# A Novel Framework for Explainable Leakage Assessment

Si Gao and **Elisabeth Oswald**

D!ARC, University of Klagenfurt,

School of Computer Science, University of Birmingham

## Roadmap

## Hardware enables security

(Secure) Integrated Circuits: processors with added crypto functionality, certification required before entering the market

Typically found in bank cards (credit + debit), access cards, travel cards, ID cards, but also as SIMs in mobile phones, embedded in laptops, USB sticks, and all sorts of IoT devices.

Must undergo “certification”.

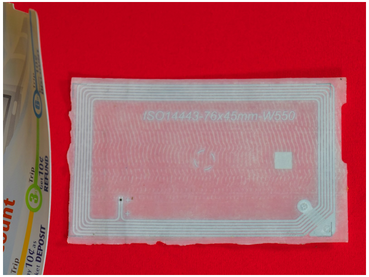


Figure: Contact-less smart card, Z22, CC BY-SA 4.0 `https://creativecommons.org/licenses/by-sa/4.0`, via Wikimedia Commons

## Roadmap

## Certification/Evaluation

Secure ICs must undergo certification before they can enter the market. Side channel attacks are important/expensive part of such an evaluation.

A **certifier** is a (national) authority that can issue certificates (under a range of schemes). An **evaluation lab** performs tests according to the selected scheme, and if demanded by the certifier, amend their procedures.

Attack based evaluations (Common Criteria/JHAS governed) vs “non-specific detection” (FIPS 140-3).

## Roadmap

## Non-specific Leakage Detection (aka Leakage Assessment)

The core idea that underpins the “omnipresent” Test Vector Leakage Assessment (TVLA) framework (non-specific detection) is that we cannot distinguish between two trace sets (acquired by choosing two different inputs):

$$\Pr[\mathbf{L}_i | X = x_i] \overset{?}{=} \Pr[\mathbf{L}_j | X = x_j].$$

“Distinguish trace sets”: via statistical testing; Gaussian assumption for trace points, univariate distribution tests, e.g. distance-of-means (with unequal and unknown variances); $X$ is data input (i.e. key is fixed, varying plaintext bytes only, or fix $x_i$ and vary $x_j$; TVLA sets $\alpha$ (rate of false positives), does not control rate of false negatives; plenty of issues, see [4], now being corrected.

Problem: it remains unclear if and how a potential leak can be turned into an attack.
We address this problem with our work.

## Roadmap

## Three Key Observations (Ideas)

- We suggest checking properties of the leakage distribution as a function of a set varying key chunks $K = \{K_i\}$
- We suggest to utilise a statistical model building approach instead of a moment-based statistic to test for evidence of leakage
- We suggest to define the notion of “exploitable leakage” via the $\underline{\text{size of the necessary key guess}}$ to assess security against differential attacks.

“Explainable”: We show how to turn the identified leakage into a certificational attack.

These ideas have been developed based on our previous work in [3], [2], and [1].

## Statistical Model Building as Detection

We use nested regression models as a way to express key dependency:

$$\tilde{L}_f(K) = \sum_j \beta_j u_j(K),\ j \in J$$

$$\tilde{L}_r(K) = \sum_j \beta_j u_j(K),\ j \in J', \text{with}\, J' \subset J$$

The full model is configured include more key dependent terms than the reduced model.

The coefficients $\beta$ for the two models are estimated from the available side channel observations. A statistical test decides if the two models differ significantly.

If there is enough evidence to distinguish the models, then the “extra” coefficients (=key chunks) matter.

## Statistical Model Building as Detection - Toy Example

Suppose that we have only two key chunks $K_1$ and $K_2$.
Then the full model is given as:

$$L_f(K) = \beta_0 + \beta_1 K_1 + \beta_2 K_2 + \beta_3 K_1 \cdot K_2.$$

The naive model would be

$$L_0 = \beta_0.$$

We say that there is key leakage on some observed data, if $L_0$ does not explain the observed data as well as $L_f$.

- Analysis will reveal points are that only key dependent as well (aka key schedule leakage)
- Test is instrumented nearly identically to TVLA but we now need key control in the device under test. Like with TVLA, test requires pre-processed trace to capture sequentially leaking shares, or any multivariate leakage.

## Framework for Exploitable/Explainable Leakage

Together this gives a novel three-step framework for explainable leakage assessments:

1. Leakage detection based on testing the full vs. the naive model (fully compatible with the flow of TVLA).
2. Identifying exploitable leakage by building/testing reduced models with degree lower than a given bound.
3. Explainable assessment: for exploitable leakage, build the smallest reduced model that is statistically equivalent to the full model, and then turns this into a "certificational" template attack.

"Certificational" attack vector: this is an attack vector that succeeds, but it is not necessarily the most trace efficient attack vector.

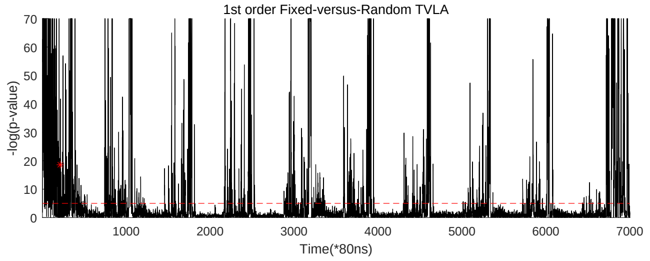
## Roadmap

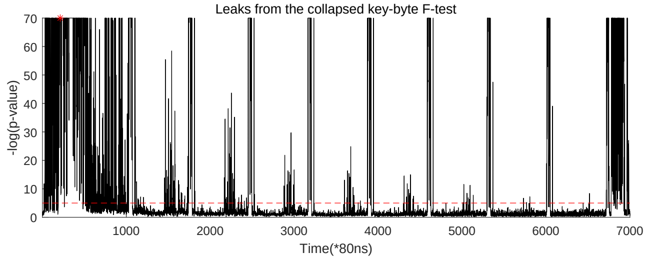
## Example: Boolean Masking on a 32-bit Processor

TVLA (top right).

