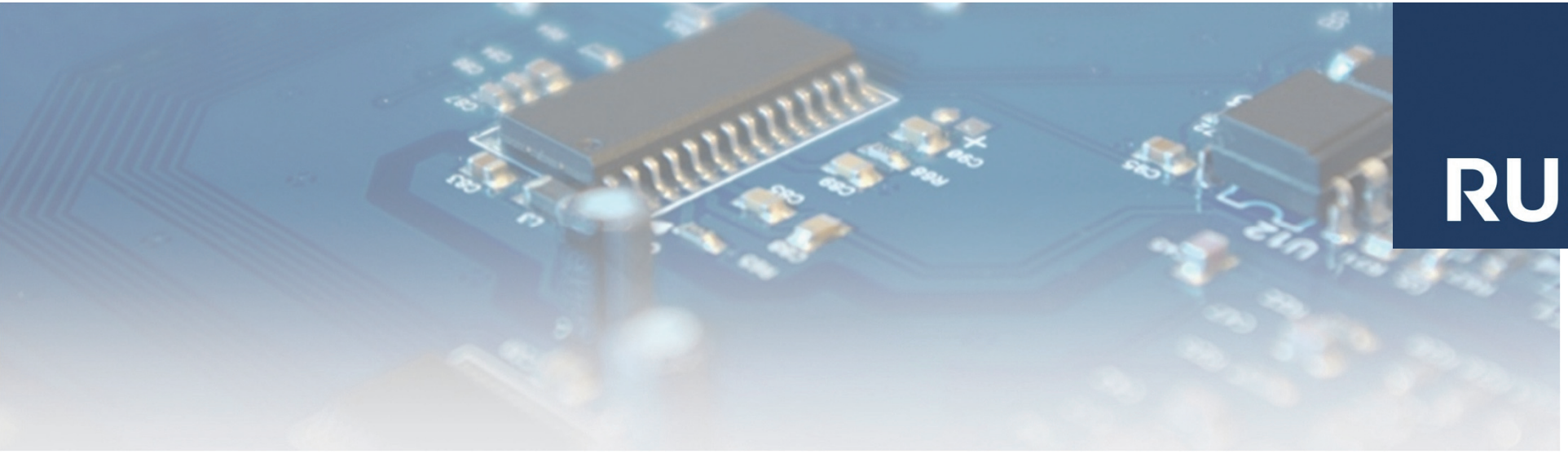


RUB

RUHR-UNIVERSITÄT BOCHUM

Leakage Assessment Methodology

- a clear roadmap for side-channel evaluations -

29. August 2015

Tobias Schneider & Amir Moradi

Ruhr-Universität Bochum

hgi Horst Görtz Institute for IT-Security

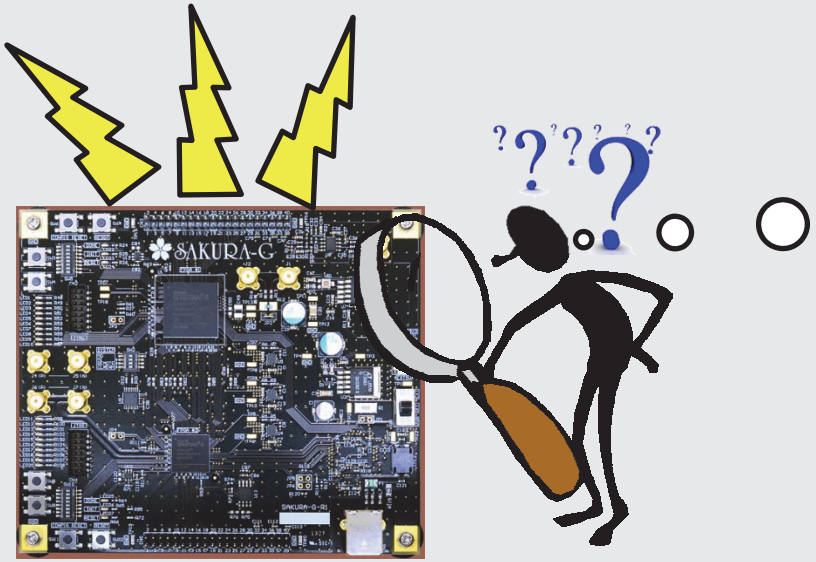
Outline

- Motivation
- Statistical Background
- Testing Methodology
- Higher-Order Testing
- Efficient Computation
- Case Studies
- Conclusion

Motivation

- Security Evaluation
- Attack-based Testing
- Information-theoretic Testing
- Testing based on *t*-Test

Motivation - Security Evaluation



How secure is this chip?

Problem: Evaluation is not trivial.

NIST *Non-Invasive Attack Testing Workshop, 2011*

Goal: Establish testing methodology capable of robustly assessing the physical vulnerability of cryptographic devices.

Motivation - Attack-based Testing

Perform state-of-the-art attacks on the device under test (DUT)

Attacks Types:
- DPA
- CPA
- MIA
- …

×

Intermediate Values:
- Sbox In
- Sbox Out
- Sbox In/Out
- …

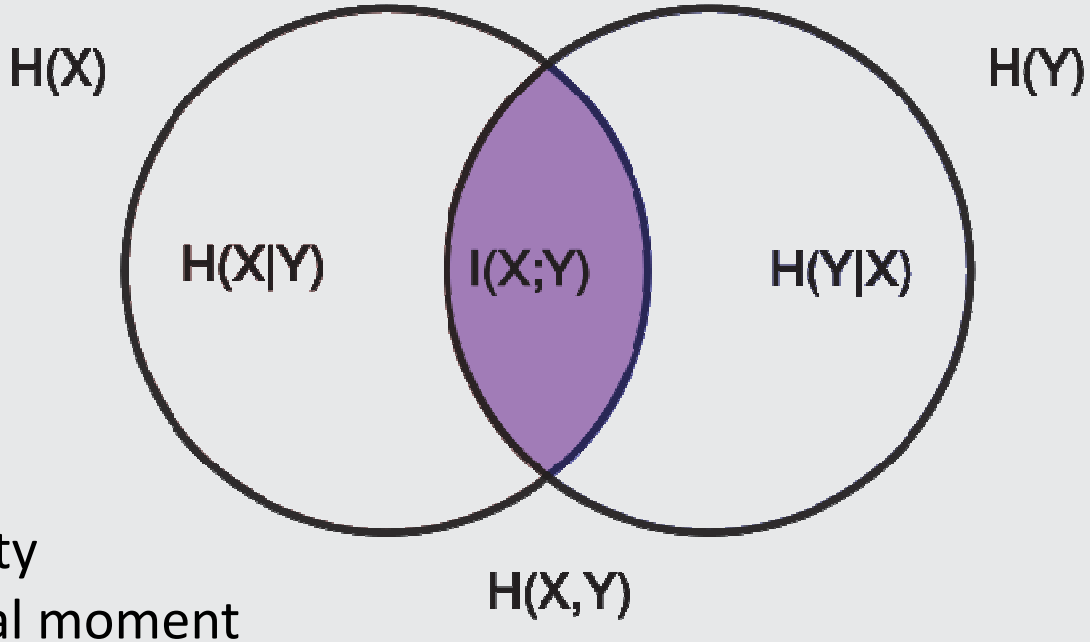
×

Leakage Models:
- HW
- HD
- Bit
- …

Problems:
- High computational complexity
- Requires lot of expertise
- Does not cover all possible attack vectors

Motivation - Information-theoretic Testing

Computation of Mutual/Perceived Information



Problems:

- High computational complexity
- Cannot focus on one statistical moment
- Dependent on PDF-Estimation
- Does not cover all possible attack vectors

Motivation - Testing based on *t*-Test

Tries to detect any type of leakage at a certain order

- Proposed by CRI at NIST workshop

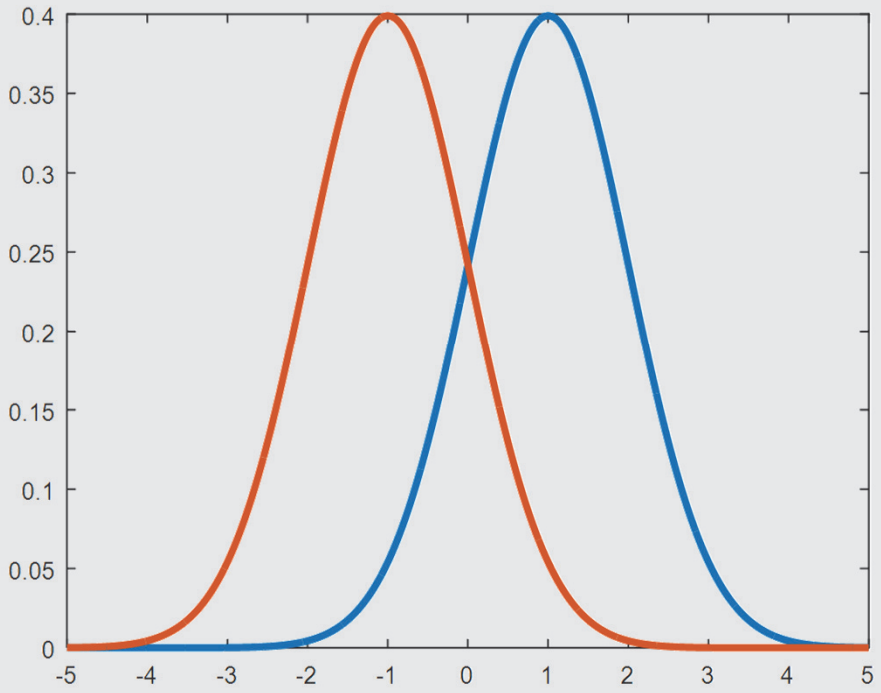
CRYPTOGRAPHY RESEARCH
a division of Rambus

