Radbound University

Side-channel analysis of cryptographic implementations: Lessons learned and future directions

BFA, Voss, Norway
September 7, 2023

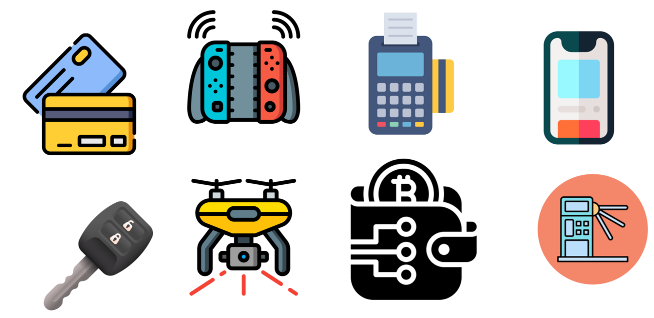
Lejla Batina

Institute for Computing and Information Sciences
Radboud University
lejla@cs.ru.nl

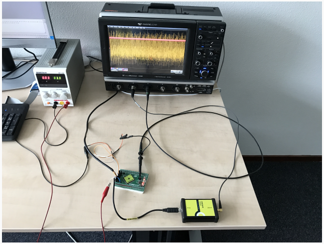
Outline

Intro to side-channel analysis

Known challenge: embedded crypto devices



Implementation attacks



Relevance

October 3, 2019

Researchers Discover ECDSA Key Recovery Method

October 3, 2019 • Add Comment • by Emma Davis

Relevance

November 13, 2019

October 3, 2019

November 13, 2019

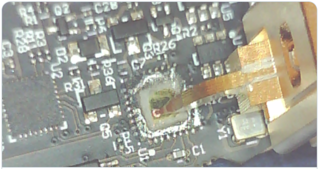
October 3, 2019

Researchers Discover ECDSA
Key Recovery Method

October 3, 2019 • Add Comment • by Emma Davis

January 7, 2021

A Side-Channel Attack on the Google Titan Security Key



November 13, 2019

October 3, 2019

Researchers Discover ECDSA Key Recovery Method

October 3, 2019 • Add Comment • by Emma Davis

March 16, 2023

No, AI did not break post-quantum cryptography

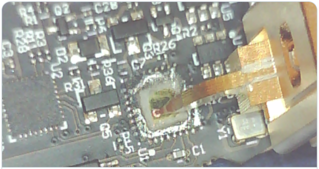
16/03/2023

Lejla Batina (Guest author) Stjepan Picek (Guest author)

Bas Westerbaan

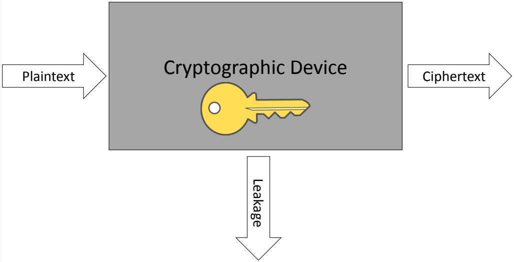
January 7, 2021

A Side-Channel Attack on the Google Titan Security Key

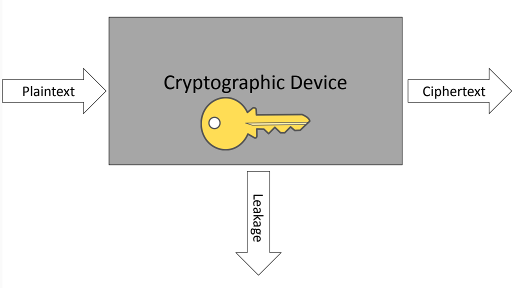


Side-channel Analysis (SCA) Attacks and Countermeasures

Greybox/Whitebox scenario



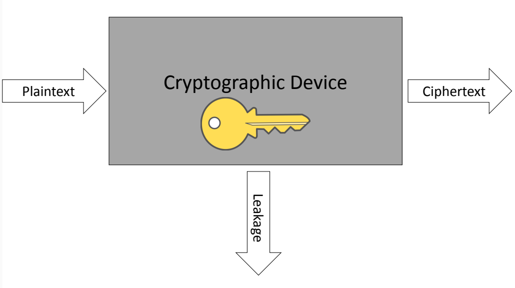
Greybox/Whitebox scenario



Greybox = SCA adversary in the wild:

- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC

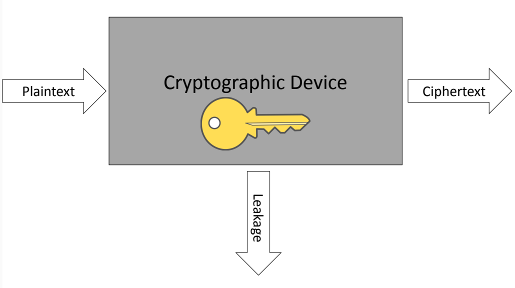
Greybox/Whitebox scenario



Greybox = SCA adversary in the wild:

- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC
- Adversary can measure and process *physical quantities* in the device's vicinity

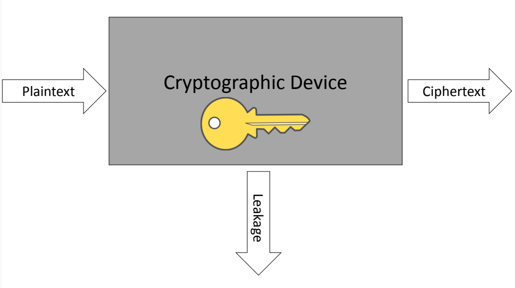
Greybox/Whitebox scenario



Greybox = SCA adversary in the wild:

- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC
- Adversary can measure and process *physical quantities* in the device's vicinity
- Adversary's goal: secret key, message recovery, IP, etc.

Greybox/Whitebox scenario

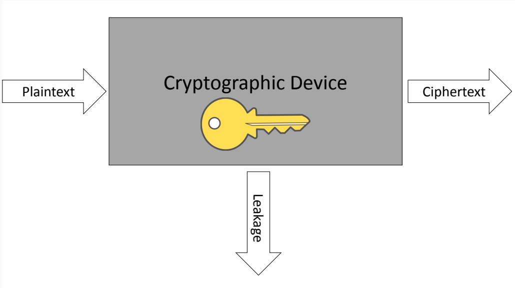


Greybox = SCA adversary in the wild:

- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC
- Adversary can measure and process *physical quantities* in the device's vicinity
- Adversary's goal: secret key, message recovery, IP, etc.

Whitebox = Security evaluator:

Greybox/Whitebox scenario



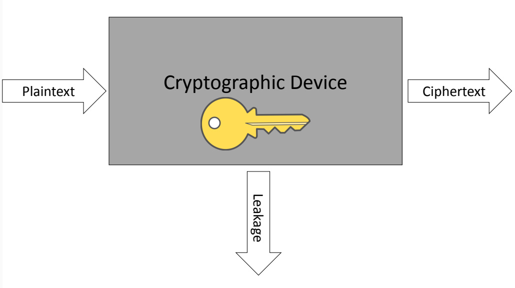
Greybox = SCA adversary in the wild:

- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC
- Adversary can measure and process *physical quantities* in the device's vicinity
- Adversary's goal: secret key, message recovery, IP, etc.

Whitebox = Security evaluator:

- Algorithms and implementation details are (partially) known

Greybox/Whitebox scenario



Greybox = SCA adversary in the wild:

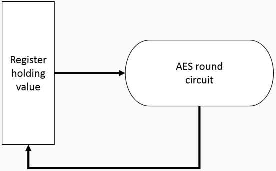
- Crypto is **implemented on a real device** such as a microcontroller, FPGA, ASIC
- Adversary can measure and process *physical quantities* in the device's vicinity
- Adversary's goal: secret key, message recovery, IP, etc.

