


Pin-Level Structural Feature Extraction for HT Detection
Netlist
Gate-Level Netlist (Trojan-infected)
Pin-Level Graph Generator
Pin-level Netlist Graph
Parameter: Logic Level=N, Cell library file
BFS-Searching Module (in N logic levels)
Netlist Block
Pin-Level Feature Traces Extractor
Pin-level Feature Traces
Dataset for HT detection
Labelled Pin-level Feature Traces
Normal/HT Instance Labeler

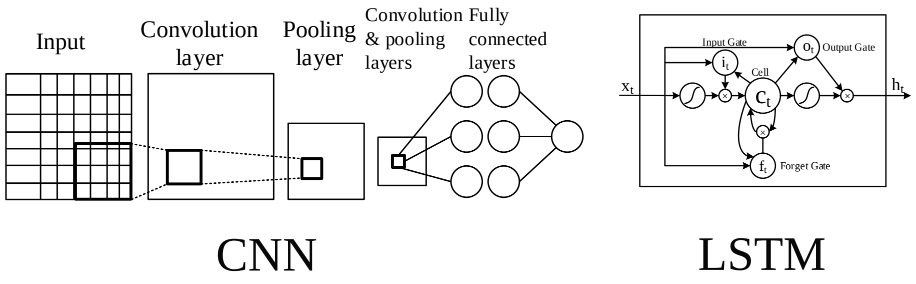
*Netlist Information Extraction Strategy*

PCP Feature Traces



*NLP-based Feature traces Encoding*

Encoded Feature Traces



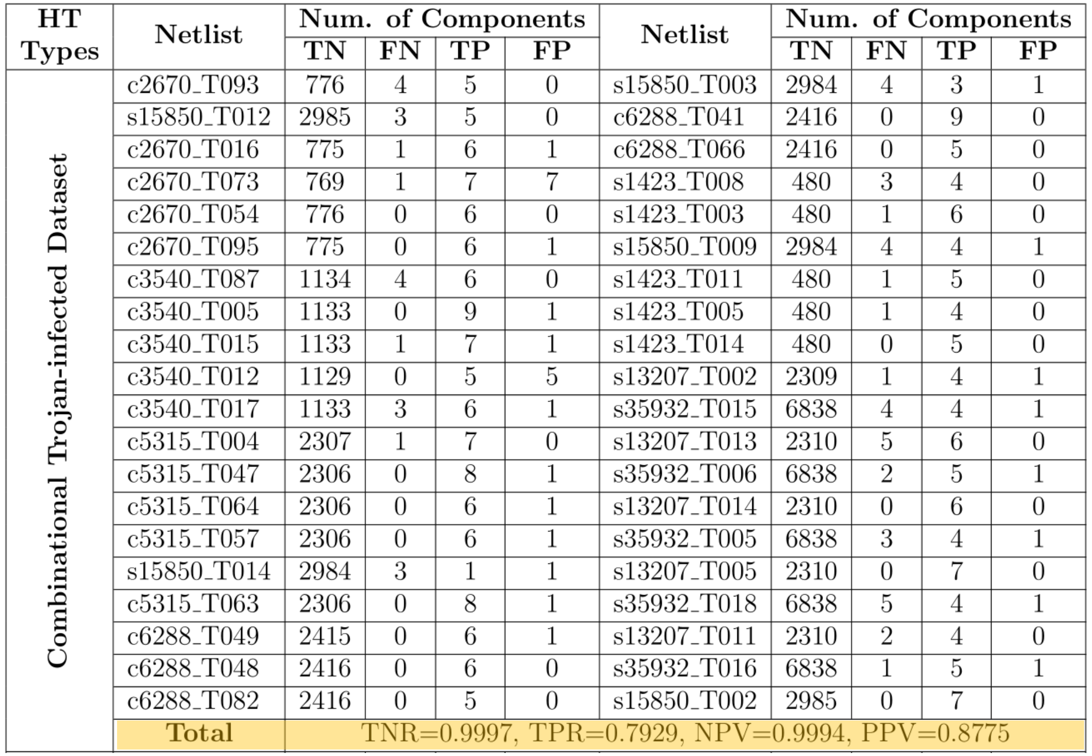
*Deep Learning Module for HT Detection*

Shichao Yu, Chongyan Gu, Weiqiang Liu and Maire O’Neill, **A Novel Feature Extraction Strategy for Hardware Trojan Detection**," In Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), pages 1-5, Seville, Spain, Oct. 2020

