Side Channels: Attacks, Defences, and Evaluation Schemes
Part 1 — Attacks and Defences

Elisabeth Oswald, James Howe

University of Klagenfurt and University of Bristol

Roadmap

Side Channels / Information Leakage

Informal: we use the terms side channel/ information leakage interchangeably and refer to any form of information that becomes available about something that was meant to be kept secret.

E.g. timing information (via cache or other), power consumption, EM, acoustics, cache access, ...

Is typically unintended (otherwise it's a covert channel).

Reveals "something" about secret inputs or keys, or helps to distinguish between messages, etc. Information leakage tends to be so strong that attacks typically directly focus on either message and more often key recovery (total break of a system).

Focus of this lecture series: power and EM side channels

Sometimes referred to as *physical* side channels or *hardware* side channels.

Better thought of as side channels that are particularly relevant for embedded devices.

Thus adversaries typically:

- Can obtain multiple devices of the type of device that they wish to attack
- Are in close physical proximity: to the target device but also duplicate devices (before, during and after an attack)
- Have at least "grey box knowledge": architectural information is available
- May be able to completely reverse engineer: micro-architectural information is available

Why do devices leak via their power consumption/EM emanation?

All devices are realised via a “large bunch” of “cleverly connected” logic gates.

Efficiency is key: logic styles aim to minimize the “unnecessary” moving around of electrical charge.

CMOS: charge only moves (i.e. a devices consumes power) if the state of a gate changes; EM field changes with moving of charge.

Other logic styles aim to “balance” the moving of electrical charge thereby reducing the data dependency.

Power/EM Leakage cannot be fully mitigated at the source

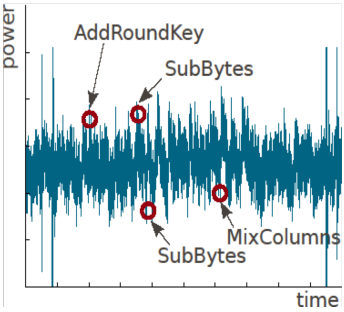
Producing perfectly balanced gates/wires is impossible, thus even with less leaky logic styles, there is some dependence between data and observable power/EM.

Types of Side Channel Scenarios

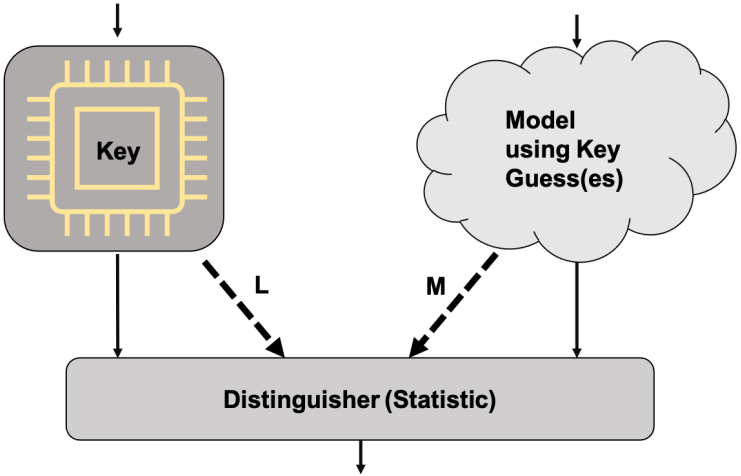
A single trace from one input offers multiple (exploitable) targets. Adversaries can often acquire more than one trace.

Attacks can take advantage of

- single input / single target (little use for key recovery)
- single input / multiple targets
- multiple inputs / single target
- multiple inputs / multiple targets



Principle of (Differential) Side Channel Attacks



(Specialised to the setting where we are trying to recover a secret key, adapted from [13].)

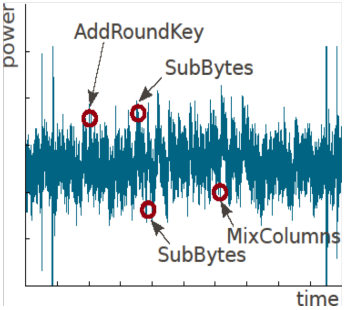
Side Channel Attack Strategies

With extended access to a device (or a duplicate device), **sophisticated models** can be built. Such **profiling** is a precursor to exploiting multiple targets simultaneously.

Multiple traces for different inputs: **"differential attacks"**.

