Efficient, Portable Template Attacks

Marios O. Choudary and Markus G. Kuhn

***Abstract*— Template attacks recover data values processed by tamper-resistant devices from side-channel waveforms, such as supply-current fluctuations (power analysis) or electromagnetic emissions. They first profile a device to generate multivariate statistics of the waveforms emitted for each of a set of known processed values, which then identify maximum-likelihood candidates of unknown processed values during an attack. We identify several practical obstacles arising in the implementation of template attacks, ranging from numerical errors to the incompatibility of templates across different devices, and propose and compare several solutions. We identify pooled covariance matrices and prior dimensionality reduction through Fisher's linear discriminant analysis as particularly efficient and effective, especially where many attack traces can be acquired. We evaluate alternative algorithms not only for the task of recovering key bytes from a hardware implementation of the Advanced Encryption Standard; we even reconstruct the value transferred by an individual byte-load instruction, with success rates reaching 85% (or a guessing entropy of less than a quarter bit remaining) after 1000 attack traces, thereby demonstrating direct eavesdropping of eight-bit parallel data lines. Using different devices during the profiling and attack phase can substantially reduce the effectiveness of template attacks. We demonstrate that the same problem can also occur across different measurement campaigns with the same device and that DC offsets (e.g., due to temperature drift) are a significant cause. We improve the portability of template parameters across devices by manipulating the DC content of the eigenvectors that form the projection matrix used for dimensionality reduction of the waveforms.**

***Index Terms*— Hardware security, side-channel attack, template attack, power analysis.**

I. Introduction

SIDE-CHANNEL attacks are powerful tools for inferring secret algorithms or data (passwords, cryptographic keys, etc.) processed inside tamper-resistant hardware, if an attacker can monitor some channel leaking such information out of the device, most notably the power-supply current and unintended electromagnetic emissions.

One of the most powerful techniques for exploiting side-channel information is the *template attack* [6], which relies on a multivariate model of the side-channel traces. While the basic algorithm is comparatively simple (Section II), there are several steps that can cause problems when implementing the attack in practice.

We examine some of the most common problems that can arise: dealing with a large number of voltage samples (Section III) and using different devices for profiling and attack (Section V). Some of the proposed solutions rely on using known statistical techniques (e.g. using a pooled covariance matrix, see Section III-B, or linear discriminant, see Section III-E), while others are based on new ways to use existing algorithms, such as Principal Component Analysis (PCA) or Fisher's Linear Discriminant Analysis (LDA) (Section V-B.1).

We evaluate our methods using an ATMEL AVR XMEGA 8-bit microcontroller, which has an AES hardware cryptographic core. Our results show that, by using the methods presented in this paper, we can improve considerably the success of template attacks in different practical scenarios (Section IV), compared to a classic implementation of these attacks (Section II). These results should be useful to both evaluators and designers of secure microcontrollers.

This journal article builds on two previous conference papers by the authors: *Efficient Template Attacks* [28] and *Template Attacks on Different Devices* [30]. We include the main results from those two papers to obtain a self-contained manuscript, along with the following improvements: (*a*) clarified likelihood computation in Section II; (*b*) updated notation for consistency; (*c*) review of more recent literature [32]–[35] (Sections III-A, III-C.3 and V-B.3); (*d*) expanded Section III-C (Compression Methods) to provide more details about the computation of the sample selection methods, the alternative PCA method, and Fisher's LDA coefficients; (*e*) merged the entire derivation of the linear discriminant into a single section, so that it is easier to follow and understand; (*f*) simplified Figure 3, for easier comparison of results; (*g*) added Section IV-C to demonstrate that a DPA-style template attack (especially with LDA compression) is a much less challenging problem on the same data set; and (*h*) added new results from a hardware AES engine (Section IV-D).

