H2020 project 731591
http://www.reassure.eu/

Understanding Leakage Detection

Part 2: Detection on Real-world Traces

Simulated leakage vs Real leakage

- **Simulated leakage** is useful to understand problems and provide intuitions, since it can be fully controllable:
  - It can be constructed with given noise level and distribution
  - It can be **univariate** or **multivariate**
- **Real leakage** comes from a **real device**, that may or may not be controllable (input plaintext choice, key, operating frequency, etc.)
  - We can only have choice on how to probe its power consumption
  - We cannot *really* control the noise level
  - We cannot control the nature of the leakage (as it depends on the implementation and/or the measurement setup)

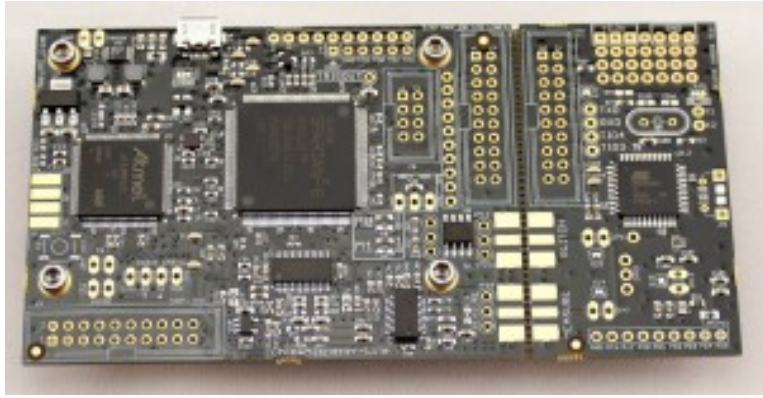
Simulated leakage vs Real leakage

- A key consideration to take into account is that with **real leakage**, we may need to understand the role of each samples
- The problem of detection (as well as the actual attack) can be very complex since there are multiple time samples to consider
  - Each time sample is considered as a **variable/dimension**
  - In case of countermeasures (e.g. masking) the information can be jointly spread over multiple samples → **multivariate problem**
- The computational complexity of the detection can be very higher due to reasons listed above

SIM. LEAKAGE ≠ REAL LEAKAGE

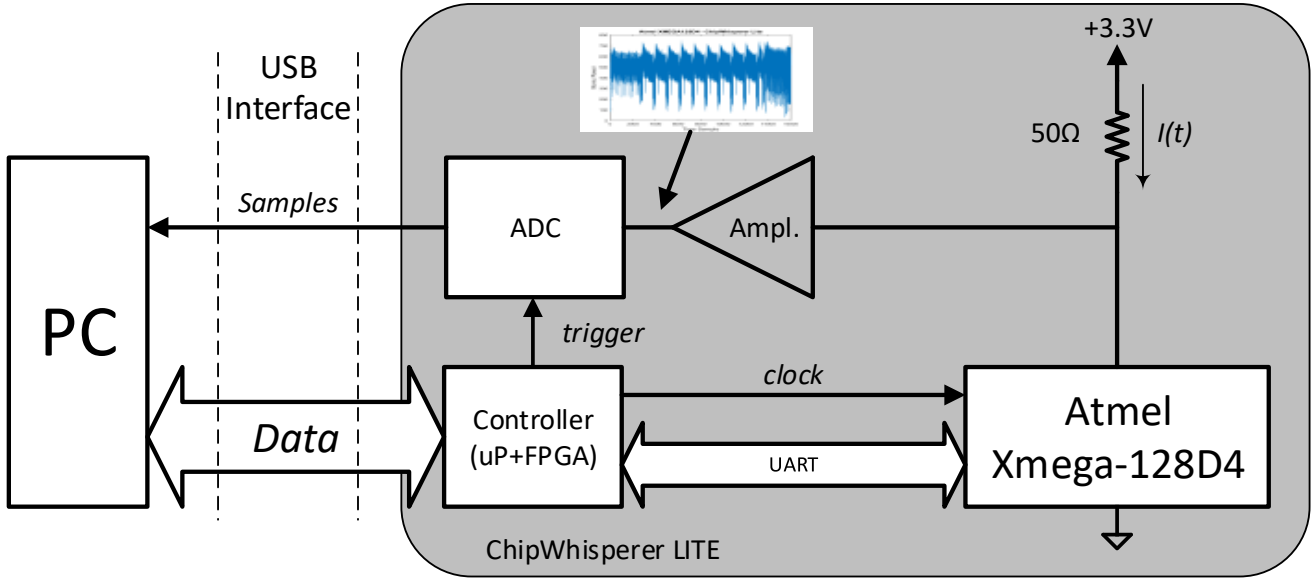
Data-collection of real power traces

To investigate further the meaning of the lesson learned in the previous part, we will use **real power traces** collected on an unprotected AES-128 software implementation (AESFurious by B. Poettering) on the popular low-cost platform ChipWhisperer-Lite from NewAE Technology, widely used for side-channel experiments.



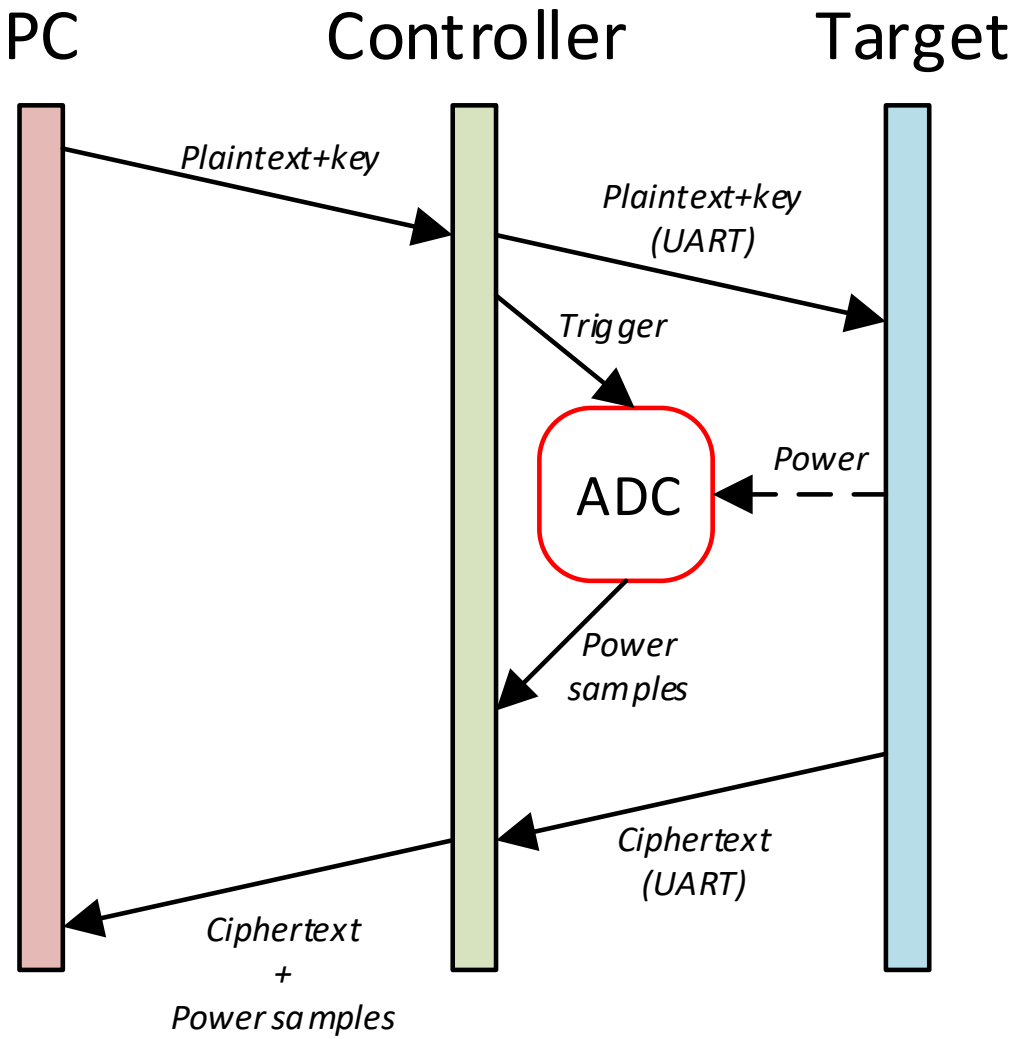
Source: https://newae.com/tools/chipwhisperer/