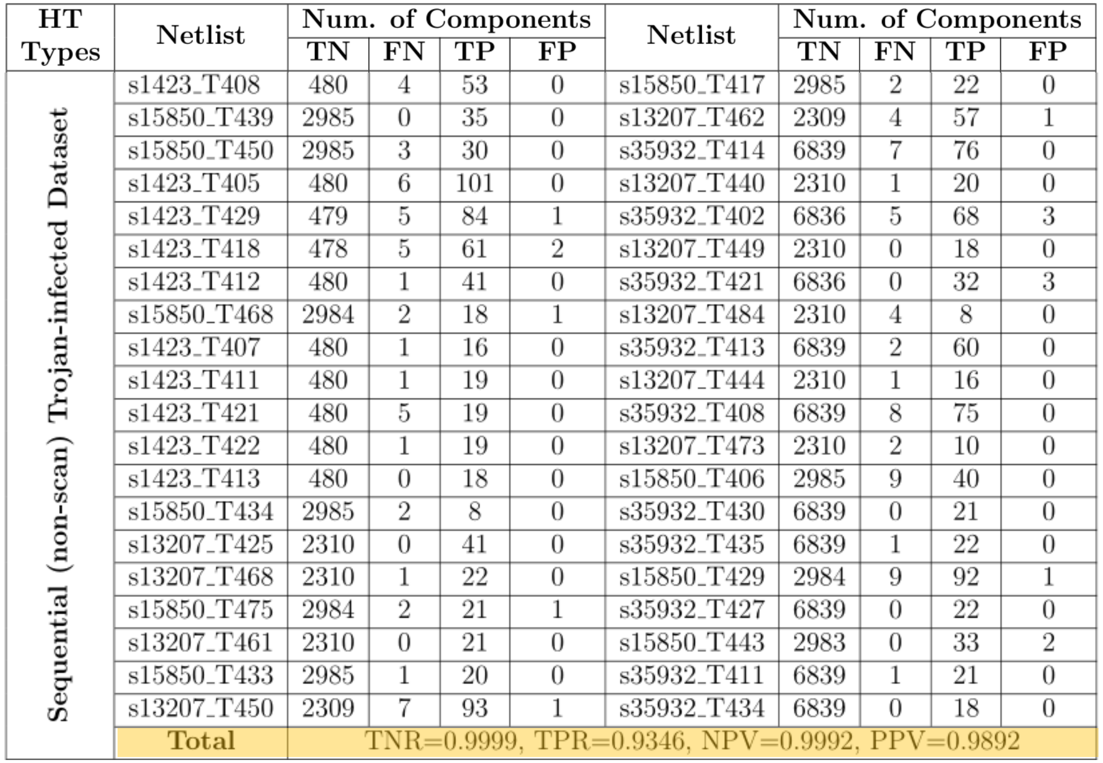
Shichao Yu, Chongyan Gu, Weiqiang Liu and Maire O’Neill, **Deep Learning-based Hardware Trojan Detection with Block-based Netlist Information Extraction**," In IEEE Trans. Emerg. Topics Comput., Oct. 2021

# DL-Based HT Detection System Evaluation

- ❑ Trust-Hub LEDA library containing 914 HT-infected netlist samples are utilized for evaluation.