Advantages:

- Independent of architecture
- Independent of attack model
- Fast & simple
- Versatile

Problems:

- No information about hardness of attack
- Possible false positives if no care about evaluation setup



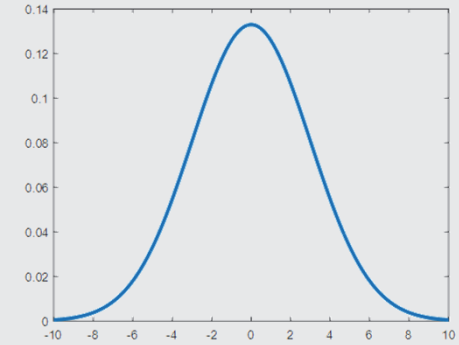
Motivation

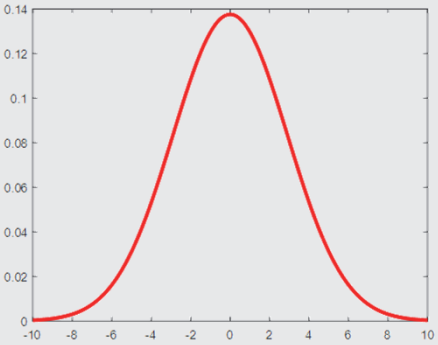
- **In this talk:**
  - (Hopefully) understandable explanation of the tests
  - Detailed explanation of how to conduct tests in higher-orders
  - Discuss efficiency and accuracy problems and provide efficient and robust formulas
  - How to design an appropriate framework to host the DUT for such tests, including both software and hardware platforms (e.g., FPGA, µController)
  - Two case studies

Statistical Background

- *t*-Test

Statistical Background - t-Test



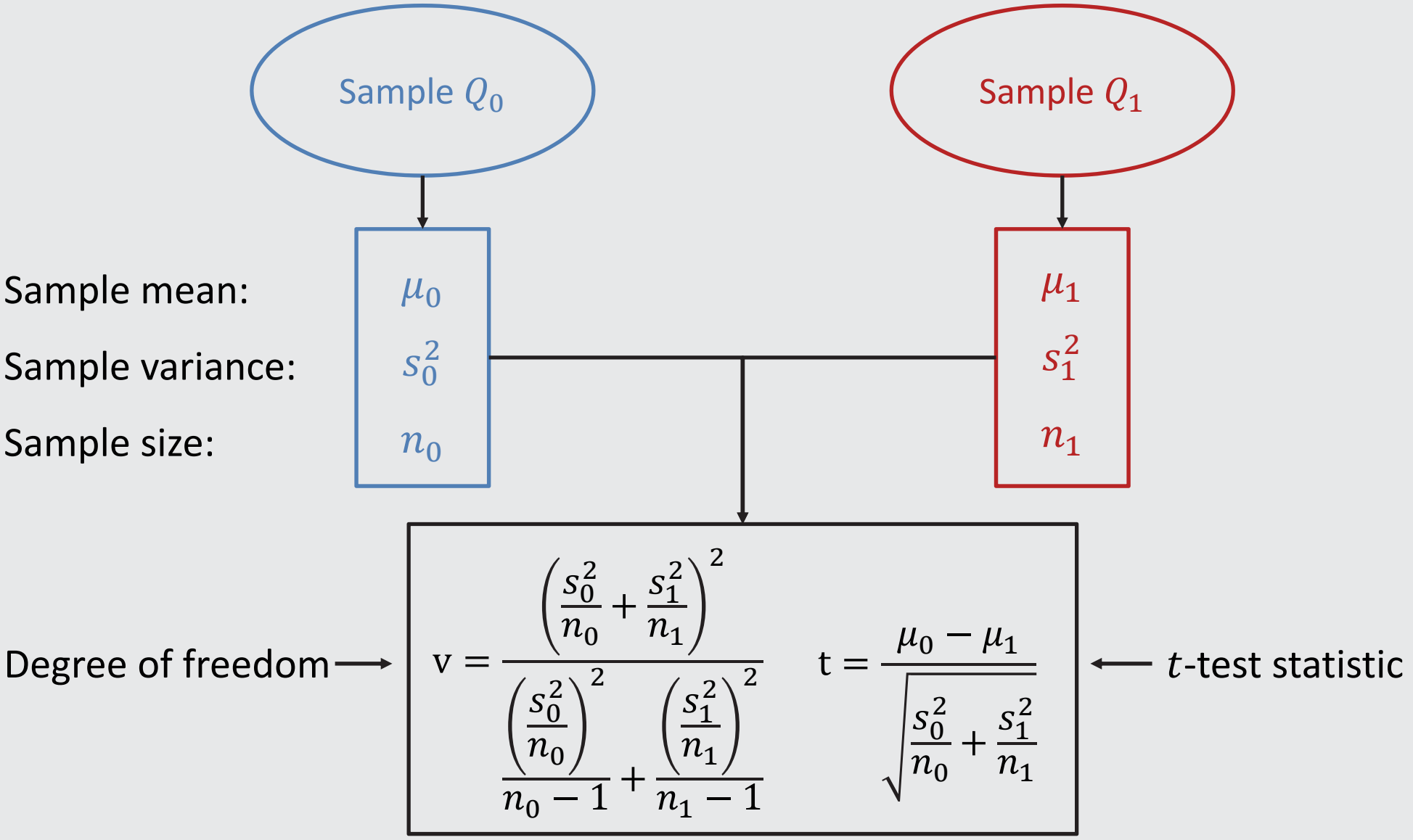


Sample Q_0

Sample Q_1

Null Hypothesis: Two population means are equal.

Statistical Background - t-Test



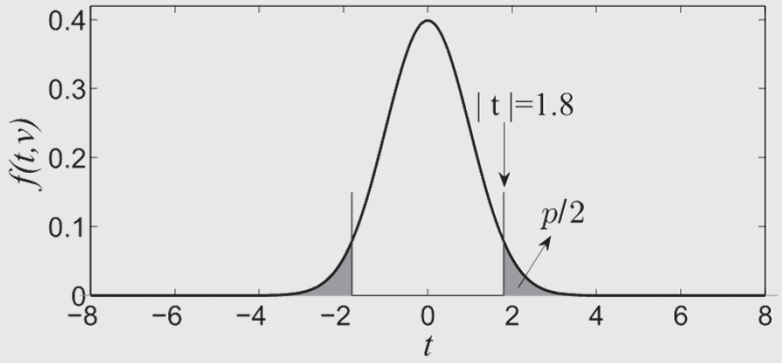
Statistical Background - t-Test

Estimate the probability to accept null hypothesis with Student's t distribution:

$$f(t,v) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{\pi v}\,\Gamma\left(\frac{v}{2}\right)}\left(1+\frac{t^2}{v}\right)^{-\frac{v+1}{2}}$$

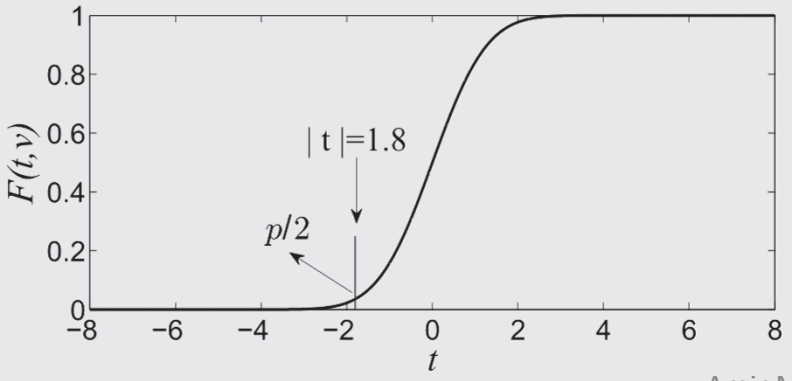
With probability density function:

$$p = 2\int_{|t|}^{\infty} f(\mathrm{t},\mathrm{v})\,dt$$



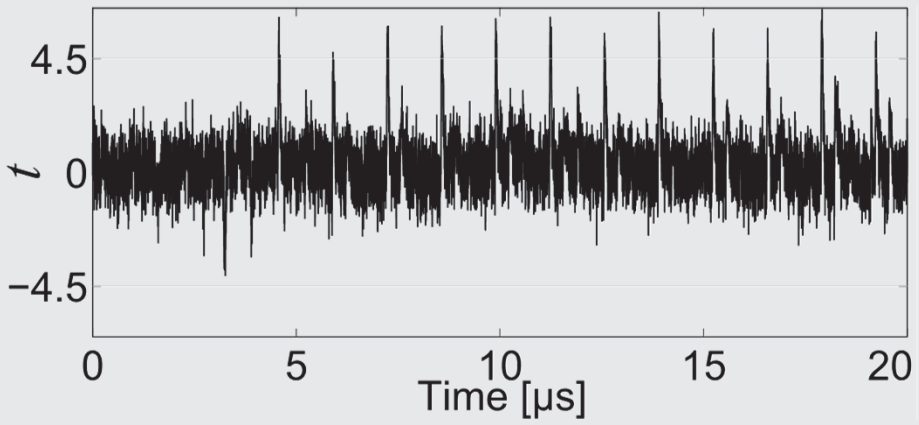
With cumulative density function:

$$p = 2F(-|\mathrm{t}|,\mathrm{v})$$



Statistical Background - t-Test

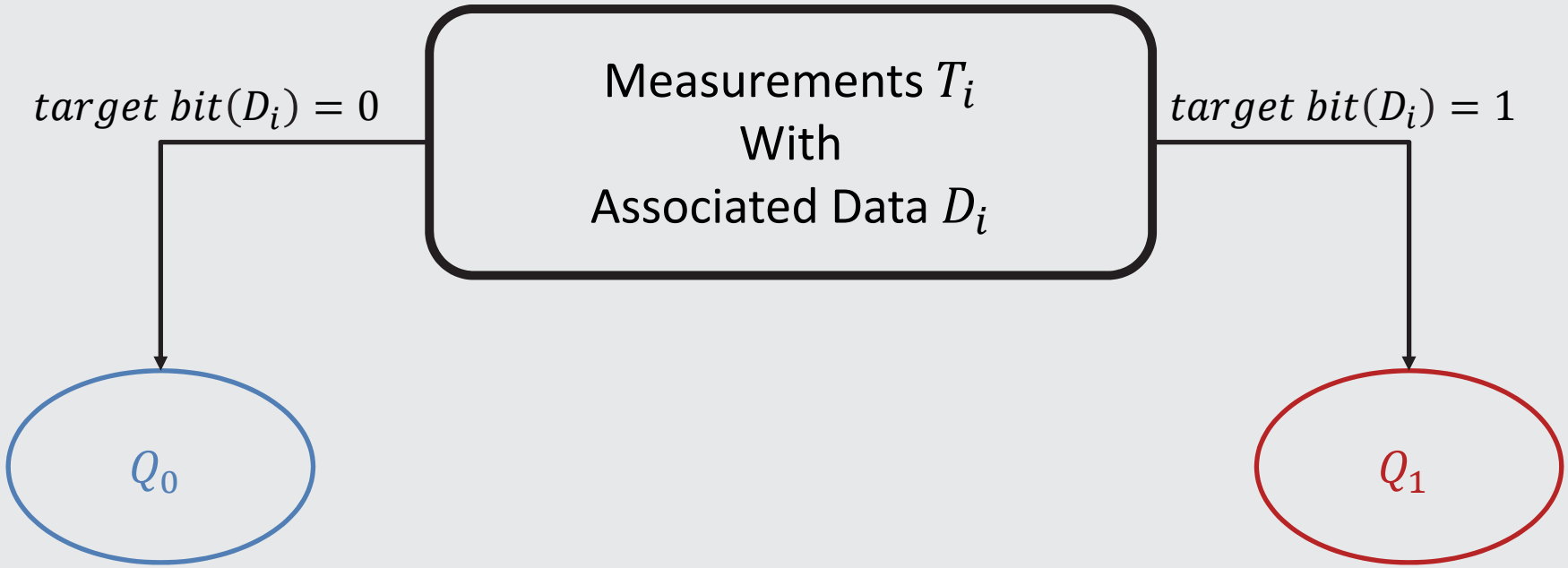
- Small p values give evidence to reject the null hypothesis
- For testing usually only the t-value is estimated
- Compared to a threshold of $|t| > 4.5$
  - $p = 2F(-4.5, v > 1000) < 0.00001$
  - Confidence of > 0.99999 to reject null hypothesis



Testing Methodology

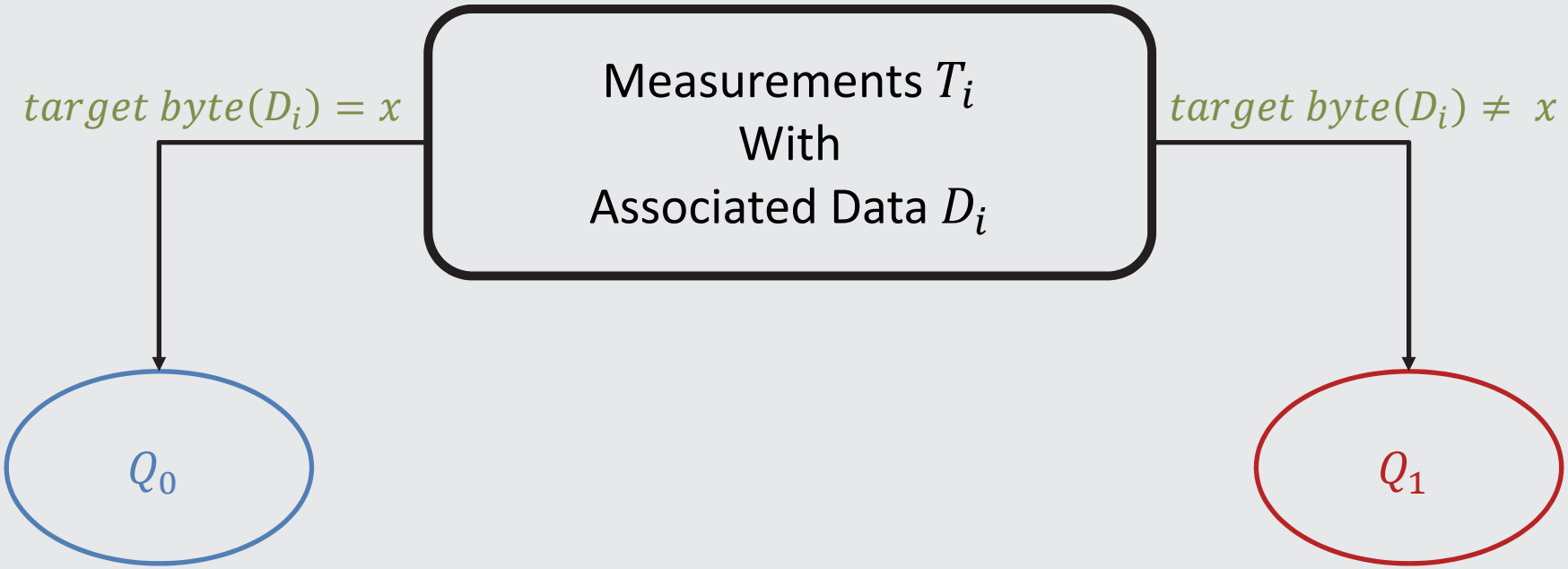
- Specific t-Test
- Non-Specific t-Test

Testing Methodology - Specific t-Test



- Test is conducted at each sample point separately (univariate)
- Key is known to enable correct partitioning
- If corresponding t-test exceeds threshold ⇒ DPA probable

Testing Methodology - Specific t-Test



- Test is conducted at each sample point separately (univariate)
- Key is known to enable correct partitioning
- If corresponding t-test exceeds threshold ⇒ DPA probable
- Other classifications possible (e.g. Sbox output byte)

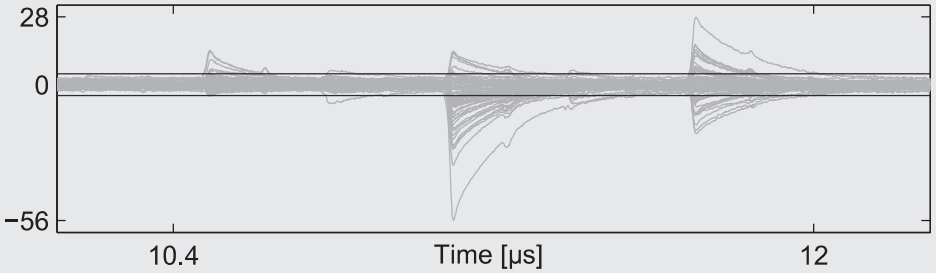
Testing Methodology - Specific t-Test

Example: PRESENT (first round)

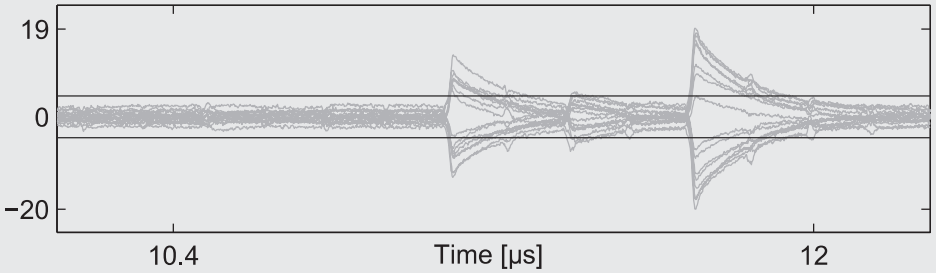
- addRoundKey, sBoxLayer, pLayer
- Bitwise: 3×64 tests
- Nibblewise: $3 \times 16 \times 16$ tests
- Other tests possible

Problems:

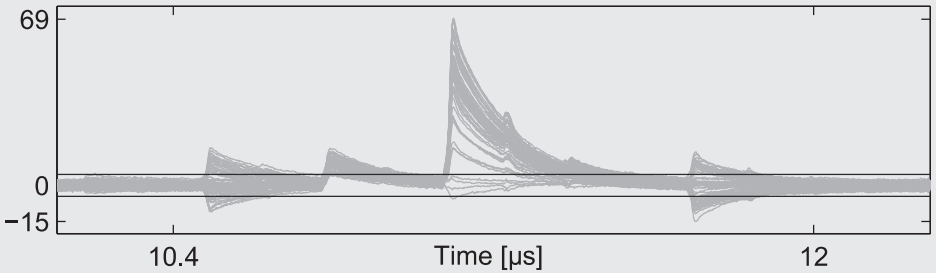
- Same as attack-based approach
- Many different intermediate values
- Many different models
- Prevents comprehensive evaluation



Sbox out bits (64 models)



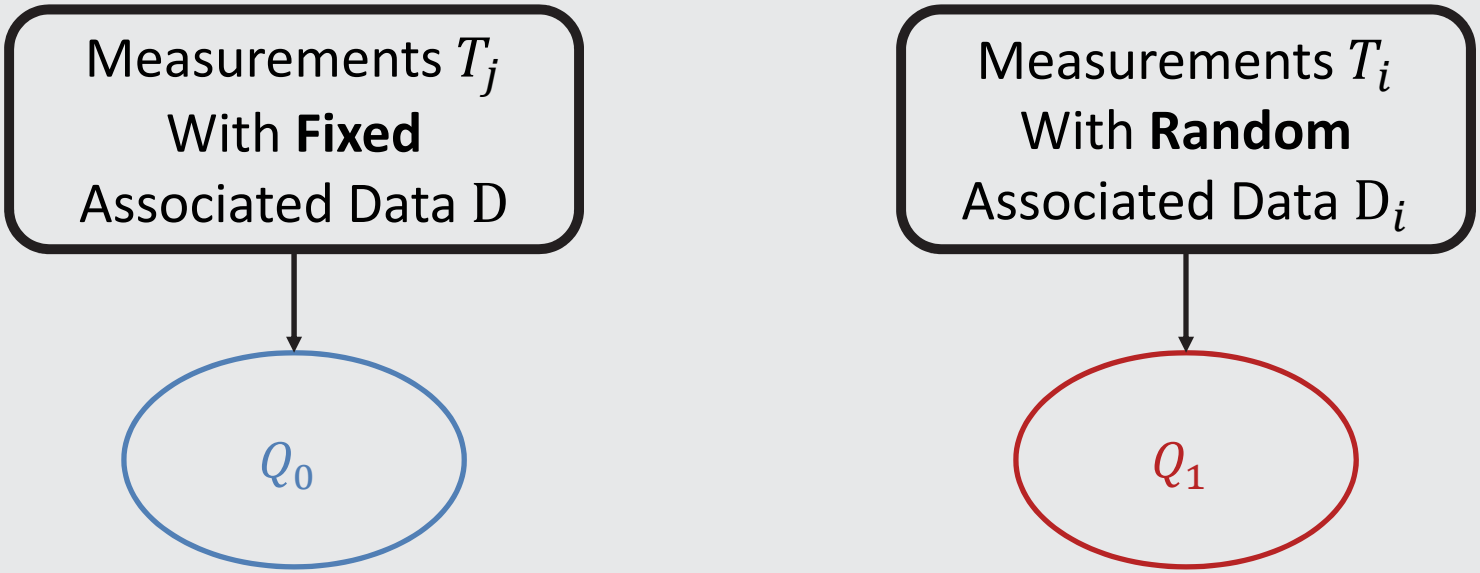
Sbox 0 nibble (16 models)



Sbox (in $\oplus$ out) bits (64 models)

Testing Methodology - Non-Specific t-Test

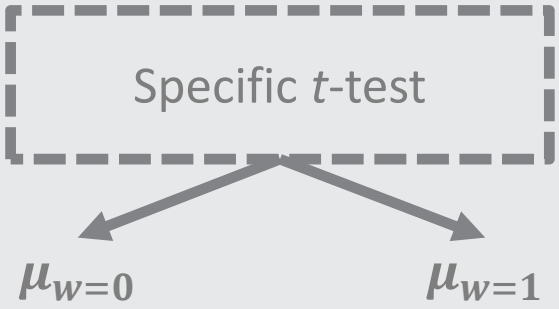
- *fixed vs. random t*-test
- Avoids being dependent on any intermediate value/model
- Needs special measurement phase:

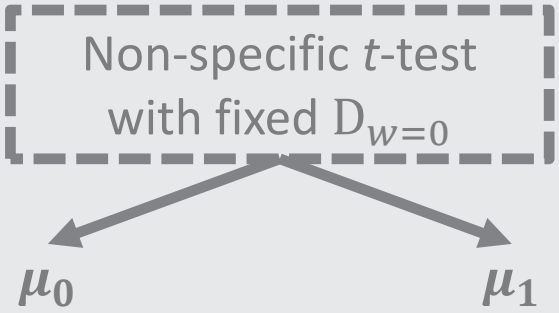


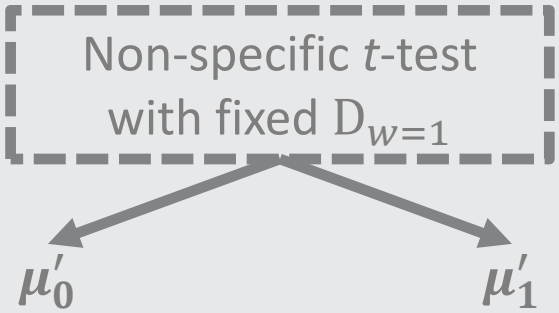
Testing Methodology - Non-Specific t-Test

Relation with *specific* *t*-test:

- Single-bit intermediate value w
- Overall mean: $\mu \approx \frac{\mu_{w=0}+\mu_{w=1}}{2}$ if $|Q_0| \approx |Q_1|$







- μ_0 close to $\mu_{w=0}$
- μ_1 close to μ

- μ'_0 close to $\mu_{w=1}$
- μ'_1 close to μ

Testing Methodology - Non-Specific t-Test

- Non-specific t-test reports a detectable leakage

  ⇒ Specific t-test reports leakage with higher confidence
- Other direction (⇐) cannot be concluded from a single non-specific *t*-test
- Recommended to perform a number of non-specific tests with different fixed data D

***Semi-fixed vs. random* test:**

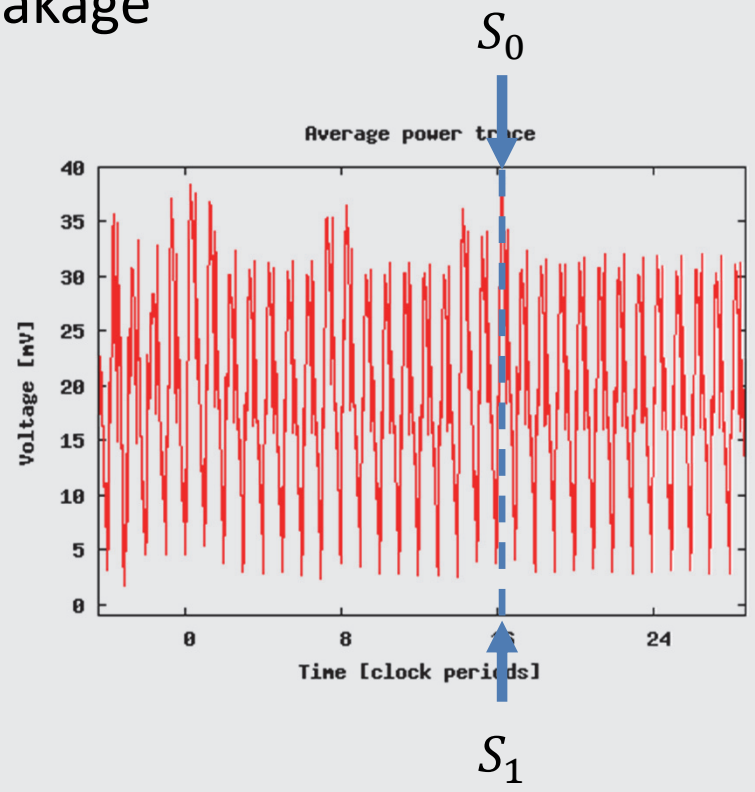
- Use a set of particular associated data D instead of D
- All lead to certain intermediate value

