REASSURE

Project ID: 731591, Horizon 2020
http://www.reassure.eu

REASSURE

Deliverable D2 . 7

Final report on automation

Editor:	V. Verneuil (NXP)
Deliverable nature:	R
Dissemination level: (Confidentiality)	PU
Delivery date:	March 31, 2020
Version:	1.0
Total number of pages:	33
Keywords:	evaluation, automation, side-channels, coverage, hypothesis testing, machine learning, neural networks

Horizon 2020
European Union funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 731591.

Executive summary

This document is an update to deliverable 2.5 detailing further progress of the REASSURE consortium towards the goal of flexible evaluation methods that are usable by non-domain experts.

We begin by clarifying what we mean by automation: the facility to perform a task with a minimum of user input and interaction (expert or otherwise). This implies that the necessary parameters for a sound, fair and reproducible analysis are derived or estimated from the available data as far as possible. It also requires that the assumptions and limitations of the analysis be made transparent in the output, in order to guide (potentially non-expert) users towards sound conclusions.

Deliverable 1.1 identifies three main components of a typical evaluation process: measurement and (raw) trace processing, detection/mapping, and exploitation. These seldom proceed in a linear workflow, as information gained in later steps is used to repeat and refine earlier steps. For example, it can be hard to confirm the suitability of acquired traces without actually performing some sort of detection or attack, so that the measurement stage may need to be revisited in the light of observed outcomes. Such backtracking creates challenges for automation, in particular implying the need for improved quality metrics at every stage of evaluation – a matter of ongoing investigation.

On the basis that detection and mapping tasks are the most promising for automation, we then provide a more detailed account of statistical hypothesis testing and of how to perform the statistical power analyses and multiple comparison corrections required to carry out such procedures fairly and transparently. In the case of leakage detection strategies based on simple assumptions, such as those relying on t-tests, it is possible to arrive at analytical formulae for the latter; for complex methods, such as those estimating information theoretic quantities, the only options currently available (as far as we are aware) involve referring to empirically-derived indicators from (hopefully) representative scenarios.

We discuss the particular challenges of extending known results for univariate, 'first order' leakages to the sorts of higher-order univariate and multivariate leakages arising from protected schemes. Tricks to 'shift' information into distribution means via pre-processing inevitably violate the assumptions which make the statistical power analysis of t-tests straightforward. We consider that more work needs to be done from a basic methodological perspective before higher-order detection can realistically be considered a candidate for automation.

Inspired by the notion of code coverage in software testing, we suggest the need for measures of leakage evaluation coverage. We offer some suggestions of what these might look like: input-based scores indicating the proportion of the total population sampled for the experiment (under different assumptions about the form of the leakage); intermediate value-based scores indicating the fraction of total sensitive intermediates targeted by the tests; or even strategies whereby profiling information is used to design and test specific corner cases.

With a non-domain expert end user in mind we then consider how to apply recommendations from this deliverable to the tools produced in work package 3. In particular, we explain how to set the default test parameters for the automated simulation and analysis of traces so as to require minimal input from the developer in an iterative design process.

We lastly consider the automation potential of deep learning as a tool for side-channel analysis. WP1 (see, in particular, D1.2) has addressed deep learning from an efficiency perspective, but its ability to extract information from large amounts of raw data with minimal input on the part of the user makes it highly relevant to this work package also. We describe our efforts towards making best use of its capabilities, including via a proposal whereby collections of traces are jointly classified as having been generated under a particular key guess or not, and the inspection of the loss function gradients with respect to the input data during the training phase. We also discuss some of the obstacles to automation, such as the opaque nature of deep learning models, which may make it difficult to draw practically applicable conclusions about the form and location of leakage after an evaluation.

List of authors

Company	Author
UNI-KLU	E. Oswald
UNIVBRIS	C. Whitnall
NXP	V. Verneuil
UCL	S. Duval
UCL	W. Wang

Other contributions gratefully received from

Company	Author
UCL	R. Poussier
SGDSN	E. Prouff
SGDSN	E. Jaulmes

Contents

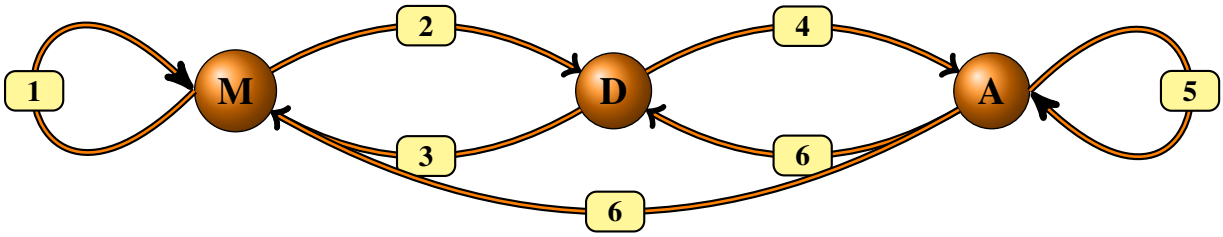

Figure 1: The three components comprising a typical evaluation, and the interactions between them.

1 Introduction

The overarching goal of work package (WP) 2 is to create flexible evaluation methods that are usable by non-domain experts. This can be achieved by addressing a number of objectives. The first objective is to research shortcut formulae that enable early statements about the security of an implementation against some side-channel attacks. The second objective is to develop some generic leakage assessment methods and to assess the feasibility of automating some evaluation steps/techniques such that they could be offered as a remote (e.g. cloud based) service. The third objective is to create a robust statistical methodology to characterise existing devices with the aim of building accurate leakage simulators. The fourth objective is to research the types of statements that can meaningfully be made following the automated analysis of simulated leakages (i.e. the combination of the second and third goal).

This deliverable is concerned with the second of these objectives: investigating which steps (of a typical evaluation process) lend themselves for automation. Task 1.1 is primarily concerned with identifying those common tasks in evaluations, which D1.1 lists as follows:

M: Measurement and (raw) trace processing. This step consists of the physical set-up (including measurement boards, any necessary alterations to the device under attack, selection of probes, amplifiers, etc.) as well as the processing of the (raw) traces as acquired by the digital oscilloscope (e.g. software filters, feature selection methods)

D: Detection and mapping of leakages. This step consists of determining whether or not information pertaining to sensitive data-dependent variables are present in the traces. Depending on the evaluation context, this step also consists of mapping the identified leaks to specific intermediate values, with the aim of providing specific feedback to the designer.

A: Attacks and exploitation of leakages. This step consists of conducting concrete attacks. The current state of the art includes profiled and unprofiled attacks (profiled attacks entail a preliminary stage in which leakage models for specific intermediate values are estimated from a training data set).

These steps are part of a complex process which, as Fig. 1 illustrates, is seldom linear. For instance it is not uncommon that, following an unsuccessful (or even a successful) attack (**A**) an evaluator elects (or is required by the certifier) to revisit the measurement setup of raw trace processing (**M**). This then leads to a costly iteration over all subsequent steps in the evaluation process. Similar 'backtracking' is indicated by the other arrows that lead from the 'end' of the evaluation (i.e. a successful attack) back towards earlier steps. The fact that such backtracking is required also hints towards a possibly important problem that we face in our endeavour to automate steps in **M** and **D**: the quality of the results of these steps is often only judged by the results of attacks at the end. This hints towards an absence of useful quality metrics for these earlier steps – a key lack that would need to be addressed in order to automate them.

2 Considering automation

The automation of a task implies that it is able to be performed with a minimum of user input and interaction. Ideally, then, the necessary parameters for a sound, fair and reproducible analysis should be derived or estimated from the available data as far as possible. Since evaluation tasks are predominantly of a statistical nature, this especially means that a procedure should be in-built with the facility to properly control the error rates of detection tests via (e.g.) automated computations to derive the required number of traces, and adjustments for multiple tests. In addition, some tasks require judgements to be made about the quality of an interim result before that result can be safely used as input to follow-on tasks. This calls for trustworthy quality measures appropriate to each step. Finally, *some* user input will always be unavoidable; we need to be precise about the level and nature of available expertise assumed by the automated parts of the procedures.

An important aspect of 'safe' automation is that the assumptions and scope of the analysis be made transparent in the output, so that conclusions drawn (potentially by non-experts) remain within the implicit limitations of the methods adopted. For evaluation tasks, this means reporting (e.g.) the power of the tests in representative circumstances (see Sec. 3.1), and perhaps some measure analogous to 'coverage' in software testing (see Sec. 3.4).

In the remainder of this section we will overview the challenges and opportunities for automation in all three evaluation stages as identified above. In Sec. 3 we will examine in more detail the particular requirements in the detection and mapping stage, as this is the one with most scope for automation.

2.1 Measurements and (raw) trace processing

2.1.1 Measurements

Trace acquisition, even with the help of designated tools such as Riscure's Inspector [65], typically involves manual set-up of the equipment as well as considerable input and feedback from the user in order to fine-tune it. In evaluations where the product developers have provided detailed specifications of the device under testing (DUT; we call these 'white-box' evaluations, by analogy to white- versus black-box software testing[1]), some of the information needed to take good quality measurements (such as the clock rate and other important frequencies) are known already, which reduces the involvement of the analyst. Usually, at least some of the characteristics of the implementation are unknown *a priori* and must be learned from preliminary measurements. For example, spectrum analysis can be used to find the important frequencies; pattern recognition can be used to locate the region of the trace associated with cryptographic operations, and guess at the cipher and possible countermeasures (if unknown); the level and range of the side channel must be observed in order to choose the resolution and offset which captures it at the appropriate granularity and without clipping; if taking EM measurements, the surface of the chip needs to be searched in order to find the physical location of the relevant activity. Only once these details have been learned can an evaluator choose the best parameters and equipment (such as physical filters and amplifiers) to take measurements of a sufficient quality and quantity for analysis.

Measurement is thus a process which entails considerable expertise as well as experimental trial-and-error. It is possible to envisage elements of automation within the individual tasks:

- Peak recognition could be used to find interesting frequencies in spectra.
- Matching against pre-acquired (and labelled) pattern snippets could help identify interesting regions and suggest possible activity.
- An EM probe could be provided with candidate coordinates to search and programmed to target 'promising' regions for taking larger acquisitions, based on preliminary on-the-fly analysis. (For example, high levels of clock frequency activity might be a valid indicator that one is 'not too far' from exploitable leakage).
- A rolling average could be fed back to the set-up to help automatically adjust the offset towards zero; on-the-fly histograms could similarly be used to find a suitable resolution.

[1] Not to be confused with white-box cryptography.

However, physical experiments are sufficiently unpredictable that it would be inadvisable to expect automated tasks to arrive at a sensible configuration without (expert) human supervision.

Moreover, the (known) criteria for identifying 'quality' in the measurement stage are inadequate to confirm whether or not the traces yield information about sensitive targets. This typically only becomes apparent once leakage detection (**D** in Fig. 1) or attacks (**A**) have been attempted. In addition, the sample *size* (that is, the number of traces) required to detect the information present depends on features which are not apparent in the measurement stage and must be learned from further (or previous) analysis (see Section 3.1.2). If the acquisition proves unsuitable to the task (in either quality or quantity), the type of iterative backtracking described in Section 1 becomes necessary.

If better metrics could be found to quantify trace quality independently of the (*a priori* unknown) leakage form and content, then these could provide improved feedback in the measurement stage and so pave the way for increased automation. One possible avenue, to our knowledge not yet explored directly in the context of side-channel acquisitions, are the techniques of blind signal-to-noise ratio (SNR) estimation, which seek to bypass the usual demands on the pre-characterisation of a signal before it can be measured [89, 55]. However, even if such an enhanced feedback mechanism could be developed, revisiting stage **M** in the light of lessons learned by further analysis in stages **D** and **A** would likely always produce further gains.

Of course, once effective set-up parameters have been ascertained for a given implementation (and detection/attack task), it does then become possible to automate repeated experiments via appropriate configuration scripts and test harnesses.

Simulations In some settings simulated traces can be used to test the leakage characteristics of a proposed implementation on a previously-profiled device, in order to pre-empt vulnerabilities during the development stage. Simulations can alternatively be performed simply to avoid the time and effort expended in taking measurements. Tools such as ELMO for the ARM Cortex-M0 [51] are able to emulate the data flow of an arbitrary code sequence and to map this to trace predictions via a power model. This is a naturally automatic process requiring as input only the code in a form readable to the tool, profiled model(s) for the form of the leakage (although this might be considered a built-in aspect of the tool), and the number of traces to be generated. Note that the 'quality' of simulated traces derives entirely from the accuracy of the simulator, and thus is fixed from acquisition to acquisition. A forthcoming WP2 deliverable (D2.6) will cover the topic of simulation in more detail, while WP3 will produce, among other things, a simulator tool (D3.2, D3.4).