| HT Types | Netlist | Num. of Components | | | | Netlist | Num. of Components | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | TN | FN | TP | FP | | TN | FN | TP | FP |
| Combinational Trojan-infected Dataset | c2670_T093 | 776 | 4 | 5 | 0 | s15850_T003 | 2984 | 4 | 3 | 1 |
| | s15850_T012 | 2985 | 3 | 5 | 0 | c6288_T041 | 2416 | 0 | 9 | 0 |
| | c2670_T016 | 775 | 1 | 6 | 1 | c6288_T066 | 2416 | 0 | 5 | 0 |
| | c2670_T073 | 769 | 1 | 7 | 7 | s1423_T008 | 480 | 3 | 4 | 0 |
| | c2670_T054 | 776 | 0 | 6 | 0 | s1423_T003 | 480 | 1 | 6 | 0 |
| | c2670_T095 | 775 | 0 | 6 | 1 | s15850_T009 | 2984 | 4 | 4 | 1 |
| | c3540_T087 | 1134 | 4 | 6 | 0 | s1423_T011 | 480 | 1 | 5 | 0 |
| | c3540_T005 | 1133 | 0 | 9 | 1 | s1423_T005 | 480 | 1 | 4 | 0 |
| | c3540_T015 | 1133 | 1 | 7 | 1 | s1423_T014 | 480 | 0 | 5 | 0 |
| | c3540_T012 | 1129 | 0 | 5 | 5 | s13207_T002 | 2309 | 1 | 4 | 1 |
| | c3540_T017 | 1133 | 3 | 6 | 1 | s35932_T015 | 6838 | 4 | 4 | 1 |
| | c5315_T004 | 2307 | 1 | 7 | 0 | s13207_T013 | 2310 | 5 | 6 | 0 |
| | c5315_T047 | 2306 | 0 | 8 | 1 | s35932_T006 | 6838 | 2 | 5 | 1 |
| | c5315_T064 | 2306 | 0 | 6 | 1 | s13207_T014 | 2310 | 0 | 6 | 0 |
| | c5315_T057 | 2306 | 0 | 6 | 1 | s35932_T005 | 6838 | 3 | 4 | 1 |
| | s15850_T014 | 2984 | 3 | 1 | 1 | s13207_T005 | 2310 | 0 | 7 | 0 |
| | c5315_T063 | 2306 | 0 | 8 | 1 | s35932_T018 | 6838 | 5 | 4 | 1 |
| | c6288_T049 | 2415 | 0 | 6 | 1 | s13207_T011 | 2310 | 2 | 4 | 0 |
| | c6288_T048 | 2416 | 0 | 6 | 0 | s35932_T016 | 6838 | 1 | 5 | 1 |
| | c6288_T082 | 2416 | 0 | 5 | 0 | s15850_T002 | 2985 | 0 | 7 | 0 |
| | Total | TNR=0.9997, TPR=0.7929, NPV=0.9994, PPV=0.8775 | | | | | | | | |



