Differential Power Analysis (DPA) With Key Ranking and Enumeration

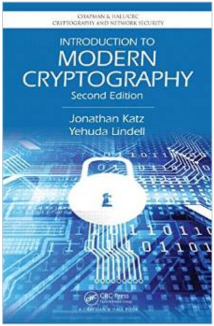
Martijn Stam

Simula
UiB

COINS Winterschool in Finse, May 2019

Modern Cryptology

From Katz and Lindell’s Classic Textbook

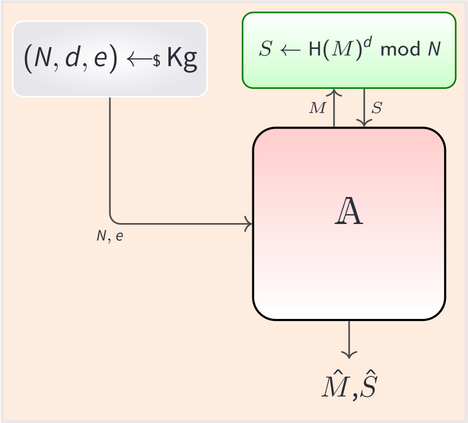


Three Principles

1. Formal Definitions: “giving a clear description of what threats are in scope and what security guarantees are desired”
2. Precise Assumptions: “that are simpler to state, since [they] are easier to study and (potentially) refute”
3. Proofs of Security: “that a construction satisfies a definition under certain specified assumptions”

FDH-RSA Cryptosystem

Black-box perspective of chosen-message attacks



$$\mathsf{Adv}^{\text{euf-cma}}_{\text{FDH-RSA}}(\mathbb{A}) = \Pr\left[\hat{S}^e \bmod N = \mathsf{H}(\hat{M}),\ \hat{M} \text{ fresh}\right]$$

The Rise of Side-Channels

Paul Kocher's Revolution

https://www.paulkocher.com/

1996	Timing Attacks
1999	Simple and Differential Power Analysis (DPA) (w. Joshua Jaffe & Benjamin Jun)
2016	https://www.youtube.com/watch?v=6lt7ExN6Kw4

Power Analysis

Measuring power consumption over time allows (relatively) easy *recovery* of secret *keys*

SPA: Simple Power Analysis

A Simple Attack Against Unprotected RSA

What is SPA

SPA exploits data-dependent *differences* in power consumption of a *single* operation to recover secret information.

$S \leftarrow \mathrm{H}(m)^d \bmod N$

$s \leftarrow 1, x \leftarrow \mathrm{H}(m)$
while $d > 0$
  if d odd **then**
    $s \leftarrow s \cdot x \bmod N$
  $x \leftarrow x^2 \bmod N$
  $d \leftarrow \lfloor d/2 \rfloor$

Simple Attack

- Assume you can tell multiplications and squarings apart.
- So you observe something like *SMSSMSSM*
- Corresponds to exponent $(10101)_2 = 21$

DPA: Differential Power Analysis

The workhorse of side-channel attacks

What is DPA

DPA exploits data-dependent *correlation* in power consumption over *multiple, related* operations to recover secret information.

Power of DPA

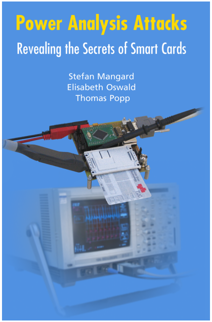
Any unprotected implementation will eventually be susceptible.

Countermeasures

All implementations will need *protection* against side channels.

Power Analysis Attacks

Stefan Mangard, Elisabeth Oswald, and Thomas Popp's Classic



Revealing the Secrets of Smart Cards

- “first comprehensive treatment of power analysis attacks and countermeasures”
- Aimed at the practitioner
- From 2007 $\Rightarrow$ no modern ideas and theory

Outline

1. **How Differential Power Attacks Work**
   - Our Setting
   - A Typical Pipeline for Key Recovery
   - Profiled Attack Example
2. **Key Enumeration and Ranking**
   - Enumeration
   - Ranking
3. **Conclusion**
   - Want to Learn More?