Higher Order Testing

- Univariate
- Multivariate

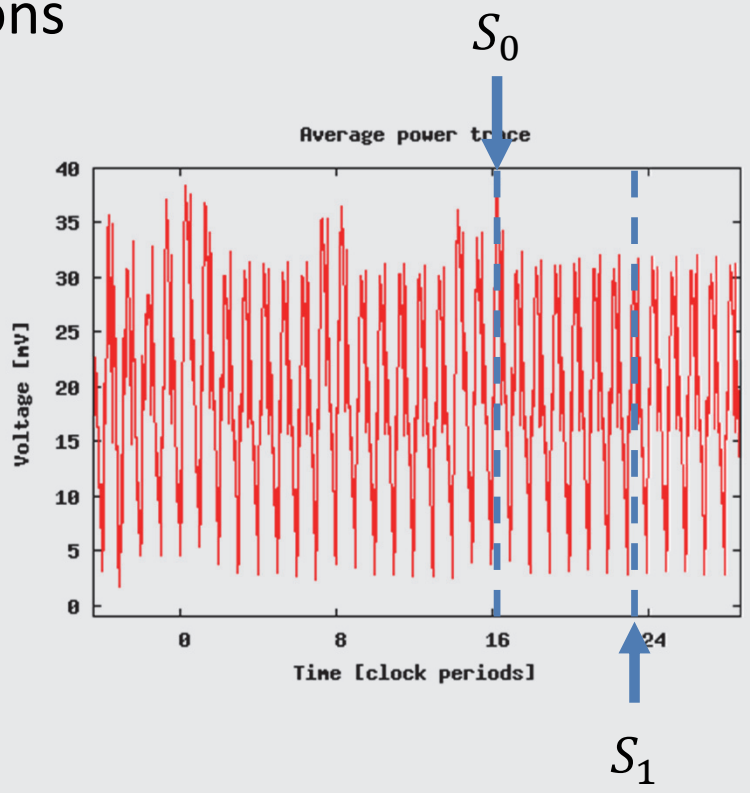
Higher Order Testing - Univariate

- Sensitive variable is masked: $S = S_0 \circ S_1$
- First-order *t*-test should not detect any leakage
- Shares are often processed in parallel in hardware circuits
- Traces need to be preprocessed
- Univariate higher-order testing:
  - 2nd-order : $(X_i - \mu_X)^2$ (centralized)
  - *d*-order: $\left(\frac{X_i - \mu_X}{s_X}\right)^d$ (standardized)



Higher Order Testing - Multivariate

- Shares are often processed at different time instances in software implementations
- Test need to consider a combination of multiple different points in time
- Finding these Points-of-Interest (POI) is computationally complex
- Different combination functions:
  - Centered product
  - 2nd-order: $(X_i - \mu_X) \cdot (Y_i - \mu_Y)$



Efficient Computation

- Naïve
- Incremental
- Raw Moments
- Central Moments
- Multivariate
- Parallelization

Efficient Computation - Naïve

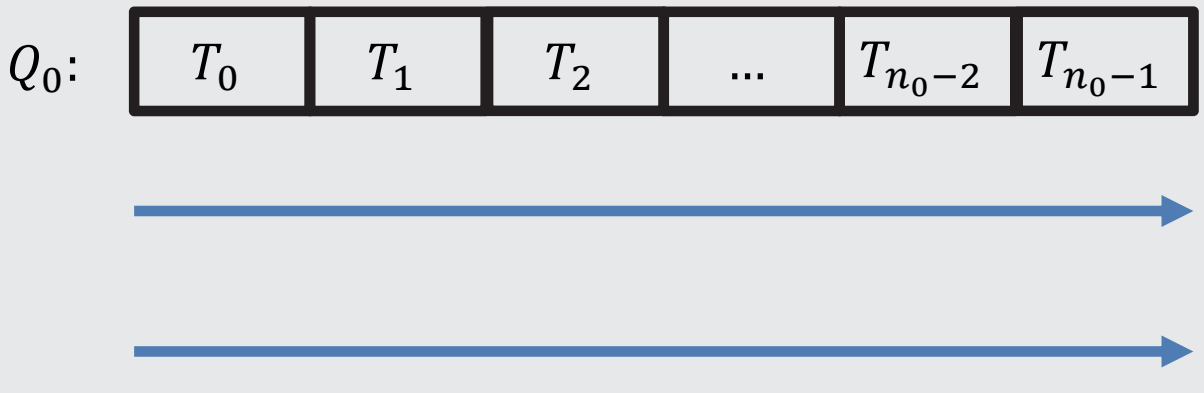
$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}}$$

Requires estimation of: (μ_0, s_0^2) (μ_1, s_1^2)

Reminder:

- $\mu = E(T)$
- $s^2 = E\left((T - \mu)^2\right)$

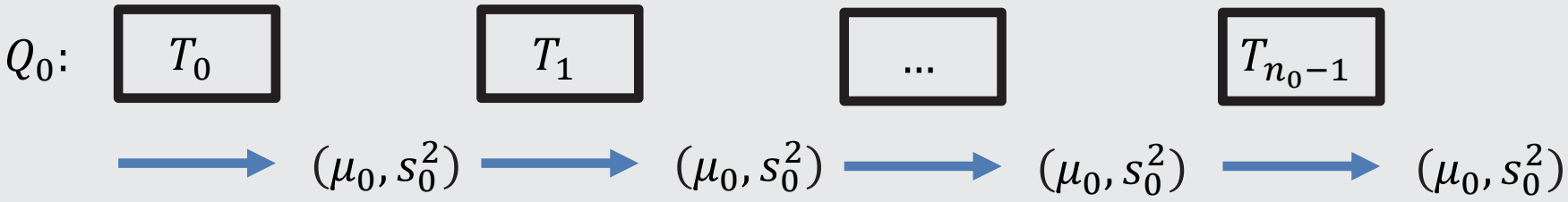
Naïve computation of (μ_0, s_0^2):



Problem: Not efficient, especially for higher orders (preprocessing)

Efficient Computation - Incremental

Idea: Update intermediate values for each new trace



Advantages:

- Requires only one pass to compute all required parameters
- Can be run in parallel to measurement phase

Efficient Computation - Raw Moments

Incremental computation of (μ_0, s_0^2) using ***raw*** moments:

- $M_d = E(T^d)$ (*raw* moments)
- $CM_d = E\left((T-\mu)^d\right)$ (*central* moments)

Idea: $\mu = M_1, \quad s^2 = CM_2 = M_2 - M_1^2 \leftarrow$ Incremental for raw moments trivial

Q_0:

First pass: M_1, M_2

Problem: Numerical unstable

- *Example:* Univariate 5th –order test requires M_{10} ... M_1

Efficient Computation - Central Moments

Incremental computation of (μ_0, s_0^2) using ***central*** moments:

Advantages:

- Efficient as other incremental algorithms
- Intermediate values are centralized ⇒ avoid numerical issues

	First Parameter	***Second Parameter***
First order:	$\mu = E(T) = M_1$	$s^2 = E\left((T-\mu)^2\right) = CM_2$
Second order (univariate):	$\mu' = E(T') = E\left((T-\mu)^2\right) = CM_2$	$s^{2\prime} = E\left((T'-\mu')^2\right) = CM_4 - CM_2^2$
$d > 2$ order (univariate):	$\mu' = E(T') = E\left(\left(\frac{T-\mu}{s}\right)^d\right) = \frac{CM_d}{\sqrt{CM_2}^d}$	$s^{2\prime} = E\left((T'-\mu')^2\right) = \frac{CM_{2d} - CM_d^2}{CM_2^d}$

Efficient Computation - Central Moments

Problem: Finding incremental formulas not as trivial as for raw moments

Idea: Incrementally compute central sums

Central sum: $CS_d = \sum_i (T_i - \mu)^d$ where $CM_d = \frac{CS_d}{n}$

For set $Q' = Q \cup \{t\}$ with $\Delta = t - M_{1,Q}$:

$$CS_{d,Q'} = CS_{d,Q} + \sum_{k=1}^{d-2} \binom{d}{k} CS_{d-k,Q} \left(\frac{-\Delta}{n}\right)^k + \left(\frac{n-1}{n}\Delta\right)^d \left[1 - \left(\frac{-1}{n-1}\right)^{d-1}\right]$$

Note: Computation of $CS_{d,Q'}$ requires $CS_{i,Q}$ for $1 < i \leq d$ and $M_{1,Q}$

Efficient Computation - Central Moments

A *t*-test of order *d* requires to estimate the central moments up to order 2*d*.

Accuracy comparison:

- Simulated traces with $\mathcal{N}(100,25)$
- 100 million traces

	1st order	2nd order	3rd order	4th order	5th order
Three Pass	25.08399	1258.18874	15.00039	96.08342	947.25523
Raw Moments	25.08399	1258.14132	14.49282	-1160.83799	-1939218.83401
Our Method	25.08399	1258.18874	15.00039	96.08342	947.25523

Efficient Computation - Multivariate

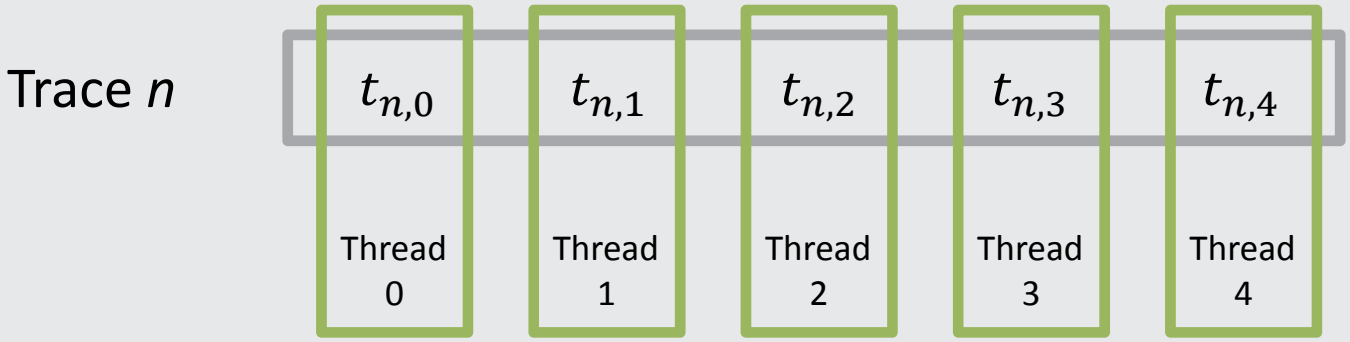
- If combination function does not use the mean, computation of the parameters is trivial (e.g., sum or product)
- Problematic for optimum combination function (centered product)

$$\prod_{j \in \mathcal{J}} (t_{i,j} - \mu_j) \quad \text{with point indices} \quad \mathcal{J}$$

