Leakage Assessment Methodology

- a clear roadmap for side-channel evaluations -

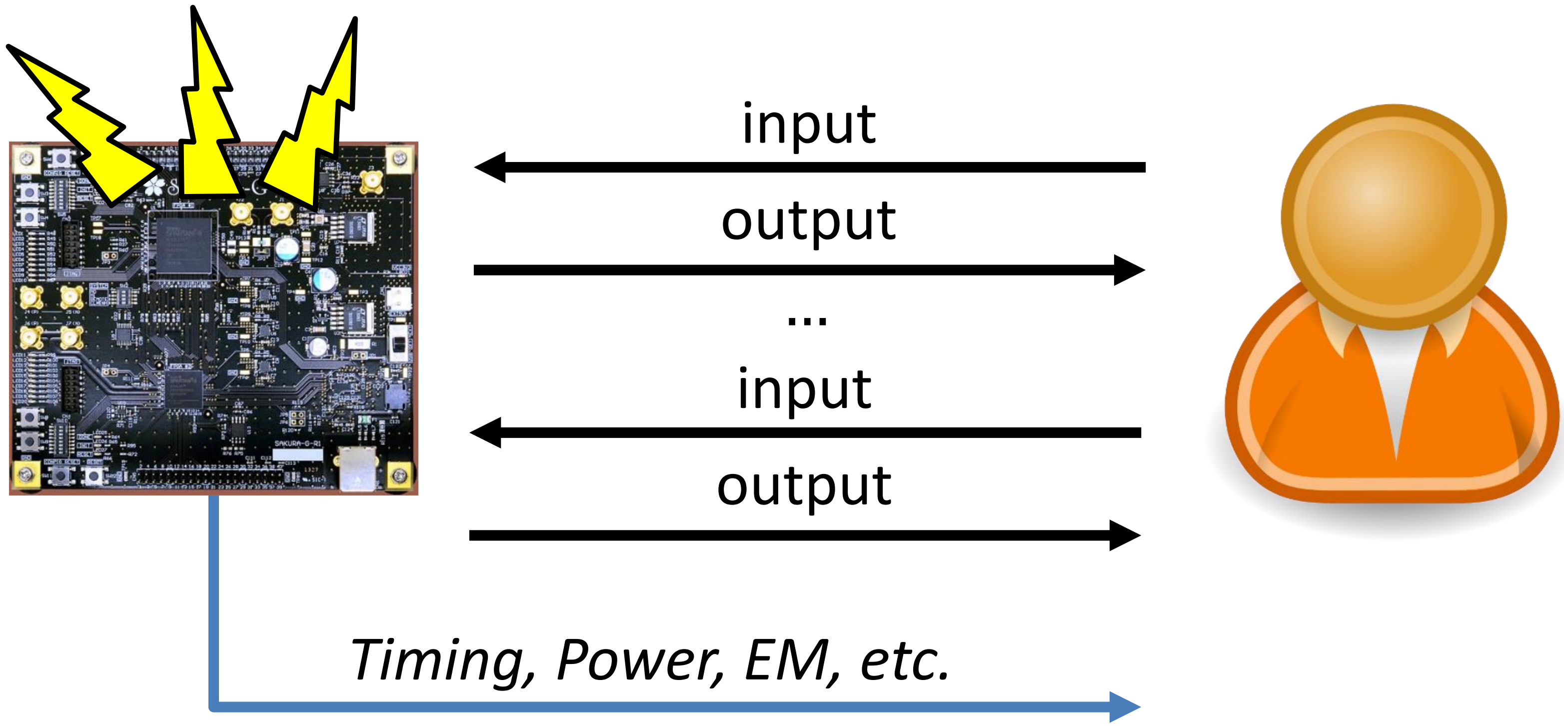
Tobias Schneider and Amir Moradi

Friday, September 11th, 2015

Horst Görtz Institute for IT-Security

RUHR-UNIVERSITÄT BOCHUM

Motivation Physical Attacks & Countermeasures



Countermeasures:

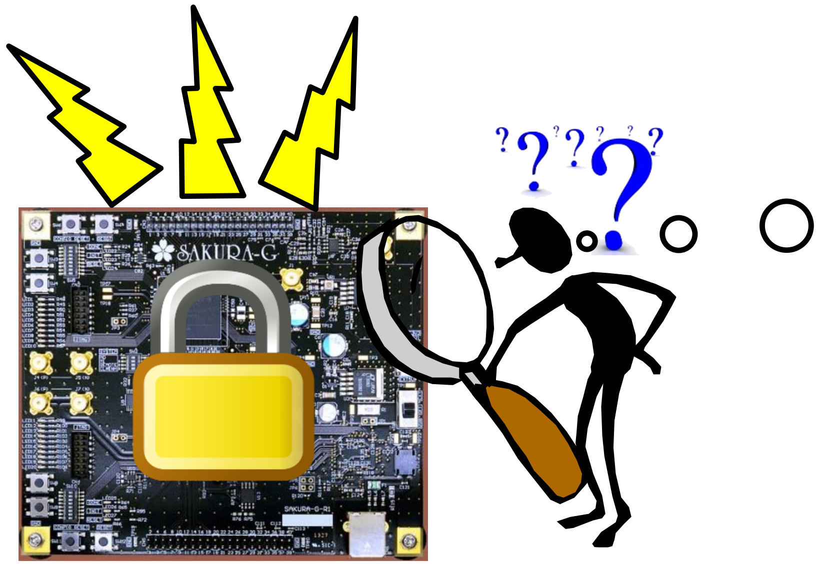
- Masking → Higher-order Attacks
- Hiding

Higher-order Attacks

Univariate

Multivariate

Motivation Security Evaluation



Does the chip leak information?

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
- …

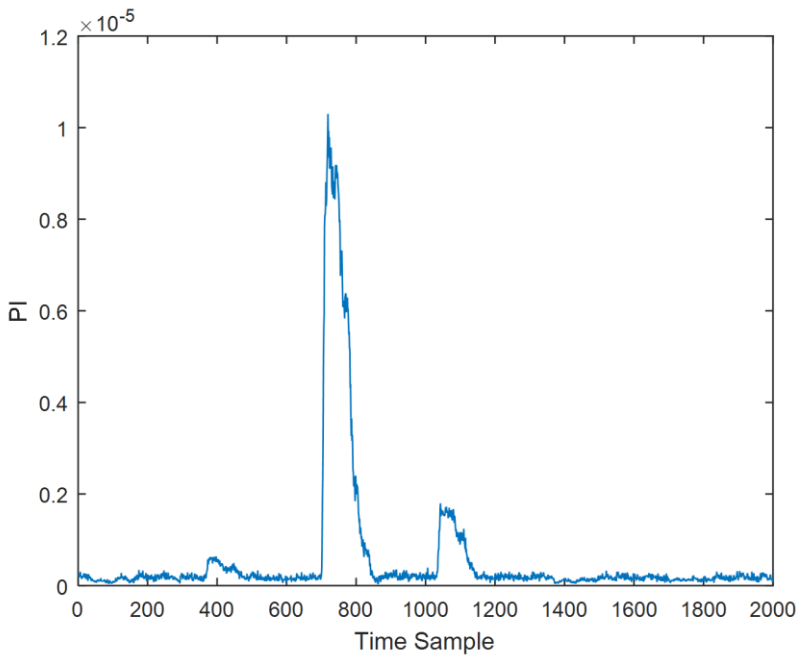
×

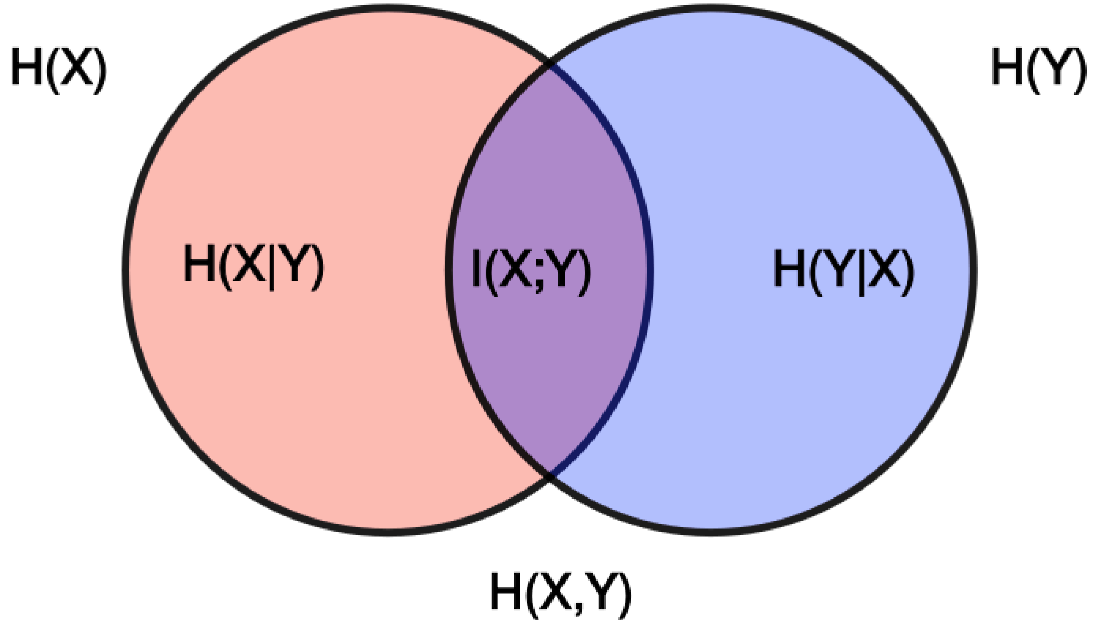
Leakage Models:
- HW
- HD
- Bit
- …

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:

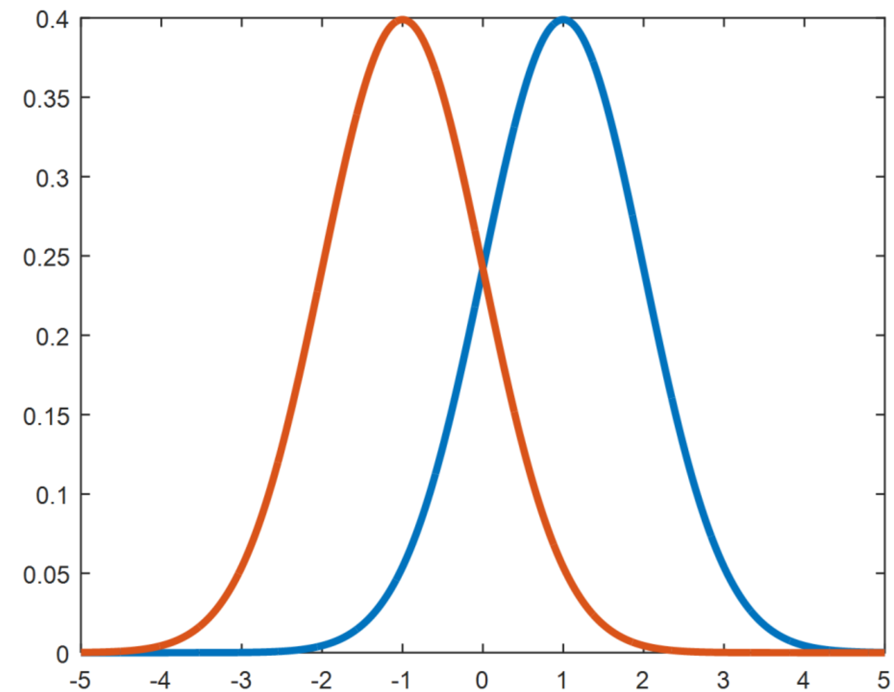
- High computational complexity
- Cannot focus on one statistical moment
- Dependent on density estimation
- Does not cover all possible attack vectors

Motivation Testing based on *t*-Test

Tries to detect any type of leakage at a certain order

CRYPTOGRAPHY™ RESEARCH
a division of Rambus

- Proposed by CRI at NIST workshop



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

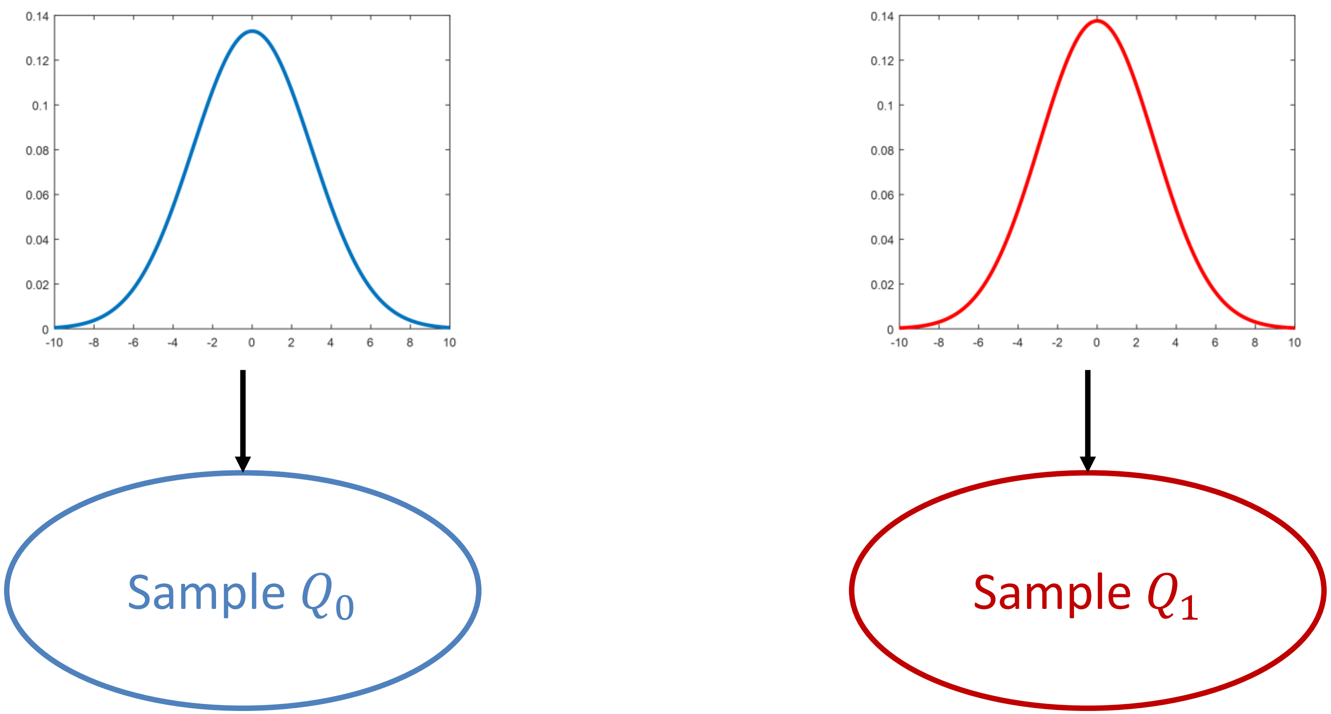
Outline

1. Statistical Background
2. Testing Methodology
3. Correct Measurement
4. Efficient Computation
5. Conclusion

Statistical Background

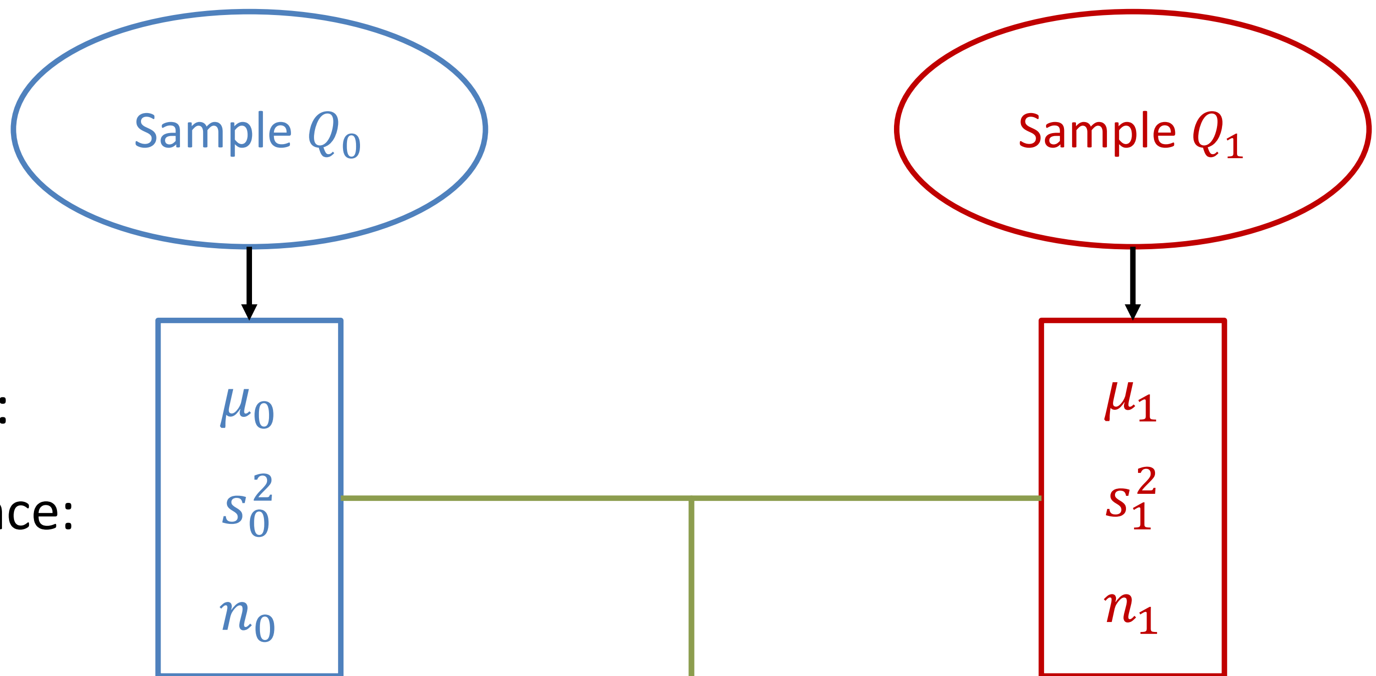
- *t*-Test

Statistical Background *t*-Test



Null Hypothesis: Two population means are equal.