Practical attack strategies can be broadly divided into:

- **profiling:** single input **/ multiple target**
- differential inputs (with or without profiling)/ single target
- **profiling:** differential inputs **/ multiple target**

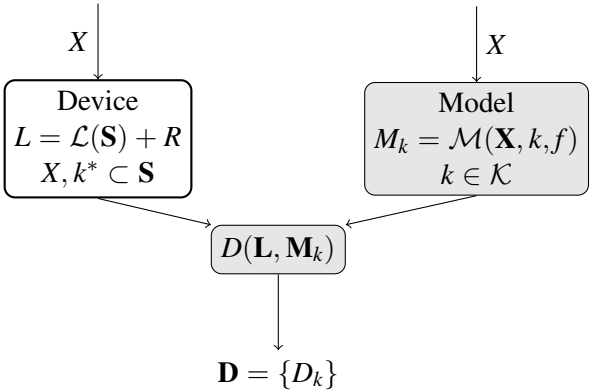


Differential Attacks: slightly more formal

The adversary captures $L + R$ and produces input and key guess dependent predictions M by selecting a target f.

The state $\mathbf{S}$ varies over the execution of the algorithm.

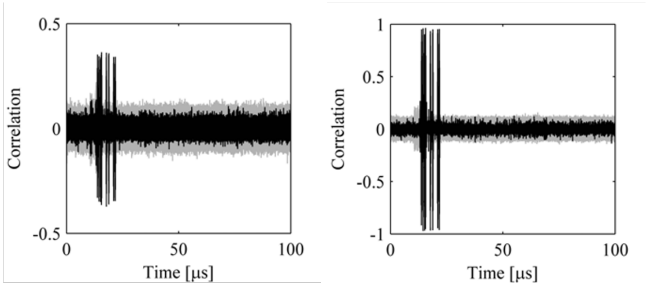
Using a distinguisher (univariate or multivariate), the adversary decides, which prediction "fits" better with the observations, assigning a "score" to each key guess.



The key guess with the highest score is likely to correspond to the true key value k^*.

Example: Differential Single Target Attack

Two examples of outcomes of a differential attack on a single target using correlation as a distinguisher: grey traces correspond to outcomes for incorrect key guesses; the black trace corresponds to the correct key guess.



(Left: very simplistic single-bit leakage model, Right: device appropriate leakage model).

Roadmap

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).

Signal to Noise Ratio (SNR)

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 $Var(R)$:

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

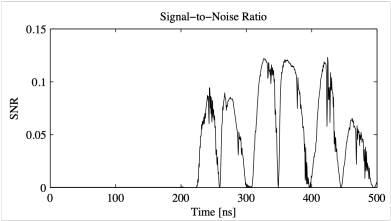
(As in [13].) Obviously: the higher the signal in relation to the noise, the better attacks can be assuming we can **exploit** the signal.

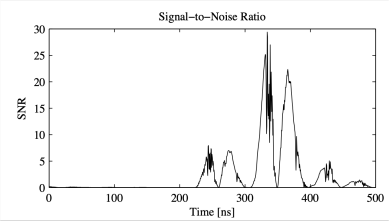
Example: SNR of Previous Attacks

Any attack strategy is about maximising the exploitation of the available signal and if possible simultaneously decreasing the noise.

In the previous example we saw a comparison between a strategy that exploits a single bit vs a strategy that exploits a byte from the signal.

Below are two plots that show the SNR for each scenario (computed over a larger time frame than what we used in the attack plots).





(Left: signal corresponds to a single bit, right: signal corresponds to a byte)

Example: SNR of Previous Attacks

Attacks require fewer traces if

- more of the available signal can be exploited
- the per-trace-noise can be reduced.

Thus the best attacks will

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

Roadmap

Decreasing the noise: differential attacks

Differential attacks: acquire multiple traces for each model input.

Can also be thought of as computing the distinguisher for the resp. average trace for each input.

If a trace point is $\mathcal{N}(\mu, \sigma^2)$ then its average (over n traces of the same input) is $\mathcal{N}(\mu, \frac{\sigma^2}{n})$.

