From Known Attacks To New Exploits

Shivam Bhasin

Temasek Labs
NTU, Singapore

VLSI Design & ES 2020 Tutorials
4-8 January 2020

Who Am I?

- Sr. Research Scientist, Center For Hardware Assurance @ Temasek Labs, NTU, Singapore
  - PhD from Telecom Paristech, France (2011)
  - M.S from Mines St Etienne, France (2008)
- Research Interest:
  - Physical Attacks (Side-Channel, Fault Injection,, Hardware Trojan, Combinations)
  - Countermeasure & Certification
  - Hardware Security of AI

Centre For Hardware Assurance (CHA)

CHA

- Physical Analysis & Crypto. Engg (PACE) Group
- Hardware Assurance (HA) Group
- Circuit Analysis (CA) Group

Centre For Hardware Assurance (CHA)

CHA

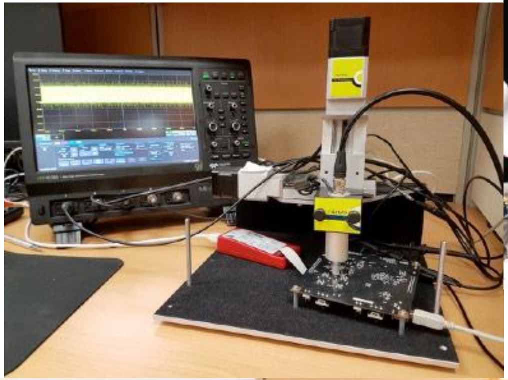
- **Physical Analysis & Crypto. Engg (PACE) Group**
  - Non-Invasive to Semi-invasive
- **Hardware Assurance (HA) Group**
  - Semi-Invasive to Invasive
- **Circuit Analysis (CA) Group**
  - everse Engineering

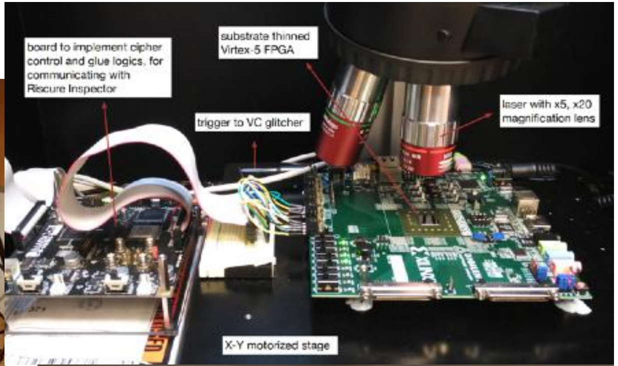
PACE Group

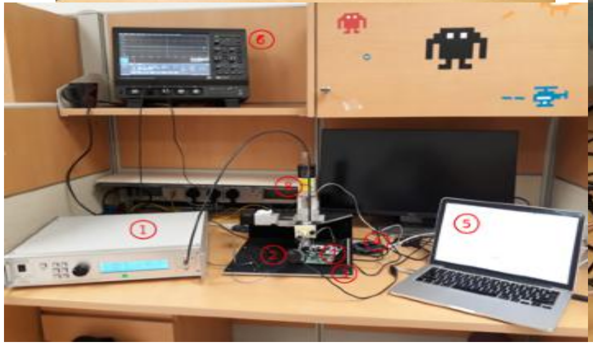
- Side-Channel Attacks
- Fault Injection Attacks
- Hardware Trojan
- Combined Attacks
- Reverse Engineering

(Non-invasive)

- AI Security
- Secure Design
- Certification







PART I

Side Channel Attacks

This Work …

- Reverse Engineering

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …

- Reverse Engineering
- Through Side-Channel

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel.
In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …

- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes

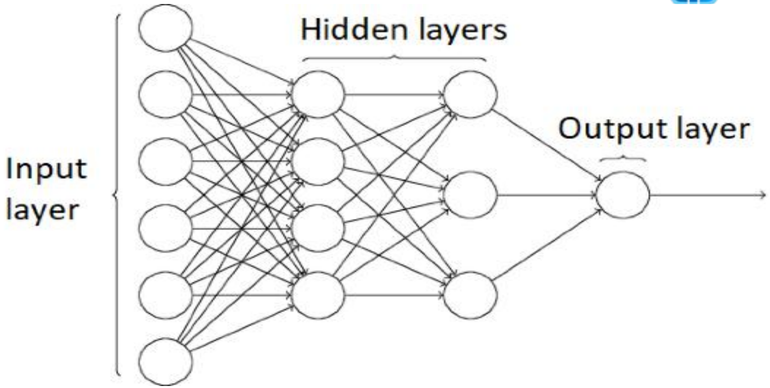
Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …

- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

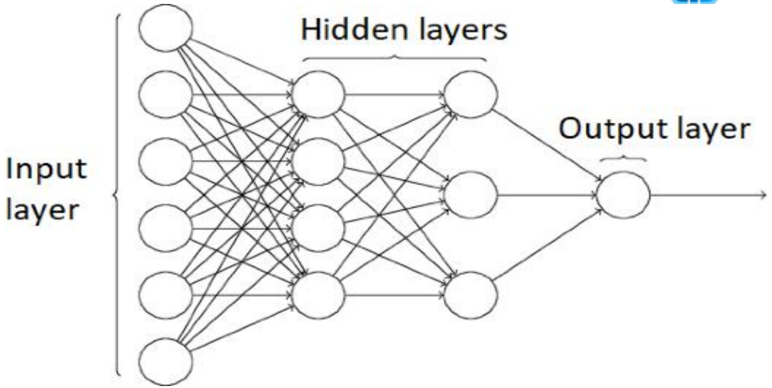
This Work …



- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

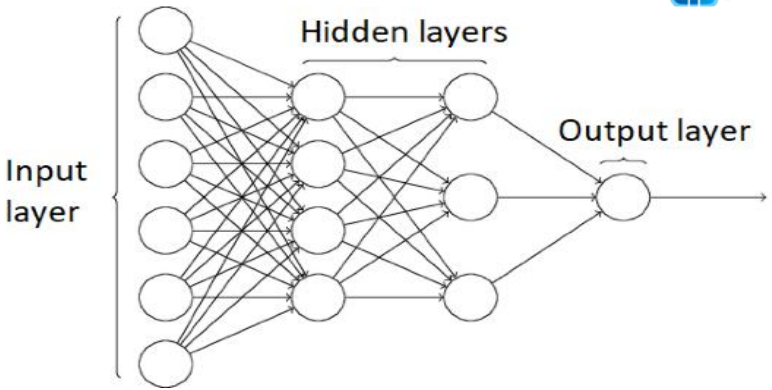
This Work …



- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

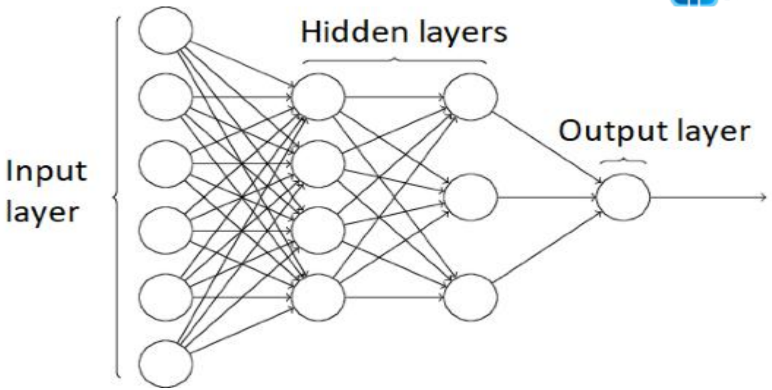
This Work …



- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:
  - Number of layers

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

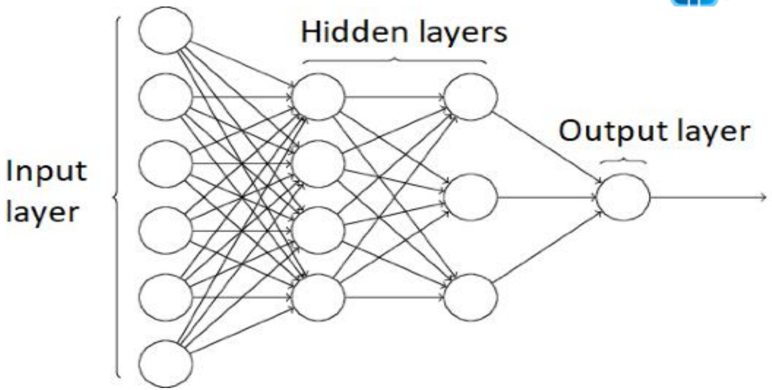
This Work …



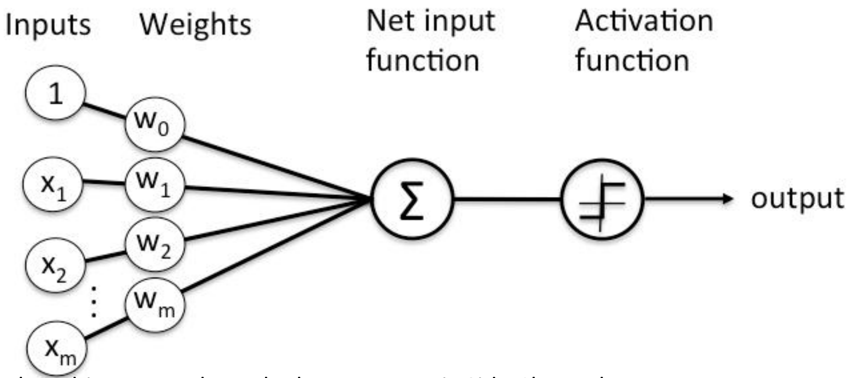
- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:
  - Number of layers
  - Number of neurons in each layer

Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …

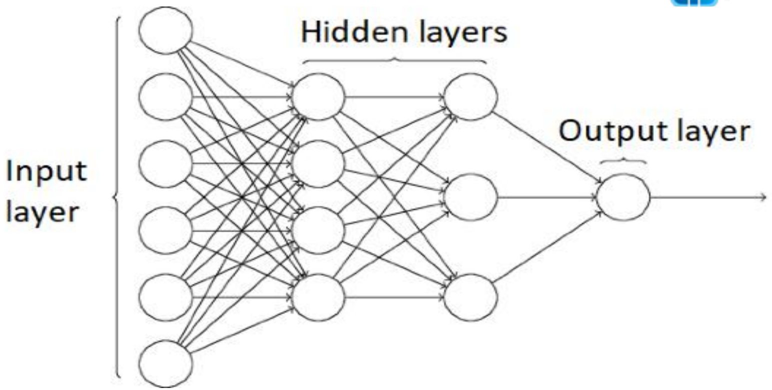


- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:
  - Number of layers
  - Number of neurons in each layer

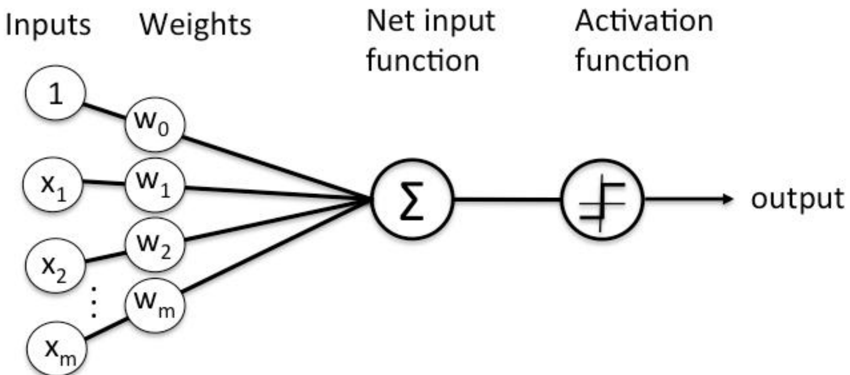


Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …

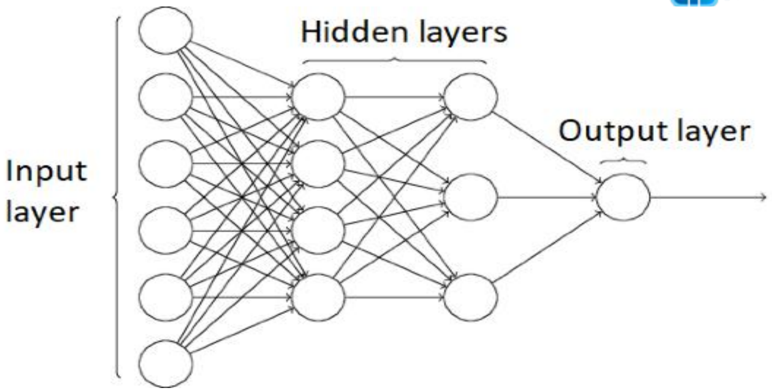


- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:
  - Number of layers
  - Number of neurons in each layer
  - Activation function in each neuron

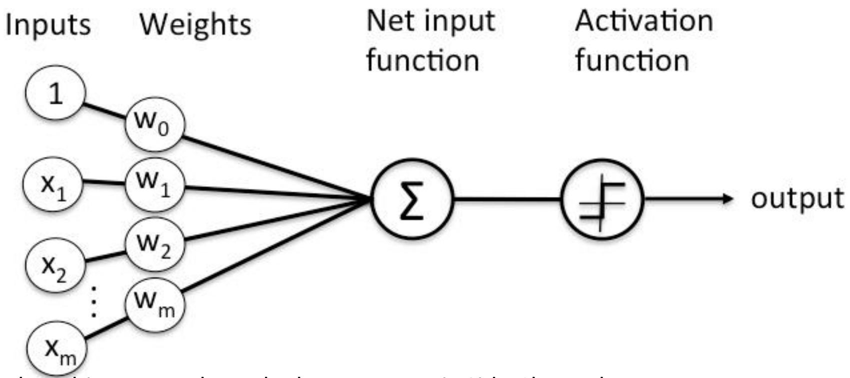


Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

This Work …



- Reverse Engineering
- Through Side-Channel
- Measured by Electromagnetic (EM) probes
- Of Deep Neural Network (DNN) on embedded devices
- To Recover:
  - Number of layers
  - Number of neurons in each layer
  - Activation function in each neuron
  - Input weights to each neuron



Batina, L., Bhasin, S., Jap, D. and Picek, S., 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (pp. 515-532).

Machine Learning & Security

- Machine learning (ML) has wide applications across industries.
- Security is just one popular application for ML
- **US$ 35 Billion Industry by 2024**[1]
- Strong ML models is an asset, on which many companies invest a significant amount of time and money to develop, resulting in Intellectual property
- Leaked models can leak information about sensitive training sets

1 https://www.marketwatch.com/press-release/artificial-intelligence-in-security-market-size-is-projected-to-be-around-us-35-billion-by-2024-2018-10-07

Neural Nets

- We consider neural networks: multilayer perceptron (MLP) and convolutional neural networks (CNN).
- We consider those networks since:
  - they are commonly used machine learning algorithms in modern applications;
  - they consist of different types of layers that are also occurring in other architectures like recurrent neural networks;
  - in the case of MLP, the layers are all identical, which makes it more difficult for SCA and could be consequently considered as the worst-case scenario.

Side-Channel Analysis (SCA)

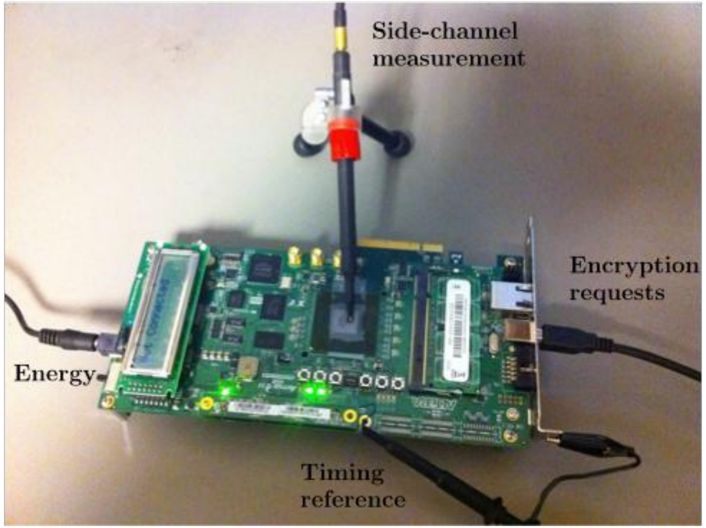
THEN

Side-Channel Analysis (SCA)

THEN

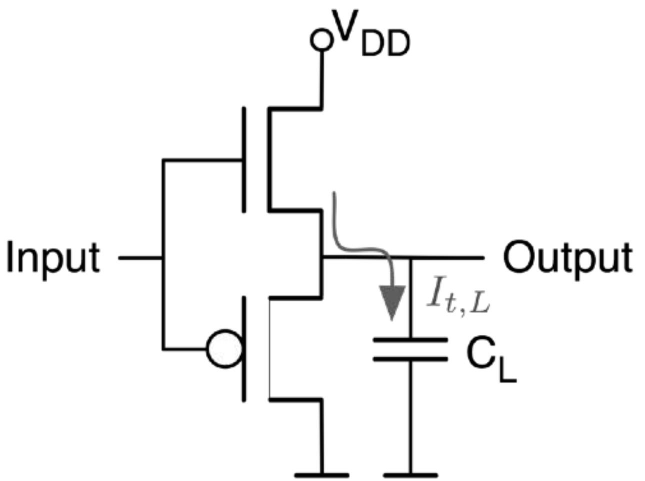
Side-Channel Analysis (SCA)

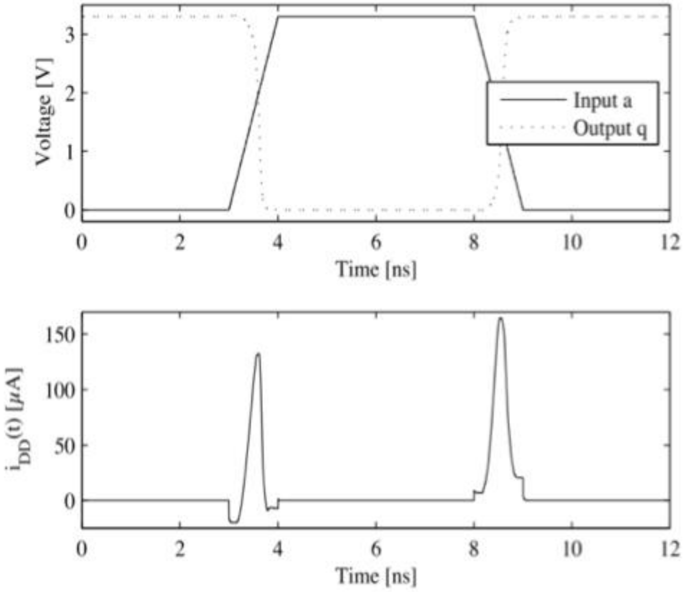
THEN



NOW

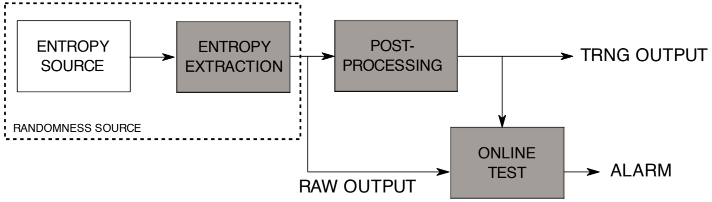
Side-Channel Analysis (SCA)

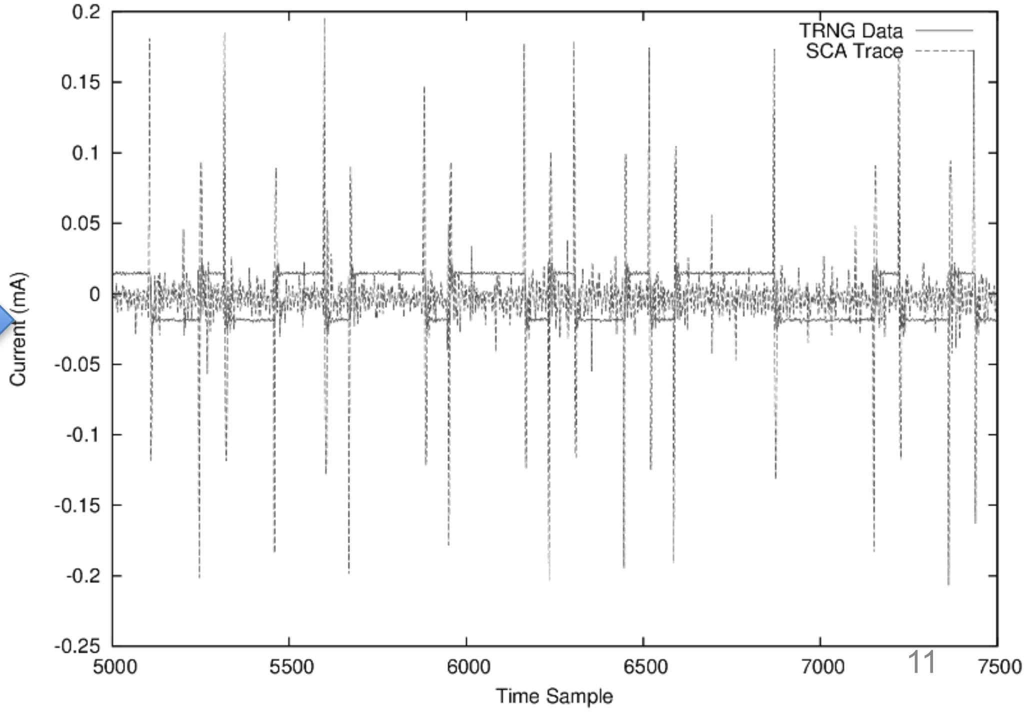




Lets look at a basic CMOS cell

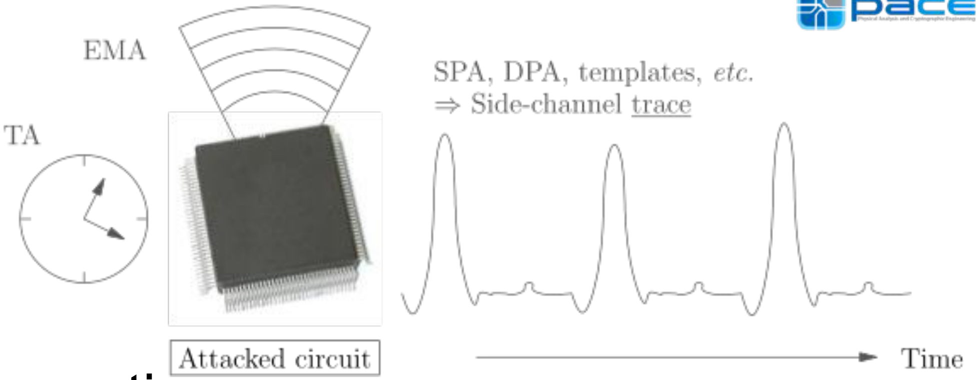
Side-Channel Analysis (SCA)





Extending from one cell to a full circuit

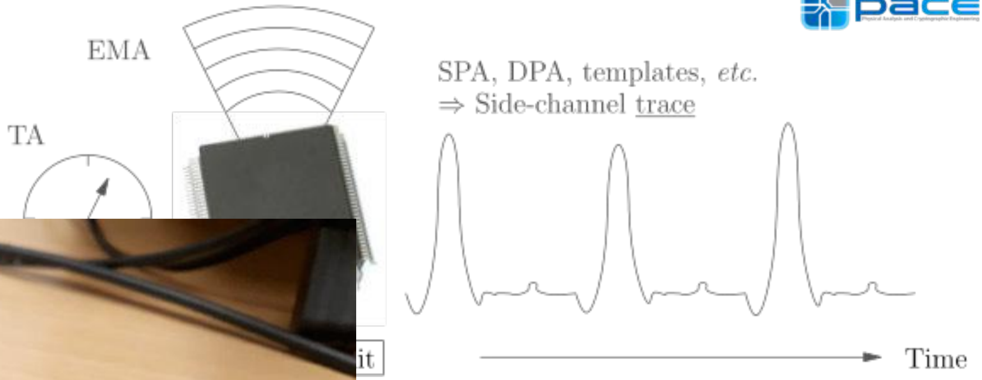
What is SCA?

