Side-Channel Attacks Revisited — an Optimization Problem Perspective: Bootstrapping and Space Reduction

Erez Tamir, Osnat Keren, Itamar Levi

Bar-Ilan University, Ramat-Gan, Israel
erez.tamir@biu.ac.il,osnat.keren@biu.ac.il,itamar.levi@biu.ac.il

Abstract. Side-channel analysis (SCA) attacks rely on leakage from a target device. It is common to assume that linear operations implemented by XOR gates produce symmetric leakage and carry negligible side-channel information. In practice, leakage from XOR gates produces complex, non-independent, and time-varying asymmetric behavior.

The paper introduces Feature Estimation based Attacks (FEbA) – a dedicated profiling attack that exploits these asymmetries. The attack is versatile; it was demonstrated to be successful against the sharing and refreshing phases in masking-based implementations by greatly narrowing the guessing key space, *with no access to intermediate values*.

Such attacks have implications for designs such as ASCON, GIBBON, and ACE, where XORs that utilize the key are vulnerable to attacks regardless of the inherent SCA protection levels used in them (e.g., sponge *rate*, the leak-free components for re-keying, and masking order d).

Experimental results indicate that the entropy of a 32-bit key can be reduced below 1 bit using (up to) $20,000$ traces from a standalone XOR without any access to intermediate values, or below 500 traces with access to intermediate values.

Keywords: Feature Estimation Attack · FEbA · Masking · Profiling · SCA · Side Channel Analysis Attacks · Sharing

1 Introduction

A central challenge in side-channel analysis (SCA) research is to take a set of adversarial capabilities and constraints and translate them into a practical methodology for extracting information about some targeted binary secret (key). These capabilities may include access to current leakages, knowledge of algorithm inputs, or visibility into specific machine-level instructions sequence, and perhaps additional abilities such as injecting faults into the hardware. SCA adversaries must process large amounts of information and evaluate many keys to extrapolate a likely key candidate. As a consequence, the runtime of attacks may quickly exceed the processing capabilities of modern processors, whether in terms of runtime or storage. As such, adversaries must significantly divide and conquer and take measures like key divisions, reduce data-complexity by dimensionality reduction or pruning points of interest (POI), and more.

Much effort has been invested into creating schemes that protect an algorithm against side channel attacks. To date, masking is the most widely discussed construction [CJRR99, ISW03, PR13, CGLS20], using a linear (XOR-based) random encoding to whiten the leakage profile of software and hardware. Such a whitening is used as a main mechanism in different cryptographic contexts, from One-Time Pad (OTP) in cryptographic

applications [Sha49] to black-box constructions such as interactive protocols [HLVA02], and in state-of-the-art SCA-aware NIST-finalist symmetric constructions such as ASCON [DEMS21]. Therefore, conducting a thorough analysis of the side-channel characteristics of a simple XOR operation implementation is highly important.

Masking/ sharing schemes are not perfectly secure, especially when intermediate values are known. It is straightforward to extract secret information using high-order attacks (e.g., [CPRR13])– attacks that cross-correlate leakages from different time-samples with the intermediate values (masked or not), giving a score to each sub-key before they are ranked. Even in a more restrictive case where intermediate values are not known, attacks that compound information from different time-samples, such as Algebraic Side-Channel Attacks [RSVC09] can extract the key. An Algebraic attack uses (e.g.,) a SAT solver that, given a Hamming Weight characterization of the leakages and a formulaic characterization of the algorithm used, the correct key can be solved for. This type of attack has inherent limitations related to the number of available constraints that are related to the key, the device's noise profile, and the ability to correctly map each leakage sample to its corresponding Hamming Weight.

Consequently, to date, no adversary was able to extract the secret key from leakages of only a (very) small number of operations without having access to intermediate values, particularly when protected schemes are used. Fig. 1 illustrates five reference scenarios for which leakages are obtained from a simple XOR operation: cryptographic solutions where sensitive information is XORed with a known intermediate (such as scenarios 1 and 3 in Fig. 1) are known to be vulnerable to attacks such as CPA, template, or MI-attacks [BCO04, CRR03, GBTP08a]. But as far as we know, there has been no thorough discussion about the potential adversaries for solutions where sensitive information is XORed with an unknown intermediate (such as scenarios 3, 4, or 5 in Fig. 1), namely leakage of sharing with an unknown mask and masked or refreshed XOR.

In this paper, we introduce a methodology for leveraging side channel traces by discarding two implicit assumptions commonly made during the leakage modeling stage of many classic attacks. The first assumption, which we dub the *symmetric leakages* assumption, assumes that different instances of the same logic gate produce the same leakage. Under this assumption, symmetric operations between a key and unknown value (such as scenario 3 in Fig. 1) reveal at most the value of $\mathrm{HW}(\mathbf{k})$; That is, up to $\log_2(n_k+1)$ bits of information may be extracted about n_k-bit keys. Without this assumption, the theoretical bound on the information extracted about the key is removed, allowing us to learn it in its entirety in practical cases, as shown in Sec. 6.1. The second assumption we discard is the *additive noise* assumption, it is typically stated as $L = f(\mathbf{k}) + N$ with $N \perp \mathbf{k}$. This assumption neglects to consider interactions that make it possible to extract higher-order and more structurally complex information from the leakage, as performed in Sec. 6.4

Throughout this paper, we allow an *additive decomposition into components whose distributions may depend on processed values* (e.g., switched random variables), and regress on statistical characteristics, or as we refer to them throughout this paper, 'features', of the distribution. *Once we discard the underlying assumptions of the classical models, it becomes not only possible but also feasible to extract the entire key using these leakages alone.*

The regression-based profiling methodology originates from Schindler's stochastic model [SLP05] which resembles FEbA on an abstract level (to some extent). However, it differs on key aspects of the attack, we extend the stochastic model along three main axes: first, we target XOR operations where the intermediates used are *inaccessible*, utilizing bit leakage variations for the attack, which were not exploited in for that purpose [SLP05]. The proposed approach differs in the features and how we select them - in its simplest abstraction, we minimize a metric between some leakage model and a tunable feature of

the distribution, which allows for large-scale data handling, rather than minimizing the mean leakage difference[1]. We also use binary optimization to enable attacks on larger key sizes, which is beyond the scope of [SLP05]. Importantly, we develop concrete bounds and analytical evaluation of the attack, while also providing rigorous statistical techniques to reduce complexity with verifiability and explainability. For the degenerated base scenario - *with known intermediates*, which is less interesting, and targeting a simple mean leakage characteristics, FEbA are functionally similar to Schindler's stochastic model, with the main differences: the search metric being generated from the attack trace's covariance, the loss function, and the provided bounds.

Compared to previous works, our methodology is resource-efficient, as it reduces attacks to a scalable optimization problem. For attacks like Template attacks [CRR03], mutual information attacks [GBTP08a] and neural network the cost is clear: the attack complexity will grow quickly in regards to one/ many of the following parameters: the number of attack traces (n), number of key bits (n_k), number of leakage traces (T), trace leakage length (l), number of Points of Interest (POIs) in the *leakage* considered (d), size of oscilloscope alphabet ($|\mathbb{X}|$) and perhaps more device or attack specific variables. The attack we propose remains stable for increments in all of the parameters above, with two caveats: our usage of a heuristic optimization algorithm for key sizes > 32 and difficulty with handling constrained space searches (models dependent on $f(\mathbf{k})$ for a complicated fs).

In this paper, we examine how to abstract leakage in a formal and usable way, enabling us to reason about it effectively. We leverage it to gain practical insight into the (in)security of concrete cryptographic schemes and the (in)security of masking and XOR *whitening*. In the remainder of this work, we use the term *feature* to refer to an estimator that captures the leakage characteristics associated with a specific key and, when applicable, plaintext.

1.1 Our Contribution

The main contribution of this paper is the definition of a new class of side-channel attacks, termed *feature estimation based attacks* (FEbAs), along with several demonstrations of its application. These attacks are built upon a probabilistic approach, which bounds and significantly reduces the key-guessing space by comparing features of the attack leakages to a profiled model. Following this rationale, it is possible to construct multiple adversaries flexibly and analyze their theoretical performance for different targets, potentially targeting different protection schemes, different algorithmic assumptions, different adversarial limitations, and more. Concretely, referring to Fig. 2 FEbAs contribute to the concept of profiling features as illustrated in Step 3 and Step 4. These two contributions allow us to create a leakage profile with more fidelity in a low number of traces, even for a vast key space (Sec. 6.3). This methodology converts side channel attacks to a discrete optimization problem with continuous parameters, and does so in such a way that theoretical analysis of the target is possible before knowledge of the attack traces. It also allows us to leverage traces without access to intermediate values to create an attack (Sec. 6.1). Finally, feature estimation attacks are built to be computationally feasible even for large key/data-space attacks that would require too many resources otherwise. A high-level overview of the experiments performed to highlight the different features of FEbA and their results are summarized in Table 3. The main theorem of this paper, discussing an apriori theoretical trace analysis on a nominal target, can be seen in Sec. 1 (Theorem 1) along the result that validates it is in Sec. 3.5.

[1] In some cases, where both model and search metrics are independent of the sample index, these two things are the same. This is not the case for [SLP05]

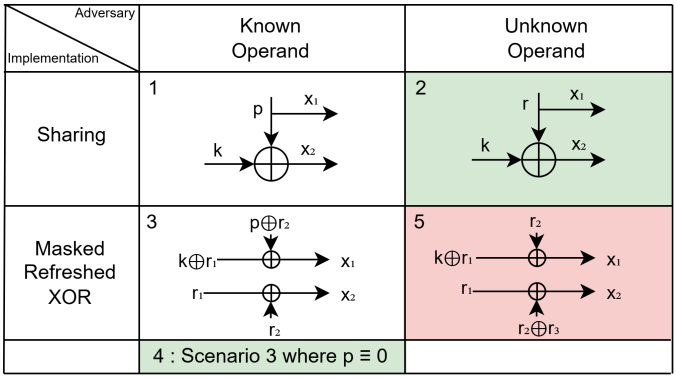


Figure 1: Side-Channel Sharing and Masking Attack Scenarios. Green (red) scenarios denote targets for which no successful side-channel attack had previously been reported and where FEbA is shown to be successful (unsuccessful).

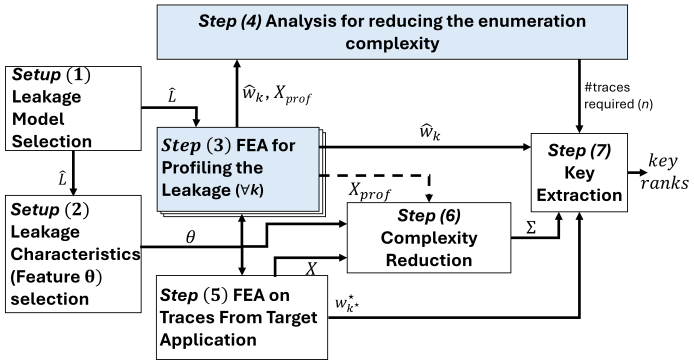


Figure 2: High-level description of the nominal FEbA attack procedure. The steps shown in blue correspond to the main innovations of this work, namely the exploitation of asymmetric leakage and the mathematical modeling of the adversary.

Outline: The remainder of this paper is organized as follows. Tab. 1 recalls notations used throughout the paper, Section 2 defines what a feature estimation attack is. Section 3 shows the idea behind feature estimation attacks. Section 4 discusses how to construct generic feature estimation attacks. Section 5 discusses the research methodology along with several important caveats. Sections 6 discusses the results. Section 7 summarizes the paper and talks about future implications and considerations.

2 Leakage Characteristics Estimators (Feature)

2.1 Leakage Model and Perfect Feature Estimator

Consider a key $\mathbf{k} \in \mathbb{B}^{n_k}$, a plaintext $\mathbf{p} \in \mathbb{B}^{n_p}$ and randomness $\mathbf{r} \in \mathbb{B}^{n_r}$. Denote by $\mathbf{y} \in \mathbb{Y} = \mathbb{B}^{n_k+n_p+n_r}$ the triplet $\mathbf{k}||\mathbf{p}||\mathbf{r}$. The key $\mathbf{k} \sim \chi$ is an unknown variable rolled once per target instance, $\mathbf{r} \sim \zeta$ unknown variable rolled once per algorithm execution (i.i.d). Denote by $A_{\mathbf{k}}(\mathbf{p};\mathbf{r})$ a keyed algorithm.

We denote S as the probability space that determines physical noise and device variations over time. A leakage model is a random function $L : \mathbb{Y} \times S \to \mathbb{X}^l$ that characterizes the leakage. Denote by $L_{\mathbf{y}} = L(\mathbf{y}, s)$ the random variable that represents the leakage given

Table 1: Recurring Symbols

Symbol	Meaning
Variables:	
$\mathbb{Z}, \mathbb{B}, \mathbb{R}, \mathbb{Y}$	The set of integers, binary set $\{0,1\}$, real numbers, leakage labels
$\mathbf{p}, \mathbf{k}, \mathbf{r}$	Plaintext, secret key, randomness
$\mathbf{k}^\star$	The correct key present in an instance of a target
$\mathbf{y}$	The triplet $(\mathbf{p}, \mathbf{k}, \mathbf{r})$
n_p, n_k, n_r	Lengths of $\mathbf{p}, \mathbf{k}, \mathbf{r}$
K	Number of allowed key guesses for the adversary
$\mathbf{x}_i$	Indexing a vector $\mathbf{x}$ for its i^{th} element (a scalar)–unweighted letter
$\mathbf{x_i}, \mathbf{x^i}$	Indexing a list of vectors $\mathbf{x}$ for its i^{th} element (a vector)–bold letter
$\mathbf{x}^{\mathbf{j}}_{\mathbf{i},m}$	m^{th} element of i^{th} vector in jth list of vectors
Leakage variables:	
ℓ	Leakage vector (trace)
l	Number of samples per trace
d	Dimension of the feature θ
$\mathbf{X}$	Set of traces $\{(\ell_i, \mathbf{p}_i)\}$
n	Number of traces, number of pairs in $\mathbf{X}$
$W_{\mathbf{k}^\star}(\mathbf{X})$	Position of the true key in key candidate list from the attack
$n(K, \delta)$	The number of traces required as a function of K and success rate $\geq 1-\delta$
Z	A generic random variable used for leakage
S	Probability space of physical noise and variations over time
$A_{\mathbf{k}}(\mathbf{p}; \mathbf{r})$	Keyed algorithm with unknown refreshed input $\mathbf{r} \sim \zeta$, known input $\mathbf{p}$, unknown constant input $\mathbf{k} \sim \chi$
$L^\star / L^\star_{A_{\mathbf{k}}(\mathbf{p};\mathbf{r})} / L^\star_{\mathbf{k},\mathbf{p},\mathbf{r}}$	The faithful leakage distribution (FLD)
$\hat{L} / \hat{L}_{\mathbf{k},\mathbf{p},\mathbf{r}}$	Estimated leakage RV
θ / θ_k	A feature of the leakage, $\theta_{\mathbf{k}} : (\mathbb{X}^l \times \mathbb{B}^{n_p})^* \to \mathbb{R}^d$
$w_{\mathbf{k}} / w^\star_{\mathbf{k}} / \hat{w}_{\mathbf{k}}$	The feature estimation random variable under a distribution/ the FLD/ the model
$f_{\mathbf{k}} / f^\star_{\mathbf{k}} / \hat{f}_{\mathbf{k}}$	The expected value of the estimation a distribution/ the FLD/ the model
$\sigma^L / (\Sigma^L)$	The covariance (matrix) of the leakages on a target key $\mathbf{k}^\star$
Σ^W	The covariance on the (linear) model
T	The number of leakages used for profiling
W	Beam width for K-best algorithm
Functions and operators:	
$unravel()$	The function that maps a matrix $\mathbb{R}^{n \times m}$ to a vector $\mathbb{R}^{n \cdot m}$
$(\cdot, \cdot)$	Row-wise matrix/vector concatenation
$\|\|$	Column-wise matrix/vector concatenation
$\|\mathbf{x}\|_\Sigma$	Mahalanobis norm $\|\mathbf{x}\|_\Sigma = \sqrt{\mathbf{x}^T \Sigma^{-1} \mathbf{x}}$
$F_{\chi^2_l(;\Lambda)}$ ($F_{\chi^2_l(m)}$)	CDF of the generalized (non-) centered chi-square distribution of dimension l with coefficients $\{\Lambda_i\}_i$ (and offset m)
μ	The mean estimator
\$	Draw from a uniform distribution
M / M_{opt}	The sampled/ ideal matrix models (matrices with d columns, equivalent to $\hat{w}_{\mathbf{k}} / \hat{f}_{\mathbf{k}}$ when considering a linear model)
$M \cdot \mathbf{v_i} \leftarrow \mathbf{w_i}$	Fitting a M via linear regression using labels $\mathbf{w_i}$ and inputs $\mathbf{v_i}$
A^*	The set $\{\} \cup A \cup A^2 \cup A^3 \ldots$
$\mathcal{O}(), \Omega(), o()$	Asymptotic complexity notations
$t\text{-}val(\mathbf{x}, \mathbf{y})$	The t-statistic between $\{x_i\}_i$ and $\{y_i\}_i$

an input $\mathbf{y}$. We say that a leakage model $L^\star$ is "faithful" to $A_{\mathbf{k}}(\mathbf{p};\mathbf{r})$ if

$$\Pr[L^\star_{\mathbf{y}} = \ell] = Pr[\ell \overset{\text{smp.}}{\leftarrow} A_{\mathbf{k}}(\mathbf{p};\mathbf{r})] \tag{1}$$

Formally, $\ell \overset{\text{smp.}}{\leftarrow} A_{\mathbf{k}}(\mathbf{p};\mathbf{r})$ denotes the random variable produced by the measurement procedure when executing $A_{\mathbf{k}}(\mathbf{p};\mathbf{r})$ on inputs $\mathbf{y}$, where 'smp.' is the side channel sampling method used by the adversary. We denote the faithful leakage distribution (FLD) with $L^\star$ (faithful to the leakage random variable).

Here, the sampling method may be any measurement, whether it is electromagnetic EM samples, power/current leakage samples (as this paper does), temperature, time, or a combination of different measurements. Notice that we do not distinguish between the 'true' measurements of the system and the noisy and quantized current sampled by the sampling method.

For simplicity, we define $L_{\mathbf{k},\mathbf{p}}$ as a random variable that represents the leakage given $\mathbf{k}$ and $\mathbf{p}$, but not $\mathbf{r}$. That is the random variable that fulfills $L_{\mathbf{y}} \sim L_{\mathbf{k},\mathbf{p}}|(\zeta = \mathbf{r})$. With a slight abuse of notation, when it clear from the context, we omit the subscript of $\mathbf{p}$. Throughout this paper, we denote n as the number of traces.