$$f_{clk} = 7{,}38MHz\,;SR = 29{,}54\,MS/s$$
$$Sample/Clock_{cycle} = 4$$



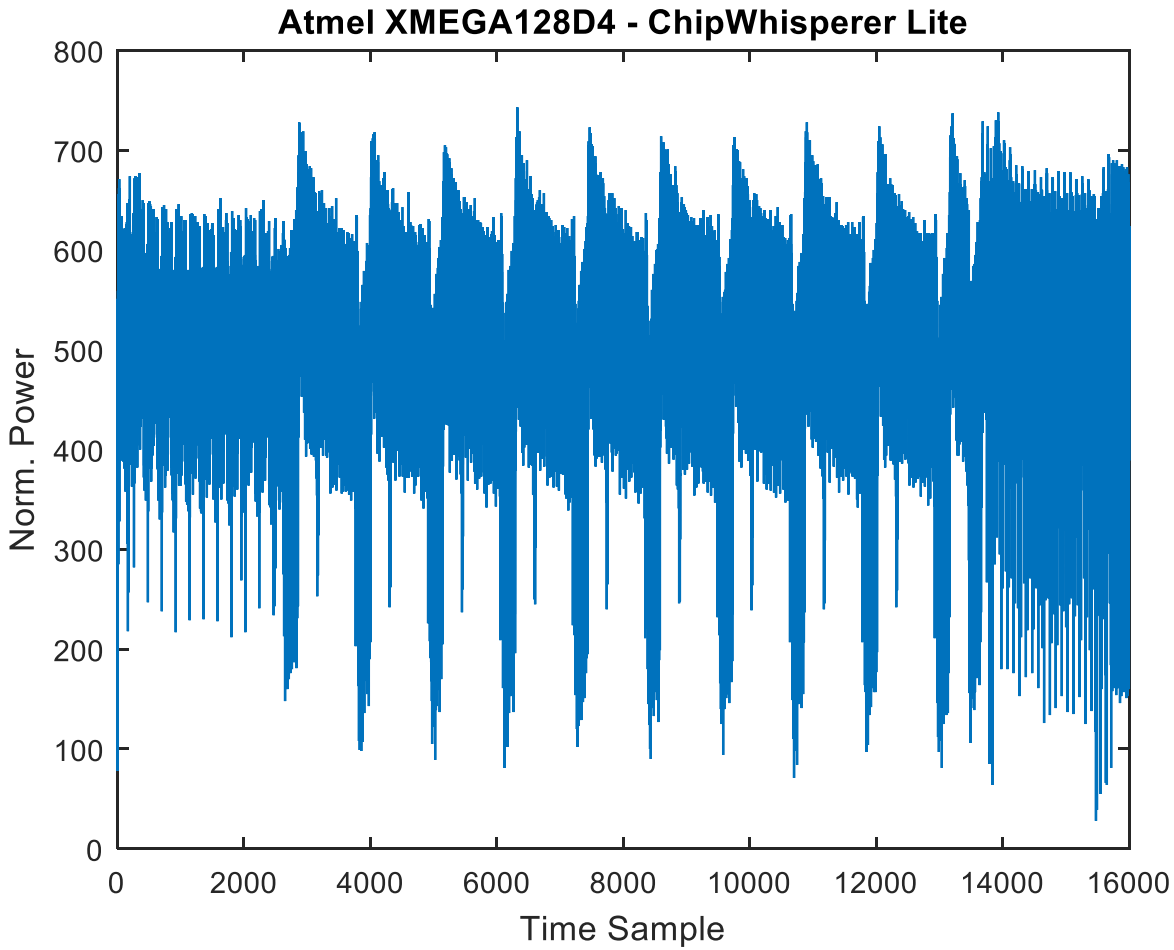
How we collected data

- Simple paradigm:
  - PC sends plaintext (and key)
  - The CW's controller forwards the plaintext (and key) to the target throught UART connection
  - The CW's controller triggers the ADC to start to sample the power consumption of the target on its power supply line
  - The target encrypt the plaintext and sends back the ciphertext to the controller
  - The power samples and the ciphertext are sent back to the PC



Anatomy of a power trace…

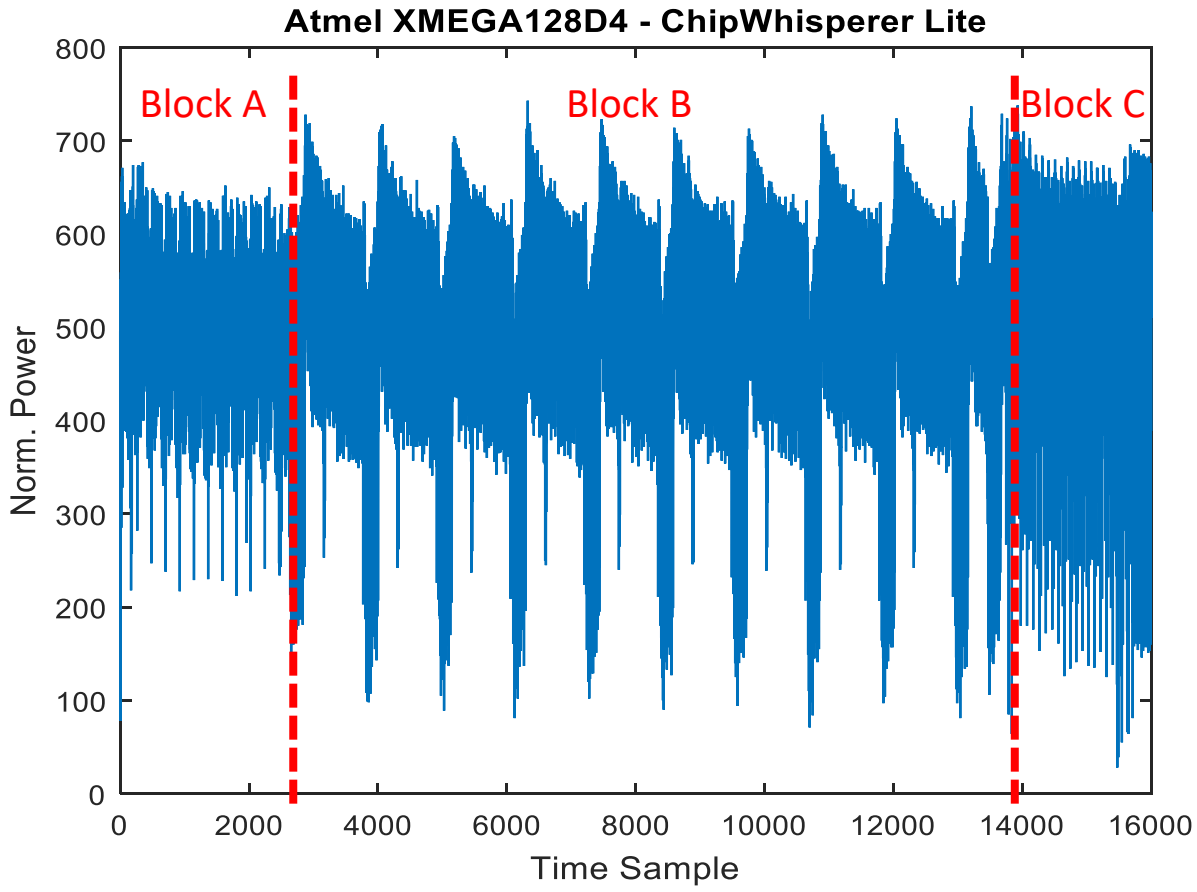
?



?

Anatomy of a power trace…

- We can observe three sections:
  - **block A**, from sample 0 to sample 2500;
  - **block B**, from sample 2500 to sample 14000;
  - **block C**, from sample 1400 to sample 16000.
- It is easy to recognize ten AES-rounds in **block B** (*simple power analysis, SPA*)



TVLA framework definitions

Let’s introduce some formal definitions:

- **Non-specific (general) test:** it aims to detect any leakage that depends on input data (or key).

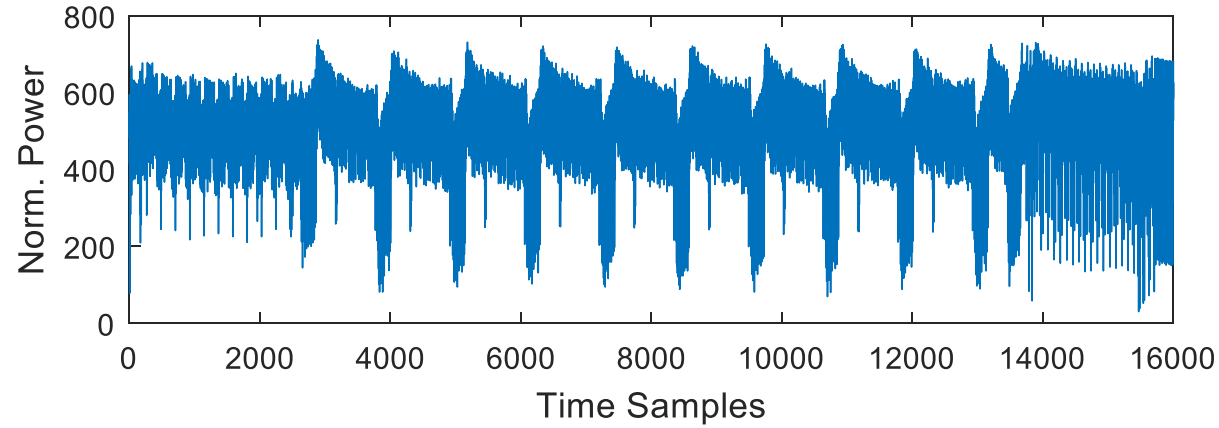
- **Specific test:** it tests target specific intermediate values of the cryptographic algorithm that could be exploited to recover keys or other sensitive information.