2.1.2 Raw trace transformations for quality improvement

One motivation for processing raw traces is to improve the 'quality', by which is typically meant the SNR. Different definitions of SNR exist but almost all of them (barring the possible exceptions discussed above) entail defining some notion of 'signal' – ideally, a model of the data-dependent part of the leakage, by which the 'noise' (all non-exploitable variation in the measurements) can be derived via the subtraction of the fitted values from the raw measurements. Hence prior knowledge about the expected form of the leakage is required, implying the need for iteration via **D** or even **A**, which makes automation challenging.

Methods for improving signal quality include:

Trace alignment which uses distinctive reference patterns present in the trace measurements to remove the effects of random offsets in the time domain. Static alignment shifts each trace by a fixed horizontal amount; elastic alignment [83] adjusts the corrective offset according to localised patterns. Both benefit from user insight, although if appropriate feedback metrics could be devised (such as the pairwise correlation between aligned traces), a degree of automation may be possible.

Spectrum analysis flags the important frequencies. This can help with filtering, or can feed back into an improved measurement stage. [61]

Filtering entails selecting frequency bands of interest and/or cutting out frequencies deemed as irrelevant 'noise'. A detailed implementation specification, combined with output from a spectrum analysis, could help to automate this process, but care should be taken not to inadvertently remove signal frequencies. [22]

Fourier transformation converts leakages into the frequency domain. In some cases this can lead to effective attacks/detection where countermeasures have successfully prevented leakages in the time domain [67, 81].

Averaging multiple traces associated with the same inputs (including any randomness) is a simple way to reduce noise. This can be automated as part of the acquisition procedure, reducing the final storage complexity. However, it is likely to be more effective if performed after some form of alignment.

The effectiveness of signal processing can depend greatly on the order in which steps are taken, and over-processing can lead to degradation or even loss of signal, which may not become evident until attempts have been made to characterise or exploit the signal.

2.1.3 Raw trace transformations for compression/feature selection

Another, potentially overlapping, motivation for processing raw traces is to reduce the size of the dataset prior to a detection or attack procedure. Since the aim of such a step is to retain (or combine) trace points containing relevant information whilst discarding those containing mostly noise it again requires either some pre-understood notion of 'signal' or iteration via **D** or even **A**.

Principal Components Analysis PCA transforms a large, possibly correlated set of variables (trace points) into a set of linearly uncorrelated variables in decreasing order of total variation. The idea is that the first M of these (for some carefully chosen M) are adequate for the purposes of future analysis. However, in the side-channel setting it has been shown that, depending on the magnitude of the variation of the non-exploitable processes, the data-dependencies may not exhibit in the high-variance components and may be just as difficult to locate in the transformed set as in the raw set [4]. This makes it a poor candidate for automation, as any *a priori* threshold for retaining components risks discarding the relevant information before analysis is even attempted.

PCA on the empirical means An alternative (and more popular) mean of applying PCA to trace measurements is to derive the projection matrix from the empirical means associated with a byte of the input, so that the raw traces are transformed in such a way as to maximise the *data-dependent* variation (rather than the total variation) for any intermediate which is an injective function of the input byte [2]. This approach *can* be relied upon to locate the relevant information in the first components, making it more suitable for inclusion in automated procedures, but is limited in scope (i.e. it would not be suitable as a preliminary to leakage detection) and needs to be repeated for each input byte (or intermediate) of interest. Note that, because the first component is a combination of leaking points constructed to maximise the total variation, it typically has a stronger SNR than any one of those points taken individually, so that PCA can be seen as a method for improving signal quality as well as reducing dimensionality.

Linear Discriminant Analysis LDA is like PCA in that it (linearly) transforms a large set of variables into a reduced set of optimally-combined features (see [17] for its use in SCA). However, where PCA optimises with respect to overall variation, LDA takes information about the classes (in our case, typically the values of an input byte) as input, and seeks to maximise the between-class and minimise the within-class variation. In this sense, it is like an enhanced version of the above adaptation of PCA – and is likewise limited in scope to leakage associated with the input or intermediate defining the classes. It also comes with the additional substantial constraint that it cannot be performed on datasets with more variables (i.e. trace points) than observations per class (i.e. measured traces per input/intermediate), so that, in practice, it requires *a priori* knowledge of interesting windows [44]. This would make it awkward to incorporate in automated procedures.

Interesting point selection A variety of methods exist for reducing trace sets only to points which contain 'relevant' information. These typically involve computing, for each trace point, some score indicating 'information amount' (SNR [46, Ch. 4], ANOVA (aka Normalised Inter-Class Variance) [7], Sum of Squared (T-)Differences [29], correlation with a standard or profiled power model [9, 25], mutual information [64]) and choosing (e.g.) the first M, or all those which exceed a certain threshold. Of course, there is a very large overlap between these methods and the task of leakage detection. It would not be

a good idea to perform such a step as a preliminary to detection, as the retained information is, by design, restricted in scope (to a particular input or intermediate and to leakage of the form assumed by the selection criteria). However, the principle behind it might be useful in facilitating the automation of the transition from detection to (targeted) attacks.

Note that all of the above methods optimise with particular targets and leakage forms in mind. Discarding the (filtered/aligned) raw traces in favour of combined or selected features would reduce storage complexity but would limit the scope of future analysis. Such measures are therefore more appropriately viewed as component steps in detection and attack procedures – in which contexts, they could be safely automated as long as the optimisation criteria matched the objectives of the procedure in question.

2.2 Detection and mapping of leakage points

We detect leaks (that is, the presence of sensitive data dependency in the trace measurements) using statistical hypothesis tests for independence. These can be based on (non-parametric) comparisons between generic distributional features or on (parametric) comparisons between moments and related quantities, and vary in complexity and scope depending on whether one is interested in univariate or multivariate settings. We stick to univariate for the time being, as it poses challenges enough, the solutions to which are necessary preliminaries for solving analogous challenges in the multivariate case. Section 3.1 elaborates on statistical hypothesis testing; for the purposes of automation, the following aspects (in brief) are of particular interest:

- In order to automate any statistical test some parameters need to be known or decided beforehand.

  Either: The analyst fixes the desired rate of false positives (α), the desired rate of false negatives (β), and the minimum 'effect size' (i.e. the magnitude of the hypothetical difference) that the test needs to be sensitive to, and determines the number of samples required for the test to fit those parameters.

  Or: The analyst fixes α, β and the available number of samples, and uses these to determine the smallest effect size which the test will be sensitive to under those parameters.

  Or: The analyst fixes α, the effect size of interest, and the number of samples, and uses these to determine the power $1 - \beta$ of the test to detect an effect of the specified size or larger.

- Depending on the test to be automated, thought must also be given to the sample design, which needs to be representative of the variation that the test is designed to be sensitive to. For example, in the case of a 'specific' test [31], the signal for a particular intermediate can be improved by fixing (for a given key) all other bits/bytes/words of the state at that point and working backwards to identify the associated inputs. Some test constructions, such as semi-fixed-versus-random test vector leakage assessment (TVLA) [20], require deriving inputs that produce the same intermediate value in some inner round.

When it comes to *mapping* detected leaks, automation is straightforward in the case that one is working with an instruction level simulator (such as in [51]), as the relationship between the indices in the trace and the instruction sequence is perfectly known. This holds true regardless of the type of test applied (i.e. specific versus non-specific). On the other hand, in the case that one is working with traces measured from a real device, mapping is only possible via specific tests, which inherently ensure that detected leakages are tied to known intermediate targets (or at least to highly correlated ones).

2.3 Attacks and exploiting leakages

Assuming that an attacker or evaluator has access to the source code (or VHDL description) of a software (or hardware) implementation, a number of options exist for automating the exploitation of the associated leakage traces.

Leakage detection-style strategy: One approach is to mimic that of leakage detection by systematically attacking all of the points in a trace using some standard attack methodology, such as difference-of-means [38], or correlation DPA [9] with a sensibly chosen power model (e.g. Hamming weight in the case of software implementations, or Hamming distance in the case of hardware). Such a strategy could be straightforwardly automated as an adaptation of an automated detection procedure, by simply replacing

tests with actual attacks. So doing has the advantage of identifying, not just trace points which are data-dependent in some more or less arbitrary sense, but trace points which are *demonstrably vulnerable* to successful exploitation. However, it might give a misleading impression of the true vulnerability of the implementation, as a real-life attacker might be 'smarter' and more resourceful in tailoring her approach, leading to enhanced outcomes. It is also not to be relied upon as a *substitute* for leakage detection, as it will only ever find points susceptible to the particular attack(s) to which the traces are subjected.

Automated profiled attacks: A higher-effort attacker could be approximated by using the output of a leakage detection procedure as input to an automated profiling phase. For instance, if data-dependent points can be successfully mapped to their corresponding intermediate states, the measurements at those points could be used to then estimate conditional models of the leakages associated with particular values of those intermediates. These could range from simple univariate conditional means, suitable for use in a correlation attack, to multivariate Gaussian templates [15] for use via naive Bayes classification. A potential obstacle here is that the attack traces may not be perfectly aligned with the profiling traces, rendering the indices of the leakages learned from the detection and profiling stages unsuitable for the attack. In this case, a degree of trace processing may need to be automated as part of the attack (parameters derived from the profiling phase could be useful here). Alternatively, in the case that a correlation attack phase suffices, the derived power models could be applied against the whole trace set as per the leakage detection-style strategy suggested above.

Pre-defined battery of attacks: The third option is to *a priori* identify a set of attacks believed to be important and to cover a good range of potential real-world threats (difference-of-means [38], correlation with a selection of power models [9], mutual information [28], etc). Output from automated leakage detection and mapping procedures could then be used to help locate the target intermediates of interest in order to reduce the computational complexity of the exploitation stage and thereby maximise the capacity for increasing the size of the attack battery. This could be viewed as a trade-off between the above two approaches, and describes more or less what currently happens in practice during CC/EMVCo-like evaluations: detection and mapping are performed first, followed by a selection of relevant attacks as identified by a recognised body such as JHAS.

For the purposes of providing guidance to evaluators, it would be useful to determine, for some suitable example cases, the delta between the most comprehensive strategy (namely the automated profiling) and the least (for example a single bit t-test applied systematically). This would give a good indication of the risks involved in relying on the simple approach.

2.3.1 Producing attack metrics

Independently of the particular strategy, a variety of experiments might be performed in order to estimate and report the metrics of interest. For example, once vulnerable points have been identified, the successful attacks could be repeated with increasing numbers of traces drawn randomly from the total sample, in order to estimate the average sample size at which the subkey is recovered, as well as the subkey guessing entropy and success rate as the sample size increases [76]. Additionally, with a bit more computational effort the global key rank as the sample size increases could also be estimated [6, 30, 60, 84, 85]. From the perspective of automation, these procedures can be easily carried out without additional inputs from the user. However, they should be in-built with the capability to forecast the computational and memory complexity of the analysis requested, and to make sensible default choices with regards to (e.g.) the granularity with which sample sizes are increased, and the number of repetitions to get stable estimates.

2.3.2 Higher-order multivariate attacks

In scenarios (such as masked implementations [14]) where sensitive data-dependencies have been successfully eliminated from individual leakage points but still persist in joint distributions of multiple points, how to proceed depends very much on the information available to the evaluator. In the case that the randomness is fully known, it is possible to obtain candidate tuples via univariate leakage detection and mapping. These can then be targeted with a battery of higher-order attacks (most of which entail a pre-processing step to transform the

multivariate leakage into a univariate quanitity [14, 52, 62]) as per the third option above. Optionally, a profiling step could be incorporated, as per the second option [40].

However, the (unfortunately more common) case that the randomness is unknown to the evaluator becomes a lot more challenging. The only options then available are strategies of the first type, which entail exhaustively attacking all possible tuples, thereby recovering the relevant trace indices simultaneously with the target sensitive data. (Alternatively, targeted attacks could be attempted after a multivariate detection and mapping procedure, but this task shares the same constraints). Even in low masking orders the number of tuple candidates to be attacked/tested can be impractically large, so that any automation procedure would require additional criteria or input to enable selective targeting. Proposals exist to aid selection without exhaustive search [26] but these rely on heuristics and have yet to be developed and understood to the point where they could be responsibly integrated into automated workflows. Only in the case that one is working with a trace simulator is it currently known (that we are aware) how to easily identify tuples of leakage indices corresponding to jointly sensitive-data-dependent masks and masked intermediates [50].