Definition 1 (Feature)**.** A (stable) feature, θ_k, of a leakage model L given a key $\mathbf{k} \in \mathbb{B}^{n_k}$, is a function

$$\theta_{\mathbf{k}} : (\mathbb{X}^l \times \mathbb{B}^{n_p})^* \to \mathbb{R}^d, \tag{2}$$

that for all $\mathbf{k}^\star, \mathbf{k} \in \mathbb{B}^{n_k}$ satisfies $\|w^n_{\mathbf{k}}(\mathbf{k}^\star) - E[w^n_{\mathbf{k}}(\mathbf{k}^\star)]\| \xrightarrow[n\to\infty]{P} 0$, (and $\forall n : E[w^1_{\mathbf{k}}(\mathbf{k}^\star)] = E[w^n_{\mathbf{k}}(\mathbf{k}^\star)]$, to be called stable[2]) Where $w^n_{\mathbf{k}}(\mathbf{k}^\star) \sim \theta_{\mathbf{k}}(\mathbf{X}_{\mathbf{k}^\star})$ (implicitly dependent on $\{\mathbf{p_i}\}$) and $\mathbf{X}_{\mathbf{k}^\star} \sim \{L_{\mathbf{k}^\star,\mathbf{p_i}}, \mathbf{p_i}\}^n$ is a set of n leakage-plaintext pairs, illustrated in Fig. 3).

The feature function provides an asymptotically stable characterization of the given leakages for every key. A feature may be a functional average estimator (i.e., $\theta_{\mathbf{k}}(\mathbf{X}) = \frac{1}{|\mathbf{X}|}\sum_i h_{\mathbf{k}}(\ell_{\mathbf{i}}, \mathbf{p}_i)$ for some h, e.g., vectorized mean estimator) or perhaps U-statistics [Hoe48]/ M-estimators [Hub64], etc...

We say that a feature is independent of $\mathbf{k}$ if $\forall \mathbf{k}, \mathbf{k}', \mathbf{X} : \theta_{\mathbf{k}}(\mathbf{X}) = \theta_{\mathbf{k}'}(\mathbf{X})$. Otherwise, the feature is said to be dependent on $\mathbf{k}$.

Denote by $f_{\mathbf{k}}(\mathbf{k}^\star) = E[w^n_{\mathbf{k}}(\mathbf{k}^\star)] = E_{\mathbf{X}\sim L^n_{\mathbf{k}^\star}}[\theta_{\mathbf{k}}(\mathbf{X})]$ the expected value of the estimation $w_{\mathbf{k}}$. From this point, to simplify notations, we use $w^n_{\mathbf{k}} = w^n_{\mathbf{k}}(\mathbf{k})$, $f_{\mathbf{k}} = f_{\mathbf{k}}(\mathbf{k})$. We also omit the number of observations n from $w^n_{\mathbf{k}}(\mathbf{k}^\star)$, when the number of traces is implicitly n i.e., $w^n_{\mathbf{k}} = w_{\mathbf{k}}$.

In what follows, we denote by $w^\star_{\mathbf{k}}(\mathbf{k}')$ and $\hat{w}_{\mathbf{k}}(\mathbf{k}')$ as the feature estimation of $L^\star$ (the FLD) and $\hat{L}$ (A profiled model), respectively. Also, we denote by $f^\star_{\mathbf{k}}(\mathbf{k}')$ and $\hat{f}_{\mathbf{k}}(\mathbf{k}')$ as the expected value of the estimation of $L^\star$ and $\hat{L}$ respectively[3].

A high level summary of the definitions is present in Fig. 3.

We denote by $P_{a|b}$ the prior key distribution

$$P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} = \Pr[w^{(|\mathbf{X}|)}_{\mathbf{k}} = \theta_{\mathbf{k}}(\mathbf{X})|\mathbf{k}, \{\mathbf{p_i}\}_{i=1}^N]$$

and the posterior key distribution $P_{\mathbf{k}|\mathbf{X},\{\mathbf{p_i}\}} = \frac{\Pr[\chi=\mathbf{k}]P_{\mathbf{X}|\mathbf{k},\mathbf{x}_p}}{\sum_{k'}\Pr[\chi=\mathbf{k}']P_{\mathbf{X}|\mathbf{k}',\mathbf{x}_p}}$ where $\mathbf{X} = \{\ell_{\mathbf{i}}, \mathbf{p_i}\}_{i=1}^n$.

[2]The distinction between a stable and non-stable features is necessary in few places; the reduction to model based attacks such as CPA seen in Appendix A.1 in which the reduced feature is non-stable. In this work, we generally refer to stable features as features.

[3]For clarity, we highlight main notations which also appear in Tab. 1: $.^\star$ denotes a 'true' value. When used with a key, it denotes the reference/ targeted key ($\mathbf{k}^\star$), and when used with a feature, it denotes the FLD ($w^\star_{\mathbf{k}}(\mathbf{k}'), f^\star_{\mathbf{k}}(\mathbf{k}')$). $\hat{.}$ is used to denote an estimation. When used with a feature, it denotes an object created with a profile of the FLD ($\hat{w}_{\mathbf{k}}(\mathbf{k}'), \hat{f}_{\mathbf{k}}(\mathbf{k}')$).

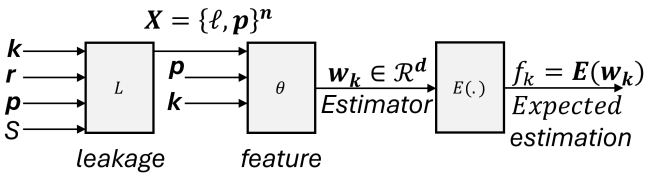


Figure 3: Visualization of the Definitions in Section 2 - Feature, Its Estimation, and Expected Value.

Definition 2 (Perfect feature)**.** A feature θ is said to be *perfect* with respect to a leakage model L, if for every key $k^\star \in \mathbb{B}^{n_k}$: $\lim_{n\to\infty} \Pr[\mathbf{k}^\star = \arg\max_{\mathbf{k}} P_{\mathbf{k}|\mathbf{X}}] = 1$, where $\mathbf{X} \sim \{L_{\mathbf{k}^\star,\mathbf{p_i}}, \mathbf{p_i}\}^n$.

The definition above characterizes a feature that asymptotically 'separates' the keys, making each key uniquely identifiable with certainty at the limit (in theory). This idea is formalized in the following lemma,

Lemma 1. *A feature θ is perfect if there exists $\epsilon > 0$ for which*

$$\forall \mathbf{k} \neq \mathbf{k}' : \|f_{\mathbf{k}}(\mathbf{k}) - f_{\mathbf{k}}(\mathbf{k}')\| > \epsilon$$

where n is the number of attack traces.

The meaning of this lemma is that as long as a feature can differentiate between any two keys based on the leakage, we may extract the key, though, it tells us nothing about the complexity (in traces or computation) of such a procedure.

Proof. For $\mathbf{X} \sim \{L_{\mathbf{k}',\mathbf{p_i}}, \mathbf{p_i}\}^n$:

$$\begin{aligned} P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} &= \Pr[w_{\mathbf{k}} = \theta_{\mathbf{k}}(\mathbf{X})|\mathbf{k}, \{\mathbf{p_i}\}_{i=1}^n] \\ &= \Pr[w_{\mathbf{k}} - f_{\mathbf{k}} = \theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}^n|\mathbf{k}, \{\mathbf{p_i}\}_{i=1}^n] \\ &\leq \Pr[\|w_{\mathbf{k}} - f_{\mathbf{k}}\|_V = \|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}^n\|_V|\mathbf{k}, \mathbf{X_p}] \\ &\leq \Pr[\|w_{\mathbf{k}} - f_{\mathbf{k}}\|_V \geq \|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}\|_V|\mathbf{k}, \mathbf{X_p}] \end{aligned} \tag{3}$$

We take $V = COV(w_{\mathbf{k}})$, which enables us to use the multidimensional Chebyshev's bound. Assuming $l = |L|$ is the number of samples in the trace vector:

$$\begin{aligned} P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} &\leq \frac{l}{\|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}^n\|_V^2} = \frac{l}{\sum_i ((\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}})^T u_i^V)^2/\lambda_i^V} \\ &\leq \frac{l}{\sum_i ((\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}})^T u_i^V)^2/\lambda_{max}^V} = \frac{l \cdot \lambda_{max}^V}{\|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}\|^2} \end{aligned} \tag{4}$$

Where $\{\lambda_i^V, u_i^V\}$ are the eigenvalues/normalized eigenvectors of V. . For $\mathbf{k} \neq \mathbf{k}'$, , $\lambda_{max}^V = \max_i \lambda_i^V$, we get that:

$$\begin{aligned} \lim_{n\to\infty} P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} &\leq \lim_{n\to\infty} \frac{l \cdot \lambda_{max}^V}{\|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}(\mathbf{k}') + f_{\mathbf{k}}(\mathbf{k}') - f_{\mathbf{k}}\|^2} \\ &\leq \lim_{n\to\infty} \frac{l \cdot \lambda_{max}^V}{(\|\theta_{\mathbf{k}}(\mathbf{X}) - f_{\mathbf{k}}(\mathbf{k}')\| - \|f_{\mathbf{k}}(\mathbf{k}') - f_{\mathbf{k}}\|)^2} \\ &= \lim_{n\to\infty} \frac{l \cdot \lambda_{max}^V}{(0-\epsilon)^2} = \frac{l}{\epsilon^2} \lim_{n\to\infty} \lambda_{max}^V = 0 \end{aligned} \tag{5}$$

In other words, $P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} \xrightarrow[n\to\infty]{P} 0$ when $\mathbf{X} \sim \{L_{\mathbf{k}',\mathbf{p_i}}, \mathbf{p_i}\}^n$

So $\forall \mathbf{k} \neq \mathbf{k}' : P_{\mathbf{k}|\mathbf{X}} = \frac{\Pr[\chi=\mathbf{k}]P_{\mathbf{X}|\mathbf{k},\mathbf{x}_p}}{\sum_{k_0}\Pr[\chi=\mathbf{k}_0]P_{\mathbf{X}|\mathbf{k}_0,\mathbf{x}_p}} \underset{n\to\infty}{\longrightarrow} 0$.

Due to that, and due to $P_{\mathbf{k}|\mathbf{X}}$ being a valid distribution (with sum 1 over the keys), we get $P_{\mathbf{k}'|\mathbf{X}} = 1 - \sum_{\mathbf{k}\neq\mathbf{k}'} P_{\mathbf{k}'|\mathbf{X}} \underset{n\to\infty}{\longrightarrow} 1$. □

2.2 Balanced (non-perfect) Feature Estimator

Perfect features discuss the asymptotic separation of different features. In real-world scenarios, distinguishing between keys and reverse-engineering the correct key from leakage may be difficult due to the potentially large key space or due to the complexity of the selected feature.

As such, we assume the adversary can verify the correctness of a maximum of K different keys (for example, via known plaintext-cipher indication).

Practically, we do not have access to a perfect estimation of $f_{\mathbf{k}}$ due to the finite amount of traces we are limited to. As such, we use an estimation of the 'perfect' feature and take the deviation from the idealized problem into account[4].

Given the adversary's new capability (which compensates for the infeasibility of perfect features), we define a relaxed condition:

Definition 3. For a feature θ, models L, L', and a function $\gamma(L, L')$, We say that θ is a γ-balanced estimator for L using L' if $\forall K$:

$$\gamma(L^\star, L') \leq K \Rightarrow \forall \mathbf{k}^\star : \lim_{n\to\infty} \Pr[\left|\{\mathbf{k} : P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}'|\mathbf{X}}\}\right| \leq K] = 1$$

for $\mathbf{X} = \{\ell_i, \mathbf{p}_i\}_{i=1}^n \sim (L^\star)_{\mathbf{k}^\star}^n$ with n the number of attack traces, and $P_{|\cdot}$ being defined over L'.

Reverse engineering of the key based on the leakage is an NP-Complete optimization problem, we focus on estimators that are *independent of the key*. Additionally, we assume $\Pr[\chi = \mathbf{k}] = \frac{1}{2^{n_k}}$.

The conceptual criteria for assessing the quality of the leakage model $\hat{L}$, and the effectiveness of the estimator θ, on which it relies are as follows:

1. **Model Complexity:** Given θ, we wish that it is reasonable to profile the leakage with a model $\hat{L}$. i.e., we are able to profile $\hat{w}_{\mathbf{k}}$ within a reasonable amount of resources (#traces, scope finite fidelity and resolution), in a way that estimates $f_{\mathbf{k}}^\star$ well enough. In other words, it is not difficult to create a model $\hat{L}$ for which $\gamma(L, \hat{L}) \leq K$ for a computationally achievable K and practically achievable number of profiling traces.
2. **Feature Validity:** We wish for a θ that results in a large value for $COV_{\mathbf{k}}(f_{\mathbf{k}}^\star)$ (variance over $\mathbf{k}$; the signal) and a small value for $E(COV(w_{\mathbf{k}}^\star|\mathbf{k}))$ (variance over $\mathbf{r}, S$; the noise). This is synonymous with the need for $\left|\{\mathbf{k} : P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^\star|\mathbf{X}}\}\right|$ to converge below K in a reasonable amount of traces with high probability.
3. **Computability:** For large key lengths (n_k), the leakage model should be exploitable in a manner that can generate key candidates within a reasonable computation time. In our case, this includes both the structure we choose for $\hat{L}$, and (ideally) the independence of $\theta_{\mathbf{k}}$ in $\mathbf{k}$. This point is usually irrelevant for $n_k < 32$, since this amount of keys is still brute-force-able.

One of the most studied problems to leverage in optimization problems [GN06] are linear models of the type $M \cdot labels(\mathbf{k})$. Using such models leaves us with the following

[4]This accounts for errors between $f^\star \leftrightarrow \hat{f}$ (the asymptotic value of the feature and the asymptotic model), or $w^\star \leftrightarrow \hat{w}$ (the estimated value of the feature and the estimated model). Sampling/ modeling limitations may cause errors between $\hat{f} \leftrightarrow \hat{w}$. Considering these types of errors, it is still reasonable to get a perfect estimator using $\theta = \mu$ for a profiling created within a reasonable amount of resources, as shown in Sec. 6.1.

optimization target (given key ratings based on the upper bounds of the probability in the proof above 2.1):

$$\hat{\mathbf{k}} \approx \arg\min_{\mathbf{k}} \|\hat{w}_{\mathbf{k}} - w^{\star}_{\mathbf{k}^{\star}}\|^2_{\Sigma} = \arg\min_{\mathbf{k}} \|M \cdot labels(\mathbf{k}) - \theta(\mathbf{X})\|^2_{\Sigma^L}, \tag{6}$$

Where the middle expression is the Mahalanobis distance between the feature profile and feature, and the second transition assumes linear modeling is used with a matrix M, some labeling function, and $\Sigma^L = COV(w^{\star}_{\mathbf{k}^{\star}})$ the covariance of the leakages of the targeted key. Further discussion on modeling and error abstractions is presented in Appendix A.2. The right-hand side of Eq. (6) is the (constrained) binary integer least square problem.

Heuristic methods (as used in [GN06]) can handle $n_k \gg 100$, which is relevant for more robust leakage models or for targets with multiple operations/ intermediate values dependent on the key in an observable way. Though, for the methods to feasibly work and converge to a correct solution, the leakage model M and/or the label function should favour blind search techniques, in a rough manner of speech. This optimization problem is also limited by the complexity of *labels* (and by virtue, of θ). For features too complex, we get a constrained optimization problem, which might prove too difficult even with the heuristic approach.

As for the computational limits, besides algorithmic complexity, the limiting factor becomes memory which is dependent on $|\theta|^2 = d^2$ without any optimizations (Σ has d^2 elements that need to be estimated). For the mean estimator, $d = l$, this does not pose a realistic issue (As seen in Sec. 6.1). Attempting to tackle masked implementations, we get a storage complexity of l^4 if a simplistic method is used, unrealistic for long, requiring feature filtering (As seen in Sec. 6.4).

Now, we show that usage of the mean estimator $\mu(\mathbf{X}) = \frac{1}{|\mathbf{X}|}\sum_i \ell_i$ fulfills the discussed criteria, leading to a theorem on a specified bitwise target (the operation $\mathbf{k} \oplus \mathbf{r}$ for example) with the following conditions (proof in Appendix A.5):

Theorem 1. *For $A_{\mathbf{k}}(\mathbf{p}; \mathbf{r})$ whose leakage is additive in its components, $\forall \hat{L}$, the mean estimator, $\mu(\{\ell_i, \mathbf{p_i}\}) = \mu(\{\ell_i\}) = \frac{1}{|X|}\sum_i \ell_i$, is a γ-balanced estimator for L using $\hat{L}$ with*

$$\gamma(L^{\star}, \hat{L}) = \binom{n_k}{n_k/2} F_{\chi^2_l(;\Lambda)}\left(\max_{\mathbf{k}} \|f^{\star}_{\mathbf{k}} - \hat{f}_{\mathbf{k}}\|^2_{\Sigma^L}\right) \tag{7}$$

In addition, the minimal number of traces needed for success w.p. $1-\delta$ is:

$$n = \min\{n' | n' \geq \frac{F^{-1}_{\chi^2_l(n' \| f^{\star}_{\mathbf{k}^{\star}} - \hat{f}_{\mathbf{k}^{\star}} \|^2_{\Sigma^L})}(1-\delta)}{F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})}\} \tag{8}$$

Where $F_{\chi^2_l(;\Lambda)}$ is the CDF of the l-dimensional generalized centered chi-square distribution with variances Λ_i. $F_{\chi^2_l(m)}$ is the CDF of the l-dimensional non-generalized non-centered chi-square distribution with offset m. And Λ is the list of eigenvalues of the matrix $\Sigma^Z = (\Sigma^L)^{-1/2}\Sigma^W(\Sigma^L)^{-1/2}$ where $\Sigma^W = COV_{\mathbf{k}}(\hat{f}_{\mathbf{k}})$ (a covariance signal matrix), $\Sigma^L = COV(w^{\star}_{\mathbf{k}^{\star}})$ (a covariance noise matrix on the targeted key).

3 Feature Estimation Based Attacks (FEbA) - The Rationale, Leakage Modeling and an Abstraction

Feature estimation-based attacks are attacks derived using the ideas gathered in the last chapter, following the rough construction shown in Fig. 2 (more formally elaborated in the next Section).

3.1 The rational

The rationale is composed of the ideas that resulted in our definitions and methodology (presented throughout the paper). Namely, (1) the characterization of leakage as a function of different random variables whose usage depends on the algorithm parameters (2) the usage of strict probability bounds as a means for key evaluation (3) the reducability of the key evaluation part to simple models that are optimizable efficiently, (4) usage of features of the leakage as a means to leverage different independent signal sources on the key.

These points are furthered elaborated throughout the paper:

1. The criteria for usable 'feature estimators' as listed in Sec. 2.2.
2. The adversary attack procedure steps in Sec. 4.
3. Usage of $\Sigma^W, \Sigma^L, \Sigma^Z$ as evaluation metrics (such as done in Appendix A.5). To rate the keys by comparing the estimated quantities that bound the posterior probability (with some probability).