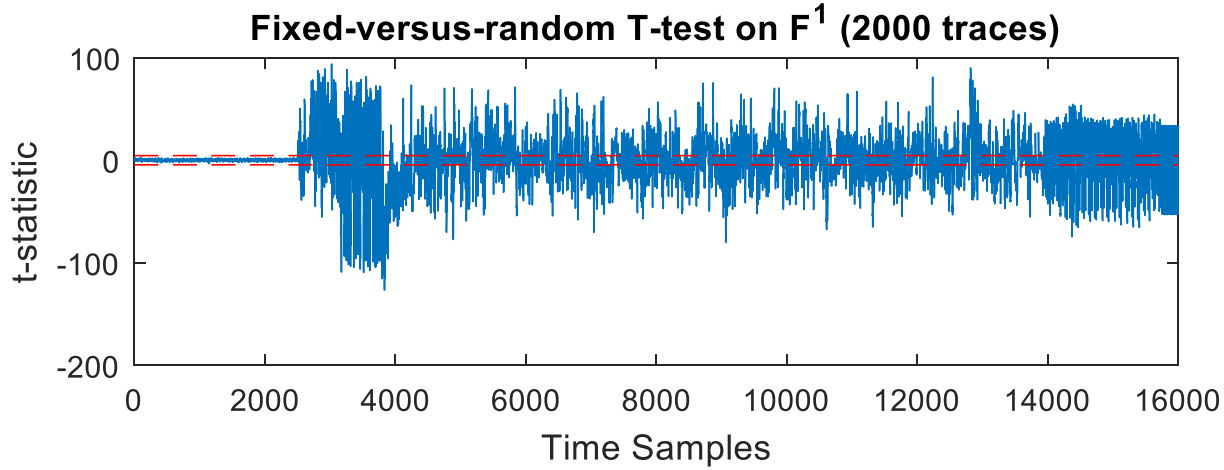
Fixed-versus-random t-test on real traces

- The fixed-versus-random t-test is a non-specific test, which can run very fast, since it is based on simple operations.
- Although, the "lightweight" property of the t-test has some drawbacks.
- As we have seen in Part 1, the t-test provides different detection rate depending on the chosen fixed class.
- The ability of the t-test to detect leakage relies heavily on the signal-to-noise ratio that the chosen fixed class can provide.
- Of course, we will have now real power traces to work with, so, we will perform the t-test on **each time point**.

Fixed-versus-random t-test on real traces

- Let's check a first experiment with 2000 traces with a given fixed input F^1
- The t-test in the standard TVLA is univariate and each time point is considered as a single dimension
- From Part 1, we have learnt if a point exceeds the +/-4,5 threshold in the t-test, it is considered as **leaky**
- <u>But…We have a huge amount of leaky points!!!</u>





Number of leaky points: 9390/12000

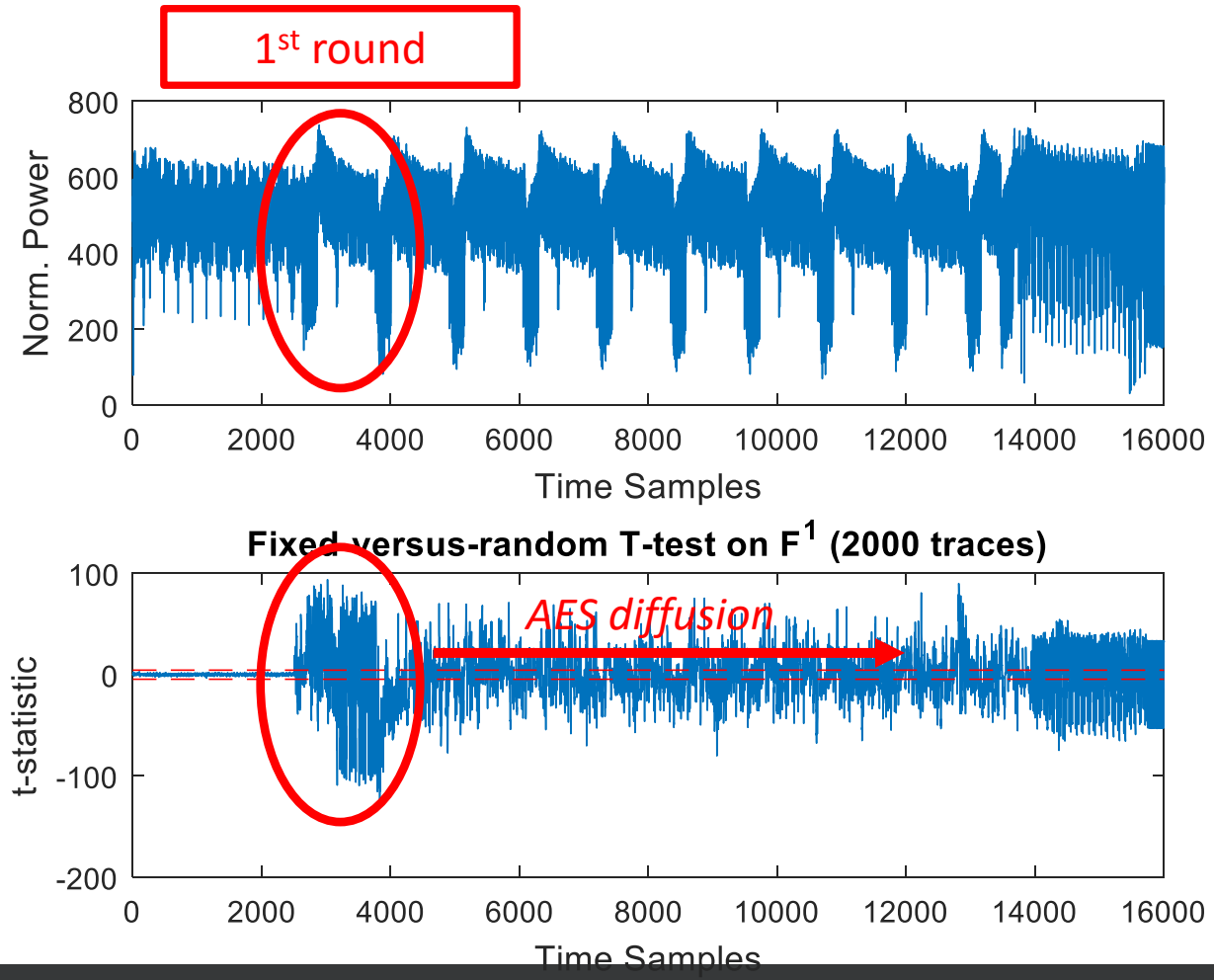
Fixed-versus-random t-test on real traces

- Why we have so much leaky points?

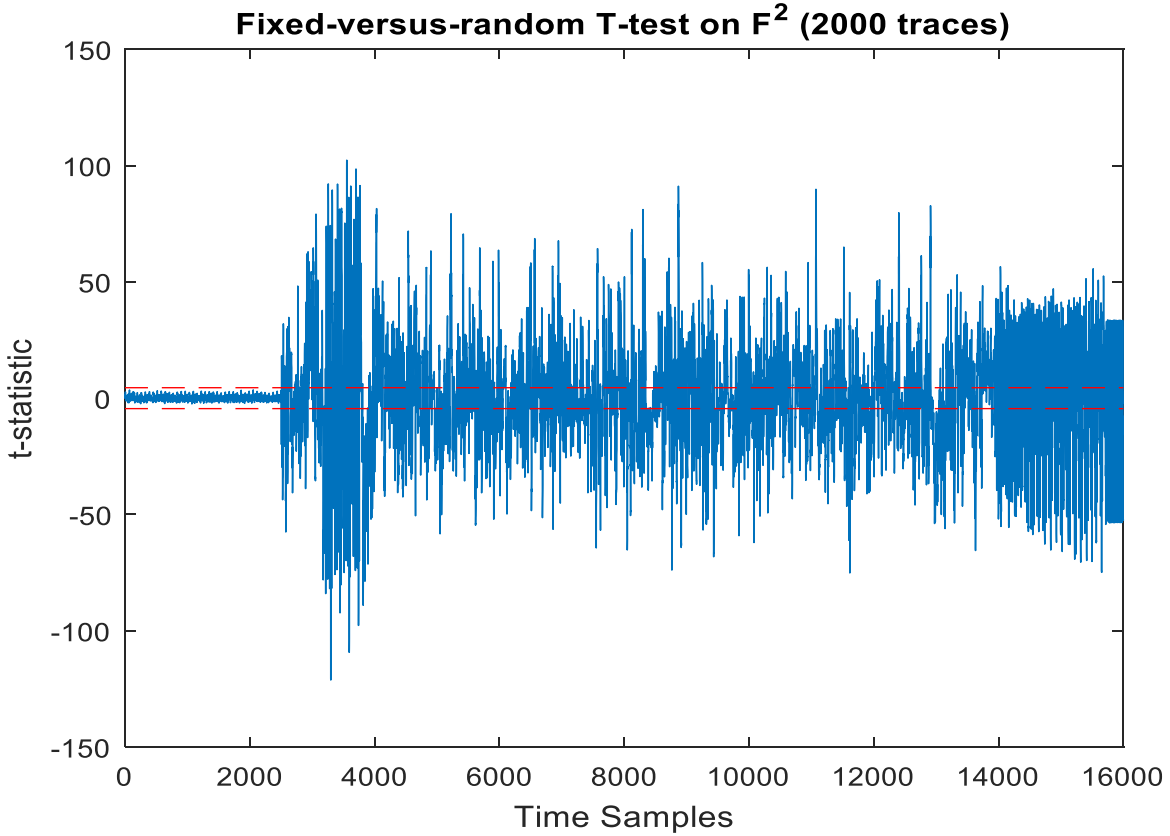
- How can we interpret the leakage from the TVLA?

Effect of the crypto algorithm

- The diffusion of the AES will uniformly distribute intermediate values all over the state after a couple of rounds.
- In fact, the t-test will detect all the points that depend on the input plaintext, since it is **non-specific**.
- Note that in the first part, there is no leakage, since the power consumption is due only to generation of round keys, which is the same for both classes.