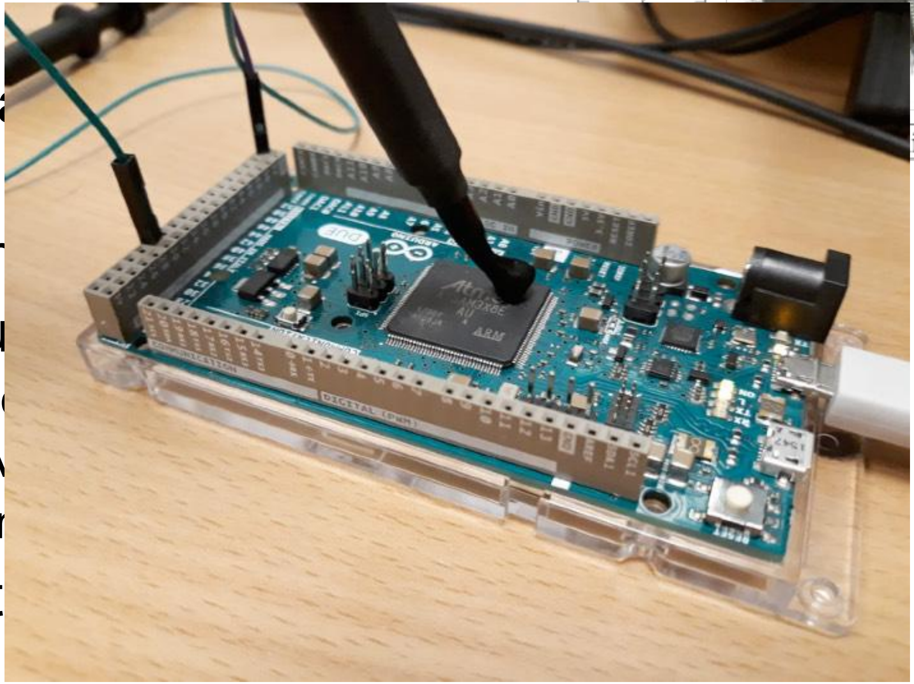


- Non-invasive
- Serious threat to pervasive computing
- Exploiting unintentional EM leakage
- Powerful & practical
  - Keeloq
  - FPGA Bitstream encryption
  - Bitcoin wallets
- Applications beyond secret key recovery

What is SCA?

- Non-inva
- Serious
- Exploitir
- Powerfu
  - Keeloc
  - FPGA
  - Bitcoir
- Applicat

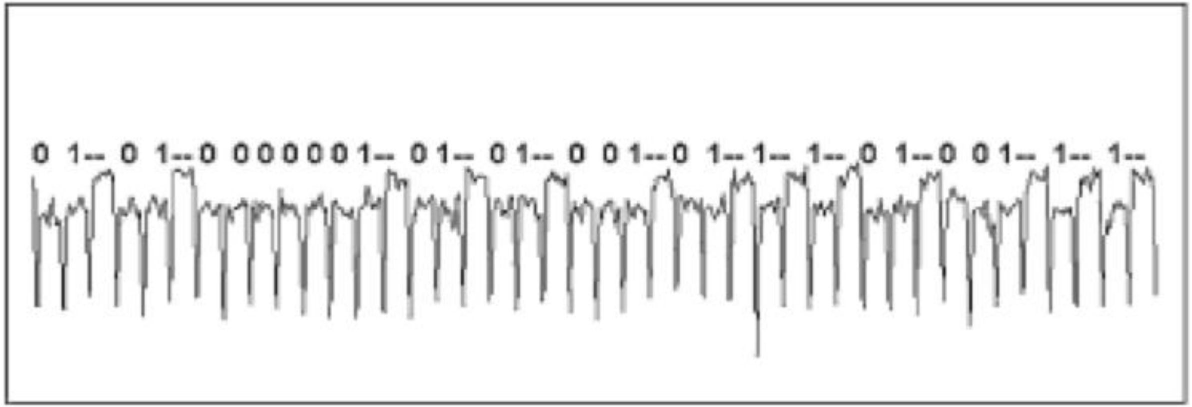




This Work: Electromagnetic (EM) SCA

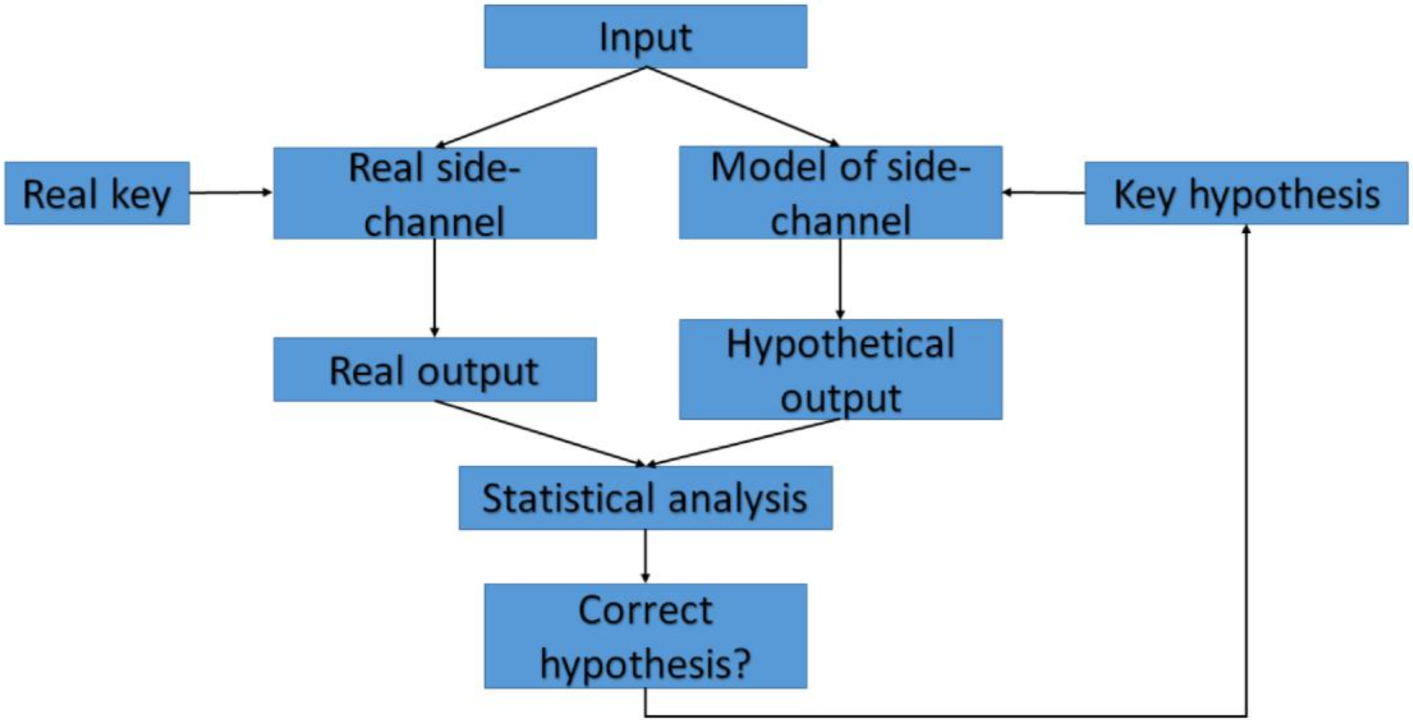
Simple EM Analysis (SEMA)

- Adversary learns secret information **by visual inspection** of (usually single) power/EM measurement
- Ex: observe square & multiply in exponentiation etc.



Source: https://anysilicon.com/side-channel-attacks-differential-power-analysis-dpa-simple-power-analysis-spa-works/

Differential EM Analysis (DEMA)



Differential EM Analysis (DEMA)

- Statistical attack over several EM observation with random/known input to recover **secret key K**
- The attack normally tests for dependencies between actual physical signature (or measurements) and hypothetical physical signature, i.e., predictions on intermediate data. The hypothetical signature is based on a leakage model and key hypothesis.
- Most commonly used leakage model is **Hamming Weight (HW)**
- A microcontroller leaks in HW when sensitive data is loaded to pre-charged data bus
- Similar models known and exploited for FPGA, ASIC, GPU ….

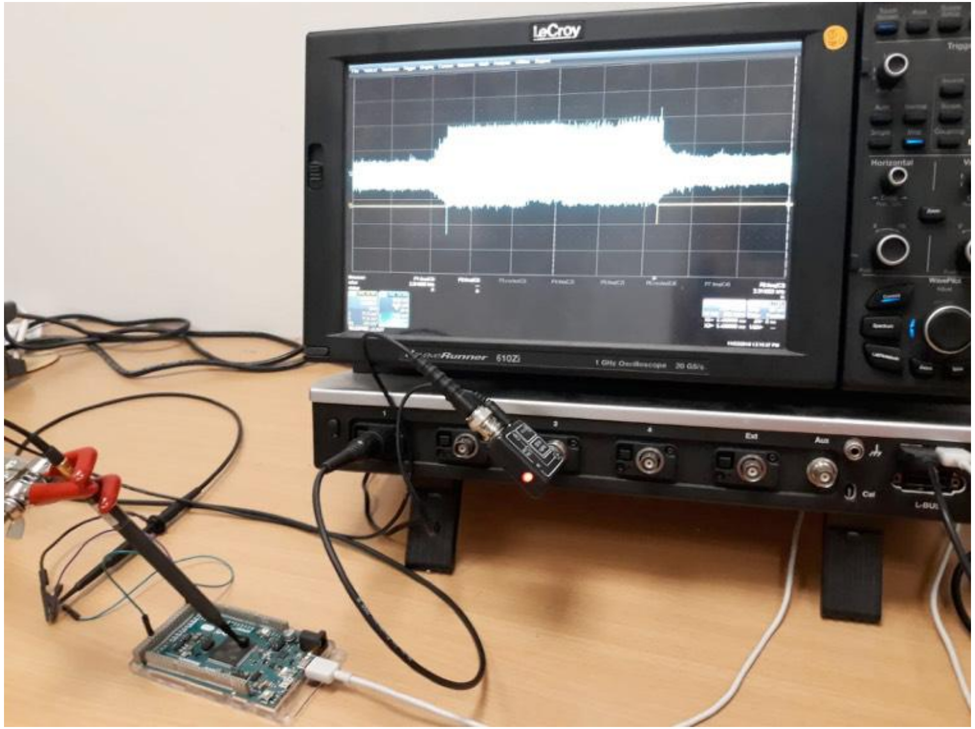
Adversary Model

- Recover the neural network architecture using only side-channel information
- Adversary does not know the architecture of the used network but can feed random/known inputs to the DNN and capture corresponding electromagnetic side-channel traces
- No assumption on the type of inputs; we work with real numbers
- **Assumption**: Implementation of the machine learning algorithm with no side-channel countermeasures

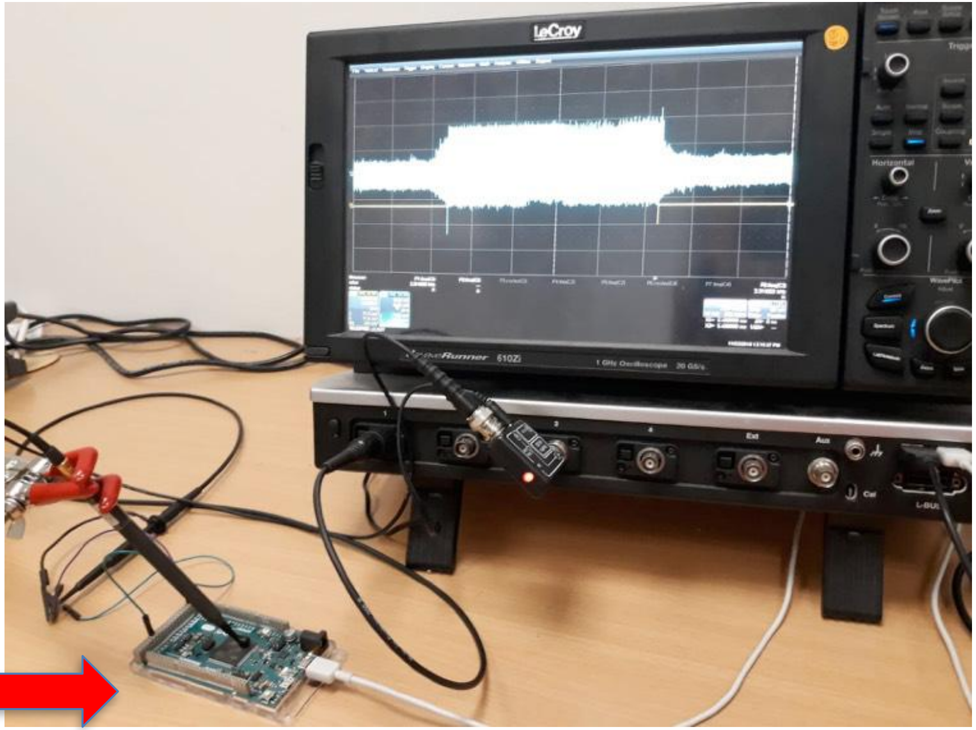
Experimental Setup

- Passive EM Measurement
- Near-field probe
- 30dB pre-amplifier for clear signal
- Measurements averaged for noise filtering
- For bigger networks, measurements are made sequentially for different layers
- Targets: ATMEGA AVR328P, ARM Cortex-M3

Experimental Setup

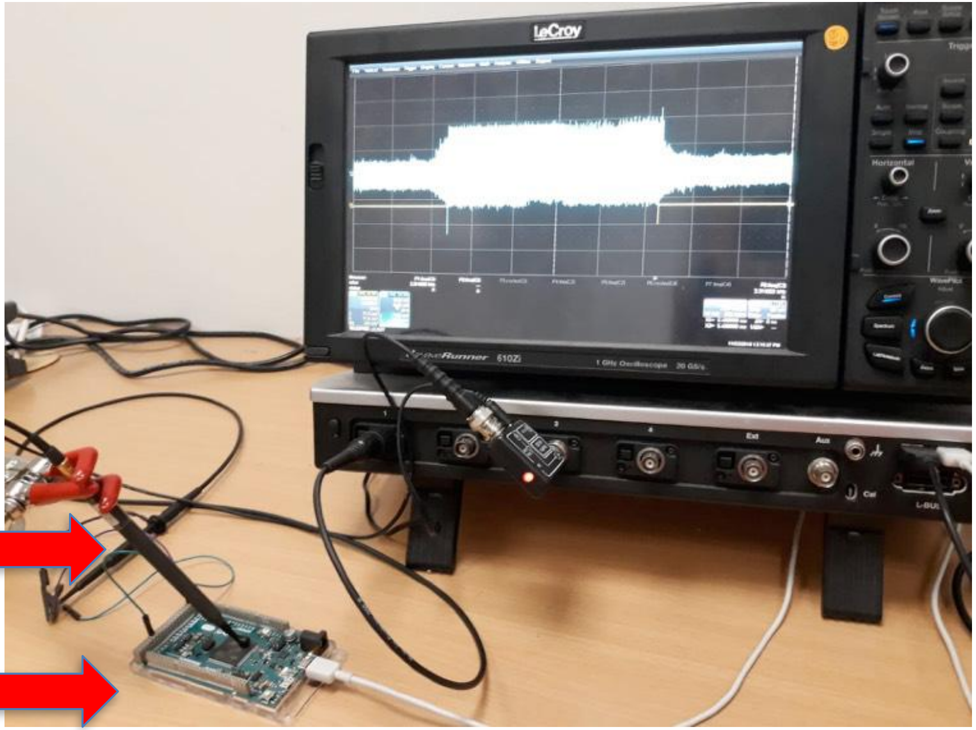


Experimental Setup



Target

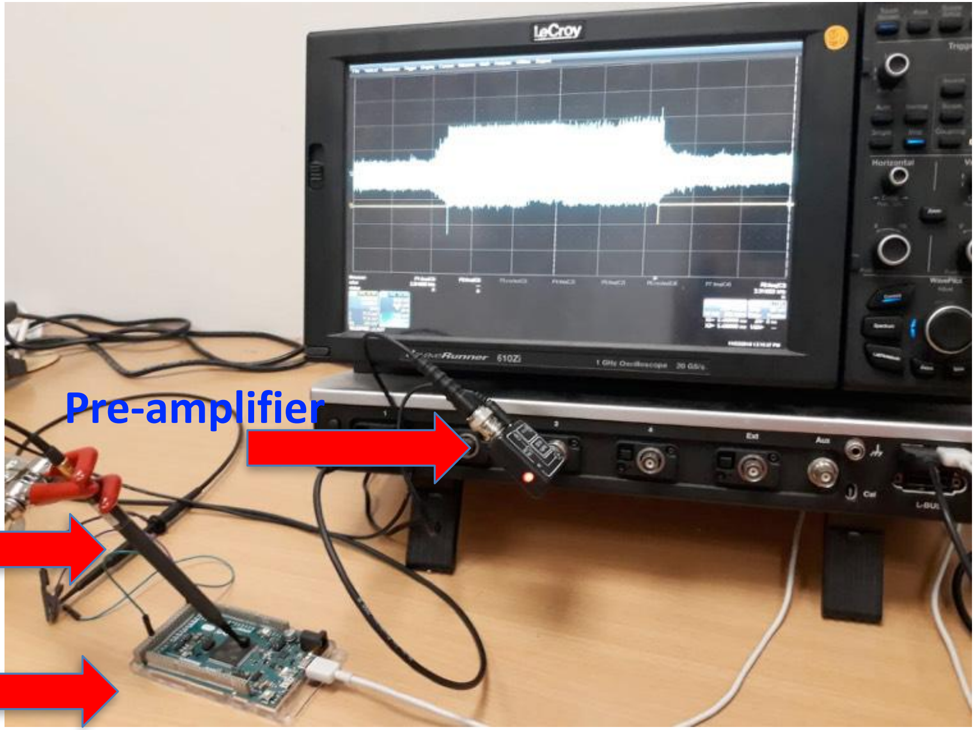
Experimental Setup



Probe

Target

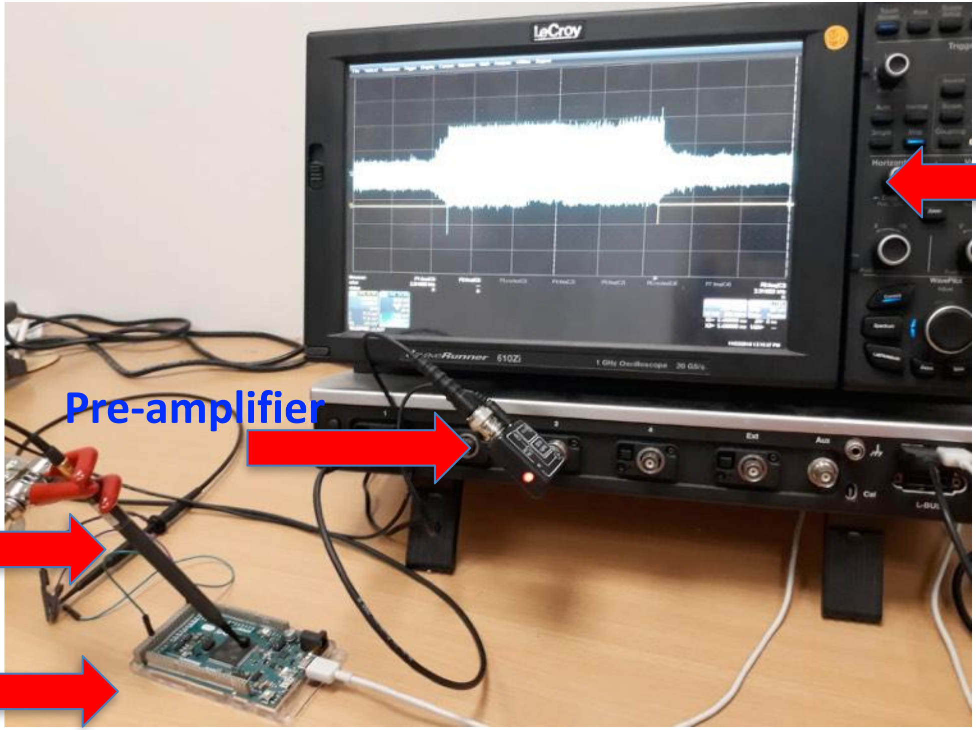
Experimental Setup



Probe

Target

Experimental Setup



Trace

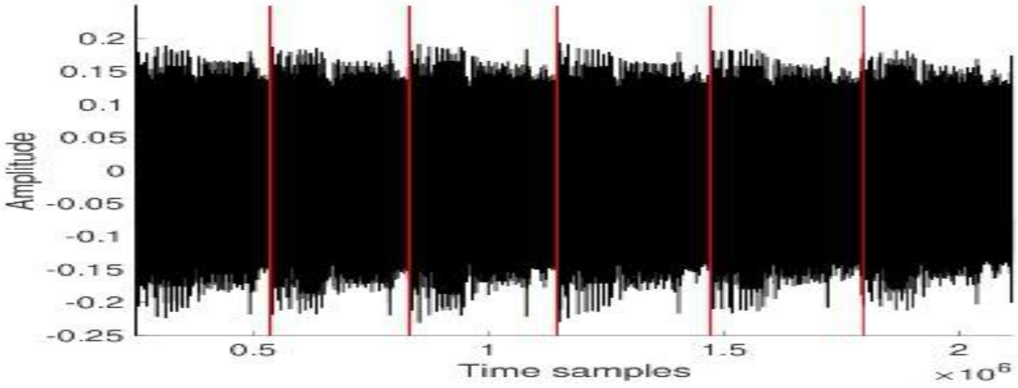
Probe

Target

Lets Start With Some Visual Inspection!!!!

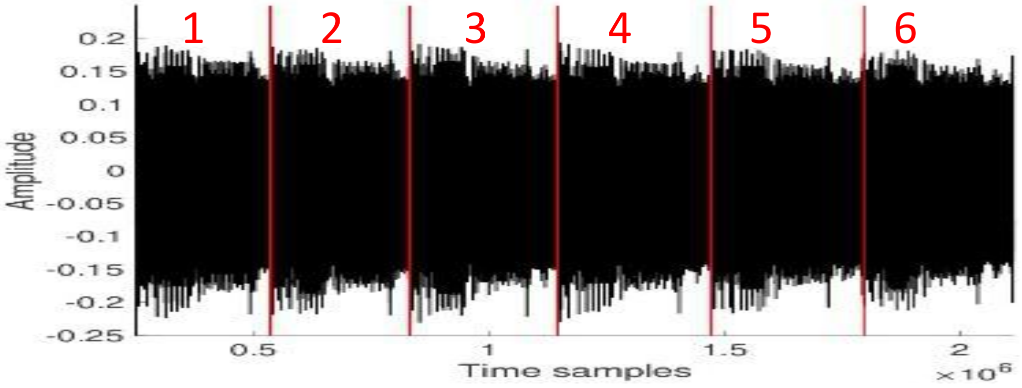
Identifying Neurons

- **Simple EM Analysis**
- Hidden layer with 6 neurons = 6 repeating patterns
- Each neuron executes a series of multiplication, followed by activation
- Activation Function in this case = Sigmoid



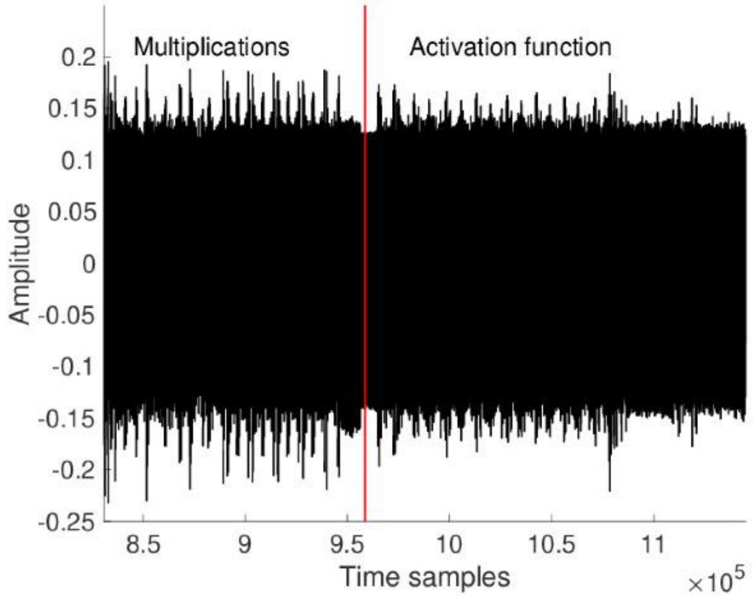
Identifying Neurons

- **Simple EM Analysis**
- Hidden layer with 6 neurons = 6 repeating patterns
- Each neuron executes a series of multiplication, followed by activation
- Activation Function in this case = Sigmoid



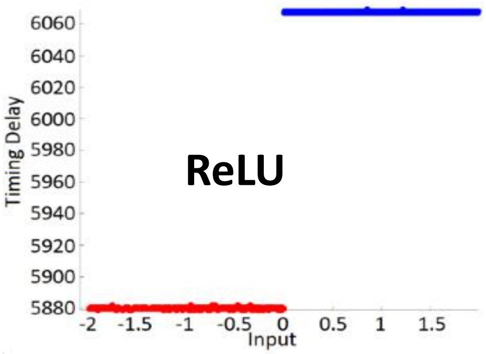
Recovering Activation Function

- **Timing Attack (with EM trace)**
- Each activation function has distinct timing pattern
- Timing patterns can be pre-characterized for different NN libraries
- We measure **precise timing** of activation function using **EM measurement** on oscilloscope.