In the rest of this section, we elaborate on these ideas and exemplify them when discussing the simplest feature estimation attack on $A_{\mathbf{k}}(;\mathbf{r}) = \mathbf{k} \oplus \mathbf{r}$ where $\mathbf{k}$ is the secret key, and $\mathbf{r}$ is an unknown randomly generated word.

The critical and implicit (conventional) assumption we shed is the symmetry of the leakage. That is, we show that the amount of leakage each bit contributes to the trace *varies*. Although it might seem like a minor modification, it actually results in increasing the limit of the information bits learned from $\leq \log_2(n_k + 1)$ (the weight of the word) to n_k, especially when considering how the leakage variations change across different leakage time samples.

The second assumption we shed is the additivity of noise (i.e., the assumption that $L = f(\mathbf{k}) + N$ with $N \perp \mathbf{k}$). *We assume our leakage model to be a sum of different random variables that are switched on or off based on the state of every input and output of every logic gate in the calculation.* For example, $\mathbf{k} \oplus \mathbf{r}$ has four possible states (one per possible input), influencing three leakage variables (two inputs, one output, at least in the way we characterize it) per key bit, per time sample.

3.2 Leakage modeling in FEbA

We denote by $Z_{t,i}^{x,b}$ a random variables that measures how much leakage the x'th interface[5] of the i'th logic gate of the operation consumes at time sample t when the bit value assigned to the interface is b. For clarity, the interfaces are denoted as $x \in \{k, r, k \oplus r\}$ for key input, random input, and output, respectively.

To estimate the leakage, we assume that leakage is additive and sum all the Z's conditioned on when the interface's value is realized (as well as an 'always on' affine variable):

$$\hat{L}_t = L'_t + \sum_i k_i \cdot Z_{t,i}^{k,k_i} + \sum_i r_i \cdot Z_{t,i}^{r,r_i} + \sum_i (r_i \oplus k_i) \cdot Z_{t,i}^{kr,k_i \oplus r_i} \tag{9}$$

Notice that *we do not assume anything about the distribution, or dependence of the different Z's.* This type of generic leakage characterization is very important in high-order attacks, such as Sec. 6.2 and Sec. 6.4, though when using the mean estimator, such a leakage model is not required (but is helpful for the conceptualization).

There are 2^{n_k} keys, and each key is associated with a different expected estimation $f_{\mathbf{k}}^{\star} = E(L_{\mathbf{k}}^{\star})$. We approximate the distribution of $f_{\mathbf{k}}^{\star}$ over the different $\mathbf{k}$'s as a Gaussian (approximation used for visualization and trace estimation purposes only, since it is a sum of random variables $k_i \sim Ber(0.5)$).

[5] Interface = one of the inputs or outputs of a logic gate. For example, a 1-bit XOR gate has 3 interfaces; 2 inputs and 1 outputs

3.3 The Single Sample Case (l = 1)

The top figure in Fig. 4a represents a distribution of the expected leakages for different keys. It represents a continuous approximation of the distribution $f^\star_{\mathbf{k}}$ (distribution of expected estimations, solid line) and $\hat{w}_{\mathbf{k}}$ (profiled model, dashed line) over $\mathbf{k}$, where the leakage vector is one-dimensional (one time sample, $l = 1$).

The lower part of Fig. 4a shows the distribution of averaging n traces from the leakage for a given key. The figure should be read as follows: Let us pick a random key, $\mathbf{k}^\star$. In Fig. 4a, the blue and red crosses represent the estimated ($\hat{f}_{\mathbf{k}^\star}$) and real ($f^\star_{\mathbf{k}^\star}$) expected leakage values associated with $\mathbf{k}^\star$. An adversary, who does not know $\mathbf{k}^\star$, samples n traces from the target $\mathbf{X} \sim (L^\star_{\mathbf{k}^\star})^n$ and averages them to obtain $\mu(\mathbf{X})$, which is $w^\star_{\mathbf{k}^\star}$ (depicted with a green line in Fig. 4a). For enough traces, $\mu(\mathbf{X}) = w^\star_{\mathbf{k}^\star}$ will approach $f^\star_{\mathbf{k}^\star}$; the distance between them becomes statistically smaller as the trace count increases - the bottom distribution becomes narrower. For a good enough model, $\hat{w}_{\mathbf{k}^\star}$ is close to $f^\star_{\mathbf{k}^\star}$ (enough profiling traces $\hat{f}_{\mathbf{k}^\star} \approx \hat{w}_{\mathbf{k}^\star}$, good enough modeling $f^\star_{\mathbf{k}^\star} \approx \hat{f}_{\mathbf{k}^\star}$).

Due to the nature of the attack, the amount of possible keys, and additional causes influenced by the difference between reality and theory, it may be difficult to pinpoint the exact key using this method. As such, we allow a number of K key guesses, assuming the adversary can confirm a key guess.

Key Space Reduction: The area of the light gray shaded area (bound by the blue lines) in the top subfigure in Fig. 4a represents the number of keys needed to be guessed (in practice) before reaching the key ($\mathbf{k}^\star$), where the guessing metric is the distance function. Guessing by this metric, in the limit, is the same as guessing based on the upper bound we achieved in the proof in Sec. 2.1 of our posterior probability $P_{\mathbf{k}|\mathbf{X}}$. $n(K, \delta)$ describes the number of traces that $\mu(\mathbf{X})$ would cause the light gray shaded area to contain at most K keys w.p. $1 - \delta$.

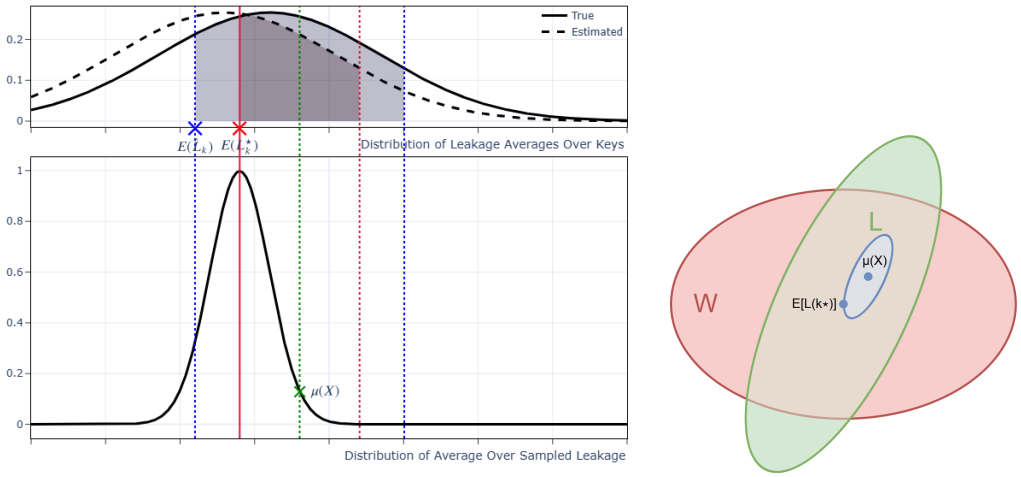
3.4 The Multidimensional Case ($l > 1$)

Feature estimation based attacks show clear performance advantages in the multidimensional case: For every time sample added to the leakage, there is a new leakage 'features' (per individual bit of the key) that gives us information which can be independent of the previous leakage time sample. Observe Fig. 4b. We see two types of distributions drawn; W is the distribution of the expected leakages $f^\star_{\mathbf{k}}$ over the keys, and L is the distribution of the leakages obtained from $A_{\mathbf{k}^\star}(\mathbf{p}; \mathbf{r})$, $\hat{w}_{\mathbf{k}}$. Both are approximated as a Gaussian due to the central limit theorem. As we sample more traces, the distribution of L becomes less variant, while W stays the same size.

Given infinitely many samples from $L^\star_{\mathbf{k}^\star}$, their average $\mu(\mathbf{X})$ approaches $E(L^\star_{\mathbf{k}^\star})$. For a finite number of traces, there is an error between the two, which requires us to (a) sample enough traces so that the key whose expected leakage is closest to $\mu(\mathbf{X})$ is $\mathbf{k}^\star$, or; (b) guess keys based on (some) metric to $\mu(\mathbf{X})$. We guess the key *based on the metric defined by L's variance*, since that is the quantity that characterizes the noise in $\mu(\mathbf{X})$'s value (assuming i.i.d.). The number of keys needed to guess before reaching the true key is the number of keys whose expected leakage is closer to the sampled values *based on the noise's metric*. In Fig. 4b, this area is depicted in blue.
Given all this, our condition for a successful attack is:

$$\Pr_{\mathbf{X} \sim L_{\mathbf{k}^\star}} \left[\left| \{\mathbf{k} : P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^\star|\mathbf{X}}\} \right| \leq K \right] \geq 1 - \delta \tag{10}$$

We will reach a probabilistic bound on n. This bound takes into account both assumptions and relaxations, which may cause the bound to be incorrect for specific cases (as shown in Appendix A.4.1), or not tight in other cases.



(a) Expected leakage vs. keys and average leakage over samples for $k^\star$ ($l = 1$).

(b) Multidimensional depiction of the leakage for $l = 2$.

Figure 4: Leakage behavior under masking.

3.5 Proof of Concept – The Mean Estimator ($\theta = \mu$)

In this section, $n_r = n_k, n_p = 0$, and we measure in software implementation leakage from running the operation $out = \mathbf{k} \oplus \mathbf{r}$. We conduct the attack following the rationale with the mean estimator $\theta = \mu$.

As we showed before, the leakage model takes the form of Eq. (9), and we estimate the feature $\mu(\hat{L})$. As such we profile the leakages with linear regression $M' \cdot (1, \mathbf{k_i}, \mathbf{r_i}, \mathbf{k_i} \oplus \mathbf{r_i}) \leftarrow \ell_i$ given $\mathbf{k_i}, \mathbf{r_i} \sim \text{Ber}[0.5]^{n_k}$ ($\mathbf{k_i} \wedge \mathbf{r_i}$ is linearly dependent on $(1, \mathbf{k_i}, \mathbf{r_i}, \mathbf{k_i} \oplus \mathbf{r_i})$, so we do not have to consider it). We notate $M' = (b||M_0||M_1||M_2)$ where $b \in \mathbb{R}^l$ and $M_i \in \mathbb{R}^{l \times n_k}$. Afterwards, we create a matrix that estimates the feature for a key $\mathbf{k}$, $M = (b + M_1/2 \cdot \vec{1} + M_2/2 \cdot \vec{1}||M)$, which originates from the fact that both r_i and $k_i \oplus r_i$ are constant quantities on average (over $\mathbf{r}$). Conversely, we may ignore both quantities when profiling the model if we wish (for the price of slower model convergence).

Fig. 5 depicts how the leakage looks over time. We can see a notable phenomenon, though, where the leakage on the key is continuously re-invoked even after the computation on the key is over. This is hypothesized to be caused by a hardware-dependency where the value of $\mathbf{k}$ is re-invoked down the processor's pipeline. This, of course, makes our attack potentially easier depending on the device's internal micro-architecture. As such, we distinguish between the two cases where the XOR's inputs' order is swapped (more secure) or not (more vulnerable), meaning that the $\mathbf{k}$'s leakage would not persist and would look like $\mathbf{r}$'s leakage instead (on the same figure). Using such a profiling taken over an arbitrarily large number of profiling traces (for the sake of proving the concept), we plot the (approximate) required number of attack traces needed to bring the guessing entropy below 1 on average in Fig. 6b.

Fig. 6a presents the approximate amount of information gained from our method on the key as a function of the number of attack traces for $n_k = 16$. We do so by performing an attack on 500 keys, obtaining a list of each key's $W_{\mathbf{k}^\star}$ and estimating the mutual information (MI)

$$MI_1(\chi, L^n) = n_k - H(\chi|L^n) \gtrsim n_k - \sum_{m=0}^{n_k-1} p_m(m - \log_2 p_m) \tag{11}$$

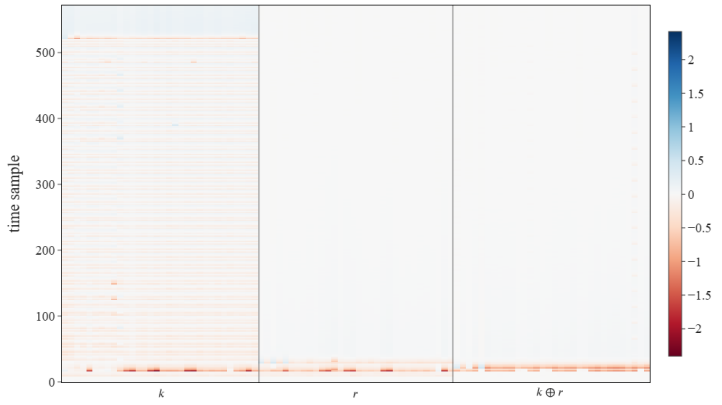


Figure 5: Leakage time samples versus the 32-bit trained features. The X-axis is partitioned into three consecutive regions corresponding to the trained features $\mathbf{k}$, $\mathbf{r}$, $\mathbf{k} \oplus \mathbf{r}$, while the Y-axis represents leakage time samples. Each colored cell represents the estimate value of $E(Z_{t,i}^{x,1} - Z_{t,i}^{x,0} | \mathbf{r}, \mathbf{k})$, where t is the vertical axis, and $(x, i) \in \{\mathbf{k}, \mathbf{r}, \mathbf{k} \oplus \mathbf{r}\} \times \mathbb{Z}_{32}$ is the horizontal one.

where $p_m = |\{\mathbf{k} : 2^m \leq W_{\mathbf{k}^\star} < 2^{m+1}\}|/2^{n_k}$ and H is the entropy. Additionally, we plot a less conservative estimate

$$MI_2(\chi, L^n) \gtrsim n_k + \sum_{m=1}^{2^{n_k}} q_m \log_2 q_m \tag{12}$$

where $q_m = |\{\mathbf{k} : W_{\mathbf{k}^\star} = m\}|/2^{n_k}$, which represents an approximate "upper bound". Notice that these equations use only the mean of the traces to extract the key. In other words, there might be additional information that we do not use (see Sec. 6.2).

To validate our theoretical results, we create two more plots. (1) rough information theoretic bound obtained via our K, n, δ bound from Appendix A.5.2

$$MI(\chi; L^n) \gtrsim -h_2(u_{\arg\min}) + \log(2^{n_k} / \binom{n_k}{n_o}) - \min_{u > u_{min}} F_{\chi_l^2(nR^2)}(nu) \log_2(F_{\chi_l^2(;\Lambda)}(u)) \tag{13}$$

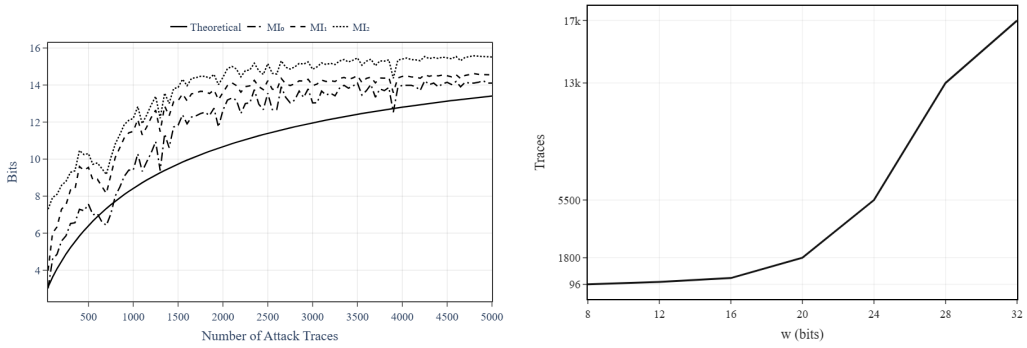
with $n_o = \frac{1}{2^{n_k}} \sum_{n_k \geq s \geq 0} \binom{n_k}{s}^2$, $h_2(p) = -\sum_{x \in \{p, 1-p\}} x \log_2 x$ and $\max_k \|E(L_{\mathbf{k}}) - E(\hat{L}_{\mathbf{k}})\|_\Sigma^2 = 0.2$ (empirical estimation). and (2) an empirical estimation matching the theoretical bound,

$$MI_0(\chi, L^n) = n_k - \min_{m=1}^{2^{n_k}-1} [h_2(v_m) + \nu_m \log_2(m) + (1 - \nu_m) \log_2(2^{n_k} - m)] \tag{14}$$

where $\nu_m = |\{\mathbf{k} : W_{\mathbf{k}^\star} \leq m\}|/2^{n_k}, h_2(x) = x \log_2 x + (1 - x) \log_2(1 - x)$.

As we can see, the mathematical analysis is validated via the results, where we can predict the required attack traces for a given feature given profiling leakages. For some perspective on how the mutual information relates to the adversary's accuracy, for 5000 attack traces more than 90% of the keys are correctly guessed with $W_{\mathbf{k}^\star} = 1$.

The results above, and throughout the rest of this paper, are performed for an arbitrarily large number of profiling traces (usually $\geq 2M$) in order to present the technique more clearly as a proof of concept. As seen in Fig. 7, the attack works as intended with a lower number of traces.



(a) Mutual information as a function of the number of attack traces, using the estimated MI from Eq. (14, 11, 12) with 500 keys and the theoretical MI from Eq. (13).

(b) Minimum number of traces for which the average guessing entropy over 10 randomly chosen keys is ≤ 1 (non-swapped XOR).

Figure 6: Mutual information and guessing entropy as functions of the number of attack traces.

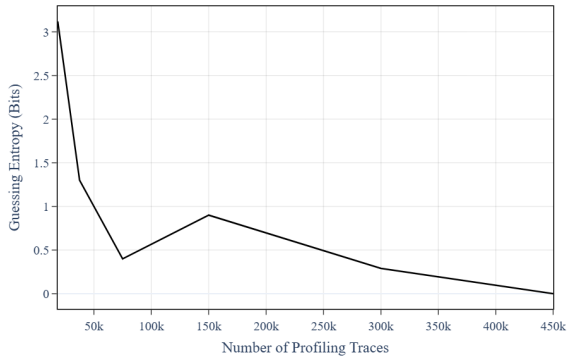


Figure 7: Average guessing entropy over 10 random keys as a function of profiling traces with $n = 50{,}000$ attack traces for $n_k = 16$.

4 FEbA – Attacks Procedure

4.1 Attack Setup and Steps

In general, FEbAs are conducted by following the common SCA procedure, but here three additional steps are added. These steps enable a more flexible attack, while keeping things conceptually simple.