Whitebox = Security evaluator:

- Algorithms and implementation details are (partially) known
- Adversary's goal: secret key or message recovery by observing input/output pairs while trying all known attacks

Power side-channel: Modeling the leakage

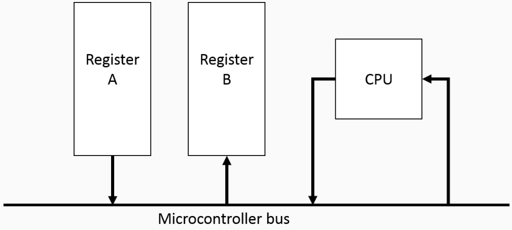
- The Hamming distance model counts the number of $0 \to 1$ and $1 \to 0$ transitions
- Example 1: Assume a hardware register R storing the result of an AES round. The register initially contains value v_0 and gets overwritten with value v_1



- The power consumption because of the register transition $v_0 \to v_1$ is related to the number of bit flips that occurred
- Thus it can be modeled as $\text{HammingDistance}(v_0, v_1) = \text{HammingWeight}(v_0 \oplus v_1)$

- Example 2: In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

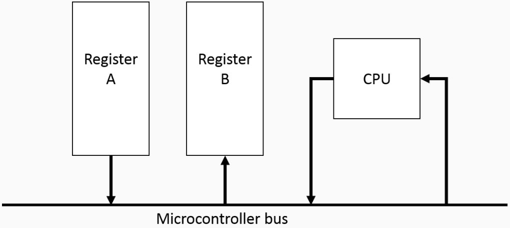
```
mov rB, rA
```



Power side-channel: Modeling the leakage

- Example 2: In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

```
mov rB, rA
```

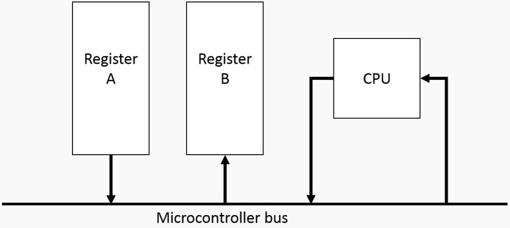


- In general-purpose processors the instruction will transfer value v_0 from register A to B via the CPU, using the bus

Power side-channel: Modeling the leakage

- Example 2: In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

```
mov rB, rA
```

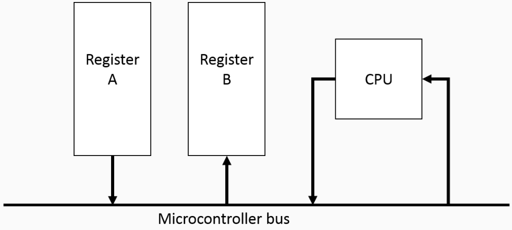


- In general-purpose processors the instruction will transfer value v_0 from register A to B via the CPU, using the bus
- Often the bus is a very leaky component and also precharged to all bits to zeros (or all to 1) i.e. busInitialValue

Power side-channel: Modeling the leakage

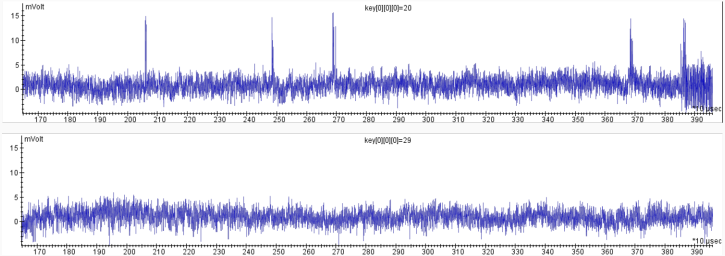
- Example 2: In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

```
mov rB, rA
```



- In general-purpose processors the instruction will transfer value v_0 from register A to B via the CPU, using the bus
- Often the bus is a very leaky component and also precharged to all bits to zeros (or all to 1) i.e. busInitialValue
- The power consumption of the assembly instruction can be modeled as $\text{HammingDistance}(\text{busInitialValue}, v_0) = \text{HammingWeight}(v_0 \oplus 0) = \text{HW}(v_0)$

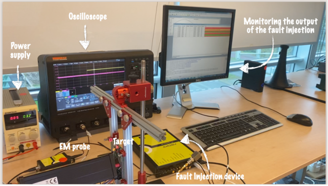
Differential Power Analysis (DPA)



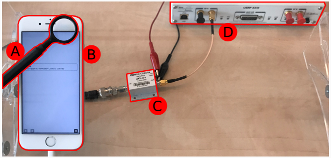
- The most popular side-channel attack
- Aims at recovering the secret key by using a large number of power measurements (traces)
- Nowadays often combined/replaced with a leakage evaluation methodology such as Test Vector Leakage Assessment (TVLA)

Actual setups

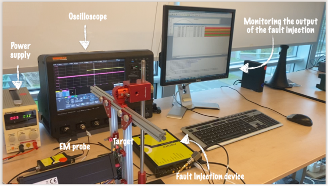
FA setup



Tempest

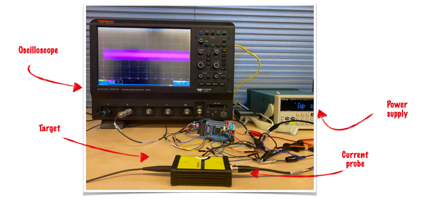


FA setup



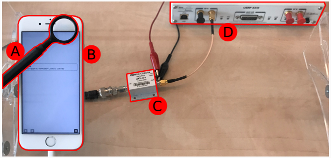
Actual setups

DPA setup

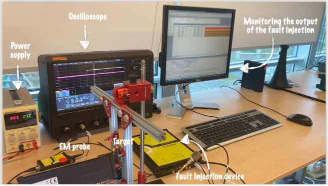


@CESCA, Radboud University

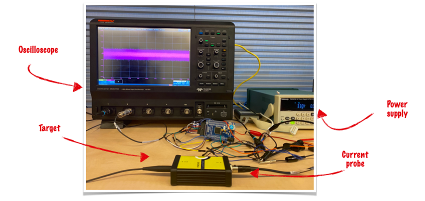
Tempest



FA setup

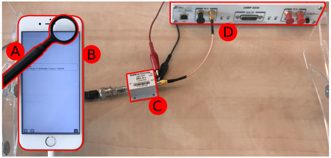


DPA setup

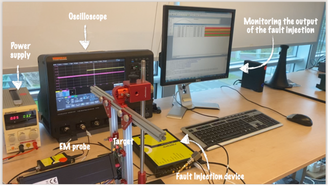


@CESCA, Radboud University

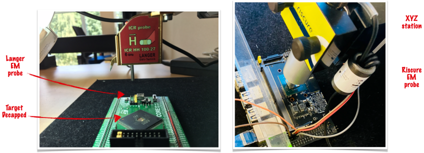
Tempest



FA setup



EM setups



@CESCA, Radboud University

Countermeasures: main idea

Goal: break the link between the actual data and power consumption

Countermeasures: main idea

Goal: break the link between the actual data and power consumption

- Masking: power consumption remains dependent on the data on which computation is performed but not the actual data

Goal: break the link between the actual data and power consumption

- Masking: power consumption remains dependent on the data on which computation is performed but not the actual data
- Hiding: power consumption is independent of the intermediate values and of the operations

Masking

Boolean masking: a dth-order (Boolean) masking scheme splits an internal sensitive value v into $d+1$ shares $(v_0, v_1, ..., v_d)$, as follows:

$$v = v_0 \oplus v_1 \oplus \cdots \oplus v_d$$

Boolean masking: a dth-order (Boolean) masking scheme splits an internal sensitive value v into $d+1$ shares $(v_0, v_1, ..., v_d)$, as follows:

$$v = v_0 \oplus v_1 \oplus \cdots \oplus v_d$$

Probing-secure scheme. We refer to a scheme that uses certain families of shares as $d-$probing-secure iff any set of at most d intermediate variables is independent from the sensitive values.