3 Focus on detection and mapping of leakages

We now focus in more detail on the particular procedures commonly used to detect and map leakages, and the requirements and challenges involved in seeking to automate these. Foundational to this discussion are the methods and theory of statistical hypothesis testing, which we begin by overviewing in fairly general terms. We then look at the specifics of leakage detection tests and the different forms these can take depending on the end goal. Considerations for automation vary across these different forms and goals.

3.1 Statistical hypothesis testing

A statistical hypothesis test uses sample data to decide whether to reject one hypothesis about the underlying population (the 'null') in favour of another (the 'alternative'). It typically involves the computation of a test statistic with a known or derivable (e.g. through randomised resampling) distribution under the null hypothesis. If the observed value is 'extreme' – i.e., the probability of such a value occurring under the null hypothesis is smaller than some fixed probability of false rejection α – then the null is rejected. Otherwise the null is 'not rejected'. Crucially, this is not the same as 'accepting' the null. For example, there may simply not be enough data to provide conclusive evidence against it. This is why it quickly becomes important to understand the concept of 'statistical power': the probability of rejecting the null in the case that the alternative hypothesis is correct, which, for a given α, depends on the magnitude of the effect as well as the sample size. (See below for more on power, α, effect size and sample size).

Hypothesis tests can, among other things, be used to make informed judgements about whether or not two populations are the same, as characterised by particular parameters (e.g. in the case of the t-test [32]) or by the empirical distribution function (e.g. two sample Kolmogorov–Smirnov [39, 72]). In particular, estimation is at best able to acquire the underlying parameters to a certain precision; estimates from two populations are not expected to coincide even when the parameters *are* the same, and the goal of hypothesis testing is to decide if they are far enough apart to be inconsistent with an underlying match.

3.1.1 Significance level and power of a test

Since the estimate of a test statistic is itself an observation of a random variable (with a sampling distribution of its own) conclusions drawn from statistical tests are subject to error. The decision to reject a null hypothesis when it is in fact true is called a Type I error (a 'false positive'). In the side-channel setting this corresponds to finding leakage when in fact there is none. A Type II error is a failure to reject the null when it is in fact false (a 'false negative'), corresponding to failing to find leakage which is in reality present. The two errors can be traded-off against one another, and mitigated (but not eliminated) by increasing the sample size and/or choosing a more powerful test.

The relative seriousness of each error type varies by statistical application. In a leakage detection scenario, for example, large numbers of Type I errors might be considered primarily a nuisance – as long as *some* of the trace points flagged up by the analysis are indeed leaky, the test can still be used for demonstrating vulnerability. Nonetheless it is clearly desirable to minimise the probability of such errors, and also to fix it from evaluation

to evaluation, so as to ensure that assessments of different devices and implementations are fair. This can be achieved by setting a suitable rate of false positives α (also called significance level).

By contrast, large numbers of false negatives pose a serious problem in the leakage detection setting, where they equate to an impression of false security. It is therefore especially desirable to minimise the probability β of a Type II error – conversely, to maximise the *power* $1-\beta$ of a test. To assess whether an evaluation is meaningful (especially in the case that it fails to find leakage) it is necessary to understand how 'powerful' the performed tests are. Moreover, it is good practice to *design* tests with power in mind from the start. This entails performing an *a priori* (statistical) power analysis to ascertain the sample size required to detect data-dependencies of the expected magnitude with the required probability of success [49]. If the required probability of success is not achievable by a given test within a feasible number of traces there is simply no point in performing the test, as failures to find leakage will have no meaningful interpretation.

The impact of multiple testing. The current application of univariate tests to the task of leakage testing entails performing them on each trace point separately and drawing inference on them simultaneously to arrive at a judgement about the 'entirety' of the leakage produced by a device. The problem with this approach is that a test configured to have a Type I error probability of $\alpha = 0.05$ in a single instance will, in expectation, produce at least one false positive in every twenty instances (assuming for simplicity that trace points are independent). Given that side-channel traces can comprise many thousands of sample points, the *overall* probability of a Type I error subsequently becomes extremely large. We present methods to mitigate for the inflation of Type I errors – and discuss the problems they introduce – in Section 3.2.2, after first explaining the formalities of a statistical test in the more straightforward single test setting.

3.1.2 Determining the sample size

The techniques of statistical power analysis aim at ascertaining the sample size required for a given test to detect an effect of a certain size with a certain probability [18]. As mentioned above, this is key to ensuring that the outcome of hypothesis tests (including those performed as part of leakage detection) can be meaningfully interpreted, even in the event that the test fails to reject the null hypothesis (e.g. fails to detect leakage).

Statistical power analysis is not straightforward in the general case (more on that later), but for well-understood scenarios such as comparisons between Gaussian means (see, e.g., TVLA [31]), it is possible to arrive at analytical formulae. We begin with a simple visual example that illustrates the concepts of α and β values and their relationship to the sample size.

Consider the following two-sided hypothesis test for the mean of a Gaussian-distributed variable Y:

$$H_0 : \mu_Y = \mu_0 \text{ vs. } H_{alt} : \mu_Y \neq \mu_0. \tag{1}$$

Note that, in the leakage detection setting, where one typically wishes to test for a non-zero difference in means between *two* Gaussian distributions Tr_A and Tr_B, this can be achieved by defining $Y = \mathrm{Tr}_A - \mathrm{Tr}_B$ and (via the properties of the Gaussian distribution) performing the above test with $\mu_0 = 0$. The variance of Y defined in this way is the sum of the variances of each of the two distributions being compared: $\sigma_Y^2 = \sigma_A^2 + \sigma_B^2$.

Suppose the alternative hypothesis is true and that $\mu_Y = \mu_{alt}$. This is called a 'specific alternative'[2], in recognition of the fact that it is not usually possible to compute power for *all* the alternatives when H_{alt} defines a set or range. In the leakage detection setting one typically chooses $\mu_{alt} > 0$ to be the smallest difference $|\mu_A - \mu_B|$ that is considered of practical relevance; this is called the effect size. Without loss of generality, we suppose that $\mu_{alt} > \mu_0$.

Figure 2 illustrates the test procedure when the risk of a Type I error is set to α and the sample size is presumed large enough (typically $n > 30$) that the distributions of the test statistic under the null and alternative hypotheses can be approximated by Gaussian distributions. The red areas together sum to α; the blue area indicates the overlap of H_0 and H_{alt} and corresponds to β (the risk of a Type II error). The power of the test – that is, the probability of correctly rejecting the null hypothesis when the alternative in true – is then $1 - \beta$, as depicted by the shaded area.

There are essentially three ways to increase the power of the test. One is to increase the effect size of interest which, as should be clear from Figure 2, serves to push the distributions apart, thereby diminishing the overlap

[2]The overloading of terminology between 'specific alternatives' and 'specific' TVLA tests is unfortunate but unavoidable.

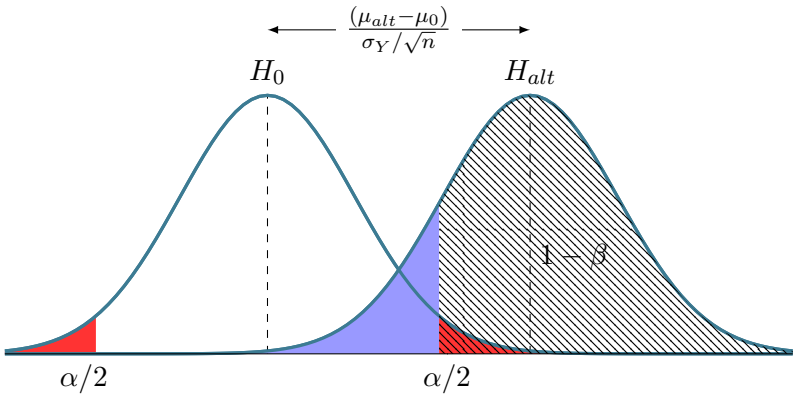

Figure 2: Figure showing the Type I and II error probabilities, α and β as well as the effect size $\mu_{alt} - \mu_0$ for a specific alternative such that $\mu_{alt} > \mu_0$.

between them. Another is to increase α – that is, to make a trade-off between Type II and Type I errors – or (if appropriate) to perform a one-sided test, either of which has the effect (in this case) of shifting the critical value to the left so that the shaded region becomes larger. (In the leakage detection case the one-sided test is unlikely to be suitable as differences in either direction are equally important and neither can be ruled out *a priori*). The third way to increase the power is to increase the sample size for the experiment. This reduces the standard error on the sample means, which again pushes the alternative distribution of the test statistic further away from null (note from Figure 2 that it features in the denominator of the distance).

Suppose you have an effect size in mind – based either on observations made during similar previous experiments, or on a subjective value judgement about how large an effect needs to be before it is practically relevant (e.g. the level of leakage which is deemed intolerable) – and you want your test to have a given confidence level α and power $1 - \beta$. The relationship between confidence, power, effect size and sample size can then be used to derive the minimum sample size necessary to achieve this.

The details of the argumentation that now follows are specific to a two-tailed t-test, but the general procedure can be adapted to any test for which the distribution of the test statistic is known under the null and alternative hypotheses.

For the sake of simplicity (i.e. to avoid calculating effectively irrelevant degrees of freedom) we will assume that our test will in any case require the acquisition of more than 30 observations, so that the Gaussian approximations for the test statistics hold as in Figure 2. Without loss of generality we also assume that the difference of means is positive (otherwise the sets can be easily swapped). Finally, we assume that we seek to populate both sets with equal numbers $|\mathrm{Tr}| = n/2$ of observed traces.

Theorem 1. *Let* Tr_A *be a set of traces of size* $n/2$ *drawn via repeat sampling from a normal distribution* $\mathcal{N}(\mu_A, \sigma_A^2)$ *and* Tr_B *be a set of traces of size* $n/2$ *drawn via repeat sampling from a normal distribution* $\mathcal{N}(\mu_B, \sigma_B^2)$*. Then, in a two-tailed test for a difference between the sample means:*

$$H_0\text{: } \mu_A = \mu_B \text{ vs. } H_{alt}\text{: } \mu_A \neq \mu_B, \tag{2}$$

in order to achieve significance level α *and power* $1 - \beta$*, the overall number of traces* n *needs to be chosen such that:*

$$n \geq 2 \cdot \frac{(z_{\alpha/2} + z_\beta)^2 \cdot ({\sigma_A}^2 + {\sigma_B}^2)}{(\mu_A - \mu_B)^2}. \tag{3}$$

3.1.3 Deriving n in practice

Of course, except in the rare case that the true population parameters μ_A^2, μ_B^2, σ_A^2 and σ_B^2 are *known* (implying that empirical analysis is anyway something of a redundant exercise), expression (3) doesn't give a definitive solution for the sample size but rather an aid to informed planning based on conjectures about the distributions and/or evidence gathered from previous experiments.

In particular, the computations require provision of an effect size. A tempting 'solution' is to estimate it from the test data. This amounts to 'post hoc' power analysis [79] – a practice strongly advised against in the

statistical literature. It is essentially a form of circular reasoning – in fact, there is a direct correspondence between the p-value and the power to detect the observed effect, so that 'post hoc power analysis' merely re-expresses the information contained already in the test outcome [35]. Unfortunately, is those cases where power diagnostics are most needed (namely those where the test fails, because the estimated difference was too small) where it produces the most meaningless results.

A far better option would be to set the effect size according to *a priori* beliefs about what constitutes a *practically meaningful* difference. In a medical statistics setting, for example, a practically meaningful effect might be the expected impact required from a treatment before it is considered cost efficient. The idealised correlative in cryptography might be how much information can be leaked from a side-channel before the theoretical security level (e.g. in terms of bits) is reduced below an acceptable threshold, although this would require knowing how to interpret the test statistic in terms of bits of security. This may be possible in the context of mutual information tests, but is unlikely to be straightforward.

In the absence (for now) of such an idealised criterion, we instead resort to looking for effects of an 'expected' size, based on previous experiments and (crucially) fixed independently of the test acquisition. Because different devices and test set-ups produce measurements of different scales and variance/covariance magnitude, it is convenient to express effect sizes in standardised form. Cohen's d is defined as the mean difference divided by the pooled standard deviation of two univariate samples $\{A\}$ and $\{B\}$:

$$d = \frac{\overline{A} - \overline{B}}{\sqrt{\frac{(n_A-1)s_A^2+(n_B-1)s_B^2}{n_A+n_B-2}}}$$

where s_A^2 and s_B^2 are the sample variances.

Notice that this is essentially a measure of signal-to-noise ratios (SNR), closely related to (and therefore tracking) the various notions that already appear in the side-channel literature. The most common definition is the variance of the characterised data-dependent part of the leakage (for example, the Hamming weight, or a linear regression model) divided by the residual variance (i.e., the noise). This is not identical to d, which corresponds to a 'signal' that arises from the construction of the test, e.g. the difference between the partitioned means – but for a given leakage characterisation and partition the one can be determined from the other.