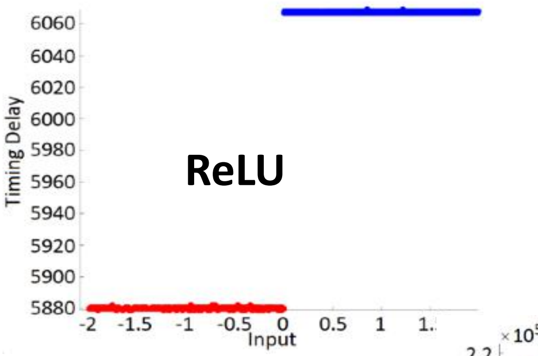


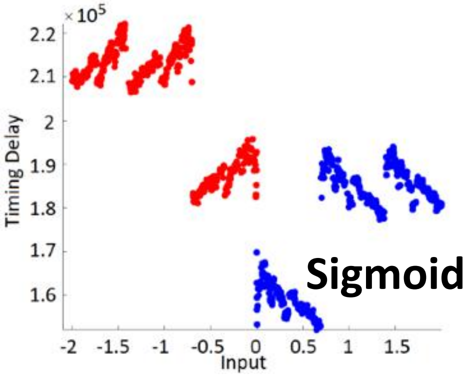
Timing Patterns of Various Activation Function

Timing Patterns of Various Activation Function

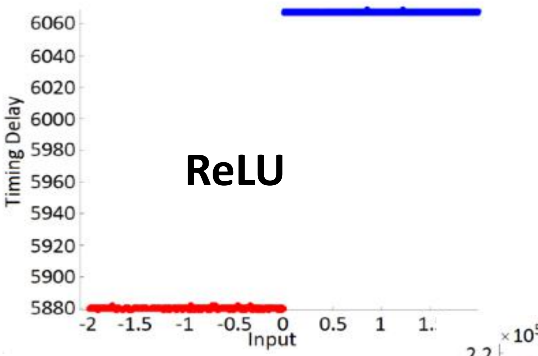


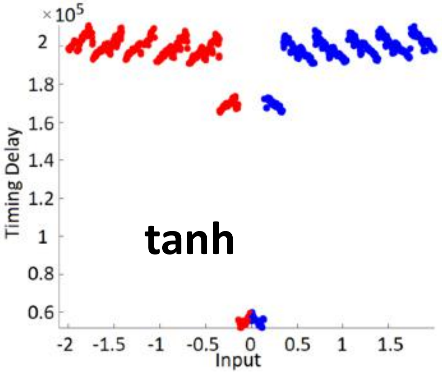
Timing Patterns of Various Activation Function

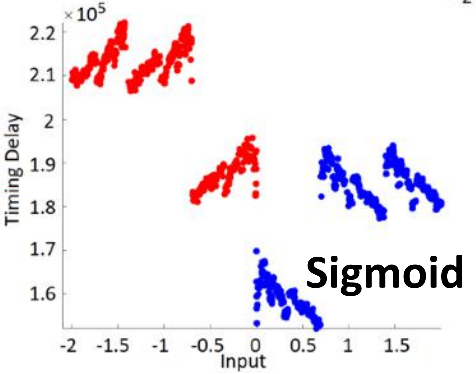




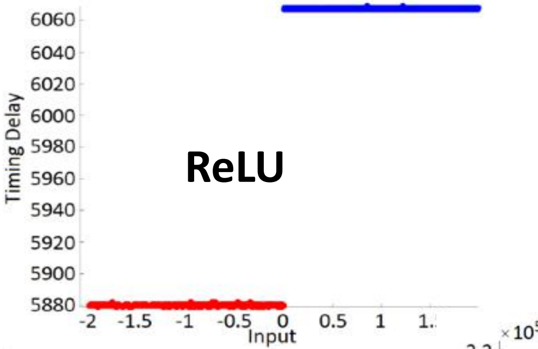
Timing Patterns of Various Activation Function

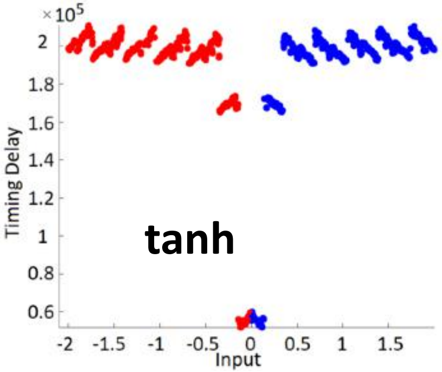


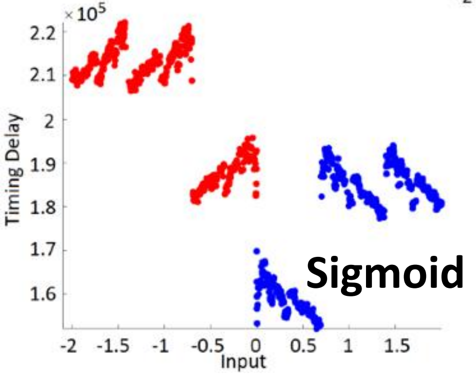


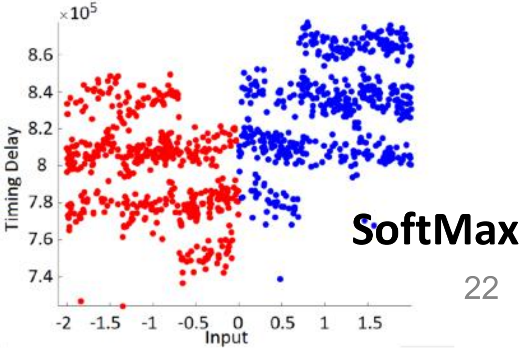


Timing Patterns of Various Activation Function

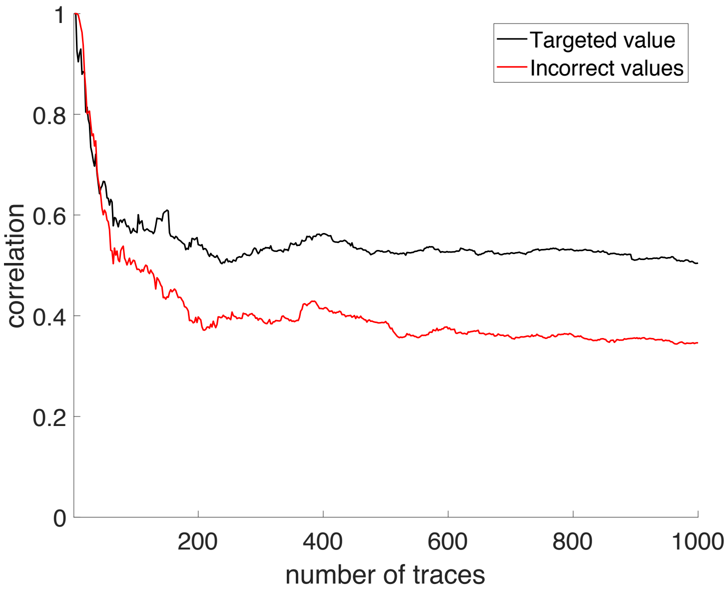




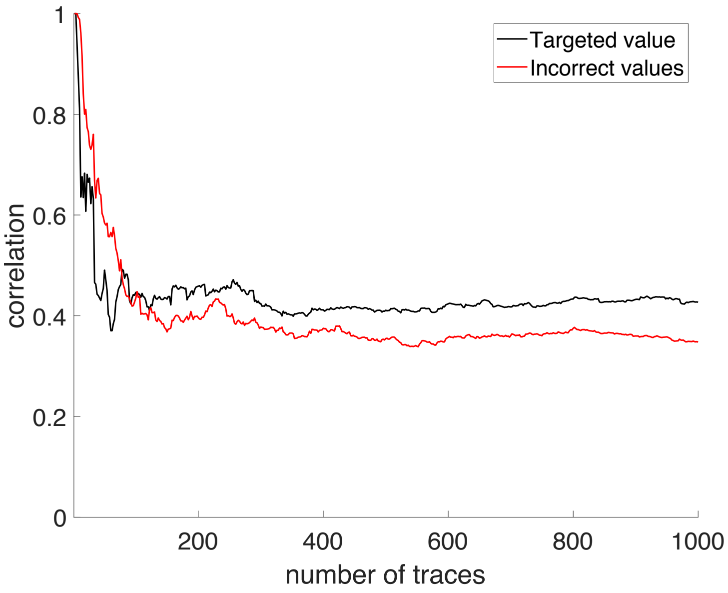




Recovering Weights

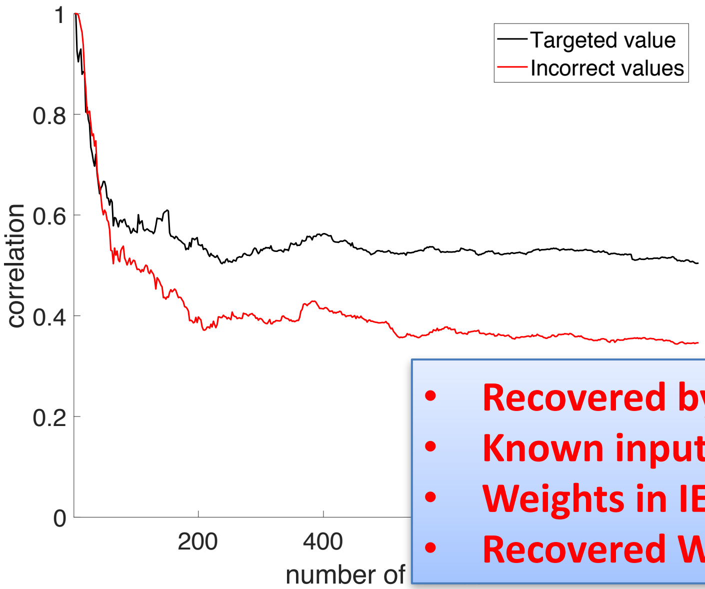


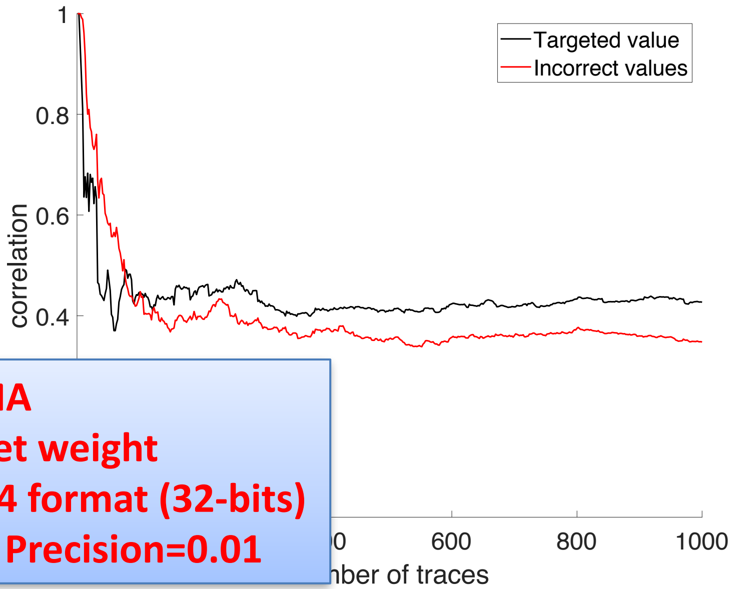
First byte recovery
(sign and 7-bit exponent)



Second byte recovery
(lsb exponent and mantissa)

Recovering Weights



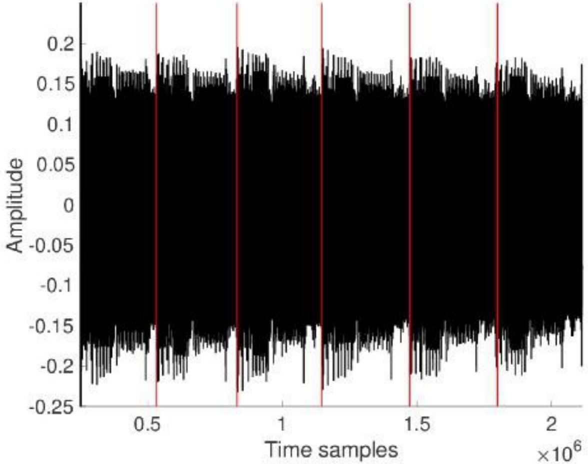


- Recovered by DEMA
- Known input, secret weight
- Weights in IEEE 754 format (32-bits)
- Recovered Weight Precision=0.01

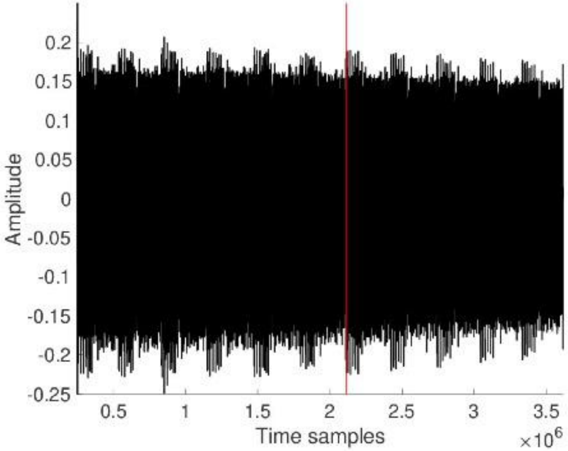
First byte recovery
(sign and 7-bit exponent)

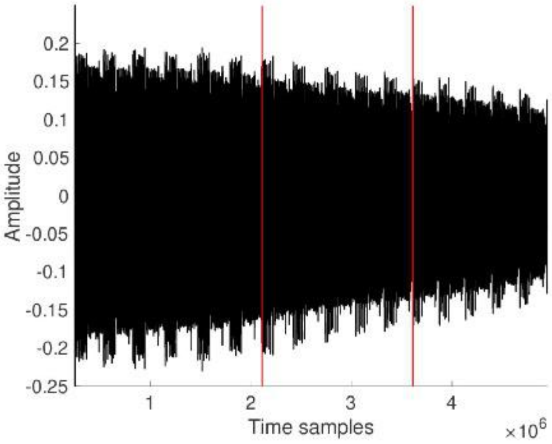
Second byte recovery
(lsb exponent and mantissa)

Recovering Number of Neurons & Layers

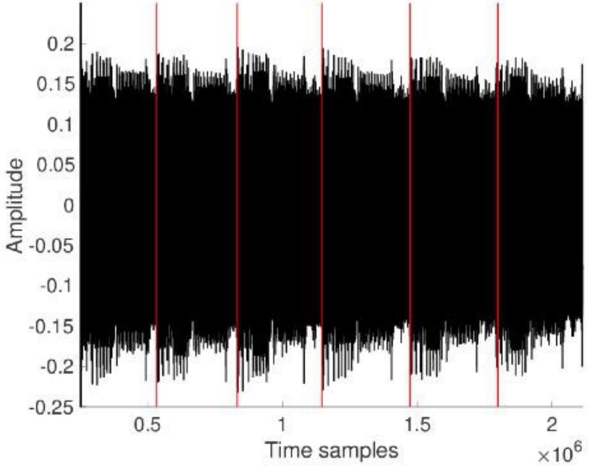


One hidden layer
6 neurons



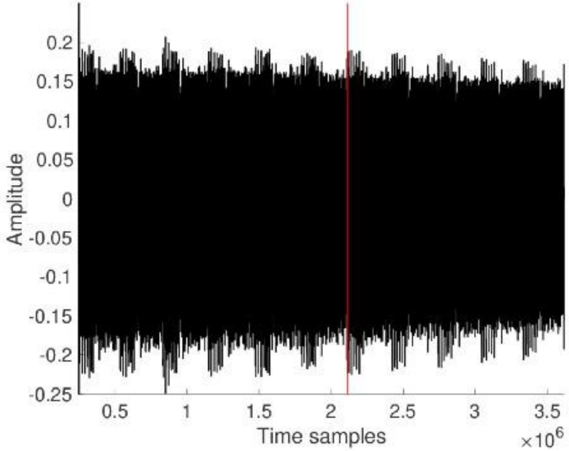


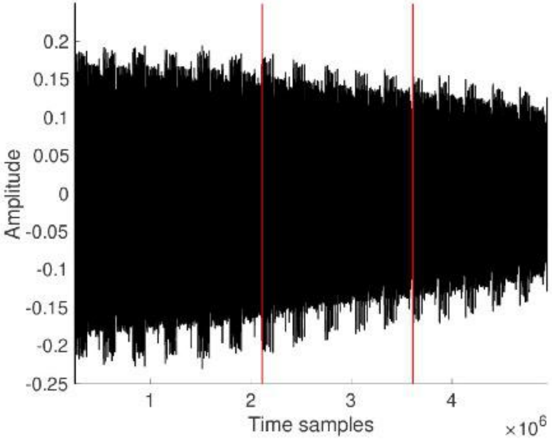
Recovering Number of Neurons & Layers



One hidden layer
6 neurons

Two hidden layer
(6,5) neurons

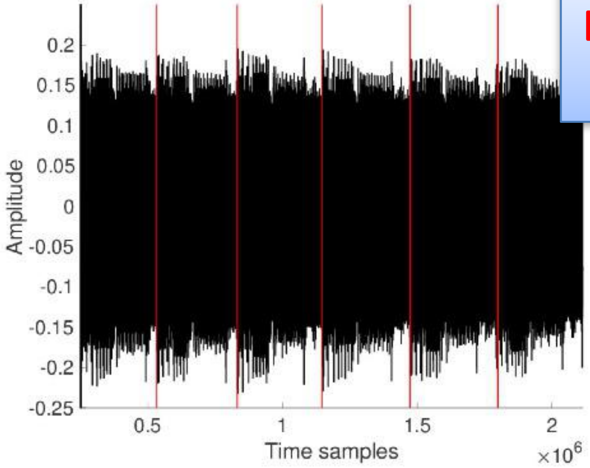




Three hidden layer
(6,5,5) neurons

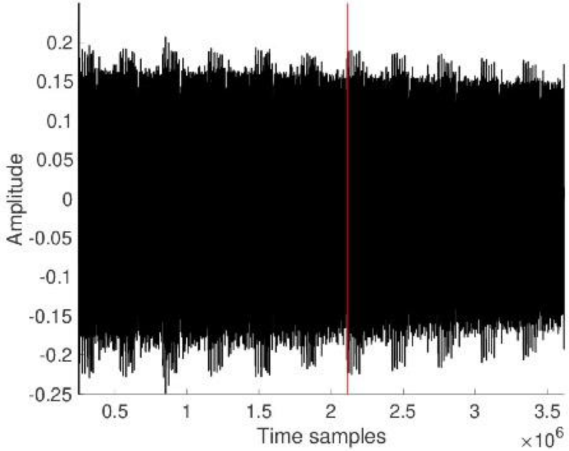
Recovering Number of Neurons & Layers

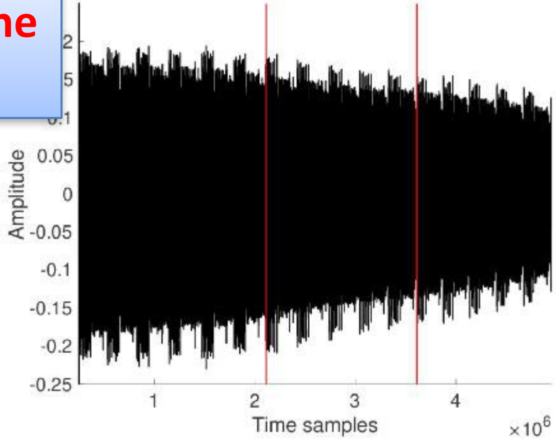
DEMA on weights used to determine layer boundaries



One hidden layer
6 neurons

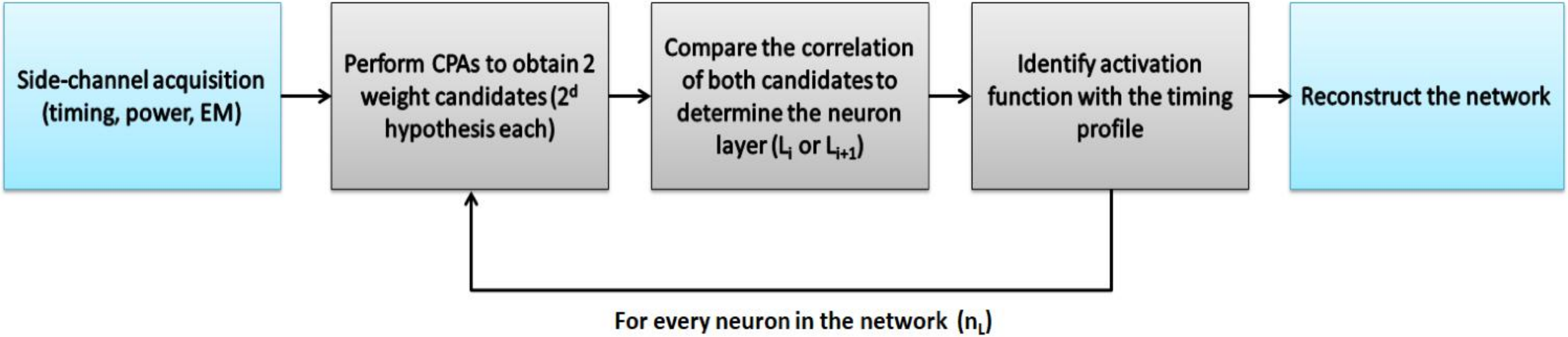
Two hidden layer
(6,5) neurons



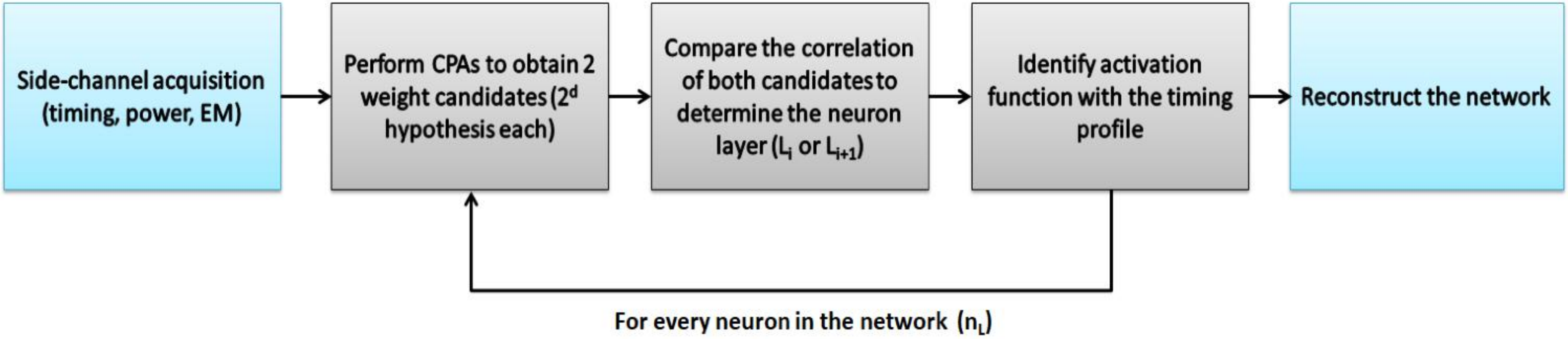


Three hidden layer
(6,5,5) neurons

Full Network Recovery

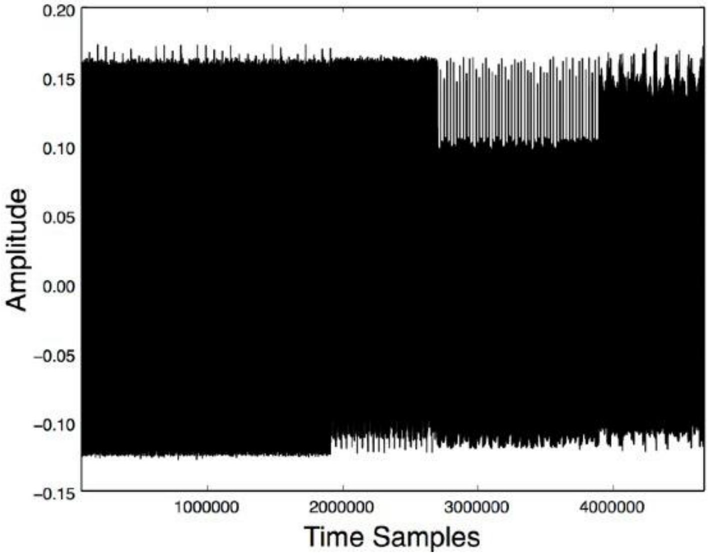


Full Network Recovery

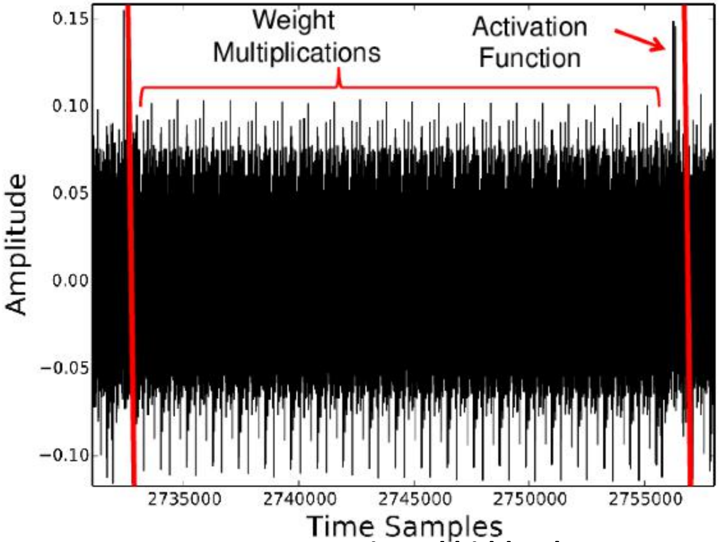


Recovery is performed layer by layer, neuron by neuron.
One neuron at a time, starting from input layer