Fixed-versus-random t-test with a different fixed class

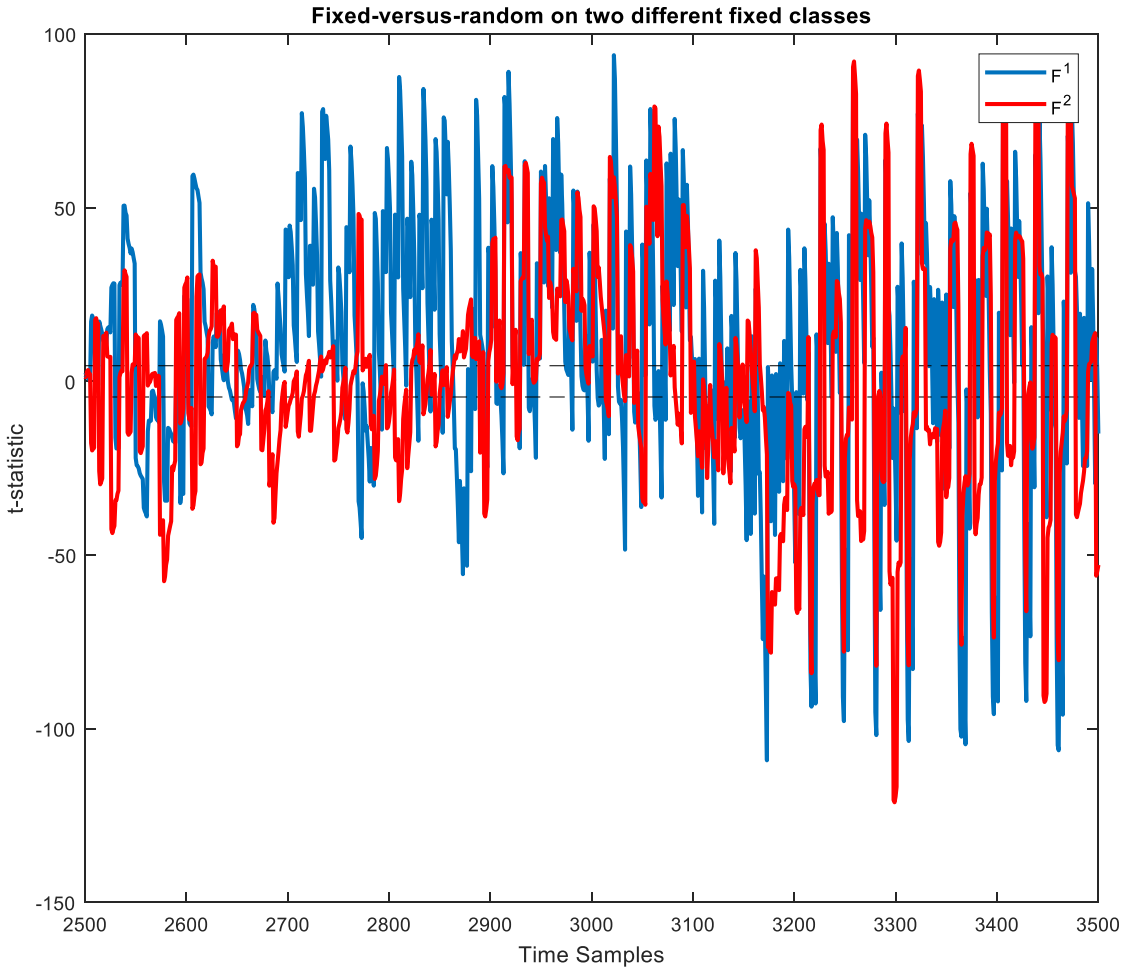


Number of leaky points : 9421/12000

- In a second experiment, we have chosen a different input plaintext F^2 for the fixed class
- Again, the number of leaky points is very high.
- Do the two experiments provide the same set of leaky points?

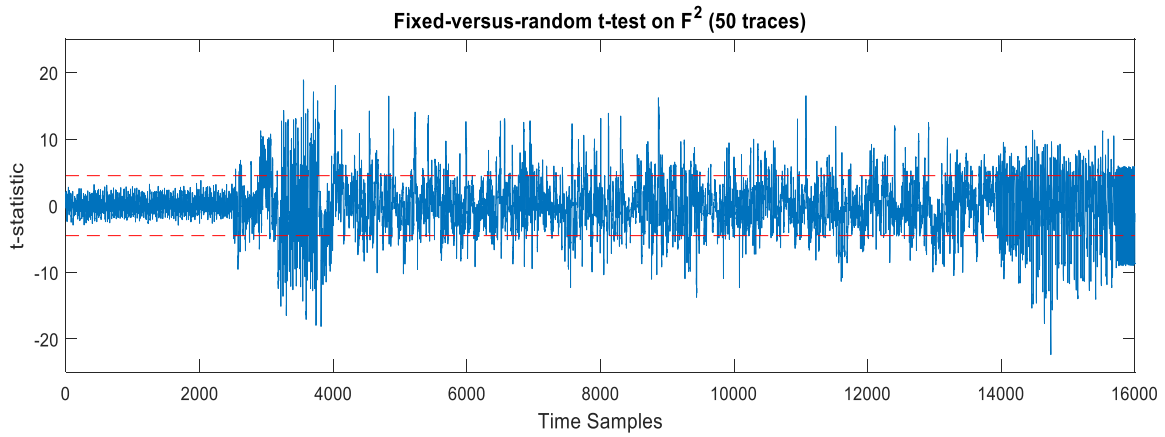
Comparing t-test outcomes for different fixed classes

- The two experiments lead to slight different results
- Different inputs (and classes) provide different results
- The TVLA is able to check if an implementation is leaking or not, but it will not tell you more on the nature of the leakage



TVLA reducing sample size

- What happens if we reduce the number of traces?
- We still detect a huge number of leaky points in block B.
- The t-test score has a reduced value due to smaller sample size, since distinguishing underlying distributions becomes harder.



Number of leaky points : 2617/12000

Experiment	# Leaky points
F^1, 2000tr	9390
F^2, 2000tr	9421
F^2, 50tr	2617

Fixed-versus-random t-test: pros

- Very simple partitions, and very lightweight computations
  - Only a standardised difference of means is performed to compute the t-statistic
- Detection of leakage can be very fast, in terms of sample size, if a fortunate choice about the fixed class has been made
- Widely used and advised in literature for assessing leakage (extended also to high order analysis)

Fixed-versus-random t-test: cons

- 😕 The **fixed-versus-random t-test** in the TVLA procedure is **non-specific**, so it will be able to detect all the differences that propagate through the trace based on the choice of fixed input classes.
- 😕 It will not be able to focus the detection of a specific intermediate function of the algorithm, since the partition is simply done on two input classes, regardless of intermediate results at any point in time.
- 😕 It is not sufficient to use the t-test as it is to make a selection of points-of-interest (POIs), especially for software implementation
  - The high number of false positive along the time window will not help in reducing the data complexity (regarding a standard DPA/CPA attack)

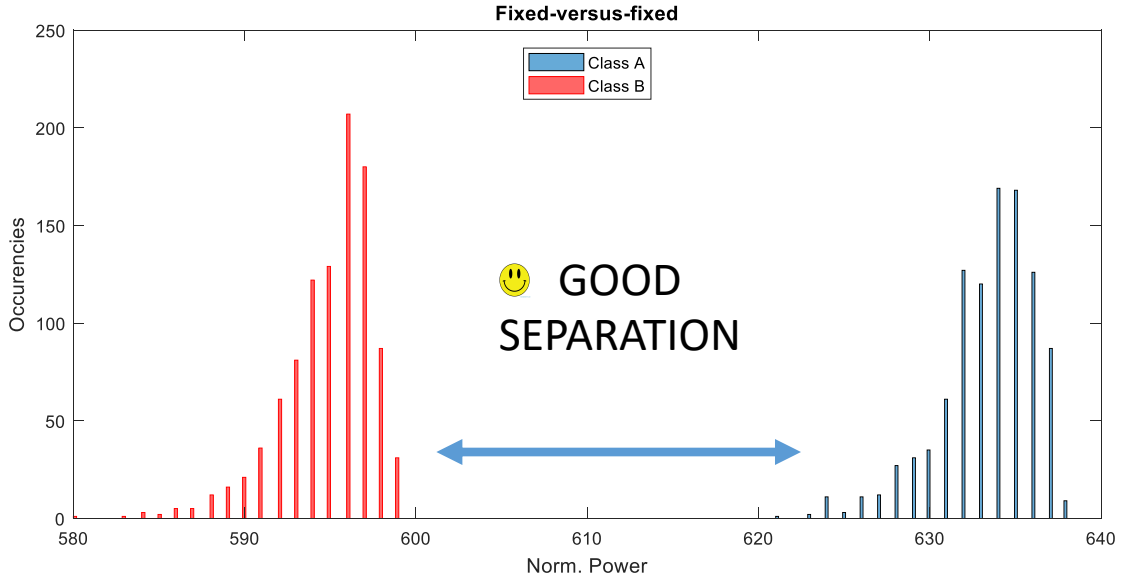
Fixed-versus-fixed t-test

- The **fixed-versus-fixed** t-test for leakage detection is quite similar to the *fixed-versus-random*
  - They share the same procedure
  - No additional operations are required
- In this case, the two classes are both fixed:
  - It has some nice property that makes it preferable to the classical fixed-versus-random t-test (that will be strongly related to efficiency in detection)
  - The fixed-versus-fixed partitioning allows to reaching larger differences making the detection easier