- Incremental formulas for both test parameters are described in the paper

➔ Efficient incremental formulas for any variate and order of the test

Efficient Computation - Parallelization

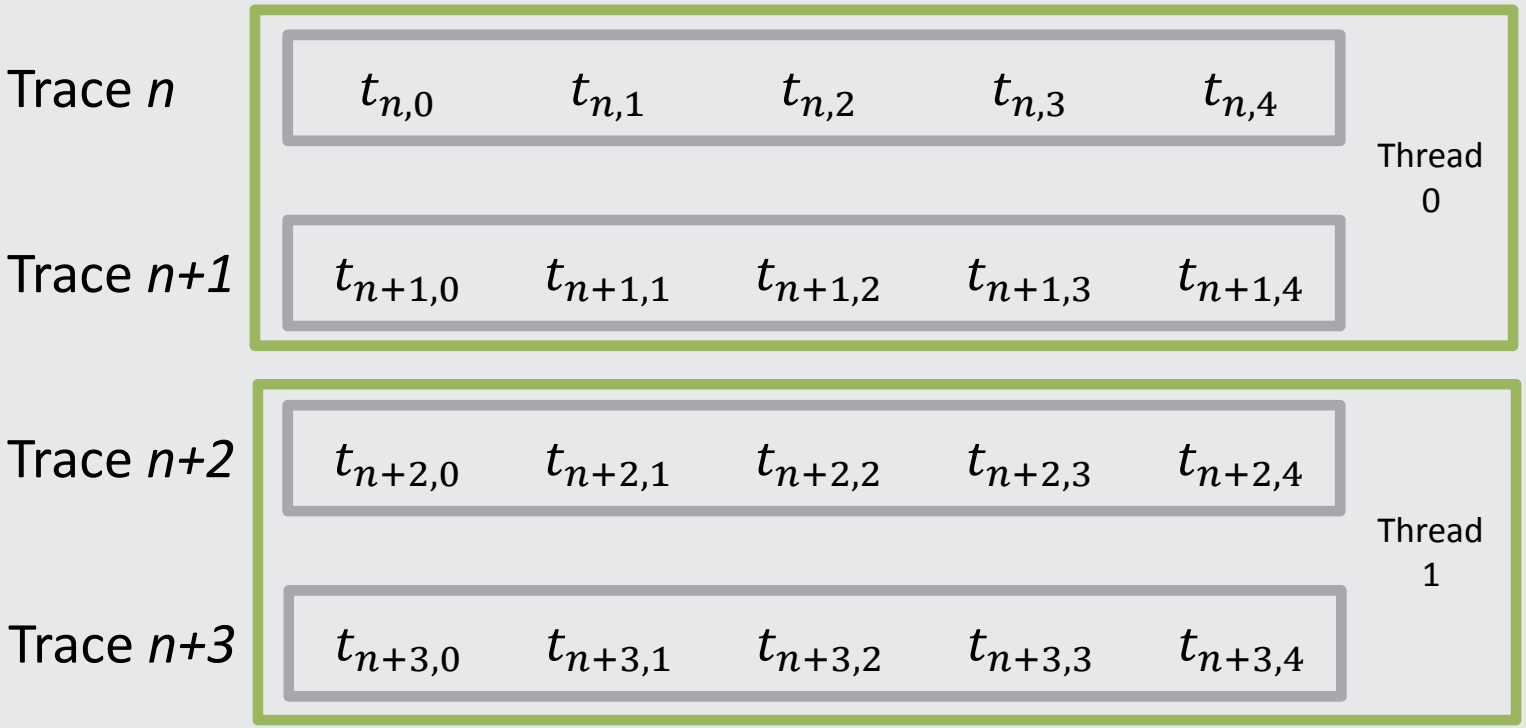


- Computations on separate points completely independent (univariate)
- No communication between threads

Example:

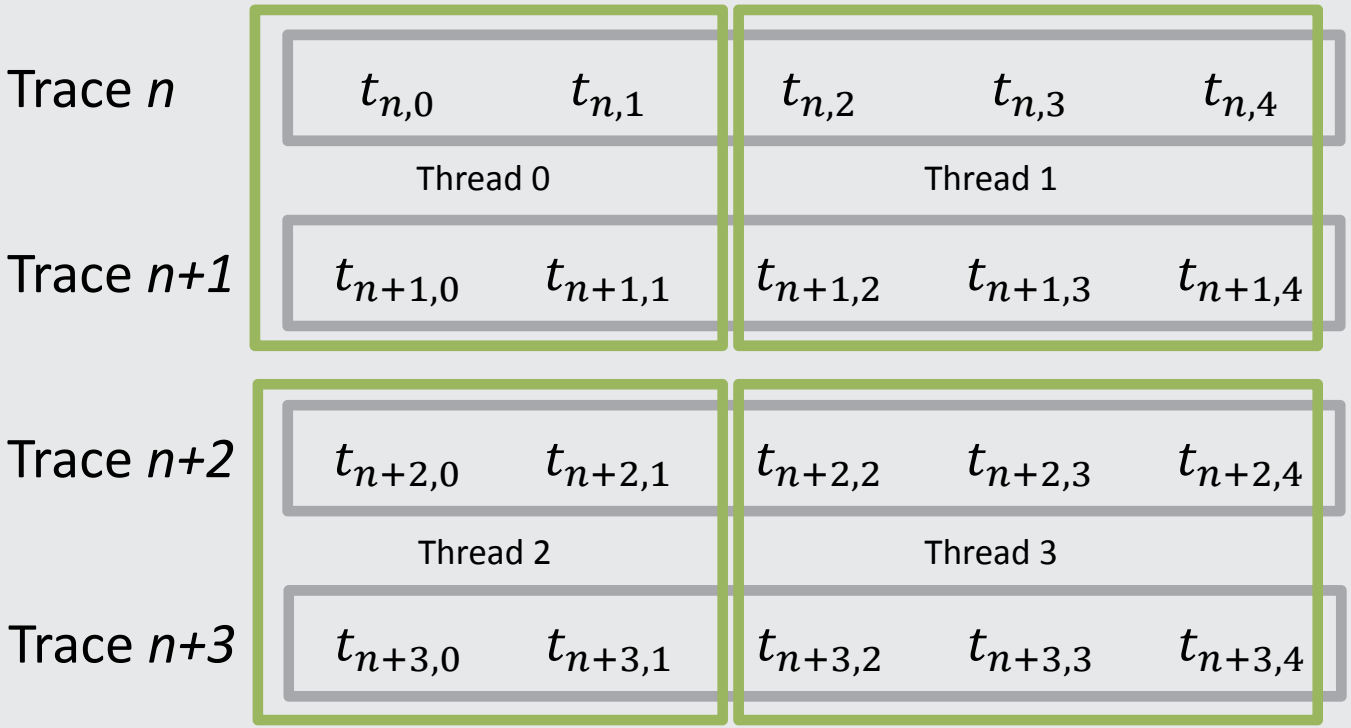
- 1st-5th order *t*-test
- 100,000,000 traces (each with 3,000 sample points)
- 9h on 2xIntel Xeon X5670 CPUs @ 2.93 GHz (24 hyper-threading cores)

Efficient Computation - Parallelization



- Useful if measurement phase already completed
- Need adjusted formulas for the central sums