Results on ARM Cortex-M3

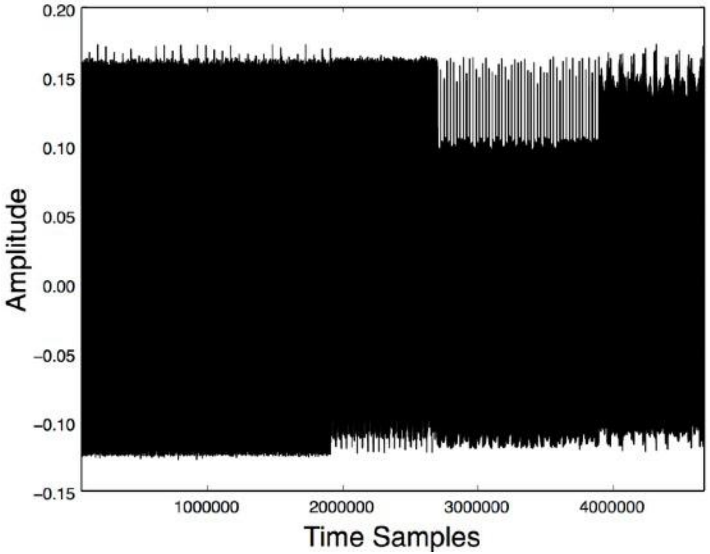


Four hidden layer
(50,30,20,50) neurons

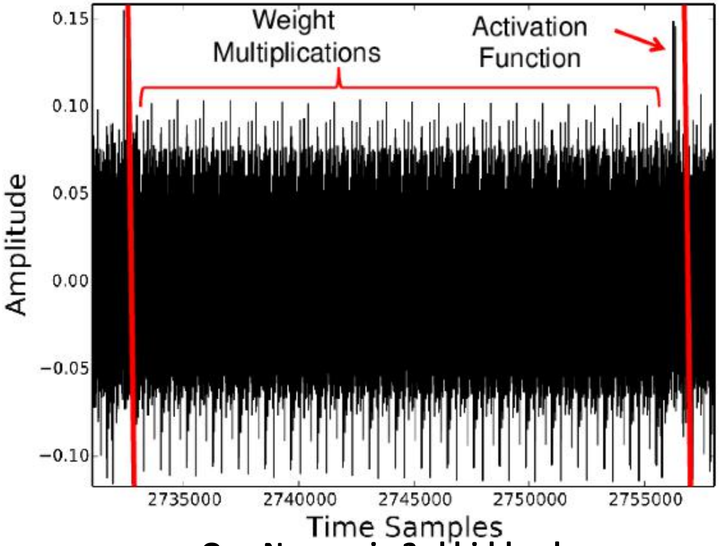


One Neuron in 3rd hidden layer
20 multiplications, 1 ReLU

Results on ARM Cortex-M3



Four hidden layer (50,30,20,50) neurons

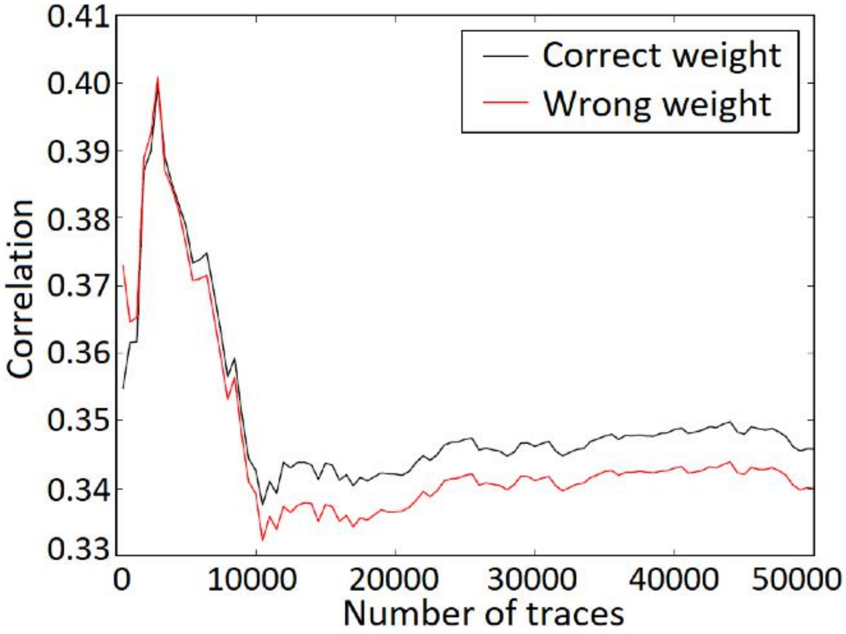


One Neuron in 3rd hidden layer
20 multiplications, 1 ReLU

With MNIST: Accuracy 98.16% (original) vs 98.15% (reverse engineered)
Average weight error: 0.0025.

Extension to CNN on ARM Cortex-M3

- CIFAR-10 dataset.
- Target the multiplication operation from the input with the weight, similar as in previous experiments.
- fixed-point arithmetic (8-bits).
- The original accuracy of the CNN equals 78.47% and the accuracy of the recovered CNN is 78.11%.



Conclusions

- With an appropriate combination of SEMA and DEMA techniques, all sensitive parameters of the network can be recovered.
- A serious threat to commercial NN Ips
- The attack methodology scales linearly with the size of the network.
- **Transfer learning is a key target**
- The proposed attacks are both generic in nature and more powerful than the previous works in this direction.
- Can be adapted for recovery of sensitive training/testing data
- SCA countermeasures (masking/hiding) would help but overhead will be too high for NN. Motivates research for optimised countermeasures.

PART II

Fault Attacks

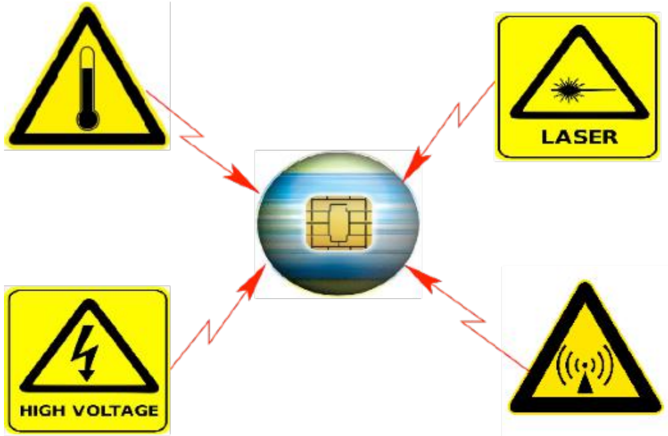
Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined Side-Channel + Faults
6. Conclusions

Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

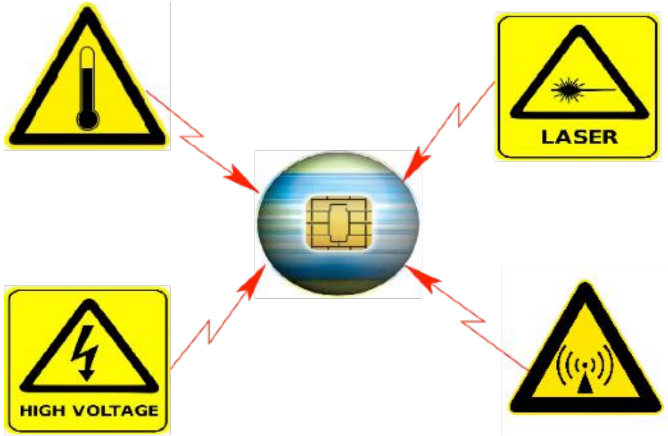
Fault Injection Attacks (FIA)



What is FIA?

- Physical Attacks
- Actively disturbs functioning of the target
- Exploits erroneous behavior

Fault Injection Attacks (FIA)



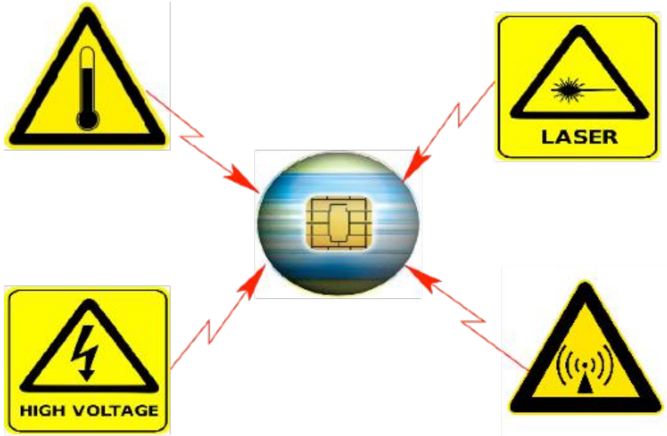
Injection Methods

- Global/Low-Cost/Low-Precision
  - Clock/Voltage glitch, temperature
- Local/High-Cost/High-Precision
  - Laser, Electromagnetic, Ion Beam
- Remote
  - Rowhammer, Plundervolt

What is FIA?

- Physical Attacks
- Actively disturbs functioning of the target
- Exploits erroneous behavior

Fault Injection Attacks (FIA)



What is FIA?

- Physical Attacks
- Actively disturbs functioning of the target
- Exploits erroneous behavior

Injection Methods

- Global/Low-Cost/Low-Precision
  - Clock/Voltage glitch, temperature
- Local/High-Cost/High-Precision
  - Laser, Electromagnetic, Ion Beam
- Remote
  - Rowhammer, Plundervolt

Impacts

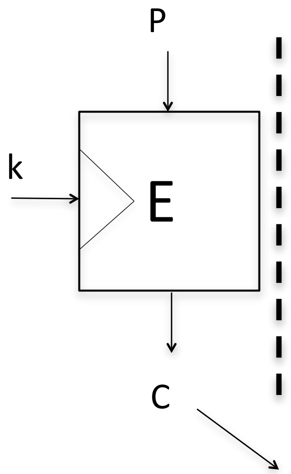
- Duration
  - Transient or Harmonic
- Effects
  - Data or Flow Modification
- Objectives
  - Corrupt computation, bypass security checks

Fault Analysis

- Differential Fault Analysis (DFA)
- Statistical Fault Analysis (SFA)

Fault Analysis

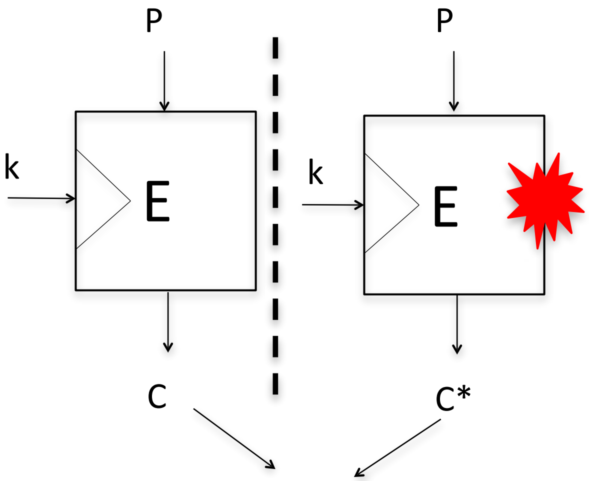
- Differential Fault Analysis (DFA)



- Statistical Fault Analysis (SFA)

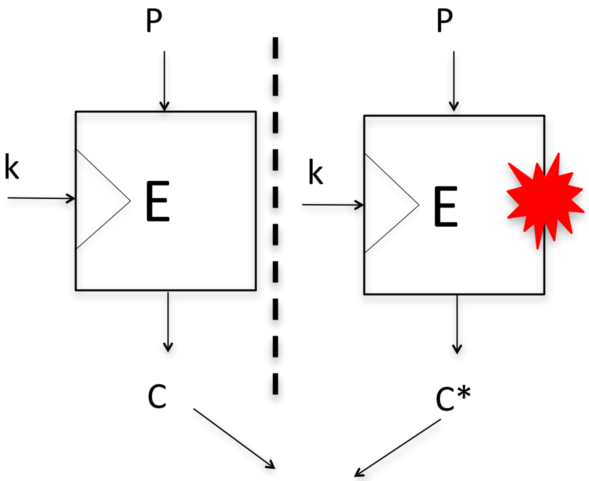
Fault Analysis

- Differential Fault Analysis (DFA)
- Statistical Fault Analysis (SFA)



Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed

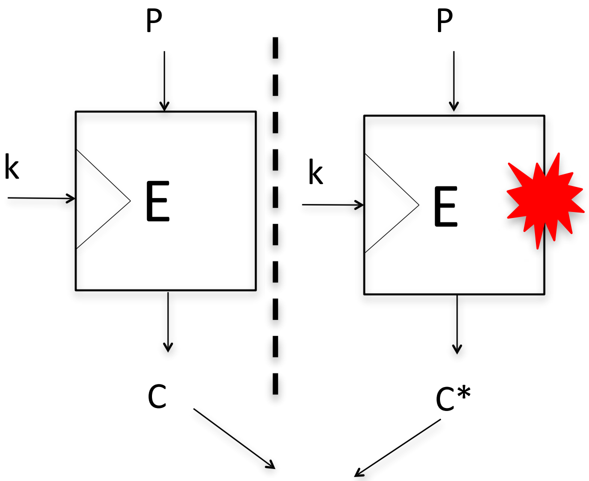


Analysis K=f(C,C*)

- Statistical Fault Analysis (SFA)

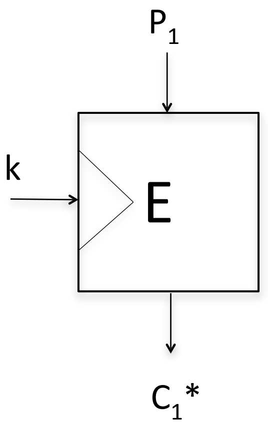
Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed



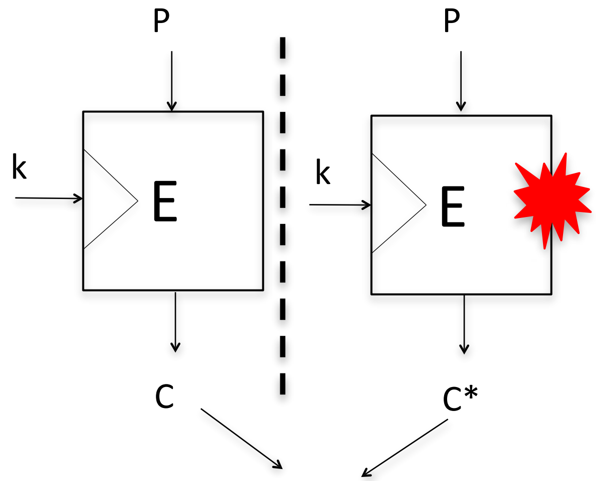
Analysis K=f(C,C*)

- Statistical Fault Analysis (SFA)



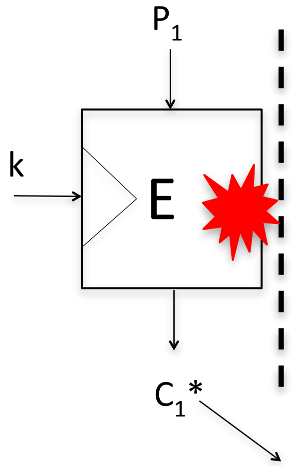
Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed



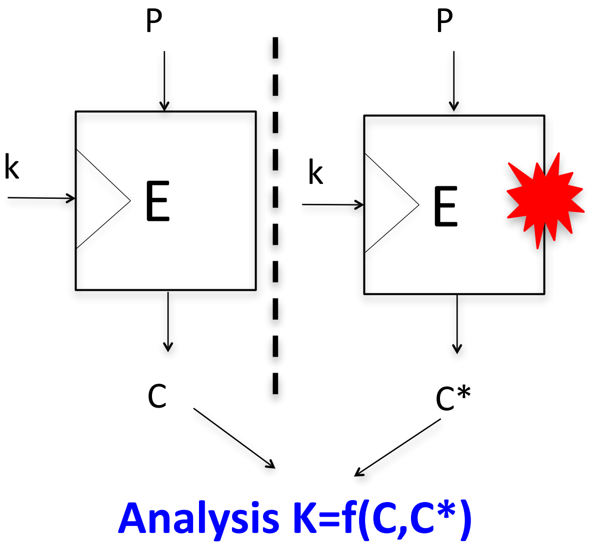
Analysis K=f(C,C*)

- Statistical Fault Analysis (SFA)

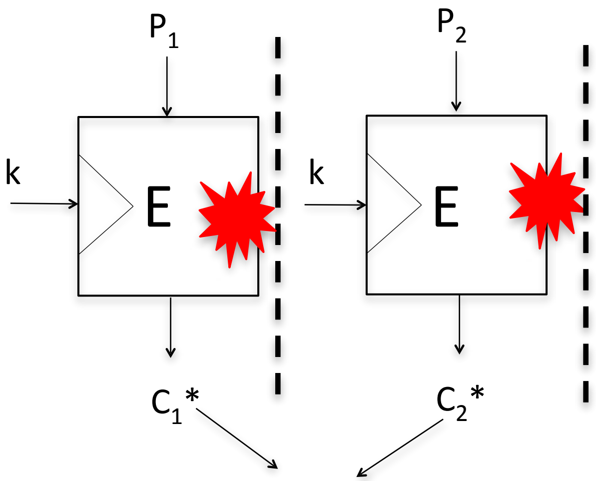


Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed

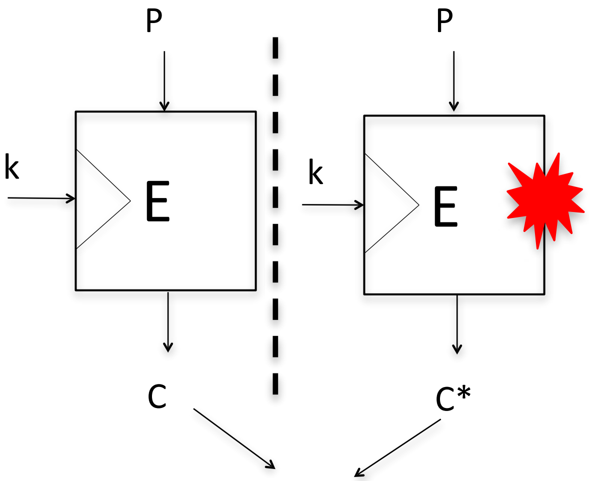


- Statistical Fault Analysis (SFA)



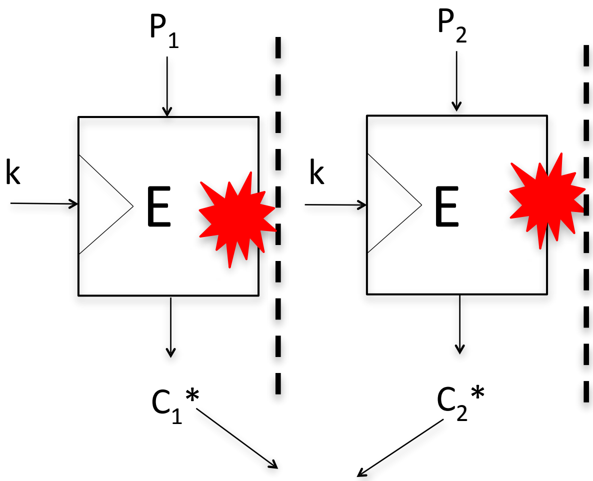
Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed



Analysis K=f(C,C*)

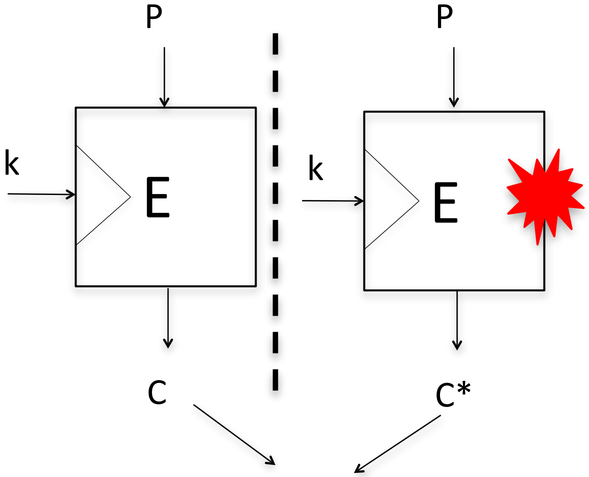
- Statistical Fault Analysis (SFA)



Analysis K=f(C_1*,C_2*, ...)

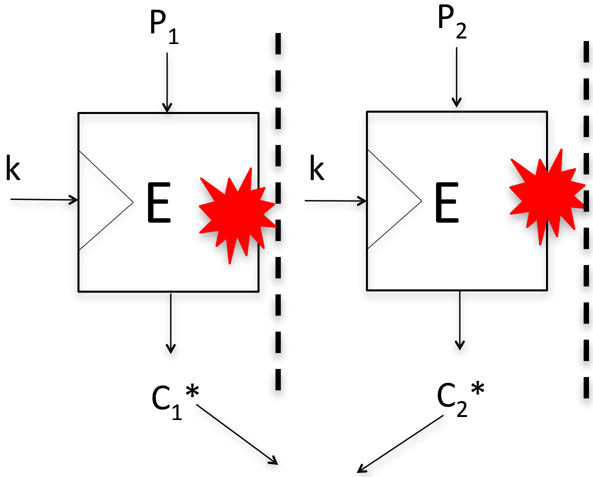
Fault Analysis

- Differential Fault Analysis (DFA)
- Usually few ciphertext pair
- Control over plaintext needed



Analysis K=f(C,C*)

- Statistical Fault Analysis (SFA)
- Need several ciphertext
- Several variants exist



Analysis $K=f(C_1*,C_2*, \ldots)$

Limitations of SoA

- Very tight time synchronization on the round calculation and the injection timing

Limitations of SoA

- Very tight time synchronization on the round calculation and the injection timing
- Very complicated analysis due to the random value and the fault propagation

Limitations of SoA

- Very tight time synchronization on the round calculation and the injection timing
- Very complicated analysis due to the random value and the fault propagation
- May not work if there are countermeasures against fault attacks

Spoiler

Spoiler

Optimization often become security threats!!!

Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

Revisiting Fault types

Revisiting Fault types

- Transient: Affect one encryption
- Permanent: Always present

- **Persistent[1]:** Hybrid model between transient and permanent. Persist over several encryptions but disappears on reboot. Typically targets stored constants (ex. Sbox in memory)

Revisiting Fault types

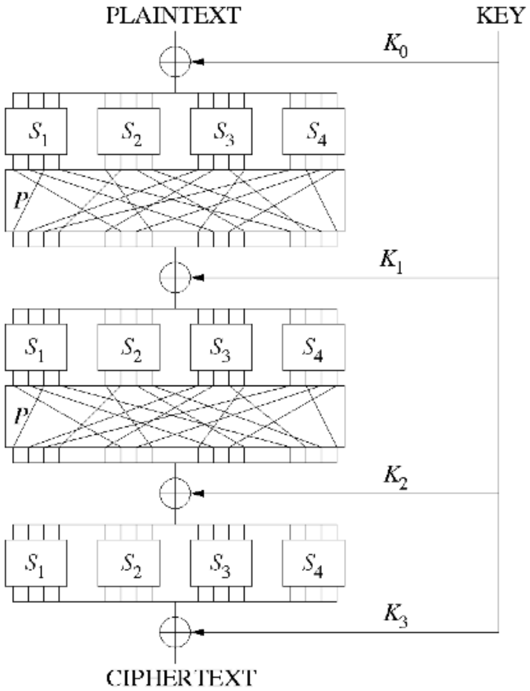
- Transient: Affect one encryption
- Permanent: Always present

- **Persistent[1]:** Hybrid model between transient and permanent. Persist over several encryptions but disappears on reboot. Typically targets stored constants (ex. Sbox in memory)

[1]Persistent Fault Analysis. CHES 2018

SPN Block Ciphers: Recall

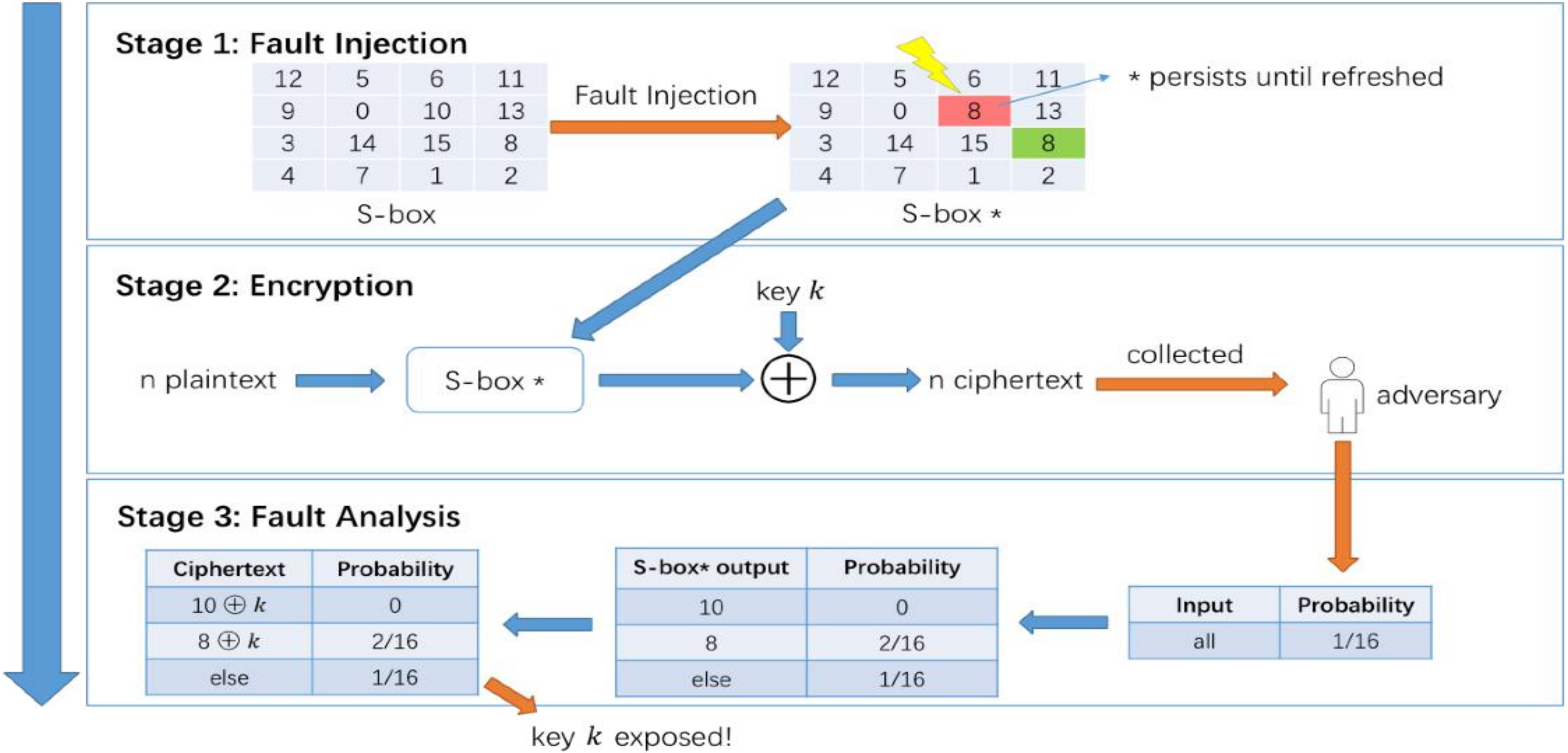
- Blocks of plaintext encrypted
- Encryption and Decryption done using same pre-shared Key
- Building blocks
  - Substitution Layer (Sboxes)
  - Permutation Layer (Linear)
  - Key Addition
- Typically Sboxes are the **only non linear component and hard to implement**
- Common Solution: Implemented as precomputed look-up tables



Adversary Model

- Block cipher with serial implementation
- Common Sbox as look-up table
- Persistent fault injected in one Sbox element
- Victim encrypts n plaintext with faulty Sbox
- Adversary can observe the n ciphertext
- No control on plaintext, except varying plaintext

Persistent Fault Analysis: Main Idea



PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:
  - t_{min}: find the missing value in Sbox table (x). Then $k = t_{min} \oplus x$;

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:
  - t_{min}: find the missing value in Sbox table (x). Then $k = t_{min} \oplus x$;
  - $t \neq t_{min}$: find values t where $t \neq t_{min}$ and eliminate candidates for k;

PFA: Modus Operandi

- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:
  - t_{min}: find the missing value in Sbox table (x). Then $k = t_{min} \oplus x$;
  - $t \neq t_{min}$: find values t where $t \neq t_{min}$ and eliminate candidates for k;
  - t_{max}: find the value with max probability (x'). Then $k = t_{max} \oplus x^*$

PFA: Modus Operandi

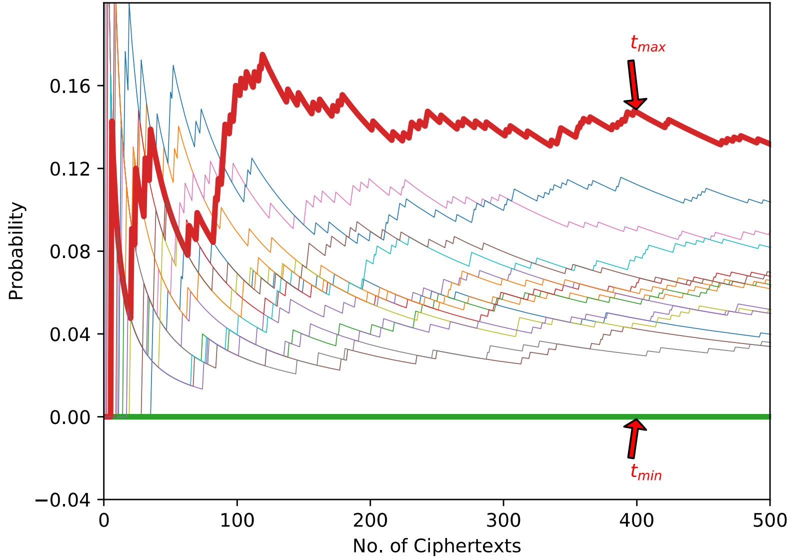
- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:
  - t_{min}: find the missing value in Sbox table (x). Then $k = t_{min} \oplus x$;
  - $t \neq t_{min}$: find values t where $t \neq t_{min}$ and eliminate candidates for k;
  - t_{max}: find the value with max probability (x'). Then $k = t_{max} \oplus x^*$
- No. of ciphertext n can be determined by coupon collector's problem

PFA: Modus Operandi

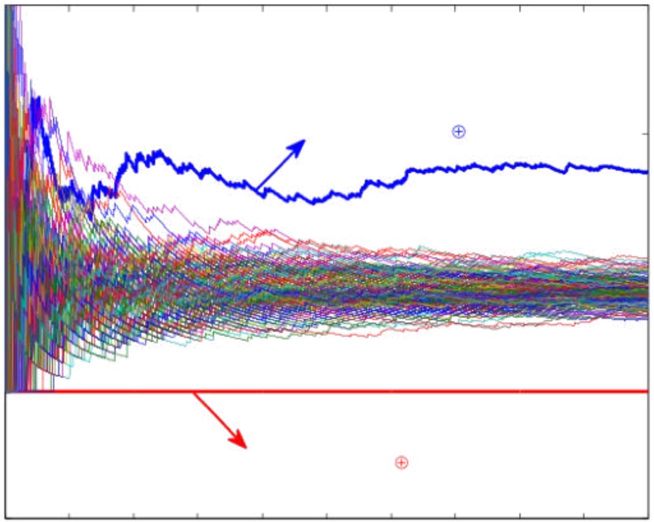
- Statistical analysis on last round with ciphertext only
- Fault changes one element $x \rightarrow x^*$ in Sbox (lets say 4X4 Sbox)
- Expectation $E(x)= 0$, $E(x^*)=2/16$, $E(y \neq (x,x^*))=1/16$
- Three analysis strategies:
  - t_{min}: find the missing value in Sbox table (x). Then $k = t_{min} \oplus x$;
  - $t \neq t_{min}$: find values t where $t \neq t_{min}$ and eliminate candidates for k;
  - t_{max}: find the value with max probability (x'). Then $k = t_{max} \oplus x^*$
- No. of ciphertext n can be determined by coupon collector's problem
- x, x* can be brute-forced if not known

PFA on PRESENT and AES

PRESENT: n ≥50

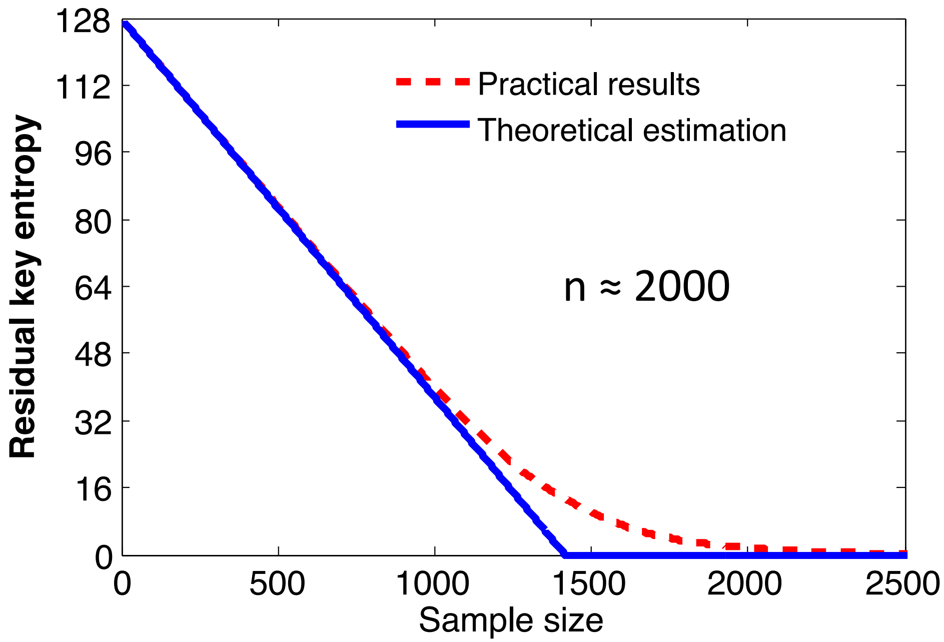


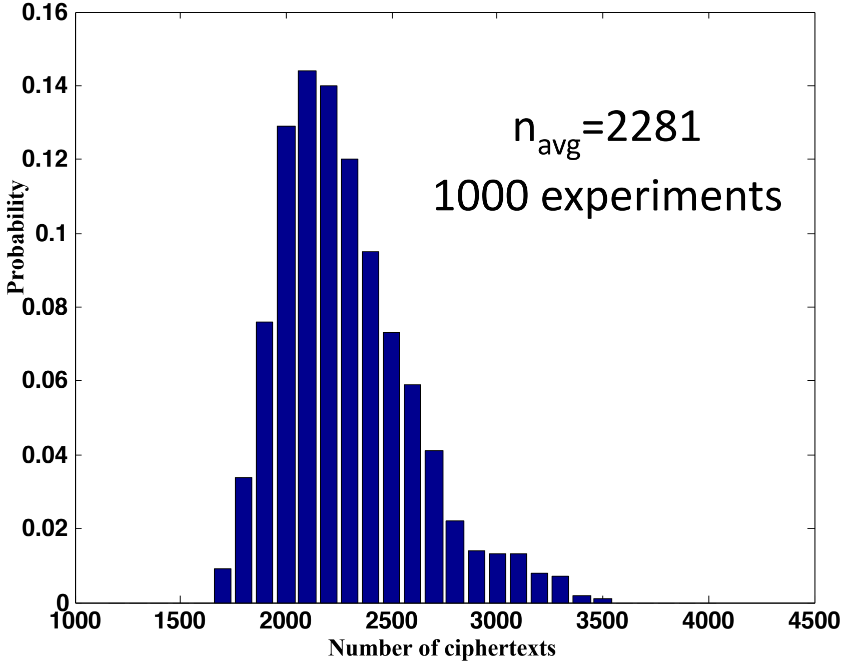
AES: n ≥ 1560



n= Minimum no of ciphertext needed by coupon collector's problem

Practical PFA on AES





Comparison vs Other Fault Attacks

(1) The attack is not differential in nature and thus the control over the plaintext is not required.

(2) The adversary does not necessarily need live synchronization

(3) The fault model remains relaxed (no biased faults needed)

(4) PFA can also be applied in multiple fault setting

(5) PFA can bypass some redundancy based countermeasures

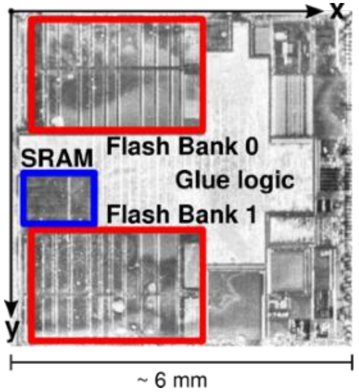
(6) An adversary can always inject the persistent fault before the victim is switched to the sensitive mode

(1) It needs higher number of ciphertexts as compared to DFA

(2) Persistent faults can be detected by some built-in health test mechanism or fault counters.

T-table corruption

- EM fault on ARM Cortex-M3 with 100% repeatability
- Public AES implementation from Schwabe and Stoffelen
- Single T-table,
- 4 columns of 32 bits in the data buffer

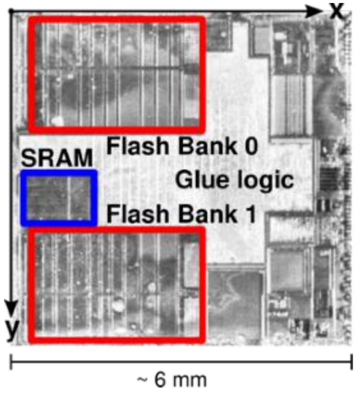


T-table corruption

```
/* relocate */
pSrc = &_etext;
pDest = &_srelocate;
for(; pDest < &_erelocate ;){
    *pDest++ = *pSrc++;
}
```

128-bit Flash memory access

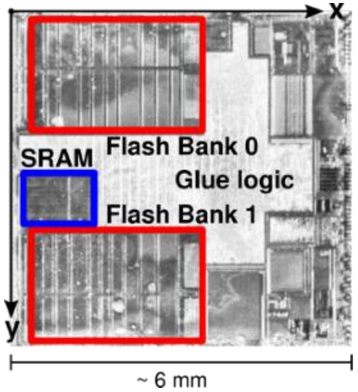
- EM fault on ARM Cortex-M3 with 100% repeatability
- Public AES implementation from Schwabe and Stoffelen
- Single T-table,
- 4 columns of 32 bits in the data buffer



T-table corruption

```
/* relocate */
pSrc = &_etext;
pDest = &_srelocate;
for(; pDest < &_erelocate ;){
    *pDest++ = *pSrc++;
}
```

128-bit Flash memory access



- EM fault on ARM Cortex-M3 with 100% repeatability
- Public AES implementation from Schwabe and Stoffelen
- Single T-table,
- 4 columns of 32 bits in the data buffer

$$T[v] = \begin{bmatrix} S[v] \circ 01 \\ S[v] \circ 03 \\ S[v] \circ 02 \\ S[v] \circ 01 \end{bmatrix} \longrightarrow T[v^*] = \begin{bmatrix} a \\ a \circ 03 \\ a \circ 02 \\ a \end{bmatrix} \quad \boxed{\Leftrightarrow a = 0}$$

Fault condition

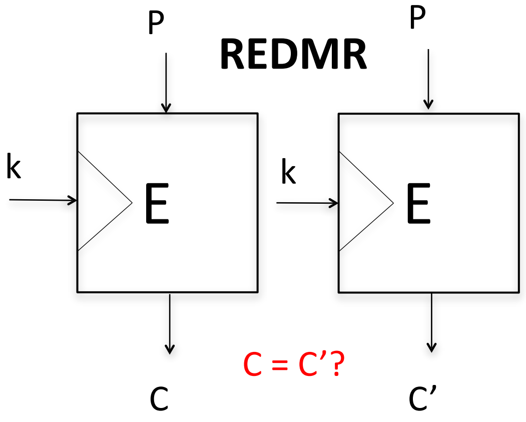
Fault on 4 columns = residual key entropy of 32 bits (practical to brute-force)

Dual Modular Redundancy (DMR) Countermeasure

- Compute twice and compare (REDMR)
- Compute forward-inverse and compare (IDDMR)
- If ≠
  - NCO: No Ciphertext output
  - ZVO: Zero Value output
  - RCO: Random Ciphertext output
- Provably secure against single fault
- Adversary can either target the encryption or comparison but not both
- REDMR broken by design if same S-box is used
- Lets target IDDMR, more difficult of the two

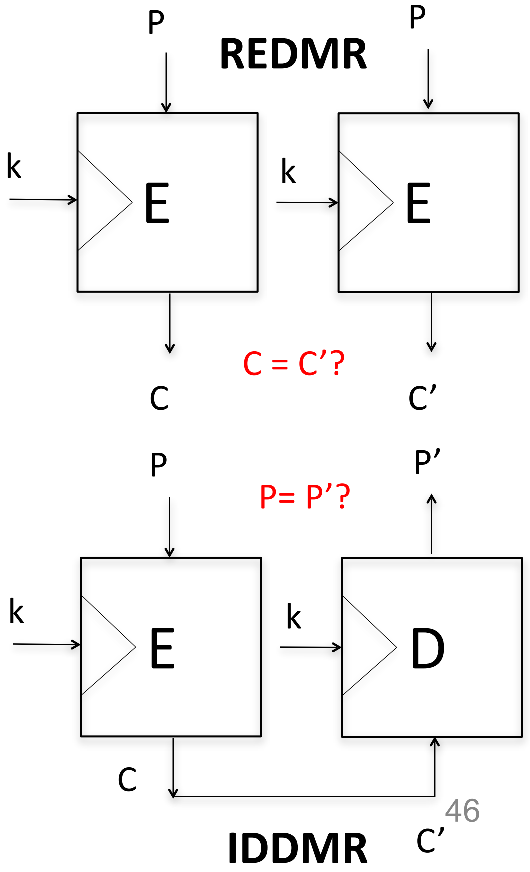
Dual Modular Redundancy (DMR) Countermeasure

- Compute twice and compare (REDMR)
- Compute forward-inverse and compare (IDDMR)
- If ≠
  - NCO: No Ciphertext output
  - ZVO: Zero Value output
  - RCO: Random Ciphertext output
- Provably secure against single fault
- Adversary can either target the encryption or comparison but not both
- REDMR broken by design if same S-box is used
- Lets target IDDMR, more difficult of the two



Dual Modular Redundancy (DMR) Countermeasure

- Compute twice and compare (REDMR)
- Compute forward-inverse and compare (IDDMR)
- If ≠
  - NCO: No Ciphertext output
  - ZVO: Zero Value output
  - RCO: Random Ciphertext output
- Provably secure against single fault
- Adversary can either target the encryption or comparison but not both
- REDMR broken by design if same S-box is used
- Lets target IDDMR, more difficult of the two



Attacking IDDMR with NCO/ZVO

- Faulty outputs are supressed
- Some output will be not affected by fault
- Probability p of correct output is f(x,k)
- *p* for AES

$$p = (1 - \frac{1}{256})^{160} = 0.5346$$

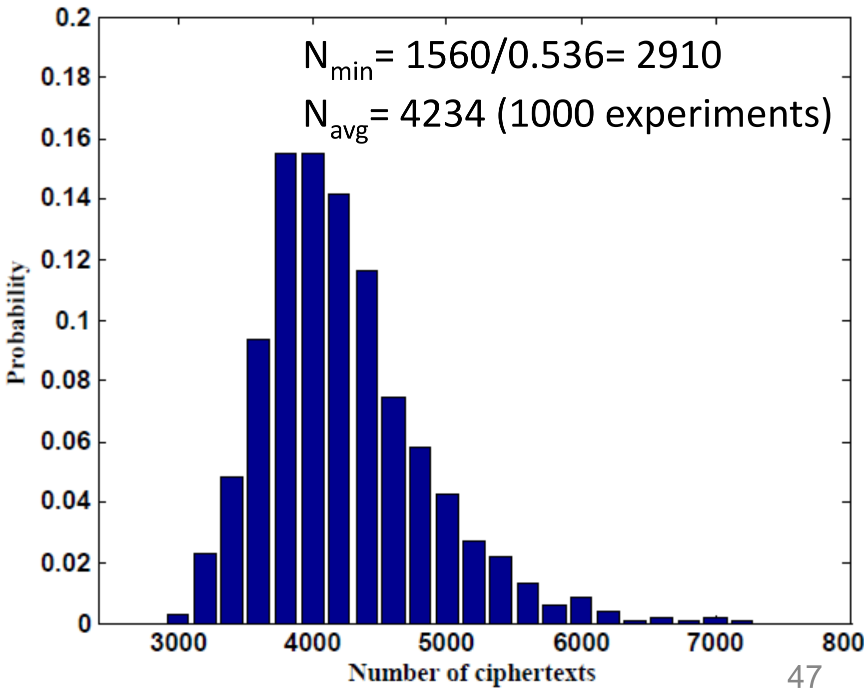
- Adversary roughly needs n/p ciphertext

Attacking IDDMR with NCO/ZVO

- Faulty outputs are supressed
- Some output will be not affected by fault
- Probability p of correct output is f(x,k)
- *p* for AES

$$p = (1 - \frac{1}{256})^{160} = 0.5346$$

- Adversary roughly needs n/ p ciphertext



Attacking IDDMR with RCO

- Faulty output is replaced by uniformly random
- Slight difference in distribution of random output and correct ciphertext
- The bias can be detected with more ciphertext (n)

$$Pr(y = x) = 0 \times p + \frac{1}{256} \times (1 - p) = \frac{0.4654}{256}$$

$$Pr(y = x^*) = \frac{2}{256} \times p + \frac{1}{256} \times (1 - p) = \frac{1.5346}{256}$$

$$Pr(y \neq x \wedge y \neq x^*) = \frac{1}{256} \times p + \frac{1}{256} \times (1 - p) = \frac{1}{256}$$