These steps are abstractly illustrated in Fig. 2. Fig. 8 describes the steps of feature estimation based attacks in more details. The steps are the following:

1. **Leakage model selection ($\hat{L}$):** A central idea in FEbAs is to assume nothing about the distribution of an operation, besides how it consists of sub-variables as a function of the operation's inputs and output[6] depending on the target. This is done by creating atomic leakage variables (down to the required fidelity) that describe the mechanisms of the hardware, which add up to the leakage random variable as a function of the intermediate values. The main, but key difference is that in regular

[6]I.e., we don't have the actual probability mass function of the leakage

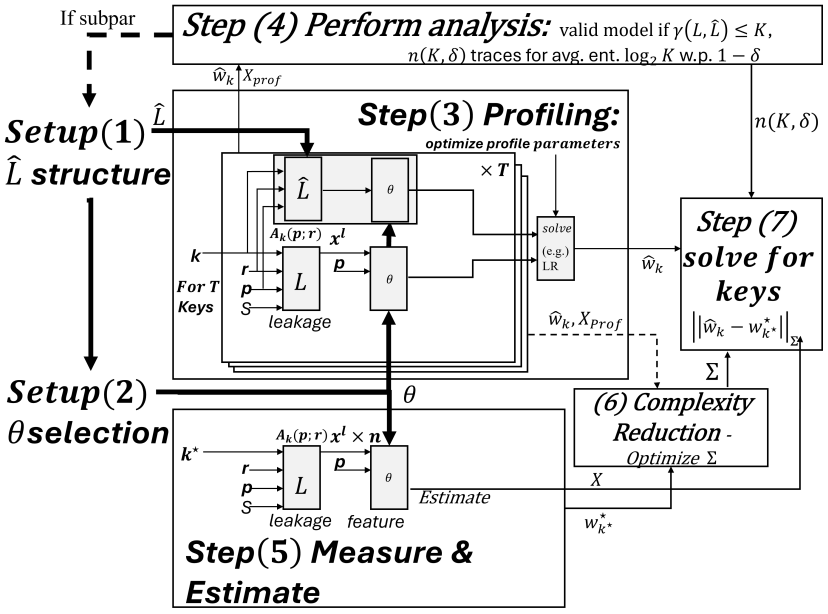


Figure 8: Feature Estimation-Based Attack - Detailed High-Level Workflow

Table 2: Model-, Profile-, and Feature-Estimation -based Attacks procedures

	Setup	Model Based Attacks	Profile Based Attacks	Feature Estimation Attacks
(0)	Probability model (implicit)	$P_{\mathbf{X}\|\mathbf{k},\mathbf{X}_p}$ increases with $\theta_{\mathbf{k}}(\mathbf{X})$	Heuristically characterize the prior distribution $P_{\mathbf{X}\|\mathbf{k},\mathbf{X}_p} \sim (\hat{L}_{\mathbf{k},\mathbf{p_i}})^n$	$P_{\mathbf{X}\|\mathbf{k},\mathbf{X}_p} \le l/\|\theta_{\mathbf{k}}(\mathbf{X}) - f^n_{\mathbf{k}}\|^2_{\Sigma^L}$
(1)	Leakage model selection ($\hat{L}$)	—	Choose g s.t. $g(\mathbf{k},\mathbf{p}) = g(\mathbf{k}',\mathbf{p}') \Rightarrow \hat{L}_{\mathbf{k},\mathbf{p}} \equiv \hat{L}_{\mathbf{k}',\mathbf{p}'}$. Assume $\hat{L}$'s distribution.	Choose a suitable model structure $\hat{L}$ for the target
(2)	Feature selection (θ)	Select a leakage metric.	—	Select the exploitative feature θ to be used.
	Step			
(3)	Profiling	Optional: Profile model from profiling traces	Profile model from profiling traces	Profile $\hat{f}_{\mathbf{k}}$ from profiling traces
(4)	Statistical analysis	—	—	Statistical analysis for the enumeration complexity. Iterate to step 2 if required.
(5)	Attack on target App.	+	+	+
(6)	Complexity Reduction	—	—	Optimize covariance Matrix
(7)	key evaluation	Iterate over sub-keys	Iterate over sub-keys	Solve the optimization problem
	Comparison	CPA	Template	FEbA (mean estimator)
	Use case	Target is splittable, leakage acts nicely, no profiling traces.	Minimum number of attack traces required, low key/ splitting complexity attacks feasible	Targeting complex targets (no plaintext, complex data dependency), Computational efficiency is required (large key/large dataset), analytical evaluation is required.
	Complexity	$\approx \mathcal{O}(2^{n_k} \cdot n \cdot l)$	$\mathcal{O}(Tl^2 + Cl^3 \approx + n \cdot 2^{n_k} \ell^2)$	$\approx \mathcal{O}(Tn_k l + Wn_k l + nl^2)$

profiled attacks, we characterize a conditional probability function that calculates the likelihood of each intermediate given the leakage, given an assumption about the leakage's distribution. In feature estimation attacks, we characterize features of the leakage without any assumption on the actual distribution, and use that to bind the said probability from above.

2. **Feature selection (θ):** Feature estimation attacks estimate, as the name implies, a

feature (characteristic) of the leakage. We may choose which feature (e.g., statistical moments) of the model to estimate, given the assumed leakage structure model $\hat{L}$. We attempt to maximize the SNR of the feature, and are limited by computational complexity. We should pick our feature in a way that $\hat{f}_{\mathbf{k}}$ can be accurately profiled.

3. **Profiling ($\hat{w}_{\mathbf{k}}$):** Given traces on many different $\mathbf{k}$'s (on the figure denoted T) we profile a model $\hat{w}_{\mathbf{k}}$ (which is dependent on $\hat{L}, \theta$)
   It is not needed to sample traces for every single $\mathbf{k}$, for example, in 6.6, we use $250,000$ traces to model a key space of size 2^{32} with success).
   For a mean estimator FEbA, the profiling trace complexity required is $T \approx \Omega\left(\frac{l}{F^{-1}_{\chi^2_l(;\Lambda)}(1/\binom{n_k}{n_o})}\right)$ (derived from the expression in Appendix A.5.3 and linear regression convergence considerations, assuming the leakage is additive in its component). For a somewhat non-degenerate Λ, this bound may even be linear in n_k (unlikely in practice). For a mean estimator FEbA The time complexity of the model's processing is $\approx \mathcal{O}\left(l \cdot T \cdot n_k\right)$ (assuming $n_k < l < T$).
4. **Statistical analysis ($n(K, \delta)$):** Use analysis calculations such as the one described in Appendix A.5 to create a complexity bound given the model assumptions and feature. We use the bound on the posterior probabilities to rate the keys - the tighter this bound can be made, the better the attack performs. We can also use the complexity bound to approximate the number of traces we need for an attack. If subpar performance is obtained, we may reiterate through the steps above *using the same set of profiling traces.*
5. **Attack on target Application ($w^{\star}_{\mathbf{k}^{\star}}$):** Take n attack traces and evaluate the feature of these traces. For a mean estimator FEbA, the time complexity of the attack traces' processing is $\mathcal{O}(l \cdot n)$.
6. **Complexity Reduction (Σ):** In this step, we take the profiling traces and use bootstrapping methodology to estimate statistical objects to improve the trace complexity of the attack. This can either be done using the conclusions obtained from step (4), using better analysis (e.g., assuming traces are a stationary variable insatead of i.i.d) or heuristics such as explained in Appendix A.2. In this paper, we stick to using Σ^L. For the mean estimator, the complexity is $\mathcal{O}(T \cdot l^2)$.
   An example for how future analysis would be done, is using variance analysis (As defined in Appendix A.2), which would result in a normalization matrix being added to the final optimization problem.
7. **key evaluation($\hat{\mathbf{k}}$):** Given the processed attack traces, we find the best-fitting set of keys using an optimization problem. For a mean estimator FEbA, the time complexity of this step is $\mathcal{O}\left(W \cdot n_k \cdot l\right)$ when using a K-best tree search algorithm with a beam search of W [GN06] (the non-heuristic approach has exponential complexity in n_k).

Notice that the ordering of this list is not absolute, and it is encouraged to iterate over the steps given conclusions from the experiments. For example, in Sec. 6.2 we use the attack traces in the profiling stage to obtain better results.

We get multiple degrees of freedom in different aspects of the attacks: The model structure, leveraged feature, statistics optimization, and the analysis. We tweak those iteratively (and perhaps independently) based on the enumeration bound (or more simply, the key evaluation results). Examples are present in Appendix A.7

5 Experiment Methodology

For simplicity and reproducibility, the hardware component used to gather measurement traces is the 'ARM STM32F303 (32-bit)' processor, with a (builtin) ChipWhisperer 10-bit oscilloscope ADC (that measures current leakages), clock frequency of $F = 7.385MHz$,

sampling rate of $4{\cdot}F$, and a built in trigger that aligns the traces. Discussions about the SNR and weight variability of the chipwhisperer are present in Sec. 5.1. Our STM32F303 target with the ChipWhisperer measurement framework is a standard and reproducible platform for embedded side-channel evaluation. The platform was introduced as an open-source platform for hardware embedded security research [OC14]. Several advanced and recent side-channel studies use this platform, making it comparable and reproducible [JWMB23, RYJ+25, WBD23].

Hardware factors such as temperature and power source instability cause S to change in time. In practice, if we first sample the model and only then record the attack traces, we would find it difficult to guess the correct key due to a discrepancy between the two. (Example for trend over time may be seen in Appendix A.10.1). To combat this effect, we take traces in a randomized interleaved manner, as typically performed in the field.

The results below were obtained for the various attacks with stalling implemented (before the targeted algorithm starts) to avoid leveraging remanent communication leakages. Note that for operations where $n_k < 32$, we emulate the results by using a truncated 32-bit register. Alg. 1 describes the trace recording procedure (measurement algorithm). In Alg. 1, MACRO is the targeted operation(s), and padding is done to remove residual dependency in leakages of consequent measurements.

These measures are intended to ensure that the reported key-recovery performance is attributable to leakage produced during the targeted MACRO itself, rather than to deterministic inter-trace carry-over effects (e.g., remanent I/O or communication activity) or temporal drift between profiling and attack acquisitions.

Algorithm 1 Measurement Algorithm

1: **input:** $k[n_k], p[n_p], r[n_r]$
2: **output:** *out*
3: $z \leftarrow 0$
4: long stall
5: **trigger high** // start measurement
6: MACRO(out, p, k, r)
7: while$(z < 100)$ $z = z + 1$ // generic padding
8: **trigger low**

5.1 The SNR of our Environment

As defined in [MOP07], the SNR is the ratio of the signal (variance over expected leakages as a function of the key) and the noise (expected variance). We can estimate it via our model and attack traces:

$$SNR_t = VAR_{\mathbf{k}}(E(L_{\mathbf{k},t}))/VAR(L_{\mathbf{k}^\star}) \approx 0.25 \sum_{i=1}^{n_k} (M_{t,i})^2 / E_{\mathbf{k}}(\Sigma^L_{t,t})$$

where $L_{\mathbf{k},t}$ is the leakage in the t'th time sample and M is the leakage matrix model. The highest SNR time samples are present during the instruction's execution, with the remanent leakages decaying afterwards. There are ~ 10 time samples with SNR in the range $(0.1, 1)$, ~ 5 time samples with SNR in the range $(0.01, 0.1)$, ~ 100 time samples with SNR in the range $(10^{-3}, 0.01)$ and the rest (up to a few hundred) in the range $(10^{-5}, 10^{-3})$. The points with the highest SNR tend to be the points whose bit leakage variety (defined as $VAR_{\mathbf{k}}(E(L_{\mathbf{k},t}) | \,\mathrm{HW}(\mathbf{k}) = n_o) \approx n_o \cdot n_z VAR_i(M_{t,i})/n_k$) is the highest (i.e., the best POIs are the points whose leakage differs the most from the regular hamming weight model). In fact, the correlation coefficient between the SNR and weight variability was calculated to be 0.956 (for one randomly chosen key).

Table 3: Summary of the evaluated targets and the properties demonstrated for each target. The evaluation metrics are defined in Sec. 5.1.1.

Target (Scenario)	Setup	Feat. Type	#Keys	Attack traces	Results	Ref.	Details
XOR $n_k = 32$ **(Fig. 1-Sn2)** unknown r sharing	non-swapped XOR[3.5]	**Mean**	50	$30,000$	$AGE = 0.9$	Sec. 6.1	Tab. 6
XOR $n_k = 16$ **(Fig. 1-Sn2)** unknown r sharing	swapped XOR[3.5]	**Cov.**	50	$50,000$	$AGE = 5.7$	Sec. 6.2	Tab. 7
XOR $n_k = 32$ **(Fig. 1-Sn1)** known p sharing	swapped XOR[3.5]	**Mean**	50	$1,000$	$AE = 0.064$	Sec. 6.3	Tab. 8
XOR $n_k = 16$ **(Fig. 1-Sn4)** unknown r mask.	swapped XOR[3.5]	**Cov.**	10	$100,000$	$AGE = 1.87$	Sec. 6.4	Tab. 9
XOR $n_k = 128$ **(Fig. 1-Sn2)** unknown r sharing	Swapped XOR[3.5]	**Mean**	50	$50,000$	$AGE = 8.99$ **w.p.** 56% **w/** $K = 1M$	Fig. 6.5.1	Tab. 10
AES MIX-COL. 32-bit (no access to p, r)		**Mean**	10	100	$AGE = 4.18$	Fig. 6.6	Tab. 11

5.1.1 Effectiveness and Tradeoffs

We refer to four measures of interest when evaluating the effectiveness of feature estimations:

- **The minimal guessing entropy (MGE).** of the key is defined as the log of the minimum number of guesses needed to reach the key: $\log_2 W_{\mathbf{k}} = \log_2 \left|\{\mathbf{k} : P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^*|\mathbf{X}}\}\right|$.
- **The average guessing entropy (AGE).** is the expected minimal guessing entropy $H(W) = E(\log W_{\mathbf{k}})$. In an evaluation, this refers to the quantity $\frac{1}{\#\text{Keys}} \sum_i \log_2(W_i)$ where W_i is the minimal guessing entropy of a particular targeted key.
- **The average error (AE).** The average error is defined as the expected Hamming distance between the true key and the most likely key candidate.
- **Enumeration bound** $n(K, \delta)$**.** describes the minimum number of traces we need to achieve a minimum guessing entropy of $\log_2 K$ with probability $1 - \delta$. Where, K is defined as the Number of allowed key guesses for the adversary

6 Attack Scenarios, Experiments and Results

For clarity and ease of navigation, we begin by presenting a high-level overview of the experiments as summarized in Tab. 3. The first column of the table describes the target algorithm, and when relevant, the specific scenario (abbreviated **Sn**) as illustrated in Fig. 1. The next four columns specify the setup, feature type, the number of attacked keys (#Keys) the experiment was performed on to gather statistics (using the same profiling/ bias leakages) and the number of traces n used for attack. The last three columns provide the key results and references to the section and tables where the experiment and its results are described in details. The table lists several masked scenarios of refreshing (with

either a known or unknown intermediate) and masked XOR with either a random masked value with a varying number of bits in an operation.

Sec. 6.5 and Sec. 6.6 expand to full algorithms using unrefined approaches. Sec. 6.7 discusses immediate relevance and applicability to ASCON.

For the attack on XOR refreshing operation *with an unknown mask* r_1 *and an unknown operator* r_2 with a key of size 16, we find out that even with no access to any parameter in such a noisy enviornment, refreshing is not impervious to information extraction - a scenario that up until now would have been deemed secure, a successful attack reducing key entropy to $AE = \{3.05,\ 1.87\}$ with as little as $n = \{25K,\ 100K\}$ attack traces, respectively.

The success metrics provided in the table and throughout the result sections are defined in Sec. 5.1.1. For some attacks, only the top K key guesses are rated. We are unable to evaluate the position of the true key if $W_i > K$. In such a case, if a portion p of the attacked keys achieved $W_i \leq K$ with average entropy X, we say that the guessing entropy is X with probability p for K guesses, or in short, the average guessing entropy is X w.p. p w/ K.

6.0.1 Comparison to alternatives

We compare FEbA against prior arts in three categories - known plaintext, unknown plaintext, and masked unknown plaintext. The comparison is summarized using Tab. 5, referencing papers attacking a target as similar to ours as possible.

Notice that within every category, feature estimation attacks can dominate the closest similar candidate we have found. For scenario one, we have chosen Jahandideh et al. [JWMB23] as it is a recent paper that targets 32 bit operations. For scenario two, we have found no paper that solely utilizes XOR leakage for a side channel attack, so we chose DL-BSCA [RYJ+25] as an alternative (leaks like scenario 2 plus SBOX leakage). Finally, for scenario 4, we have found no paper that can extract any information about the key solely from masked XOR operations *with no access to intermediate values*. In depth:

The template attack listed in the table uses the same (profiling and attack) traces as the corresponding FEbA traces. It uses 30 points of interest, as a trade-off between prolonged runtime and accuracy. Template attacks, in their base form, require too many resources for $n_k = 32$. As such, we perform two template attacks with $n_k = 16$, each time for half the key. We note that even when FEbA uses trace lengths larger by 20 times than the template attack for $n_k = 16$-bits, or 32-bits, the evaluation period for FEbA is shorter. Template attacks are *not computationally tractable* for scenario four, especially when $n_k = 32$.

Table 4: Average Error (AE) of a template adversary on Scenario 1 from Fig. 1

Configuration	**# Attack Traces** n					
	25	**50**	**100**	**250**	**500**	**1000**
16-bit (T=60K, #Keys=50)	AE= 0.02	0	0	0	0	0
32-bit (T=60K, #Keys=50)	AE= 12.68	10.04	7.32	2.00	0.36	0.04

Comparing our attack to the methodology used in Jahandideh et al, we find that our attack performs two orders of magnitude better. This is due to Jahandideh's methodology using only the sign of the learned linear regression parameters (which might not agree with the key's value even at the limit) instead of all the available information. Additionally, the mentioned paper does not use the entire breadth of the trace.
Compared to FEbA, we find that DL-BSCA is limited. When a non-linear operation such as *Sbox* is not present, the (assumed HW) leakage wouldn't manifest 2^{nd} order joint distributions; this is where techniques such as DL-BSCA would struggle.
Finally, as mentioned above, the attack targeting the masked implementation (last scenario

Table 5: Comparison of FEbA with prior side-channel attacks on XOR-based targets. A dash ("–") indicates that the corresponding attack is not applicable to the considered scenario. n_k denotes the key length (in bits), and n denotes the number of attack traces. AGE refers to average guessing entropy (bits) and AE denotes average error (Hamming distance). All FEbA results use a linear regression model profiled on a clone device.