Modern Cryptology

Black-box Blockciphers

What is a Blockcipher

A blockcipher E is family of keyed permutations

$$\mathrm{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

where k is the key length and n the block length

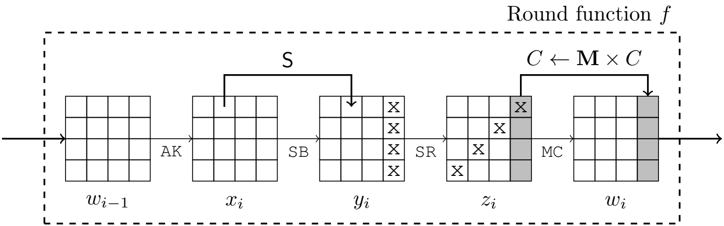
Blockcipher Usage

- Use a mode-of-operation like GCM to create an encryption scheme
- GCM security proof assumes the blockcipher E is a “PRP”
- So E is treated as a *black box*
- What happens if you can see it “work”?

Modern Cryptology

AES-128

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15



Design

- $k = 128$, $n = 128$, where $128 = 16 \times 8$ (16 bytes)
- 10 rounds of whitened SP network
- Non-linearity comes from bytewise S-boxes

Images: TikZ for Cryptographers, Jérémy Jean, www.iacr.org/authors/tikz/

SCALE: A Resource by Dan Page

https://github.com/danpage/scale

Side-Channel Attack Lab. Exercises

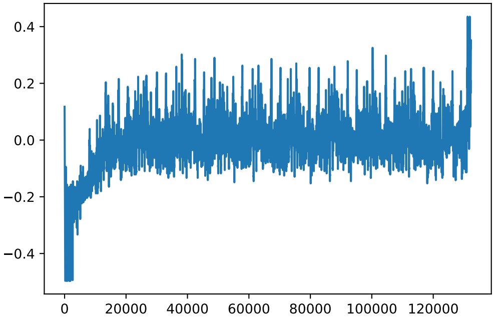
Provides a suite of material related to side-channel (and fault) attacks that is low-cost, accessible, relevant, coherent, and effective.

SCALE Data Sets

1. Four platforms: an *Atmel atmega328p* (an AVR) plus three NXP ARM Cortex-M processors
2. Implementation uses an 8-bit datapath and look-up tables for the S-box and xtime operations (but code not known)
3. 2×1000 traces of AES-128 each (known vs. unknown key)
4. Traces acquired using a Picoscope 2206B, using triggers for alignment

Plotting a Trace

SCALE's AES-128 on an Atmel

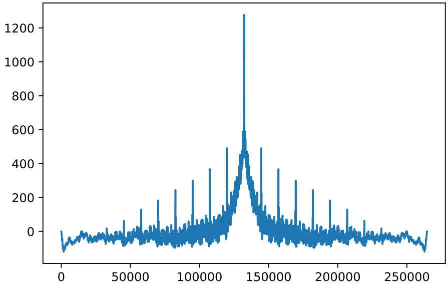


A full trace $k = \text{2B7E151628AED2A6ABF7158809CF4F3C}$

- Total of 132,292 points
- You can see a pattern repeating roughly 10 times

Finding the Rounds

Using crosscorrelation



Crosscorrelation of a trace

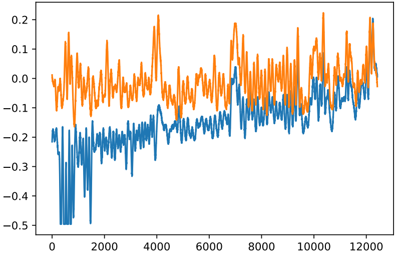
Compares how well shifts of the trace match the original