Cohen [18] proposed that effects of 0.2 or less should be considered 'small', effects around 0.5 are 'medium', and effects of 0.8 or more are 'large'. Sawilowsky [66] expanded the list to incorporate 'very small' effects of 0.01 or less, and 'very large' and 'huge' effects of over 1.2 or 2.0 respectively. The relative cheapness of sampling leakage traces (and subsequent large sample sizes) compared with studies in other fields (such as medicine, psychology and econometrics), as well as the high security stakes of side-channel analysis, make 'very small' effects of more interest than they typically are in other statistical applications.

This helps to put the analysis on a like-for-like footing for all implementations. But of course acquisitions vary greatly in the standardised effects they are prone to exhibit, so it doesn't remove the need for knowledge of the particulars of the device in order for meaningful interpretation. With this caveat in mind, we next provide some ball-park figures based on three different existing acquisitions.

Observed effect sizes It is not straightforward to 'simply' observe effect magnitudes. This is because all differences will be non-zero, regardless of whether they represent actual leakage, and deciding which ones are 'meaningful' essentially corresponds to the task of detection itself. Choosing 'real' effects based on the outcomes of t-tests, and then using the magnitudes of those effects to make claims about 'detectable' effect sizes, amounts to circular reasoning, and depends on the choice of significance criteria. Fortunately the end goal of leakage detection provides us with a natural, slightly more objective, criterion for identifying 'real' effects, via the outcomes of key recovery attacks. That is, if leakage detection is geared towards identifying (without having to perform attacks) points in the trace which are vulnerable to attack, then an effect size which is 'large enough' to be of interest is one that can be successfully exploited.

We take this approach, and perform distance-of-means attacks on all 128 bits of the first round SubBytes output for three AES acquisitions, taken on an ARM board, an 8051 microcontroller and an RFID device. We also compute the sample effects for each of those bits, which enables us to report estimated effect sizes of interest. Two remaining draw-backs are 1) especially for small effects (i.e. those with a low SNR), the sample sizes may not be large enough for the estimates to be precise; and 2) more points may become vulnerable

Implementation	Proportion interesting	Standardised Min	Standardised Max	Standardised Median	Raw Min	Raw Max	Raw Median
ARM	0.0226	0.0444	0.9087	0.1155	0.0388	1.0265	0.1073
8051	0.0150	0.0413	1.4265	0.1670	0.0254	5.3808	0.1469
RFID	0.0049	0.0624	0.3935	0.0933	0.2272	3.4075	0.3836

Table 1: Summary of effect magnitudes associated with stable distance-of-means key recovery attacks.

to attack given even larger samples (which is always the case). So smaller effect sizes may be important to evaluators concerned with more powerful adversaries.

Since there are only 256 subkey candidates, among thousands of distance-of-means attacks a proportion will inevitably rank the correct key first purely by chance (approximately $\frac{1}{256}$ of those on irrelevant points). Adapting from [78], we take measures to confirm the stability of an attack outcome before classifying a point as 'interesting'. Our approach involves repeating the attack on 99% of the full sample and retaining only those points where the correct subkey is ranked first in both instances. (Even then the length of the traces makes it likely that some of the retained points will be 'false positives', a problem that affects multiple statistical tests in general, as we discuss in the next section).

Figure 3 shows the raw (top) and standardised (bottom) observed effect sizes (i.e. mean differences associated with an S-box bit) of first round AES traces measured from an ARM board, an 8051 microcontroller and an RFID device respectively. As expected, because of the different scales of the measurements (arising from different pre-processing, etc), the raw effects are not necessarily useful to compare. The ARM effects range up to about 0.8, while effects on the 8051 and the RFID implementation range up to 3 and 2 respectively. The standardised effects are much more comparable (≈ 0.6 and ≈ 1 for ARM and 8051 respectively; ≈ 0.4 for the RFID, although this is for the second rather than the first S-box as the latter is less 'leaky' in this instance).

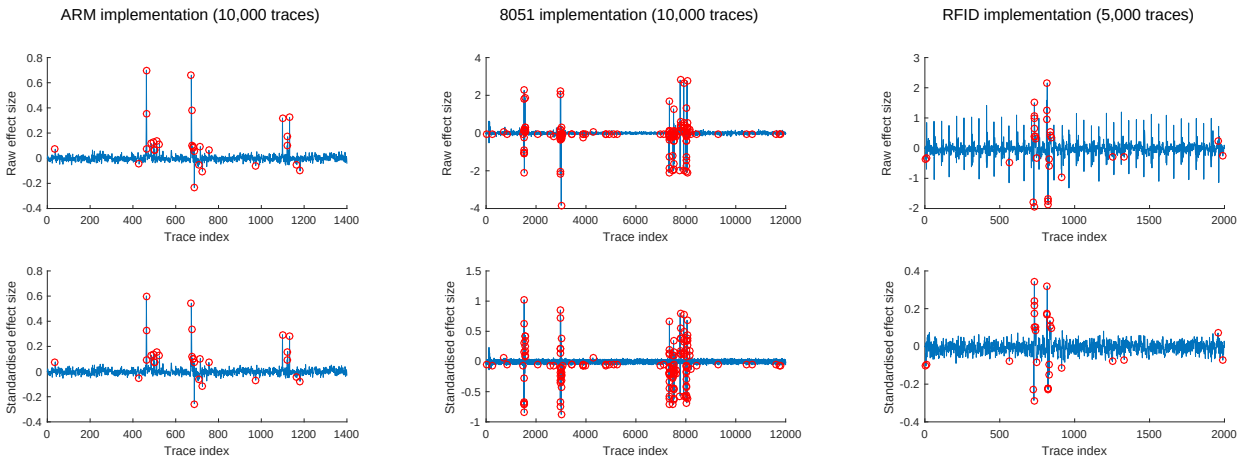

Figure 3: Difference of means (top) and standardised equivalent (bottom) associated with the first bit of the first S-box of two software AES implementations and the first bit of the second S-box of one hardware implementation (chosen because the first S-box is not very 'leaky' in that particular acquisition). Red circles denote points where a distance-of-means attack achieves stable key recovery.

Table 1 summarises the standardised and raw effect sizes associated with distance-of-means key recoveries over *all* bits of all S-boxes. The smallest standardised effect detected is 0.0413 for the 8051 microcontroller; the ARM and RFID smallest effects are in a similar ballpark. One might therefore choose 0.04 as a minimum effect of interest – although, as we have warned, larger samples will reveal ever-smaller effects, if present. This motivates approaching the question with a 'worst case' adversary in mind, rather than relying only on the data we have access to ourselves.

Population standard deviations If raw (rather than standardised) effect sizes are considered, statistical power analysis will also require knowledge of the population standard deviations of the partitioned samples, which may or may not be the same. These are usually assumed to have been obtained from previous exper-

iments and/or already-published results, which can be especially tricky when approaching a new target for evaluation. Various rules-of-thumb exist, for example, dividing the expected range of values by 5 [19, 70]. This reduces the reliance on *a priori* knowledge (at the cost of lower accuracy) but still requires expert input in the form of a meaningful approximation of the range. Whatever the information available, it is generally preferred to choose conservative estimates for the standard deviation (e.g. the largest among those previously observed) as underestimates will lead to overestimates of the power, risking a false sense of security.

Deriving n in the general case Since statistical power analysis requires characterising the test statistic distributions under both the null and a specific alternative hypothesis, it is non-straightforward in all but the simplest of cases. Sample size and power derivations for quantities such as mutual information typically rely on methods such as randomised resampling [49], which can provide useful (and hopefully transferable) insights, but do not permit the type of quick and cheap preliminary computations that incorporate neatly into automated procedures.

3.1.4 Implications for automation

In summary, and to reiterate Section 2.2, (semi-)automating a statistical test for the purposes of leakage detection might proceed as follows:

- The user provides, as input, an acceptable rate of false positives α, a desired power $1-\beta$, their best knowledge of the population variance σ^2, and an expected (or interesting) effect size $\mu_A - \mu_B$ (alternatively, a standardised effect size d).
- The procedure computes the per-test significance level $\alpha_{\{\text{per test}\}}$ corresponding to the overall significance criteria implied by α for the number of tests to be performed.
- The procedure computes (and outputs) the sample size required for the test to have the desired power $1 - \beta$.
- The procedure then awaits input of the acquired data before performing the analysis.

It is unfortunate that this sequence requires a break from the detection procedure **D** in order to return to the measurement phase **M**. An alternative which avoids this would be to fix the sample size according to the acquisition, and rearrange the computations in Section 3.1.2 to instead output either: the minimum effect size that the test would have the stated power to detect; or, the power with which the test would be able to detect the stated effect size. This would aid interpretation of the results in the case that the tests do not reject the null.

3.2 How to choose (and adjust) α and β

For a given sample size and effect size one essentially trades-off between Type I and Type II errors. Choosing α and β thus requires understanding the relative undesirability of these for a particular application. For example, in a medical statistics setting, false negatives might be of considerably more concern than false positives, particularly in a screening procedure designed to select at-risk patients for the purposes of further testing.

In the case of leakage detection, we argue that the various different methods can be placed into four categories that are distinctive in their intent and purpose, and that the appropriate error trade-off varies across these differing goals.

3.2.1 Four goals of leakage detection

Certifying vulnerability: Find a leak in **at least one** trace point. A test with *high confidence* is required for this purpose, to avoid flagging information leaks where there are none. Any method can, in principle, be used for this task provided the Type I error is adequately controlled for.

Certifying security: Find no leaks having tested thoroughly. Procedures for this either have to test all intermediate values exhaustively, or make no assumptions about intermediate values. (The only two candidates in this latter category, as far as we are aware, are called the continuous mutual information (CMI) test [16], and the non-specific t-test [31]; see Section 3.3.1 below for details). They need to be able to identify leaks which are present in first and higher-order moments, but ideally should not be restricted to testing

moments at all (as in the case of CMI). In this setting the important thing is to guarantee a *high power* (i.e. a low risk of Type II errors) which can be achieved by choosing an appropriate sample size for the expected effect size. Only then can a failure to reject the null of 'no leakage' be interpreted as any sort of reassurance about the true security of the device.

Demonstrating an attack: Map a leaking point (or tuple) to its associated intermediate state(s) and perform an attack. Typically it is of interest to report attack outcomes and/or projections derived from those outcomes, such as the number of traces required for key recovery, and/or the global key rank for a given sample size. 'Specific' tests such as round output/S-box TVLA with random inputs [31] or the normalised inter-class covariance (NICV; see Section 3.3.2) [7] are the most natural choices with this goal in mind as they reveal target values as part of their usual output. False positives are especially undesirable as they represent wasted effort in the attack phase; thus it is necessary to carefully control for the Type I error.

Highlighting vulnerabilities: Map all exploitable leakage points to their associated intermediate states in order to guide designers seeking to secure the device. This has something in common with certifying security, as both require an 'exhaustive' analysis, and something in common with demonstrating an attack, as both require being able to locate the source of the leakage – suggesting the need for 'specific' tests, but lots of them. False negatives are of greater concern than false positives as they represent vulnerabilities that will remain unfixed, so the risk of Type II errors needs to be kept low.

Clearly, highlighting vulnerabilities is the most ambitious of these goals: it demands a large number of statistical tests and also potentially requires large numbers of measurements due to the link between power and sample size.

3.2.2 Multiplicity corrections

There are two main ways of correcting for the inflation of Type I error rates when multiple tests are performed: controlling the *familywise error rate* (FWER) [34] and controlling the *false discovery rate* (FDR) [5]. Both of these were discussed in the context of leakage detection by Mather *et al.* [49]. A recent proposal [91] takes an alternative approach, which decides to collectively reject or not reject a set of null hypotheses based on the distribution of the p-values output by the tests performed separately. Their method appears to be more powerful than tests using the Å idÃ¡k correction [86] to control the FWER, but the authors did not provide a comparison with tests controlling for the FDR. Moreover, the test is unable to conclude *which* of the null hypotheses are untrue, just that at least one of them is, so does not identify the location of the leakage.

The original TVLA recommendations [31] also include an often-overlooked measure that implicitly guards against inflated false positive rates. The instruction is made to repeat the entire leakage detection on two independent samples (without adjusting the significance criteria) and only retain the points that are detected in both instances. Of course, this supposes perfect alignment between the two acquisitions. But practical challenges aside, this strategy of repetition can be very effective in reducing the risk of false positives. If the per-test significance level is $\alpha_{per-test}$, then the probability of observing at least one false positive might be as large as $\alpha_{overall} = 1-(1-\alpha_{per-test})^N$ (where N is the number of points in a trace) if the tests are independent. The probability of observing at least one false positive in each of a pair of independent experiments is then $(1 - (1 - \alpha_{per-test})^N)^2$. However, the probability of observing two false positives *in the same position* is $\alpha_{repeat} = 1 - (1 - \alpha^2_{per-test})^N$, which grows much slower as N increases.