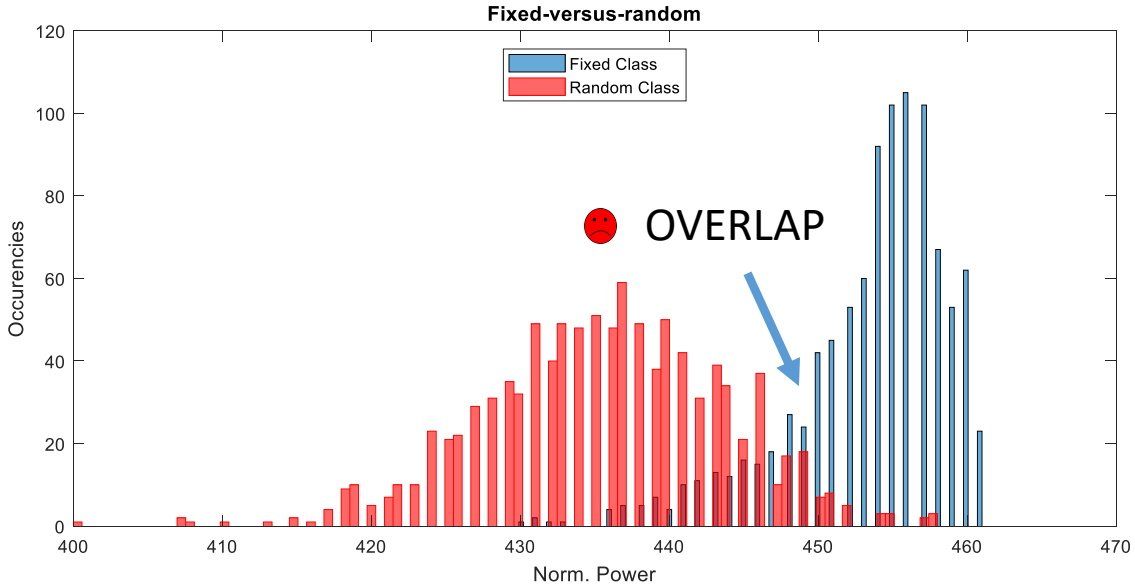
	Fixed-versus-Random	Fixed-versus-Fixed
Class A	Fixed	Fixed
Class B	Random	Fixed

Fixed-versus-fixed t-test: a SNR fact

In the fixed-versus-fixed t-test it is easier to distinguish the two classes compared to the fixed-versus-random. For an unprotected implementation, the overall noise is reduced from $2\sigma_N^2 + \sigma_{alg}^2$ to $2\sigma_N^2$ and the signal is, in general, larger.