E.g. Correlation:

$$R(M_k, L) = \frac{\sum_i (m_{k,i} - \bar{m_k})(l_i - \bar{l})}{\sqrt{\sum_i (m_{k,i} - \bar{m_{k,i}})^2 \sum_i (l_i - \bar{l})^2}} = \frac{\sum_x \sum_{m_k = x} (m_k - \bar{m_k})(l_x - \bar{l_x})}{\sqrt{\sum_x \sum_{m_k = x} (m_k - \bar{m_k})^2 \sum_x \sum_{m_k = x} (l_x - \bar{l})^2}}$$

Thus differential attacks improve the SNR by decreasing the noise

SNR AND CORRELATION AS DISTINGUISHER

It is easy to rewrite the correlation coefficient in terms of the SNR:

$$\rho(M, L) = \frac{\rho(M, \mathcal{L})}{1 + \frac{1}{SNR}}.$$

With some more simplifications (we called this "the rule of thumb" in [13]) and assuming a small signal, one can show that the number of side channel observation that is needed to succeed with near certainty scales with the (inverse) SNR:

$$n_{obs} \approx \frac{1}{SNR}.$$

Thus: SNR affects the number of needed observations in a predictable manner.

Clearly: the setup is incredibly important; device evaluations must be based on excellent and somewhat standardised setups; comparing attacks by comparing n across devices/setups is meaningless.

DIFFERENTIAL ATTACKS: CHOICE OF DISTINGUISHER

Optimal distinguishers: defined as those that maximise the probability of returning the correct key [12]. This definition was lifted from the signal detection literature where it is know as the "maximum a posteriori" (MAP) decoding rule; it leads to what we call template attacks [8].

Asymptotically (i.e. with large trace numbers), and assuming accurate models, distinguishers such as correlation and Bayes templates are equivalent in the sense that they will return the correct key with near certainty, [12, 14]. Non-asymptotically there exist (small) differences.

Are distinguishers then irrelevant? **No**, but they work in tandem with the available leakage model.

The bottom line: the statistical distinguisher must "fit" the model, and the model must fit the device, non-asymptotic differences are of some theoretical interest.

Roadmap

Increasing the signal: profiling

The data dependent component of leakage can depend on multiple bits/bytes/words depending on the architecture of the device/implementation.

Exploiting more of "what is there" helps adversaries because more of the available signal becomes exploitable.

Profiling: mainly reduced to "model coefficient estimation": the leakage model is fixed, and only the coefficients are estimated; e.g. we fix the number of templates and then estimate (μ, σ) for each model from the data [8], e.g. we choose the regression polynomial and estimate the coefficient for each term from the data [20].

Statistical Modelling: We consider **how** the model should look like and then estimate coefficients [16].

Model Quality

Predictive power: important for attacks, in particular if we want to demonstrate a "worst case adversary", we need to judge how "close" model predictions are to "new observations": R^2, cross validation, key rank

Explanatory power: important for simulations and proofs, we need to judge how much of the leakage our model can predict: $F-$test

Data complexity requirements: important for estimation accuracy; the amount of data implicates how complex a model can be.

Typically one trades off one property for the other: I know of no method that would lead to a model that can achieve very high explanatory power AND predictive power simultaneously AND does not require huge amounts of data.

Predictive Modelling

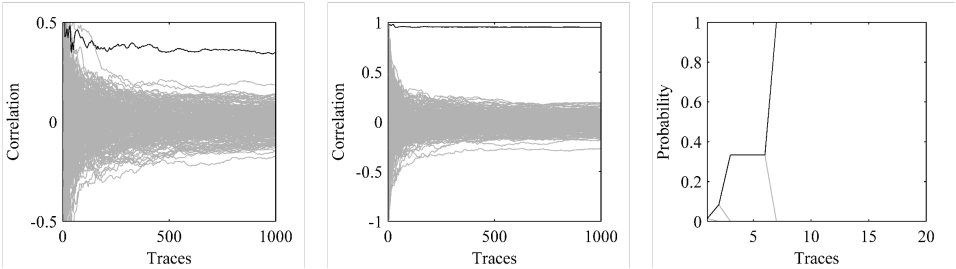
The goal is to derive proportional or direct models: we aim to be as close to the observed power consumption as possible.

With a fixed/limited amount of training data; full control over the training device (i.e. inputs, key, and any randomness). This is also called supervised learning.

Classical statistical techniques are based on directly estimating (mean vector, covariance matrix) (aka "templates") or regression (with or without covariance estimation); ML/DL techniques are also possible.

In a Common Criteria based security evaluation scheme, profiling (aka predictive modelling) is always attempted. In FIPS 140-3/ISO 17825 this is currently out of scope.

