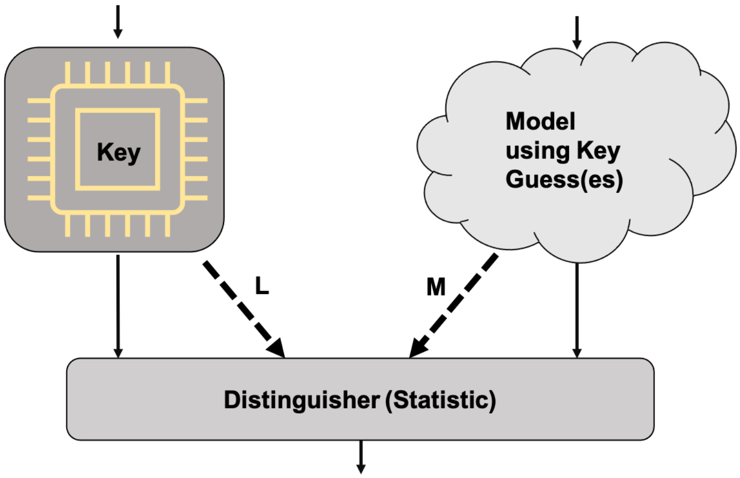
Side Channels: Attacks, Defences, and Evaluation Schemes
Part 2 — Evaluation Techniques

Elisabeth Oswald, James Howe

University of Klagenfurt and University of Bristol

ROADMAP

Principle of Side Channel Attacks for Key Recovery



Attack Strategies

Without Profiling

Single trace, multiple targets: using "visual inspection"

Multiple trace, single target: aka "Differential Attacks" (DPA, DTA, DEMA)

With Profiling

Single trace, multiple targets: using belief propagation, or pragmatic enumeration, ...

Multiple trace, single targets: aka "template based DPA attacks"

Multiple trace, multiple targets: using belief prop, combining probabilities

Profiling techniques: pdf estimation (univariate or multivariate, directly going for parameters, or via regression), ML/DL techniques (supervised and unsupervised)

"Generic Attacks": using"nominal models" together with "generic distinguishers"

Signal vs Noise

A trace consists of many points $L = \mathcal{L}(\mathbf{S}) + R$ with $X, k^* \subset \mathbf{S}$. Each data point is thus the “sum” of data depending leakage (aka, signal) and data independent leakage (aka, noise).

One way to define a signal to noise ratio (SNR) is to base it on the variance of the “data dependent” observable leakage $Var(\mathcal{L})$ over the variance of the “data independent” observable leakage $Var(R)$:

$$SNR = \frac{Var(\mathcal{L})}{Var(R)}.$$

(As in [15].) Obviously: the higher the signal in relation to the noise, the better attacks can be assuming we can **exploit** the signal.

Three Mitigation Strategies

There are only three known mitigation strategies to counteract the aforementioned attack strategies: all relate to the SNR.

- Reduce the signal: hiding in hardware
- Increase the noise: masking (and hiding)
- Limit the number of observations: mode level mitigation

Over the years a trend towards “provably secure” countermeasures has emerged in the literature.

But the fact that a technique comes with a proof does not change the fact it does either reduce the signal, or increase the noise, or limit the number of observations. No existing countermeasure makes attacks impossible!

Worst Case Attack Strategies

All attacks aim to exploit the signal and try and reduce the noise.

Thus the strongest attacks (aka most trace efficient) attacks will

- exploit as much of the available signal (by guessing large chunks of the key, and by profiling)
- exploit as many targets as computationally feasible (by profiling)
- utilise several traces (to reduce the per-trace-noise and to increase the information about the key).

There are different ways in which to combine information from multiple targets: all of them require profiling. The most efficient way (as far as we know) represents an algorithm as a factor graph and uses message passing to update the belief about the key bytes based on the leakage observations.

Multi Target Attacks (1)

Multi target: using more than a single target during an attack. “Pedestrian approach” aka “template based DPA”: we work with a single target but use other targets in the process.

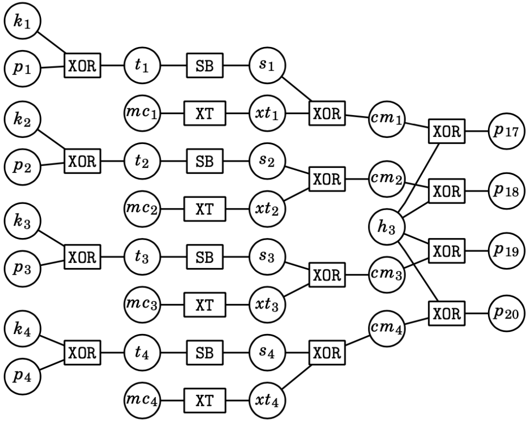
E.g. a masking countermeasure is based on pre-processing masked tables. We first recover the mask from the preprocessing and then do a single target attack on the S-box, [23].