Unfortunately, as should be evident from the preceding discussion, all of these methods to guard against inflated Type I error rates simultaneously increase the Type II error rates (for a given sample size) – that is, they degrade the *power* of the test. Most methods to control the FWER (for example, the Bonferroni [24] and the Å idÃ¡k procedures [86]) do so for a 'worst case' scenario of independence, so are considerably more conservative than needed for leakage detection, where there exist dependencies between the tests. FDR controlling procedures are known to be less conservative [49], but become harder to analyse. In particular, they do not correspond to simple α adjustments and there is less of a consensus about how to correctly derive power in such instances (compare, e.g., [8, 27, 42, 59, 82]). Progressing beyond the over-simplifying independence assumption (which leads to conservative power approximations) requires knowing the dependence structure of the tests in advance. The difficulty of (correctly) analysing the power of multiple tests presents a serious

obstacle to 'safe' automation which, we have already argued, requires the careful management of error rates so as to provide guarantees that evaluations are fair across different devices and different human analysts.

In order to say *something* about power and to provide some provisional recommendations (under simplifying assumptions) we restrict further investigation to those methods that can be straightforwardly analysed. The Bonferroni correction controls the FWER by deriving the per-test significance level as follows: $\alpha_{per-test} = \frac{\alpha_{overall}}{m}$. The per-test power can thus be obtained by substituting $\alpha_{per-test}$ into the usual computations. Meanwhile, the power of a test which has been repeated as per the TVLA recommendations is the probability of the detection happening in both tests, which is the square of the power of a single test.

The blue and red lines in Figure 4 compare the power for a single experiment with the power for a repeat experiment for two values of α and a fixed effect size as the *total* sample size increases. That is, in the case of the repeat experiment the two separate acquisitions add up to the sample size for a single experiment. The impact on the power for a given sample size is substantial. For a significance level of 0.05, a total sample size sufficient to achieve a power of 80% ($\approx 20,000$) if one test outcome suffices has only 26% probability of identifying a true positive in both of two tests when it is required to confirm the results with the same overall amount of resources. For a significance level of 0.00001 (requiring $\approx 70,000$ in the single experiment case), the probability drops even lower, to around 6%. Over twice as many traces total are needed to achieve the same power when the experiment is repeated. The yellow, green and purple lines show the power attained when the Bonferroni correction is applied to the traces. When the per-test significance level is large (0.05) the power of the Bonferroni strategy is close to that of the repeated experiment for a short (100 point) trace; for longer traces it drops below it. For the very small significance level (chosen to correspond approximately to the TVLA criteria [31]), the traces have to be much longer (on the order of 100,000,000) before the repeated experiment becomes the more powerful option.

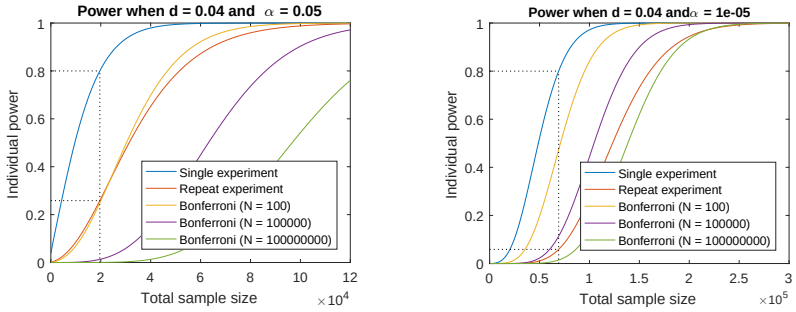

Figure 4: Per-test power to detect a standardised effect of size 0.04, for two different per-test significance levels, using a fixed total quantity of traces.

Of course, methods which allow for dependencies between the tests are likely to incur less damage to the power – but until we can provide concrete approximations of their impact it is not obvious how to incorporate them into *automated* test procedures. Interestingly, though, the situation may not be as bleak as it first appears: so far, we have only considered the per-test power (to detect a real leak of a certain minimum effect size). But just as the idea of multiple testing gives rise to different notions of overall Type I errors, it also implies alternative notions of power which, depending on the goal of the evaluation, may in fact be more relevant.

Different notions of power Porter [58] describes the per-test power (the probability of detecting a true effect wherever one exists) as 'individual' power, and contrasts it with the notion of 'r-minimal' power[3] – the probability of detecting at least r true effects. The relevant notion varies depending on the goal of the leakage evaluation: mapping the leakage or certifying the security (i.e. by finding no leaks having tested thoroughly) requires conserving the individual power of each test, while controlling the 1-minimal power may well be sufficient for certifying leakage or finding an attack, when what is important is that *some* leaks are found, not that *all* leaks are found.

In the case that the tests are independent, the probability of detecting *all* true effects is the product of the individual powers. (In a leakage scenario, we don't really expect independence so the product is likely to be conservatively low). The r-minimal power is naturally greater than or equal to this quantity. In particular, the

[3]Porter uses the terminology d-minimal; we use r instead of d to avoid confusion with Cohen's d.

1-minimal power can actually be *higher* in a multiple testing scenario than in a single test – as long as the true number of false positives is greater than one, each such test represents an additional opportunity to find an effect. So the situation for leakage detection, at least in the case that it is sufficient to simply show the existence of leakage, may not be as disheartening as the impact of multiplicity adjustments on individual power would imply.

By way of illustrative example, suppose (inspired by further analysis of the first of the three scenarios depicted in Figure 3) that there are around 20 true leakage points in an (AES software implementation) trace of length 12,000, and that the TVLA repeat experiment method is used to guard against false positives. Figure 5 presents the individual power, the power to detect all 20 leaks (under a simplifying independence assumption), and the r-minimal power for $r = 1$ and $r = 10$. For $\alpha = 0.05$ the sample size required to map leakage (i.e. find all leakage points, with a probability of at least 99%) is around 9 times the sample size required to certify vulnerability (i.e. conclude that there is leakage with the same probability). For $\alpha = 0.00001$ the sample sizes are all considerably larger, but the number of traces to map leakage is only around 3.5 times the number needed to certify leakage.

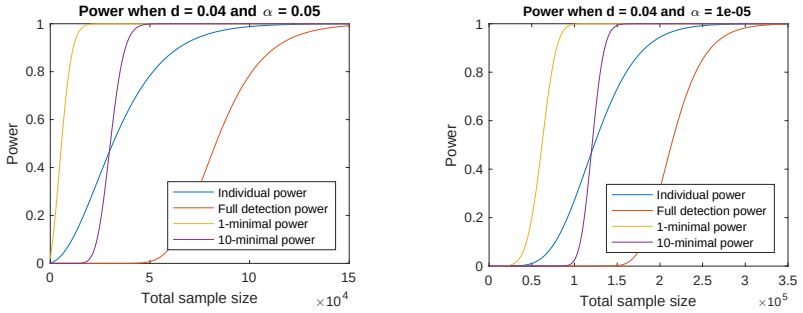

Figure 5: Different notions of power to detect a standardised effect of size 0.04, for two different per-test significance levels, where the TVLA recommendations are followed. We suppose that there are 20 'true' effects in a trace set of length 12,000.

3.3 Statistical methods for detection

All of the various proposals for leakage detection essentially have their basis in statistical hypothesis testing, although many have not been formalised as such and have tended to be applied in a more ad-hoc manner. However, correct and careful formalisation is necessary if the tests are to be automated in a manner that ensures fair and consistent evaluations across devices and leakage scenarios.

Common methods can be classified into 'arbitrary' (or 'non-specific') tests, aimed at detecting sensitive leakages of *any* form without necessarily being able to link them back to the operations producing them, and 'specific' tests, which target particular intermediates of interest. As briefly discussed in Section 3.2.1 above, neither type of test (even if applied comprehensively) can alone suit the needs of all the different potential goals of an evaluation process, so that both have roles to play.

3.3.1 Detecting arbitrary leaks

An 'ideal' arbitrary leakage detection would be one that tested the null hypothesis of independence between the side-channel trace and the sensitive data, versus the alternative that they are related, without further specifying the form of that relationship or relying on particular features of the algorithm. Such a situation-agnostic method would require virtually no input from the user aside from the traces, the sensitive data, and the parameters to achieve the desired Type I and Type II error rates, and would thus be an excellent candidate for automation.

However, this ideal is not fully realisable in practice: performing a hypothesis test requires the formulation of a test statistic that can be computed from the data sample and that has a known distribution under the null. (Meanwhile, evaluating the statistical power of the test requires also knowing the distribution under the alternative). This inevitably constrains the scope of the test, and also increases the burden on the user to provide additional information.

The most popular proposals aiming towards arbitrary detection are as follows:

Continuous Mutual Information (CMI) The CMI test [16] computes the mutual information between (bytes of) sensitive data and the points in the trace, and uses randomised resampling to approximate the 'zero MI' distribution, thus enabling inference about whether or not the observed quantities are significantly different from zero. Used in this manner, CMI is only able to detect leakages that exhibit before the cryptographic scheme performs 'mixing' operations. Still, of all currently available options, it imposes the fewest additional constraints on what can be detected, at the cost of having by far the highest computational and data costs [49]. Automation of this test would be challenging, due to the absence of neat analytical formulae for the power and sample size computations (arising from the difficulty of characterising the distribution under the alternative hypothesis). However, (non-automated) experimental analysis in well-understood representative scenarios, similar to the approach taken in [49], might produce parameters that can be safely re-used in future leakage detection tests.

Normalised Inter-Class Variance (NICV) This proposal (by Bhasin *et al.* [7]) is essentially an ANalysis Of VAriance (ANOVA) test against each point in the traces. Traces are partitioned according to a given portion of the input (usually a byte) and the ratio of explained-to-unexplained variation is computed. If the noise is Gaussian and the same magnitude for all classes, this quantity is known to have an F-distribution under the null hypothesis that the measurements do not depend on the partition, giving rise to a criterion for deciding on the presence of leakage (under some reasonable assumptions). Whilst not as comprehensive in scope as CMI (in particular, it is only able to detect differences in the first moments of the trace partitions, and remains sensitive only to pre-mixing operations) it can be far more efficiently computed and utilises known distributions in the decision criteria, thus avoiding the need for costly resampling methodologies. Statistical power analysis is likely to be more straightforward for ANOVA than for CMI, though the side-channel literature currently lacks an attempt to establish analytical formulae in this setting, which would be a useful next step towards automation.

Correlation test A similar approach to the above is to compute input-byte-dependent means at each point and to correlate them with the raw traces [25]. Intuitively, this also gives a measure of the extent to which the data 'explains' the measurements. If one is willing to assume that the traces and the fitted means have a bivariate Gaussian distribution, a formal hypothesis test can be constructed via Fisher's z-transform, which has an approximate standard normal distribution under the null hypothesis of zero correlation. Statistical power analysis is again lacking so far in the side-channel literature; it might be useful to explore which of correlation and NICV can be most readily automated in that regard.

Fixed-versus-random t-test The fixed-versus-random test is one of the many t-test-based proposals of Goodwill *et al.* collectively referred to as 'Test Vector Leakage Assessment' (TVLA) methodology [31]. Unlike the previous examples, it aims to capture differences induced by a change in the *full input* in one go, rather than testing separately for differences induced by changes in enumerable portions of the input. It does so by generating two (randomly interleaved) leakage sets: one corresponding to a single fixed input, the other to randomly generated input. A t-test is then performed to decide whether or not (and at which points in the trace) the two sets have statistically significant means. The relative ease of performing statistical power analysis for t-tests makes it much easier to envisage the fixed-versus-random test as an automated procedure, although it imposes very particular requirements on the measurement stage, and results in tailored acquisitions that are not necessarily transferable to other tasks. Another drawback is that it inevitably covers only a tiny part of a very large sample space. (The 'fixed' acquisition, for example, represents the leakage distribution for just one of a possible $2^b \times 2^b$ possible (input,key) pairs). It is advised to repeat the test with different fixed inputs; additional assumptions on the form of the leakage (such as 'equal images under different subkeys' [68]) reduce the *de facto* sample space, but the extent to which a small number of test cases can be trusted to represent device operation in the general case is open to question. This gives rise to the call for automated tests to be equipped with some means of reporting an indication of the 'coverage' they achieve (possibly a range of scores under different sets of assumptions) – a notion we discuss in Section 3.4.