| HT Types | Netlist | Num. of Components | | | | Netlist | Num. of Components | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | TN | FN | TP | FP | | TN | FN | TP | FP |
| Sequential (non-scan) Trojan-infected Dataset | s1423_T408 | 480 | 4 | 53 | 0 | s15850_T417 | 2985 | 2 | 22 | 0 |
| | s15850_T439 | 2985 | 0 | 35 | 0 | s13207_T462 | 2309 | 4 | 57 | 1 |
| | s15850_T450 | 2985 | 3 | 30 | 0 | s35932_T414 | 6839 | 7 | 76 | 0 |
| | s1423_T405 | 480 | 6 | 101 | 0 | s13207_T440 | 2310 | 1 | 20 | 0 |
| | s1423_T429 | 479 | 5 | 84 | 1 | s35932_T402 | 6836 | 5 | 68 | 3 |
| | s1423_T418 | 478 | 5 | 61 | 2 | s13207_T449 | 2310 | 0 | 18 | 0 |
| | s1423_T412 | 480 | 1 | 41 | 0 | s35932_T421 | 6836 | 0 | 32 | 3 |
| | s15850_T468 | 2984 | 2 | 18 | 1 | s13207_T484 | 2310 | 4 | 8 | 0 |
| | s1423_T407 | 480 | 1 | 16 | 0 | s35932_T413 | 6839 | 2 | 60 | 0 |
| | s1423_T411 | 480 | 1 | 19 | 0 | s13207_T444 | 2310 | 1 | 16 | 0 |
| | s1423_T421 | 480 | 5 | 19 | 0 | s35932_T408 | 6839 | 8 | 75 | 0 |
| | s1423_T422 | 480 | 1 | 19 | 0 | s13207_T473 | 2310 | 2 | 10 | 0 |
| | s1423_T413 | 480 | 0 | 18 | 0 | s15850_T406 | 2985 | 9 | 40 | 0 |
| | s15850_T434 | 2985 | 2 | 8 | 0 | s35932_T430 | 6839 | 0 | 21 | 0 |
| | s13207_T425 | 2310 | 0 | 41 | 0 | s35932_T435 | 6839 | 1 | 22 | 0 |
| | s13207_T468 | 2310 | 1 | 22 | 0 | s15850_T429 | 2984 | 9 | 92 | 1 |
| | s15850_T475 | 2984 | 2 | 21 | 1 | s35932_T427 | 6839 | 0 | 22 | 0 |
| | s13207_T461 | 2310 | 0 | 21 | 0 | s15850_T443 | 2983 | 0 | 33 | 2 |
| | s15850_T433 | 2985 | 1 | 20 | 0 | s35932_T411 | 6839 | 1 | 21 | 0 |
| | s13207_T450 | 2309 | 7 | 93 | 1 | s35932_T434 | 6839 | 0 | 18 | 0 |
| | Total | TNR=0.9999, TPR=0.9346, NPV=0.9992, PPV=0.9892 | | | | | | | | |

- ❑ 79% TPR, 99% TNR, 87% PPV and 99% NPV for **combinational** Trojan detection (40 training samples/40 validating samples, 5 epochs, LSTM);
- ❑ 93% TPR, 99% TNR, 98% PPV and 99% NPV for **sequential** Trojan detection (40 training samples/40 validating samples, 5 epochs, LSTM)