Scenario	Method	Target Op.	n_k	Attack n	Result
Known plaintext (Fig. 1-Sn1)	Jahandideh et al.* [JWMB23]	$k \oplus p$	32	1 000	AE≈6
	Jahandideh et al. [JWMB23]	$k \oplus p$	32	10 000	AE≈2
	FEbA* [6.3]	$k \oplus p$	32	1 000	**AE=.06**
	FEbA [6.3]	$k \oplus p$	16	250	**AE=.48**
	HW Template	$k \oplus p$	32	1 000	**AE=.04**
	HW Template	$k \oplus p$	16	25	**AE=.00**
Unknown plaintext, no mask (Fig. 1-Sn2)	DL-BSCA* [RYJ+25]	$\text{Sbox}(k \oplus r)$	8	5 000†	AGE≤ 1
	DL-BSCA [RYJ+25]	$k \oplus r$	any	any	AGE≈ n_k‡
	FEbA [Fig. 6b]	$k \oplus r$	8	100	AGE≤ 1
	FEbA [6.1]	$k \oplus r$	32	30 000	AGE=0.9
	Template	$k \oplus r$	≥ 16	-	Impractical
Unknown plaintext, masked ($d=2$) (Fig. 1-Sn4)	DL-BSCA	$(k \oplus r_1) \oplus r_2$	-	-	Impossible
	ASCA	and $r_1 \oplus r_2$	-	-	Impossible
	FEbA [6.4]	$(k \oplus r_1) \oplus r_2$ and $r_1 \oplus r_2$	16	100 000	**AGE=1.87**

* Jahandideh et al. uses STM32F405. DL-BSCA uses STM32F4 Cortex-M4. Our paper samples the leakage from the operations 'ldrd r3, r7, [r5]; eors r7, r3'.
† DL-BSCA uses 80 000 additional training traces with the correct fixed key. In that paper, AGE is averaged across different trace sets for the same key.
‡ Under the symmetric HW model, Pr[HW(r), HW($k \oplus r$)] is identical for all keys of equal Hamming weight, at most $\log(n_k + 1)$ bits of information may be extracted on the key.

in Tab. 5), we find that no other prior art in the literature has managed to extract any amount of information on a similar implementation to the best of our knowledge.

6.0.2 Comparison to algebraic side channel attacks

ASCA [RSVC09] falls under the category of attacks that reduce its leakages to an optimization problem whose solution is hopefully the key, given knowledge of the algorithm and a simple profiling. Feature estimation-based attacks also fall under this category.

While FEbA shares conceptual similarities with ASCA in that aspect, ASCA differs in its requirements, analysis, and more. ASCA attacks on a trace-by-trace basis, because it requires capturing information from a large number of operations and considers a large set of constraints, usually targeting a single trace. ASCA also requires a high enough SNR to bring the expected maximal hamming weight error from a trace to a low enough value, which would still require the use of an error-tolerant solver.

FEbA operates on the statistical features of a distribution, which allows it to 'denoise' the traces, thereby leveraging them while solving for the key using a discrete minimization problem instead of a discrete satisfiability problem. The use of leakage features makes the attack intrinsically error-solvent. This presents a potential improvement beyond ASCA, where we may leverage every single leakage time-sample for the attack, even multiple different leakages per operation, instead of those that only leak like the (e.g.,) Hamming weight of an operation.

As such, while there are similarities between these attacks, they are not directly comparable. Comparison to other adversaries may be found in Appendix A.1

6.1 FEbA - Proof of Concept

We attack the operation $MACRO = \mathbf{k} \oplus \mathbf{r}$ using the mean estimator μ as our feature, and the same assumptions used in Sec. 3.5.

We devise the following pseudo-algorithm as per the rationale:

---

1. Sample a profiling set $\{\mathbf{P_i}, \mathbf{k_i}\}$
2. Create the model via linear regression $M \cdot (1, \mathbf{k_i}) \leftarrow \mathbf{P_i}$.
3. Estimate the number of traces $n(K, \delta)$ needed.
4. Sample an attack set $\mathbf{A_i}$ on $\mathbf{k}^\star$.
5. Calculate the feature $\theta(\mathbf{A}) = E_i(\mathbf{A}_i)$
6. Calculate the attack covariances $\Sigma^L = COV_i(\mathbf{A}_i)$.
7. Find the fitting keys $\arg\min_{\mathbf{k}} \|M \cdot (1, \mathbf{k}) - \theta(\mathbf{A})\|_{\Sigma_L}$.

Table 6: Average guessing entropy of adversaries on Scenario 2 from Fig. 1 when using the mean estimator.

Configuration	$n = 30,000$	$n = 60,000$	$n = 120,000$
16-bit, XOR not swapped (T=2.1M[a], #Keys=50[&])	0	0	0
16-bit, XOR swapped (T=3M, #Keys=50)	4.72	4.4	3.92
32-bit, XOR not swapped (T=5.2M, #Keys=50)	0.9	0.8	0.8
32-bit, XOR swapped (T=6M, #Keys=50)	15.94 w.p. 38% w/ 10M	16.14[b] w.p. 46% w/ 10M	16.27 w.p. 52% w/ 10M

[a] Notice that this amount of profiling traces is not necessarily required for the attack to work. An arbitrarily large set of profiling traces was taken as insurance, since this work serves as a proof of concept for FEbAs.

[b] The average entropy increases with more traces since the new keys that fall within the threshold of 10M will likely be in the upper range of the guesses.

From now on, we neglect to mention $n(K, \delta)$ assuming it is built into the attack. The results in Tab. 6 clearly displays that FEbAs may depend on seemingly mundane implementation details, with a difference of $4 - 10$ bits in the guessing entropy obtained by just switching the order of the XOR's arguments. Clearly, this issue is more constraining for large n_k's, as for the 16-bit case, both the best and worst cases (with swapped registers or not) are quite vulnerable.

The XOR 'swapped'/ 'unswapped' experiments, as done above, display the worst-case and best-case scenarios for the attack, when combined with the code in Alg. 1. FEbAs are shown to be dependent on the processor's RTL / underlying hardware beyond normal attacks.

Although XOR is mathematically commutative, the dependence of its input operand order on the result may be attributed to the fact that the implementation may not be symmetric in how operands are sourced, routed, and stored inside the hardware (we hypothesize that the first XOR operand is saved in the processor pipeline and is re-invoked in hidden/ unselected ALU operations).

As a final note for this subsection, since the SNR of an operation is largest near the leakage's peak, and the leakage variability ($VAR_i(M_{t,i})$) is highly correlated with SNR (as shown in Sec. 5.1), we hypothesize that increasing the sampling rate would bring us new and independent features (defined in Theorem 1), which would make the attack on the swapped-XOR feasible, even for 32 bits.

6.2 FEbA - Mean vs. Covariance Estimators

Just like the previous section, we evaluate the operation $MACRO = \mathbf{k} \oplus \mathbf{r}$ using the same assumptions (using the swapped implementation). This time, we use the covariance estimator instead of the mean estimator $\theta(\ell) = (\ell - \mu_\ell) \cdot (\ell - \mu_\ell)^T$ where $\mu_\ell = E(\ell)$. Given our leakage model as described above, one may notice that because $\hat{L}$ is affine in $\mathbf{k}$ for our model (due to the relation $k_i \oplus r_i = k_i(1 - 2r_i) + r_i$), we get that

$$COV(\hat{L}) = E(\hat{L}\hat{L}^T) + E(\hat{L})E(\hat{L})^T \tag{15}$$

which is at most a 2^{nd} degree polynomial of $\mathbf{k}$, meaning the expectation is of the form $C + \sum_{i,j} C_{i,j} k_i k_j$. This type of feature lets us use information from Z^r and Z^{kr} to create an attack, unlike the mean estimator.

Conceptually, this estimator uses the differences in parity between the leakages on each r_i and $k_i \oplus r_i$ to extract information on the key. The outline of the pseudo-algorithm listing performed:

---

1. Sample a profiling set $\{\mathbf{P_i}, \mathbf{k_i}\}$ and an attack set $\mathbf{A_i}$ on $\mathbf{k}^\star$.
2. Calculate $\mu_A = E(\mathbf{A_i})$.
3. Create the model via linear regression $M \cdot (1, pair(\mathbf{k_i})) \leftarrow \text{unravel}((\mathbf{P_i} - \mu_A)(\mathbf{P_i} - \mu_A)^T)$, where $pair(\mathbf{k}) = \{k_i \cdot k_j | \forall i, j\}$ and unravel maps a matrix to a vector elementwise.
4. Calculate $POI_i = Var_{j|\exists m: pair(\mathbf{k})_j \equiv k_m}(M_{i,j})$
5. Find as many POIs as your environment can handle by sorting through the max arguments of POI_i, $IDX = \text{top } d - \arg\max_i POI_i$.
6. Calculate the feature $\theta(\mathbf{A}) = E_i((\mathbf{A_i} - \mu_A)(\mathbf{A_i} - \mu_A)^T)[IDX])$
7. Calculate the attack covariances $\Sigma^L = COV_i((\mathbf{A_i} - \mu_A)(\mathbf{A_i} - \mu_A)^T)[IDX])$.
8. Find the fitting keys $\min \arg_{\mathbf{k}} \| M[IDX, :] \cdot (1, pair(\mathbf{k})) - \theta(\mathbf{A}) \|_{\Sigma_L}$

It is important to note that this methodology adds a layer of heuristics due to the POI choice metric. Our current POI selection is based on degree-one coefficient variance (step 3 in the procedure). This criterion is not optimized for key discriminability under the attack noise; it ignores the structure of Σ^L, it neglects to consider the asymmetries introduced by $k_i k_j$, and may also choose features that are all highly correlated. Additionally, a high variance in the model may indicate unstable profiling (model variance errors, as per Appendix A.2) instead of a good feature.

Table 7: Average guessing entropy adversaries on Scenario 2, $n_k = 16$ from Fig. 1 using the covariance estimator

# Attack traces	# Features used d				
	100	**250**	**500**	**1000**	**2500**
n=25,000	6.61	6.06	7.57	7.19	N/A
n=50,000	7.22[a]	5.7	6.74	6.84	7.18

[a] The average entropy increases with doubling n due to noise, evidently because the top 100 rated features don't contain enough information to guess the key reliably.

Tab. 7 displays that the limited methodology discussed above presents itself as significant. This discussion is further consolidated when contrasting the results in Tab. 7 to those in Sec. 6.4, where we obtain more information under a more challenging setting (though this isn't an equivalent comparison). We hypothesize that the gap in our results occurs since we are using an attack derived for an additive feature model to guess keys for a 'pairwise' model. Contrary to all that, we see that the variance estimator's results are only slightly inadequate compared to those shown in Sec. 6.1.

We observe a non-monotonic dependence on the number of selected features: beyond ≈ 250 POIs, the guessing entropy increases. We hypothesize that incrementing d beyond a certain threshold adds highly correlated or low-SNR features, which degrade the key ranking metric and the model accuracy. This effect may be additionally amplified by our POI heuristic (variance over regression coefficients).

Finally, due to the nature of FEbAs, we could stack the mean and covariance estimator to create a feature θ that uses both of them at once, theoretically improving results since both features use unrelated quantities to guess the keys. Attempting this is beyond the scope of this paper.

6.3 FEbA with Known Plain-text Analysis

Clearly the proposed attack objective is to computationally manage a *hard* case where intermediate values are unknown. To complement our results, we also highlight that our attack (even in the base setting with known intermediate), can at the very least compete with conventional Template attacks, with a grounded argument for when it will surpass it. Therefore, we provide results from the known plaintext scenario. For more complex cases where we utilize more leakage time-samples or the size of the attacked key is ≥ 32-bits, or intermediate values are unknown, Template attacks will exponentially become infeasible where FEbA will flourish.

In this section, we attack the XOR swapped scenario. Observing the model

$$\hat{L}_t = L'_t + \sum_i k_i \cdot Z_{t,i}^{k,k_i} + \sum_i p_i \cdot Z_{t,i}^{r,p_i} + \sum_i (p_i \oplus k_i) \cdot Z_{t,i}^{kr,k_i \oplus p_i}$$

We substitute $k_i \oplus p_i = k_i + p_i - 2k_i p_i$, preprocess the trace with $\tilde{L}_t = L_t - E(L'_t) - \sum_i p_i \cdot \left(E(Z_{t,i}^{r,p_i}) + E(Z_{t,i}^{kr,k_i \oplus p_i})\right)$ and take the expectation on $\mathbf{r}, S$ (the same as expectation conditioned on $\mathbf{p}, \mathbf{k}$) on both sides to get:

$$E_{\mathbf{r},S}(\tilde{L}_t) = \sum_i k_i \cdot \left(E_{\mathbf{r},S}(Z_{t,i}^{k,k_i}) + E_{\mathbf{r},S}(Z_{t,i}^{r,p_i}) - 2p_i E_{\mathbf{r},S}(Z_{t,i}^{kr,k_i \oplus p_i})\right)$$

This attack might work well enough, but it loses one of the advantages of FEbAs, where the key guessing step is a simple matrix optimization problem. As such, we take the expectation over $\mathbf{p}$ to obtain the optimization problem:

$$E_{\mathbf{p},\mathbf{r},S}(\tilde{L}_t) = \sum_i k_i \cdot \left(E_{\mathbf{r},S}(Z_t^{k,i}) + E_{\mathbf{r},S}(Z_t^{kp,i}) - 2E_{\mathbf{p}}(p_i) E_{\mathbf{r},S}(Z_t^{kp,i})\right)$$

At first sight, this might be very similar to the unknown input scenario, where we guess based on the input distribution, but there are some differences.

1. The exact distribution is known - This lets us perform the attack without needing to gather enough statistics for the average to converge strictly.
2. Trace preprocessing - the knowledge of $\mathbf{p}$ is used to denoises the traces.
3. We may control the distribution - this lets us selectively choose certain $\mathbf{p}$s, which allows us to artificially inflate the coefficients of k_i to obtain an easier attack (this is the key concept).

The points above give us leeway in how we may utilize our plaintext distribution. But since feature estimations evaluate features of the leakage, it may not be surprising to see that using a feature independent of $\mathbf{p}$ (even with the pre-processing step described above) in an attack similar to the one in Sec. 6.1 would result in subpar results, as seen in Appendix A.8. As such, we wish to use features that leverage $\mathbf{p}$ better.

Since the final distribution of the $\mathbf{p}$'s influences the eigenvalues of Λ, we can sub-sample the wanted $\mathbf{p}$'s to inflate certain eigenvalues pertaining to features we may want to observe.

This will increase the variance across the specific eigenvalue which will make it easier differentiating keys across this specific feature. In the simplest form, we can differentiate a single bit at a time, and sub-sample $\mathbf{p}$ based on the value of the targeted bit. In total, the feature is equal to the following:

$$\theta_{m,b}(\mathbf{X}) = \frac{1}{n}\sum_i 1_{p_{i,m}=b} \cdot (\ell_i - b - M'\mathbf{p_i})$$

where b is a vector and $M' = M_1 + M_2$ a matrix (defined below).

As such, we propose a simplistic attack procedure:

---

1. Sample a profiling set $\{\mathbf{P_i}, \mathbf{k_i}\}$ and an attack set $\mathbf{A_i}$ on $\mathbf{k}^\star$.
2. Build model via linear regression $(b||M_0||M_1||M_2) \cdot (1, \mathbf{k_i}, \mathbf{p_i}, \mathbf{k_i} \oplus \mathbf{p_i}) \leftarrow \mathbf{P_i}$.
3. Preprocess the attack traces $\mathbf{A_i}' \leftarrow \mathbf{A_i} - b - (M_1 + M_2) \cdot \mathbf{p_i}$ for all
4. For each bit index j:
5. For $b \in \{0, 1\}$, split $I_b = \{i | p_{i,j} = b\}$.
6. calculate the feature $\theta_b(\mathbf{A}) = E_{i \in I_b}(\mathbf{A_i}')$
7. Calculate the attack covariances $\Sigma_b^L = COV_{i \in I_b}(\mathbf{A_i}')$.
8. Define $M^b_{t,i} = (M_0)_{t,i} + [1 - 2E_{j \in I_b}(p_{j,i})](M_2)_{t,i}$
9. Find the most fitting key $\mathbf{k^b} = \min \arg_{\mathbf{k}} \|M^b \cdot \mathbf{k} - \theta_b(\mathbf{A})\|_{\Sigma_b^L}$
10. Compare the $\mathbf{k^b}$'s using the distance metric and select $b^\star$.
11. Guess $k_j \leftarrow k_j^{\mathbf{b}^\star}$

Table 8: Average error of adversaries on Scenario 1 from Fig. 1 for the mean estimator per individual key bit.

Configuration	**# Attack Traces n**					
	25	**50**	**100**	**250**	**500**	**1000**
16-bit (T=60K, #Keys=50)	5.6	4.42	2.56	.48	.08	.02
32-bit (T=60K, #Keys=50)	13.3	10.94	8.7	2.68	.44	.06

The attack still has many ways of improving. *It is fairly plausible to assume that under different settings, our attack will perform better than template attacks.* For example, faster sampling speeds (our attack is not as limited as template in that aspect), creating a prior profile of the error[7], attacking larger keys, and incorporating different features.

6.4 Feature Estimation Based Attack on Masks Refreshing

Observing Scenario 4 (from Fig. 1), we get samples from two XOR operations $(\mathbf{k} \oplus \mathbf{r}_{mask}) \oplus \mathbf{r}_{remask}$ and $\mathbf{r}_{\text{mask}} \oplus \mathbf{r}_{\text{remask}}$ (sampled in different acquisitions, not consecutive in time).

The leakage of the XOR operation is:

$$\hat{L}_t^j = L_t' + \sum_i s_i^j \cdot Z_{t,i}^{k,s_i^j} + \sum_i r_{\text{remask},i} \cdot Z_{t,i}^{r,r_{\text{remask},i}} + \sum_i (s_i^j \oplus r_{\text{remask},i}) \cdot Z_{t,i}^{kr,(s_i^j \oplus r_{\text{remask},i})}$$

where $\mathbf{s}^1 = \mathbf{k} \oplus \mathbf{r}_{\text{mask}}$ for the first operation, and $\mathbf{s}^2 = \mathbf{r}_{\text{mask}}$ for the second one.

Since every argument is random by itself, the mean estimator would be ineffective for an attack due to the linearity of expectations. Notice that for $i \neq j$, both k_i and k_j are always masked with different i.i.d masking bits, which implies that $\forall x_1, x_2, t, \tau, i \neq j$: $COV(Z_{t,i}^{x_1,s_i^0}, Z_{\tau,j}^{x_2,s_i^1}) = 0$ which means that $COV(L_t^1, L_t^2|\mathbf{k}) = b_t + \sum_i m_{t,i} k_i$, for some scalars $b_t, \{m_{t,i}\}_i$.

[7] Concretely, since only a small number of traces were used, Σ^L did not converge, meaning further profiling may be beneficial, such as described in Appendix A.2

As such, we may use the covariance between the leakages of the two XORs as a reference. We use the estimator $\theta(\ell_1, \ell_2) = (\ell_1 - \mu)(\ell_2 - \mu)^T$, where $\mu = E(L)$.

This causes an issue, since the complexity of the attack becomes l^4, which forces us to predetermine the best points of interest to be chosen for long traces.