3.3.2 Detecting specific leaks

An advantage shared by the above approaches is that they minimise the number of different tests needing to be performed, as well as the requirement for detailed knowledge of the implementation. They are ideal

if the goal of the evaluator is merely to certify vulnerability or to certify security. However, if the goal is instead to demonstrate a successful attack or to map the leakages to corresponding intermediate values, a so-called 'specific' test is needed, which looks directly for associations between the measured leakages and *known intermediates*. (The latter can be computed as long as the evaluator has access to the key, the inputs, and any randomness).

Goodwill *et al.* propose, for example, to perform t-tests on partitions constructed around known intermediate *bits* arising from random inputs, or around known *bytes* in a 'one value versus all other values' manner [31].

The NICV and correlation tests described above can be easily adapted to this 'specific test' setting, reducing the number of different tests needing to be performed in order to get reasonable coverage of the many potentially leaking states. (For example, in the latter case one F-test readily replaces 2^8 separate t-tests, and is more statistically rigorous).

Note that all of these tests are once more only able to capture data-dependencies that exhibit in the first moment of the trace distribution, and that they all depend to a greater or lesser extent on the assumption of normality.

As previously, the t-test based methods are most suited to automation, as the sample size computations are well understood in this case.

3.3.3 Higher-order moment-based detection

With the exception of CMI, all of the tests described above are sensitive only to information leaks which manifest in the first moment (i.e. the mean) of the partitioned traces. A popular solution for detecting higher-order data dependencies (arising, for example, from masking countermeasures [14]) is to process the raw traces in such a way as to 'shift' the information down from the higher-order moments into the mean of the resulting new distribution [69]. In the case that the higher-order data-dependencies reside in the univariate distributions of individual points (as has been observed of hardware masking) this can be achieved by raising (de-meaned) trace points to some respective power. Where the data dependencies occur instead in joint distributions (e.g. in the case of software masking), it entails multiplying tuples of distinct (de-meaned) points.

The quantities produced by either of these two procedures will have distributions that are distinctly not normal, violating the assumptions of the t-test in increasing measure as the number of multiplications increases. With a large enough number of traces, we can still expect that the distribution of the average in a standard distance-of-means test will itself tend to normality (by the central limit theorem), which is why in practical papers these methods perform reasonably well.

However, sound automation (we have argued) requires performing statistical power analysis so that, in the event the test fails to detect leakage, we can have some confidence that it is not simply a consequence of insufficient data. Unfortunately, our analytical formula 3 relies heavily on those assumptions of normality which have been inevitably violated by the multiplications. Thus, new sample size formulae need to be derived in order to safely automate t-tests on processed traces. This remains an avenue for future work.

The task of testing joint moments carries the additional challenge of efficiently searching all possible tuples, or of safely pre-selecting a reduced set of candidates without losing important information. More work needs to be done from a methodological perspective before discussions can be had about appropriate automation of such procedures.

3.4 Determining 'coverage'

We have already seen that interpreting the outcome of a detection test can be difficult, especially if it fails to find any leakage. Is the implementation therefore invulnerable, or is it simply that we chose the wrong attacks, targeted the wrong values, and/or failed to acquire enough trace measurements to confidently reject the null? In addition to correctly controlling for Type I and Type II errors (which guards against the latter possibility), any automated procedure needs to be able to output some meaningful indication of the extent to which all possible avenues have been explored. This can be seen as analogous to the idea of 'coverage' in code testing [53].

Typical metrics in this setting include code coverage (have all lines of code been touched by the test procedure?), function coverage (has each function been reached?), and branch coverage (have all branches been executed?) [1]. In a hardware setting one might alternatively (or additionally) test for toggle coverage (have all

binary nodes in the circuit been switched?) [77]. The appropriate choice of coverage metric can also depend on whether one assumes white- or black-box testing. The given examples all stem from white-box testing as they evidently assume access to the code. In the context of leakage detection, we could intuitively strive to 'test all intermediate values' that are created, which extends the idea of code coverage by making it explicit that in each line of code, one or several values are being touched and thus potentially leaked on. In black-box testing, lacking knowledge of and access to the source code, coverage tends to be defined in functional terms. Hence in this case, we could intuitively strive to 'test all input combinations' for leakage.

It is not always clear from the existing literature whether leakage detection is/should be considered a white- or a black-box pursuit. Within some specific evaluation schemes, e.g. Common Criteria, it *is* made explicit: for higher security levels, white-box testing becomes a necessity (excluding some hardware scenarios where it is simply not possible to access randomness). Under such conditions, it might be tempting to believe that it is (theoretically) possible to test leakages exhaustively.

However, leakage detection gives rise to unique challenges by comparison with the general software testing scenario, in that we are not aiming for a property which is present (or not) in a single line of code, or in a single function/unit at a time. The relationship between sensitive intermediates and side-channel leakage is potentially highly involved, even when examined at a single instance, with previous code, current and previous data (i.e. the 'state' that the device is in), as well as environmental factors all bringing an influence to bear (although we tend to ignore the latter based on the assumption that they amount to independent noise). Thus the simple-looking definition of single-point leakage that we gave earlier, $\mathrm{tr} = L \circ f_k(x) + \varepsilon$, is somewhat deceptive because L is in fact a complex function for many of the devices that we care about (see previous work on profiling [51]).

3.4.1 Coverage based on inputs

In settings where we do not want to make any assumptions about the internal working principles of a device, we are only able to formulate leakage detection tests based on inputs (outputs). An exhaustive evaluation would require running a statistical hypothesis test for *each possible* combination of inputs and keys (to the device/component performing the cryptographic procedure): i.e. $\forall(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$ and $k \in \mathcal{K}$ one would test a null hypothesis of the form (e.g.) $\mathrm{tr}(x_1, k) = \mathrm{tr}(x_2, k)$ or $\mathrm{CMI}(x, k) = 0$. This is clearly infeasible in practice and thus all suggestions in the literature imply a compromise to a greater or lesser extent. E.g. the fixed-vs-random test [31] suggests to collect one set of data for a fixed 16-byte (in the case of AES) input and another where the 16-byte input is chosen to vary at random. A variation [25] suggests to acquire trace sets for two fixed inputs and to test for differences between these, which can be more efficient (in terms of the sample size needed to detect). Clearly, should such a test reject the null hypothesis of equal means, one has successfully certified the presence of leakage. However, what can reasonably be concluded if such a test does *not* detect a difference? Choosing a single fixed input along with a fixed key means that of the total $2^{128} \cdot 2^{128}$ possible combinations of (x, k), only one has been sampled. In the fixed-versus-random case the other, 'random' sample still has a fixed key, so that in effect it is drawn uniformly from a greatly restricted subspace (of size 2^{128}) of the same total population; moreover, depending on how prevalent vulnerabilities are supposed to be across the subspace, the sample size may or may not be large enough to be considered 'representative'. In short, the fixed-versus-random test (and even more so the fixed-versus-fixed variant) provides extremely poor coverage of the input space, even if repeated with multiple input values, and even assuming that we have 'equal images under different subkeys' (EIS, [68]). It is also only sensitive to leakages in the first central moment – an aspect of 'coverage' which would not be reflected in an input-based score.

The CMI test has the advantage of capturing arbitrary (univariate) leakages (i.e. not merely those exhibiting in the central moments) and draws on a smaller de facto population as it tests directly for the relationship between a given input byte and the leakage without reference to a particular fixed input. However, unless we assume EIS, it is still testing only a small portion of the overall space since the computations are again necessarily with respect to a fixed key (or alternatively plaintext).

Since full input coverage is not a realistic aspiration, a useful output from an automated detection procedure might be the size of the *de facto* input space under a range of different assumptions about the form of the leakage (e.g. EIS), contrasted with the size of the sample tested in the analysis.

3.4.2 Coverage based on intermediate values

Given access to a description of the internal workings of a device, it becomes possible to formulate leakage detection tests based on intermediate values that we know to be present. Leveraging the t-test in this context entails partitioning the traces according to the value of a particular bit of the intermediate state – examples of what the TVLA framework describes as 'specific' tests, in contrast to the 'generic' fixed-vs-random approach [31]. A *comprehensive* evaluation of this type would require thus testing every single bit of every single intermediate. It might be tempting to assume that significantly better coverage can be achieved in this way than via tests that make no assumptions about the intermediates. However, our continued reliance on the t-test means that we are, by construction, incapable of recognising *any* leaks that are not contained in the first central moments of the partitions, e.g. leaks that are produced by crosstalk between bus wires, or transitions between consecutive states, and therefore require taking the interaction between bits into account. Using the CMI test overcomes this limitation in the first example, but not the second (unless the mutual information takes as input the XOR of consecutive bytes instead of the bytes themselves).

As with the above, in the interests of transparency an automated procedure based on 'specific' tests might output a summary of the number of intermediates touched by the test relative to the number untouched. The implications of such a metric depend again on the assumptions made about the leakage as well as the adversarial model of interest. For example, standard DPA typically targets the first and/or the last round (or two rounds, in the case of AES-256), where the intermediates depend on enumerable portions of key material (with some reliance on already recovered results in the two round case). Leakage of inner round intermediates may not be considered relevant against such an adversary, due to the difficulty of hypothesising over candidate values. This would reduce the effective denominator of the coverage metric.

3.4.3 Coverage based on leakage models

The most sophisticated reasoning for the quality of a leakage detection procedure can be made when some profiling information is available. Such a setting could be compared to *functional* testing, in which a developer, based on detailed understanding of the implementation, is able to design specific corner cases expected to produce 'interesting' behaviour. For instance, a leakage detection approach tailored for implementations on an ARM Cortex M0 [51] might aim to test specifically for Hamming-distance leakage in the case of specific assembly sequences (using e.g. a specific t-test or CMI). To do so would not increase the extent of the coverage, but it would improve the quality of the informed reasoning about that extent.

3.5 Application in the context of a non crypto-expert developer

In this section we consider what our findings and recommendations look like in practice for a non crypto-expert developer. The end-user we have in mind for WP2 is somebody working with embedded systems who is aware of (and concerned about) side-channels but lacks the precise knowledge and experience to conduct a battery of state-of-the-art attacks themselves. They wish to produce an implementation with appropriate software countermeasures (masking or hiding, for example), but because of the impossibility of pre-empting all possible vulnerabilities, they ideally need the facility to iteratively test and improve their design until it is fit for purpose.

Since developers are interested in flagging and addressing leaks, they need an analysis tool that is geared towards the particular goal of *highlighting vulnerabilities*, and that prioritises code coverage. To this end, we present ELM0 – a leakage emulator that uses carefully profiled instruction-level leakage models to predict the power consumption of arbitrary instruction code sequences when running on an ARM Cortex M0 (although the methodology can be extended to other devices). ELM0 has been developed, tested and released under WP3, building on an earlier version predating the project [51].

Unlike traces acquired from a real device, which are confounded by noise, may require pre-processing, and are very much not directly linkable to the (possibly unknown) code sequences producing them, ELM0 traces can be readily analysed and all discovered leaks related back to the instructions responsible, providing clear and actionable feedback throughout the development process.

Since the end-user is also assumed to be non-expert in the field of statistics, we also need to fit the tool with suitable default parameters for its built-in analysis functionalities – hence the overlap with WP2 and with this deliverable in particular. Since the aim is to detect arbitrary leaks, we are especially concerned about keeping the rate of false negatives low (although, of course, we still want a 'sensible' rate of false positives). Because

the effect size is fixed (that is, it is inherent to the profiled instruction-level models), the only way to increase the power (for a fixed α) is to vary the sample size N. We therefore consider that the safest (that is, most defensible) way to automate leakage detection in the simulated leakage setting is as follows:

- fix α to something widely accepted as 'sensible';
- fix the standardised effect size d according to our analysis of the profiled leakage models;
- allow the user to choose the sample size N;
- compute the associated β and report it as part of the test outcome, so that the developer can decide if it is adequate for purpose;
- repeat with a new (larger) choice of sample size N if the user requires.

Note that N is also implicitly a measure of 'effort' or time, which can be tuned by developer according to budget.

We have implemented these recommendations in our latest version of ELM0 which is released concurrently with this deliverable as D3.2.

4 (Deep) machine learning for automated attacks and detection

Machine Learning (ML) is a branch of Artificial Intelligence whereby relationships found in data are used to inform predictions or decisions, bypassing the need for explicit task programming. Ideally it minimises reliance on prior knowledge and user assumptions and maximises the scope of what may be discovered – all in a consistent (because automated) manner which is resilient to human error. Deep Learning (DL) is a special case of ML, which mimics the multi-layer structure of neurons in the brain in order to handle increasingly complex data and find increasingly complex relationships. In particular, DL methods are typically designed with 'raw' data in mind, so that pre-processing and feature selection are implicit to the learning process, thereby reducing even further what is required in the way of user input or knowledge.