E.g. some shuffling is implemented and we use profiling to extract information from multiple targets to reverse engineer the permutation. Then we run a single target attack on the S-box, [23, 2, 6].

These type of multi target attacks are often called “horizontal attacks” in the literature.

MULTI TARGET ATTACKS (2)

Or: represent an algorithm as a factor graph and using some message passing on the graph to extract updated beliefs about the key bytes.



Factor graph for one column of Mixcolumns.

We can produce factor graphs for a round or many rounds.

We can exclude nodes which don't leak strongly.

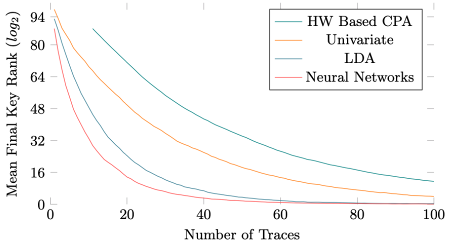
No explicit key guessing is necessary. We can exploit every leaking intermediate value. Can potentially include information about shuffling. Does link to deep learning.

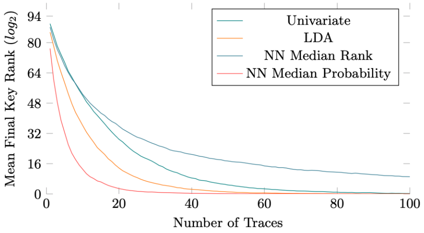
Worst Case Attack Strategies: Gap

There is a big gap between profiled and non-profiled single target attacks (left figure).

There is a big gap between single target and multiple target attacks (right figure).

Below are figures illustrating these gaps on an AES Furious implementation, running on an M0.





Masking countermeasures

Masking amounts to secret sharing all intermediates that depend on key information: i.e. the state as well as the key state. Each intermediate is represented via multiple shares: straightforward exploitation is hence impossible. However the joint leakage of shares reveals the secret!

Masking Security Statements

1. Attacks using less than a defined number of intermediates are provably prevented.
2. The number of leakage observations for successful attacks grows exponentially in the number of shares.

The first statement relates to proofs in a probing model: over the years there were a number of refinements and extensions to the “types of probes” leading to more composable “gadgets”.
The second statement implies that attacks that fully recombine the shares become exponentially harder.

Masking countermeasures: single target attack strategy

Standard single-target differential attack techniques require pre-processing of traces if masking is used.

Serial leakage: the “mean-free product combining ” of k trace points is the optimal function if the device leaks the HW.

Parallel leakage: we “shift” the leakage into the first moment by raising trace points to the power k.

Both techniques imply an exponential increase in the noise variance for the resulting traces. In the serial leakage case, the resulting traces are also considerable longer!

Roadmap

Masking countermeasures: single target - multiple trace

Chari et al. [7]: an adversary given observations for k shares, can distinguish if the product distributions (mean free observations, Gaussian assumption, single bit leakage) corresponding to the unshared value, given $(2\sigma^2)^k$ samples. ("single bit DPA style scenario")

They also provide a bound for **any attack strategy**, to succeed with probability α for a single bit leakage model an adversary needs in the order of $\sigma^{k+4\log\alpha/\log\sigma}$ traces. But no computation on the shares is considered.

Thus for masking to work, there needs to be enough noise: e.g. if $\sigma < 1$ no noise amplification can take place.

Masking countermeasures: multi target attacks

Prouff and Rivain [21] eventually provide a more comprehensive and general analysis. Their results holds in the "noisy leakage model": they show that the statistical distance between $\Pr[X = x]$ and $\Pr[X = x|L(x)]$ (and L is a noisy observation of x with noise σ) is bounded (i.e. there is a "bounded bias", $L(x)$ does not reveal everything about x).

In particular they show that the mutual information between (X, K) and $\{L(x_i)\}$ is upper bounded by cst/σ^k (cst is some device dependent constant). Assuming perfect independence of shares and computations on them, and leak free refreshing, and using a single trace.

Thus if $\sigma \leq 1$ then the mutual information does not decline with an increase of the number of shares.

Beware that there is no such thing as a "single" σ, SNR, or MI that characterises a device. Each devices has multiple of these. In software they can be manipulated by dispatching instructions in a certain order. The setup changes them too.