Additionally, we do not use the knowledge of $\mathbf{r}$ in the profiling of the model, since this is beyond the scope of this paper. The procedure's pseudo-code listing:

---

1. Sample a profiling set $\{\mathbf{P_{i,1}}, \mathbf{P_{i,2}}, \mathbf{k_i}\}$ (sets of the first and second masking XOR) and an attack set $\mathbf{A_{i,1}}, \mathbf{A_{i,2}}$ on $\mathbf{k}^\star$, and calculate $\mu = E(\mathbf{P_{i,1}})$.
2. Profile the matrix M via linear regression $M \cdot (1, \mathbf{k_i}) \leftarrow \text{unravel}((\mathbf{P_{i,1}} - \mu)(\mathbf{P_{i,2}} - \mu)^T)$ where unravel maps a matrix to a vector elementwise.
3. Calculate $POI_i = Var_j(M_{i,j})$ and find the indices of the maximal d POIs, denoted by IDX.
4. Calculate the feature $\theta(\mathbf{A}) = E_i((\mathbf{A_{i,1}} - \mu)(\mathbf{A_{i,2}} - \mu)^T)[IDX])$
5. Calculate the attack covariances $\Sigma^L = COV_i((\mathbf{A_{i,1}} - \mu)(\mathbf{A_{i,2}} - \mu)^T)[IDX])$.
6. Find the fitting keys $\min \arg_{\mathbf{k}} \|M[IDX, :] \cdot (1, \mathbf{k}) - \theta(\mathbf{A})\|_{\Sigma_L}$

Note: both XOR traces were take independently. Tab. 9 describes the results (remember, no access to plaintexts anywhere). As one can see, we can easily extract more than 10 bits of information.

As seen in Appendix A.10.2, for unswapped XOR with $n = 100,000$ and $d = 250$ we obtain $AGE = 1.04$.

Notably, the performance gap between the switched/non-switched XOR is much smaller, since the masked scenario also extracts leakage from the leakage features that are dependent on $\mathbf{r}$.

Table 9: Average guessing entropy of adversaries on Scenario 4 from Fig. 1 using the covariance estimator, $n_k = 16$, $T = 2M$.

# Attack traces (n)	**# Features used** d				
	100	**250**	**500**	**1000**	**2500**
25,000	3.33	2.66	3.05	4.38	N/A
50,000	3.33	2.25	2.6	4.19	N/A
100,000	3.08	1.87	2.3	3.5	4.65

6.5 FEbA on full algorithms

Following the principles of algebraic attacks, we define feature estimation-based attacks that targets full cryptographic applications.

For adversaries that target a full algorithm, it is possible incorporate leakages from multiple instructions to extract a key by identifying all the relevant intermediate values, defining a leakage model $\hat{L}$ dependent on the intermediate values, and creating the corresponding optimization problem for estimating the intermediate values. In practice, it may be near impossible to optimize the expression $||M \cdot f(\mathbf{k}) - \theta(\mathbf{X})||_{\Sigma^L}$ for a complex function f for a large n_k, even using heuristic algorithms.

For $n_p = 0$, using a divide-and-conquer approach for feature estimation attacks is challenging due to the lack of intermediate values. Since operations will (in most cases) occur consecutively, residual leakages from one operation will interfere with the leakages of future operations, inducing a biased offset that is difficult to compensate for, as illustrated in Fig. 9. The figures clearly demonstrate how the leakage from each subkey appears in (overlapping) time slots. Compared to guessing 128 bits simultaneously, this increases

the complexity of the attack beyond the theoretical lower limit of targeting each subkey independently, assuming no interference.

6.5.1 Scope reduction

Let us take the ASCON algorithm as described in Fig. 10 implemented on a 32-bit device using 16-bit integers to store the key an example. This algorithm has a 128 bit key size with thousands of constraints. Of course, the attack is still feasible with $n_k = 32$, only harder.

We do a 'scope reduction' to reduce the key guessing space:

1. Looking at the orange XORs in Fig. 10, we assume that the arguments XORed with the key (the argument which is fed to the XORs from the left) are uniformly distributed.
2. We ignore all leakage time samples except those of the first XOR with the key for each subkey.

We are left with a set of eight 'standalone' 16-bit XOR instructions performed in a loop. This reduces the entire AES/ASCON algorithm to multiple XOR problems, which we solved individually above.

We perform the same attack described in Sec. 6.1, using the heuristic algorithm [GN06] on $n_k = 128$ bits, to obtain the results in Tab. 10, where we reduced the empirical entropy by ~ 100 bits by guessing 128 bits at once. Fig. 9 presents the leakage model over time. This type of attack is also applicable to AES, among other algorithms, as long as there is a single unmasked instance of the key. Referring to Fig. 5, it is clear to see how operations not only interfere with each other.

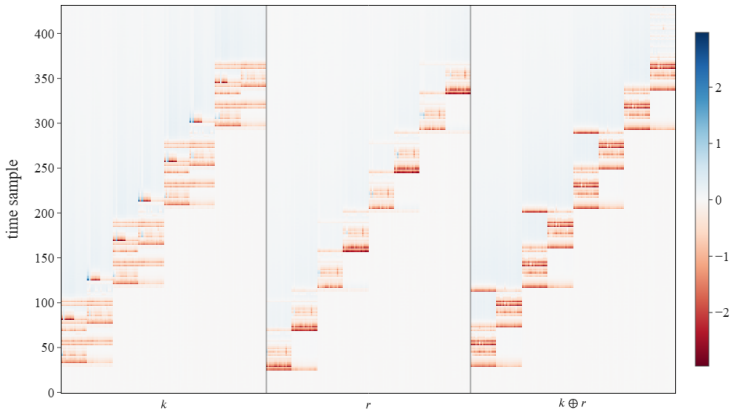


Figure 9: Leakage time samples versus the 32-bit trained features. The X-axis is partitioned into three consecutive regions corresponding to the trained features $\mathbf{k}$, $\mathbf{r}$, $\mathbf{k} \oplus \mathbf{r}$, while the Y-axis represents leakage time samples. Each colored pixel represents the estimate value of $E(Z_{t,i}^{x,1} - Z_{t,i}^{x,0} | \mathbf{r}, \mathbf{k})$, where t is the vertical axis, and $(x, i) \in \{\mathbf{k}, \mathbf{r}, \mathbf{k} \oplus \mathbf{r}\} \times \mathbb{Z}_{128}$ is the horizontal one.

To obtain better performance, one may use leakage samples from instructions related to the targeted XOR, at the cost of additional algorithmic innovation.

Table 10: Guessing entropy of adversaries on Scenario 4 from Fig. 1 for the mean estimator (ASCON).

Configuration	**# Attack Traces n**		
	n=12,500	**n=25,000**	**n=50,000**
16x8-bit (T=2.5M, #Keys=50)	9.74 w.p. 42% w/ 1M	8.91 w.p. 52% w/ 1M	8.99 w.p. 56% w/ 1M

6.6 Targets with multiple instructions

We attack a target that performs the AES column-mix function on a 32-bit key. We denote $f(\mathbf{k})$ as the set of intermediate values for this calculation. Appendix A.9 describes the full algorithm and set of intermediate values.

Given that, we perform a simple feature estimation attack:

---

1. Sample a profiling set $\{\mathbf{P_i}, \mathbf{k_i}\}$ and an attack set $\mathbf{A_i}$ on $\mathbf{k}^\star$.
2. Create the model via linear regression $M \cdot (1, f(\mathbf{k_i})) \leftarrow \mathbf{P_i}$.
3. calculate the feature $\theta(\mathbf{A}) = E_i(\mathbf{A}_i)$
4. Calculate the attack covariances $\Sigma^L = COV_i(\mathbf{A}_i)$.
5. Find the fitting keys $\arg\min_{\mathbf{k}} \|M \cdot (1, f(\mathbf{k})) - \theta(\mathbf{A})\|_{\Sigma_L}$

Since the key rating algorithm is constricted, it's more difficult to create a heuristic version thereof that runs instantly. Notice that since $f(\mathbf{k})$ is linear over $\{0, 1\}$, we may create a quadratic Boolean optimization problem to solve somewhat efficiently. Tab. 11 shows that FEbAs may extract significant information in a setting similar to ASCA (albeit in a smaller scope). Results in the table are quite impressive and clearly dependent to some extent on the functional structure of the (say) mixing (linear) part of the substitution-permutation network (SPN).

Table 11: Guessing entropy of adversaries on MIX-COLUMNS operation (no intermediates).

Configuration	**# Attack Traces n**		
	n=10	**n=25**	**n=100**
32-bit (T=250K, #Keys=10)	10.87 w.p. 0.9 w/ 1M	6.69	4.18

Since only a small number of traces were used, the covariance statistics did not converge. A better iteration of the attack would use a profile of the covariance.

6.7 Application to (e.g.,) ASCON, ACE, GIBBON

The different adversaries we have shown above have immediate implications to modern cryptographic algorithms. Observing Fig. 10. Under the assumption that p^a is 100% secure with its output distributing uniformly, we still have (at least) three potential points for an attack.

We may use the ideas in Sec. 6.1/Sec. 6.5 to create a simple attack that guesses the key, reducing the expected guessing entropy of a 128 bit key to a range between $\sim 0 - 20$ bits depending on the implementation[8] by targeting one of the three XORs operations labeled in Fig. 10. This allows us to exploit the leakage regardless of the sponge rate or the masking order.

We may also extract information by using another feature to replace/ augment the mean estimator as seen in Sec. 6.2.

[8]For the case of an un-swapped XOR, full extraction may be possible. For the case of a swapped XOR, the results along with an assumption that $AGE < 32$ for unsuccessful keys (reasonable).

If the algorithm is protected via masking (with key remasking), we may use the adversary in Sec. 6.4.

Additionally, we can use the ideas in Sec. 6.3 to leverage the tag to extract **k** efficiently even in scenarios where attacks like CPA or template may find it difficult to be executed.

We can even combine all of the above into a singular FEbA that takes all of the operations into consideration at once (by performing vector concatenation).

We may also use the ideas in Sec. 6.6 to create a more comprehensive attack that uses the operations adjacent/ related to the targeted operations.

For SCA software implementations, we would recommend reviewing seemingly benign implementation choices, like the assembly register operation order of XOR, array indexing, and operation separation. The importance of these details may be seen above; for an attack on the shared-XOR operation with an unknown operator r with a key of size 32 bits, only $17K$ attack traces are needed to reduce key space entropy to below 1 bit, as compared to the swapped XOR version, which doesn't drop below 20 bits of information even for $100K$ attack traces. As such, leakage depends heavily on the algorithm's target. At the very least, this result shows that the leakage sensitivity as a function of logically equivalent implementation details is significant, and as such, we argue to increase the bar for what is considered a mundane implementation detail in side channel attacks.

Finally, since FEbA are at heart a dimensionality reduction methodology, they can be used in sequence with other types of attacks, being used to filter unlikely from likely keys.

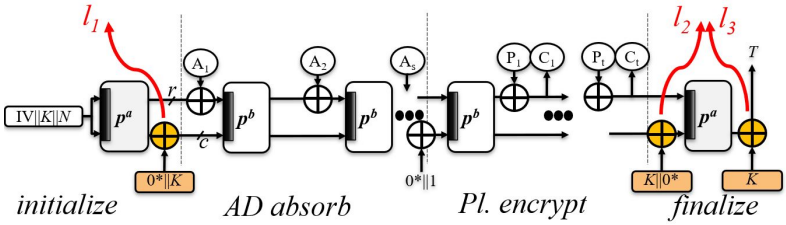


Figure 10: Ascon scheme and FEbA access-points (denoted by red leakage trajectories from three c-bit XORs in orange)

7 Summary and Discussion

This paper proposes a new side-channel attack methodology called Feature Estimation based Attacks (FEbA), which exploits asymmetric leakage in simple XOR operations (for example, masked key XOR with a random mask). We show that a standalone XOR, and even a masked implementation ($d = 2$), can exhibit exploitable, key-dependent leakage in practical implementations. We show that the magnitude of this information can depend strongly on micro-architectural realization and operand handling (e.g., operand order), as observed empirically. We hypothesize that with higher sampling rates, the effectiveness of this attack increases nonlinearly. Additionally, we note that for low-bandwidth operations (e.g., 8-bit XORs) our attack will be highly effective. Our conclusions should be interpreted as a need for caution with the idea that simple or linear operations (especially when masked) are hard to exploit by side channels, affecting various classical schemes' design choices. Alternatively said, it highlights the risk of assuming a leakage function that holds a symmetric or low-entropy structure (for modeling or attack simplicity) to carry a significant amount of information.

A Appendix - supplemental material

A.1 Conventional SCAs in View of Feature Estimation Based Attacks

Note that $\theta_{\mathbf{k}}$ depends on $\mathbf{k}$, and thus, it may be computationally challenging to exploit (or reduce to an optimized problem) when the key length $n_k \geq 32$ and the problem is not divisible into sub-problems.

Additionally, complex features depends on $\mathbf{k}$ which make it difficult to create a mathematical analysis. Alternatively said, such dependency in turn may result in heuristics/ additional assumptions or simplifications for feature estimation.

When viewing the subject from a very generic angle given the definitions above, it is beneficial to look at other side channel attacks and observe which assumptions are implicitly made by trying to apply our definitions to them. The goal is to give more intuition to the reader before the next chapters.

For specific algorithms or specific hardware/ software implementations, attacks like the ones discussed next may converge faster than the one we will define. Attacks fail only when the (explicit and implicit) assumptions they make do not fit the target they go up against.

A.1.1 Model Based Attacks

Model-based attacks are attacks that use a leakage model that compares a set derived from intermediate values (as a function of a key guess) to a set derived from the attack leakages by assigning a rank to the notion of how much both sets 'match'.

Attacks like CPA [Man04], MCP-DPA [MS16], and MIA [GBTP08b] all fit the criteria. The concept of these attacks is to compare the intermediate values received by all key guesses to the leakage and find the optimal key (with or without any initial profiling).

For these types of attacks, the feature looks as follows:

$$\theta_{\mathbf{k}}(\{\ell_i, \mathbf{p_i}\}_i) = \mathrm{comp}(\{\mathrm{model}(\mathbf{p_i}, \mathbf{k}), f(\ell_i)\}_i)$$

where $f(\ell)$ is some pre-processing of ℓ, model is a function whose statistics is supposed to 'align' with $f(\ell)$ for the correct key, and comp is a procedure that is supposed to evaluate the statistic.

For example, for CPA targeting AES, we get that $\mathrm{model}(\mathbf{p_i}, \mathbf{k}) = \mathrm{HW}(\mathrm{SBOX}(\mathbf{p_i} \oplus \mathbf{k}))$, $f(\ell) = \ell$, $\mathrm{comp}(\{v_i, \ell_i\}_i) = \max_j |\rho(v, \ell_i[j])|$. MIA is the same as CPA but with a different comparison function, and MCP-DPA is the same as CPA, but with a profiled model and f being a moment calculator. Finally, we pick the key with the largest feature, implicitly assuming (by our convention) that $P_{\mathbf{k}|\mathbf{X}}$ is monotonically increasing with respect to θ.

The difference between this attack and ours (from the perspective of *our* rationale, which is discussed below) is mainly attributed to three factors, which are discussed rigorously below:

1. The feature estimation is dependent on the key, making the key-guessing strategy computationally difficult for big (sub) keys.
2. Complicated features, making it difficult to mathematically bound this setup.
3. Difficult to coherently incorporate multiple 'features' at once.
4. Usually neglects to encode leakage asymmetry (not necessary for known $\mathbf{p}$).

A.1.2 Profile Based Attacks

Profile-based attacks are attacks that characterize a probability distribution and guess based on it. Examples are template attacks, TA [CRR02], or neural-networks (NN) based attacks (e.g., [KPH+19]).

From a feature estimation perspective, profiling attacks, explicitly assume a distribution $\hat{L}$ given parameters and regress the parameters to fit $L^\star$, instead of pattern matching $w^\star_{\mathbf{k}}$ to $\hat{w}_{\mathbf{k}}$. In profiling attacks, leakage equivalence classes are used to lower the profiling (and attack) complexities: $f(\mathbf{k},\mathbf{p}) = f(\mathbf{k}',\mathbf{p}') \Rightarrow \hat{L}_{\mathbf{k},\mathbf{p}} \equiv \hat{L}_{\mathbf{k}',\mathbf{p}'}$

As for key guessing, profile attacks try to estimate the key based on the evaluated probability, without using bounds as we did. For completeness, we note that profile attacks use the identity function $\theta_{\mathbf{k}}(\mathbf{X}) = \{\ell_{\mathbf{i}}\}_i$ as a feature.

For example, in a template attack we assume $\hat{L}_{\mathbf{k},\mathbf{p}}, \sim \mathcal{N}(\mu_{f(\mathbf{k},\mathbf{p})}, \Sigma_{f(\mathbf{k},\mathbf{p})})$ and profile the different μ, Σ's and attack accordingly.

Once again, this results in disadvantages similar to those attributed to CPA.

1. The characterization of the model depends on the key and is heuristically assumed to be a normal distribution (or some other distribution).
2. The model depends on the key in a complicated fashion, which may result in an inability to leverage the model's structure to search for keys efficiently for larger key spaces/ more POIs.
3. The utility of the plaintext bits is superior to the feature estimation attacks we show in Sec. 6.3 and Appendix A.8.

The usual choice for a template is the Gaussian, which is characterized using its mean and variance. In practice, this characterization is similar to the adversary we propose (or at least, to the mean estimation one). The characterization differs in the following ways:

- The variance in our case is (for now) calculated solely based on the attack traces; it is not profiled (for the mean estimator).
- The key is picked based on an evaluation of some expectation of the traces, not on the sum of evaluations of different traces.
- We can arbitrarily add more features to our estimator in a mathematically concise way (for example, adding variance analysis). For template attacks, it may be difficult to find (and evaluate) probabilities with many features.

A.2 Modeling abstraction

During an attack, different useful quantities can be considered, which were defined above:

1. The expected value of the estimation of the FLD, $f^\star_{\mathbf{k}}$ - the value of the feature which is obtained asymptotically (in the number of attack traces) from the hardware for every key $\mathbf{k}$.
2. The estimated feature of the FLD, $w^\star_{\mathbf{k}}$ - the estimated value of $f^\star_{\mathbf{k}}$ using finite profiling traces.
3. The expected value of the estimation of the profiled model, $\hat{f}_{\mathbf{k}}$ - the leakage model obtained from an asymptotically (number of profiling traces) profiled model.
4. The estimated feature of the profiled model $\hat{w}_{\mathbf{k}}$ - the estimated value of $\hat{f}_{\mathbf{k}}$ using finite profiling traces.

Using Fig. 11 as a reference, we define the types of errors that may cause a mismatch between the model and the calculated average.

1. **Noise:** Error originating from the estimation of the feature - lack of samples in the attack phase, or from the non-i.i.d.ness of the attack traces (since we assume i.i.d).
2. **Variance:** Error originating from the estimation of the model - lack of samples in the profiling phase, from variations between different devices, or from the non-i.i.d.ness of the profiling traces (since we assume i.i.d).
3. **Bias:** Error originating from the innate inaccuracy of the model - caused by the limitations of the model with respect to how the hardware behaves in action. For example, a linear model will have a biased error if the leakage behaves nonlinearly.

All errors can be mitigated in two different ways.