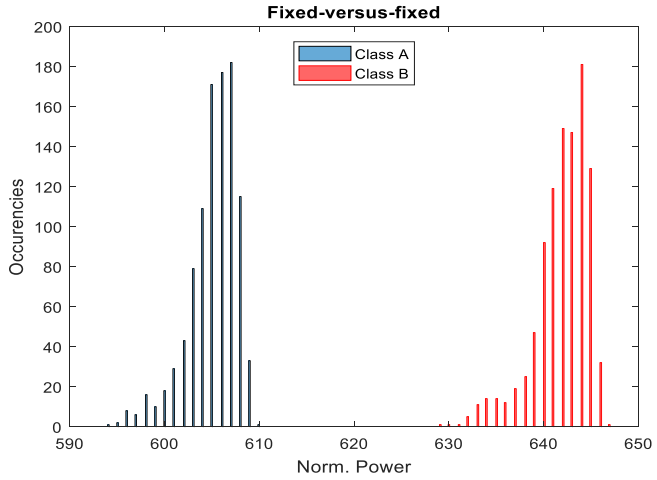
FIXED-VERSUS-FIXED

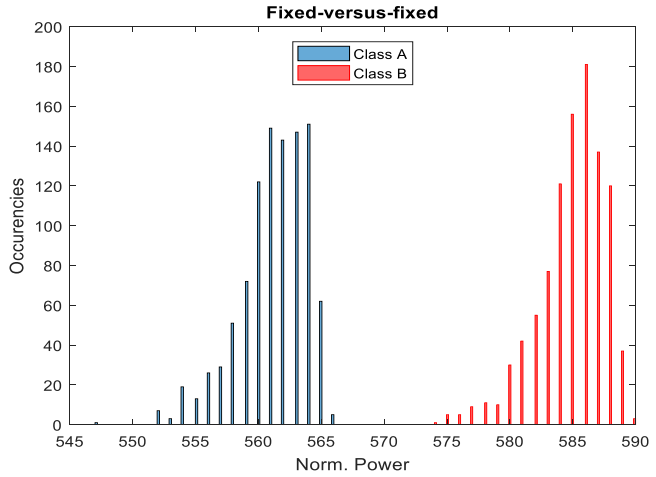


FIXED-VERSUS-RANDOM

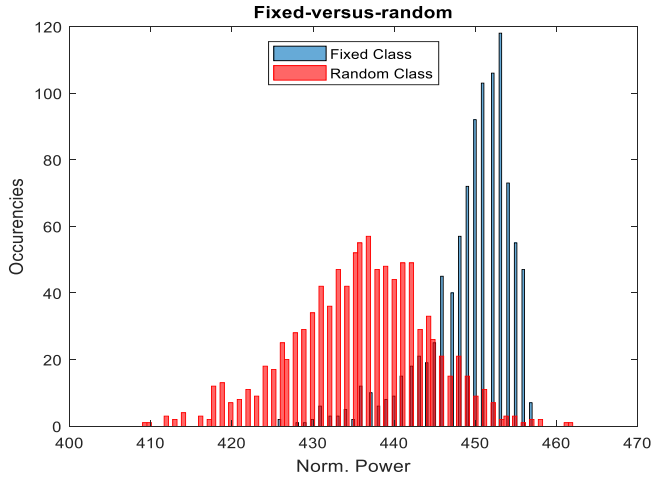
FvF and FvR: examples

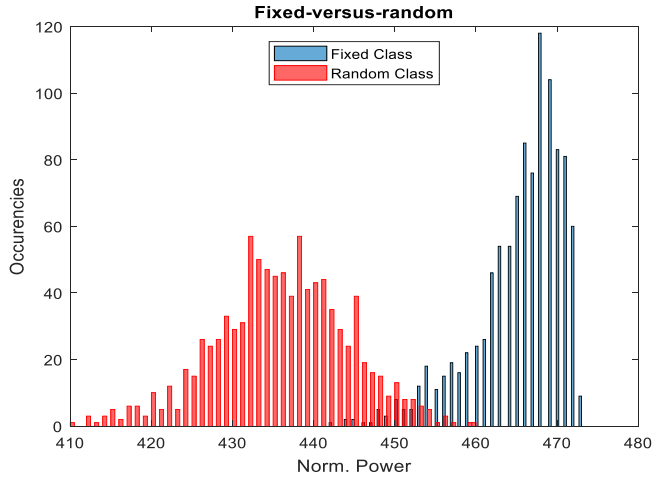
FIXED VERSUS FIXED





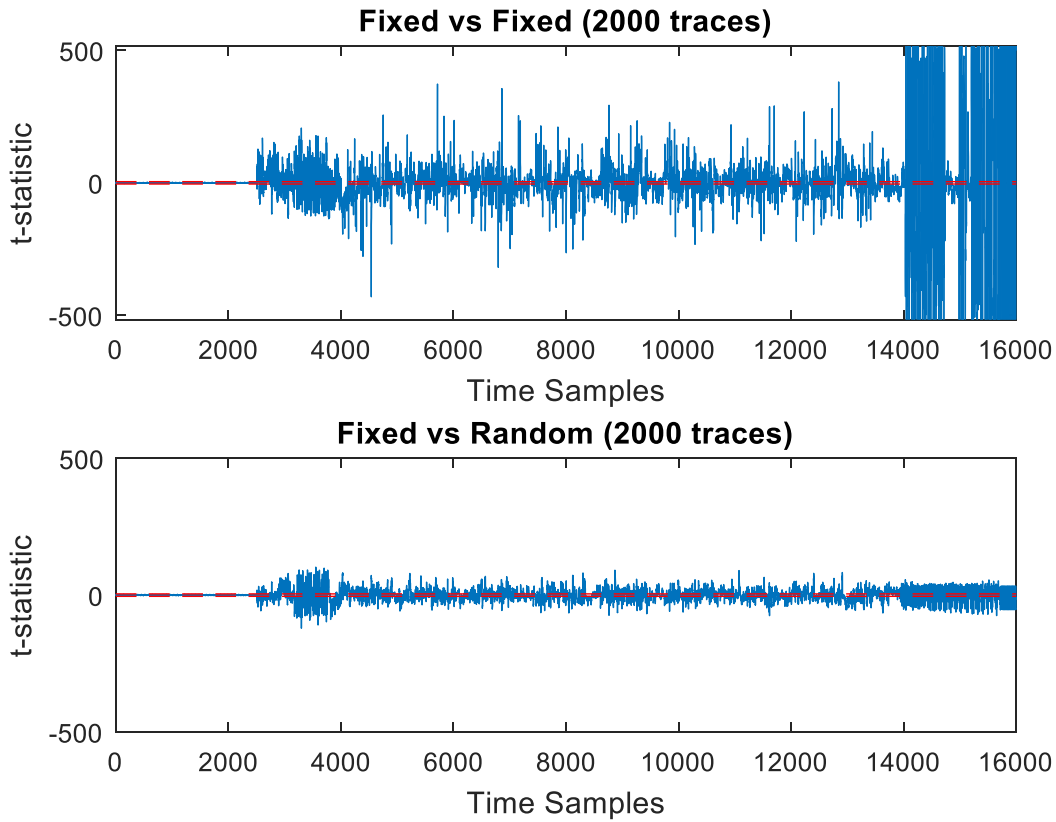
FIXED VERSUS RANDOM





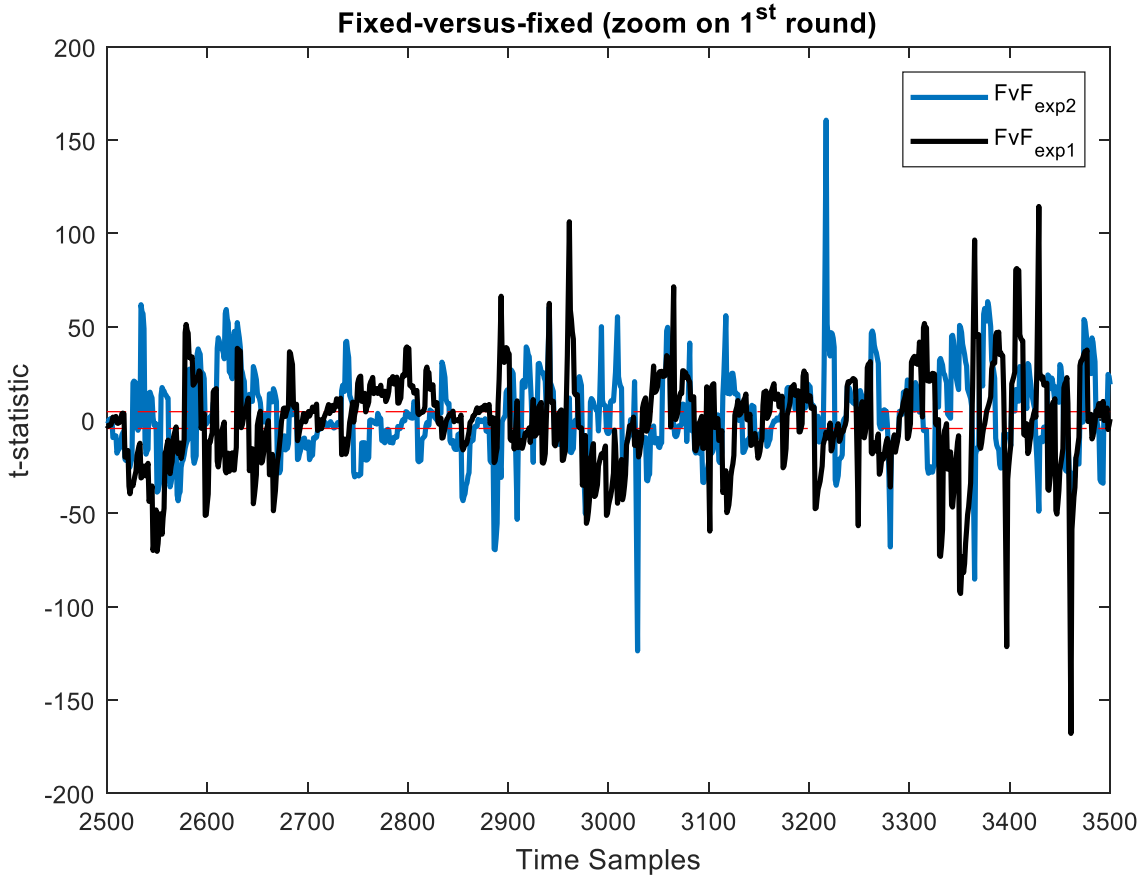
Fixed-versus-fixed t-test experiments

- Repeating the same experiment with adopting the fixed-versus-fixed approach (2000 traces)
- We reach twice the signal around the first round with same sample size
- Again, the **diffusion** tends to highlight a huge number of leaky time points after the first round



Fixed-versus-fixed t-test experiments

- Repeating the experiment for a different pair of fixed input values (2000 traces).
- We can notice that the t-statistic values and spotted leaky points are different:
  - Different fixed pairs provide different results (as expected)



Improving the detection

- The TVLA has low sampling complexity in the average, but it is not able to provide useful information about useful leaky points over a huge time windows.
- To reduce the risk of *false negatives* in fixed-versus-random t-test, a **correlation-based** test has been proposed at Eurocrypt 2016 [DS16].
- Instead of testing for a difference of means in the power consumption between two classes, it tests correlation between measured power consumption and estimated power model, adopting a *divide&conquer* approach.

[DS16] F. Durvaux and F.-X. Standaert, From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces, Eurocrypt 2016

Correlation-based test for detection

- The aim of this test is to detect leaky points in a trace, finding linear correlation between the actual power consumption and an *estimated* power model of the device, which is based on measurements.
- It requires, in fact, a two-step procedure*:
  - **Profiling phase:** a power model is estimated according to a specific data-partitioning on the target device.
  - **Detection phase:** the linear correlation is computed between the estimated power model and the actual power consumption, according to chosen data-partitioning.
- It is easy to interpret since it states the *linear* relationship between the power measurement and the estimated power model by means of *Pearson's correlation coefficient*:
  - It provides intuitions on the outcome of a *specific* attack that is the *Correlation Power Analysis (CPA).*

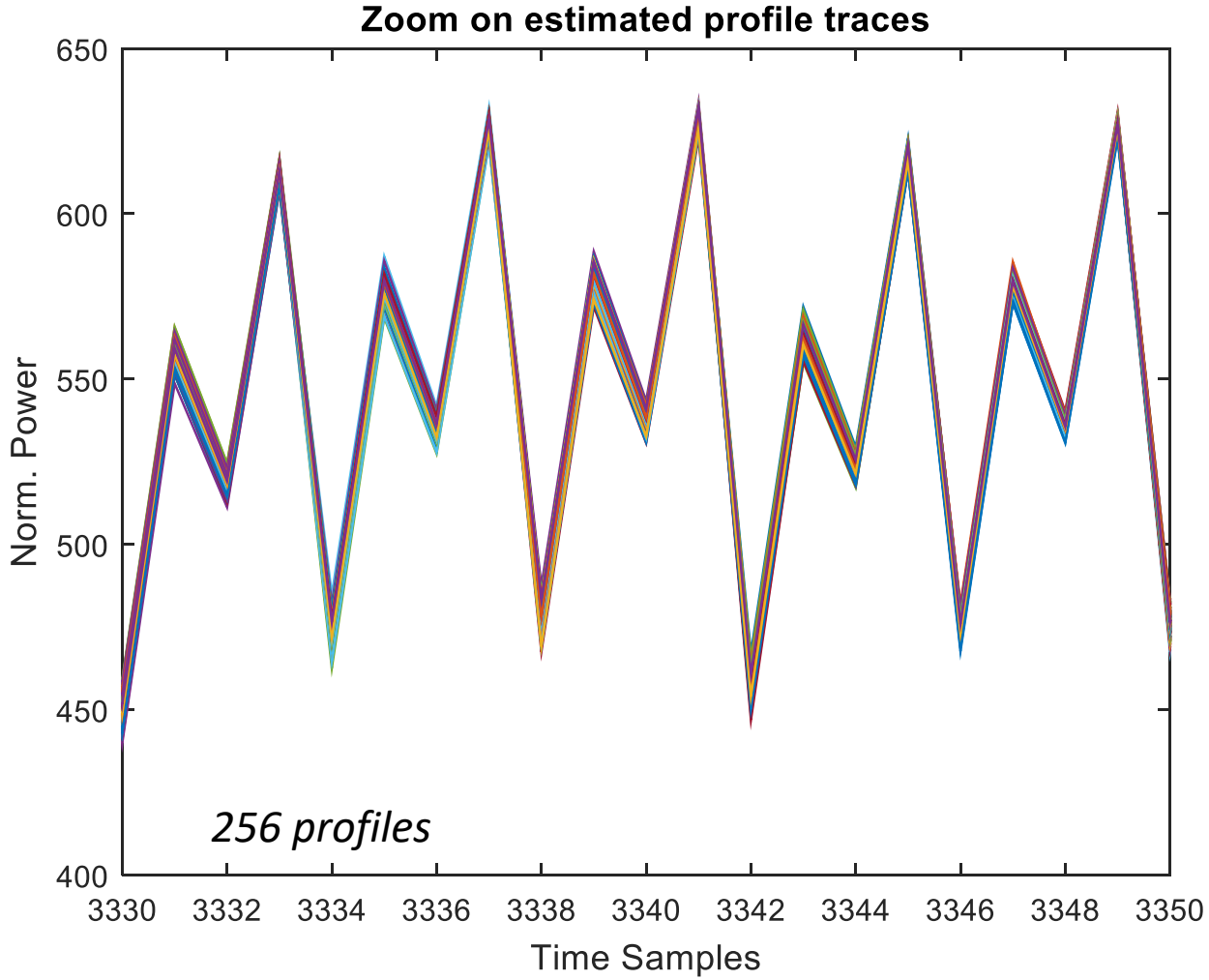
*On the **online** version of this tutorial, we have used a *k*-fold cross-validation to deal with bias and variance of the test.
Visit http://reassure.eu/leakage-detection-tutorial/ for codes, datasets and more details!