Consequently, the leakage of up to d values does not disclose any information to the attacker.

Masking in practice: unintended interactions between values in the processor cause leakage in 1st order (caused often by transitional effects and glitches).

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$
- The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- It does not reveal info on the value of X when a DPA is performed,

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$
- The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- It does not reveal info on the value of X when a DPA is performed, in theory

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$
- The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- It does not reveal info on the value of X when a DPA is performed, in theory

x	x_1	x_2	$\mathcal{L}(x)$	Mean($\mathcal{L}(x)$)	Var($\mathcal{L}(x)$)
0	0	0	0	1	1
	1	1	2		
1	0	1	1	1	0
	1	0	1		

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$
- The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- It does not reveal info on the value of X when a DPA is performed, in theory

x	x_1	x_2	$\mathcal{L}(x)$	$\mathrm{Mean}(\mathcal{L}(x))$	$\mathrm{Var}(\mathcal{L}(x))$
0	0	0	0	1	1
	1	1	2		
1	0	1	1	1	0
	1	0	1		

If a program that processes a secret value X contains two consecutive instructions (the first uses X_1 and the second uses X_2), then the transitional effect of changing the contents of the bus leaks the Hamming distance between X_1 and X_2.

Masking with 2 shares

- Split the intermediate variable X into two shares X_1 and X_2 such that $X_1 \oplus X_2 = X$
- The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- It does not reveal info on the value of X when a DPA is performed, in theory

x	x_1	x_2	$\mathcal{L}(x)$	$\mathrm{Mean}(\mathcal{L}(x))$	$\mathrm{Var}(\mathcal{L}(x))$
0	0	0	0	1	1
	1	1	2		
1	0	1	1	1	0
	1	0	1		

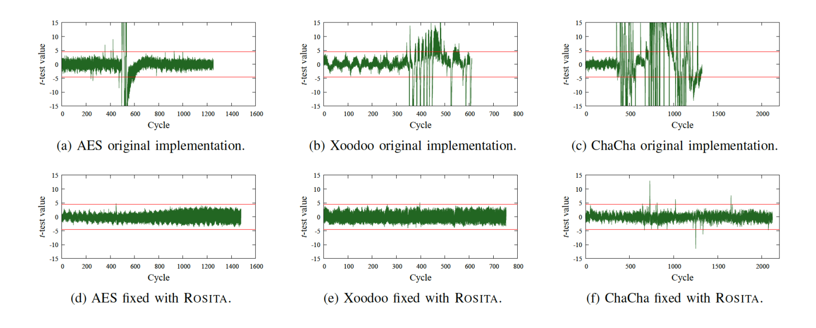
If a program that processes a secret value X contains two consecutive instructions (the first uses X_1 and the second uses X_2), then the transitional effect of changing the contents of the bus leaks the Hamming distance between X_1 and X_2.

Actually, Balasch et al. show [BGGRS14] that unintended interactions typically halve the number of intermediate values the adversary needs to acquire.

Test Vector Leakage Assessment (TVLA)

- Leakage assessment of a device is very important for the semiconductor and the security evaluation industries
- Number of attacks to check the device's resistance against keeps on growing
- Various attackers' models possible but security evaluation often goes for the strongest adversary
- It is using Welch's t-test to differentiate between two sets of measurements, one with fixed inputs and the other with random inputs
- Leakage from combining multiple points is not detected

TVLA on masked implementations



The slowdowns of the "fixes" for ChaCha, Xoodoo and AES are 61% (1 322 vs. 2 122 cycles), 18% (637 vs. 753 cycles) and 15% (1 285 vs 1 479).

M. A. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, Y. Yarom: Rosita: Towards Automatic Elimination of Power-Analysis Leakage in Ciphers. NDSS 2021.

SCA and AI

- Machine learning for SCA was a natural direction:

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking
  - leakage assessment/simulators

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking
  - leakage assessment/simulators
  - TEMPEST-like techniques e.g. screen gleaning

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking
  - leakage assessment/simulators
  - TEMPEST-like techniques e.g. screen gleaning
- SCA attacks on AI:
  - SCA for reverse engineering neural net (NN) implementations

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking
  - leakage assessment/simulators
  - TEMPEST-like techniques e.g. screen gleaning
- SCA attacks on AI:
  - SCA for reverse engineering neural net (NN) implementations
  - applied to various platforms such as FPGAs, GPUs

AI and Side-channel Analysis (SCA): Why

- Machine learning for SCA was a natural direction:
  - PCA to assist profiling/template attacks (dimensionality reduction)
  - PCA for pre-processing measurement traces
  - Machine Learning (ML)-based SCA distingushers
- Deep learning in SCA:
  - neural nets for profiling attacks
  - defeating countermeasures e.g. attacking higher-order masking
  - leakage assessment/simulators
  - TEMPEST-like techniques e.g. screen gleaning
- SCA attacks on AI:
  - SCA for reverse engineering neural net (NN) implementations
  - applied to various platforms such as FPGAs, GPUs
  - input recovery from NN implementations

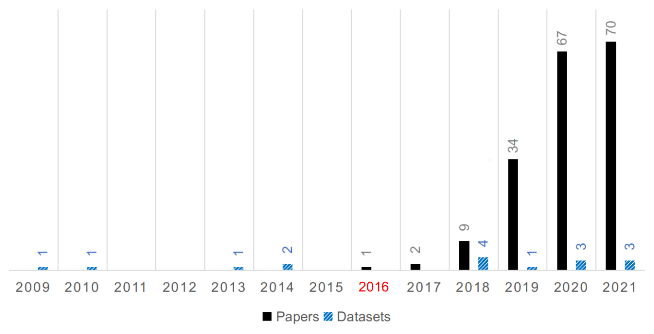
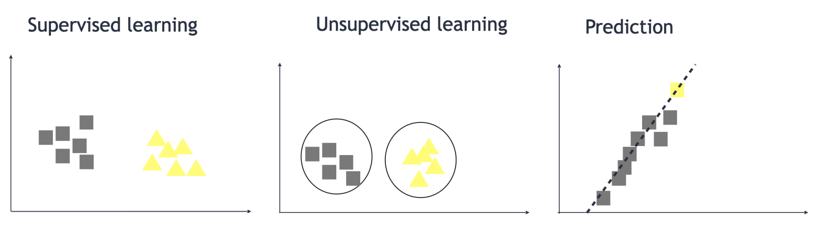
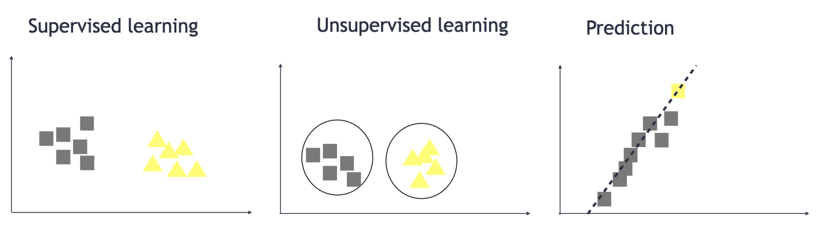


Figure: **Deep learning papers and datasets**.