1. Resource augmentation: taking more traces or using a more complicated model.

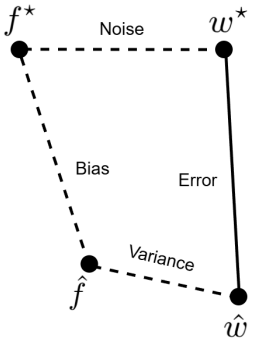


Figure 11: Partition of the total error between the feature's profile and the estimated feature (solid line) into three types of 'virtual' errors (dashed lines)

2. Profiling circumvention: creating a better statistical profile of the error (for example, error direction characterization or stationary variable analysis).

We denote $\mathbf{X^k}$ as a leakage trace originating from an execution with the key $\mathbf{k}$. In this paper, we discuss models of the form $M \cdot f(\mathbf{k})$ with M being a matrix and f being some Boolean function. As such, we define:

- The model matrix M (equiv. to $\hat{w}_\mathbf{k}$) - In our case, trained via linear regression (from labels $labels(\mathbf{k_i})$ and traces $\theta(\mathbf{X^{k_i}})$).
- The optimal model $M_{opt} = \arg\min_M E_\mathbf{k}(\|E(\theta(\mathbf{X^{k_i}})) - M \cdot \mathbf{k}\|^2)$ (equiv. to $\hat{f}_\mathbf{k}$).
- The model error $M_{err} = M - M_{opt}$.
- A bias term $B(\mathbf{k}) = f^\star_\mathbf{k} - M_{opt} \cdot \mathbf{k}$.
- A noise R.V. $\epsilon_\mathbf{k} = \mathbf{X^k} - M_{opt} \cdot \mathbf{k} - B(\mathbf{k})$.
- The attack noise covariance matrix $\Sigma^L = COV_i(\theta(\mathbf{X_i^{k^\star}}))$.

In this paper, we mainly use Σ^L as a way to reduce the attack's complexity (as done in Appendix A.5 and Sec. 6.1). This, along with taking more attack traces, is our primary method for mitigating errors.

We have attempted to reduce the bias error by statistical characterization (We assumed $B(\mathbf{k})$ to be taken from a random distribution independent of $\mathbf{k}$ and guess against a metric Σ created form Σ^L and $\Sigma^B = COV(B(\mathbf{k}))$) or by better models (models with higher polynomial order) always caused the adversaries' performance to diminish. This is assumed to be caused by the bias between the optimal linear model and the faithful leakage model (the FLD) is small enough under our guessing space such that making attempts to model it only increases the noise in the system[9].

The variance error was mitigated by taking a larger number of profiling traces. Doing a statistical analysis about the variances of the weights in the model is beyond the scope of this paper.

A.3 Leakage Weights Parameterization

We denote $\nu_t = E(L'_t)$, and $\mu^b_{t,i} = E(Z^{k,b}_{t,i})$, $\mu^D_t = \frac{1}{n_k}\sum_i \mu^1_{t,i} - \mu^0_{t,i}$, $n_z = n_k - n_o$

[9]Perhaps for different hardware, this would not be the case. Even in our results, the bias from the linear model is very noticeable, but it doesn't affect the results as much since it occurs in low SNR leakage samples

$$\begin{aligned}\mu_t^W &= E(L^\star_{\mathbf{k},t}|\mathbf{k})|\,\mathrm{HW}(\mathbf{k}) = n_o) \\ &= \nu_t + \frac{1}{\binom{n_k}{n_o}} \sum_{\mathrm{HW}(\mathbf{k})=n_o} \sum_{i=1}^{n_k} \mu_{t,i}^{k_i} \\ &= \nu_t + \sum_i (\frac{n_z}{n_k}\mu_{t,i}^0 + \frac{n_o}{n_k}\mu_{t,i}^1)\end{aligned} \tag{16}$$

$$\begin{aligned}\Sigma_{t,\tau}^W &= COV_{\mathbf{k}}(E(L^\star_{\mathbf{k},t}), E(L^\star_{\mathbf{k},\tau})|\,\mathrm{HW}(\mathbf{k}) = n_o) \\ &= COV_{\mathbf{k}}(\sum_{i=1}^{n_k} \mu_{t,i}^{k_i}, \sum_{i=1}^{n_k} \mu_{\tau,i}^{k_i})|\,\mathrm{HW}(\mathbf{k}) = n_o) \\ &= COV_{\mathbf{k}}(\sum_{i=1}^{n_k} \mu_{t,i}^0 + \sum_{i=1}^{n_k} k_i(\mu_{t,i}^1 - \mu_{t,i}^0), \sum_{i=1}^{n_k} \mu_{\tau,i}^0 + \sum_{i=1}^{n_k} k_i(\mu_{\tau,i}^1 - \mu_{\tau,i}^0)|\,\mathrm{HW}(\mathbf{k}) = n_o) \\ &= COV_{\mathbf{k}}(\sum_{i=1}^{n_k} k_i(\mu_{t,i}^1 - \mu_{t,i}^0), \sum_{i=1}^{n_k} k_i(\mu_{\tau,i}^1 - \mu_{\tau,i}^0)|\,\mathrm{HW}(\mathbf{k}) = n_o) \\ &= \sum_i (\mu_{t,i}^1 - \mu_{t,i}^0)(\mu_{\tau,i}^1 - \mu_{\tau,i}^0) V(k_i|\,\mathrm{HW}(\mathbf{k}) = n_o) + \sum_{i\neq j} (\mu_{t,i}^1 - \mu_{t,i}^0)(\mu_{\tau,j}^1 - \mu_{\tau,j}^0) COV(k_i k_j) \\ &= \frac{n_o(n_k - n_o)}{n_k^2} \sum_i (\mu_{t,i}^1 - \mu_{t,i}^0)(\mu_{\tau,i}^1 - \mu_{\tau,i}^0) - \frac{n_o(n_k - n_o)}{n_k^2(n_k - 1)} \sum_{i\neq j} (\mu_{t,i}^1 - \mu_{t,i}^0)(\mu_{\tau,j}^1 - \mu_{\tau,j}^0) \\ &= \frac{n_o \cdot n_z}{n_k(n_k - 1)} \sum_i (\mu_{t,i}^1 - \mu_{t,i}^0 - \mu_t^D)(\mu_{\tau,i}^1 - \mu_{\tau,i}^0 - \mu_\tau^D) = \frac{n_o n_z}{n_k} COV(\mu^1 - \mu^0)\end{aligned} \tag{17}$$

A.4 Furhter intuition on the attack

A.4.1 Shedding the Key Weight Distribution Assumption

Let us compare the probabilities for just two keys for the one-dimensional case. In the limit:

$$\begin{aligned}\Pr_X[P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^\star|\mathbf{X}}] &= \Pr_X[P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} \geq P_{\mathbf{X}|\mathbf{k}^\star,\{\mathbf{p_i}\}}] \\ &= \int_A \Pr_X[\mu(X) = x]dx\end{aligned} \tag{18}$$

Where $A = \{x : g_d^{k^\star}(x|\{p_i\}) \geq g_d^k(x|\{p_i\})\}$ and X is taken from $L_{k^\star}(;r)$ the FLD.

Assuming Gaussian distributions:

$$\mu(X \leftarrow (L')^n) = N(\sum_i \mu_i^{xk_i^\star} + \mu_{ns}, \frac{\sigma_L^2}{n}) \tag{19}$$

WLOG, we will assume $\sum_i \mu_i^{xk_i^\star} \leq \sum_i \mu_i^{xk}$ (the opposite case follows a similar process with the same result):

$$\begin{aligned}A &= \{x : P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} \geq P_{\mathbf{X}|\mathbf{k}^\star,\{\mathbf{p_i}\}}\} \\ &= \{x : \left| x - \left(\sum_i \mu_i^{xk_i^\star} + \mu_0 \right) \right| \leq \left| x - \left(\sum_i \mu_i^{xk} + \mu_0 \right) \right|\} \\ &= \{x : x \geq \frac{1}{2} \sum_i \mu_i^{xk_i^\star} + \mu_i^{xk_i}\}\end{aligned} \tag{20}$$

We get for every key $k \neq k^\star$:

$$\begin{aligned}\Pr_X[f_d^{k^\star}(k|X) \geq f_d^{k^\star}(k^\star|X)] &= \int_A \Pr_{X \leftarrow L}[d(X) = x]dx \\ &\approx \int_{\frac{1}{2}\sum_i \left(\mu_i^{k_i^\star} + \mu_i^{k_i}\right) + \mu_0}^{\infty} f_N(x; \sum_i \mu_i^{k_i^\star} + \mu_0, \frac{\sigma_T^2}{n})dx \\ &= \Phi\left(-\frac{\frac{1}{2}\left|\sum_i \left(\mu_i^{k_i^\star} - \mu_i^{k_i}\right)\right|}{\sqrt{\frac{\sigma_T^2}{n}})}\right)\end{aligned} \tag{21}$$

This matches the rationale, where the key with the closest average distance to the true key is the bottleneck to choosing the correct key.

Notice that $\mu_i^{k_i}$ is a constant, and that $\Pr_X[P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} \geq P_{\mathbf{X}|\mathbf{k}^\star,\{\mathbf{p_i}\}}]$ is determined for set values of X, k. In addition, due to how we choose the keys, we get that if the key k_1 is less likely to be chosen than the key k_2, and the key k_2 wasn't chosen, it means that the estimated mean $\mu(X)$ is closer to $E(L_{k^\star})$ which means that k_1 also wasn't chosen (as long as $E(L_{k^\star})$ isn't between the mean of the true leakage of both keys).

This means that to find the probability $\Pr_X[\left|\{k : P_{\mathbf{X}|\mathbf{k},\{\mathbf{p_i}\}} \geq P_{\mathbf{X}|\mathbf{k}^\star,\{\mathbf{p_i}\}}\right| > K]$ is at most the sum of the two probabilities $\sum_{i\in 1,2} \Pr_X[P_{\mathbf{X}|\mathbf{k_i},\{\mathbf{p_i}\}} \geq P_{\mathbf{X}|\mathbf{k}^\star,\{\mathbf{p_i}\}}]$ where k_1 is the key whose value matches the K-th smallest value of $\left|\sum_i \left(\mu_i^{k_i^\star} - \mu_i^{k_i}\right)\right|$ between the keys $E(L_{\mathbf{k}}) \geq E(L_{\mathbf{k}^\star})$ and k_2 is the same but for $E(L_{\mathbf{k}}) \leq E(L_{\mathbf{k}^\star})$.

This derivation is important to show the true nature of the key-picking algorithm. The point is that the 'true' probabilities of finding the key has to do with the distance of the value of the leakage estimator. This is why for a leakage model like HW($\mathbf{k}$), or if the leakage variations per bit are too small, this entire procedure is not possible or extremely difficult.

In total, this means that Σ^W is actually just a "nice" object that approximates the idea shown above on average (second moment approximation), but it is not perfect. If $\mu_i^{k_i} = k_i + (i\%2)\cdot 50$ then even though Σ^W would be quite large for our uses, the performance will still be bad since a significant number of keys will all cause the same leakage, leaving us unable to differentiate them.

This is why the (implicit) assumption that our weights come from a distribution with a large support is important, since it prevents degeneracy (with some likelihood) in the mean leakages of the different keys.

A.4.2 The Effective Value of l

Looking at Σ^Z, it is reasonable to call each eigenvector a feature, with its effectiveness being the corresponding eigenvalue. Since Σ^Z is symmetric (it is also semi-positive definite), every feature is independent of the other, synonymous with assuming that the list of eigenvalues of the matrix represents the strengths of independent sources of information on the key.

The effective value of l, l_{eff}, is the number of such eigenvectors with eigenvalues that are not too small. i.e., the number of eigenvalues that result in a non-negligible amount of information on the key (as a function of the number of traces n, and the accuracy of the model $\hat{L}$).

A single feature may extract the entire key, but the difficulty of extracting B bits of information from a single features grows as B increases, since averaging over traces reduces the variance only by a factor of $1/n$ and there might be billions of keys.

Due to how Σ^W is defined, it will have at most $n_k - 1$ non-zero eigenvalues. This means that l_{eff} is bound by n_k. For the attack to work, we need a good enough model

and/or enough traces for each feature to result in ≈ 1 bits of information about the key in average.

Usually, increasing the scope sampling rate by a factor of 2 would imply either 'a $\times 2$ reduction in the number of traces due to better noise filtering' or 'a potential benefit if we obtained a slightly better POI'. In feature estimation attacks, due to the inherent instability of leakages near the leakage's peak (which also depends on the adjacent operations), such an improvement would also introduce a new orthogonal feature, which would augment one of the signal of one of the eigenvector with a small eigenvalue in Σ^Z (with some probability). This has the potential to improve the *complexity* of the attack in the number of traces needed - beyond what a $\times 2$ noise reduction would give us.

A.5 Enumeration bound for the mean estimator

Here, we assume that $n_p = 0$. The proof below can be easily reduced to the case where $n_p > 0$ by creating an instance of the proof for $n'_r = n_p + n_r$, and making $\mathbf{r}'$ distribute like $\mathbf{X}_p || \zeta$ (where $\mathbf{X}_p$ are the sampled plaintexts/ intermediate values). This reduction is not efficient since we neglect to use the leakage labels provided, but it is straightforward.

We work under the following conditions:

1. l-dimensional leakage under $\mathbb{R}^l$ (instead of finite support).
2. We assume the FLD is additive. This is a strong requirement, but in practice the attack is (evidently) resistant to deviations from the linearity on $\mathbf{k}$ as seen in Sec. 6.1. This implies that $f^\star_{\mathbf{k}}$ is a rank 1 polynomial of the vector $\mathbf{k}$ plus some small error (and as such, so is the model $\hat{w}_{\mathbf{k}}$) - at least on the high SNR clock cycles .
3. We estimate the distributions of $\hat{w}_{\mathbf{k}}$ over $\mathbf{k}$ with a multivariate Gaussian (originates from the central limit theorem; the model leakage is a sum of n_k Bernoulli variables). In other words, this is a second moment analysis. Discussion about this assumption exists in Appendix A.4.1. In the derivations below, we use a closed-form formula for the mean and variance of $\hat{w}_{\mathbf{k}}$, which are provided in Appendix A.3. Both of these assumptions originate from the central limit theorem on the sum of Bernoulli variables.
4. We estimate the distribution of $w^\star_{\mathbf{k}}$ with a multivariate Gaussian (also due to the central limit theorem). This is less of an assumption and more of a shortcut to prevent discussion about the Berry-Essen theorem, which has a negligible effect when $n \gg 1$.
5. We assume that the leakage over different time samples is i.i.d with leakage covariance matrix $\Sigma^L = COV_i(\ell_{\mathbf{i}})$ (similar results are obtained when assuming a stationary process, with a different definition for Σ^L). We additionally attribute the effect of $\mathbf{r}$ (the algorithmic noise source) to be a part of Σ^L, to simplify the calculations (in the limit, this doesn't influence the results).
6. We assume $\mathrm{HW}(\mathbf{k}) = n_o$ is known to simplify things. We denote $n_z = n_k - n_o$. This choice is valid since the number of traces we need to find the hamming weight is $O(1)$ (from observations - at most a double-digit number of traces is needed until the hamming weight of the key is converged on). In practice, this assumption doesn't change much.

These assumptions were made in mind to model an attack on an algorithm composed of a single operation on a register containing $\mathbf{k}$, with an oscilloscope that measures the leakage and residual leakages of the targeted operation, and none of the assumptions above are required for the attack to work in practice, only to create a proof and (a 2^{nd} order) complexity bound.

A.5.1 Counting Valid Key Guesses

Since $P_{\mathbf{k}|\mathbf{X}}$ is proportional to $P_{\mathbf{X}|\mathbf{k}}$ and the keys distribute uniformly $P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^\star|\mathbf{X}} \Leftrightarrow P_{\mathbf{X}|\mathbf{k}} \geq P_{\mathbf{X}|\mathbf{k}^\star}$. Since $\mu(\mathbf{X})$ is (eventually) distributed similarly to a Gaussian with variance Σ^L/n:

$$P_{\mathbf{X}|\mathbf{k}} \geq P_{\mathbf{X}|\mathbf{k}^\star} \Leftrightarrow \|E(\hat{L}_{\mathbf{k}}) - \mu(\mathbf{X})\|_{\Sigma^L} \leq \|E(\hat{L}_{\mathbf{k}^\star}) - \mu(\mathbf{X})\|_{\Sigma^L}$$

These result in the following calculation:

$$\begin{aligned} W_{\mathbf{k}^\star}(\mathbf{X}) &= \left|\{\mathbf{k} : P_{\mathbf{k}|\mathbf{X}} \geq P_{\mathbf{k}^\star|\mathbf{X}}\}\right| \\ &\approx \left|\{\mathbf{k} : \|E(\hat{L}_{\mathbf{k}}) - \mu(\mathbf{X})\|_{\Sigma^L} \leq \|E(\hat{L}_{\mathbf{k}^\star}) - \mu(\mathbf{X})\|_{\Sigma^L}\}\right| \\ &\approx \binom{n_k}{n_o} \int_{\{\mathbf{x}:\|\mathbf{x}-\mu(\mathbf{X})\|_{\Sigma^L} \leq \|\mu(\mathbf{X})-E(\hat{L}_{\mathbf{k}^\star})\|_{\Sigma^L}\}} f_N(\mathbf{x}; \mu^W, \Sigma^W)\mathbf{dx} \\ &\leq \binom{n_k}{n_o} \int_{\{\mathbf{x}:\|\mathbf{x}-\mu^W\|_{\Sigma^L} \leq \|\mu(\mathbf{X})-E(\hat{L}_{\mathbf{k}^\star})\|_{\Sigma^L}\}} f_N(\mathbf{x}; \mu^W, \Sigma^W)\mathbf{dx} \end{aligned} \tag{22}$$

Where f_N is the probability density function of a Gaussian. We define a new distribution $\mathbf{Z} = (\Sigma^L)^{-1/2}(\mathbf{D_k} - \mu^W)$, with $\mu^Z = 0$, $\Sigma^Z = (\Sigma^L)^{-1/2}\Sigma^W(\Sigma^L)^{-1/2}$. We denote $R = \|E(\hat{L}_{\mathbf{k}^\star}) - \mu(\mathbf{X})\|_{\Sigma^L}$, then diagonalize $\Sigma^T = U_Z \Lambda U_Z^T$ and denote $\mathbf{Z'} = \Lambda^{-1/2} U_Z^T Z$ with $\mu^{Z'} = 0$, $\Sigma^{Z'} = I$

$$\begin{aligned} &= \binom{n_k}{n_o} \Pr[(\mathbf{D_k} - \mu^W)^T(\Sigma^L)^{-1}(\mathbf{D_k} - \mu^W) \leq \|E(\hat{L}_{\mathbf{k}^\star}) - \mu(\mathbf{X})\|^2_{\Sigma^L}] \\ &= \binom{n_k}{n_o} \Pr[\mathbf{Z}^T\mathbf{Z} \leq R^2] = \binom{n_k}{n_o} \Pr[\mathbf{Z'}^T \Lambda \mathbf{Z'} \leq R^2] = \binom{n_k}{n_o} F_{\chi^2_l(;\Lambda)}(R^2) \equiv U_{k^\star}(\mu(\mathbf{X})) \end{aligned} \tag{23}$$

Where $F_{\chi^2_l(;\Lambda)}$ is the CDF of the generalized central chi-squared distribution with Gaussian squared coefficients being $\lambda_i = \Lambda_{i,i}$.