Masking countermeasures: multi target attacks

The result of Prouff and Rivain [21] was a massive step towards giving hope that masking would even withstand sophisticated adversaries. But what is cst?

[21] assumes that a constant exists for each target (i.e. a k-tuple of shares), and that each intermediate is independent (i.e. remasking and the remasking is leak free). The constants then "sum up" to an overall "constant" (see Theorem 4 of their paper): $Ncst/\sigma^k$.

The sum goes over all the intermediates, and the number of partial products in fact increases quadratically with the number of shares, thus N includes a quadratic factor.

Therefore the numerator gets much much larger for a multi target attack: unsurprisingly, because more shares mean more leakage!

Masking countermeasures: multi trace and multi target attacks

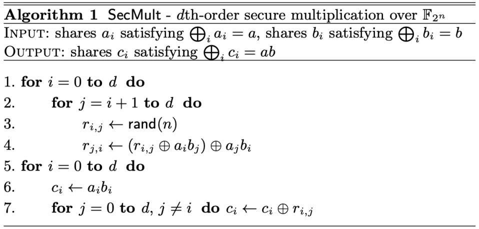
For low masking orders and low noise, a multi target approach becomes a serious threat: [2] demonstrate this clearly; the noise amplification is partially offset by a (smaller) amplification of the signal.

Naturally extending multi target attacks to utilise multiple traces makes them even more efficient (again shown in [2]): $Ncst/\sqrt{\sigma/nt}^k$. The effect of more sophisticated multi target approaches using belief propagation was studied heuristically in [12] and empirically in [11].

So the bound on the MI, which inversely translates to the number of required traces (=effort) for an adversary changes from:

single target - multiple trace	multiple target - single trace	multiple target - multiple trace
$\frac{cst}{\sigma^k}$	$\frac{N \cdot cst}{\sigma^k}$	$\frac{N \cdot cst}{(\frac{\sigma}{nt})^k}$

Masking: SecMult (aka ISW)



Algorithm 1 SecMult - dth-order secure multiplication over $\mathbb{F}_{2^n}$

INPUT: shares a_i satisfying $\bigoplus_i a_i = a$, shares b_i satisfying $\bigoplus_i b_i = b$

OUTPUT: shares c_i satisfying $\bigoplus_i c_i = ab$

```
1. for $i = 0$ to $d$ do
2.     for $j = i + 1$ to $d$ do
3.         $r_{i,j} \leftarrow \mathsf{rand}(n)$
4.         $r_{j,i} \leftarrow (r_{i,j} \oplus a_i b_j) \oplus a_j b_i$
5. for $i = 0$ to $d$ do
6.     $c_i \leftarrow a_i b_i$
7.     for $j = 0$ to $d$, $j \neq i$ do $c_i \leftarrow c_i \oplus r_{i,j}$
```

The SecMult gadget (Alg. 1):
$(d+1)^2$ multiplications,
$2d(d+1)$ additions, and
$d(d+1)/2$ fresh random values.

With SecMult a finite field inversion costs: $8d^2 + 12d$ additions, $4d^2 + 8d + 4$ multiplications, $3d + 3$ squarings, $2d^2 + 4d$ randomness and $0.5d^2 + 3.5d + 3$ bytes of memory.

Sequential execution: enables a multi target attacks. [2] use leakage from touching each share multiple times ("horizontal" attack, it increases the *nt* from the previous slide).

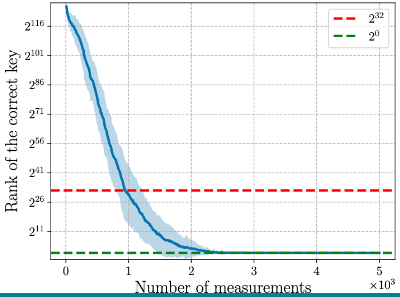
Using factor graphs [5] demonstrate that in a high SNR regime the "exponential" increase is very slow. **For 2, 3 shares less than 10 traces are always sufficient.**

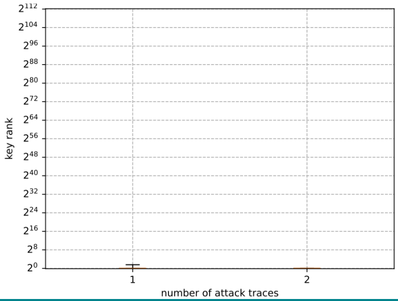
Worst Case Adversary: Gap

ANSSI released open source implementations: e.g. a very simple masked AES that also employs some simple hiding strategy. Similar to implementations analysed before [22, 23].

They provide an analysis based on TVLA (which we will discuss later): it is found to be secure against TVLA with 100k traces.