S. Picek, G. Perin, L. Mariot, L. Wu and L. Batina, SoK: Deep Learning-based Physical Side-channel Analysis, `https://eprint.iacr.org/2021/1092`, ACM Comput. Surv. 55(11): 227:1-227:35 (2023)

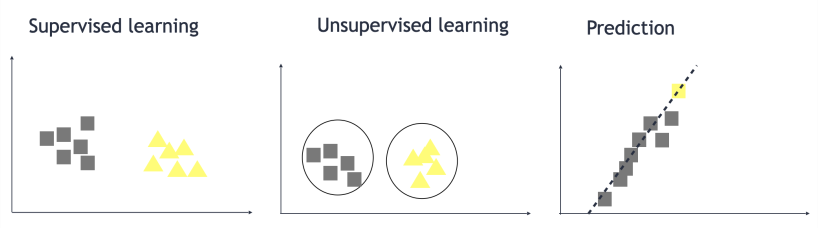

Supervised learning
Unsupervised learning
Prediction

Applications for machine learning



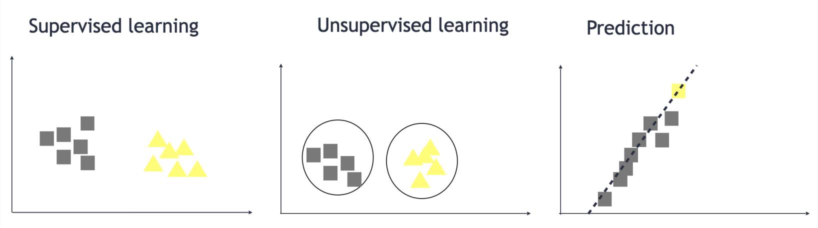
- Supervised learning (the most common in SCA): the machine learns with a supervisor, for every example we tell the machine what the correct answer is;

Applications for machine learning



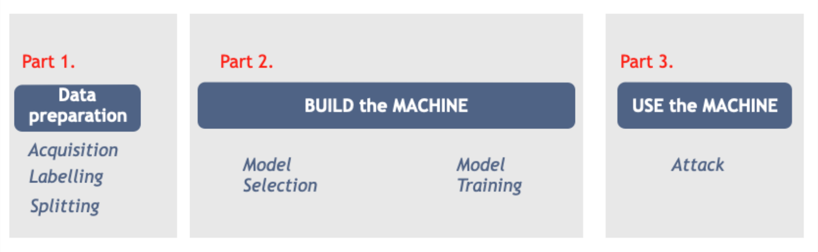
- Supervised learning (the most common in SCA): the machine learns with a supervisor, for every example we tell the machine what the correct answer is;
- Unsupervised learning: the machine discovers hidden patterns in the data; training is done on unlabelled data;

Applications for machine learning

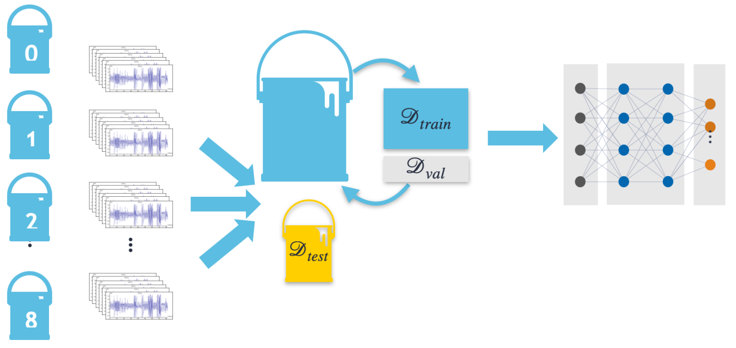


- Supervised learning (the most common in SCA): the machine learns with a supervisor, for every example we tell the machine what the correct answer is;
- Unsupervised learning: the machine discovers hidden patterns in the data; training is done on unlabelled data;
- Prediction: the machine predicts the future, based on past events

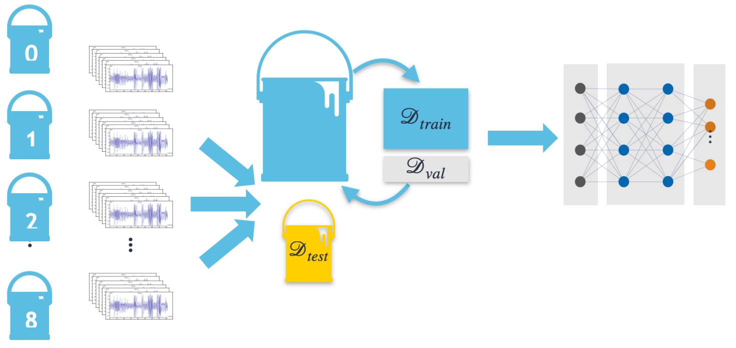
Workflow machine learning



A visual overview of how data is split:



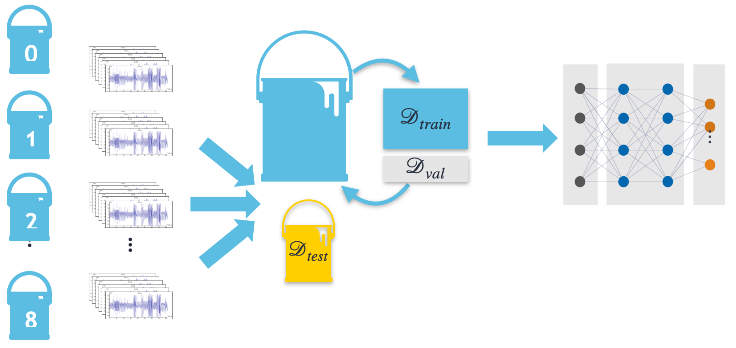
A visual overview of how data is split:



- Over-fitting: an undesired phenomena, when the performance of the model on the training data is very good, while the performance on testing data is poor;

Data preparation for SCA

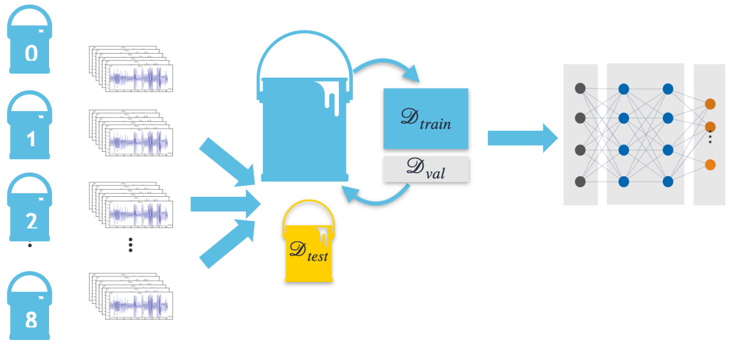
A visual overview of how data is split:



- Over-fitting: an undesired phenomena, when the performance of the model on the training data is very good, while the performance on testing data is poor;
- Cross-validation: the process of splitting the training data repeatedly into training and validation sets for more reliable results, which avoid over-fitting;

Data preparation for SCA

A visual overview of how data is split:



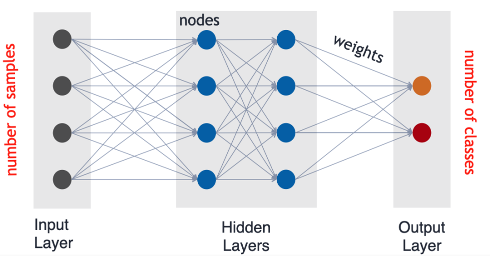
- Over-fitting: an undesired phenomena, when the performance of the model on the training data is very good, while the performance on testing data is poor;
- Cross-validation: the process of splitting the training data repeatedly into training and validation sets for more reliable results, which avoid over-fitting;
- Under-fitting: when the model does not produce accurate results on the training data;

A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.