II. Template Attacks

To implement a template attack, we need physical access to a pair of identical devices, which we refer to as the *profiling* and the *attacked* device. We wish to infer some secret value $k\star \in \mathcal{S}$, processed by the attacked device at some point. For an 8-bit microcontroller, $\mathcal{S} = \{0, \ldots, 255\}$ might be the set of possible byte values manipulated by a particular machine instruction.

We assume that we determined the approximate moments of time when the secret value $k\star$ is manipulated and we are able

Manuscript received November 2, 2016; revised June 23, 2017; accepted September 11, 2017. Date of publication September 27, 2017; date of current version November 28, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Yiorgos Makris. *(Corresponding author: Markus G. Kuhn.)*

M. O. Choudary is with the Faculty of Computer Science, University Politehnica of Bucharest, 060042 Bucharest, Romania.

M. G. Kuhn is with the Department of Computer Science and Technology, University of Cambridge, Cambridge CB3 0FD, U.K. (e-mail: mgk25@cl.cam.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2757440

to record signal traces (e.g., supply current or electro-magnetic waveforms) around these moments. We refer to these traces as *leakage vectors*. Let $\{t_1, \ldots, t_{m^r}\}$ be the set of time *samples* and $\mathbf{x}^r \in \mathbb{R}^{m^r}$ be the random vector from which leakage traces are drawn.

During the *profiling* phase we record n_p leakage vectors $\mathbf{x}^r_{ki} \in \mathbb{R}^{m^r}$ $(1 \le i \le n_p)$ from the profiling device for each possible value $k \in \mathcal{S}$, and combine these as row vectors $\mathbf{x}^{r\prime}_{ki}$ in the leakage matrix $\mathbf{X}^r_k \in \mathbb{R}^{n_p \times m^r}$.[1]

Typically, the *raw* leakage vectors $\mathbf{x}^r_{ki}$ provided by the data-acquisition device contain a very large number m^r of samples (random variables), due to high sampling rates used. Therefore, we might *compress* them before further processing, either by selecting only a subset of $m \ll m^r$ of those samples, or by applying some other data-dimensionality reduction method, such as Principal Component Analysis (PCA) or Fisher's Linear Discriminant Analysis (LDA).

We refer to such compressed leakage vectors as $\mathbf{x}_{ki} \in \mathbb{R}^m$ and combine all of these as rows into the compressed leakage matrix $\mathbf{X}_k \in \mathbb{R}^{n_p \times m}$. (Without any such compression step, we would have $\mathbf{X}_k = \mathbf{X}^r_k$ and $m = m^r$.)

Using $\mathbf{X}_k$ we can compute the template parameters $\bar{\mathbf{x}}_k \in \mathbb{R}^m$ and $\mathbf{S}_k \in \mathbb{R}^{m \times m}$ for each possible value $k \in \mathcal{S}$ as

$$\bar{\mathbf{x}}_k = \frac{1}{n_p} \sum_{i=1}^{n_p} \mathbf{x}_{ki}, \tag{1a}$$

$$\mathbf{S}_k = \frac{1}{n_p - 1} \sum_{i=1}^{n_p} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)', \tag{1b}$$

where the sample mean $\bar{\mathbf{x}}_k$ and the sample covariance matrix $\mathbf{S}_k$ estimate the true mean $\mathrm{E}(\mathbf{x}_k)$ and true covariance $\mathrm{Cov}(\mathbf{x}_k)$ of random vector $\mathbf{x}_k$. Note that

$$\sum_{i=1}^{n_p} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)' = \widetilde{\mathbf{X}}'_k \widetilde{\mathbf{X}}_k, \tag{2}$$

where $\widetilde{\mathbf{X}}_k$ is $\mathbf{X}_k$ with $\bar{\mathbf{x}}'_k$ subtracted from each row, and the latter form can help with fast vectorized computation of the covariance matrices in (1b).