Profiling phase: estimating the power model

- Assuming to choose the input plaintext value as partition-criterion, for AES-128, we build 256 profiles:

$$L_p^i(\tau) = \frac{1}{N}\sum_{j=1}^{N} L^i(\tau)\,, \forall i = 0, \dots, 255$$

- Each profile is basically an averaged power trace of all traces generated with the same input plaintext



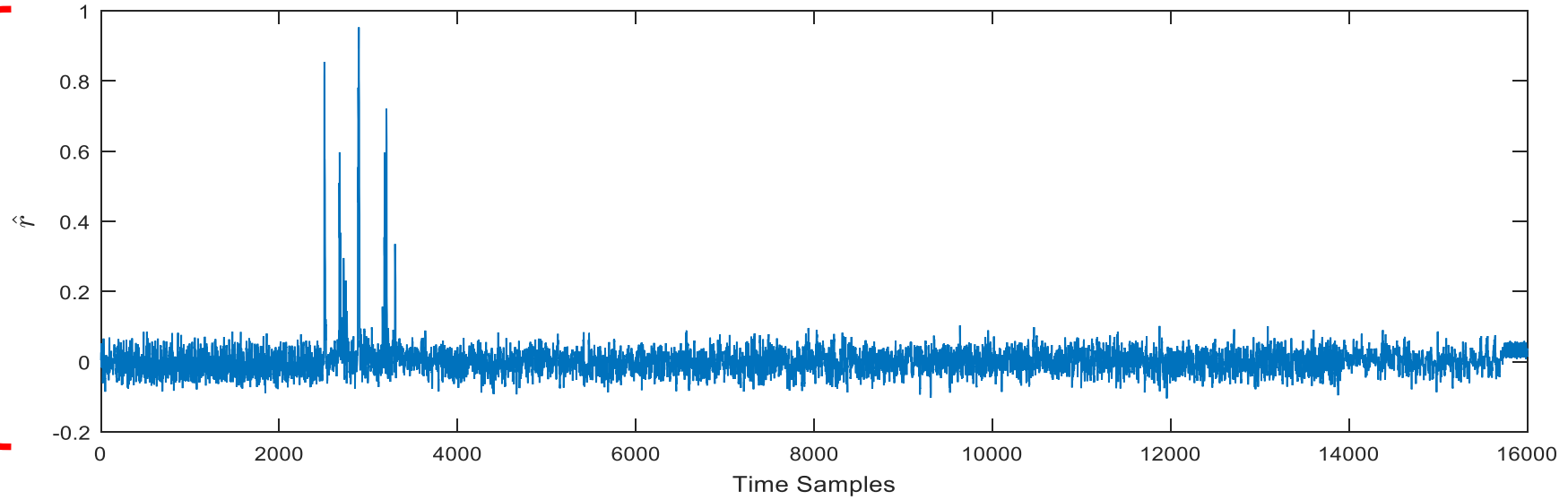
Detection phase: is there correlation?

- The profiles built at the previous step are then compared point-wise with another set of traces, collected on the same device, by means of Pearson's correlation coefficient:

$$\hat{r}(\tau) = \rho(L_t^i(\tau), L_p^i(\tau))$$

Note the i index

Leaky points??

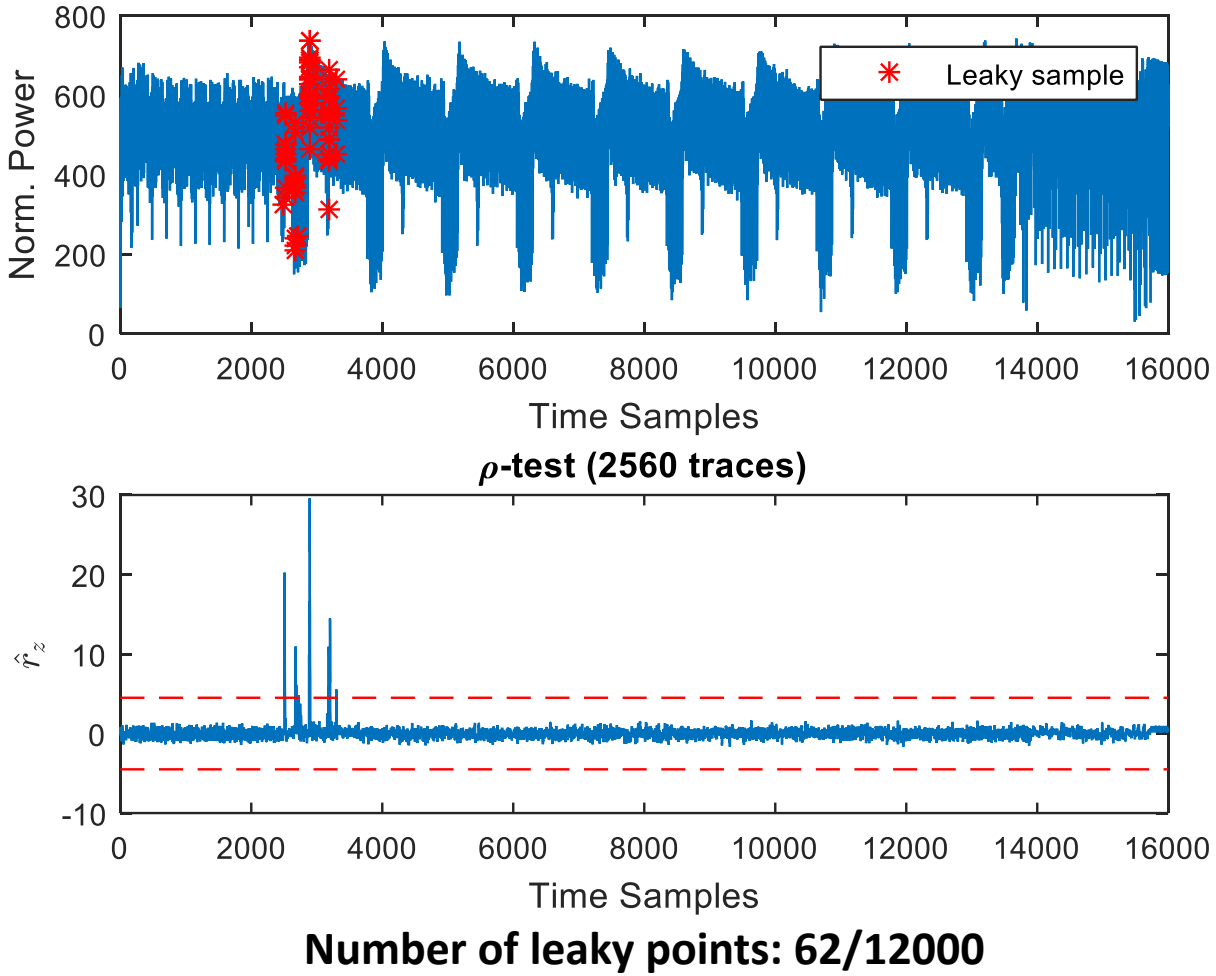


Detection phase: leaky or not leaky?

- Now, we have to find a way to interpret the result we got to state if a time point is leaking information or not
- A handy solution is to **transform** the correlation plot from the previous slide in something similar to the t-statistic plots we have seen so far:
  - The **Fisher's z-transform** is used to interpret the outcome of the correlation-based test according to a normal distribution $\boldsymbol{N}(0,1)$ and derive a threshold to **test** for "**non-zero correlation**" versus "**zero correlation**" (needs to be normalized with the standard deviation $\frac{1}{\sqrt{N-3}}$)

$$\widehat{r_z}(\tau) = \frac{1}{\sqrt{N-3}} \cdot \frac{1}{2} \cdot \ln\left(\frac{1+\hat{r}(\tau)}{1-\hat{r}(\tau)}\right)$$

Detection phase: what do you notice?



Number of leaky points: 62/12000

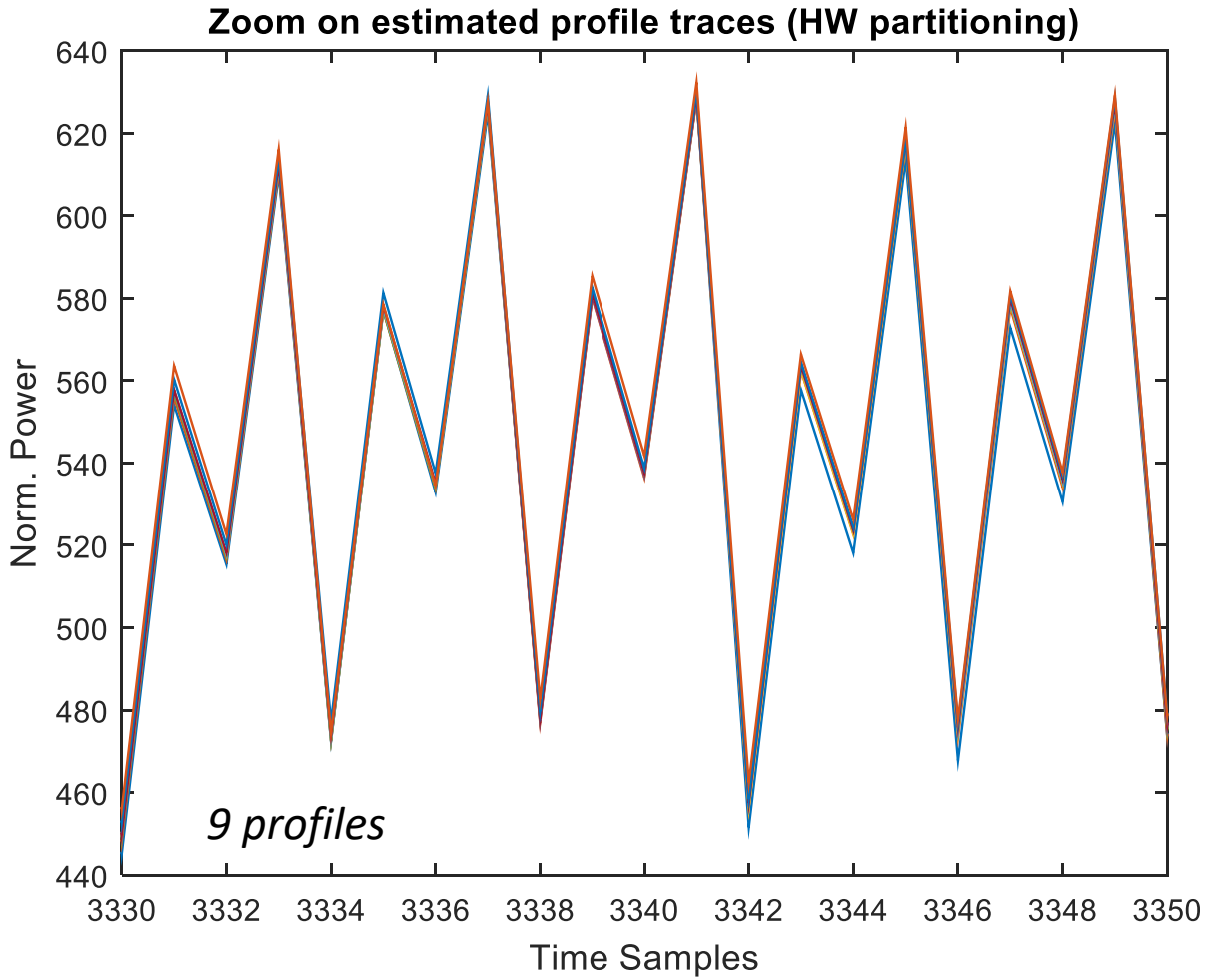
- Not all the peaks we have seen in the correlation plot lead to a leaky time point.
- The number of leaky points is extremely reduced (62 points) and localized just around the first round.
- The output statistic is way lower compared to the t-test, which give us an intuition on the needs of larger sampling complexity of the correlation-based test.