EXAMPLE: INCREASING THE EXPLOITABLE SIGNAL



Super simple example already given in the DPA book.
(Left: single bit attack, ca. 160 traces; Middle: HW based differential attack, ca. 20 traces; Right: template based differential attack, ca 6 traces)

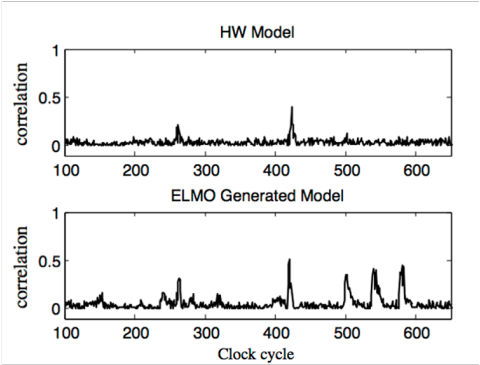
Example: Increasing the exploitable signal

Using a wrong or badly fitted model can give a false sense of security.

Top: HW model, many leaks remain unidentified

Bottom: Leakage models that were derived from identifying model equations and then estimating coefficients

Images taken from [16].



Working with an unknown/complex signal

But the choice of model also enables attacks when a characterisation of the signal is missing (perhaps in an initial attack).

Generic distinguisher: works using a **nominal** leakage model (e.g. identity mapping). Examples are MI or DOM.

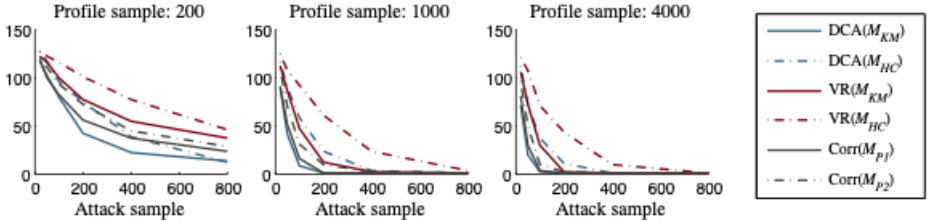
Require non-injective target functions or a "hack" (bit dropping trick) when applied to injective targets such as AES SubBytes.

Unsupervised learning methods: like "differential cluster analysis" [4] (can be seen as an extension of the variance ratio); as well as robust models as per [23]; work by applying clustering techniques to traces deriving **nominal models**.

Example: Working with an unknown/complex signal

From [23]: what if countermeasures, or setup inconsistencies change trace characteristics between profiling and attack?

Mitigation using classical statistics is possible (using unsupervised clustering, and PCA/LDA), and there are trade offs between profiling and attack:



DL for predictive Modelling

There are many papers now around that experiment with both MLPs and CNNs. There are a few clear results/pointers:

- MLPs train more quickly than CNNs. Their representation is also more memory efficient.
- CNNs can learn how to correct some "misalignment" because they have convolution layers. However, with decent pre-processing, any MLP can also cope with misaligned data.
- Finding the "best" set of hyperparameters for a device/setup is a trial and error process. No help is on the horizon.
- Papers find conflicting results and there is no way to conclude that DL approaches are strictly/always better than classical templating/linear regression with preprocessing.
- DL seems to be very promising to deal with multivariate leakage.

PROFILING ALSO ENABLES MULTI TARGET ATTACKS

They exploit leaks from several targets, either from a single or from multiple leakage traces.

Multiple target attacks are the “natural form” for single trace attacks: early examples include exploiting leakage from the key schedule (symmetric world), and patterns corresponding to multiplication/scalar multiplication (RSA, ECC).

Multiple target attacks are extremely powerful in exploiting the available signals, but require some form of profiling.

Two strategies: [15] require knowledge of time points, but no leakage model in their multi-target DPA style attacks.

[22] require full profile and propose **belief propagation** to accumulate partial knowledge about keys.

Roadmap

Three Mitigation Strategies

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

- Reduce the signal
- Increase the noise
- Limit the number of observations

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.

HIDING COUNTERMEASURES

We defined them in [13] as techniques that aim to make the leakage independent of the data. This can be done by either making the leakage as “uniform” as possible accross all data, or as “random” as possible accross all data.

Making leakage “uniform” requires changing the hardware: design logic gates that move “equal charges” (little to no signal).

Much research in the early 2000s on leakage resilient logic styles, mostly DRP (kind of a simple 2 share representation). Used in actual products. Fallen out of favour by the academic community.

Now “provably secure” gadgets (we’ll come back to these).

Roadmap

HIDING COUNTERMEASURES

Making leakage look random is the second option for hiding countermeasures.

Either by increasing the noise in hardware: parallel execution of the primitive, parallel execution of unrelated stuff (used in products), self timed logic/multiple clocks/clock domains, shuffling/dummy operations.