Bronchain et al. break this implementation easily: without belief propagation using 2k traces, with belief propagation using a single trace.





Masking in Hardware

Existing (i.e. deployed by industry) logic styles try to ensure uniform power consumption by a 2-share representation.

Masked logic was considered early on, and is now seeing a revival due to a better understanding of of how to deal with glitches (not just per design but also via improved proof techniques).

But the cost of the resulting masked gates is extraordinary! And more than 2/3 shares are required as security against DPA (aka single target non profiled attacks) is not sufficient for high protection levels. And tooling needs to integrate with existing design flows.

The security industry is very successfully producing devices with very high protection levels, combining logic styles and software countermeasures. Is there an industrial need for a standardised masking scheme for hardware?

Standardised Masking?

Several important questions were raised at the workshop in 2020 relating to the idea of standardising a masking scheme for AES.

Standardisation of masking seems difficult: you need to ensure a low enough SNR throughout, the number of shares must depend on your SNR (which you might not know a priori), tooling needs to fit design flows, testing needs to be deviced, ... ?

Verification and/or testing requirements are imperative: the methodology included in the standard must be sound and does not under(/over)estimate the security of the target implementation. If standardised, it could have a strong impact on certification procedures.

Standardising just the masking scheme: it’s a bit like saying “here is a blue print for a round function, but we don’t know how much key material is required, we don’t know how many rounds you need, and we can’t give you any test vectors’.

Roadmap

How to best measure the success of an attack?

A simple way is to “fix a distinguisher” and then compare the outcomes.

For univariate differential attacks (single target, multiple trace), the correlation coefficient is very well suited (normalised, it scales easily with noise, well understood behaviour, can use non-linear models).

For multivariate attacks the correlation cannot be used anymore: alternatives are the MI and the SNR.

Empirical estimation of the MI can be problematic (no convergence guarantee for pdf based estimation), but lot’s of connections to masking theory.

Shortcuts vs Attacks

It is possible to calculate various distinguisher scores based on characterising a device (i.e. a priori estimation of the leakage function(s)).

Together with knowledge of the behaviour of the target it is then possible to compute the likely outcome of a differential attack.

Knowing the distribution of the distinguisher output for the correct and incorrect keys enables then (using a bit of statistical trickery) to predict how many side channel observations are needed in practice.

These lead to so-called **shortcut formulas**. A nice overview with some implementation examples can be found on the REASSURE website `reassure.eu` (under deliverables and tools).

Another type of shortcut would be leakage detection.

Roadmap

Attack success (1)

It should be clear that different types of practical attacks perform differently w.r.t. the number of observations needed to “succeed”, memory requirements, time, etc.

What is “success”? In the context of embedded systems, taking power and EM as side channels, we nearly always care about **key recovery** attacks.

Metric: Success Rate (SR)

We call an attack strategy successful if the highest ranked key is indeed the secret key. The key recovery **success rate** (SR_N) is the probability with which the attack strategy is successful given N side channel observations corresponding to different inputs.

Attack success (2)

A sensible criticism of the SR is that it ignores the capability of adversaries to use partial information that becomes available after an attack: perhaps the secret key does not have the "highest score" but it may be among the high ranked key candidates.

Metric: *o*-th order SR

We call an attack strategy successful if the secret key is among the o top ranked keys. The o-th order key recovery **success rate** ($o - SR_N$) is the probability that the attack strategy is successful given N side channel observations corresponding to different inputs.

Metric: Key Rank (KR)

The key rank (KR) is the actual position of the secret key in the sorted list of all key candidates. We call an attack using N observations successful if KR_N is below some set bound.

(Definition is adapted from [17].)

Ranking Entropy

Thus the key rank of a single experiment is not meaningful in itself. The key rank naturally links to the guessing entropy: $GE = \mathbb{E}(KR)$.

But interesting key ranks can be large, thus to compare attacks meaningfully, we want to compare "log ranks" (e.g. suppose among 1024 attacks, 1023 put the key at place 1 and one single attack returns 2^{32}, the mean rank would then be 2^{22}).