Statistical Background *t*-Test



Sample mean:

Sample variance:

Sample size:

Degree of freedom →

$$v = \frac{\left(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}\right)^2}{\frac{\left(\frac{s_0^2}{n_0}\right)^2}{n_0 - 1} + \frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1}} \qquad t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}}$$

← t-test statistic

Statistical Background *t*-Test

t v

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

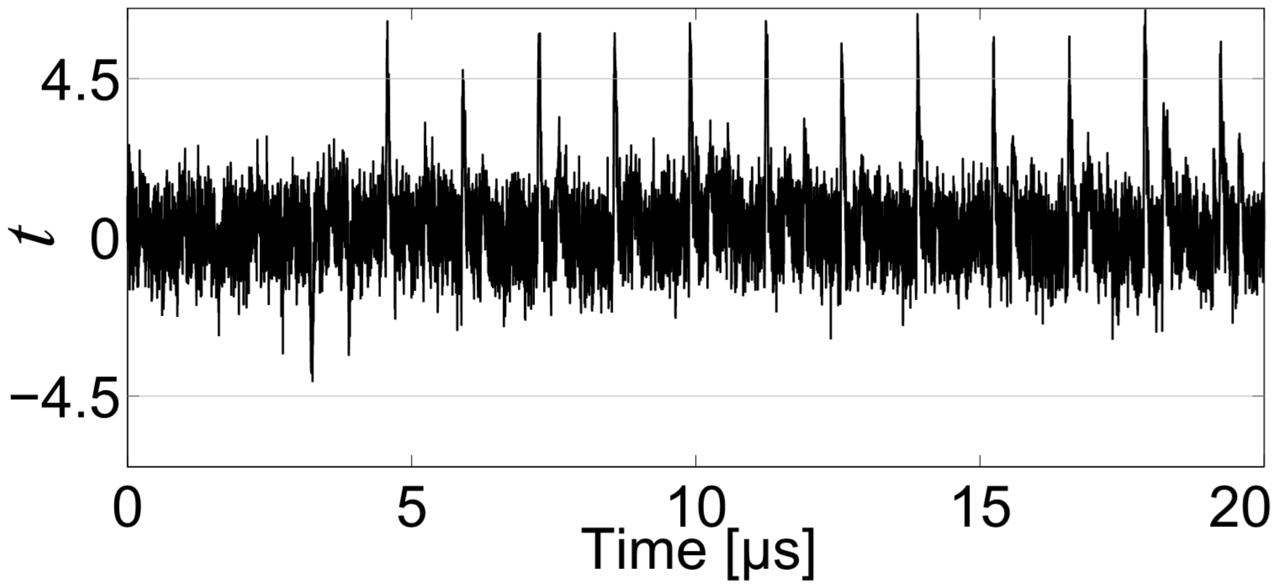
$$f(t, v) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{\pi v}\,\Gamma\left(\frac{v}{2}\right)}\left(1 + \frac{t^2}{v}\right)^{-\frac{v+1}{2}}$$

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

Small p values give evidence to reject the null hypothesis

Statistical Background *t*-Test

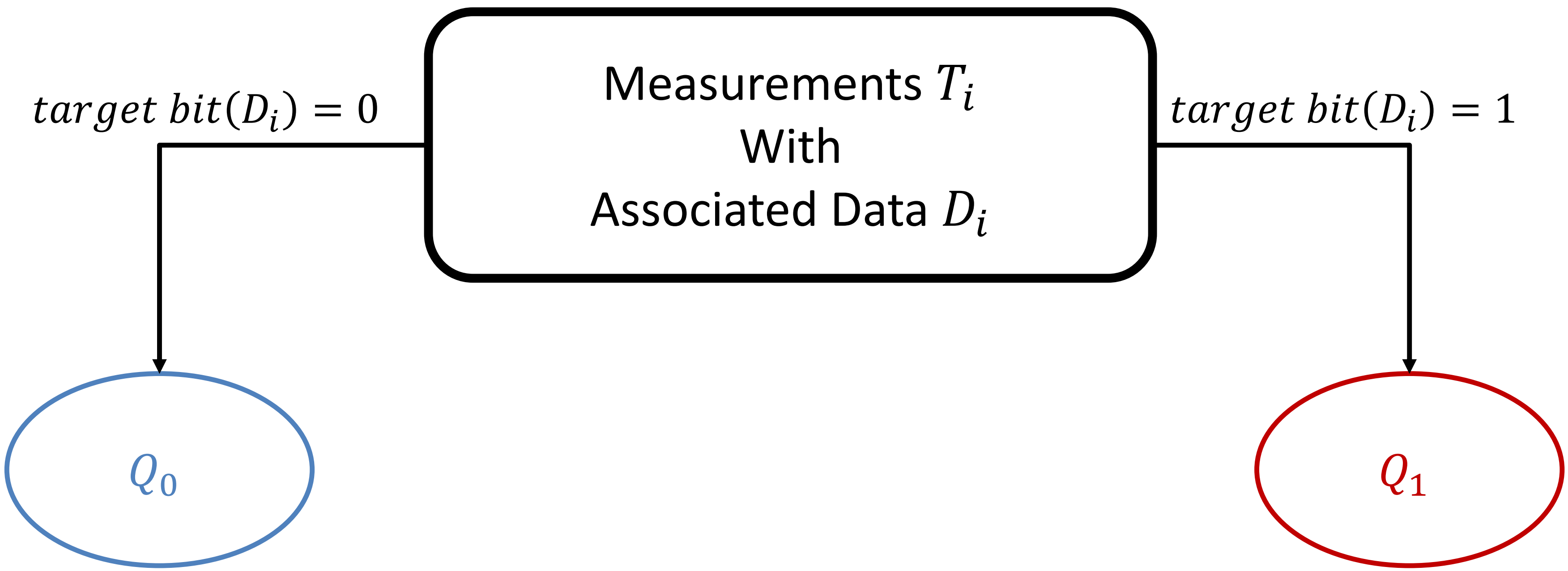
- For testing usually only the t-value is estimated
- Compared to a threshold of $|t| > 4.5$
  - $p = 2F(-4.5, v > 1000) < 0.00001$
  - Confidence of > 0.99999 to reject the null hypothesis



Testing Methodology

- Specific t-Test
- Non-Specific t-Test
- Higher Orders

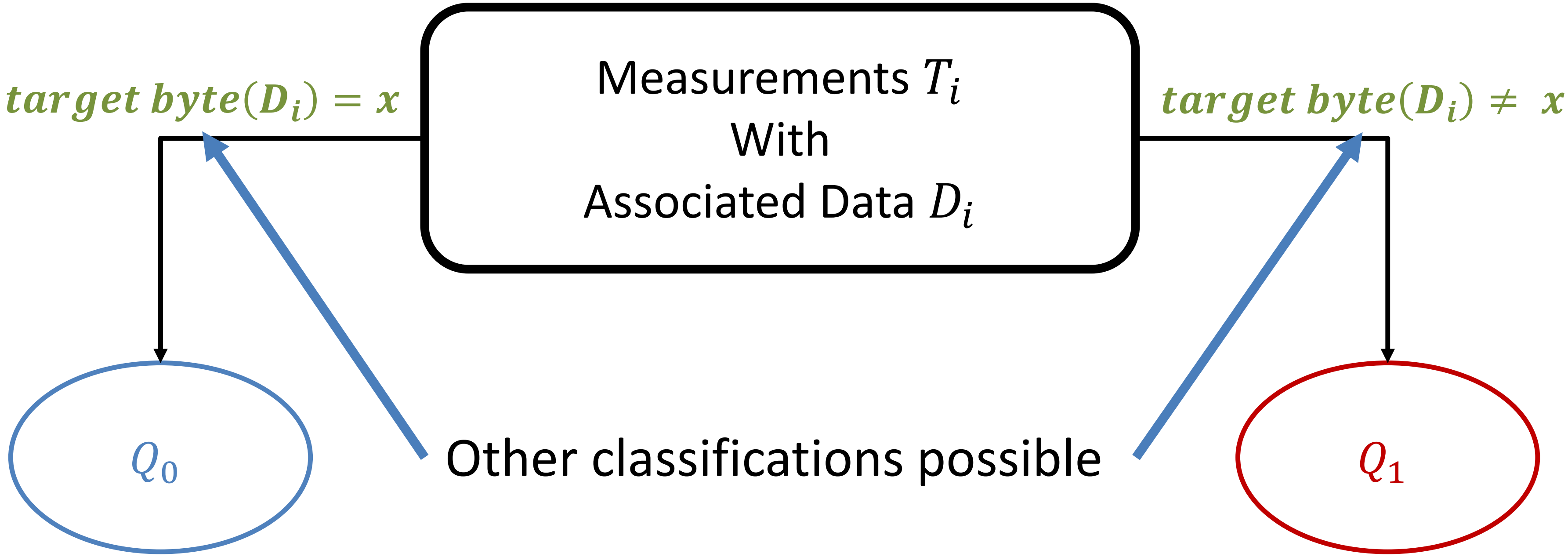
Testing Methodology Specific *t*-Test



Specific *t*-Test:

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

Testing Methodology Specific *t*-Test



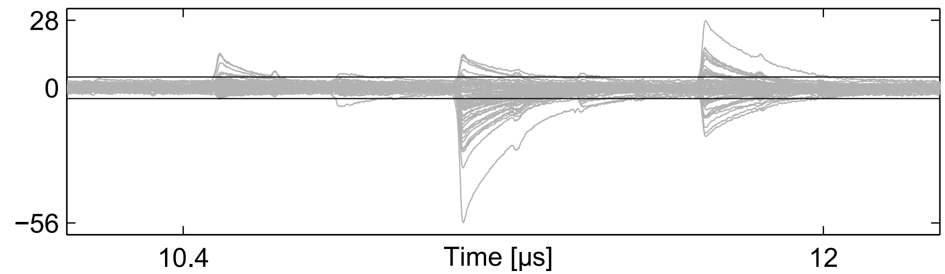
Specific *t*-Test:

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

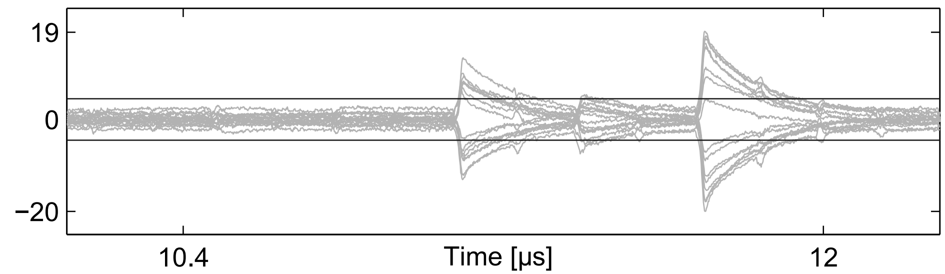
Testing Methodology Specific *t*-Test

Example: PRESENT (last round)

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



Sbox out bits (64 models)



Sbox 0 nibble (16 models)

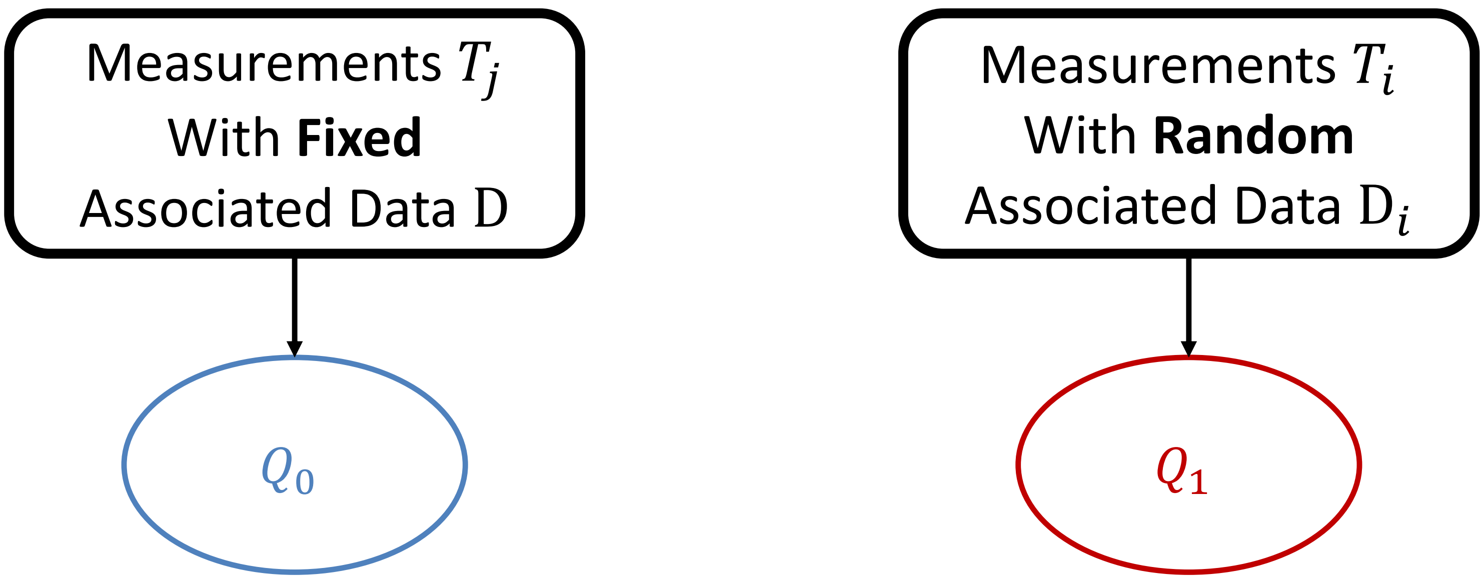
Problems:

- Same as attack-based approach
- Many different intermediate values
- Many different models

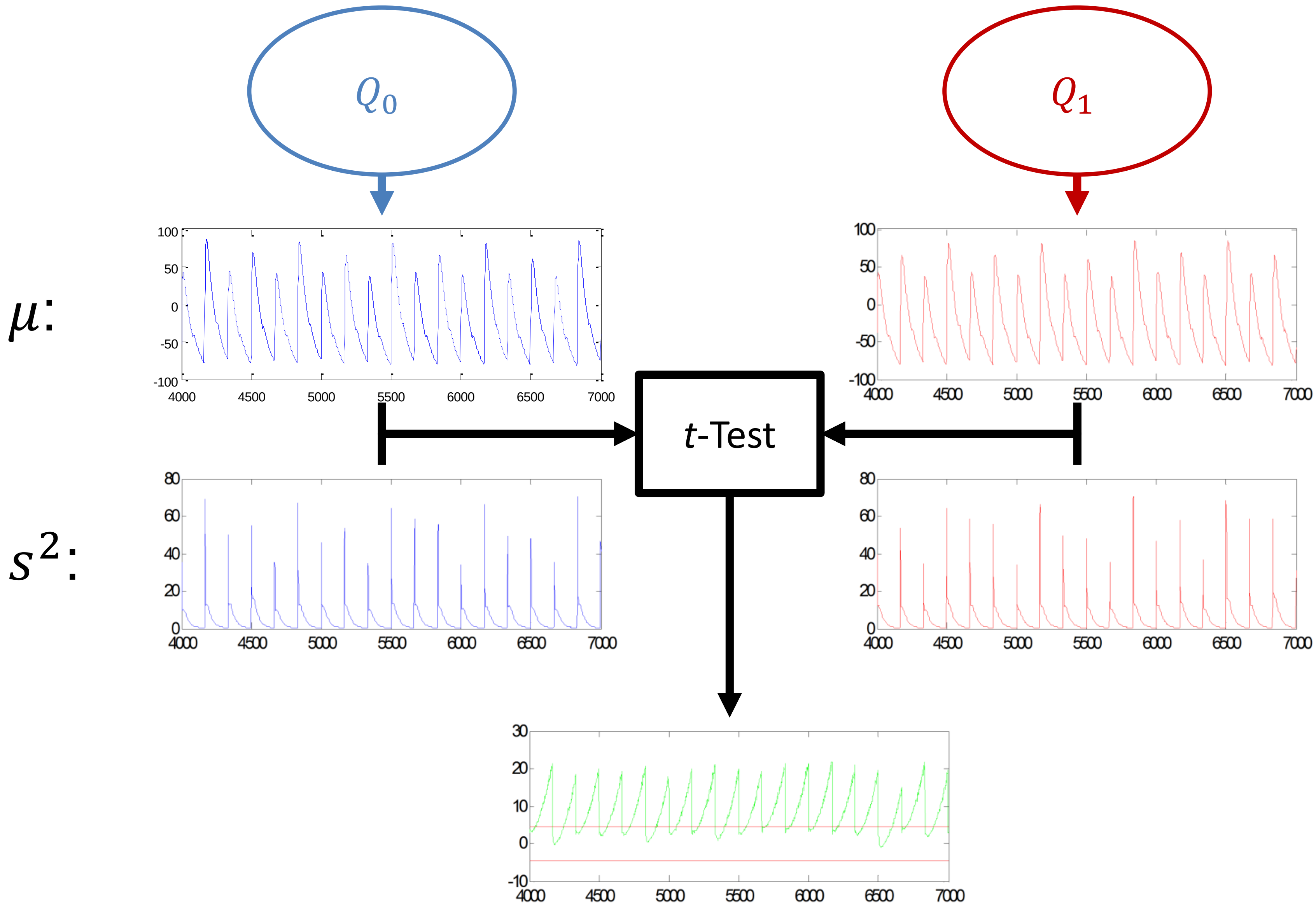
Testing Methodology Non-Specific *t*-Test

Non-Specific *t*-Test:

- *fixed vs. random t*-test
- Avoids being dependent on any intermediate value/model
- Detected leakage of single test is not always exploitable



Testing Methodology Non-Specific t-Test



Testing Methodology Non-Specific *t*-Test

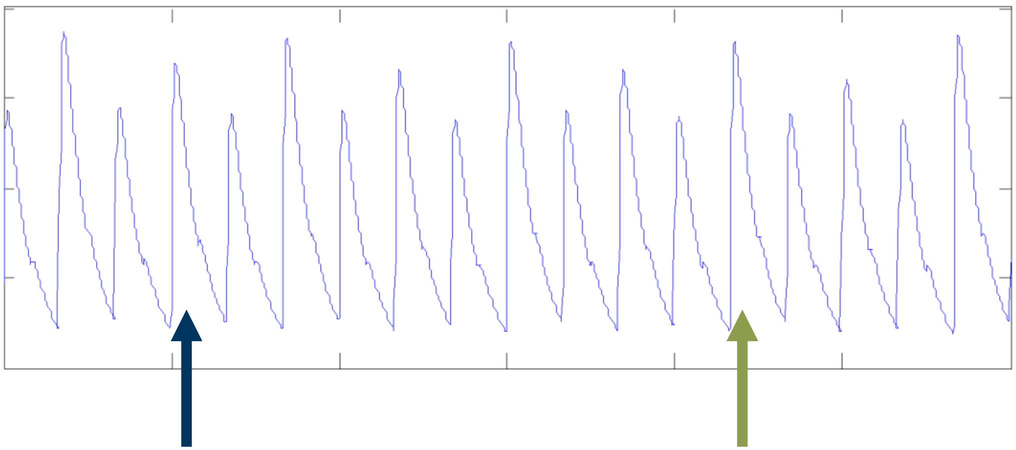
- Non-specific t-test reports a detectable leakage

  ⇒ 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

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

- Use a set of particular associated data instead of only one
- All lead to certain intermediate value
- Eliminates some of the drawbacks of *fixed vs. random*