"Provable security": amounts to reasoning probabilistically how the expected noise reduction will be. E.g. if we can reason that the single target occurs with probability p in "the right spot" then the correlation scales with the factor p.

Multi trace/single target: assumes "alignment" of operations/data across observations — countermeasure "provably" changes the alignment, thereby lowering the ability of differential attacks to lower the noise.

But in the case of profiled attacks, the effect can often be negated, e.g. integration techniques, or using CNNs.

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 target is now represented via multiple shares, and each of the shares look random: straightforward exploitation is hence impossible. However the joint leakage of shares reveals the secret!

Masking Security

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.

Masking Security

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.

1: Often misunderstood as “masked implementations are completely secure” but they are not. This guarantee is typically based on a proof that considers a “high level” description of a gadget, with many independence assumptions that can be hard to fulfill in practice.

There is a lot of work currently being done to make such proofs for actual implementations, because of the many attacks that show that provably secure gadgets are leaking.

Masking Security

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.

2: Often misunderstood as “masked implementations require lots of traces to break” but they might not. The statement comes from a proof that assumes that an adversary must somehow produce the joint leakage of intermediates, which leads to a distribution that has a much higher variance.

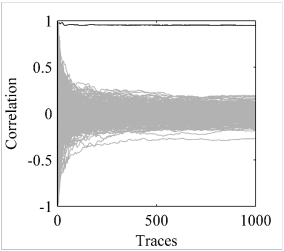
But few shares imply a small exponential factor.

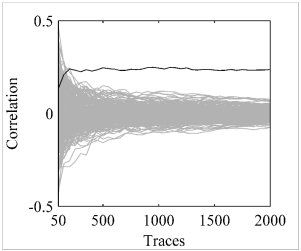
Profiling attacks enable the exploitation of each intermediate separately so the variance remains unchanged.

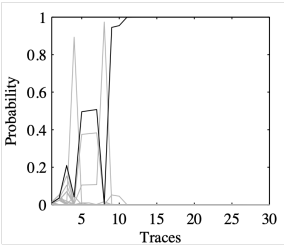
Example: Masking Security

On the same device, we show evolution plots for an unmasked implementation and an implementation representing intermediates with two shares.

The SNR is high on this platform, thus masking with two shares provides no practical security gain.







(Left: unmasked implementation, Middle: differential attack on preprocessed traces, Right: multi-target attack using classical templates)

Roadmap

Provably Secure Key Refreshing

Options: refreshing the representation of the key (e.g. via updating shares), updating the key, or **using a freshly derived key**, perhaps in a tweakable BC setting

Security models: any bounded function vs. **"secure against SPA vs DPA"** vs **leaks vs does not leak**

Stateful vs. stateless

"Leak resistant" (Kocher et al): statefull session keys, derived from long term key

"Fresh rekeying" (Medwed, 2010): from "rekeying" (Abdalla, Bellare, 2000); stateless, fresh key derived from long term key

Using a tweakable BC: Mennink et al 2015, again stateless

Roadmap

Intrinsic Resilience to Side Channel Attacks?

An interesting question is if any construction offers intrinsic resilience.

Early on [19] looked at the link between properties of target functions that relate to the "ease" by which differential attacks succeed: "transparency order".

Among several researchers [6] looked at this connection more recently, and they evaluated a range of properties in relation to some actual attacks on a nicely leaking target; they looked at a range of target functions using different instructions on their processor.

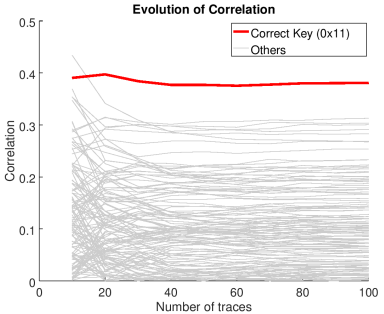
Their findings are in line with the existing knowledge: some instructions leak more per se, and some target functions enable easier discrimination between key guesses.

The conclusion was that ARX constructions seem to be preferable because they use instructions that leak less and the targets that they offer make discriminating key guesses harder.

Intrinsic Resilience to Side Channel Attacks?

But ARX constructions imply processing “similar intermediates” in consecuting clock cycles; i.e. in parallel on any modern pipelined architecture.

The signal amplification effect (also commented on in [6]) leads to a dramatic improvement on attacks just using [25].