Metric: Ranking Entropy

The ranking entropy RE is a faithful representative of the remaining effort of a side channel adversary.

$$RE = \mathbb{E}(\log KR).$$

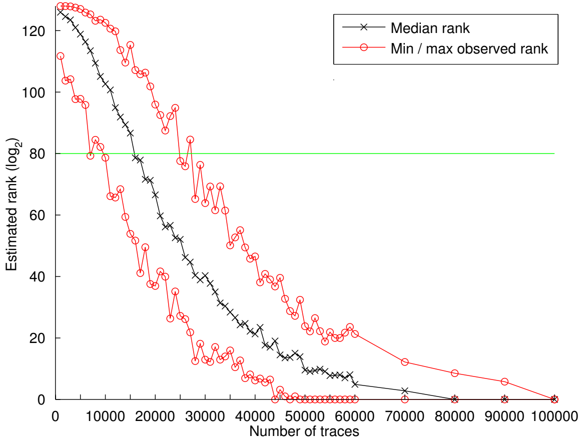
It is imperative to realise that log and $\mathbb{E}$ do not commute. An alternative to taking mean logs is to compute the median.

Ranking Entropy: practical considerations

Ranks from a hardware AES.

Observe the gap between best and worst ranks.

2^{80} was reported as the cutoff by JHAS for failing a device.



Ranking Entropy: practical considerations

[17] show that at least 50 experiments are required for the ranking entropy to stabilise; about 100 experiments give a good approximation of the ranking entropy.

They also show that ranks int the 2^{40} and 2^{80} range have the highest variance, and their variance is very large (they observed up to ± 20 bits). The variance seems most dependent on the rank itself, not on the SNR.

The fastest rank implementation (to the best of my knowledge) is by Mather:

bits	sec	bits	sec
128	0.005	1024	1
256	0.05	2056	10
512	0.5	4096	100

Roadmap

Key Enumeration

One can turn key ranking algorithms into key enumeration algorithms: i.e. we order and test keys, in order to find the secret key: powerful way to improve attack outcomes.

[14] show that enumerating 2^{48} keys is easy within 30hrs on a University cluster. (We kept on playing and our current estimate is that we could do 2^{56}).

From my perspective: any attack that leads to less than 2^{48} enumeration effort should count as a complete break (as we could do this years ago on University resources).

Realistic attacks (from a nation state adversary) could probably do 2^{60}. Thus the 2^{80} bound that was allegedly set by JHAS seems sound.

Key Enumeration: Post Quantum

[18] show that it is possible to **non-trivially combine Grover and key enumeration**: i.e. we can leverage the square root speed up of Grover and simultaneously use the sorted list produced from side channel scores!

Quantumly key enumeration would pass on “batches of keys” to an implementation of Grover that would require a quantumly AES oracle.

A cut-off at 2^{80} would not be safe post quantum: with Grover this reduces to 2^{40}.

The more conservative cut-off of 2^{100}, which is also sometimes attributed to some European countries, would also not be safe post quantum.

Roadmap

Detection vs Attacks

Any successful attack detects leakage. But it detects “specific” leakage, i.e. leakage that is due to the target(s).

Thus attempting to detecting **all** leaks via attacks would imply that every intermediate and/or any combination would need to be attacked: this is way too time consuming!

Alternative: “non-specific” leakage detection. We attempt to find all/any key dependent behaviour in the leakage traces.

CRI (now Rambus) proposed to find any data dependency via the so-called “Test Vector Leakage Assessment” (TVLA) in the form of “fixed vs random” tests.

TVLA

Fixed vs Random: in relation to inputs. Thus you detect lot's of data dependency, but you couldn't find the key schedule.

Engineering Guidelines: very helpful to minimise creating bias during the assessment.

Univariate Statistics: based on univariate *t*-statistic . . . sadly without any reference to how to account for the fact that there are many correlated leakage points, thus leading to a serious "multiple comparison" problem.

Moment Specific: You can only detect leaks that sit in a central moment of a distribution.

False Positives: the threshold for false positives is set very low to minimise false positives, probably to account for the multiple comparison issue. But this implies, unless sufficiently many traces are used, a low probability to detect leaks.

Univariate "Single Variable" Leakage Detection

TVLA is a great (i.e. quick) way to detect leaks that you suspect to be there, or to show that your masking implementation is not trivially insecure [19].

But it is ill suited as a comprehensive form to assess leakage, in particular in a black box setting:

- It is a univariate test that can only detect differences in the central moment of a distribution.
- Varying the plaintext creates unnecessarily many false positives.
- Complete lack of guidance to correctly configure the test: the α level is provided, but without knowing the effect size, it is impossible to select the number of traces to get a good $1-\beta$.
- No acknowledgment of the multiple comparison problem and how to account for it.
- Weak link with non-profiled differential attack: typically a DPA attack using correlation and a HW model will require much less traces to reveal a leak.

Univariate "Single Variable" Leakage Detection

ISO 17825 (which seems to be the basis for side channel evaluation in the context of symmetric crypto) within FIPS 140-3, instantiates TVLA, and gets tripped up by its lack of clarity in terms of configuration.