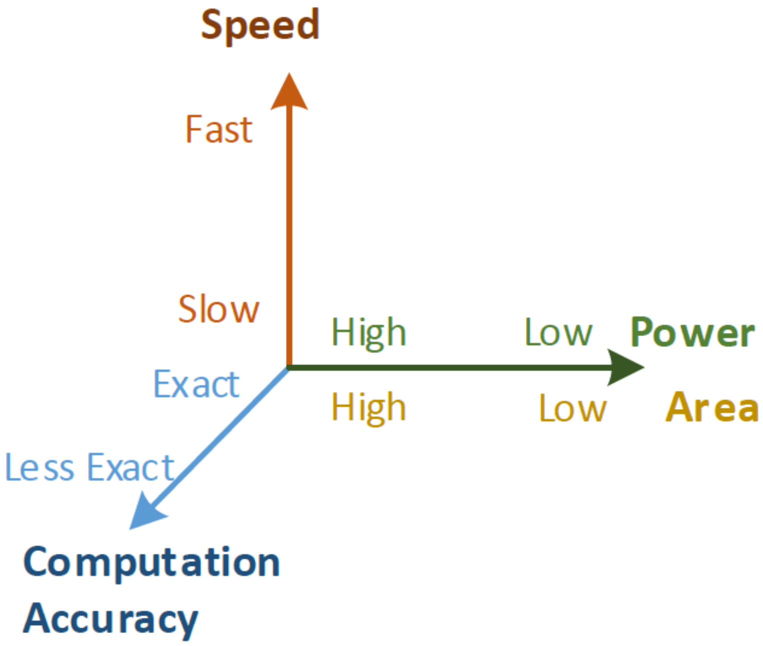
# Security in/for Approximate Computing

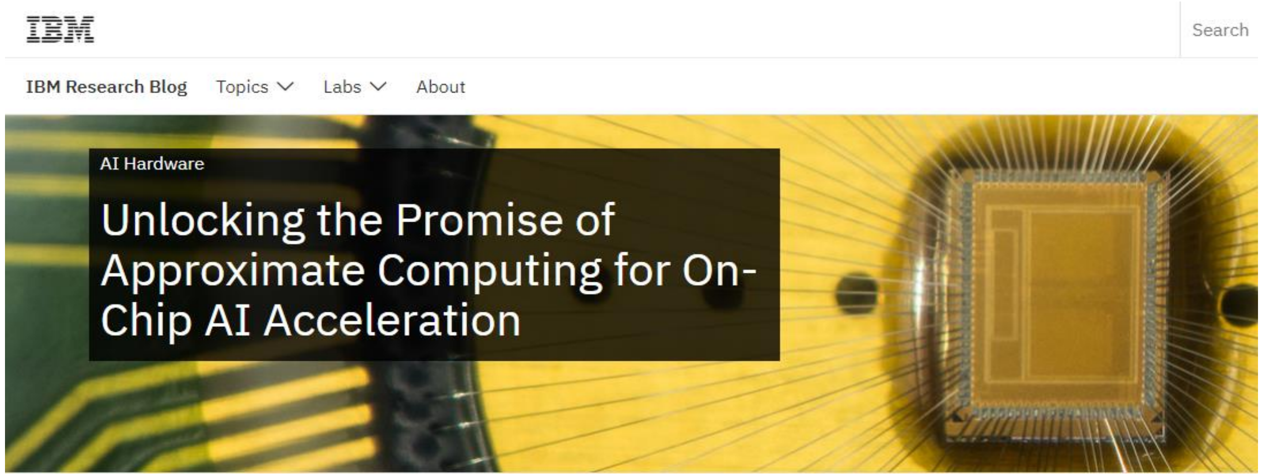
CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Approximate Computing

Approximate computing has emerged as a potentially preferable paradigm for energy-efficient applications that are error-tolerant

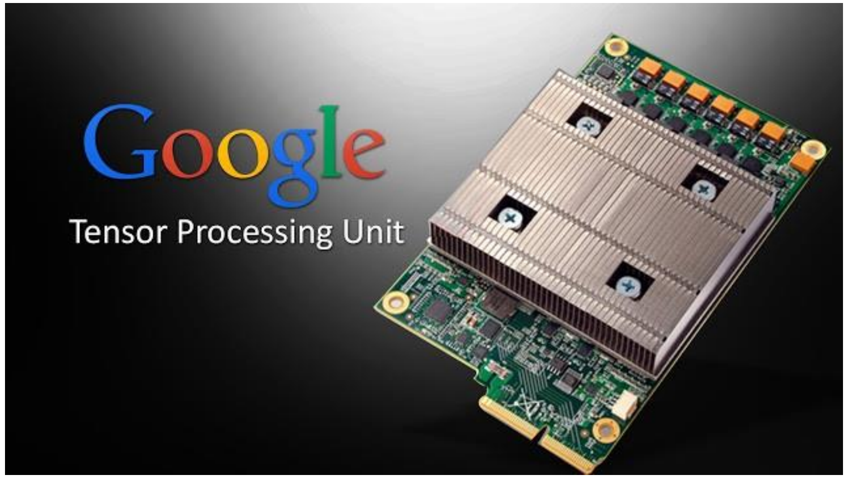
Example applications: AI, robotics, image and video processing, NLP