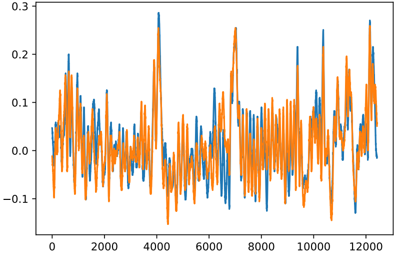
$$c_i = \sum_j a_j a_{i+j}$$

Leads to round duration of 12421

Finding the Rounds

Using crosscorrelation





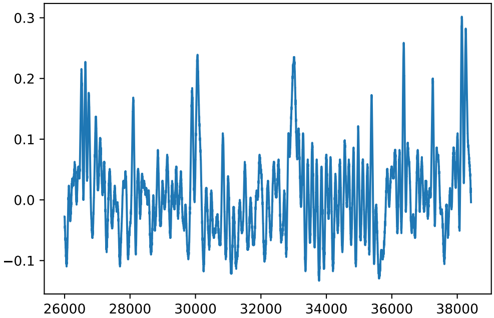
Plotting the Rounds Jointly

Left Rounds 1 and 2 superimposed
Round 1 is building up power

Right Rounds 5 and 6 superimposed
Peaks and jittery areas match well

Plotting a Trace

SCALE’s AES-128 on an Atmel

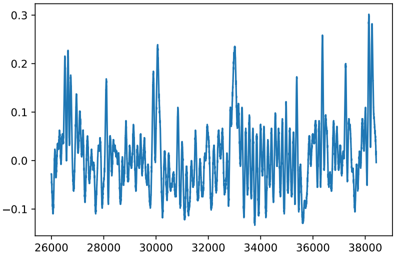


3rd round close up

1. Some peaks, some jitter
2. Hard to really discern much of interest...

Signal versus Noise

What determines the power consumption?



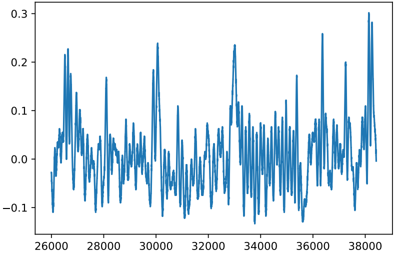
Engineer's Perspective (MOP, Ch. 4)

$$P_{total} = P_{op} + P_{data} + P_{el.\ noise} + P_{const}$$

- P_{op} d.o. the operation
- P_{data} d.o. the data
- $P_{el.\ noise}$ electrical noise
- P_{const} constant base

Signal versus Noise

What determines the power consumption?



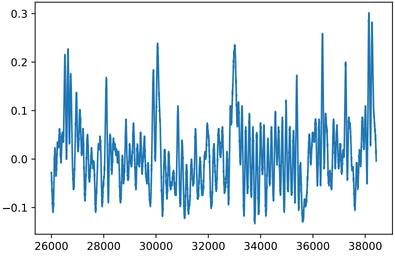
Engineer's Perspective (MOP, Ch. 4)

$$P_{op} + P_{data} = P_{exp} + P_{sw.\ noise}$$

- P_{op} d.o. the operation
- P_{data} d.o. the data
- P_{exp} exploitable signal
- $P_{sw.\ noise}$ switching noise

Signal versus Noise

What determines the power consumption?



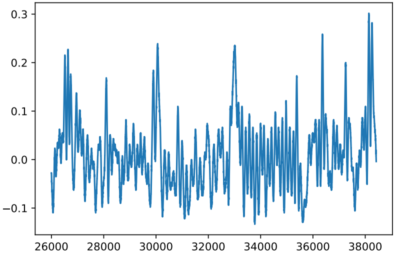
Engineer's Perspective (MOP, Ch. 4)

$$P_{total} = P_{exp} + P_{sw.\ noise} + P_{el.\ noise} + P_{const}$$

- $P_{el.\ noise}$ electrical noise
- P_{const} constant base
- P_{exp} exploitable signal
- $P_{sw.\ noise}$ switching noise

Signal versus Noise

What determines the power consumption?



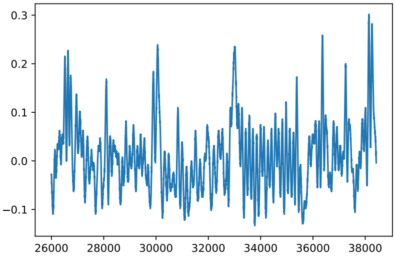
Theoretician's Perspective

$$P_{total} = f(data) + \mathcal{N}(0, \sigma)$$

- $f(data)$ mainly models P_{exp}, function f incorporates P_{op} and P_{const}
- σ depends on $P_{sw.\ noise}$ and $P_{el.\ noise}$

Signal versus Noise

What determines the power consumption?