[24] show how ISO 17825 makes things worse and make suggestions for a saner form of configuring the statistical parameters.

Industry is equally concerned about the use of any TVLA style test in the context of a black box evaluation [10].

But no "singe variable" test can answer the question "does x contribute to leakage" (it can only tackle the much narrower question "does x alone leak").

Univariate “Single Variable” Leakage Detection: Public Key Setting

The situation is even worse for a potential use in public key implementations due to the increase in trace length.

There is a multi-variate counterpart (Hotellings) but it assumes independence (which is obviously violated); [4] investigate this. MI could be an alternative to detect leaks that are not in central moments [19].

This can also be seen in the PQ setting: the vulnerability of the FO transform against side-channel attacks is known and the difficulty to protect it.

While TVLA or other similar leakage detection techniques will mostly likely deem any high order masked implementation secure, the leakage present in the FO transform and magnified by chosen inputs is highly multivariate and yet very easy to exploit by an attacker.

Roadmap

Sound Leakage Assessment Requires a Priori Knowledge

White-box setting: if a device is fully characterised, and the implementation is known then perhaps a sound statistical testing regime could be established.

Leakage simulators might offer an “economic” middle ground for evaluations that need to be cheap: rather than open sourcing a processor, an accurate leakage model could be open sourced, and used to test software implementations.

ELMO (and GILES) [20] provides quite accurate power leakage models of the processor core of an M0 and and M3 (as well as an M4 but based on EM traces). But because the memory runs asynchronously their leakage model of the memory is not accurate (ROSITA does not do any better).

But devices of the same architecture can have subtly different leaks (as ELMO shows by providing multiple models for an M3), and which is well demonstrated in [16].

Leakage Assessment Benefits from Worst Case Adversaries

In [1] we argue (based on 3 years of work with evaluators, hardware developers, integrators and certifiers) that basing evaluations on “worst case adversaries” is actually a very useful thing.

Worst case adversary: can enumerate, can use multiple targets (if available), multiple traces, knows implementation details, controls inputs/key, controls randomness, thus either can profile or has access to profiling data.

Worst case adversaries represent long term security goals: evaluators don’t have the time/budget to do a lot of reverse engineering, but a determined adversary will (potentially using multiple devices, and further attack strategies including faults) eventually be able to create a reasonably good leakage model.

Thus giving an evaluator the information so they can do a worst case attack is cost efficient and it enables to judge the long term security of a device/implementation. They can also assess the “gap” via a “backwards evaluation”.

Roadmap

Security Estimations using Side Channel Information

Integrating side-channel information into security estimates could be one method for considering SCA into real-world applications [14, 8, 13].

This can be used to accommodate the many use-cases where one anticipates, expects, or even allows some side-channel information loss/leakage on a device in the wild.

- Similar to how DPA-resistant cores[1] ‘guarantee’ no leakage up to 100m traces using TVLA.
- One could potentially integrate bounds into SCA-secure implementations, allotting some ‘wiggle-room’ for adversarial gains in SCA.

How do we integrate this into post-quantum cryptography security estimates?

[1] `https://www.rambus.com/security/dpa-countermeasures/dpa-resistant-core/`

LWE with Side Channel Information [8][2,3]

In the cryptanalysis of lattice-based cryptography, we might be in two camps:

(1) Lattice reduction
via primal attack.
Cost: β block size of BKZ.
Models: say, 128 bits of ‘work’ needed for key recovery.

(2) Side-channel attack
via, let’s say, DPA.
Cost: X number of traces.
Models: say, 100 power traces needed for key recovery.

Consider an example where we might be in the middle:

- We may only have access to, say, 20 power traces, and we can’t run a full key recovery attack.
- We can run a hybrid attack instead, to essentially decrease bit security.

[2] https://www.youtube.com/watch?v=wCaLcbWnwDI&ab_channel=TheIACR
[3] https://www.youtube.com/watch?v=RRMtKtbx1Yo&ab_channel=SimonsInstitute

LWE with Side Channel Information

Side-channel information (or hints) can be integrated into lattice reduction in four ways:

(1) Perfect hints:
$\langle \mathbf{s}, \mathbf{v} \rangle = \ell$.
(Learning inner product).

(2) Modular hints:
$\langle \mathbf{s}, \mathbf{v} \rangle = \ell \mod k$.
(Same as (1) but mod integer).

(3) Approx. hints:
$\langle \mathbf{s}, \mathbf{v} \rangle \approx \ell$.
(Similar to (1) but with an error).