Efficient Computation - Parallelization



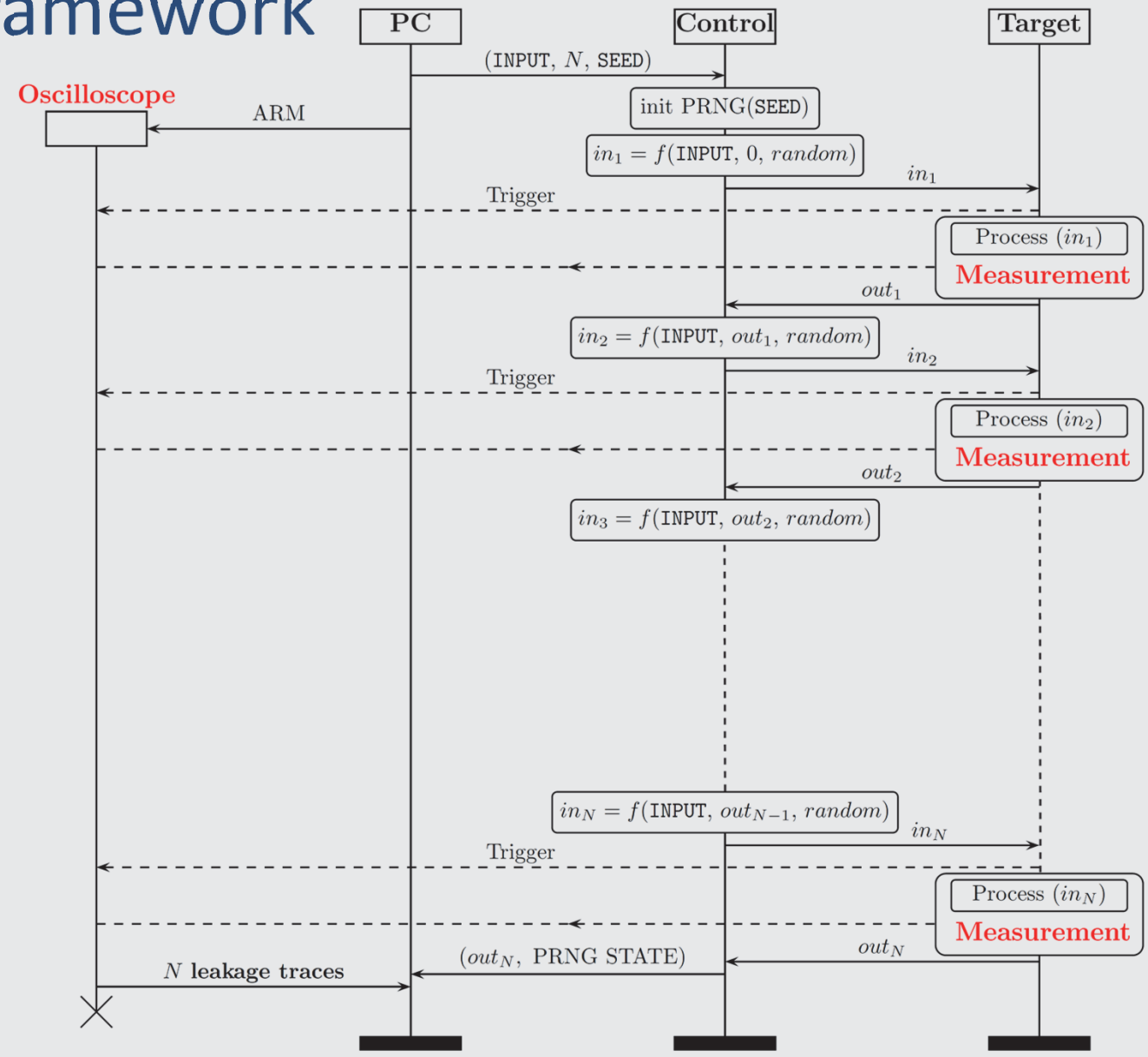
- Possible to combine both approaches for maximum performance

Case Studies

- Framework
- Case Study: Microcontroller
- Case Study: FPGA

Case Studies - Framework

- Malpractice in Measurement Phase can lead to faulty results
- Use *sequence/rapid block mode* to speed up Measurement Phase
- Random order for non-specific tests



Case Studies - Framework

Example Function Non-Specific:

$$in_{i+1} = f(\texttt{INPUT}, out_i, random) = \begin{cases} \texttt{INPUT} & \text{if } random_{bit} \text{ is } 0 \\ random & \text{if } random_{bit} \text{ is } 1 \end{cases}$$

Example Function Non-Specific (Shared):

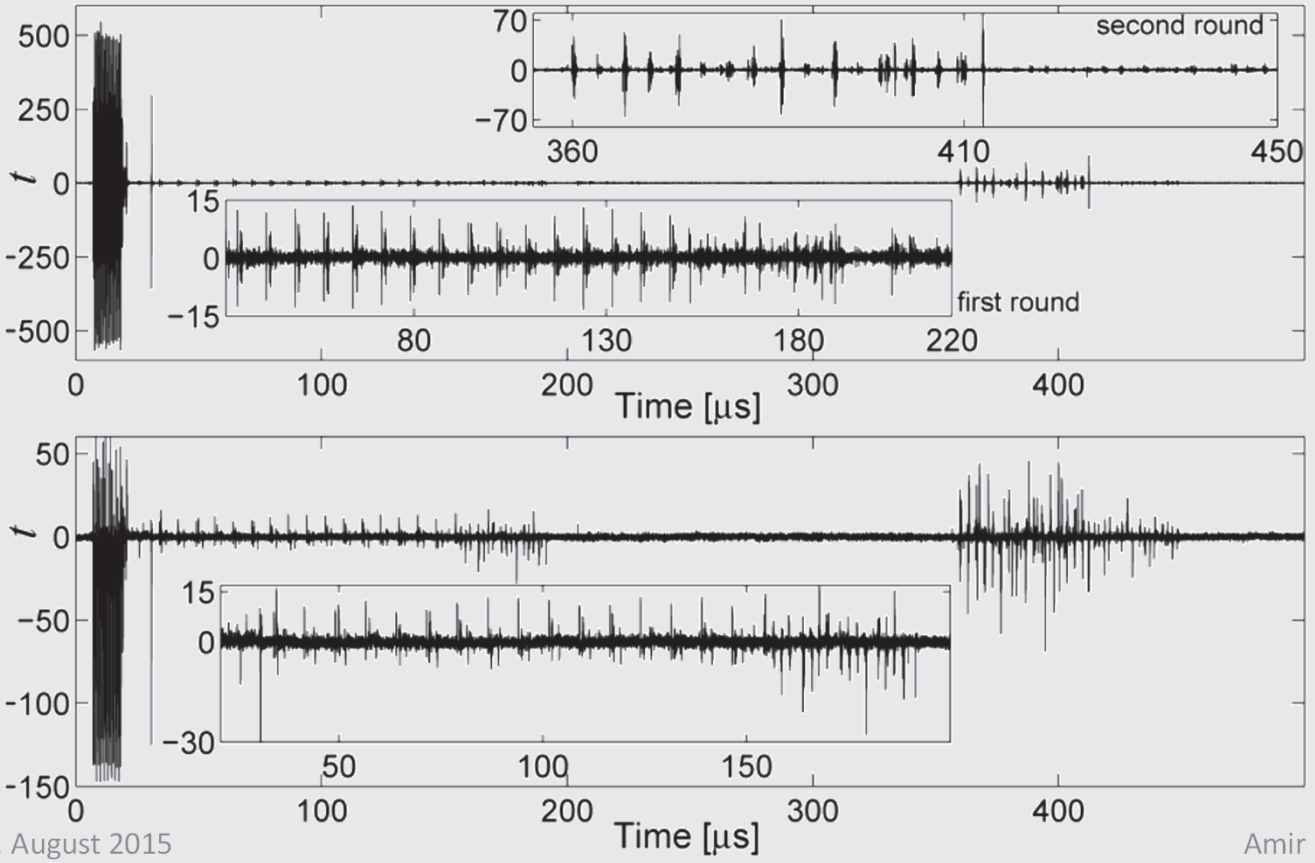
$$\begin{aligned} in_{i+1} &= f(\texttt{INPUT}, out_i, random) \\ &= \begin{cases} (\texttt{INPUT}^1 \oplus r^{.1}, \texttt{INPUT}^2 \oplus r^{.2}, \texttt{INPUT}^3 \oplus r^{.1} \oplus r^{.2}) & \text{if } random_{bit} \text{ is } 0 \\ (r^{.1}, r^{.2}, r^{.3}) & \text{if } random_{bit} \text{ is } 1 \end{cases} \end{aligned}$$

Recommendations:

- Fixed vs. random: with shared communication if DUT involves masking countermeasures
- Semi-fixed vs. random: without shared communication if DUT has hiding
- Specific *t*-test:
  - Identify suitable intermediate values for key-recovery
  - DUT has no countermeasure
  - Failed in former non-specific tests

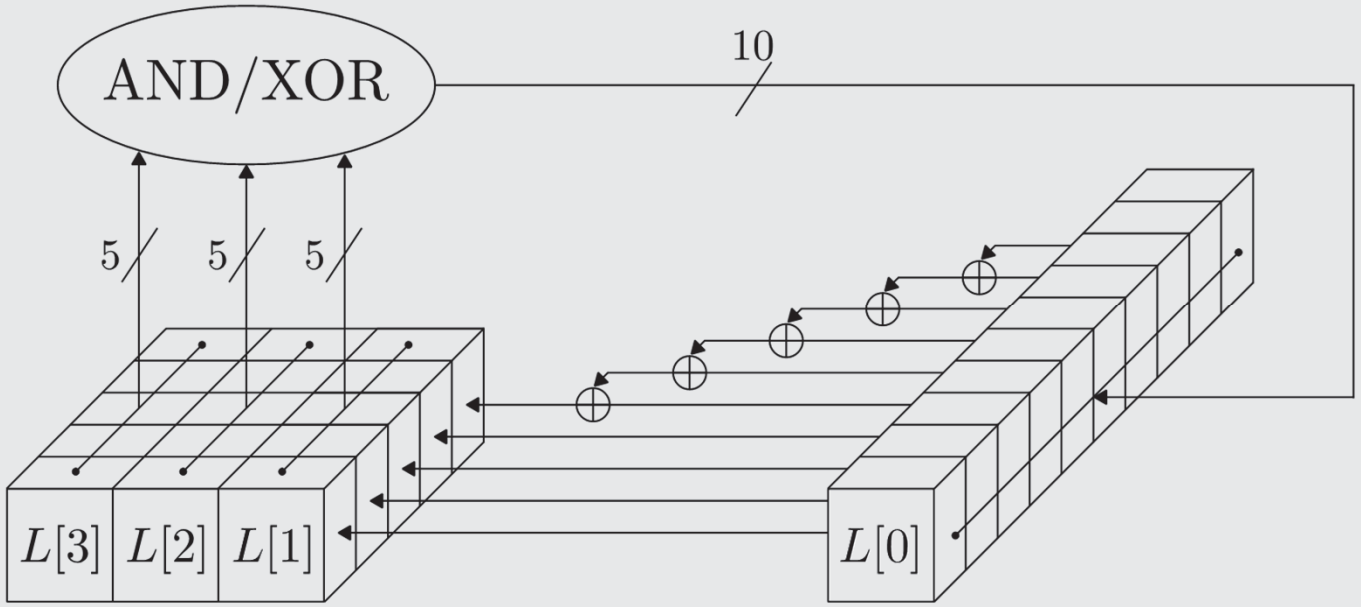
Case Studies - Microcontroller

- DPA contest v4.2 for an Atmel microcontroller
- AES-128 with low-entropy masking & shuffling
- PicoScope @ Sampling rate: 250 MS/s 100,000 traces