Testing Methodology Higher Orders

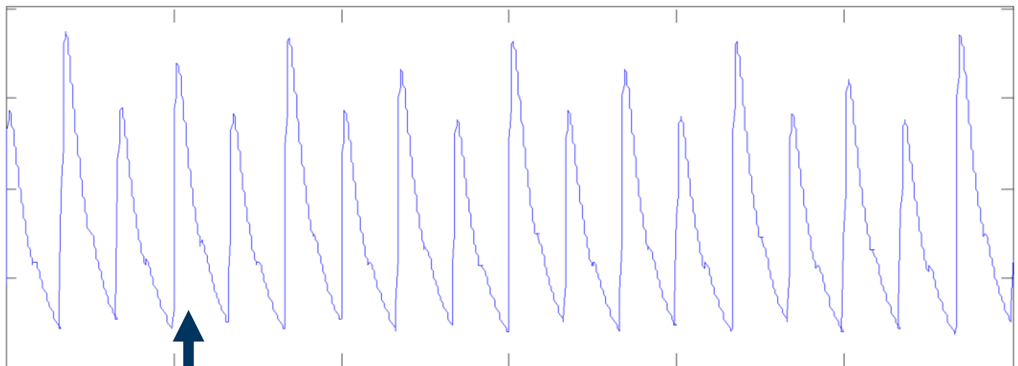


Multivariate: S_1 S_2

- Sensitive variable is shared: $S = S_1 \circ S_2$
- Shares are processed at different time instances (SW)
- Leakages at different time instances need to be combined first

$$\boxed{\text{Centered Product: } x' = (x_1 - \mu_1) \cdot (x_2 - \mu_2)}$$

Testing Methodology Higher Orders



S_1 S_2

Univariate:

- Shares are processed in parallel (HW)
- Leakages at the same time instance need to be combined first

Variance: $x' = (x - \mu)^2$

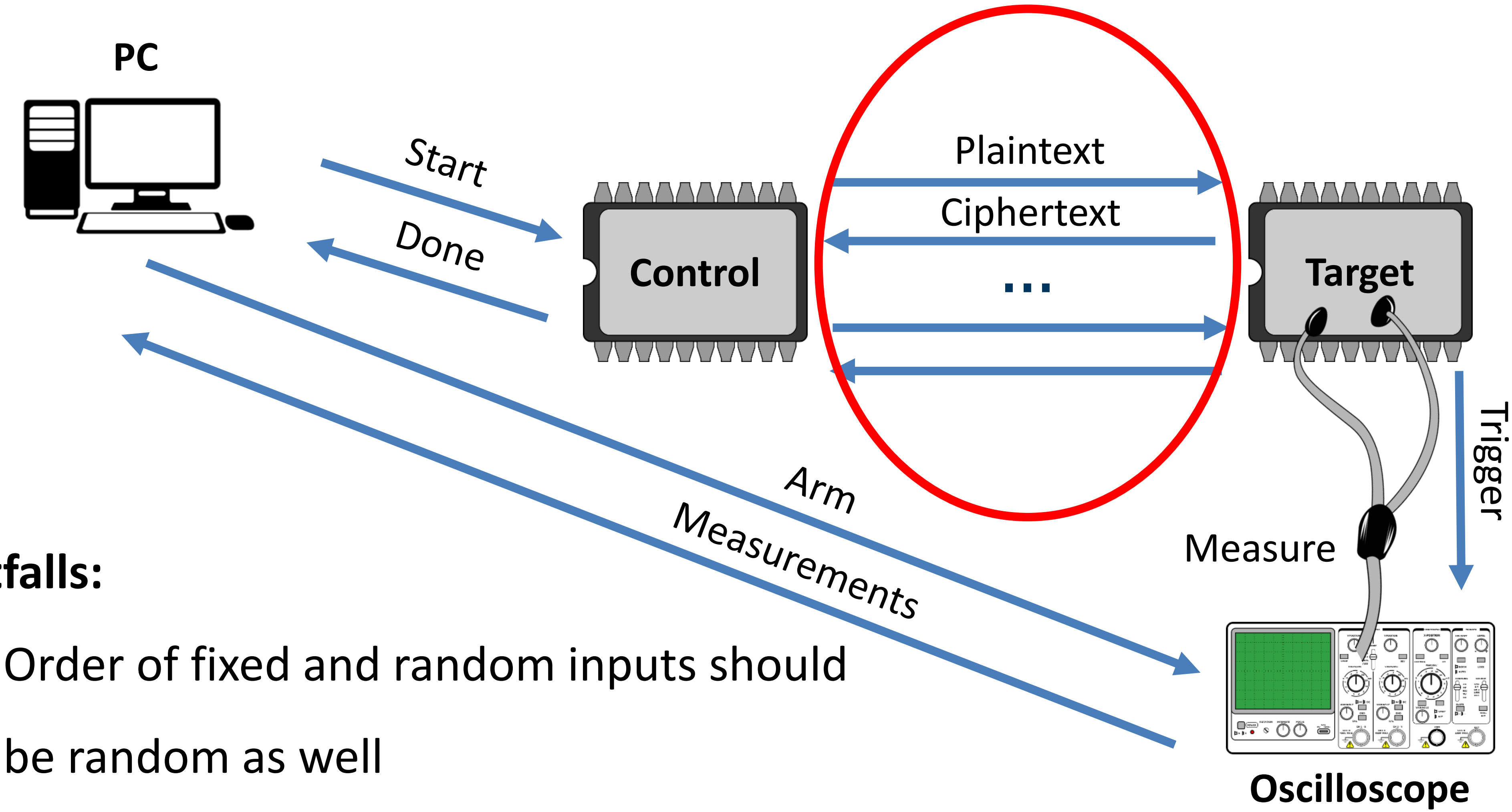
In general: $x' = (x - \mu)^d$

In some cases: $x' = \left(\frac{x-\mu}{s}\right)^d$

Correct Measurement

- Setup
- Case Study: Microcontroller
- Case Study: FPGA
- Recommendations

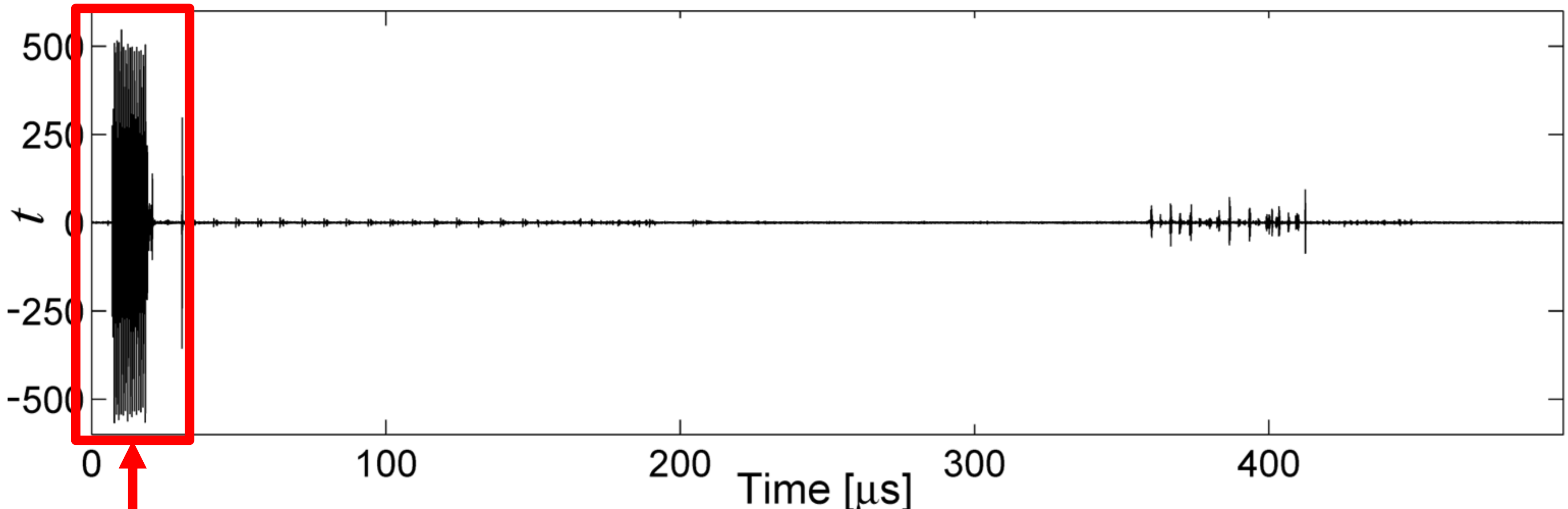
Correct Measurement Setup



Pitfalls:

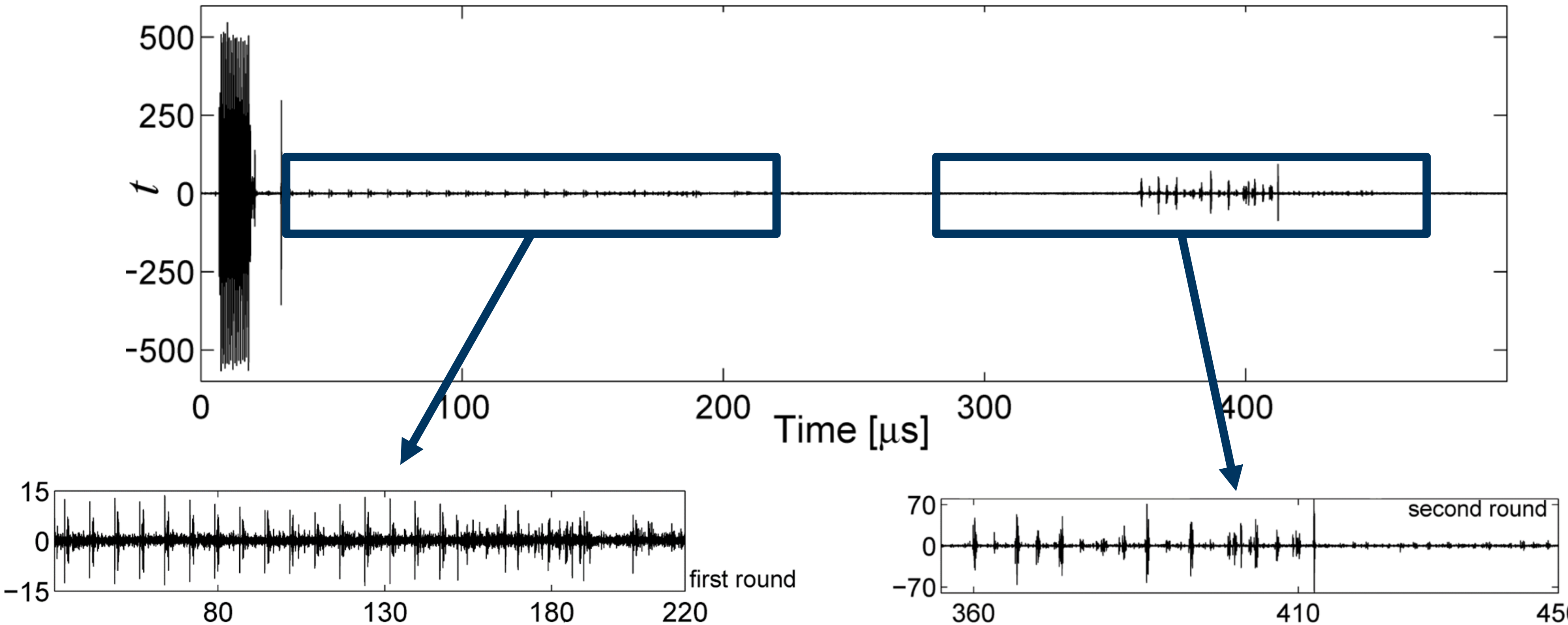
- Order of fixed and random inputs should be random as well
- Communication between **Control** and **Target** should be masked (if possible)

Correct Measurement CS: Microcontroller



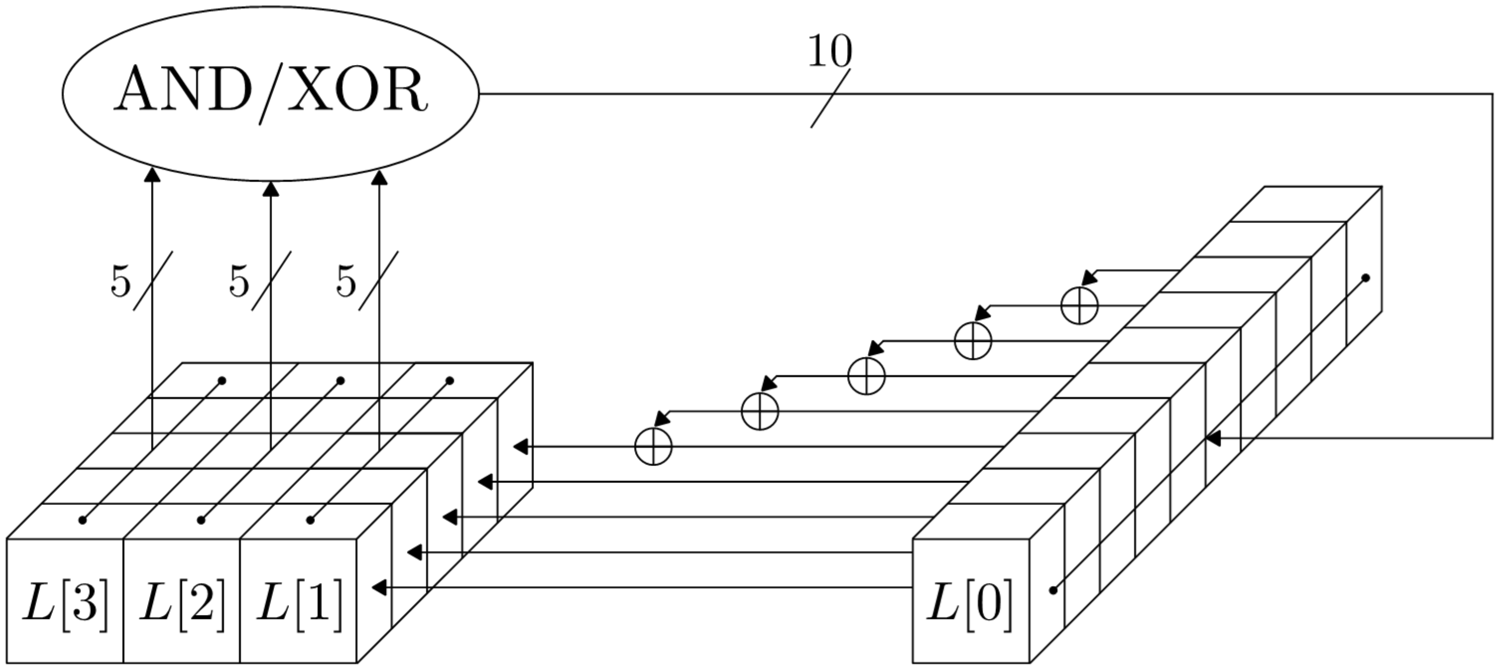
- AES with masking & shuffling (DPA contest v4.2)
- **No shared communication**
- First-order test
- **Leakage associated to unmasked plaintext**

Correct Measurement CS: Microcontroller



Detectable first order leakage

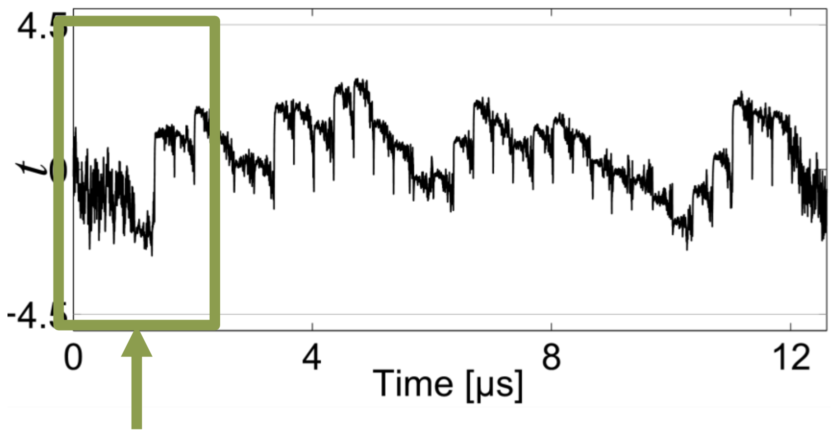
Correct Measurement CS: FPGA



- NLFSR [1]
- 2nd –order threshold implementation
- Test at different orders

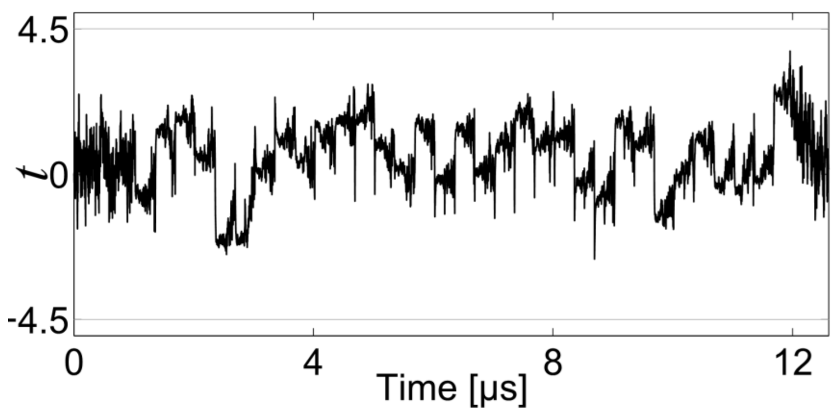
[1] *A note on the security of Higher-Order Threshold Implementations*
Oscar Reparaz, ePrint Report 2015/001

Correct Measurement CS: FPGA



No plaintext leakage

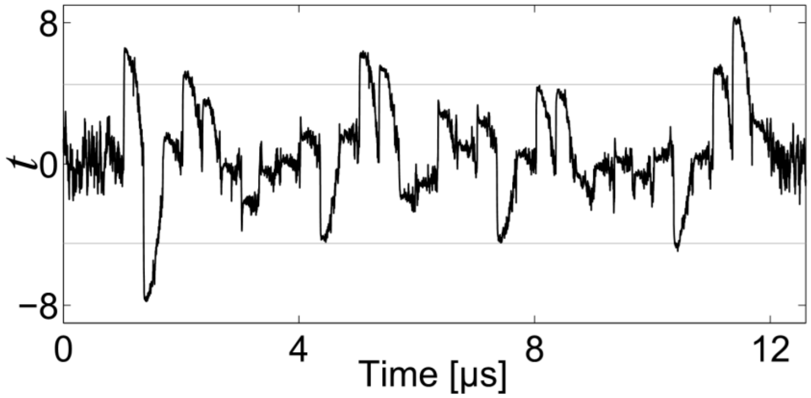
First Order



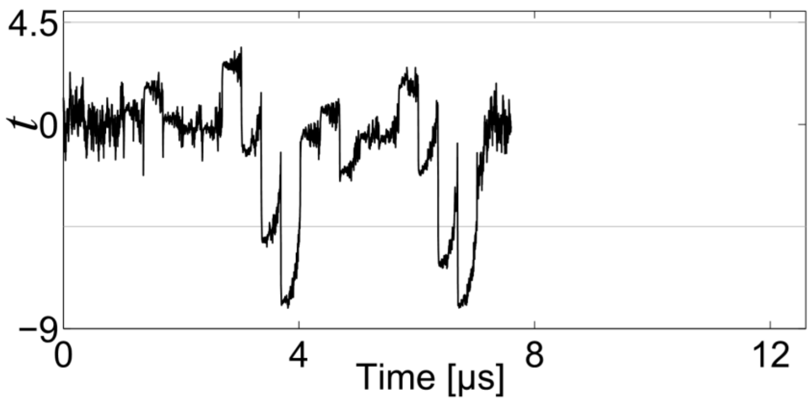
Second Order

No detectable leakage in first two orders (univariate)