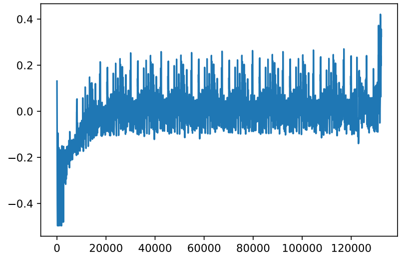


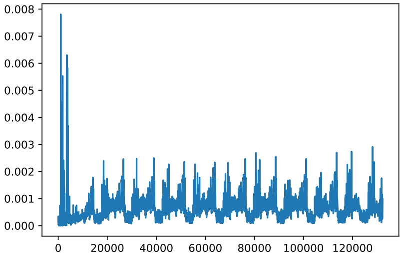
Some Caveats

1. Which operations are performed on which registers can be relevant
2. Looking at multiple points might lead to multivariate dependencies
3. Sometimes noise levels (σ) are data-dependent
4. The function f and noise level σ are unknown

Signal versus Noise

What determines the power consumption?





Atmel AES, Based on 1000 traces

Assuming no branches in the execution

Left Pointwise sample mean: $P_{const} + P_{op}$

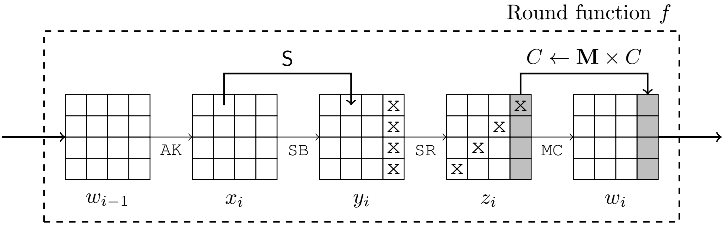
Right Pointwise sample variance: $P_{data} + P_{el.\ noise}$

Both P_{exp} and $P_{sw.\ noise}$ depend on your target...

Signal versus Noise

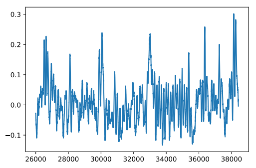
Intermediate values and target selection

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15



The Locality of Leakage

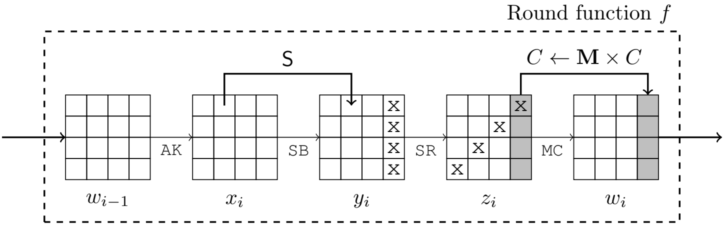
- *Intermediate value*:
  the (few) byte(s) involved in a specific operation
- Locality assumption:
  leakage primarily depends on the intermediate value operated upon



Signal versus Noise

Intermediate values and target selection

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

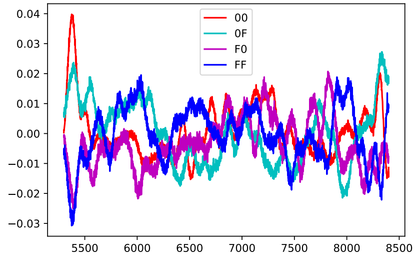


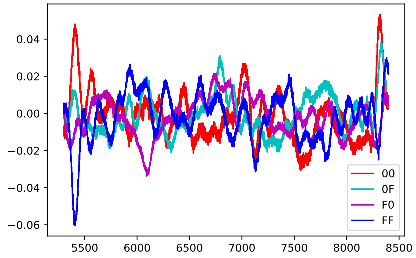
The Locality of Leakage

- *Intermediate value*: the (few) byte(s) involved in a specific operation
- Locality assumption: leakage primarily depends on the intermediate value operated upon
- *Target* intermediate value captured by P_{exp}
- The "rest" contributes to $P_{sw.\ noise}$