We configure the full model to contain all 16 bytes of an AES key and run our non-specific detection (bottom right).

Model based detection discovers more key dependent points in first round than TVLA (as expected).



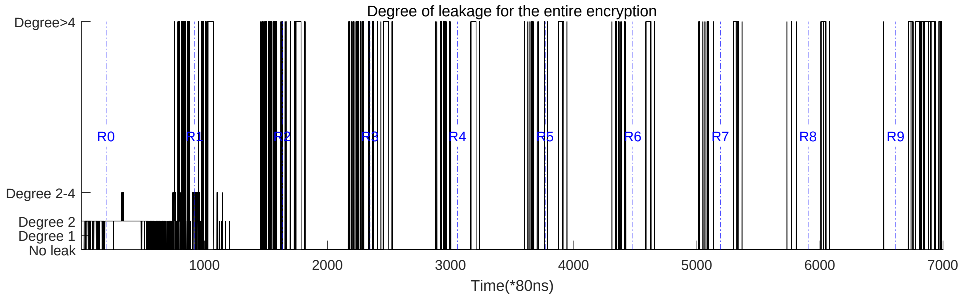


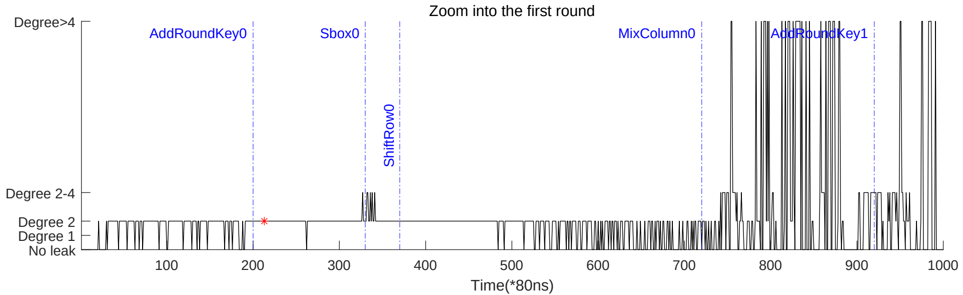
## Example: Boolean Masking on 32-Bit Processor

But which of the identified points offer **exploitable key leakage**?

We next configure our reduced model to contain 1, 2, 3, or 4 key bytes and compare with the full model which is based on 16 key bytes.

# Example: Boolean Masking on 32-Bit Processor





## Example: Boolean Masking on 32-Bit Processor

We pick one of the low degree points (marked with a red asterisk in the previous figure).

It only depends on at most 2 key bytes: but which ones?

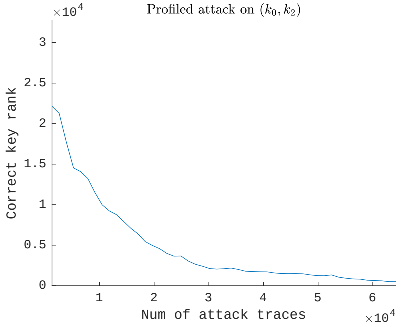
We find out by configuring the respective models, and check their test results (see Table on the right).

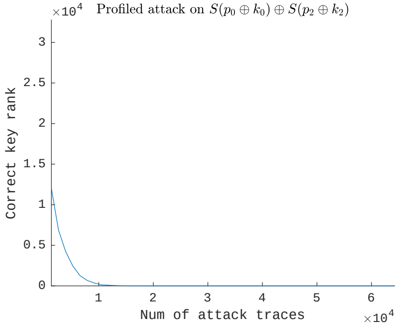
| Key bytes | $-log_{10}(p-value)$ |
|---|---|
| $(k_0, k_1)$ | 14.93 |
| $(k_0, k_2)$ | max |
| $(k_1, k_2)$ | max |
| $(k_0, k_3)$ | 140.26 |
| $(k_1, k_3)$ | 93.12 |
| $(k_2, k_3)$ | max |

## Example: Boolean Masking on 32-Bit Processor

A certificational attack vector can be established easily: build templates for the identified key pairs, and use them in a profiled attack.

The left hand side shows a certificational attack for the key pair $(k_0, k_2)$. The right hand side shows an attack vector for the same two key bytes but with knowledge of the implementation.





## Roadmap

## Conclusions and Open Questions

It is possible to use model building techniques for detection, and get some form of “explainability” in this way.

Nested regression models seemed a logic choice, but are there any other model building techniques that enable meaningful comparisons between statistical models?

Deep learning is the side channel community’s “favourite” model building approach at the moment: I can’t quite see if/how the qualities of our approach could be implemented with deep learning. Are there results for comparing “nested” deep net models ? Presumably we are looking for non-parametric methods to compare models ...

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 725042).

## References I

[1] S. Gao and E. Oswald.
A novel completeness test for leakage models and its application to side channel attacks and responsibly engineered simulators.
In EUROCRYPT, 2022.

[2] S. Gao, E. Oswald, and D. Page.
Towards micro-architectural leakage simulators: Reverse engineering micro-architectural leakage features is practical.
In EUROCRYPT, 2022.

[3] D. McCann, E. Oswald, and C. Whitnall.
Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages.
In USENIX Security 2017.

## References II

[4] C. Whitnall and E. Oswald.
A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules').
In ASIACRYPT 2019.