Correlation-based test: what's the difference?

- The correlation-based test goes into a different direction compared to the t-test
- It makes the detection adopting a huge number of classes at once
  - 256 classes for AES128 with divide&conquer approach
- The correlation-based test is able to provide POIs that are vulnerable to a specific attack (CPA)
  - It is much more localized compared to t-test
- It can be extremely powerful in both grey and black box model of the device

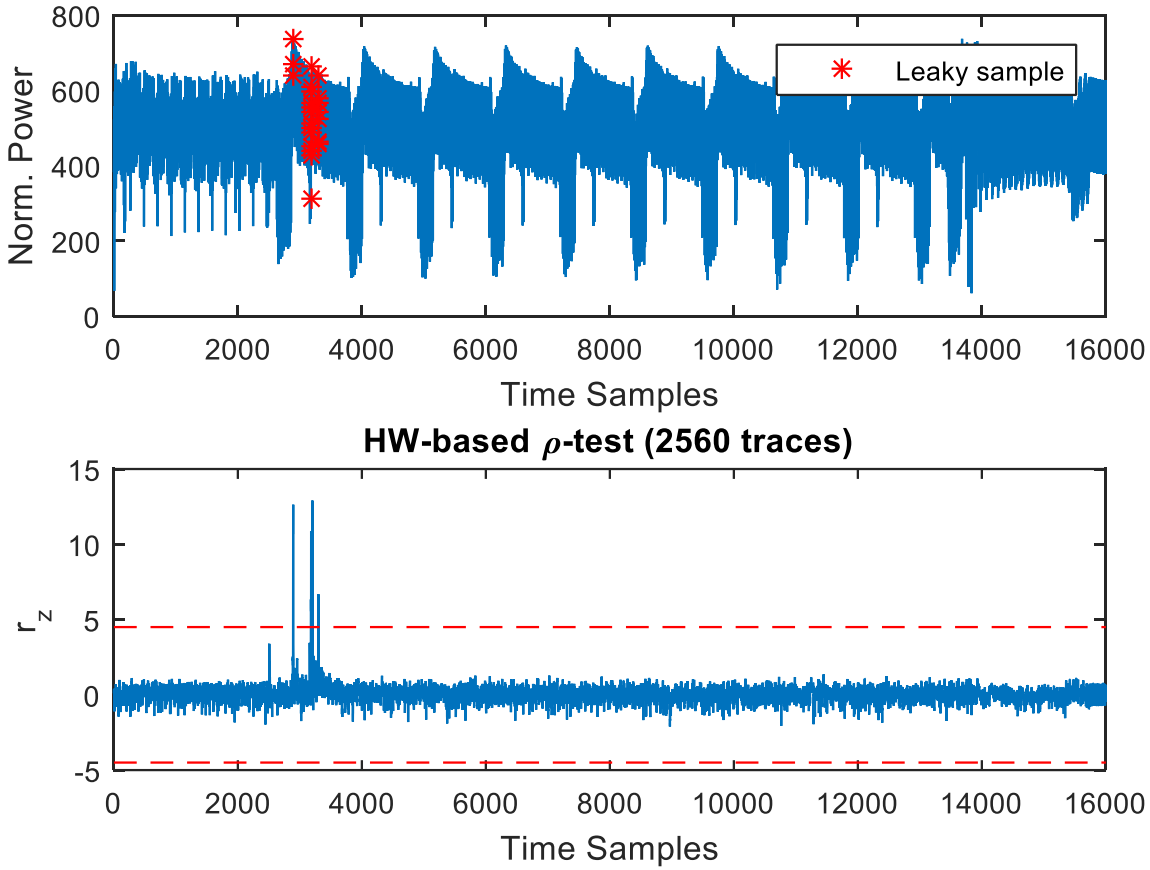
Correlation-based test: another example

- We provide another example adopting a different approach to profiling:
  - The profiles are built upon the HW of the output value of the Sbox of the 1st byte at the 1st round.
  - Stronger assumptions are required: device leaking following HW and good hypothesis for a key.
  - It focus directly on the output function 'Sbox' → *SPECIFIC TEST*!



Correlation-base test: HW detection

- The number of leaky point is even smaller (23 versus 62):
  - We are using a **specific** target intermediate value and a **specific** power model to get the partitioning
  - Very local and focused detection

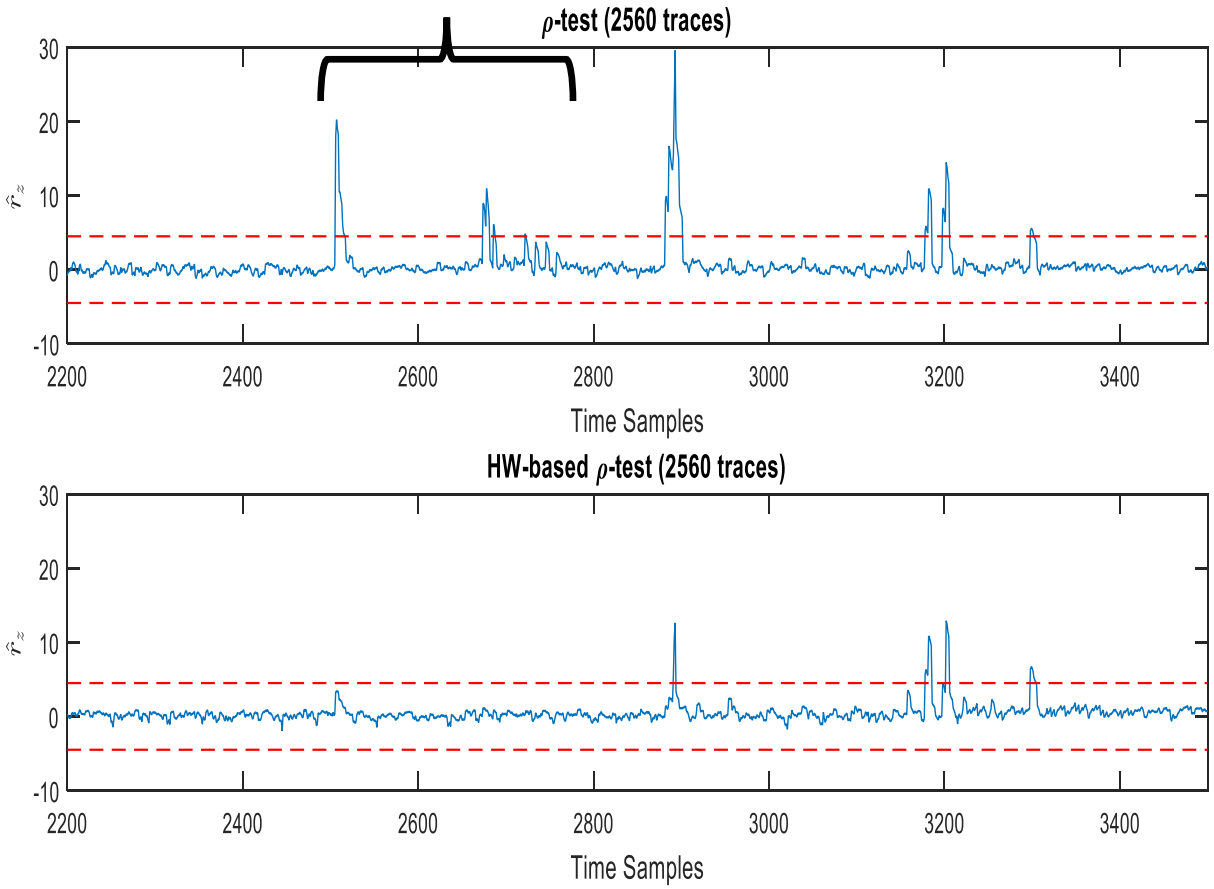


Number of leaky points: 23/12000

Comparison of partitioning methods

- Peaks observed in the first case (input values profiling) are due to manipulations of *useless* plaintext byte and *useful* intermediate values, that depends **bijectively** on it (loading plaintext, Sbox and XOR).
- Peaks in the HW-partitioning are strictly related to the target Sbox operations.
- *Note*: in this case the test is **specific**.

Non related to Sbox



Take home messages of Part 2

- Working with real traces is different from working with simulated leakage, even if it helps a lot in understanding problems
- Detection on real traces showed us an import thing of this evaluation process:
  - **T-test** offers a detection procedure that requires low sampling complexity with a very easy-to-do procedure, that can be useful in a pass/fail leakage assessment
  - If we want to get a better understanding of the leakage, we have to use more classes to test and increase the sampling complexity, as we have seen for **correlation-based test**
  - There is a **trade-off**: more information → higher sampling complexity and more classes required for testing

Thanks for your attention!
Q&A

Visit http://reassure.eu/leakage-detection-tutorial/ for codes, datasets and more details!!