REASSURE

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

REASSURE

Deliverable D2 . 2

Interim report on automation

Editor:	E. Oswald (BRI)
Deliverable nature:	R
Dissemination level: (Confidentiality)	PU
Delivery date:	December 31, 2017
Version:	1.0
Total number of pages:	24
Keywords:	evaluation, automation, side-channels, coverage, hypothesis testing

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 represents interim 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 multiple comparison corrections and statistical power analyses 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. A number of currently used methods (in particular, those based on ANOVA-like tests, and those based on correlation) sit in a middle ground, where the side-channel literature has not yet fully explored the solutions already proposed in the statistics literature. This represents an avenue for future work, which could potentially make it possible to safely automate a wider range of tests, as well as providing insights into the trade-offs between them.

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. Meanwhile, unless evaluators have control over (or at least access to) any randomness, as well as sufficient details about the implementation specification, the task of searching for jointly leaking tuples can quickly become infeasible as the leakage order increases. 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.

Finally, we propose that there exists a need to devise measures of 'coverage', inspired by the notion of code coverage in software testing. We offer some suggestions of what this 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.

We intend to make the formal definition of coverage metrics an avenue for exploration during the remainder of the project, along with other outstanding questions identified in this document. A future deliverable (D2.5) will report on progress made to that end, and finalise our findings and recommendations.

List of authors

Company	Author
BRI	E. Oswald
BRI	C. Whitnall

Other contributions gratefully received from

Company	Author
NXP	V. Verneuil
UCL	R. Poussier

Revision history

Revision number	Date	Comment
1.0	December 2017	Final Public Version

Contents

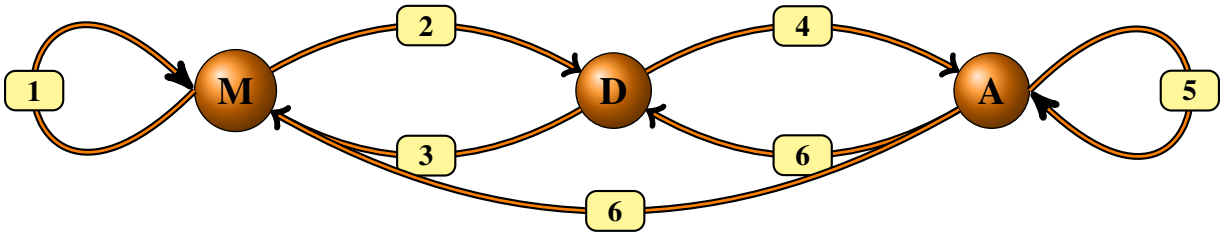

Figure 1: The three components comprising a typical evaluation, and the interactions between them.

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 [38], 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 [51, 33]. 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 [30] 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 [47] 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. [35]

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. [14]

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 [39, 46].

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 [3]. 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 [11] 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 [26]. 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 [27, Ch. 4], ANOVA (aka Normalised Inter-Class Variance) [6], Sum of Squared (T-)Differences [18], correlation with a standard or profiled power model [7, 15], mutual information [37]) 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 [20], 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) [13], 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 [30]), 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 [23], or correlation DPA [7] 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 [9] 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 [23], correlation with a selection of power models [7], mutual information [17], 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 [43]. Additionally, with a bit more computational effort the global key rank as the sample size increases could also be estimated [5, 19, 34, 48, 49]. 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 [8]) 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 [8, 31, 36]) as per the third option above. Optionally, a profiling step could be incorporated, as per the second option [25].

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 [16] 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 [29].

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 [21]) or by the empirical distribution function (e.g. two sample Kolmogorov–Smirnov [24, 42]). 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) and implementing the appropriate multiplicity corrections (we discuss this latter measure separately).

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 [28]. 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.

There are two main ways of correcting for multiple tests: controlling the *familywise error rate* (FWER) [22] and controlling the *false discovery rate* (FDR) [4]. Both of these were discussed in the context of leakage detection by Mather *et al.* [28]. To summarise, the FDR – defined as the expected proportion of *all* discoveries (tests which decide to reject the null hypothesis) which are in fact *false* discoveries (i.e. the null was true after all) – was deemed the more appropriate of the two for leakage detection because it retains the larger power (i.e. it maintains a reasonable Type II error). A recent proposal [52] 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 Šidák correction [50] 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.

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 [12]. 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 [20]), 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 $\mu_0 = 0$.

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

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

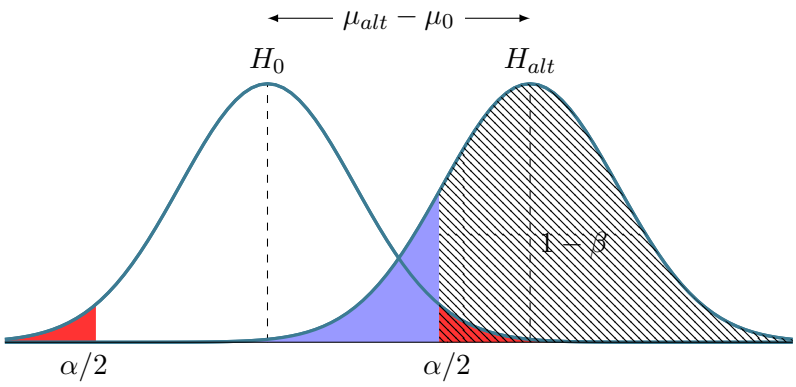

Figure 2: Figure showing the Type I and II error probabilities, α and β as well as the effect size $\mu_{alt} - \mu_0$ for a specific alternative such that $\mu_{alt} > \mu_0$.

$|\mu_A - \mu_B|$ 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 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 of the test, which is less obvious from Figure 2, is to increase the sample size for the experiment. This reduces the sampling variance of the test statistic, thereby 'squeezing' the distributions towards their means resulting in a smaller overlap without compromising on the effect size.

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 $n = |\mathrm{Tr}|/2$ 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:

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

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 evidence (such as suitable values for the variance and effect size) gathered from previous experiments. It is generally recommended to choose conservative values, e.g. the largest of previously observed variances, and the smallest effect size, and ideally to provide a sensitivity analysis by reporting outcomes under different choices. Alternatively, the effect size can be chosen based on *a priori* beliefs about what would constitute 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).

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 [28], 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.

3.1.3 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 $1 - \beta$, an expected (or interesting) effect size $\mu_A - \mu_B$, and their best knowledge of the population variance σ^2.
- The procedure computes the per-test significance level $\alpha_{\{\text{per test}\}}$ corresponding to the false discovery rate 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. With all these options, more needs to be done to address how best to understand and/or set effect sizes in the context of leakage detection.

A tempting 'solution' to the effect size problem is to estimate it from the test data. This amounts to 'post hoc' power analysis [45] – a practice strongly advised against in the statistical literature. It is essentially a form of circular reasoning, and it 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.

Of course, once meaningful parameters have been decided on for a given test scenario, they can be provided to all future measurements so that repeated experiments can be performed without interruption.

3.2 How to choose 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 [10], and the non-specific t-test [20]; 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 [20] or the normalised inter-class covariance (NICV; see Section 3.3.2) [6] 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.

3.3 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 [10] 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 [28]. 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 [28], 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.* [6]) 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 [15]. 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 [20]. 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' [40]) 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 [20].

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 [8]) 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 [41]. 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 [32].

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?) [44]. 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 [30]).

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 [20] 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 [15] 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, [40]). 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 [20]. 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 [30] 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.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

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

[43] 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.

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

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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

[51] 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.

[52] 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.