Signal versus Noise

Intermediate values and target selection





First Round, Byte Pos. “0”, keybyte *2B*

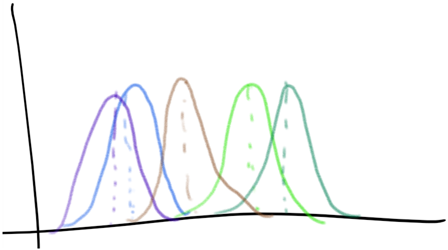
Left Average leakage based on select plaintext values

Right Average leakage based on select sbox inputs

- Initial peak correlates more with plaintext
- Final peak correlates more with sbox input

SNR: Signal to Noise Ratio

Visualizing “Simple” Leakage

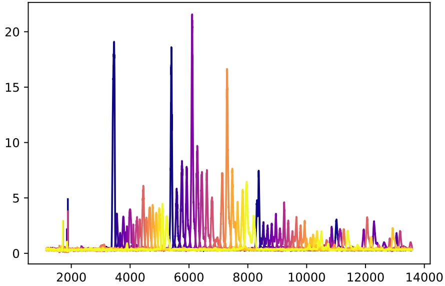


Mangard’s SNR

- Recall we said $P_{total} = f(data) + \mathcal{N}(0, \sigma)$
- $f(data)$ is called the *signal*, the other term the *noise*
- $$SNR = \frac{Var(signal)}{Var(noise)} = \frac{Var_{data} f}{\sigma^2}$$

SNR: Signal to Noise Ratio

Visualizing “Simple” Leakage

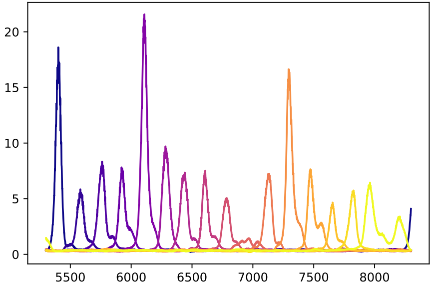


Round 1 SNRs, sample estimate

- 16 Sbox inputs as separate targets
- Fixed key, so equivalent to 16 plaintext bytes
- Clearly see the different bytes leak repeatedly, one after the other

SNR: Signal to Noise Ratio

Visualizing “Simple” Leakage



Round 1 SNRs, zoom in

- Clearly see the different bytes leak repeatedly, one after the other
- Peaks differ in height
- At the bases consecutive bytes leak jointly

Hamming Weight and Hamming Distance

Two Common Leakage Models

Hamming Weight

Power consumption is linear in the Hamming weight of the target data

$$f(data) = a \cdot HW(data) + b$$

Correspond to model where power depends primarily on “setting” bits

Hamming Distance

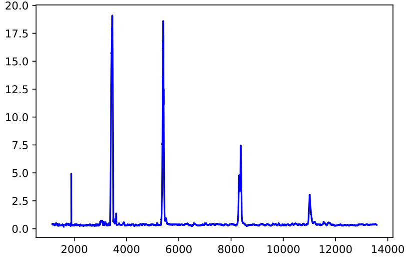
Power consumption is linear in the Hamming distance of the target data input with the output

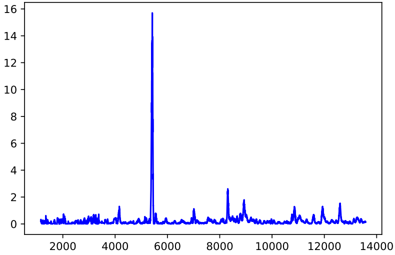
$$f(data) = a \cdot HD(data_{in}, data_{out}) + b$$