- Roughly n≈10000 resulted in attack success

Attacking IDDMR with RCO

- Faulty output is replaced by uniformly random
- Slight difference in distribution of random output and correct ciphertext
- The bias can be detected with more ciphertext (n)

$$Pr(y = x) = 0 \times p + \frac{1}{256} \times (1-p) = \frac{0.4654}{256}$$

$$Pr(y = x^*) = \frac{2}{256} \times p + \frac{1}{256} \times (1-p) = \frac{1.5346}{256}$$

$$Pr(y \neq x \wedge y \neq x^*) = \frac{1}{256} \times p + \frac{1}{256} \times (1-p) = \frac{1}{256}$$

- Roughly n≈10000 resulted in attack success

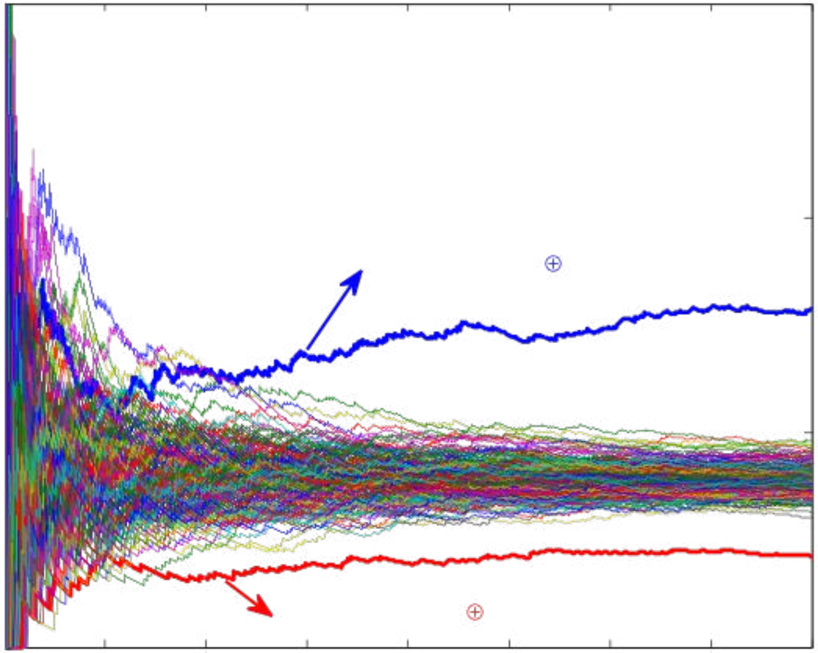


Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

Masking Countermeasure

- Masking is an algorithmic side-channel countermeasure
- Based on Shamir's secret sharing
- Boolean Masking:
  - Secret x split into tuple (x_m,m)
  - $x_m = x \oplus m$
  - m is randomly chosen on each execution
  - For higher order masking m is split in further shares
  - At masking order d: $m = m_1 \oplus m_2 \oplus m_3 \ldots \oplus m_d$
- Removes dependency between x and side-channel leakage

Masking Countermeasure

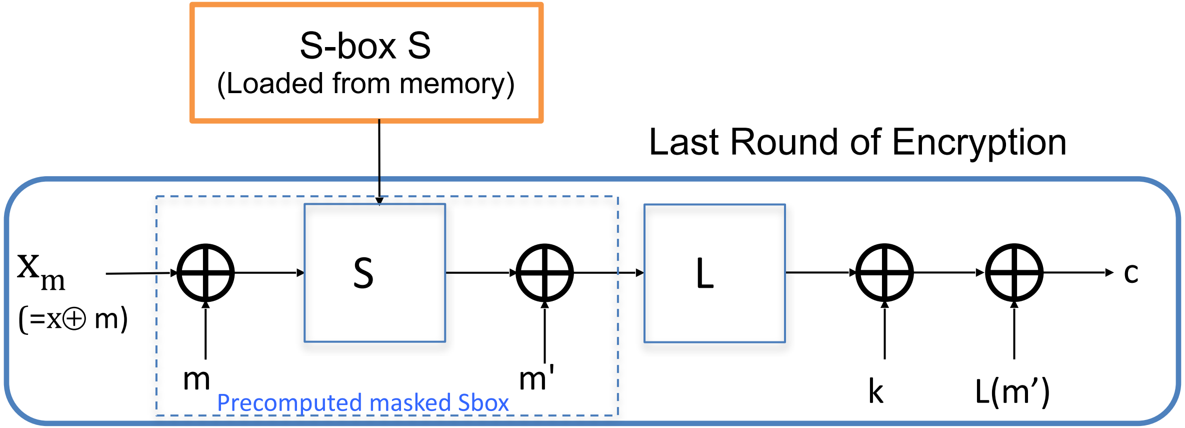
- Masking is an algorithmic side-channel countermeasure
- Based on Shamir's secret sharing
- Boolean Masking:
  - Secret x split into tuple (x_m, m)
  - $x_m = x \oplus m$
  - m is randomly chosen on each execution
  - For higher order masking m is split in further shares
  - At masking order d: $m = m_1 \oplus m_2 \oplus m_3 \dots \oplus m_d$
- Removes dependency between x and side-channel leakage

[2] One Fault is All it Needs: Breaking Higher-Order Masking with Persistent Fault Analysis DATE 2019

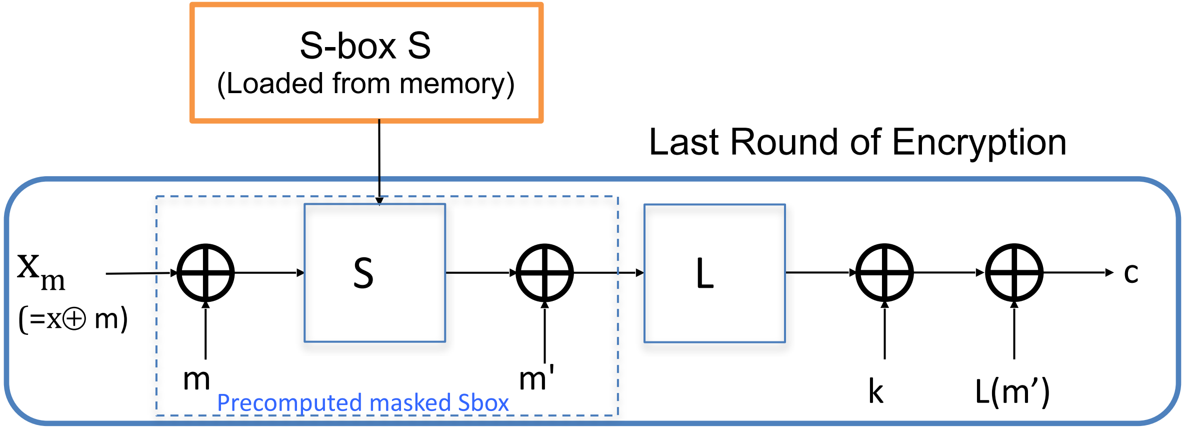
Masking vs PFA

- Theoretically masking does not resist fault attacks
- Several previous attack were presented on masking
- They work in restrictive setting (advanced fault model, high no. of faults etc.)
- **Only One Fault** to break 4 various **public implementation of masking**
- Target Implementations:
  - Byte-wise Masking [SES, Virginia Tech]
  - Coron's Table Masking [EuroCrypt 2014]
  - Rivian and Prouff Masking [CHES 2010]
  - Software Threshold [COSADE 2018]

PFA on Masking: Generic Attack

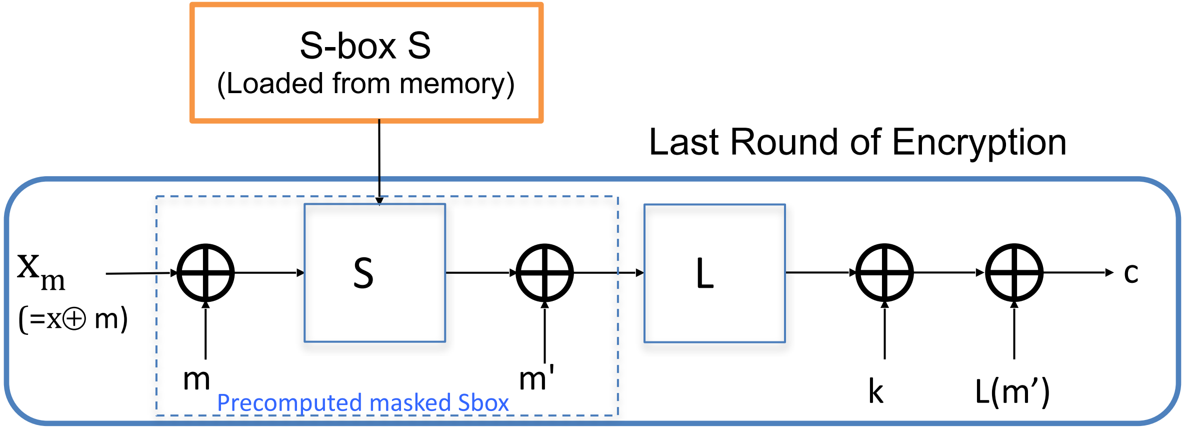


PFA on Masking: Generic Attack



$$
\begin{aligned}
c &= L(S(x_m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x \oplus m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x)) \oplus k \oplus L(m') \oplus L(m') \\
&= L(S(x)) \oplus k
\end{aligned}
$$

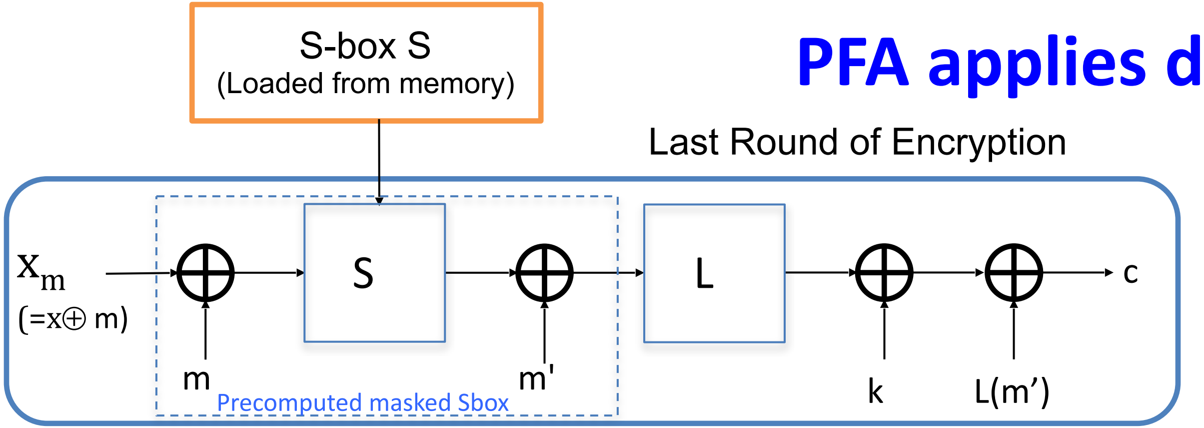
PFA on Masking: Generic Attack



$$
\begin{aligned}
c &= L(S(x_m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x \oplus m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x)) \oplus k \oplus L(m') \oplus L(m') \\
&= L(S(x)) \oplus k
\end{aligned}
$$

Masking has no effect on the distribution of the final ciphertext

PFA on Masking: Generic Attack

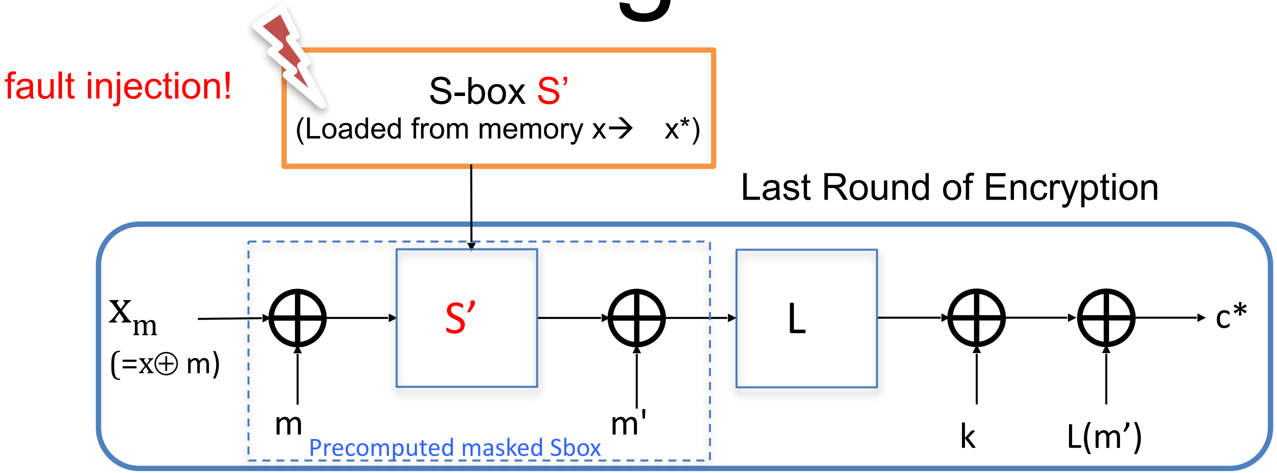


PFA applies directly!!!

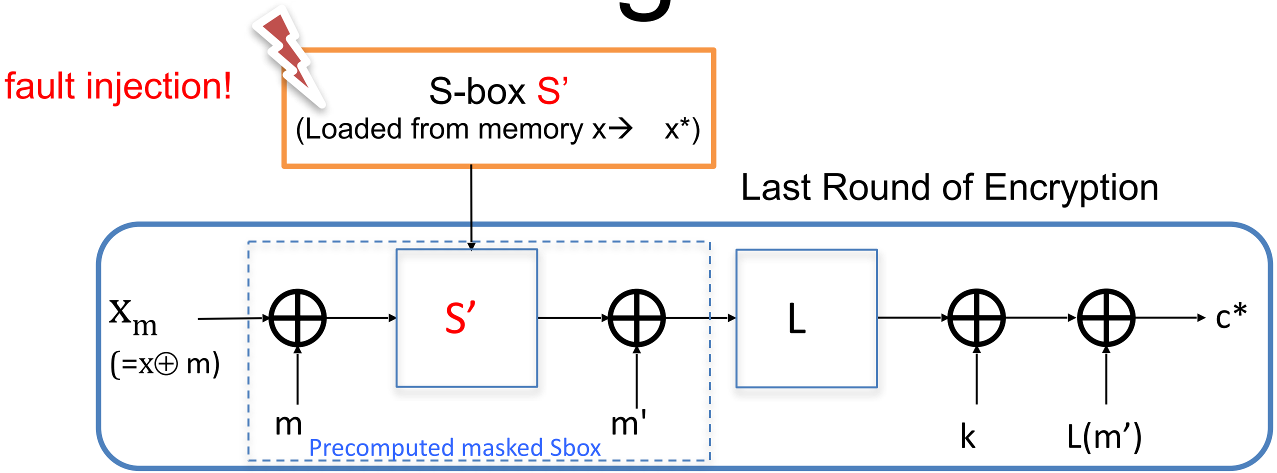
$$
\begin{aligned}
c &= L(S(x_m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x \oplus m \oplus m) \oplus m') \oplus k) \oplus L(m') \\
&= L(S(x)) \oplus k \oplus L(m') \oplus L(m') \\
&= L(S(x)) \oplus k
\end{aligned}
$$

Masking has no effect on the distribution of the final ciphertext

PFA on Masking: Generic Attack

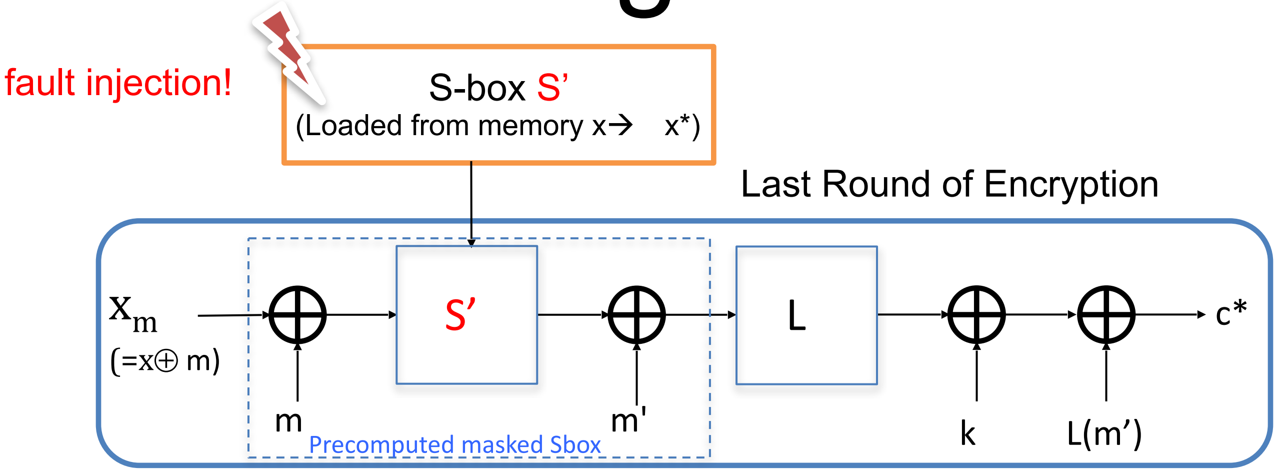


PFA on Masking: Generic Attack



$$c^* = L(S'(x_m \oplus m) \oplus m') \oplus k) \oplus L(m')$$
$$= L(S'(x \oplus m \oplus m) \oplus m') \oplus k) \oplus L(m')$$
$$= L(S'(x)) \oplus k \oplus L(m') \oplus L(m')$$
$$= L(S'(x)) \oplus k$$

PFA on Masking: Generic Attack



$c^* = L(S'(x_m \oplus m) \oplus m') \oplus k) \oplus L(m')$

$= L(S'(x \oplus m \oplus m) \oplus m') \oplus k) \oplus L(m')$

$= L(S'(x)) \oplus k \oplus L(m') \oplus L(m')$

$= L(S'(x)) \oplus k$

Value $c^*=L(S'(x) \oplus k)$ will be missing

Value $c^*=L(S'(x^*) \oplus k)$ will be doubled

➔ Allows key recovery with PFA

m,m' do not appear

Also masking order does not matter

Attack Results on Public Code

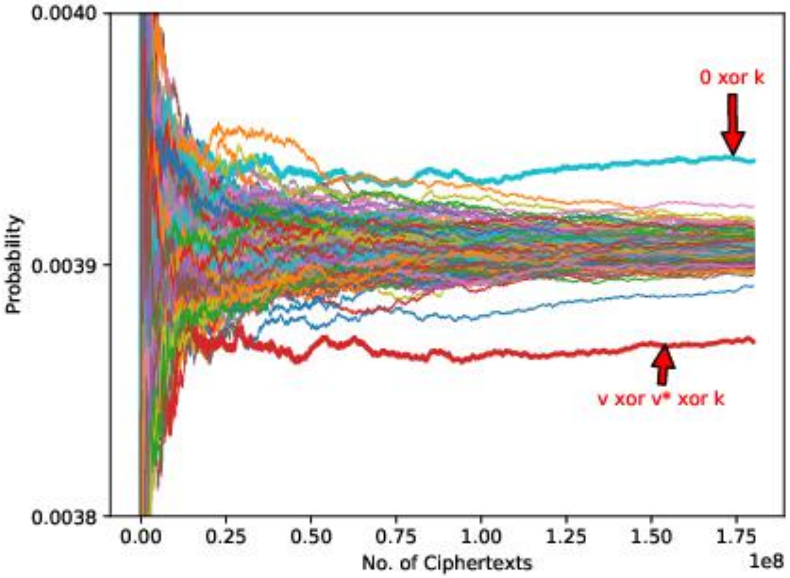
Design	Fault Target	No. of Ciphertext (Masking Order)
Bytewise Masking (Virginiatech)	Sbox Recomputation	1560 (any)
Coron’s higher Order Masking (Eurocrypt 2014)	Sbox Recomputation	1560 (any)
Rivian & Prouff Masking (CHES 2010)	Affine transformation	2,500,000 (1) [α 2^{14d}]
Software Threshold (COSADE 2018)	Decomposition A’’’	400,000 (1)

Attack Results on Public Code

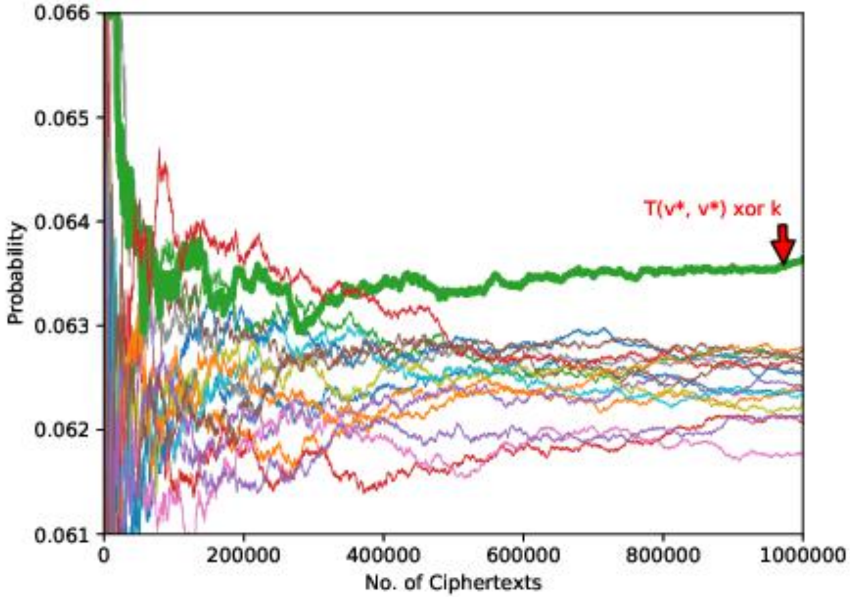
Design	Fault Target	No. of Ciphertext (Masking Order)
Bytewise Masking (Virginiatech)	Sbox Recomputation	1560 (any)
Coron's higher Order Masking (Eurocrypt 2014)	Sbox Recomputation	1560 (any)
Rivian & Prouff Masking (CHES 2010)	Affine transformation	2,500,000 (1) [α 2^{14d}]
Software Threshold (COSADE 2018)	Decomposition A'''	400,000 (1)

ONLY ONE FAULT

Attack Results on Public Code



Rivian & Prouff Masking



Software Threshold

Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

Post Quantum Cryptography (PQC)

- Quantum computers are coming
- A serious threat to current public key cryptography (RSA, ECC)
- Ongoing initiative to standardize Post Quantum Cryptography (PQC)
- Based on hard problems not threatened by quantum computing
- Key candidates:
  - Lattice-based
  - Code-based
  - Multivariate
  - Hash-based
  - Super singular isogeny
  - …

Response to NIST PQC Call

Table: **ROUND 1**

Type	Signatures	KEM/Encryption	Overall
Lattice-based	**5**	**23**	**28**
Code-based	3	17	20
Multivariate	8	2	10
Hash-based	3	0	3
Isogeny-based	0	1	1
Others	2	5	7
Total	21	48	69

Response to NIST PQC Call

Table: **ROUND 2**

Type	Signatures	KEM/Encryption	Overall
Lattice-based	**3**	**9**	**12**
Code-based	0	7	7
Multivariate	4	0	4
Hash-based	2	-	2
Isogeny-based	0	1	1
Others	0	0	0
Total	9	17	26

Learning With Error (LWE) Problem

- $\mathbf{T} = (\mathbf{A} * \mathbf{S} + \mathbf{E}) \in Z_q$
  - Secret $\mathbf{S} \in Z_q^n$
  - $\mathbf{A} \in Z_q^n$ is public
  - Error $\mathbf{E}$ derived from Gaussian distribution
- The hard problem is to solve for $\mathbf{S}$ given several pairs $(\mathbf{A}, \mathbf{T})$
- Error component $\mathbf{E}$ is essential to hardness guarantees

NewHope

- ***NewHope*** is a suite of KEM (NewHope-CPA/CCA-KEM)
- Based on *RLWE* problem (LPR Encryption Scheme)
- ***Sample*** *generates randomness with SHAKE256 (SHA-3)*
- ***Sample*** *<u>32-byte seed and a nonce</u> as input to generate* **`(S,E)`**
- ***nonce*** can be integer ideally in range [0,255]
- In NIST submission, designers use **nonce=(0,1)**

Vulnearbility in ***NewHope***

- The fault attack targets nonce optimization
- Targets generation of **`(S,E)`**
  - `S = Sample(noiseseed, 0)`
  - `E = Sample(noiseseed, 1)`

Vulnearbility in ***NewHope***

- The fault attack targets nonce optimization
- Targets generation of **`(S,E)`**
  - `S = Sample(noiseseed, 0)`
  - `E = Sample(noiseseed, 1 → 0) = S`

Vulnearbility in ***NewHope***

- Assume a Ring-LWE instance

$$\mathbf{T} = (\mathbf{A} * \mathbf{S} + \mathbf{E}) \in R_q$$

- Inject fault such that **E=S**
- Ring-LWE instance is faulted to:

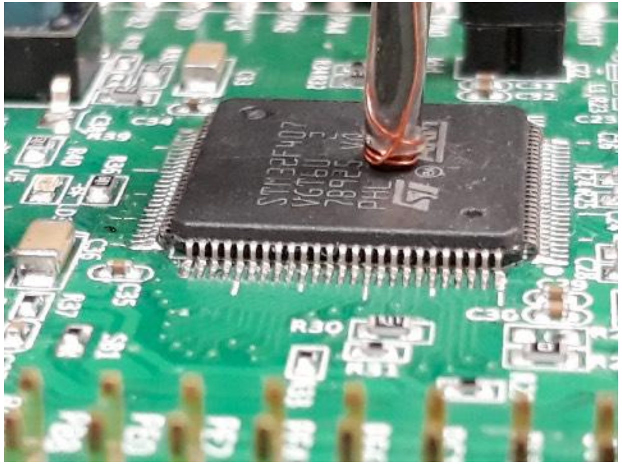
$$\mathbf{T} = (\mathbf{A} * \mathbf{S} + \mathbf{S}) \in R_q$$

- Modular linear system of equations with n equations and n unknowns
- Solved using Gaussian elimination

Impact of Fault Attack

- Leads to **secret key recovery** followed by **message recovery**
- Applies to all variants of LWE (general LWE, Ring-LWE, Module-LWE)
- Same vulnerability in ***Kyber, FRODO, Dilithium.***

Experiments with EMFI



The Fault injection Setup on ARM Cortex-M4

- Attacks target public ***pqm4 library***
- Fault repeatability is 100% at (few) identified locations

Fault Complexity[1]

Attack Objective	Fault Complexity			
	NEWHOPE		FRODO	
	NEWHOPE512	NEWHOPE1024	Frodo-640	Frodo-976
Key Recovery	1	1	1	1
Message Recovery	1	1	1	1

	KYBER			DILITHIUM			
	KYBER512	KYBER768	KYBER1024	Weak	Med.	Rec.	High
Key Recovery	2	3	4	2	3	4	5
Message Recovery	2	3	4	-	-	-	-

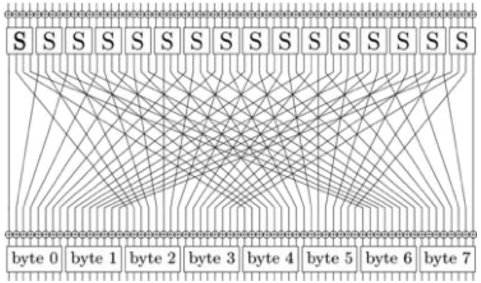
- Security of Kyber depends on LWE and LWR. This attack only removes the LWE instance.
- Round 2 version of Kyber removed LWR.