The gain from fewer memory instructions + less leaky targets does not seem to be strong enough to really promote ARX as a “better” design principle w.r.t. side channel attacks. The implementation of masking countermeasures is also not that much less painfull than for other schemes.

Lightweight Ciphers and Side Channel Attacks

Heuser et al. [10, 11] conducted a comparative study covering several lightweight ciphers: “AES, Zorro, Robin, Klein, Midori, Mysterion, LED, Piccolo, Present, Pride, Prince, Rectangle, Skinny ”

They study both a profiled and an unprofiled setting, always in a single target setting. They do study how difficult it is to recover 4-bit vs 8-bit intermediate states though (but do not really do a multi target attack).

The experiments are all based on simulations, using only HW and HD leakage models.

There focus is on the size of the S-box and the S-box itself. The conclusion is that there are marginal differences only: to reach the same SR, attacks use between 150 and 400 traces, depending on the cipher.

Banciu et al.[3] provide a comparative study as well: AES, Present, Klein, LED.

They focus on profiled attacks in a single trace, multi-target setting, using a so-called "pragmatic" evaluation technique (i.e. direct enumeration of the remaining key space, no use of solvers, no use of belief propagation).

The experiments are all based on simulations, using only HW and HD leakage models.

Their conclusion is that the impact of the S-box design, or diffusion characteristics irrelevant for resilience, but the number of statistically independent intermediates that leak is a good predictor for vulnerability.

Lightweight Ciphers and Side Channel Attacks

A 2017 Study by Diel et al from GMU compared implementations of several lightweight ciphers when implemented on an FPGA:

`https://people-ece.vse.gmu.edu/~kgaj/publications/conferences/GMU_FPT_2017_LWBC_DPA_slides.pdf`

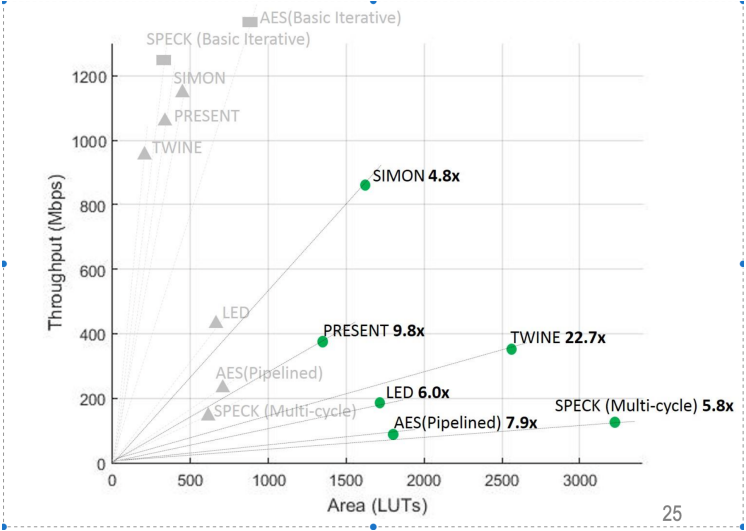
The study compared unprotected and **protect implementations**.

The side channel protection was based on a threshold masking scheme (3 shares, should offer security against standard differential attacks, evaluated by TVLA, which we'll cover in Part 2).

But they have to adust the implementation for AES and Speck because the threshold scheme makes assumptions that the actual implementation doesn't adhere to.

They provide a performance comparison (see next slide), which highlights the enormous penalty that is paid for masking.

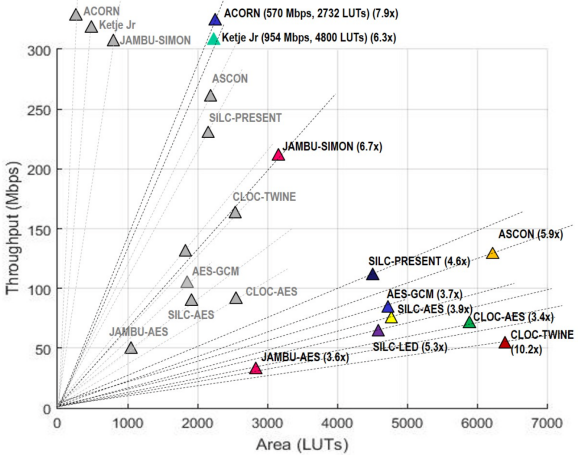
Lightweight Ciphers and Side Channel Attacks



25

Lightweight Ciphers and Side Channel Attacks

Using the same countermeasure, the same group also looked at the performance of AEAD schemes. https://vtechworks.lib.vt.edu/bitstream/handle/10919/85067/cryptography-02-00026.pdf



What about mode level considerations? Can tie in with “re-keying”, which can be advertised as “provable security” on the mode level.