Correspond to model where power depends primarily on “flipping” bits

SNR: Signal to Noise Ratio

AES Atmel Hamming Leakage





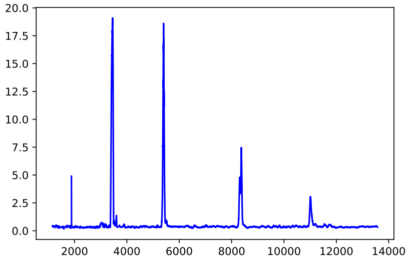
Round 1, "0" byte SNR vs *Sbox input* Hamming weight

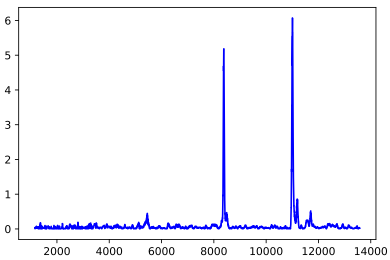
plaintext Hamming weight (right)

- Explains the third SNR peak (left)
- No non-linearity ⇒ hard to exploit

SNR: Signal to Noise Ratio

AES Atmel Hamming Leakage





Round 1, “0” byte SNR vs *Sbox output* Hamming weight

Sbox input Hamming weight (right)

- Explains the final two SNR peaks (left)
- Non-linearity $\Rightarrow$ exploitable for key recovery

Kerckhoffs Principle

Known Knowns and Unkown Unknowns

Kerckhoffs's Principle

A cryptosystem's security should

- reside in the the secrecy of its keys (known unknown)
- without any need to keep the cryptosystem secure (known known)

What about Implementations

- What device is being used?
- Which cryptosystem is implemented how?
- Auxiliary inputs (plaintexts/ciphertexts/randomness)?
- But what about the leakage such as power consumption?

Kerckhoffs Principle

Known Knowns and Unkown Unknowns

Kerckhoffs's Principle

A cryptosystem's security should

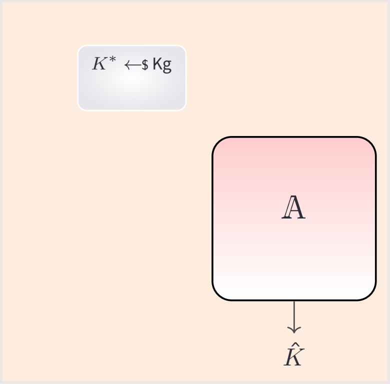
- reside in the the secrecy of its keys (known unknown)
- without any need to keep the cryptosystem secure (known known)

What about Power Consumption?

Realistically, *even* if you know what operations are being performed, *how* a device leaks is too unpredictable (unknown unknown).
Not-Quite-Kerckhoff Principle

Different Adversarial Scenarios

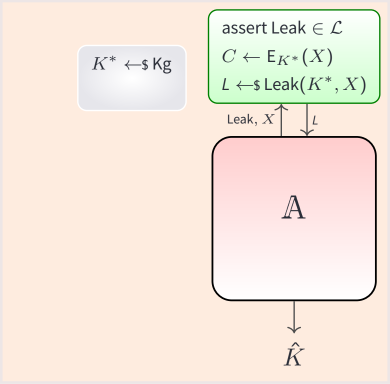
Not-Quite-Kerckhoffs Principle



The naked guess-the-key game: $\mathsf{Adv}^{\mathrm{kr}}_{\mathrm{E}}(\mathbb{A}) = \Pr\left[K^* = \hat{K}\right]$