The techniques of DL (and ML more generally), combined with computing technology's rapidly growing capacity for collecting, storing and operating on "big data," are transforming (though not unproblematically) the performance of tasks which once required human judgement – including many applications where quality control and threat detection procedures are in place (production lines[4], fraud detection[5], border control[6], surveillance[7], cyber security[8], software testing[9]). In keeping with this trend, they have caught the attention of many side-channel analysts as an avenue for more effective and/or more automated leakage exploitation and detection.

4.1 Existing literature

Attacks Most of the proposed applications so far have been for new attacks, typically mirroring the basic procedure of existing attacks, but swapping out the usual statistical components for more advanced alternatives. For example, profiled attacks can be performed by replacing Gaussian template building and Bayesian classification [15] with other non-'deep' ML procedures such as support vector machines [33, 36], random forests [41], and k-nearest neighbours [3], as well as emerging DL procedures such as self-organising maps [41], convolutional neural networks (CNN) [12, 63] (both report work carried out within this project), and multi-layer

[4] https://www.forbes.com/sites/insights-intelai/2018/07/17/how-ai-builds-a-better-manufacturing-process

[5] https://www.forbes.com/sites/theyec/2018/06/04/artificial-intelligence-and-the-future-of-financial-fraud-detection

[6] https://www.cnbc.com/2018/05/15/lie-detectors-with-artificial-intelligence-are-future-of-border-security.html

[7] https://www.theverge.com/2018/1/23/16907238/artificial-intelligence-surveillance-cameras-security

[8] https://www.wired.com/story/ai-machine-learning-cybersecurity/

[9] https://www.aitesting.org/

perceptrons (MLP) [43, 63]. An initiative from Google following the same strategy was presented at CHES 2018 [10] and Defcon 2019 [11].

In the realm of non-profiled attacks, any method that learns a clustering or a model from labelled data can be repeatedly applied to an enumerable number of different hypothetical labelling schemes (i.e., each derived from a different key guess), with quality metrics for the resulting arrangements or models being then used to hopefully distinguish the 'correct' scheme and the associated key guess from the incorrect alternatives. This has been demonstrated using several ML methods (principal components analysis [73], linear discriminant analysis [45], kernel discriminant analysis (by us) [92], linear regression [23], stepwise regression [87]) including, in a recent work, 'deep' ones (MLPs and CNNs [80]).

Detection Whilst demonstrating an attack does implicitly prove the presence of leakage, so far little (that we are aware of) has been done to apply either DL or ML more generally to the specific task of leakage detection. The closest ML candidates are the use of linear discriminant analysis [75] and (within this project) kernel discriminant analysis [13] to find or construct points (or tuples) of interest from trace acquisitions corresponding to (potentially protected) implementations. Whilst the resulting lower-dimension sets are processed combinations of the original points, nonetheless the weights can help to indicate the location of the leakage. Recent works [90, 47] have also explored visualization techniques like weight visualization, gradient visualization, and heatmaps to identify leakages based on trained neural network models.

Portability The question of model portability, i.e. *can a model built on a device A be used to attack a similar device B?*, is a general issue with profiled attacks. A recent work [21] has concluded positively to this question, presenting a successful experiment of DL-based SCA across devices. More research is required on this question, but these results are promising.

4.2 Towards automated leakage detection using deep learning

Part of the appeal of MLPs, CNNs and other DL methods for side-channel analysis lies in their potential (when implemented appropriately) to learn and apply knowledge from raw data, with minimal user input about the form of the relationships expected and without the need to pre-select relevant features or perform dimensionality reduction or pre-processing. This has already proved promising in scenarios involving countermeasures such as masking, or where traces are desynchronised (naturally or otherwise): DL methods have been able to recover sensitive information without the prior realignment and/or point combining steps that would be required by 'classical' attacks such as correlation DPA [80, 63]. In fact, work done within this project has shown that artificially *added* desynchronisation can actually *improve* the performance of CNNS [12]. The performance of DL methods against uncompressed traces has not yet been much explored, as most existing works (including all those using the dedicated publicly available ASCAD dataset [63]) begin with measurements that have already been truncated around the activity of interest, in order to reduce storage and computational complexity.

An obstacle to the useful application of DL for side-channel analysis is that there tends to be numerous ways to configure a particular architecture (Kim et al. point out that, for a given design principle, any parametrisation can essentially be considered a "new architecture" [37]) and whilst some of the DL-based side-channel studies have explored different hyperparameter choices [57, 63], still they are barely able to scratch the surface relative to the full array of possibilities. In addition, various (formal and informal) "no free lunch" theorems highlight the impossibility of arriving at a one-size-fits-all solution that works equally well across all learning environments [88]. This increases the dependence on prior knowledge in order to choose a well-adapted method, and is particularly problematic in the context of leakage evaluation, where *fairness* requires procedures that operate predictably, replicably and like-for-like against different test targets. We would not, for instance, have any means of understanding or controlling the error rates, making it especially difficult to 'certify security' (i.e. the *absence* of a vulnerability) on the basis of a DL-based evaluation.

Another downside of using DL for side-channel evaluation is that the resulting models tend to be non-intuitive and somewhat opaque. This means that, even if a test is successful at identifying the presence of leakage, it may not be possible to unpick the precise contributory factors so as to locate and address the cause. Thus, DL might be able to achieve a goal of 'certifying leakage' but more work would be needed if the evaluator wished to 'highlight vulnerabilities'.

Drawbacks aside, DL for SCA is a very recent field of exploration and notable progress is expected in the future. For instance, in the CHES' and Defcon's talks mentioned above, researchers from Google presented an approach based on automatic hyperparameter selection using a large computational effort (more than 1,000 models trained to attack a single unprotected AES implementation). In an attempt to bypass the usual key-guessing process (namely, DPA), the researchers tried to train deep neural nets to recover the AES key *directly* from the unprocessed measurements. This approach was unsuccessful, and more classical distinguisher-based DPA techniques were subsequently applied, using the aforementioned automatic search to select model parameters. It succeeded in finding efficient models, however none of them were simple or intuitive. This illustrates that specifying a good model structure for side-channel evaluation is still an open question, and seems even more complex when considering protected implementations.

There is thus a sense in the field that, since DL represents the state-of-the-art in the tools available to an attacker, evaluation procedures need to also take such methodologies into account so that their conclusions are relevant to the worst case. Cyber-security agencies SGDSN and BSI, for instance, now require labs to perform some NN-based procedures in settings where adversaries with profiling capabilities are taken into consideration. It therefore continues to be a hot research topic, and one which is relevant to several aspects of this project.

Project contributions (beside WP2) Work package 1 (see D1.2 and D1.4) has explored the advantages of deep learning methodologies from an efficiency perspective: CNN-based attacks have been applied to both AES and to masked ECC using Montgomery ladder implementations of scalar multiplication, and were found capable of bypassing misalignment, jitter-based countermeasures and masking countermeasures without pre-processing. Performance enhancements have been identified, including: class balancing, the use of appropriate leakage models, depth and width reductions (reducing computational complexity whilst preserving classification accuracy), smaller learning rates and smaller batch rates, and the avoidance of regularisation where possible. The adaptations made have produced considerable efficiency gains (shorter training times and fewer training epochs) relative to existing published work.

Another contribution of work package 1 is the introduction of the multi-channel approach, which performs binary classification on (correctly or incorrectly) labelled sets of traces in order to decide whether the key guess that was used to produce the labels is the true key or a false alternative (see D1.2 for details). This approach has been initially developed in order to tackle some limitations of the standard classifier approach in the context of SCA and improve the performance of DL-based profiled attacks, but it also shows promising features towards automated leakage detection as detailed below.

4.2.1 Contributions

Multi-channel classification The multi-channel classifier approach for SCA has been introduced in work package 1 (see D1.2 and D1.4). It is automated in two respects: first, it can operate on raw, possibly misaligned traces without prior processing, leveraging the well-known property of CNNs. The same potential applies to CNN-based standard classifiers, but we have shown that the results obtained with standard classifiers are very dependent on the leakage model assumptions. This implies that a thorough characterization of the leakage of a device is a necessary step to ensure the soundness of an evaluation using a standard classifier, which goes against the intent to automate the evaluation process. Moreover, even assuming that an automated leakage characterization would be possible, the standard classifier is known to work poorly with unbalanced leakage models – an issue known as 'class imbalance' [56] in the literature, that does not affect the multi-channel classifier due to its different design. Therefore, we deem the multi-channel neural network method better suited than CNN-based standard classifiers for automated leakage detection purposes.

Secondly, and most importantly, the multi-channel approach relaxes the assumptions on the leakage model due to the different input/output structure of the neural network. In a classical classifier, the output of the NN being the possible intermediate leakage values, the choice of a correct leakage model is fundamental, otherwise learning cannot happen efficiently, as mentioned above. Since the NN of the multi-channel classifier does not aim at classifying the intermediate leakage values but rather at distinguishing trace permutations corresponding to a correct or wrong key hypothesis, it does not require a prior leakage model characterization. This feature seems especially interesting in the context of automated leakage detection, particularly in a black-box context, or when an evaluator doesn't have access to values required for the characterization of the leakage of an implementation.

Sensitivity mapping Recently, the SCA community has benefited from the resurgence of Convolutional Neural Networks (CNNs) in the 2010's by applying them to profiling attacks. They are seen as a black-box tool whose main advantage is that they do not require pre-processing, and they have been found at least as efficient as other state-of-the-art profiling attacks. However, from an evaluator's point-of-view, this is not sufficient. On the one hand she wants to make sure that a CNN attack succeeds for good reasons i.e. that the learned model can generalize to new data. On the other hand the evaluator also wants to help the developer to localize and understand where the vulnerability comes from in order to remove or at least reduce it. This issue is part of a more general problematic in DL-based systems, namely their explainability and interpretability. To address it, a theoretical framework has recently been proposed [54], and several methods have been tested to tackle the issue. In particular, some computer vision research groups have studied the so-called *sensitivity analysis* [71, 74] which is derived from the computation of the loss function gradient with respect to the input data during the training phase.

Recent work within this project [48] proposes to apply a particular sensitivity analysis method called Gradient Visualization (GV) in order to better understand how a CNN can learn to predict the sensitive variable based on the analysis of a single trace. The main claim is that CNN-based models succeed in discriminating PoIs from non-informative points, and their localization can be deduced by simply looking at the gradient of the loss function with respect to the input traces for a trained model. This work theoretically shows that the method can be used to localize PoIs in the case of a perfect model. The efficiency of the proposed method does not decrease when countermeasures like masking or misalignment are applied. In addition, the characterization can be made for each trace individually. The theoretical efficiency was verified on simulated data and on experimental traces from a public side-channel database, empirically demonstrating that GV is at least as good as state-of-the-art characterization methods, in the presence (as well as absence) of different countermeasures.

A (partially) automated evaluation framework would certainly benefit from this work's use of CNNs to find points of interest even in the presence of masking and misalignment – a task which, as observed above, overlaps considerably with that of (automated) leakage detection, and suggests the potential, perhaps, to move from the goal of 'certifying leakage' towards that of 'highlighting vulnerabilities'.

5 Conclusion

In this deliverable we have set out the challenges that make leakage detection and side-channel evaluation hard to automate. Prominently, the dependencies between the evaluation steps that necessitate backtracking in the evaluation workflow imply that any automation strategy is, to some extent, a compromise between the degree of evaluator interaction and the level of assurance one has about the evaluation outcome. However, we have identified some promising areas in which measures can be taken to mitigate this trade-off, such as: statistical hypothesis testing for leakage detection, leakage detection coverage assessment, and leakage detection using deep-learning techniques. Within these themes we have provided a portfolio of methods to apply to improve the automation of side-channel evaluation, and identified promising research directions for further investigation.

References

[1] P. Ammann and J. Offutt. *Introduction to Software Testing*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.

[2] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template Attacks in Principal Subspaces. In L. Goubin and M. Matsui, editors, *CHES 2006*, volume 4249 of *LNCS*, pages 1–14. Springer, 2006.

[3] V. Banciu, E. Oswald, and C. Whitnall. Reliable information extraction for single trace attacks. In W. Nebel and D. Atienza, editors, *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, pages 133–138. ACM, 2015.

[4] L. Batina, J. Hogenboom, and J. van Woudenberg. Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In O. Dunkelman, editor, *CT-RSA 2012*, volume 7178 of *LNCS*, pages 383–397. Springer Berlin / Heidelberg, 2012.

[5] Y. Benjamini and Y. Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[6] D. J. Bernstein, T. Lange, and C. van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptology ePrint Archive*, 2015:221, 2015.