Side-channel leakage traces can generally be modeled well by a multivariate normal distribution [6], which we also observed in our experiments. In this case, the sample mean $\bar{\mathbf{x}}_k$ and sample covariance $\mathbf{S}_k$ are *sufficient statistics*: they completely define the underlying distribution [12, Ch. 4]. Then the probability density function (pdf) of a leakage vector $\mathbf{x}$, given the parameters $\bar{\mathbf{x}}_k$ and $\mathbf{S}_k$ for each k, is

$$f(\mathbf{x} \mid k) = \frac{1}{\sqrt{(2\pi)^m |\mathbf{S}_k|}} \cdot \mathrm{e}^{-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}}_k)' \mathbf{S}_k^{-1} (\mathbf{x} - \bar{\mathbf{x}}_k)}. \tag{3}$$

In the *attack* phase, we try to infer the secret value $k\star \in \mathcal{S}$ processed by the attacked device. We obtain n_a leakage vectors $\mathbf{x}_i \in \mathbb{R}^m$ from the attacked device, using the same recording technique and compression method as in the profiling phase, resulting in the leakage matrix $\mathbf{X}_{k\star} \in \mathbb{R}^{n_a \times m}$. Then, for each $k \in \mathcal{S}$, we compute a *discriminant score* $D(k \mid \mathbf{X}_{k\star})$. Finally, we try all $k \in \mathcal{S}$ on the attacked device, in order of decreasing score (optimized brute-force search, e.g. for a password or cryptographic key), until we find the correct $k\star$. Given a trace $\mathbf{x}_i$ from $\mathbf{X}_{k\star}$, a commonly used discriminant [9], [15], [18], derived from Bayes' rule, is

$$D(k \mid \mathbf{x}_i) = f(\mathbf{x}_i \mid k) P(k), \tag{4}$$

where the denominator from Bayes' rule

$$L(k \mid \mathbf{x}_i) = \frac{f(\mathbf{x}_i \mid k) P(k)}{\sum_{k'} f(\mathbf{x}_i \mid k') P(k')} \tag{5}$$

is omitted, as it is the same for each k. Assuming a uniform a-priori probability $P(k) = |\mathcal{S}|^{-1}$, applying Bayes' rule becomes equivalent to computing the likelihood

$$L(k \mid \mathbf{x}_i) = \frac{f(\mathbf{x}_i \mid k)}{\sum_{k'} f(\mathbf{x}_i \mid k')} \tag{6}$$

where f can be computed from (3).

III. Numerical Problems

We start our exposition of practical problems by showing the issues that appear when using a large number of samples m.

A. Inverse of Covariance Matrix

Several authors [18], [19] noted that inverting the covariance matrix $\mathbf{S}_k$ from (1b), as needed in (3), can cause numerical problems for large m, but it is important to understand why $\mathbf{S}_k$ can become singular ($|\mathbf{S}_k| \approx 0$), causing these problems. Since $\mathbf{S}_k$ is essentially the matrix product $\widetilde{\mathbf{X}}'_k \widetilde{\mathbf{X}}_k$ (2), both $\mathbf{S}_k$ and $\widetilde{\mathbf{X}}_k$ have the same rank. Therefore $\mathbf{S}_k$ is singular iff $\widetilde{\mathbf{X}}_k$ has dependent columns, which is guaranteed if $n_p < m$. The constraint on $\widetilde{\mathbf{X}}_k$ to have zero-mean rows implies that it has dependent columns even for $n_p = m$. Therefore, $n_p > m$ is a *necessary* condition for $\mathbf{S}_k$ to be non-singular. See [12, Result 3.3] for a more detailed proof.

The restriction $n_p > m$ is one main reason for reducing m through compression (see Section III-C). Note that in practice some samples can be highly correlated, in which case n_p needs to be somewhat larger than m (e.g., $n_p \ge 3000$ for $m = 1250$ with our Section IV-B data) just to ensure a non-singular $\mathbf{S}_k$, and we often need $n_p \gg m$ to make $\mathbf{S}_k$ an effective estimate of $\mathrm{Cov}(\mathbf{x}_k)$. (See [36] for an empirical relationship, in one setting, for good choices of m and n_p, depending on SNR.)