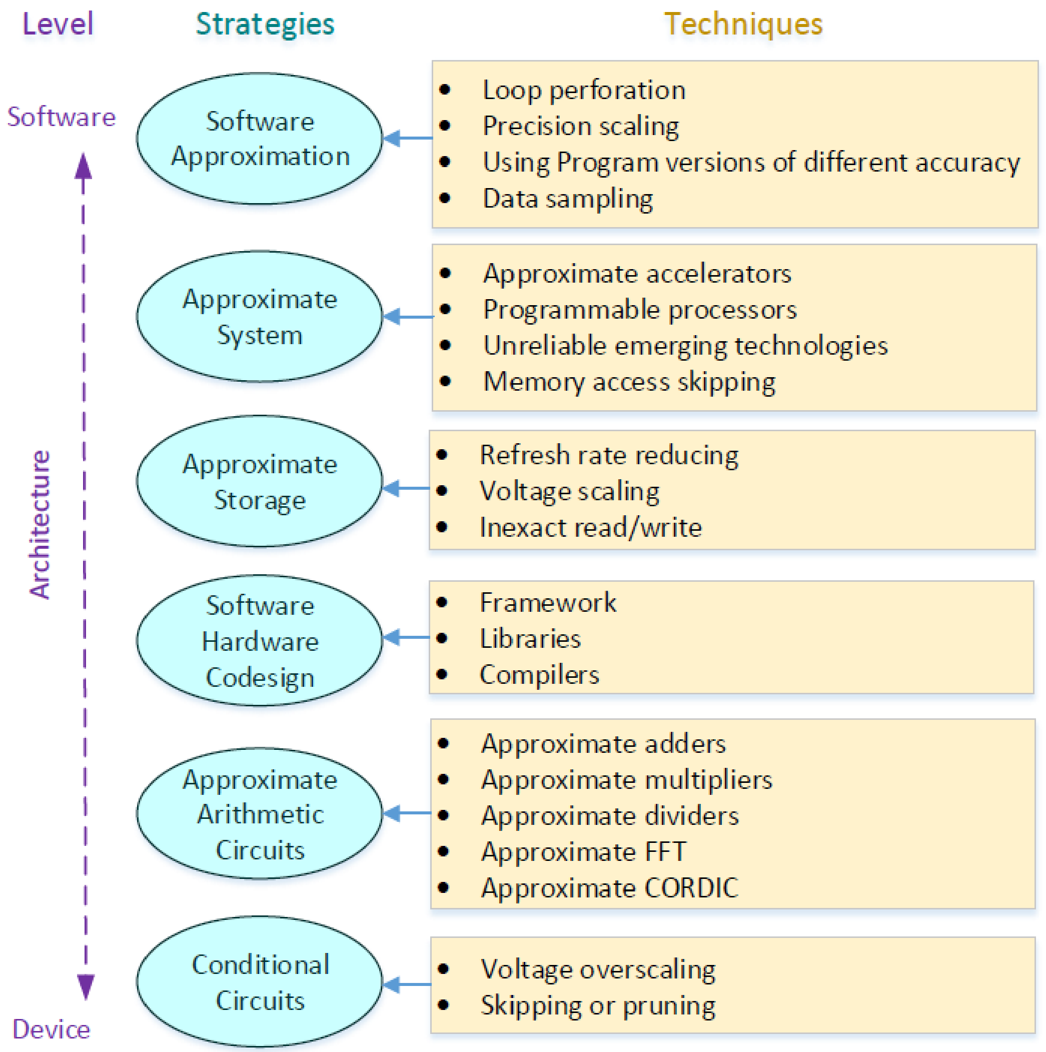


AI Hardware

Unlocking the Promise of Approximate Computing for On-Chip AI Acceleration



# Approximate Computing Strategies



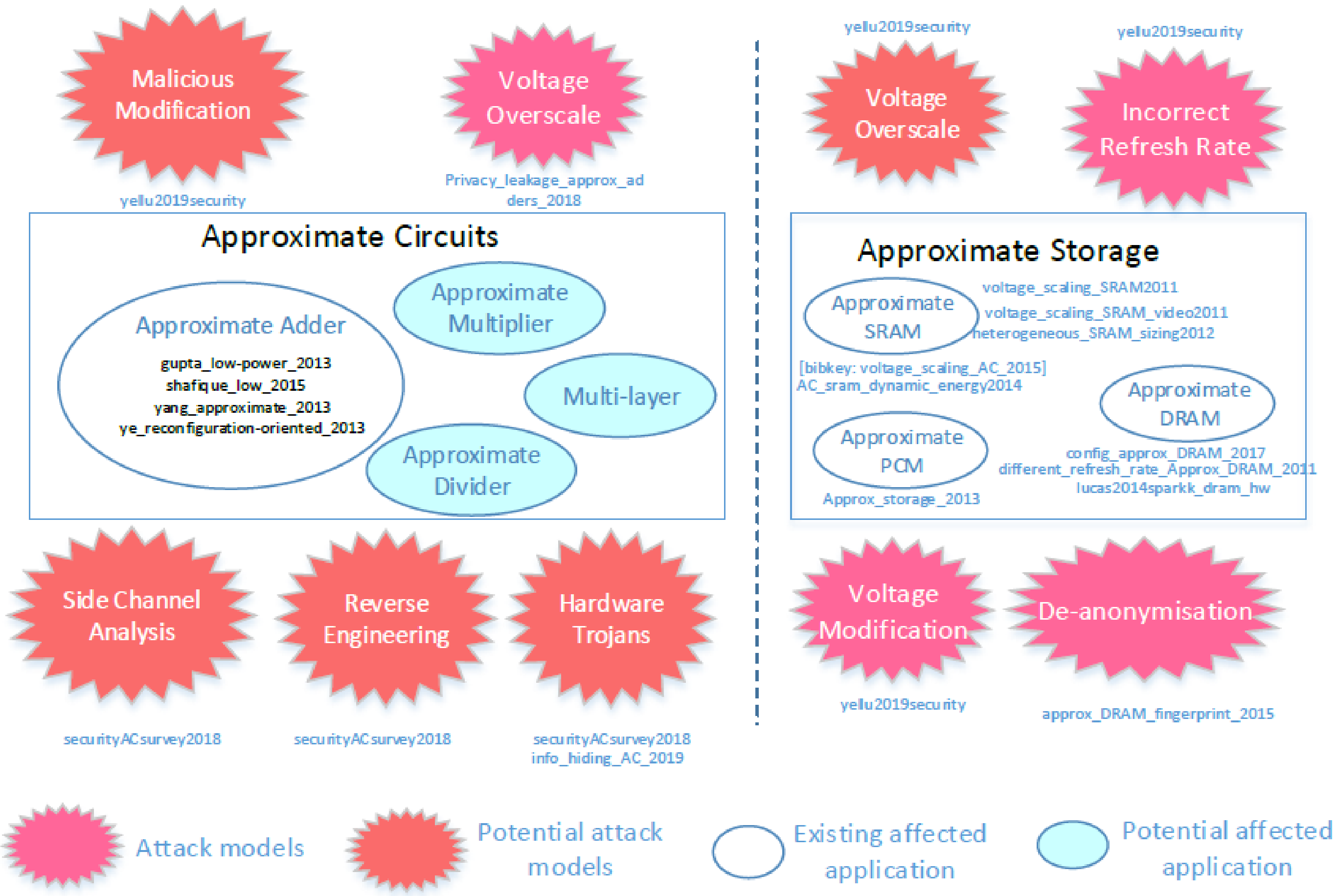
| Level | Strategies | Techniques |
|---|---|---|
| Software | Software Approximation | Loop perforation<br>Precision scaling<br>Using Program versions of different accuracy<br>Data sampling |
| | Approximate System | Approximate accelerators<br>Programmable processors<br>Unreliable emerging technologies<br>Memory access skipping |
| Architecture | Approximate Storage | Refresh rate reducing<br>Voltage scaling<br>Inexact read/write |
| | Software Hardware Codesign | Framework<br>Libraries<br>Compilers |
| | Approximate Arithmetic Circuits | Approximate adders<br>Approximate multipliers<br>Approximate dividers<br>Approximate FFT<br>Approximate CORDIC |
| Device | Conditional Circuits | Voltage overscaling<br>Skipping or pruning |

**Approximate computing strategies and techniques**

- Approximate computing can be applied to different levels, from hardware to software.
- A system with approximation should have the same **security** as its non-approximation parts.
- However, the reality is that approximate computing may be even more **vulnerable**.

CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Security Threats in Approximate Computing



Security threats in Approximate Computing

Security attacks & solutions for Approximate Circuits and Memory

- SCA
- Hardware Trojans
- Approximate PUF

Using Approximate computing to improve security

- Advanced Crypto

CSIT CENTRE FOR SECURE INFORMATION TECHNOLOGIES

# Thank you