Correct Measurement CS: FPGA



Fifth Order



Second Order (bivariate)

Might be vulnerable to bivariate second order attack

Correct Measurement Recommendations

Fixed vs. random:

- DUT with *masking* countermeasure
- With masked communication

Semi-fixed vs. random:

- DUT with *hiding* countermeasure
- Without masked communication

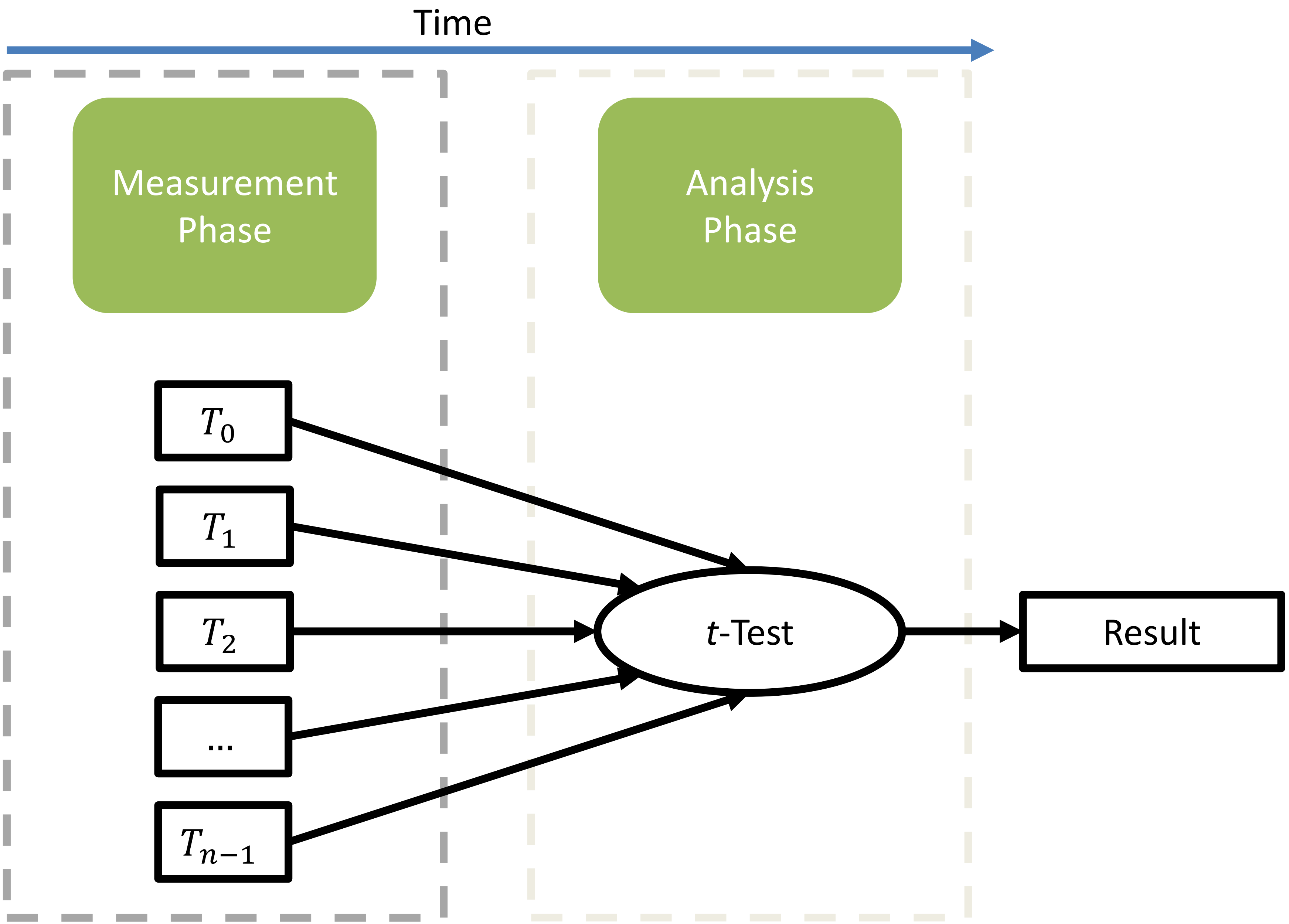
Specific t-test:

- DUT with *no* countermeasures
- Failed in former non-specific tests
- Identify suitable intermediate values for key recovery

Efficient Computation

- Classical Approach
- Incremental
- Multivariate
- Parallelization

Efficient Computation Classical Approach



Efficient Computation Classical Approach

t-Test $\rightarrow$ $\mathrm{t} = \dfrac{\mu_0 - \mu_1}{\sqrt{\dfrac{s_0^2}{n_0} + \dfrac{s_1^2}{n_1}}}$

Reminder:

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

Requires estimation of: $\boxed{(\mu_0, s_0^2)}$ $\boxed{(\mu_1, s_1^2)}$

	Pass 1	Pass 2	Pass 3
T_0, T_1, …, T_{n-1}	$\mu = E(T)$	$s^2 = E\left((T-\mu)^2\right)$	*Required for certain higher-order tests*

Efficient Computation Classical Approach

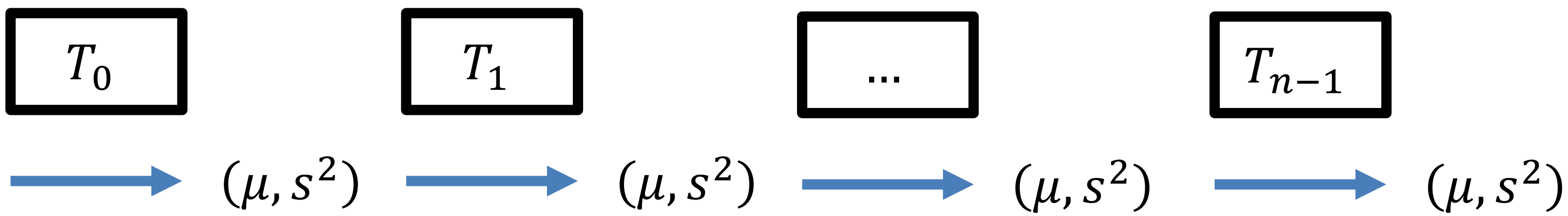
Problems:

1) Measurement phase need to be completed
2) All measurements need to be stored
3) Traces need to be loaded multiple times

Solution: ***Incremental Computation***

Efficient Computation Incremental

Idea: Update intermediate values for each new trace



Higher-order tests require the computation of additional values

Advantages:

1) Can be run in parallel to measurement phase
2) Does not require that all measurements are stored
3) Loads each trace only once

Efficient Computation Incremental

Problem: Computation of intermediate values

Approach 1: Use raw moments

$$\boxed{\text{d}^{\text{th}}\text{-order raw moment:} \quad M_d = E\left(T^d\right)}$$