Different Adversarial Scenarios

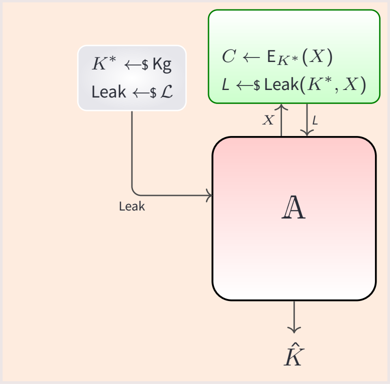
Not-Quite-Kerckhoffs Principle



The adversary selects how the leakage is derived

Different Adversarial Scenarios

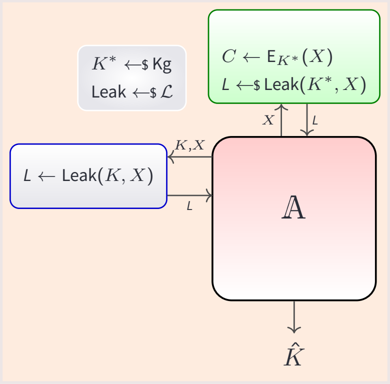
Not-Quite-Kerckhoffs Principle



The adversary knows exactly how to model the leakage

Different Adversarial Scenarios

Not-Quite-Kerckhoffs Principle

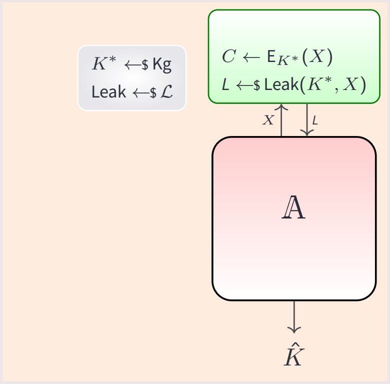


The adversary learns how the leakage profile $\hat{\theta}$ looks:

$$\mathsf{Leak}(data) \approx M_{\hat{\theta}}(data)$$

Different Adversarial Scenarios

Not-Quite-Kerckhoffs Principle



The adversary infers a leakage model M:

$$\mathsf{Leak}(data) \approx M(data)$$

Different Adversarial Scenarios

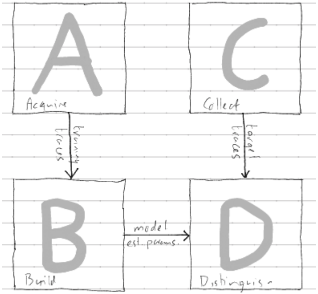
Not-Quite-Kerckhoffs Principle

Different Scenarios

1. The adversary selects how the leakage is derived
   includes *leakage-resilience* and formal *probing* models
2. The adversary knows exactly how to model the leakage
   used for *simulated* leakage models
3. The adversary learns how the leakage profile looks
   captures real-life *profiled* attacks
4. The adversary infers a leakage model
   captures real-life attacks *without profiling*

Caveat: “Stronger” models (higher in the list) tend to be relative to less realistic and potentially “weaker” forms of leakage

A Typical Side-Channel Attack Pipeline



A *Acquire training data*

- control over keys and plaintexts
- signal processing to clean up traces

B *Build a profile*

1. select features or PoIs
2. fix model M, estimate parameters $\hat{\theta}$

C *Collect target traces*

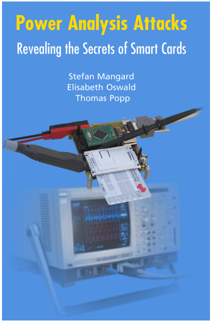
- unknown target key, known plaintexts
- use signal processing as before

D Distinguish

1. extract features or PoIs
2. using model M and parameters $\hat{\theta}$,
   for *each* key candidate,
   compute *distinguishing score*

A Typical Side-Channel Attack Pipeline

Acquisition and Collection



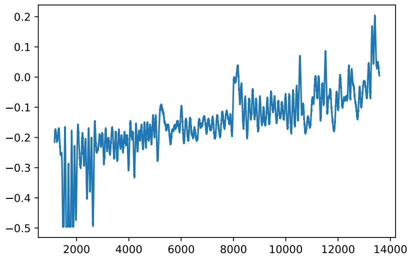
Experimental setup

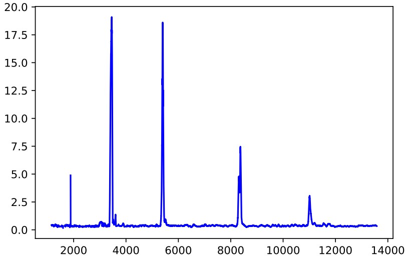
- Use oscilloscope to measure power
- (Optional) Use triggers to align data
- Use signal processing to clean up raw trace

$\Leftarrow$ see the textbook for details!

A Typical Side-Channel Attack Pipeline

Feature Extraction and Points of Interest





Where does a target intermediate leak?