[7] S. Bhasin, J. Danger, S. Guilley, and Z. Najm. Side-channel leakage and trace compression using normalized inter-class variance. In R. B. Lee and W. Shi, editors, *HASP 2014, Hardware and Architectural Support for Security and Privacy, Minneapolis, MN, USA, June 15, 2014*, pages 7:1–7:9. ACM, 2014.

[8] R. Bi and P. Liu. Sample size calculation while controlling false discovery rate for differential expression analysis with RNA-sequencing experiments. *BMC Bioinformatics*, 17(1):146, Mar 2016.

[9] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Proceedings of CHES 2004*, volume 3156 of *LNCS*, pages 135–152. Springer Berlin / Heidelberg, 2004.

[10] E. Bursztein. Leveraging deep-learning to perform sca attacks against aes implementations, 2018. Invited talk at CHES 2018.

[11] E. Bursztein and J.-M. Picod. A hacker guide to deep-learning based aes side channel attacks. `https://elie.net/tag/scaaml/`, 2014. Talk at Defcon 2019.

[12] E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In W. Fischer and N. Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[13] E. Cagli, C. Dumas, and E. Prouff. Kernel discriminant analysis for information extraction in the presence of masking. In K. Lemke-Rust and M. Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2017.

[14] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.

[15] S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski, Ç. Koç, and C. Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 51–62. Springer Berlin / Heidelberg, 2003.

[16] T. Chothia and A. Guha. A Statistical Test for Information Leaks Using Continuous Mutual Information. In *CSF*, pages 177–190, 2011.

[17] O. Choudary and M. G. Kuhn. Efficient template attacks. In *CARDIS 2013*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.

[18] J. Cohen. *Statistical power analysis for the behavioral sciences*. Routledge, 1988.

[19] M. Columb and M. Atkinson. Statistical analysis: Sample size and power estimations. *BJA Education*, 16(5):159–161, 2016.

[20] J. Cooper, E. De Mulder, G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. Test vector leakage assessment (TVLA) methodology in practice. International Cryptographic Module Conference, 2013.

[21] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-deepsca: Cross-device deep learning side channel attack*. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2019.

[22] A. Dehbaoui, V. Lomné, P. Maurine, L. Torres, and M. Robert. Enhancing electromagnetic attacks using spectral coherence based cartography. In J. Becker, M. O. Johann, and R. Reis, editors, *VLSI-SoC: Technologies for Systems Integration - 17th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2009, Florianópolis, Brazil, October 12-14, 2009, Revised Selected Papers*, volume 360 of *IFIP Advances in Information and Communication Technology*, pages 135–155. Springer, 2011.

[23] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate Side Channel Attacks and Leakage Modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.

[24] O. J. Dunn. Multiple Comparisons among Means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[25] F. Durvaux and F. Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In M. Fischlin and J. Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[26] F. Durvaux, F. Standaert, N. Veyrat-Charvillon, J. Mairy, and Y. Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. In S. Mangard and A. Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2015.

[27] B. Efron. Size, power and false discovery rates. *The Annals of Statistics*, 35(4):1351–1377, 08 2007.

[28] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis: A Generic Side-Channel Distinguisher. In E. Oswald and P. Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer–Verlag Berlin, 2008.

[29] B. Gierlichs, K. Lemke-Rust, and C. Paar. Templates vs. Stochastic Methods. In L. Goubin and M. Matsui, editors, *Proceedings of CHES 2006*, volume 4249 of *LNCS*, pages 15–29. Springer, 2006.

[30] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F. Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In G. Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.

[31] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side-channel resistance validation. In *NIST Non-invasive attack testing workshop*, 2011.

[32] W. S. Gosset. The probable error of a mean. *Biometrika*, 6(1):1–25, March 1908. Originally published under the pseudonym "Student".

[33] A. Heuser and M. Zohner. Intelligent Machine Homicide. In W. Schindler and S. Huss, editors, *COSADE 2012*, volume 7275 of *LNCS*, pages 249–264. Springer Berlin Heidelberg, 2012.

[34] Y. Hochberg and A. C. Tamhane. *Multiple Comparison Procedures*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[35] J. M. Hoenig and D. M. Heisey. The Abuse of Power. *The American Statistician*, 55(1):19–24, 2001.

[36] G. Hospodar, B. Gierlichs, E. D. Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptographic Engineering*, 1(4):293–302, 2011.

[37] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic. Make Some Noise: Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. *IACR Cryptology ePrint Archive*, 2018:1023, 2018.

[38] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[39] A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell' Istituto Italiano degli Attuari*, 4:83–91, 1933.

[40] K. Lemke-Rust and C. Paar. Gaussian Mixture Models for Higher-Order Side Channel Analysis. In P. Paillier and I. Verbauwhede, editors, *Proceedings of CHES 2007*, volume 4727 of *LNCS*, pages 14–27. Springer, 2007.

[41] L. Lerman, G. Bontempi, and O. Markowitch. Power analysis attack: an approach based on machine learning. *IJACT*, 3(2):97–115, 2014.

[42] P. Liu and J. T. G. Hwang. Quick calculation for sample size while controlling false discovery rate with application to microarray analysis. *Bioinformatics*, 23(6):739–746, 2007.

[43] H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. In C. Carlet, M. A. Hasan, and V. Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.

[44] R. Mahmudlu, V. Banciu, L. Batina, and I. Buhan. Lda-based clustering as a side-channel distinguisher. In G. P. Hancke and K. Markantonakis, editors, *Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers*, volume 10155 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2016.

[45] R. Mahmudlu, V. Banciu, L. Batina, and I. Buhan. Lda-based clustering as a side-channel distinguisher. In G. P. Hancke and K. Markantonakis, editors, *Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers*, volume 10155 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2017.

[46] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[47] L. Masure, C. Dumas, and E. Prouff. Gradient visualization for general characterization in profiling attacks. In I. Polian and M. Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 145–167, Cham, 2019. Springer International Publishing.

[48] L. Masure, C. Dumas, and E. Prouff. Gradient visualization for general characterization in profiling attacks. In I. Polian and M. Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 145–167. Springer, 2019.

[49] L. Mather, E. Oswald, J. Bandenburg, and M. Wójcik. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 486–505, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[50] D. McCann and E. Oswald. Practical evaluation of masking software countermeasures on an IoT processor. In *IEEE 2nd International Verification and Security Workshop, IVSW 2017, Thessaloniki, Greece, July 3-5, 2017*, pages 1–6. IEEE, 2017.

[51] D. McCann, E. Oswald, and C. Whitnall. Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In E. Kirda and T. Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, pages 199–216. USENIX Association, 2017.

[52] T. S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Ç. Koç and C. Paar, editors, *Proceedings of CHES 2000*, volume 1965 of *LNCS*, pages 27–78, London, UK, 2000. Springer Berlin / Heidelberg.

[53] J. C. Miller and C. J. Maloney. Systematic Mistake Analysis of Digital Computer Programs. *Commun. ACM*, 6(2):58–63, Feb. 1963.

[54] G. Montavon, W. Samek, and K. Müller. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15, 2018.

[55] R. Ondusko, M. Marbach, R. P. Ramachandran, and L. M. Head. Blind Signal-to-Noise Ratio Estimation of Speech Based on Vector Quantizer Classifiers and Decision Level Fusion. *Journal of Signal Processing Systems*, 89(2):335–345, Nov 2017.

[56] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni. The curse of class imbalance in side-channel evaluation. 2019(1):209–237, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/7339`.

[57] S. Picek, I. P. Samiotis, A. Heuser, J. Kim, S. Bhasin, and A. Legay. On the performance of deep learning for side-channel analysis. *IACR Cryptology ePrint Archive*, 2018:4, 2018.

[58] K. E. Porter. Statistical power in evaluations that investigate effects on multiple outcomes: A guide for researchers. *Journal of Research on Educational Effectiveness*, 0(0):1–29, 2017.

[59] S. Pounds and C. Cheng. Sample size determination for the false discovery rate. *Bioinformatics*, 21(23):4263–4271, 2005.

[60] R. Poussier, V. Grosso, and F. Standaert. Comparing approaches to rank estimation for side-channel security evaluations. In *CARDIS 2015*, pages 125–142, 2015.

[61] S. M. D. Pozo and F. Standaert. Blind source separation from single measurements using singular spectrum analysis. In T. Güneysu and H. Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 42–59. Springer, 2015.

[62] E. Prouff, M. Rivain, and R. Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Transactions on Computers*, 58(6):799–811, June 2009.

[63] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[64] O. Reparaz, B. Gierlichs, and I. Verbauwhede. Selecting time samples for multivariate DPA attacks. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012 – 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

[65] Riscure. Inspector SCA Security Tool. `https://www.riscure.com/security-tools/inspector-sca/`.

[66] S. S. Sawilowsky. New effect size rules of thumb. *Journal of Modern Applied Statistical Methods*, 8(2):597–599, 2009.

[67] O. Schimmel, P. Duplys, E. Boehl, J. Hayek, R. Bosch, and W. Rosenstiel. Correlation power analysis in frequency domain. In *COSADE 2010 First International Workshop on Constructive SideChannel Analysis and Secure Design*, 2010.

[68] W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In J. Rao and B. Sunar, editors, *CHES 2005*, volume 3659 of *LNCS*, pages 30–46. Springer Berlin / Heidelberg, 2005.

[69] T. Schneider and A. Moradi. Leakage assessment methodology – A clear roadmap for side-channel evaluations. In T. Güneysu and H. Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[70] W. R. Schumm, M. Higgins, L. Lockett, S. Huang, N. Abdullah, A. Asiri, K. Clark, and K. McClish. Does dividing the range by four provide an accurate estimate of a standard deviation in family science research? a teaching editorial. *Marriage & Family Review*, 53(1):1–23, 2017.

[71] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, december 2013. arXiv:1312.6034 [cs].

[72] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *Ann. Math. Statist.*, 19(2):279–281, 06 1948.

[73] Y. Souissi, M. Nassar, S. Guilley, J. Danger, and F. Flament. First principal components analysis: A new side channel distinguisher. In K. H. Rhee and D. Nyang, editors, *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*, volume 6829 of *Lecture Notes in Computer Science*, pages 407–419. Springer, 2011.

[74] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net, december 2014. arXiv:1412.6806 [cs].

[75] F. Standaert and C. Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[76] F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In A. Joux, editor, *EUROCRYPT '09*, volume 5479 of *LNCS*, pages 443–461, Berlin, Heidelberg, 2009. Springer–Verlag.

[77] S. Tasiran and K. Keutzer. Coverage metrics for functional validation of hardware designs. *IEEE Des. Test*, 18(4):36–45, jul 2001.

[78] A. Thillard, E. Prouff, and T. Roche. Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, pages 21–36, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[79] L. Thomas. Retrospective power analysis. *Conservation Biology*, 11(1):276–280, 1997.

[80] B. Timon. Non-profiled deep learning-based side-channel attacks. *IACR Cryptology ePrint Archive*, 2018:196, 2018.

[81] S. Tiran, S. Ordas, Y. Teglia, M. Agoyan, and P. Maurine. A model of the leakage in the frequency domain and its application to CPA and DPA. *J. Cryptographic Engineering*, 4(3):197–212, 2014.

[82] T. Tong and H. Zhao. Practical guidelines for assessing power and false discovery rate for fixed sample size in microarray experiments. *Statistics in medicine*, 27:1960–72, 05 2008.

[83] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker. Improving Differential Power Analysis by Elastic Alignment. In A. Kiayias, editor, *Topics in Cryptology – CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 104–119, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[84] N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F.-X. Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.

[85] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert. Security Evaluations beyond Computing Power. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.

[86] Z. Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

[87] C. Whitnall, E. Oswald, and F.-X. Standaert. The myth of generic DPA...and the magic of learning. *IACR Cryptology ePrint Archive*, 2012:256, 2012.

[88] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr. 1997.

[89] D. Wu, X. Gu, and Q. Guo. Blind Signal-to-Noise Ratio Estimation Algorithm with Small Samples for Wireless Digital Communications. In D.-S. Huang, K. Li, and G. W. Irwin, editors, *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006*, pages 741–748, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[90] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019.

[91] L. Zhang, A. A. Ding, F. Durvaux, F.-X. Standaert, and Y. Fei. Towards Sound and Optimal Leakage Detection Procedure. Cryptology ePrint Archive, Report 2017/287, 2017. `http://eprint.iacr.org/2017/287`.

[92] X. Zhou, C. Whitnall, E. Oswald, D. Sun, and Z. Wang. A novel use of kernel discriminant analysis as a higher-order side-channel distinguisher. In T. Eisenbarth and Y. Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2018.