A.5.2 The K-n Tradeoff

We calculate the probability that there are more than K keys likelier than $\mathbf{k}^\star$. We denote $B = F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})$:

$$\Pr[W_{\mathbf{k}^\star}(\mu(\mathbf{X})) > K] \leq \Pr[U_{\mathbf{k}^\star}(\mu(\mathbf{X})) > K] = \Pr[\|\mu(\mathbf{X}) - E(\hat{L}(\mathbf{k}^\star))\|^2_{\Sigma^L} > B] \tag{24}$$

We define $\mathbf{Y} = (\Sigma^L)^{-1/2}\left(\mu(\mathbf{X}) - E(\hat{L}(\mathbf{k}^\star))\right)$, so
$\mu^Y = (\Sigma^L)^{-1/2}(E(L^\star_{\mathbf{k}^\star}) - E(\hat{L}_{\mathbf{k}^\star}))$, $\Sigma^Y = I/n$ (since $Var(\mu(X)) = \Sigma^L/n$), and:

$$\begin{aligned} \Pr[\|\mu(\mathbf{X}) - E(\hat{L}(\mathbf{k}^\star))\|^2_{\Sigma^L} > B] &= \Pr[n\|\mathbf{Y}\|_2^2 > nB] \\ &\leq 1 - F_{\chi^2_l(n\|E(L^\star(\mathbf{k}^\star)) - E(\hat{L}(\mathbf{k}^\star))\|^2_{\Sigma^L})}(nB) \leq \delta \end{aligned} \tag{25}$$

By inverting, we obtain:

$$n \geq \frac{F^{-1}_{\chi^2_l(n\|E(L^\star_{\mathbf{k}^\star}) - E(\hat{L}_{\mathbf{k}^\star})\|^2_{\Sigma^L})}(1-\delta)}{F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})} \tag{26}$$

This is the inverse of the CDF of the non-central chi-square distribution. This tells us about the relationship between K and n, and the other. Appendix A.6 describes the relationship in the signal-wise and noise-wise i.i.d scenario.

A.5.3 Leakage Model Success Condition

As per the previous subsection, the condition for getting the correct key is:

$$\frac{n}{F^{-1}_{\chi^2_l(n\|E(L^\star_{\mathbf{k}^\star})-E(\hat{L}_{\mathbf{k}^\star})\|^2_{\Sigma^L})}(1-\delta)} \geq \frac{1}{F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})}$$

Let us calculate an asymptotic approximation of $\frac{n}{F^{-1}_{\chi^2_l(nc)}(p)}$ using the Cornish-Fisher expansion for $\chi^2_l(n\cdot c) \to E(\chi^2_l(n\cdot c)) = l + n\cdot c, V(\chi^2_l(n\cdot c)) = 2(l+2cn)$. denote $q = \Phi^{-1}(1-\delta)$.

Since $K(t) = \log E(e^{t\chi^2_l(n\cdot c)}) = \frac{nct}{1-2t} - \frac{k}{2}\log(1-2t)$, we get that $K(t) = O(n)$.

This means that $\gamma_c(t=q) = \frac{O(n)}{O(n)^{1+c/2}} = O(n^{-c/2})$ which results in

$$\frac{n}{F^{-1}_{\chi^2_l(nc)}(p)} = \frac{n}{\mu_{\chi^2_l} + \sigma_{\chi^2_l} o(1)} = \frac{1}{c} + o(1) \underset{n\to\infty}{\to} \frac{1}{c}$$

So the condition for our model to eventually work is:

$$\frac{1}{c} = \frac{1}{\|E(L^\star_{\mathbf{k}^\star}) - E(\hat{L}_{\mathbf{k}^\star})\|^2_{\Sigma^L}} \geq \frac{1}{F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})}$$

Meaning that if we want the attack to work for every key, we obtain the sufficient condition: $\|E(L^\star_{\mathbf{k}^\star}) - E(\hat{L}_{\mathbf{k}^\star})\|^2_{\Sigma^L} \leq F^{-1}_{\chi^2_l(;\Lambda)}(K/\binom{n_k}{n_o})$

Intuition: This is synonymous with requiring that the smallest ball that contains K keys, which is positioned around the point $E(\hat{L}_{\mathbf{k}^\star})$ should contain the point $E(L^\star_{\mathbf{k}^\star})$, which is logically sound.

For n to converge to a realistic value, we should attempt to fulfill this bound with a wide margin. Further discussion on how l and Λ are affected by the leakage and affect the #traces complexity (n) is present in Appendix A.4.2.

A.6 Enumeration bound i.i.d leakages

we take a look at the scenario where $\Sigma^L = \sigma^2_L/n\cdot I$, $\Sigma^W = \sigma^2_W I$. Instinctively, this case is the best case for us since there is no information overlap since both Σ^W, Σ^L are diagonal (a specific case of the general best-case scenario - where the directions of the eigenvectors of Σ^L, Σ^W are aligned in a way that maximizes Σ^Z's eigenvalues - I.e., both matrices are simultaneously diagonalize). We get a non-generalized and central chi-square distribution, which is easier to work with.

$$n \geq \frac{\sigma^2_L F^{-1}_{\chi^2_l(n\|E(L(\mathbf{k}^\star))-E(\hat{L}(\mathbf{k}^\star))\|^2/\sigma^2_L)}(1-\delta)}{\sigma^2_W F^{-1}_{\chi^2_l}(K/\binom{n_k}{n_o})} \tag{27}$$

Which shows that n is proportional to a quantity that depends on our success rate and model error, and inversely proportional to some quantity dependent on how many key guesses are allowed, and to a quantity best described as the SNR.

Also, the condition for the model leakage success is:

$$\forall k : \|E(L^\star(k)) - E(\hat{L}(k))\|^2 \leq \sigma^2_W F^{-1}_{\chi^2_l}(K/\binom{n_k}{n_o})$$

.

A.7 Utilization of different attack blocks

We may choose the attack staging blocks independently of each other. Each remark holds two examples:

Remark 1 (Leakage model selection). e.g. 1)Assume leakage is additive across the different XOR gates, and switch random variables depending on the values of the inputs/ output $L = G_0 + \sum_i k_i \mathbf{G}^1_{i,k_i} + r_i \mathbf{G}^2_{i,r_i} + (k_i \oplus r_i)\mathbf{G}^3_{i,k_i \oplus r_i}$

e.g. 2) Assume there is leakage parity between the different gates and a temperature dependence, and profile them only based on $\mathbf{k}$ output $L = G_0(temp) + \sum_{i,j} k_i k_j \mathbf{G}_{i,j,k_i \cdot k_j}$

Remark 2 (Feature Selection). e.g. 1) The mean estimator $\theta(X) = \frac{1}{n}\sum_m \ell_{\mathbf{m}}$

e.g. 2) The covariance estimator $\theta(X) = \frac{1}{n-1}\sum_m \ell_{\mathbf{m}} \cdot \ell^T_{\mathbf{m}}$

Remark 3 (Complexity Reduction). e.g. 1) Guess against Σ^L

e.g. 2) Bias/ Variance considerations - As discussed in Appendix A.2

A.8 Naive feature estimation attack on scenario (1)

1. Sample profiling $\mathbf{P_i}$ and attack sets $\mathbf{A_i}$.
2. Create the model via linear regression $(M_0, M_1, M_2) \cdot (1, \mathbf{k_i}, \mathbf{p_i}, \mathbf{k_i} \oplus \mathbf{p_i}) \leftarrow \mathbf{P_i}$.
3. Preprocess the attack traces $\mathbf{A_i}' \leftarrow \mathbf{A_i} - M[:, 1] - (M_1 + M_2) \cdot \mathbf{p_i}$
4. calculate the feature $\theta(\mathbf{A}) = E_i(\mathbf{A_i}')$
5. Calculate the attack covariances $\Sigma^L = COV_i(\mathbf{A_i}')$.
6. Define $M'_{t,i} = (M_0)_{t,i} + [1 - 2E_j(p_{j,i})](M_2)_{t,i}$
7. Find the fitting keys $\min \arg_{\mathbf{k}} \|M' \cdot \mathbf{k} - \theta(\mathbf{A})\|_{\Sigma_L}$

This approach is the closest one to the one used in Sec. 6.1, but it does not leverage our ability to choose $\mathbf{p}$'s distribution by sub-sampling. We can see this in Tab. 12 where the results are sub-par. This is in part because we have no statistics on Σ^L (could be fixed via profiling), but mostly because we do not use the full amount of information $\mathbf{p}$ provides via this type of attack.

Table 12: Guessing entropy of adversaries on scenario 1 from Fig. 1 for the mean estimator.

Configuration	**# Attack Traces n**					
	25	**50**	**100**	**250**	**500**	**1000**
16-bit (T=60K, #Keys=50)	12.75	12.19	11.2	11.03	9.8348	9.74

A.9 MIX-COLUMN'S pseudo-code and labeling boolean function

Algorithm 2 describes the block MIX-COLUMN block in AES, as implemented in tiny-aes-c [kok19]. Equation (28) describes the set of intermediate values in Algorithm 2 as a function of the input.

Algorithm 2 MACRO($\mathbf{k}$)

Input $\mathbf{k} \in \mathbb{B}^{32}$ as four 1 byte variables, indexed [i], $i \in \{0, \ldots 3\}$.
$t \leftarrow k[0]$
$Tmp \leftarrow k[0] \oplus k[1] \oplus k[2] \oplus k[3]$
$Tm \leftarrow k[0] \oplus k[1]$
$Tm \leftarrow \text{XTIME}(Tm)$
$k[0] \leftarrow k[0] \oplus Tm \oplus Tmp$
$Tm \leftarrow k[1] \oplus k[2]$
$Tm \leftarrow \text{XTIME}(Tm)$
$k[1] \leftarrow k[1] \oplus Tm \oplus Tmp$
$Tm \leftarrow k[2] \oplus k[3]$
$Tm \leftarrow \text{XTIME}(Tm)$
$k[2] \leftarrow k[2] \oplus Tm \oplus Tmp$
$Tm \leftarrow k[3] \oplus t$
$Tm \leftarrow \text{XTIME}(Tm)$
$k[3] \leftarrow k[3] \oplus Tm \oplus Tmp$

$$
\begin{aligned}
f(k) =& \{x_0, x_1, x_2, x_3, x_{01}, x_{12}, x_{23}, x_{03}, x_{012}, x_{0123}, t_{01}, t_{12}, t_{23}, t_{03}, v_0, v_1, \\
& v_2, v_3, v_4, v_5, v_6, v_7\} \\
& x_0 = k[0], \quad x_1 = k[1], \quad x_2 = k[2], \quad x_3 = k[3], \\
& x_{01} = x_0 \oplus x_1, \quad x_{12} = x_1 \oplus x_2, \quad x_{23} = x_3 \oplus x_2, \quad x_{03} = x_0 \oplus x_3, \\
& x_{012} = x_0 \oplus x_1 \oplus x_2, \quad x_{0123} = x_0 \oplus x_1 \oplus x_2 \oplus x_3, \\
& t_{01} = \text{XTIME}(x_{01}), \quad t_{12} = \text{XTIME}(x_{12}), \quad t_{23} = \text{XTIME}(x_{23}), \\
& t_{03} = \text{XTIME}(x_{03}), v_0 = t_{01} \oplus x_{0123}, \quad v_1 = t_{01} \oplus x_{0123} \oplus x_0, \\
& v_2 = t_{12} \oplus x_{0123}, \quad v_3 = t_{12} \oplus x_{0123} \oplus x_1, \\
& v_4 = t_{23} \oplus x_{0123}, \quad v_5 = t_{23} \oplus x_{0123} \oplus x_2, \\
& v_6 = t_{03} \oplus x_{0123}, \quad v_7 = t_{03} \oplus x_{0123} \oplus x_3.
\end{aligned}
\tag{28}
$$

A.10 Additional Figures

A.10.1 Leakage variation over time

Fig. 12 clearly shows that there is a leakage trend. This trend may be caused by temperature drifts, power-delivery stability, etc.

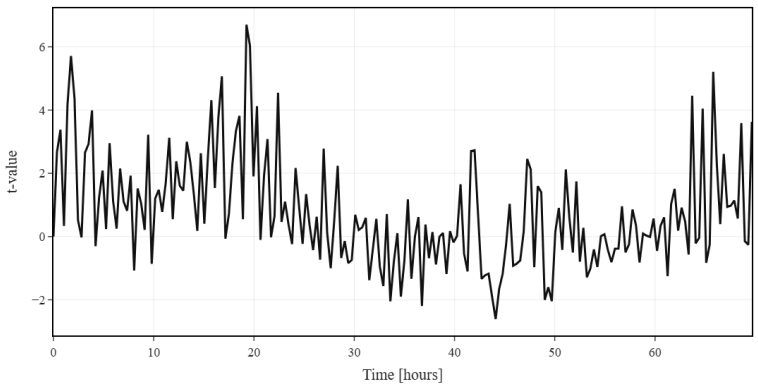


Figure 12: Leakage variation over time. Evaluated using the t-test between sets of leakages taken in different time points using the formula: $y(t) = E_j\left(t\text{-}val\left(\{\ell_{\mathbf{i},j}\}_{i=0}^{10,000}, \{\ell_{\mathbf{i},j}\}_{i=10,000t}^{10,000(t+1)}\right)\right)$ where $\ell_{\mathbf{i},j}$ is the jth time sample of the ith trace.

A.10.2 Additional results for attack on masking

Table 13: Additional attack configurations for guessing entropy of adversaries on Scenario 4 from Fig. 1 using the covariance estimator (non-swapped XOR).

Key Entropy	**Features used (d)**	**Attack Trace # (n)**	**Avg. Guessing Entropy**
8	500	40,000	0.4
16	250	100,000	1.04
16	500	100,000	1.42
16	2500	100,000	2.3
32	500	160,000	14.43
32	2500	160,000	11.6

A.10.3 Figures of Σ^L, Σ^Z for visualization

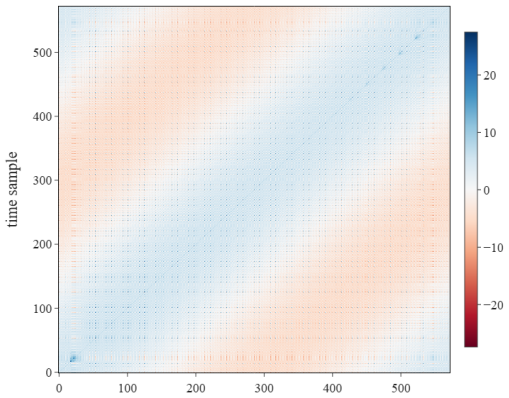


Figure 13: Visualization of Σ^L on $n_k = 32$ mean estimator

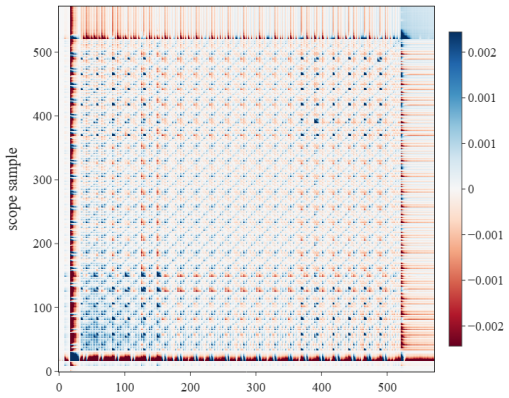


Figure 14: Visualization of Σ^Z on $n_k = 32$ mean estimator

In Fig. 13, we can clearly see how the trace noise is like a stationary random variable, correlating with nearby scope samples in waves. Fig. 14 is less intuitive, but it is clear from the structure (many repetitive evenly spaced squares) that there is at least one strong eigenvalue.

References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29, Berlin, Heidelberg, 2004. Springer.

[CGLS20] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and Franccois-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, 70(10):1677–1690, 2020.

[CJRR99] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer, 1999.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In *International Workshop on Fast Software Encryption*, pages 410–424. Springer, 2013.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International workshop on cryptographic hardware and embedded systems*, pages 13–28. Springer, 2002.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, cCetin K. Kocc, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28, Berlin, Heidelberg, 2003. Springer.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, 34(3):33, 2021.

[GBTP08a] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis: A generic side-channel distinguisher. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442, Berlin, Heidelberg, 2008. Springer.

[GBTP08b] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis: A generic side-channel distinguisher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008.

[GN06] Zhan Guo and Peter Nilsson. Algorithm and implementation of the K-best sphere decoding for MIMO detection. *IEEE Journal on Selected Areas in Communications*, 24(3):491–503, March 2006.

[HLVA02] Nicholas J Hopper, John Langford, and Luis Von Ahn. Provably secure steganography. In *Annual International Cryptology Conference*, pages 77–92. Springer, 2002.

[Hoe48] Wassily Hoeffding. A class of statistics with asymptotically normal distribution. *Annals of Mathematical Statistics*, 19(3):293–325, 1948.

[Hub64] Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[JWMB23] Vahid Jahandideh, Léo Weissbart, Bart Mennink, and Lejla Batina. A new leakage exploitation framework and its application to authenticated encryption. In *NIST Lightweight Cryptography Workshop*. NIST, jun 2023.

[kok19] kokke. tiny-AES-c: Small portable AES128/192/256 in C. GitHub repository, February 2019. Version v1.0.0 (released 2019-02-01), commit e72b6ef; licensed under the Unlicense/public domain.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[Man04] Stefan Mangard. Hardware countermeasures against dpa–a statistical analysis of their effectiveness. In *Cryptographers' Track at the RSA Conference*, pages 222–235. Springer, 2004.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, New York, NY, 2007.

[MS16] Amir Moradi and Franccois-Xavier Standaert. Moments-correlating dpa. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, pages 5–15, 2016.

[OC14] Colin O'Flynn and Zhizhang Chen. Chipwhisperer: An open-source platform for hardware embedded security research. volume 8622, 04 2014.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 142–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[RSVC09] Mathieu Renauld, Franccois-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the aes: Why time also matters in dpa. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 97–111. Springer, 2009.

[RYJ+25] Azade Rezaeezade, Trevor Yap, Dirmanto Jap, Shivam Bhasin, and Stjepan Picek. Breaking the blindfold: Deep learning-based blind side-channel analysis. In *Proceedings of the 34th USENIX Security Symposium*. USENIX Association, 2025.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 30–46, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[WBD23] Ruize Wang, Martin Brisfors, and Elena Dubrova. A side-channel attack on a bitsliced higher-order masked CRYSTALS-kyber implementation. Cryptology ePrint Archive, Paper 2023/1042, 2023.