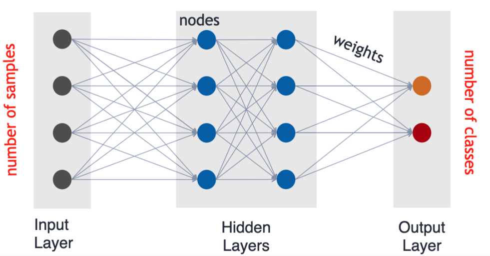
A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.



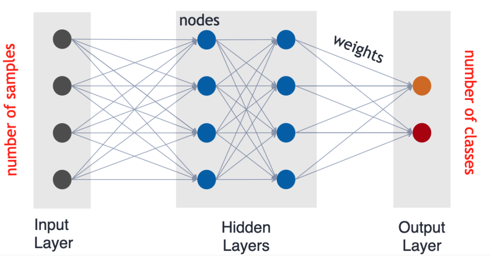
A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.



- In SCA, we use relatively small networks and simple arch.: MLP and CNN

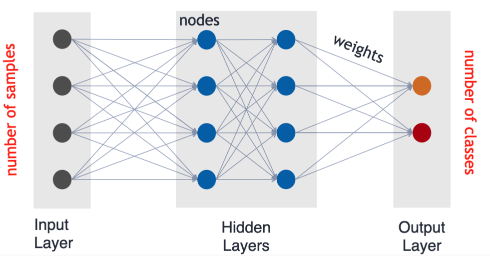
A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.



- In SCA, we use relatively small networks and simple arch.: MLP and CNN
- During training the value of the weights and biases are adjusted;

A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.



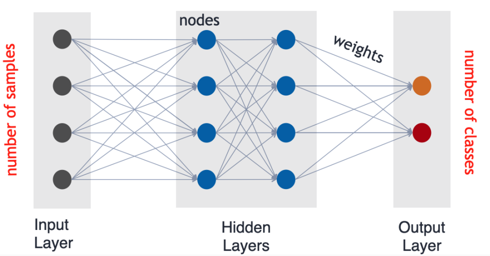
- In SCA, we use relatively small networks and simple arch.: MLP and CNN
- During training the value of the weights and biases are adjusted;

SCA-context:

- # nodes in the input layer = # of samples in a trace;

A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.



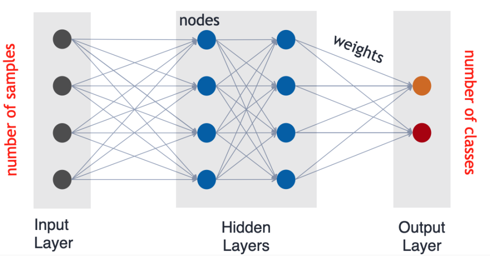
- In SCA, we use relatively small networks and simple arch.: MLP and CNN
- During training the value of the weights and biases are adjusted;

SCA-context:

- # nodes in the input layer = # of samples in a trace;
- # nodes in the output layer = # the number of labels (a.k.a. classes);

A simple MLP architecture

A simple multilayer perceptron (MLP) neural network architecture, which contains a series of layers formed of connected neurons. The strength of the connection between two neurons is determined by the associated weight.

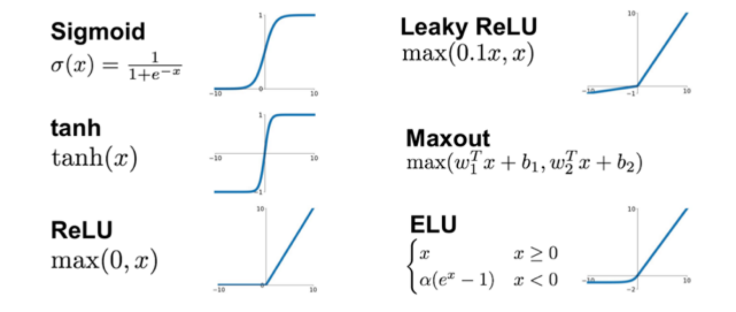


- In SCA, we use relatively small networks and simple arch.: MLP and CNN
- During training the value of the weights and biases are adjusted;

SCA-context:

- # nodes in the input layer = # of samples in a trace;
- # nodes in the output layer = # the number of labels (a.k.a. classes);
- We only need to make decisions about the hidden layer

We need to deal with non-linear functions.



SCA of PQC Implementations

Similarities to SCA attacks “classical” PKC

- Application is providing an attack scenario i.e. a threat model

Similarities to SCA attacks "classical" PKC

- Application is providing an attack scenario i.e. a threat model
- Ephemeral i.e. 1-time vs static keys: profiled or non-profiled attacks

Similarities to SCA attacks "classical" PKC

- Application is providing an attack scenario i.e. a threat model
- Ephemeral i.e. 1-time vs static keys: profiled or non-profiled attacks
- Partial key exposure attacks

Similarities to SCA attacks "classical" PKC

- Application is providing an attack scenario i.e. a threat model
- Ephemeral i.e. 1-time vs static keys: profiled or non-profiled attacks
- Partial key exposure attacks
- Horizontal attacks: a bunch of techniques all basically focusing on a single-trace attack

Similarities to SCA attacks "classical" PKC

- Application is providing an attack scenario i.e. a threat model
- Ephemeral i.e. 1-time vs static keys: profiled or non-profiled attacks
- Partial key exposure attacks
- Horizontal attacks: a bunch of techniques all basically focusing on a single-trace attack
  - using ideas from collision-based attacks on symmetric crypto
  - Online Template Attacks (OTA)
  - as PKC implementations typically take a "long" time, there is much more info to exploit in a trace

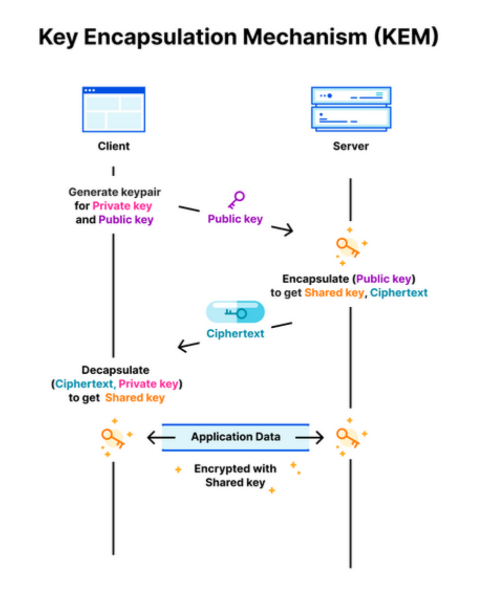
Similarities to SCA attacks "classical" PKC

- Application is providing an attack scenario i.e. a threat model
- Ephemeral i.e. 1-time vs static keys: profiled or non-profiled attacks
- Partial key exposure attacks
- Horizontal attacks: a bunch of techniques all basically focusing on a single-trace attack
    - using ideas from collision-based attacks on symmetric crypto
    - Online Template Attacks (OTA)
    - as PKC implementations typically take a "long" time, there is much more info to exploit in a trace