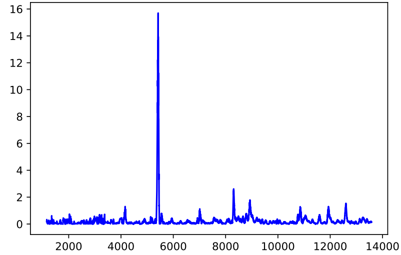
An intermediate leaks mostly where it is being used

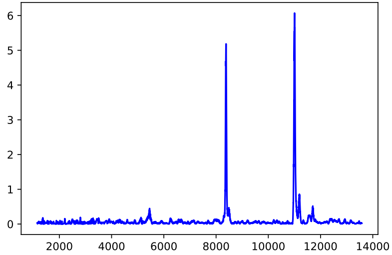
- Good to identify where this is
- Then reduce dimension of interesting points if possible

Easiest is to select a *point of interest*

A Typical Side-Channel Attack Pipeline

Build a Model, Profiling





How does a target intermediate leak?

- Assume Leak($data$) $= M_\theta(data)$ for known model M
- Estimate the “real” parameters θ by $\hat{\theta}$

A Typical Side-Channel Attack Pipeline

Build a Model, Profiling

Lessons from Machine Learning

Suppose the real leakage follows data-dependent distribution Leak($data$)

1. Assume that unknown Leak($data$) follows known *model M* with unknown parameters θ

$$\text{Leak}(data) \approx M_\theta(data)$$

2. Estimate the "real" parameters θ by $\hat{\theta}$

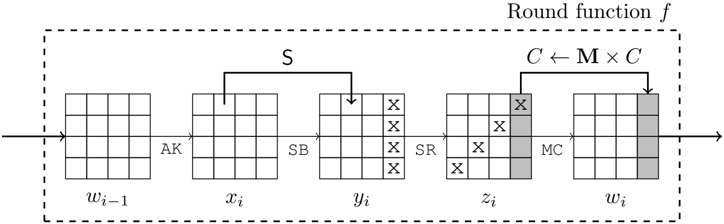
$$M_\theta(data) \approx M_{\hat{\theta}}(data)$$

Warning: More complex models have *smaller* modelling errors (first point) at the expense of *larger* estimation errors (second point)

A Typical Side-Channel Attack Pipeline

Distinguish using Divide-and-Conquer

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15



Divide-and-Conquer

Idea: Recover each subkey byte separately

- For all 256 candidates, calculate a *distinguishing score*
- The lowest (or highest) score indicates the likely true subkey byte

Guess all 16 subkey bytes correctly $\Leftrightarrow$ guess the AES key correctly

A Typical Side-Channel Attack Pipeline

Distinguish using Divide-and-Conquer

k_0	score
0	0.123...
1	0.127...
$\vdots$	$\vdots$
255	0.238...

k_1	score
0	0.134...
1	0.116...
$\vdots$	$\vdots$
255	0.098...

...

k_{15}	score
0	0.184...
1	0.167...
$\vdots$	$\vdots$
255	0.152...

Divide-and-Conquer

Idea: Recover each subkey byte separately

- For all 256 candidates, calculate a *distinguishing score*
- The lowest (or highest) score indicates the likely true subkey byte

Guess all 16 subkey bytes correctly $\Leftrightarrow$ guess the AES key correctly

A Typical Side-Channel Attack Pipeline

Distinguish using Divide-and-Conquer

k_0	score
0	0.123...
1	0.127...
$\vdots$	$\vdots$
255	0.238...

k_1	score
0	0.134...
1	0.116...
$\vdots$	$\vdots$
255	0.098...

...

k_{15}	score
0	0.184...
1	0.167...
$\vdots$	$\vdots$
255	0.152...

Distinguishing Scores using Leak$(data) \approx M_{\hat{\theta}}(data)$

1. Assume data relevant for leakage only depends on one subkey (easiest to attack AES first or last round)
2. For each 256 possibilities and each trace, calculate how the modelled leakage would look
3. Compare modelled leakage with observed trace, combine into score