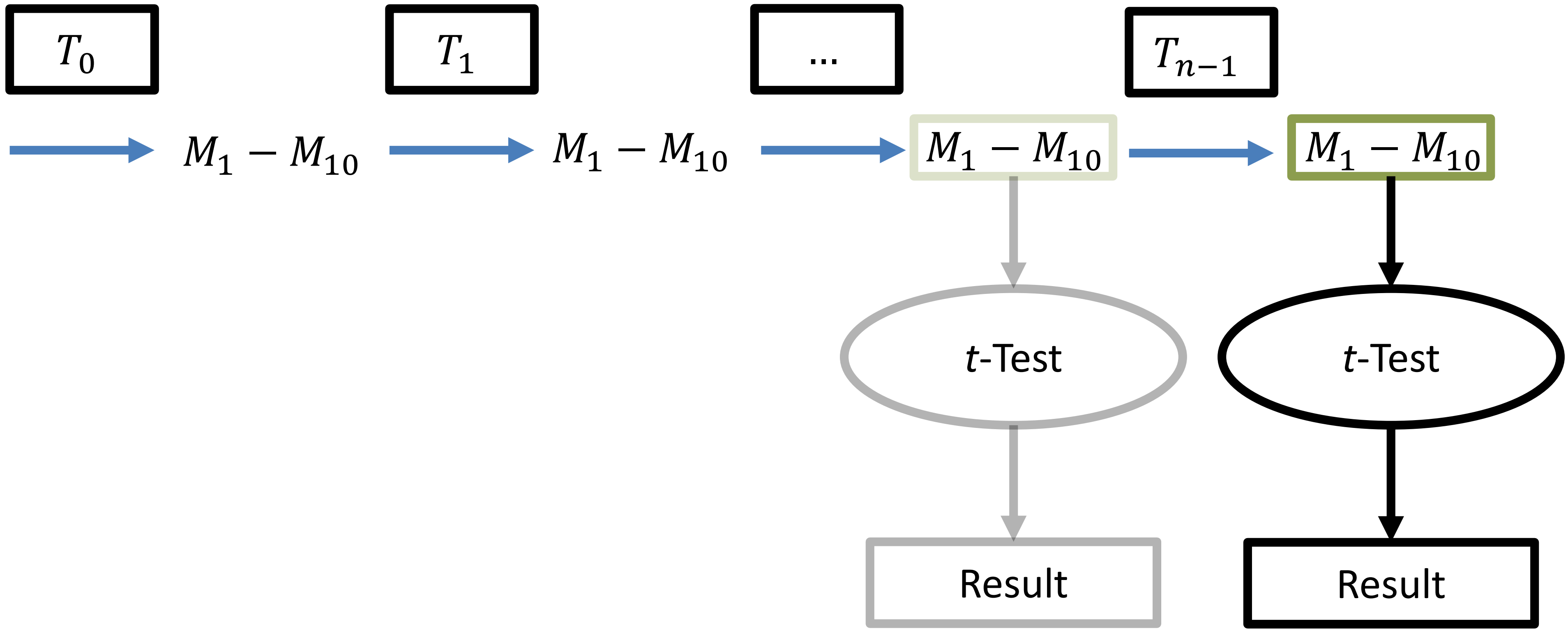
Given: $\boxed{M_1}$ $\boxed{M_2}$

Compute: $\mu = M_1$ $\quad s^2 = M_2 - (M_1)^2$

Higher-order test require additional moments

Example: Univariate 1st-5th order tests require $M_1 - M_{10}$

Efficient Computation Incremental



Efficient Computation Incremental

T_0 | T_1 | … | T_{n-1}

$\rightarrow M_1 - M_{10} \rightarrow M_1 - M_{10} \rightarrow M_1 - M_{10} \rightarrow \boxed{M_1 - M_{10}}$

$\downarrow$ t-Test $\downarrow$ Result

Easy to find update formulas for:

$$M_d = \sum_{i=0}^{n-1} \frac{(T_i)^d}{n}$$

Problem: Numerical unstable for large number of traces

Example: Computation of variance based on simulations (100M traces) with $\mathcal{N}(100,25)$

Method	Order 1	Order 2	Order 3	Order 4	Order 5
3-Pass	25.08399	1258.18874	15.00039	96.08342	947.25523
Raw	25.08399	1258.14132	14.49282	-1160.83799	-1939218.83401

Efficient Computation Incremental

Approach 2: Use *central* moments (and M_1)

$$\boxed{\text{d}^{\text{th}}\text{-order central moment:} \quad CM_d = E\left((T-\mu)^d\right)}$$

Given: $\boxed{M_1}$ $\boxed{CM_2}$

Compute: $\mu = M_1$ $s^2 = CM_2$

Higher-order test require additional central moments

$$\mu_d = \frac{CM_d}{\sqrt{CM_2}^d} \qquad (s_d)^2 = \frac{CM_{2d} - CM_d^2}{CM_2^d}$$

Efficient Computation Incremental

Not that easy to find update formulas for:

$$CM_d = \sum_{i=0}^{n-1} \frac{(T_i - \mu)^d}{n}$$

Idea: Use incremental formulas for central sums from [2]

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

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

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

[2] *Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments* Philippe Pébay, Sandia Report SAND2008-6212

Efficient Computation Incremental

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

Comparison to the raw moments approach:

- Slightly higher computational effort
- Less numerical problems, higher accuracy

Method	Order 1	Order 2	Order 3	Order 4	Order 5
3-Pass	25.08399	1258.18874	15.00039	96.08342	947.25523
Raw	25.08399	1258.14132	14.49282	-1160.83799	-1939218.83401
Our	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)

$$\boxed{T_i = A_i \cdot B_i}$$

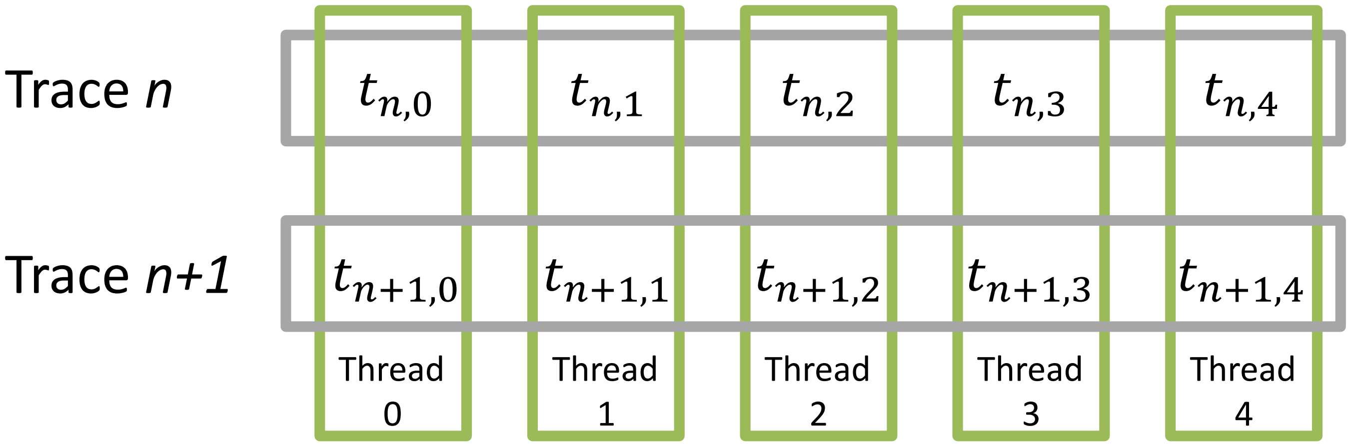
$$\boxed{T_i = A_i + B_i}$$

- Problematic for optimum combination function (centered product)

$$\boxed{T_i = (A_i - \mu_A) \cdot (B_i - \mu_B)}$$

- Incremental formulas need to be adjusted

Efficient Computation Parallelization



- Computations on separate points completely independent (univariate)

Time Comparison (8 Threads):

- 10M traces
- 22500 sample points
- 1^{st}-5^{th} order

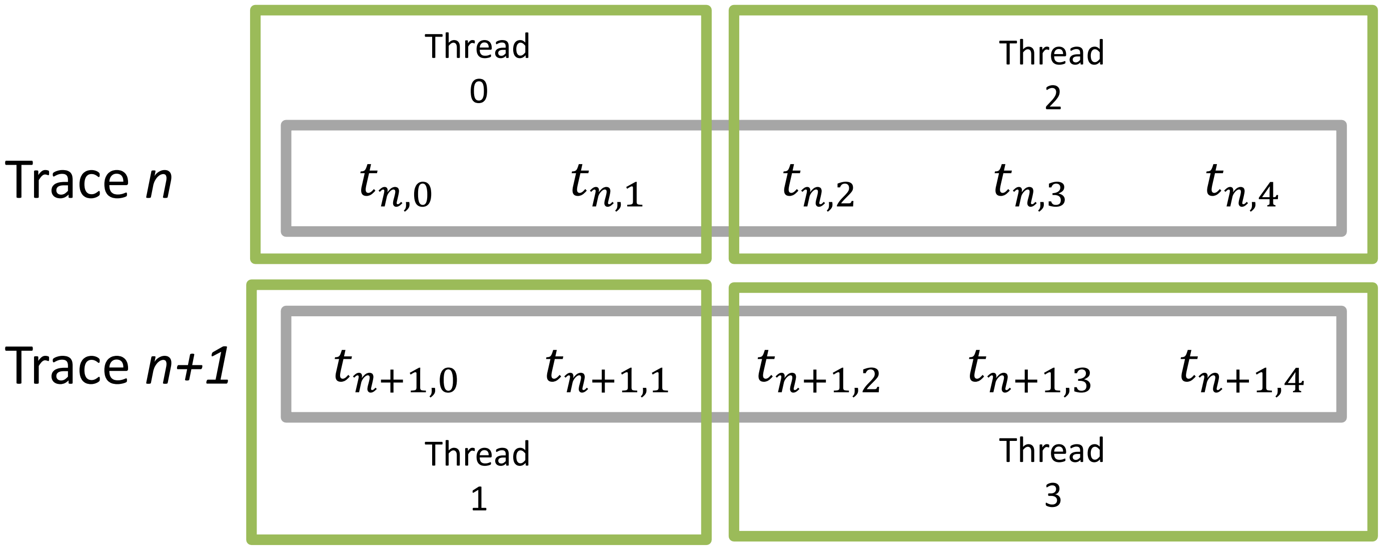
Method	Time	Memory
3-Pass	10.7 h	108.280 KB
Raw	5.6 h	108.452 KB
Our	5.9 h	108.592 KB

Efficient Computation Parallelization

Trace *n*	$t_{n,0}$	$t_{n,1}$	$t_{n,2}$	$t_{n,3}$	$t_{n,4}$	Thread 0
Trace *n+1*	$t_{n+1,0}$	$t_{n+1,1}$	$t_{n+1,2}$	$t_{n+1,3}$	$t_{n+1,4}$	Thread 1

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

Efficient Computation Parallelization



- Possible to combine both approaches for maximum performance

Example:

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

Conclusion

Conclusion

- *t*-test is simple and fast
- Some aspects need to be considered for correct testing
  - Measurement Phase
  - Analysis Phase
- *t*-test for security evaluation has become popular

Thanks for Listening!

Any Questions?