Attacking (lattice-based) PQC

- Due to the NIST PQC competition a lot of research is done on implementations
- Implementation attacks on all finalists were discussed
- Algorithm-specific and other countermeasures proposed
- SCA attacks on implementations protected against higher order attacks found to be feasible
- Deep learning attacks made a difference $\rightarrow$ profiling attacks

Example: Kyber



SCA attacks on Kyber

- SCA attack goals: msg recovery $\rightarrow$ secret-key recovery
- Attacks focus:
  - decapsulation step i.e. re-encryption step (encoding the key into a polynomial), tricky to mask, assuming chosen ciphertext attack
  - leakage in the Number-Theoretic Transform (NTT)
- Recent DL attacks broke a 6-shares implementation
- Countermeasures deemed very expensive

Screen Gleaning

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 m

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 m
- NSA published TEMPEST Fundamentals in 1982 referring to spying on systems through leaking emanations, including radio or el. signals, sounds and vibrations

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 m
- NSA published TEMPEST Fundamentals in 1982 referring to spying on systems through leaking emanations, including radio or el. signals, sounds and vibrations
- TEMPEST covers both methods to spy and to shield equipment against such spying

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 m
- NSA published TEMPEST Fundamentals in 1982 referring to spying on systems through leaking emanations, including radio or el. signals, sounds and vibrations
- TEMPEST covers both methods to spy and to shield equipment against such spying
- Van Eck phreaking: In 1985 published the first unclassified analysis of the security risks of emanations from computer monitors using just 15$ equipment+TV set

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 m
- NSA published TEMPEST Fundamentals in 1982 referring to spying on systems through leaking emanations, including radio or el. signals, sounds and vibrations
- TEMPEST covers both methods to spy and to shield equipment against such spying
- Van Eck phreaking: In 1985 published the first unclassified analysis of the security risks of emanations from computer monitors using just 15$ equipment+TV set
- Van Eck phreaking was used to successfully compromise ballot secrecy for electronic voting in Brazil

TEMPEST: Cause and History

Oscillating electric currents create EM radiation in the RF range and those signal drive the video display of various screens.

- Bell Labs noted this vulnerability for teleprinter communications during WW II recovering 75% of the plaintext being processed from a distance of 24 *m*
- NSA published TEMPEST Fundamentals in 1982 referring to spying on systems through leaking emanations, including radio or el. signals, sounds and vibrations
- TEMPEST covers both methods to spy and to shield equipment against such spying
- Van Eck phreaking: In 1985 published the first unclassified analysis of the security risks of emanations from computer monitors using just 15$ equipment+TV set
- Van Eck phreaking was used to successfully compromise ballot secrecy for electronic voting in Brazil

Motivation:

- Using TEMPEST the adversaries can reconstruct the images displayed through leaking emanations

Motivation and Outcomes

Motivation:

- Using TEMPEST the adversaries can reconstruct the images displayed through leaking emanations
- TEMPEST attack is known for a long time but **no methodology** has been established to evaluate it **on mobile devices**

Motivation and Outcomes

Motivation:

- Using TEMPEST the adversaries can reconstruct the images displayed through leaking emanations
- TEMPEST attack is known for a long time but **no methodology** has been established to evaluate it **on mobile devices**

In this work we:

- Introduce Screen Gleaning, a new electromagnetic TEMPEST attack targeting mobile phones

Motivation:

- Using TEMPEST the adversaries can reconstruct the images displayed through leaking emanations
- TEMPEST attack is known for a long time but **no methodology** has been established to evaluate it **on mobile devices**

In this work we:

- Introduce Screen Gleaning, a new electromagnetic TEMPEST attack targeting mobile phones
- Demonstrate the attack and its portability to different targets using machine learning

Motivation and Outcomes

Motivation:

- Using TEMPEST the adversaries can reconstruct the images displayed through leaking emanations
- TEMPEST attack is known for a long time but **no methodology** has been established to evaluate it **on mobile devices**

In this work we:

- Introduce Screen Gleaning, a new electromagnetic TEMPEST attack targeting mobile phones
- Demonstrate the attack and its portability to different targets using machine learning
- Provide a testbed and parameterized attacker model for further research

Screen gleaning (Theory)

Screen gleaning (Practice)

The signal we observe is, in most cases, not interpretable to the human eye.

Attacker model: Motivating story and assumptions

Alice keeps her phone on a stack of magazines on her desk (face down) to block the visual line of sight to the screen. Eve has hidden an antenna under the top magazine to read the security code via electromagnetic emanations of the phone.

Attacker model: Motivating story and assumptions

Alice keeps her phone on a stack of magazines on her desk (face down) to block the visual line of sight to the screen. Eve has hidden an antenna under the top magazine to read the security code via electromagnetic emanations of the phone.

- The set of symbols displayed on the phone is finite and known (digits 0-9)

Attacker model: Motivating story and assumptions

> *Alice keeps her phone on a stack of magazines on her desk (face down) to block the visual line of sight to the screen. Eve has hidden an antenna under the top magazine to read the security code via electromagnetic emanations of the phone.*

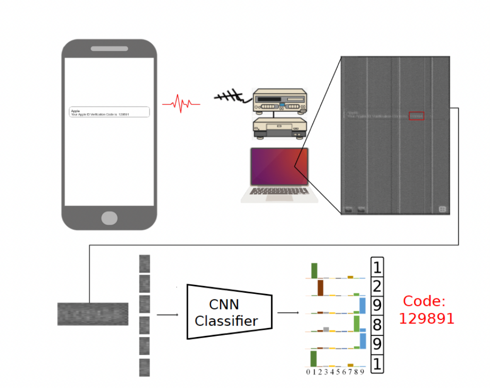
- The set of symbols displayed on the phone is finite and known (digits 0-9)
- The attacker has access to a profiling device that is “similar” to the target device

Attacker model: Motivating story and assumptions

Alice keeps her phone on a stack of magazines on her desk (face down) to block the visual line of sight to the screen. Eve has hidden an antenna under the top magazine to read the security code via electromagnetic emanations of the phone.

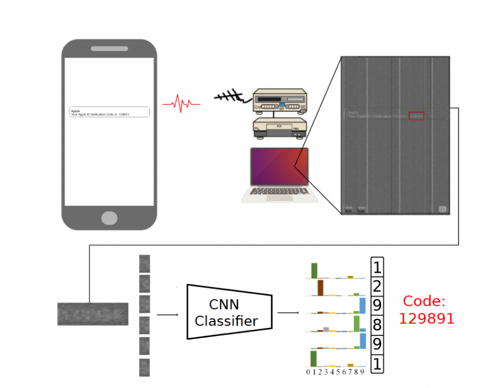
- The set of symbols displayed on the phone is finite and known (digits 0-9)
- The attacker has access to a profiling device that is "similar" to the target device
- The attacker can collect electromagnetic traces from the target device (representing the image displayed on the screen)

Attacker model: setup



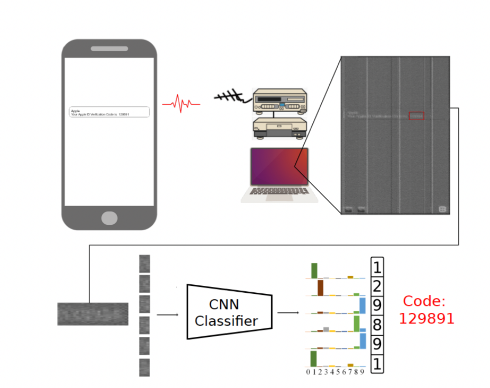
- The target emits EM signal intercepted by an antenna connected to a software-defined radio (SDR)