(4) Short vector hints:
$\mathbf{v} \in \Lambda$.
(Info on the lattice, not the secret).

The hints range from (1) helpful, to (4) less helpful.

One ‘embeds’ hints into an instance of (D)BDD to decrease its hardness.

Examples of Hints

A toy example of this in practice.

Say our secret $\mathbf{s}_i \in \{-5, \dots, 5\}$, from power analysis learn $\mathrm{HW}(\mathbf{s}_0) = 2$.

Thus we know $\mathbf{s}_i \in \{3, 5\}$, this can be encoded as either:

- A modular hint, where $\langle \mathbf{s}, (1, 0, \dots, 0) \rangle = 1 \mod 2$.
- Or an approximate hint, where $\langle \mathbf{s}, (1, 0, \dots, 0) \rangle \approx 4$, with variance 1.

Examples of Hints

An example with a real target. This framework was used from data of a previous attack [3] on FrodoKEM. The paper consisted of two single-trace attacks, one of which was too weak for key recovery.

		NIST1	NIST2	CCS1	CCS2
Attack without hints	(bikz)	487	708	239	448
Attack with hints	(bikz)	**337**	**471**	**190**	**297**
Attack with hints & guesses	(bikz)	**298**	**403**	**126**	**110**
Number of guesses		100	250	200	300
Success probability		0.86	0.64	0.87	0.77

Table: Cost of the attacks with/without hints and with/without guesses.

One might say that bikz values around 100 are computable in practice.

Examples of Hints

The framework also uses examples of two other use-cases.

Decryption failure results [9] are reproducible using this framework, observing similar degradation in security given an increase in the number of decryption failures.

Also, some real-world schemes (LAC, Round5, and NTRU) used fix-weight ternary secrets, i.e., $\mathbf{s}_i \in \{-1, 1\}$, which can be embedded as perfect hints.
This reduces the block size by either 1, 2, or 3.

INFORMATION-SET DECODING WITH HINTS

A similar result [13] was also found for code-based cryptography, namely incorporating small leakages into information-set decoding (ISD) algorithms.

These hints are actually used when solving the (n, k, t)-syndrome decoding problem (SDP), for length n, dimension k, and error weight t.

The work analyses Classic McEliece, BIKE, and HQC, giving the number of hints required to reduce the claimed security level.

e.g., for mceliece348864, the work factor is reduced below 2^{128} for 175 known message entries, 9 known error locations, 650 known error-free positions, or known Hamming weights of 29 sub-blocks of roughly equal size.

How Can Hints Affect Evaluations?

Can tools like this be used in evaluations where SCA is an important consideration?

e.g. remaining x-bits secure, after giving away y-information, (via attack type z).

Might we have alternative parameter sets for schemes assuming this y-information loss?

Roadmap

FIPS 140-3 vs CC (JHAS)

FIPS 140-3	**CC (JHAS)**
no profiling	mandatory profiling (try DL)
sym. crypt: detect and stop	attack potential
asym. crypt: SPA/DPA	attack vectors
levels by num traces (detection)	part white box

IMHO they only care about “physical leakage”. The biggest concern is cost.

CC is close to the worst case adversary approach. BSI (Germany) now rewrites their guidelines (widely used for CC evaluations) to take the worst-case adversary approach better into account.

Detection in 17825 is flawed, without some form of “white box” is cannot work. Number of traces for non-specific detection not immediately linked with number of traces for attacks (even for standard DPA).

Cost Effective Evaluations: Procedural

Procedural point of view: make certification regimes more modular so certificates can be “reused”.

SESIP: “Security Evaluation Standard for IoT Platforms”, pushed by GlobalPlatform, methodology to integrate certificates from different regimes under a common framework (aka plug and play or a Lego approach). Works under the CC framework (protection profiles).

Interesting recent example: company X who produces processor IP cores gets an evaluation of EAL6 (under CC) for their “processor”. But they don’t manufacture processors. And if you read into the certificate then it becomes clear that if that IP was integrated, then the integration would need to be evaluated and assuming all works out this would lead to EAL4.

Cost Effective Evaluations: Technological

Technological point of view: make technical changes to increase the speed of evaluations and increase the confidence in their outcomes.