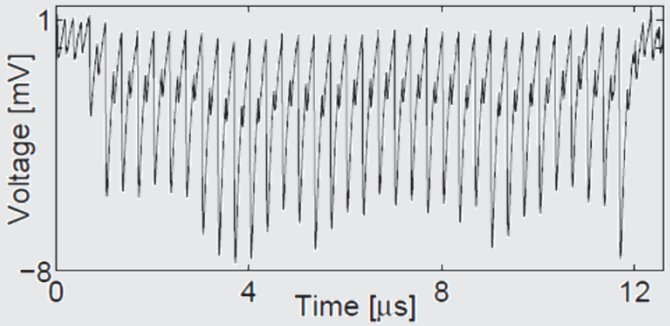


Case Studies - FPGA

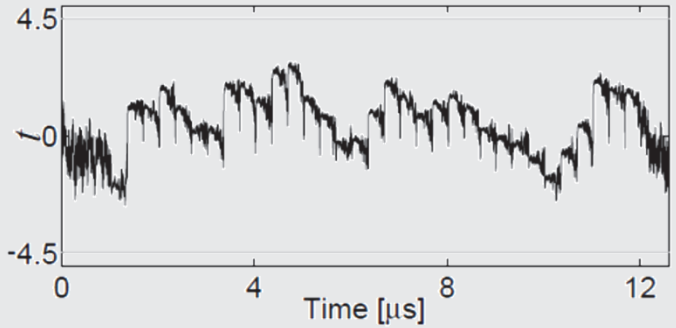
- TI-NLFSR on Spartan-6 FPGA
- 5 shares
- 2,000,000 power traces with 500 MS/s
- AND/XOR = $L[3] \oplus (L[2] \cdot L[1])$



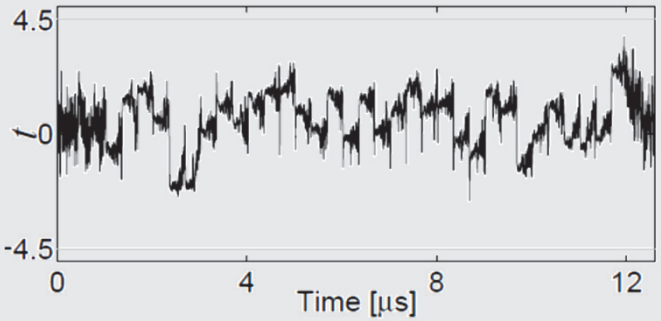
Case Studies - FPGA



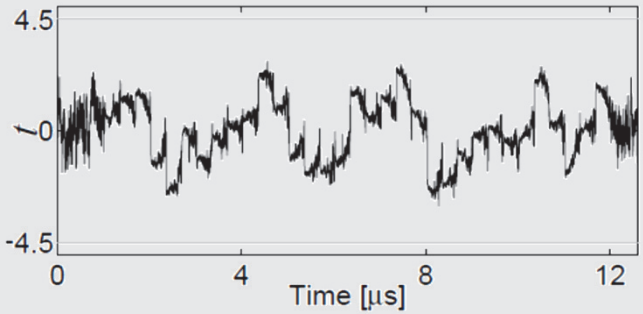
(a) Sample Trace



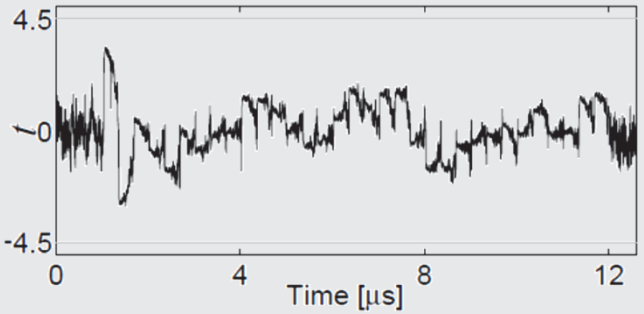
(b) first-order



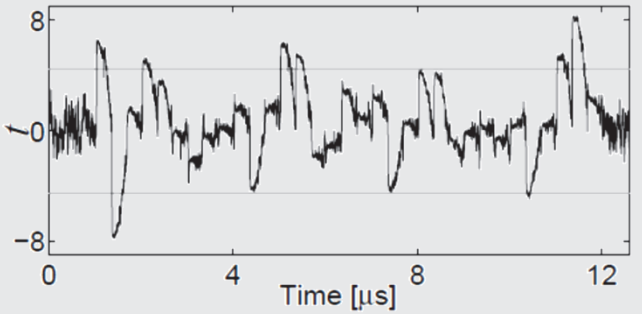
(c) second-order



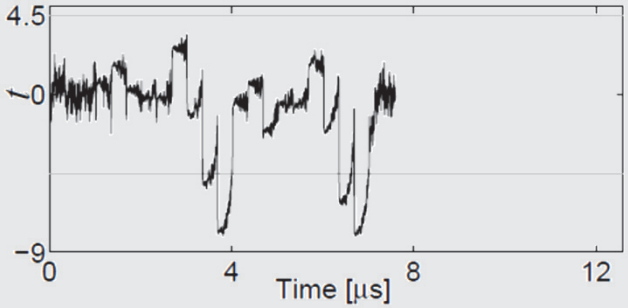
(d) third-order



(e) fourth-order



(f) fifth-order



(g) bivariate second-order, 15 clock cycles offset

Conclusion

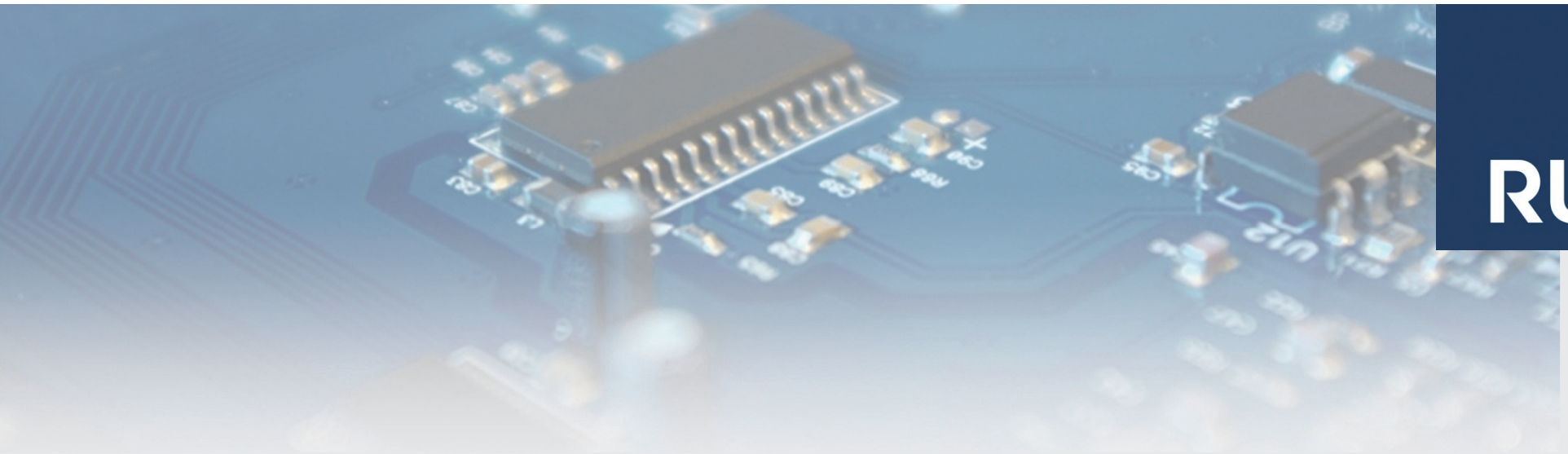
Conclusion

- *t*-test for security evaluation has become popular
- Extended the theoretical foundation guidelines
- Detailed instructions how to perform the test correctly and efficient in practice at any order or variate
- Optimized and correct measurement setup

"Future Work":

- Extension of the incremental approach to correlation-based evaluation schemes

Robust and One-Pass Parallel Computation of Correlation-Based Attacks at Arbitrary Order
Tobias Schneider, Amir Moradi, Tim Güneysu, ePrint Report 2015/571



RUB

Thanks!
any questions?

amir.moradi@rub.de

Embedded Security Group, Ruhr-Universität Bochum, Germany

hgi Lehrstuhl für Embedded Security