Attacker model: setup



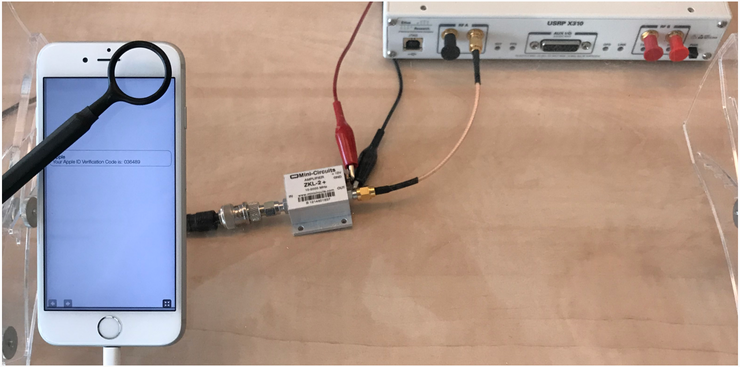
- The target emits EM signal intercepted by an antenna connected to a software-defined radio (SDR)
- The leaked information is collected and reconstructed as a gray-scale image (emage)

Attacker model: setup



- The target emits EM signal intercepted by an antenna connected to a software-defined radio (SDR)
- The leaked information is collected and reconstructed as a gray-scale image (emage)
- From emage, the 6-digit security code is cropped and fed into a CNN classifier for recognition

Screen gleaning setup



Security code results

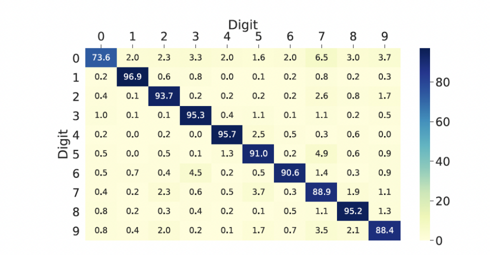


Figure: Confusion matrix of the inter-session accuracy of the security.

Digits	0	1	2	3	4	5	6	7	8	9	All
Acc. (%)	87.2	86.8	97.4	75.8	99.1	97.4	95.1	93.1	82.5	86.1	89.8

Table: Accuracy with respect to different digits (0-9) and overall accuracy in our security code attack.

	6 digits	$\geq$ 5 digits	$\geq$ 4 digits
Acc. (%)	50.5	89.5	99.0

Table: Accuracy of predicting partial security code correctly.

Security code results

- Attack on different phones of the same model
  E.g., cross-device accuracy of 61.5%, where the classifier is trained and tested on two distinct iPhone 6.
- Attack on different phone of different model
  E.g., accuracy of 74.0% on Huawei Honor 6X.
- Attack at a greater distance (through a magazine)
  E.g., accuracy of 65.8% on Huawei Honor 6X through 200 pages.

Z. Liu, Niels Samwel, L. Weissbart, Z. Zhao, D. Lauret, L. Batina, M. Larson, Screen Gleaning: A Screen Reading TEMPEST Attack on Mobile Devices Exploiting an Electromagnetic Side Channel, NDSS 2021.

Conclusions

- Screen gleaning is a new TEMPEST attack that uses an antenna and SDR to capture an electromagnetic side channel, i.e., emanations leaking from a mobile phone

Conclusions

- Screen gleaning is a new TEMPEST attack that uses an antenna and SDR to capture an electromagnetic side channel, i.e., emanations leaking from a mobile phone
- We demonstrated the effectiveness of it on three different phones with an example of the recovery of a security code

Conclusions

- Screen gleaning is a new TEMPEST attack that uses an antenna and SDR to capture an electromagnetic side channel, i.e., emanations leaking from a mobile phone
- We demonstrated the effectiveness of it on three different phones with an example of the recovery of a security code
- We introduced 5-dimension attacker model that can be extended further

Conclusions

- Screen gleaning is a new TEMPEST attack that uses an antenna and SDR to capture an electromagnetic side channel, i.e., emanations leaking from a mobile phone
- We demonstrated the effectiveness of it on three different phones with an example of the recovery of a security code
- We introduced 5-dimension attacker model that can be extended further
- We proposed a testbed providing a standard setup in which screen gleaning can be tested further with different attacker models

Reverse Engineering of NN Architectures Through SCA

- Well-trained models are valuable for certain industries

Motivation for SCA to reverse engineer NNs

- Well-trained models are valuable for certain industries
- In some cases hyper-parameters and other training details are considered IP

Motivation for SCA to reverse engineer NNs

- Well-trained models are valuable for certain industries
- In some cases hyper-parameters and other training details are considered IP
- By 2024, the number of edge-based AI chips will be doubled

- Well-trained models are valuable for certain industries
- In some cases hyper-parameters and other training details are considered IP
- By 2024, the number of edge-based AI chips will be doubled
- Neural nets are being deployed on various platforms on low-power processors for always-on systems e.g. ARM Cortex-M microcontrollers, FPGAs, GPUs etc.

Motivation for SCA to reverse engineer NNs

- Well-trained models are valuable for certain industries
- In some cases hyper-parameters and other training details are considered IP
- By 2024, the number of edge-based AI chips will be doubled
- Neural nets are being deployed on various platforms on low-power processors for always-on systems e.g. ARM Cortex-M microcontrollers, FPGAs, GPUs etc.
- Implementations on those platforms are common targets for side-channel adversaries

Threat model and attacker's capabilities

Goal: Recover the neural network architecture using only side-channel information

Threat model and attacker's capabilities

Goal: Recover the neural network architecture using only side-channel information
Target: Pretrained neural network model executed on an embedded device while running inference

Threat model and attacker's capabilities

Goal: Recover the neural network architecture using only side-channel information
Target: Pretrained neural network model executed on an embedded device while running inference
Fact 1: Implementations of ML algorithm are not protected against SCA

Goal: Recover the neural network architecture using only side-channel information

Target: Pretrained neural network model executed on an embedded device while running inference

Fact 1: Implementations of ML algorithm are not protected against SCA

Fact 2: This approach does not need access to training data

Threat model and attacker's capabilities

Goal: Recover the neural network architecture using only side-channel information
Target: Pretrained neural network model executed on an embedded device while running inference
Fact 1: Implementations of ML algorithm are not protected against SCA
Fact 2: This approach does not need access to training data

Threat model:

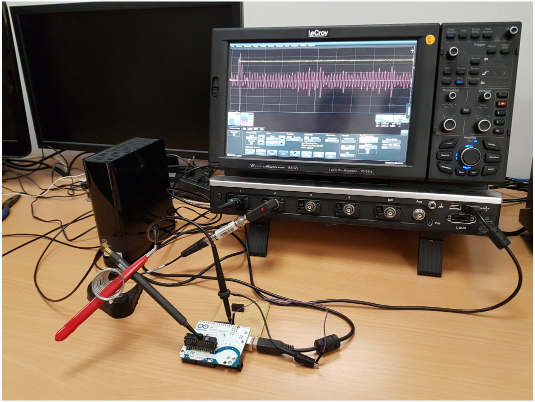
- Adversary can query the model with known/chosen inputs and passively observe side-channel information corresponding to the executed inference

Threat model and attacker's capabilities

Goal: Recover the neural network architecture using only side-channel information
Target: Pretrained neural network model executed on an embedded device while running inference
Fact 1: Implementations of ML algorithm are not protected against SCA
Fact 2: This approach does not need access to training data

Threat model:

- Adversary can query the model with known/chosen inputs and passively observe side-channel information corresponding to the executed inference
- No specific assumption on the type of inputs or its source, as we work with real numbers


LeCroy

The adversary feeds known random inputs in a form of floating point real numbers and observes side channels.