[1] Ravi, Prasanna et al.,"Number Not Used Once-Practical Fault Attack on pqm4 Implementations of NIST Candidates." In COSADE 2019.

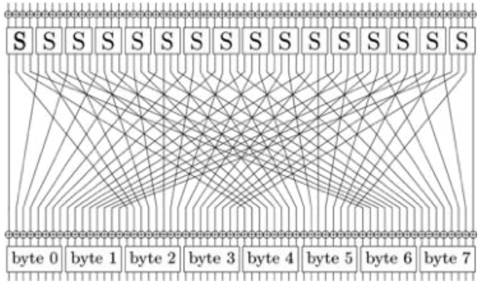
Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

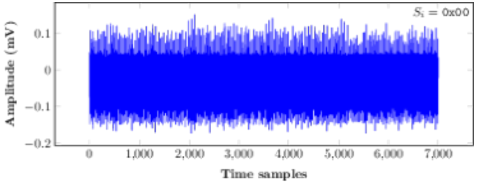
Attacking Bit Permutations



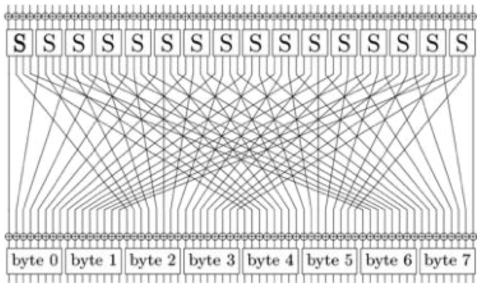
Attacking Bit Permutations



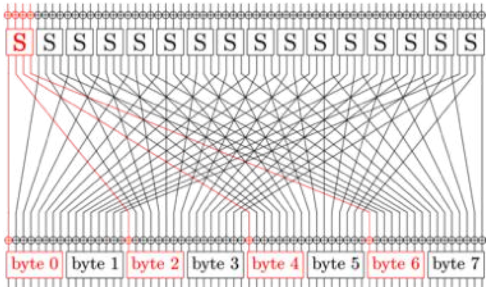
Measure
Power



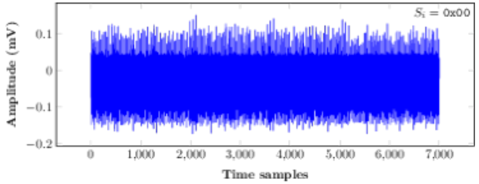
Attacking Bit Permutations



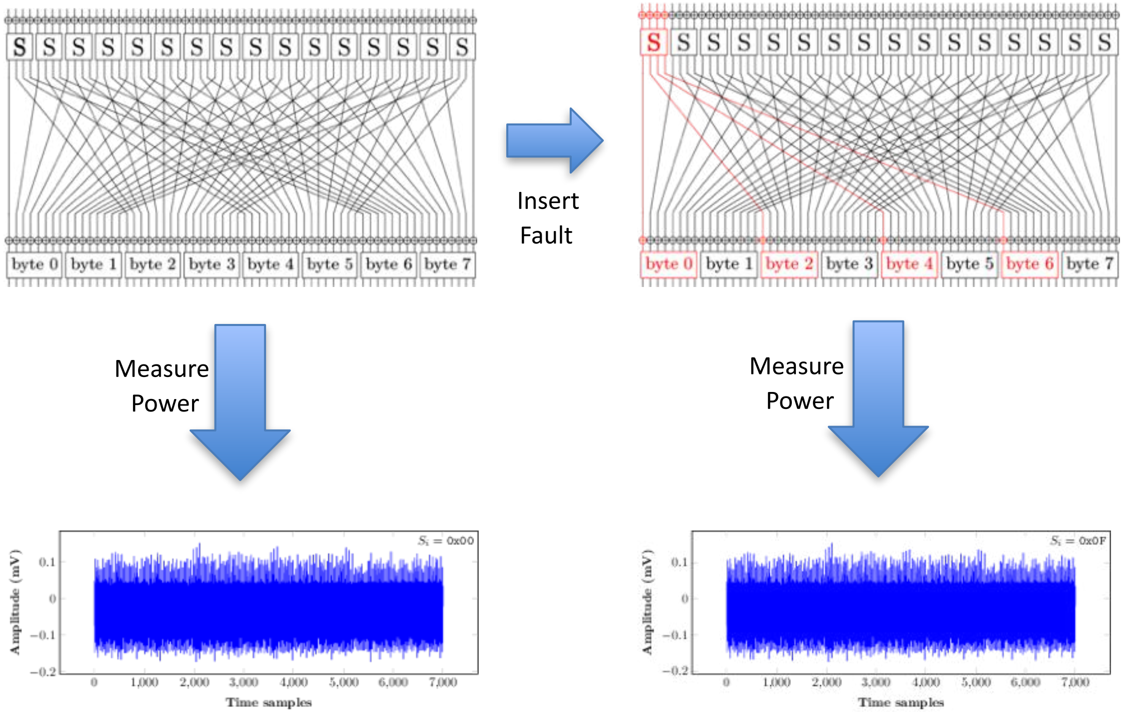
Insert
Fault



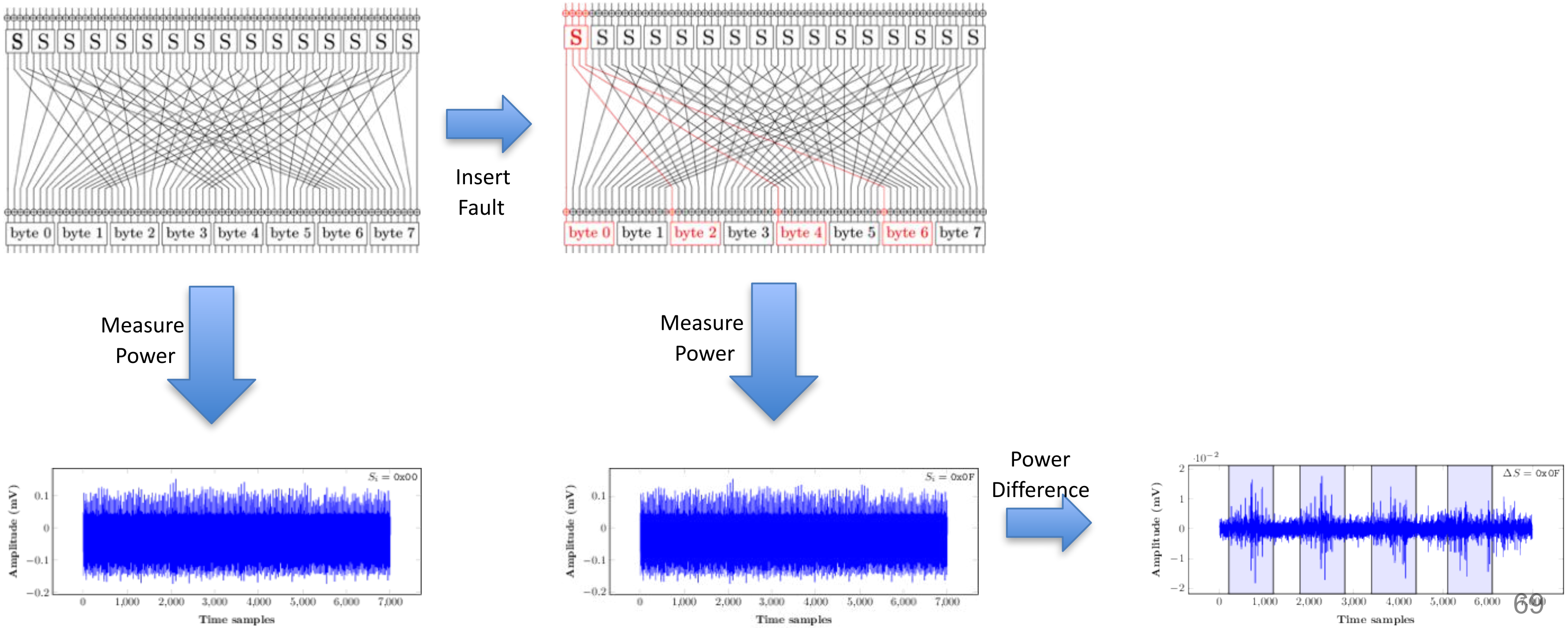
Measure
Power



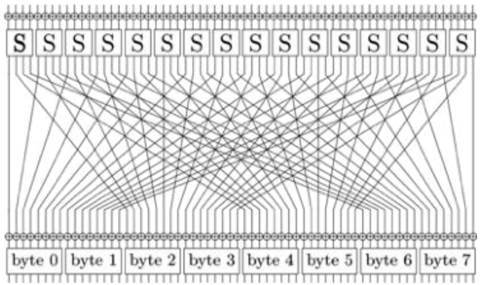
Attacking Bit Permutations



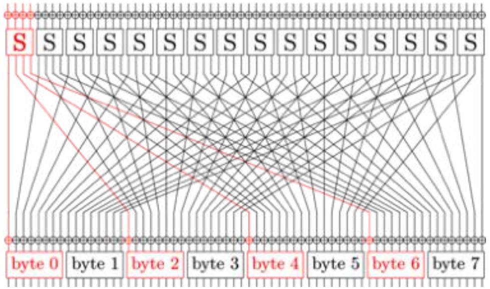
Attacking Bit Permutations

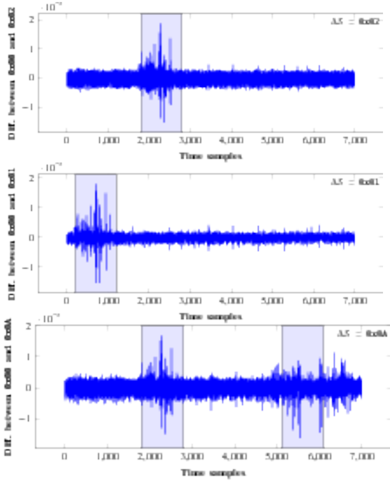


Attacking Bit Permutations



Insert
Fault

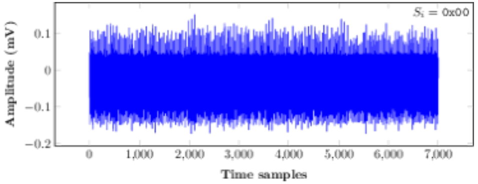


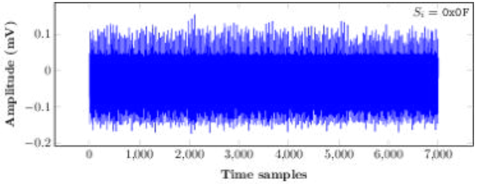


Measure
Power

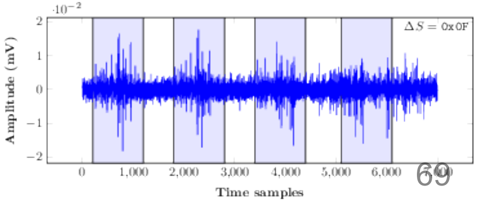
Measure
Power

Bit Permutation converts Difference to value model. Leaks fault mask



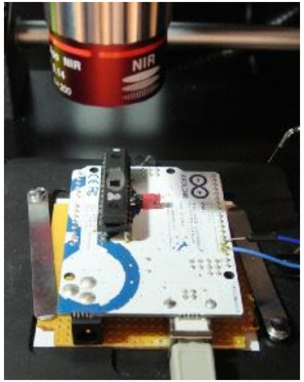


Power
Difference



Mounting Combined Attack

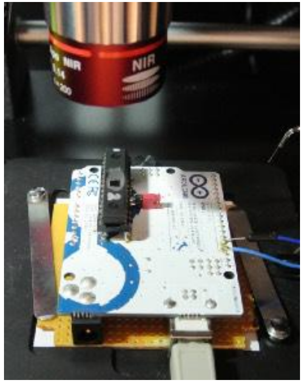
Mounting Combined Attack

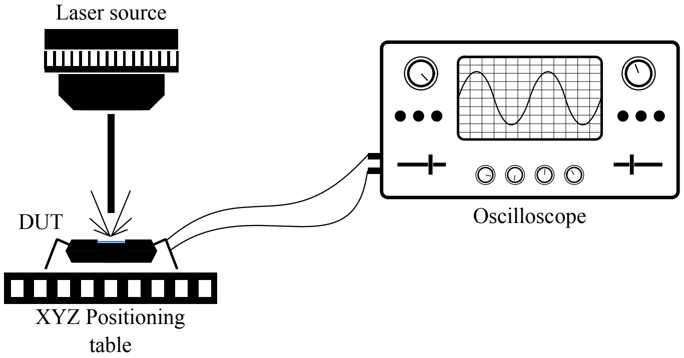


Combined Attack Setup:

- 1064 nm Laser
- 8-bit Atmega328P

Mounting Combined Attack

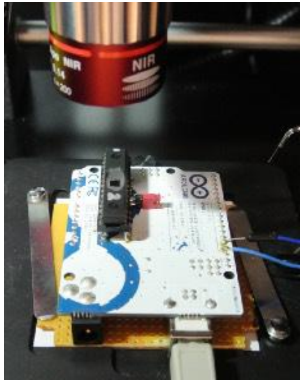


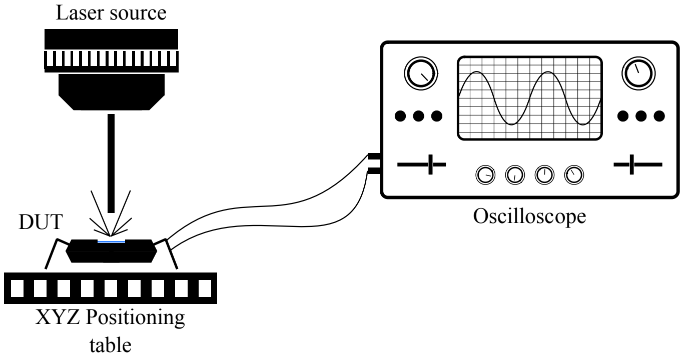


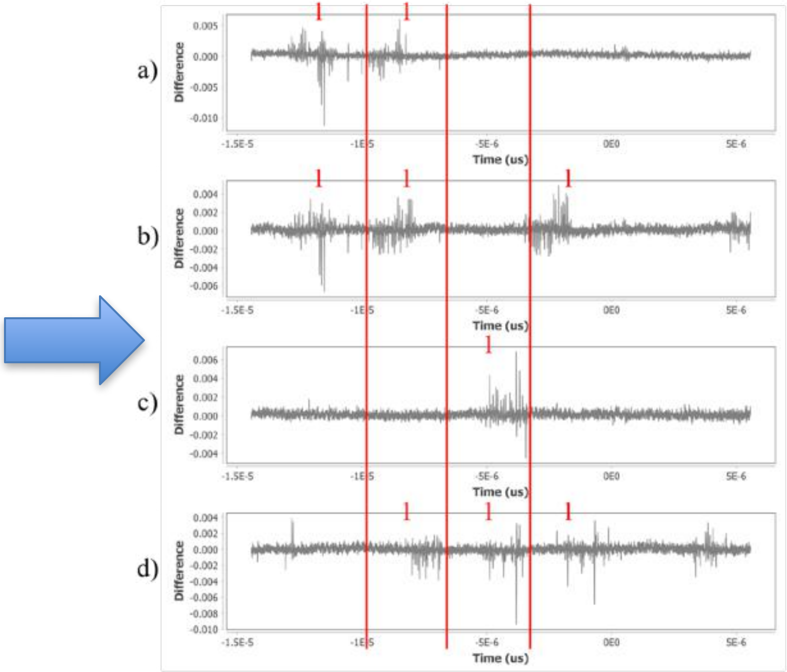
Combined Attack Setup:

- 1064 nm Laser
- 8-bit Atmega328P

Mounting Combined Attack



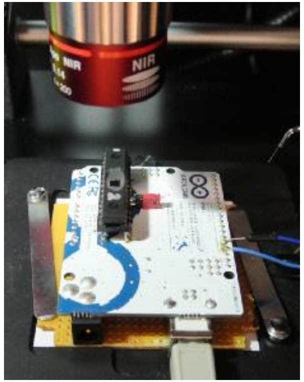


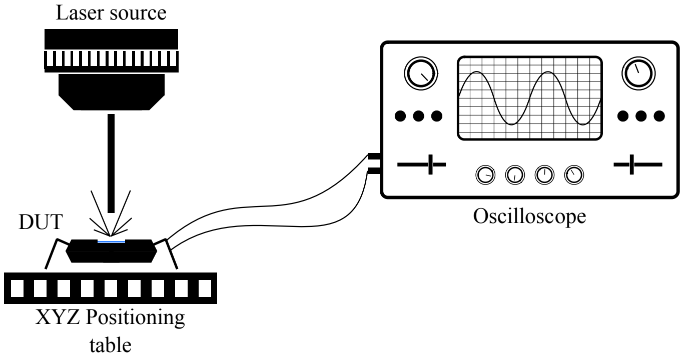


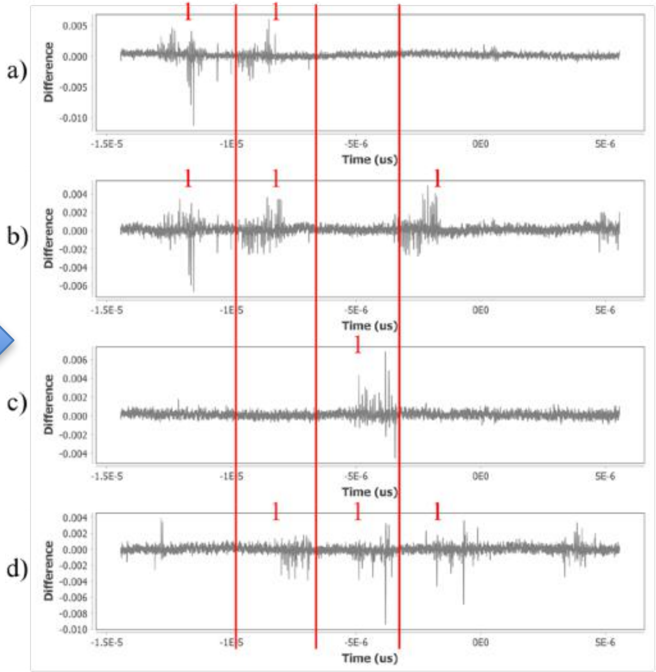
Combined Attack Setup:

- 1064 nm Laser
- 8-bit Atmega328P

Mounting Combined Attack



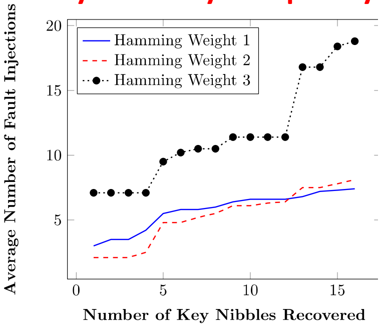




Combined Attack Setup:

- 1064 nm Laser
- 8-bit Atmega328P

Key Recovery Complexity



Attacks Against Redundancy

2014
- Guo et. al. (JCEN)
- Practically bypass concurrent error detection with biased faults.

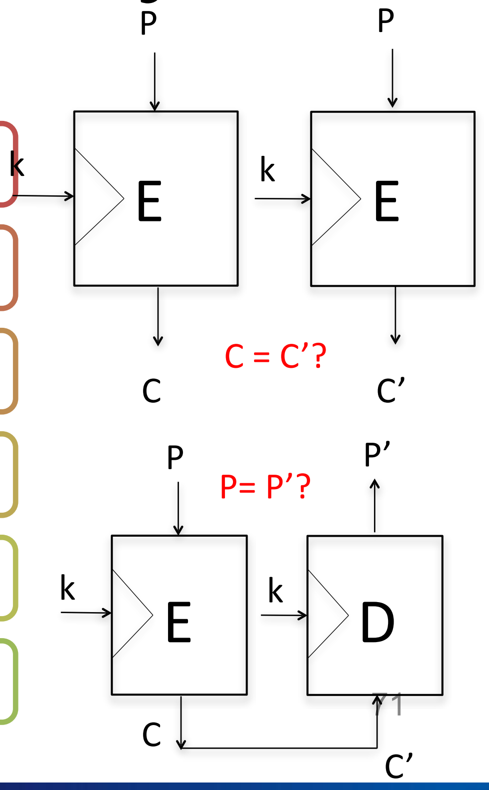
2015
- Patranabis et. al. (COSADE)
- Biased faults to bypass time-redundancy.

2016
- Selmake et. al. (FDTC)
- Biased faults to bypass hardware-redundancy.

2017
- Breier et. al. (JCEN)
- Practically bypass information redundancy (parity).