- worst case adversary: fully white box for the evaluator plus full control over the device; also gives long term security guarantees
- “standardised” leakage emulators with standardised statistical tests for evaluating software: not fully white box w.r.t. the hardware, enables to give full control, could perhaps be automated
- “standardised” HW architectures that support countermeasures and provide a low enough SNR, together with tooling that creates “provably” secure implementations

Conclusions

- Any attack strategy manipulates the SNR. Any countermeasures manipulates the SNR.
- Multi-Trace Multi-Target attacks (aka SASCA, aka horizontal attacks) make most of the signal, whilst averaging out the noise.
- Multi-Trace Multi-Target attacks are our best approximation of the "worst case adversary".
- We have no idea if any specific crypto architecture is truly "better" with regards to leakage.
- But crypto that is efficient and enables a single type of masking is seen as preferable today.
- To enable more secure crypto implementations we need: less leaky hardware and better tooling.

Conclusions

You asked “what could NIST do?”

Consider a competition for:

- a “security friendly” open source processor
- a “side channel countermeasure enabling instruction set” for an existing open source processor
- a “leakage assessment framework” (i.e. concept and tooling) for some existing (open source) processors
- a “generic implementation framework” (i.e. countermeasure instantiation and assessment) for e.g. symmetric encryption on a 32-bit (or 8-bit) platform

Security against side channel adversaries is enabled (and limited) by the hardware.

We urgently need open source hardware that enables security: low SNR, instructions to support countermeausures, full characterisation/leakage emulators for early assessment.

Reading Material I

[1] M. Azouaoui, D. Bellizia, I. Buhan, N. Debande, S. Duval, C. Giraud, É. Jaulmes, F. Koeune, E. Oswald, F. Standaert, and C. Whitnall.
A systematic appraisal of side channel evaluation strategies.
In SSR 2020.

[2] A. Battistello, J. Coron, E. Prouff, and R. Zeitoun.
Horizontal side-channel attacks and countermeasures on the ISW masking scheme.
In CHES 2016.

[3] J. W. Bos, S. Friedberger, M. Martinoli, E. Oswald, and M. Stam.
Assessing the feasibility of single trace power analysis of frodo.
In SAC 2018.

[4] O. Bronchain, T. Schneider, and F. Standaert.
Multi-tuple leakage detection and the dependent signal issue.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019.

Reading Material II

[5] O. Bronchain and F. Standaert.
Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2021.

[6] O. Bronchain and F. Standaert.
Side-channel countermeasures' dissection and the limits of closed source security evaluations.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2020.

[7] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi.
Towards sound approaches to counteract power-analysis attacks.
In CRYPTO 99,.

READING MATERIAL III

[8] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi.
Lwe with side information: attacks and concrete security estimation.
In CRYPTO '20.

[9] J.-P. D'Anvers, F. Vercauteren, and I. Verbauwhede.
On the impact of decryption failures on the security of lwe/lwr based schemes.

[10] F. Deboyser, M. Hinkelmann, and V. Vernuil.
A journey towards side-channel attack resistances.
Invited Talk at ICMC21.

[11] J. Green, A. Roy, and E. Oswald.
A systematic study of the impact of graphical models on inference-based attacks on AES.
In CARDIS 2018.

Reading Material IV

[12] Q. Guo, V. Grosso, F. Standaert, and O. Bronchain.
Modeling soft analytical side-channel attacks from a coding theory viewpoint.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2020.

[13] A.-L. Horlemann, S. Puchinger, J. Renner, T. Schamberger, and A. Wachter-Zeh.
Information-set decoding with hints.
Cryptology ePrint Archive, Report 2021/279, 2021.

[14] J. Longo, D. P. Martin, L. Mather, E. Oswald, B. Sach, and M. Stam.
How low can you go? using side-channel data to enhance brute-force key recovery.
IACR Cryptol. ePrint Arch., 2016.

[15] S. Mangard, E. Oswald, and T. Popp.
Power analysis attacks: Revealing the secrets of smart cards .
Springer, 2007.

Reading Material V

[16] B. Marshall, D. Page, and J. Webb.
MIRACLE: micro-architectural leakage evaluation.
IACR Cryptol. ePrint Arch., 2021.

[17] D. P. Martin, L. Mather, E. Oswald, and M. Stam.
Characterisation and estimation of the key rank distribution in the context of side channel evaluations.
In ASIACRYPT 2016.

[18] D. P. Martin, A. Montanaro, E. Oswald, and D. J. Shepherd.
Quantum key search with side channel advice.
In SAC 2017.

Reading Material VI

[19] 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 ASIACRYPT 2013.

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

[21] E. Prouff and M. Rivain.
Masking against side-channel attacks: A formal security proof.
In EUROCRYPT 2013,.

Reading Material VII

[22] S. Tillich and C. Herbst.
Attacking state-of-the-art software countermeasures-a case study for AES.
In CHES 2008.

[23] M. Tunstall, C. Whitnall, and E. Oswald.
Masking tables - an underestimated security risk.
In FSE 2013.

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