The leakage model used is Hamming weight (HW).

The attacker wants to learn:

The adversary feeds known random inputs in a form of floating point real numbers and observes side channels.

The leakage model used is Hamming weight (HW).

The attacker wants to learn:

- Information about layers

SCA analysis

The adversary feeds known random inputs in a form of floating point real numbers and observes side channels.

The leakage model used is Hamming weight (HW).

The attacker wants to learn:

- Information about layers
- Information about neurons

SCA analysis

The adversary feeds known random inputs in a form of floating point real numbers and observes side channels.

The leakage model used is Hamming weight (HW).

The attacker wants to learn:

- Information about layers
- Information about neurons
- Information about activation functions

SCA analysis

The adversary feeds known random inputs in a form of floating point real numbers and observes side channels.

The leakage model used is Hamming weight (HW).

The attacker wants to learn:

- Information about layers
- Information about neurons
- Information about activation functions
- Information about weights

Activation functions

- Function f of a node defining the output of a node given an input or set of inputs

Activation functions

- Function f of a node defining the output of a node given an input or set of inputs

$$y = Activation(\sum(weight \cdot input) + bias). \tag{1}$$

Activation functions

- Function f of a node defining the output of a node given an input or set of inputs

$$y = Activation(\sum(weight \cdot input) + bias). \tag{1}$$

- Examples: Sigmoid, tanh, softmax, ReLU

Reverse engineering activation functions

Timing behavior can be observed directly from the EM trace

Reverse engineering activation functions

Timing behavior can be observed directly from the EM trace

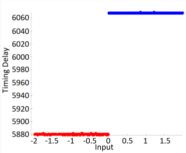
Table: Minimum, Maximum, and Mean computation time (in *ns*)

Activation Function	Minimum	Maximum	Mean
ReLU	5 879	6 069	5 975
Sigmoid	152 155	222 102	189 144
Tanh	51 909	210 663	184 864
Softmax	724 366	877 194	813 712

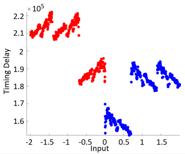
Timing behavior can be observed directly from the EM trace

Table: **Minimum, Maximum, and Mean computation time (in *ns*)**

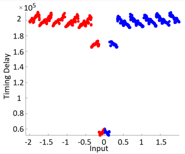
Activation Function	Minimum	Maximum	Mean
ReLU	5 879	6 069	5 975
Sigmoid	152 155	222 102	189 144
Tanh	51 909	210 663	184 864
Softmax	724 366	877 194	813 712



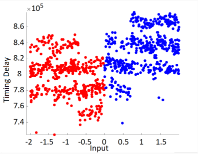
(a) **ReLU**



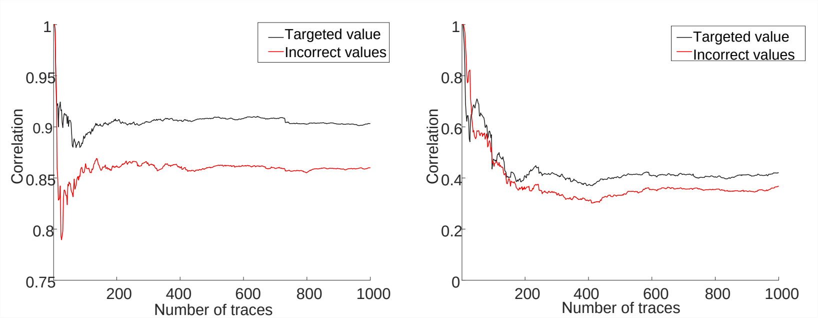
(b) **Sigmoid**



(c) **Tanh**

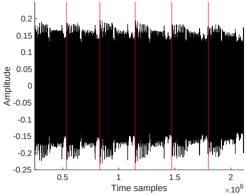


(d) **Softmax**

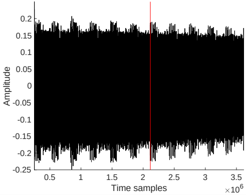


(a) First byte recovery (sign and 7-bit exponent) **(b) Second byte recovery (lsb exponent and mantissa)**

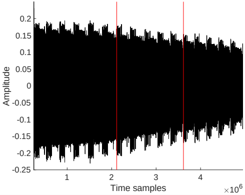
SEMA on hidden layers



(a) One hidden layer with 6 neurons



(b) 2 hidden layers (6 and 5 neurons each)



(c) 3 hidden layers (6,5,5 neurons each)

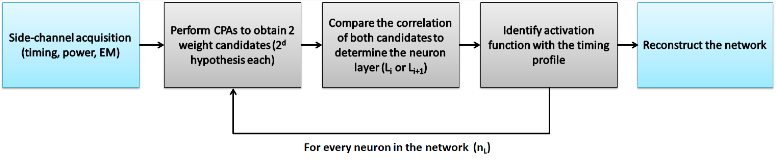


Figure: Methodology to reverse engineer the target neural network

ARM Cortex M-3 and MLP

- Tests with MNIST and DPAv4 datasets

ARM Cortex M-3 and MLP

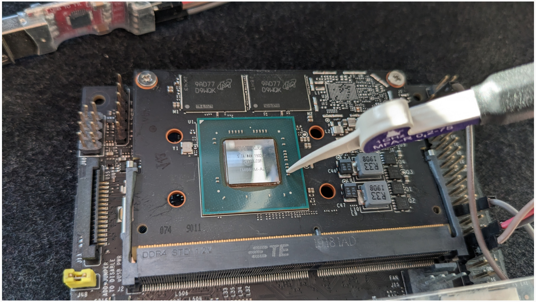
- Tests with MNIST and DPAv4 datasets
- DPAv4: the original accuracy equals 60.9% and the accuracy of the reverse engineered network is 60.87%

ARM Cortex M-3 and MLP

- Tests with MNIST and DPAv4 datasets
- DPAv4: the original accuracy equals 60.9% and the accuracy of the reverse engineered network is 60.87%
- MNIST: the accuracy of the original network is equal to 98.16% and the accuracy of the reverse engineered network equals 98.15%, with an average weight error converging to 0.0025

Lejla Batina, Shivam Bhasin, Dirmanto Jap, Stjepan Picek: CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. USENIX Security Symposium 2019: 515-532.

Recent work: reverse engineering commercial GPUs



- Architecture recovery from NVIDIA Jetson Nano device with 128-core GPU
- Weights recovery
- Known input assumption

- The context in which a cryptosystem is used defines the adversarial model

Lessons learned

- The context in which a cryptosystem is used defines the adversarial model
- “Provably” secure implementations are regularly broken

Lessons learned

- The context in which a cryptosystem is used defines the adversarial model
- "Provably" secure implementations are regularly broken
- But we should not give up on theory
- AI-assisted SCA attacks are more powerful in some use cases
- But, in many SCA evaluations "classical" techniques could be more efficient

Lessons learned

- The context in which a cryptosystem is used defines the adversarial model
- "Provably" secure implementations are regularly broken
- But we should not give up on theory
- AI-assisted SCA attacks are more powerful in some use cases
- But, in many SCA evaluations "classical" techniques could be more efficient
- SCA and AI are getting more and more intertwined

Thank you for your attention!

https://cescalab.cs.ru.nl/