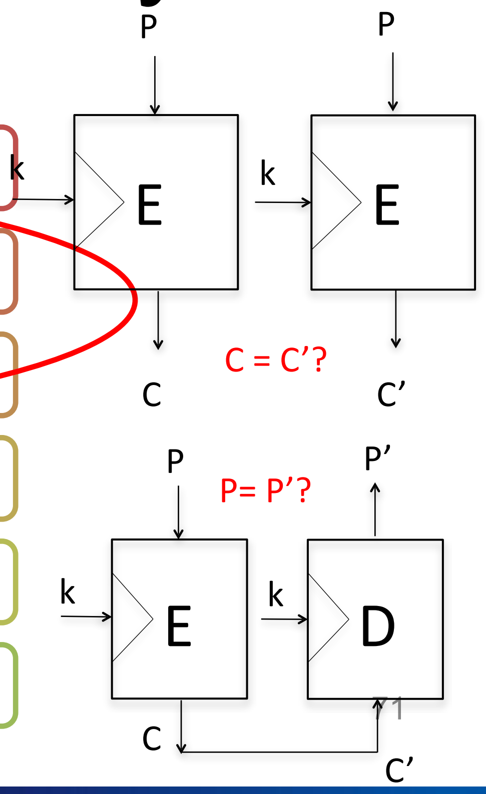
2017
- Wiersma et. al. (FDTC)
- Attack on commercial processors having ASIL-D security.

2018
- Zhang et al. (FDTC)
- Persistent fault analysis



Attacks Against Redundancy

- **2014**
  - Guo et. al. (JCEN)
  - Practically bypass concurrent error detection with biased faults.
- **2015**
  - Patranabis et. al. (COSADE)
  - Biased faults to bypass time-redundancy.
- **2016**
  - Selmake et. al. (FDTC)
  - Biased faults to bypass hardware-redundancy.
- **2017**
  - Breier et. al. (JCEN)
  - Practically bypass information redundancy (parity).
- **2017**
  - Wiersma et. al. (FDTC)
  - Attack on commercial processors having ASIL-D security.
- **2018**
  - Zhang et al. (FDTC)
  - Persistent fault analysis



Advancing State of the Art

- Previous Works
  - Bypass the countermeasure itself
  - Use biased faults
  - Corrupt all computation branches i.e. > 1 fault injection
- Our Proposal
  - Exploit the countermeasure itself
  - Use random faults (relaxed model)
  - **Only 1 fault injection**
  - Side-channel assisted

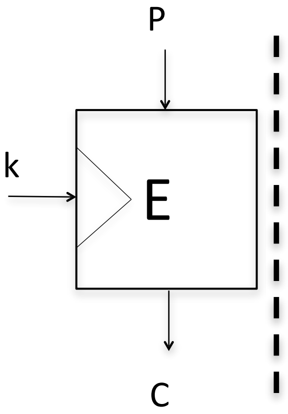
Advancing State of the Art

- Previous Works
  - Bypass the
  - Corrupt all computation branches i.e. > 1 fault injection
- Our Proposal
  - Exploit the
  - (relaxed model)
  - **Only 1 fault injection**
  - Side-channel assisted

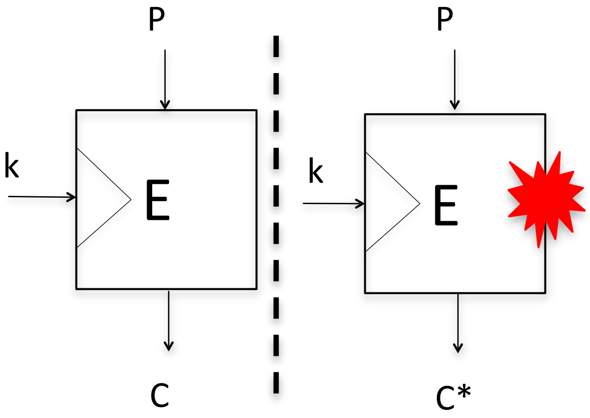
!!!EVEN COUNTERMEASURE LEAK!!!

Combined SCA+DFA on REDMR

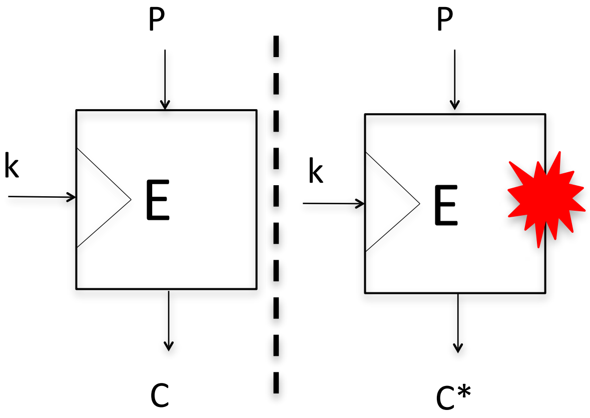
Combined SCA+DFA on REDMR



Combined SCA+DFA on REDMR

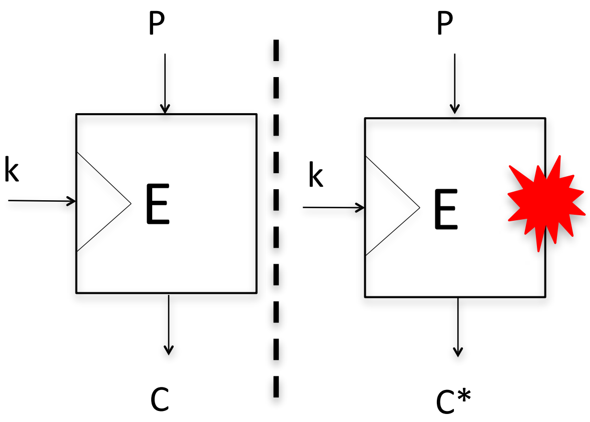


Combined SCA+DFA on REDMR



C XOR C’ = 0 ??

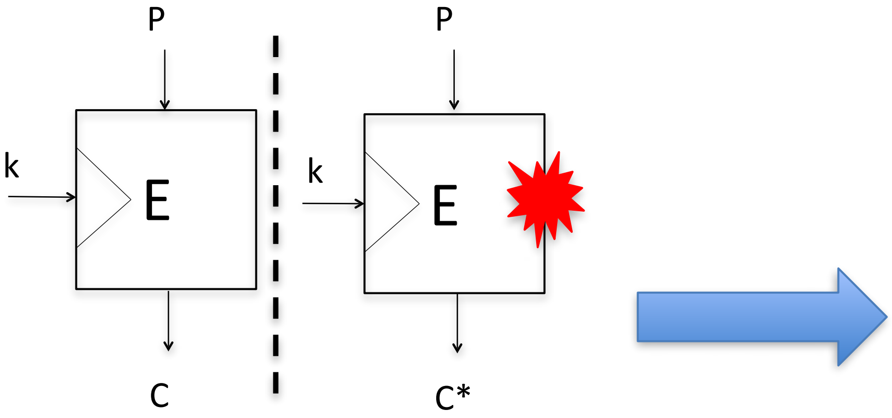
Combined SCA+DFA on REDMR



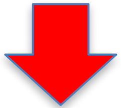
C XOR C’ = 0 ??

SCA Leakage = HW (C XOR C’)

Combined SCA+DFA on REDMR

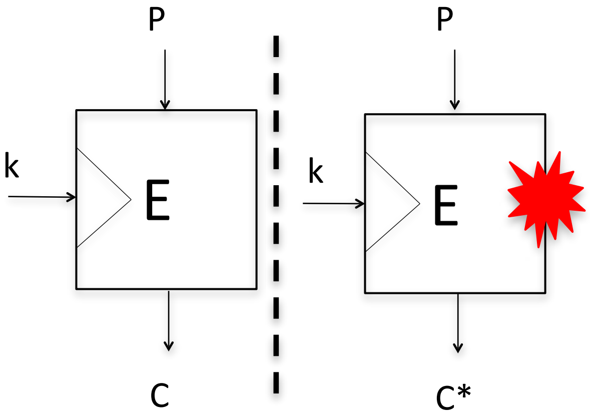


C XOR C' = 0 ??



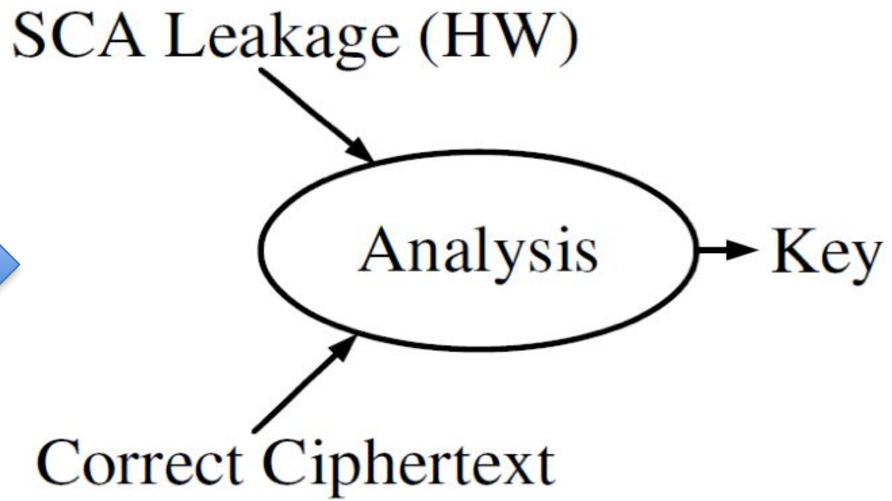
SCA Leakage = HW (C XOR C')

Combined SCA+DFA on REDMR

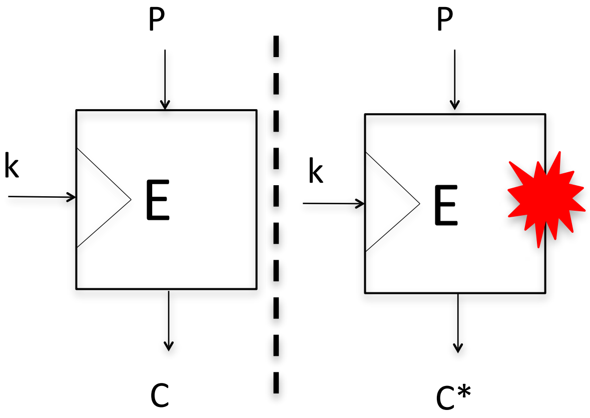


C XOR C' = 0 ??

SCA Leakage = HW (C XOR C')

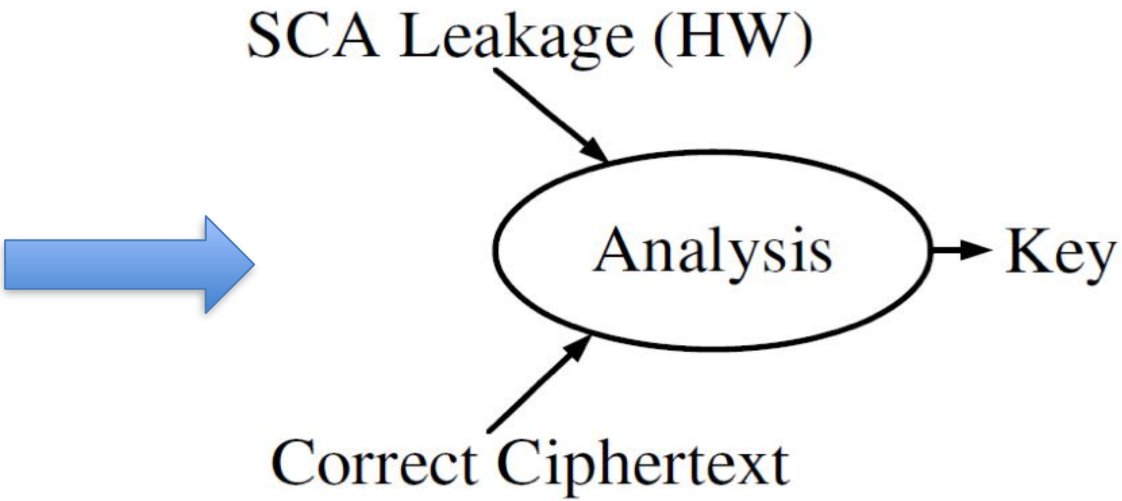


Combined SCA+DFA on REDMR



C XOR C' = 0 ??

SCA Leakage = HW (C XOR C')



[3] Breaking Redundancy-Based Countermeasures with Random Faults and Power Side Channel. FDTC 2019

The Attack Idea

- What we have and what we don't :
  - Correct ciphertexts: C → Known
  - Faulty Ciphertexts: $C*$ → Unknown
  - $HW(\delta) \;=\; HW(C \oplus C^*)$ → Know n (bytewise)

The Attack Idea

- What we have and what we don't :
  - Correct ciphertexts: C → Known
  - Faulty Ciphertexts: $C*$ → Unknown
  - $HW(\delta) = HW(C \oplus C^*)$ → Known (bytewise)

With:

$HW(\delta) = w$

$\binom{8}{w}$ choices for $C*$

Without:

2^8 choices for $C*$

The Attack Idea

- What we have and what we don't :
  - Correct ciphertexts: C → Known
  - Faulty Ciphertexts: $C*$ → Unknown
  - $HW(\delta) \;=\; HW(C \oplus C^*)$ → ~~Know n (bytewise)~~

With

Without

$HW(\delta) \;=\; w$

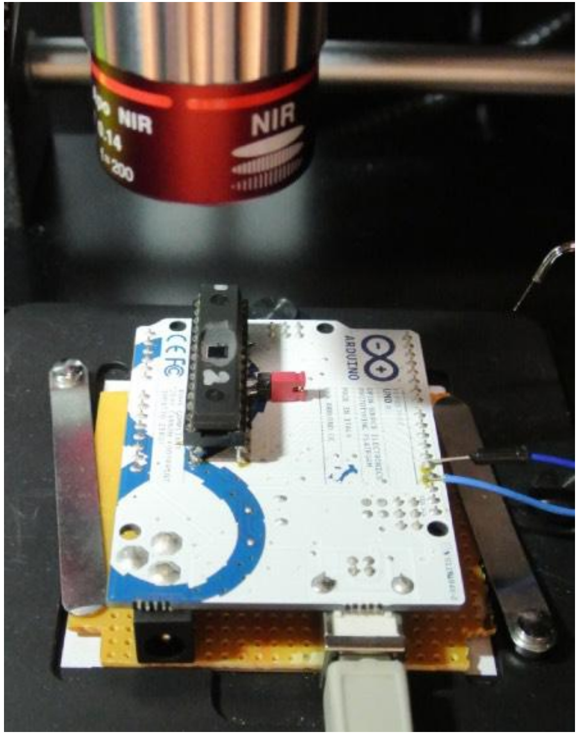
$\binom{8}{w}$choices for $C*$

2^8choices for $C*$

W=8,
C* Known

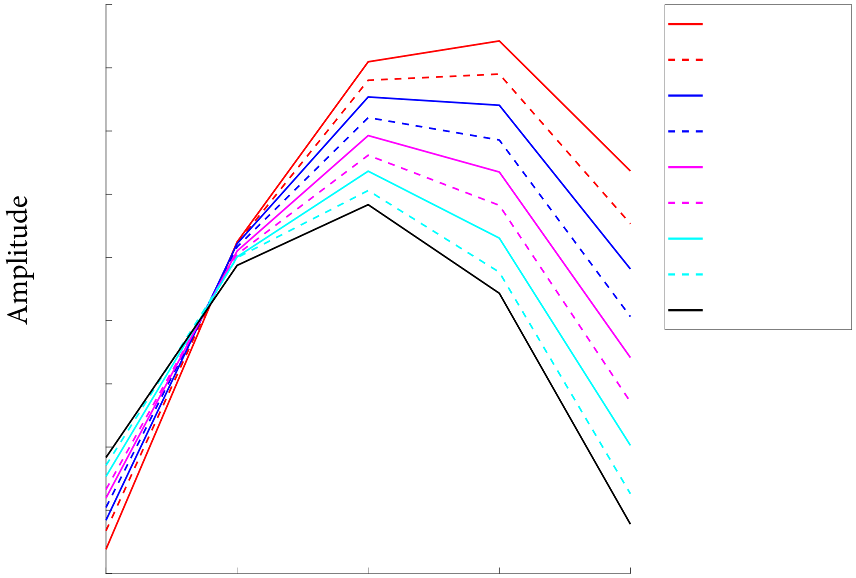
Less choices
for extreme w

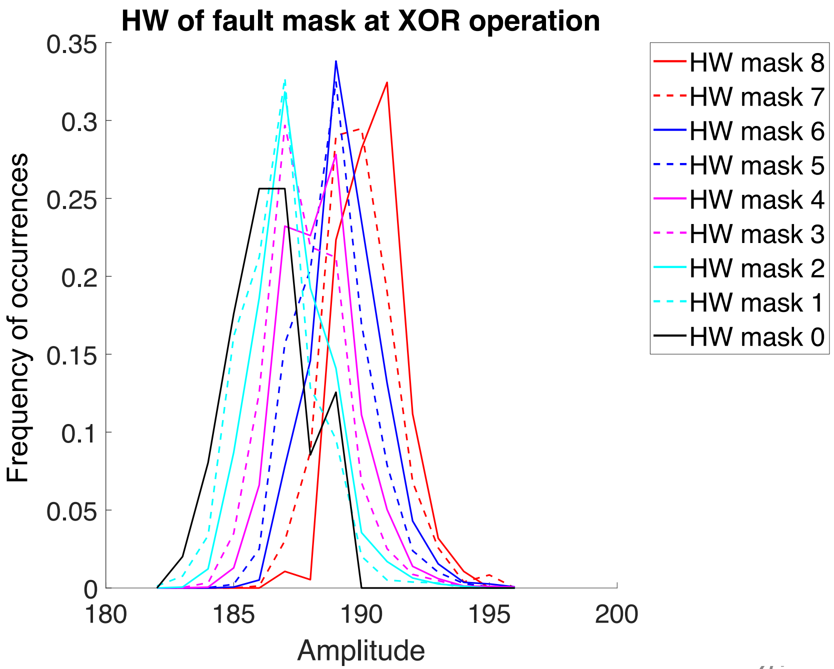
Attack Setup



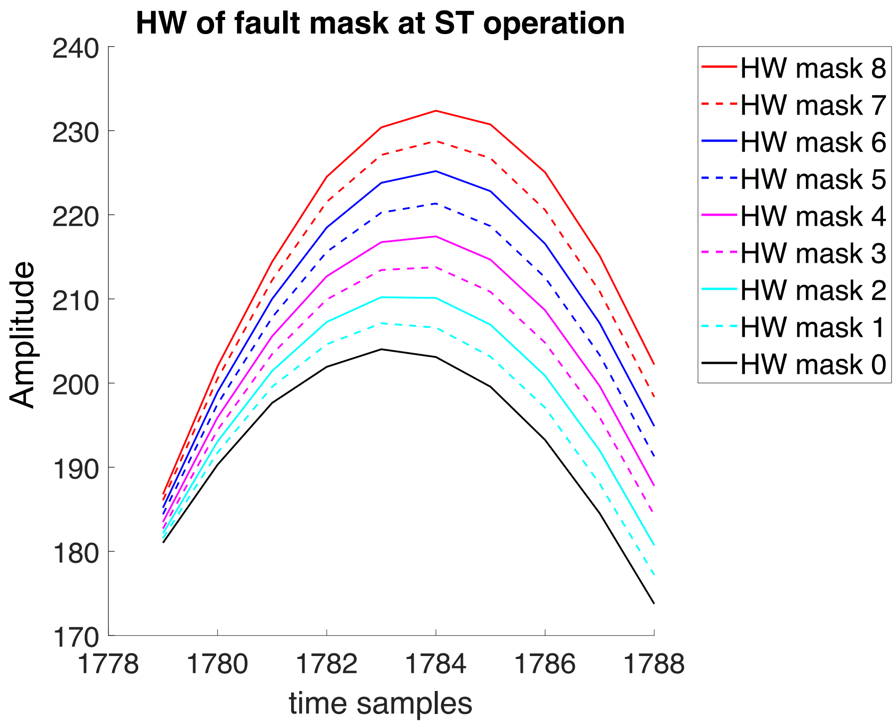
- Near-infrared diode pulse laser
- Maximum output power of 20 W
- For the experiments, 20x magnifying objective lens was used
- As a DUT, ATmega328P was used – an 8-bit microcontroller running at 16 MHz
- Chip was depackaged from the backside to be accessible by the laser
- Total area vulnerable to experiments was <1% of the entire chip area
- Reproducibility of faults was near to **100%** with the same laser settings

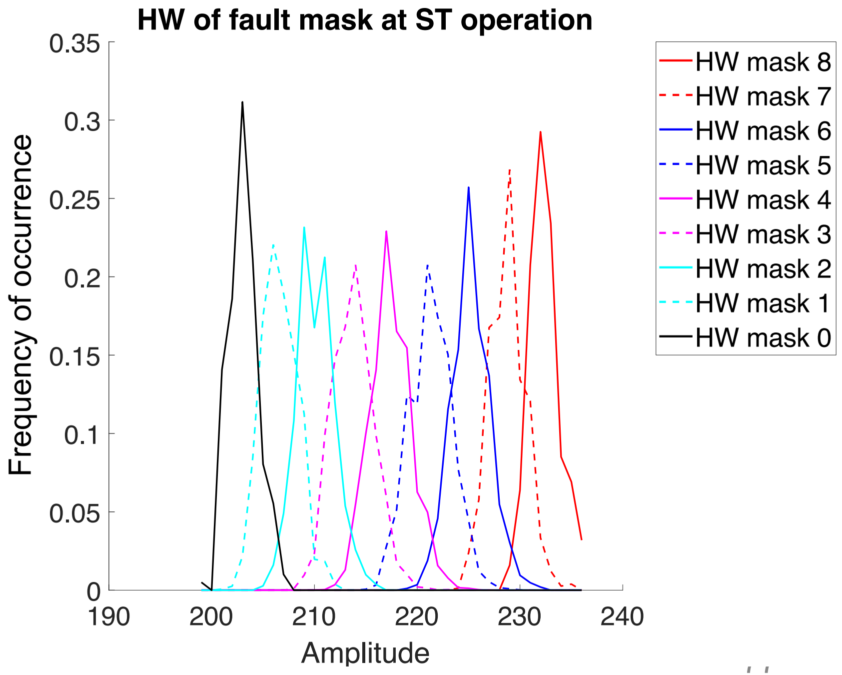
Observing C XOR C' (Sniff *XOR*)





Observing C XOR C' (Sniff *ST*)





Practical validation

Cipher	Code Size (bytes)	T_{ENC}	N_{EXP}	$(\lvert\mathcal{E}\rvert,\lvert\mathcal{F}\rvert,\lvert\mathcal{R}\rvert)$
AES-128	7570	0.326	$2^{26.98}$	$(2^{43}, 2^{25}, 1)$
PRESENT-80	7110	4.01	$2^{23.36}$	$(2^{4}, 4, 1)$

(|ε|,|ƒ|,|R|)= (Computation complexity, No. Of Faults, Remaining Key Space)

2^{25} Practical injections/day possible with our setup

Table of Contents

1. Introduction to Fault Attacks
2. Persistent Fault Analysis (PFA)
3. PFA on Higher-Order Masking
4. Fault Attack on Lattice based PQC
5. Combined SCA+DFA
6. Conclusions

Conclusions

- Persistent Fault Analysis (PFA)
  - A novel attack on general block ciphers
  - Defeat popular fault countermeasures & masking
  - Can work with multiple faults
  - Only one fault injection required

Conclusions

- Persistent Fault Analysis (PFA)
  - A novel attack on general block ciphers
  - Defeat popular fault countermeasures & masking
  - Can work with multiple faults
  - Only one fault injection required
- Fault attack on Lattice based PQC
  - Identified fault vulnerabilities in nonce optimization
  - Targets multiple schemes like NewHope, Frodo, Kyber and Dilithium
  - Validated on ARM Cortex-M4 with public pqm4 library

Conclusions

- Persistent Fault Analysis (PFA)
  - A novel attack on general block ciphers
  - Defeat popular fault countermeasures & masking
  - Can work with multiple faults
  - Only one fault injection required
- Fault attack on Lattice based PQC
  - Identified fault vulnerabilities in nonce optimization
  - Targets multiple schemes like NewHope, Frodo, Kyber and Dilithium
  - Validated on ARM Cortex-M4 with public pqm4 library
- Combined SCA+DFA
  - Pushing the limits
  - Exploiting leakage from design optimization choices and countermeasures

Thank You !!!