But beware: most constructions offer (arguably) a weak notion of provable security. The argument is that

- the rekeying part must be protected against SPA and DPA (i.e. single trace and differential attacks)
- the block cipher part must only be protected against SPA,
- then the overall construction is secure against side channel attacks (aka DPA).

This is not the same as arguing security in the (X) probing model or in the “leakage resilient” model.

Why should single trace attacks using multiple targets not be considered a serious threat? Is protection against single trace/multi target attacks cheaper than protection against multi trace attacks? (Perhaps in hardware, but not in software)

Masking: n-shares typically implies n-fold increase in something. Efficiency can be improved by reusing randomness, and minimising multiplications (counts and depth); check out de Meyer: `https://eprint.iacr.org/2020/699`.

Boolean masking: AND is much more expensive than anything else (perhaps a useful proxy to determine how painful masking will be)

The number intermediates determines the "attack surface" for multi target attacks: more efficient implementations tend to minimise the attack surface.

Conversion between types of masking is also expensive and can introduce more options for errors (thus ciphers that require more than one type are probably not an advantage).

True provable security at the mode level comes at a cost, and "re-keying" modes offer something, but not in the "standard" (X) probing or LR model.

Roadmap

Side Channel Attacks on Post-Quantum Schemes

The already vast literature on SCA on NIST PQC schemes [2] is clearly only in its infancy and we can expect many more attacks and countermeasures for years to come.

Thus, we provide a non-comprehensive list of SCA on NIST PQC schemes which illustrate the paradigms we've introduced earlier; single-/multi-trace and single-/multi-target.

The discussions and examples focus on lattice-based KEMs as this has been a big focus in the literature.

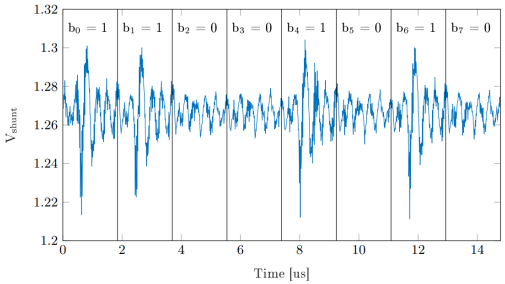
Single-Trace, Single-Target Attacks on Lattice-Based KEMs

Using a single-trace and focusing on a single-target are powerful attacks but limit what you can achieve.

The targets for two attacks, on NewHope [1] and Kyber [21], thus target the shared secret, via the message encoding, rather than the secret-key. Attacking the shared secret also limits to a single-trace attack anyway.

In [1], one can even read the message here from the oscilloscope via the naked eye. Here we see the byte value 83 (01010011 in binary).

We've seen similar attacks like this in the past on RSA and ECC for algorithms like square-and-multiply!



Single-Trace Attacks on Lattice-Based KEMs

In [21] they focus more on countermeasures on message encoding, all of which are attackable with single traces attacks, with certain success rates, using different amounts of points (within the single trace) ranging from 525-1755, see [21, Table 1].

- Ref. implementation – **1535** (POIs).
- Encoding via multiplication – **796**.
- Data indep. poly. gen. – **525**.
- Balanced data indep. poly. gen. – **700**.
- Polynomial randomization – **1231**.
- Byte / bit level random order – **1755**.

Both papers study implementations on a Cortex M4: the Cortex Family is well used/studied in the context of symmetric crypto implementations and there is ample evidence that **no serious side channel security is possible**.

Attacking and Defending Lattice-Based KEMs I

The Saber team designed a (first-order) masked design [5] which has been the target of a few attacks recently.

In [18], session keys and secret keys are retrieved using deep neural networks (created at a profiling stage), but require 2.5k and 62k traces. They target multiple points in the masked logical shifting on arithmetic shares function (`poly_A2A()`) and its shuffled variant (`poly_A2A_shuffled()`).

In [17] the session key and long-term secret-key are recovered from 16 traces using deep learning power analysis, without needing to profiling the scheme with masks deactivated. They target similar points-of-interest (`poly_A2A()` and `POL2MSG()`) for message recovery, and for the secret-key using maps from error-correcting codes using single and multiple traces.

The target device is once more a very leaky Cortex M4, running at a lower frequency compared to [21].

Attacking and Defending Lattice-Based KEMs II

Kyber have also designed a masked scheme [7], for first-order as well as high-order protection. This is a relatively new publication so attacks have not been seen yet.

In [24] the unmasked Kyber is targeted to first find the secret key using EM, requiring 4 traces, on the reference implementation. The same attack is not viable for the assembly-optimised version, thus they target message recovery by also attacking the decoding function which requires profiling to find POIs.

More countermeasures are designed in [9] to protect against DPA and fault attacks. The DPA protection is realised in a RNR-protected NTT multiplier and achieves a relatively low overhead of 1.13x compared to unprotected.

Reading Material I

[1] D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden.
Defeating newhope with a single trace.
PQCrypto, 2020:368, 2020.

[2] D. Apon and J. Howe.
Attacks on NIST PQC 3rd Round Candidates.
IACR Real World Crypto Symposium, January 2021.

[3] V. Banciu, E. Oswald, and C. Whitnall.
Exploring the resilience of some lightweight ciphers against profiled single trace attacks.
In COSADE 2015.

[4] L. Batina, B. Gierlichs, and K. Lemke-Rust.
Differential cluster analysis.
In CHES 2009.

Reading Material II

[5] M. V. Beirendonck, J.-P. D'anvers, A. Karmakar, J. Balasch, and I. Verbauwhede.
A side-channel-resistant implementation of saber.
ACM Journal on Emerging Technologies in Computing Systems (JETC), 17(2):1–26, 2021.

[6] A. Biryukov, D. Dinu, and J. Großschädl.
Correlation power analysis of lightweight block ciphers: From theory to practice.
In ACNS 2016, 2016.

[7] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal.
Masking kyber: First-and higher-order implementations.
IACR Cryptol. ePrint Arch., 2021:483, 2021.

[8] S. Chari, J. R. Rao, and P. Rohatgi.
Template attacks.
In CHES 2002.

Reading Material III

[9] D. Heinz and T. Pöppelmann.
Combined fault and dpa protection for lattice-based cryptography.
IACR Cryptol. ePrint Arch., 2021:101, 2021.

[10] A. Heuser, S. Picek, S. Guilley, and N. Mentens.
Side-channel analysis of lightweight ciphers: Does lightweight equal easy?
In RFIDSec 2016.

[11] A. Heuser, S. Picek, S. Guilley, and N. Mentens.
Lightweight ciphers and their side-channel resilience.
IEEE Trans. Computers, 69(10):1434–1448, 2020.

[12] A. Heuser, O. Rioul, and S. Guilley.
Good is not good enough - deriving optimal distinguishers from communication theory.
In CHES 2014.

Reading Material IV

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

[14] S. Mangard, E. Oswald, and F.-X. Standaert.
One for All – All for One: Unifying Standard DPA Attacks.
IET Information Security, 2011.

[15] L. Mather, E. Oswald, and C. Whitnall.
Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer.
In ASIACRYPT 2014.

[16] 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.

Reading Material V

[17] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson.
A side-channel attack on a masked ind-cca secure saber kem.
IACR Cryptol. ePrint Arch., 2021:79, 2021.

[18] K. Ngo, E. Dubrova, and T. Johansson.
Breaking masked and shuffled cca secure saber kem by power analysis.
Cryptology ePrint Archive, 2021.

[19] E. Prouff.
Dpa attacks and s-boxes.
In International Workshop on Fast Software Encryption, pages 424–441. Springer, 2005.

[20] W. Schindler, K. Lemke, and C. Paar.
A stochastic model for differential side channel cryptanalysis.
In J. R. Rao and B. Sunar, editors, CHES 2005.

Reading Material VI

[21] H. M. Steffen, L. J. Kogelheide, and T. Bartkewitz.
In-depth analysis of side-channel countermeasures for crystals-kyber message encoding on arm cortex-m4.
CARDIS 2021, 2021.

[22] N. Veyrat-Charvillon, B. Gérard, and F. Standaert.
Soft analytical side-channel attacks.
In ASIACRYPT 2014.

[23] C. Whitnall and E. Oswald.
Robust profiling for dpa-style attacks.
In CHES 2015.

Reading Material VII

[24] Z. Xu, O. M. Pemberton, S. S. Roy, D. Oswald, W. Yao, and Z. Zheng.
Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber.
IEEE transactions on computers, 2021.

[25] Y. Yan and E. Oswald.
Examining the practical side channel resilience of arx-boxes.
In ACM CF 2019, 2019.