B. The Pooled Covariance

In our experiments, we observed that the particular covariance matrices $\mathbf{S}_k$ are very similar and seem to be independent of the candidate k. This means that the signal related to k is entirely contained in the mean vectors $\bar{\mathbf{x}}_k$, while the $\mathbf{S}_k$ model the measurement noise. In this case, we can use a pooled covariance matrix

$$\mathbf{S}_{\text{pooled}} = \frac{1}{|\mathcal{S}|(n_p - 1)} \sum_{k \in \mathcal{S}} \sum_{i=1}^{n_p} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)', \tag{7}$$

to obtain a much better estimate of the true covariance matrices $\mathrm{Cov}(\mathbf{x}_k)$. In this case, $\mathbf{S}_{\text{pooled}}$ is computed from $|\mathcal{S}| \cdot n_p$ traces. Hence, the necessary condition for $\mathbf{S}_{\text{pooled}}$ being invertible becomes $m < |\mathcal{S}| \cdot n_p$, or $n_p > \frac{m}{|\mathcal{S}|}$, which is easier to satisfy than when using individual covariance matrices.

[1] Throughout this paper $\mathbf{x}'$ is the transpose of $\mathbf{x}$.

C. Compression Methods for Template Attacks

In order to deal with the problems mentioned above, we should also compress the leakage traces $\mathbf{x}_i^{\mathrm{r}} \in \mathbb{R}^{m^{\mathrm{r}}}$ into $\mathbf{x}_i \in \mathbb{R}^m$ ($m \ll m^{\mathrm{r}}$), in order to reduce the number of variables involved while at the same time keeping as much information as possible. It turns out that the choice of compression method is an essential step for the success of profiled attacks. The first proposed methods [6] relied on selecting some samples that maximise the data-dependent signal, but this can be error-prone. Later, Principal Component Analysis (PCA) [9] and Fisher's Linear Discriminant Analysis (LDA) [15] helped to maximise the information used in the attack step with a very small number m of samples. Below we briefly describe these methods in the context of template attacks. All of these approaches rely on the between-groups vectors

$$\tau_k = \bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}} \tag{8}$$

that define the signal of interest, where

$$\bar{\mathbf{x}}^{\mathrm{r}} = \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} \bar{\mathbf{x}}_k^{\mathrm{r}}. \tag{9}$$

1) Selection of Samples: In this method we first compute a signal-strength estimate $s(t)$, $t \in \{t_1, \ldots, t_{m^{\mathrm{r}}}\}$, and then we select a subset of m points based on this estimate.

There are several proposals for producing $s(t)$, such as *Difference of Means (DOM)* [6, Sec. 2.1], the *Sum of Squared Differences (SOSD)* [10], the *Signal-to-Noise Ratio (SNR)* [19] and *SOST* [10]. All these are similar, with the notable difference that the first two do not take the variance of the traces into consideration, while the latter two do.

The DOM method was first proposed by Chari *et al.* [6, Sec. 2.1], to select samples at which large pairwise differences between the means show up. Later, Rechberger and Oswald [8, Sec. 3.2] explicitly suggested to sum these pairwise differences and then select the samples from the traces with largest peaks. Gierlichs *et al.* [10, Sec. 2.1] observed that using the sum is not appropriate, proposing the sum of squared differences (SOSD) instead.

We found that the sum of the absolute value of pairwise differences

$$s_{\mathrm{DOM}}(t) = \sum_{1 \le k < k' < |\mathcal{S}|} |\bar{\mathbf{x}}_k^{\mathrm{r}}(t) - \bar{\mathbf{x}}_{k'}^{\mathrm{r}}(t)| \tag{10}$$

gives very good results, which is what we refer to as DOM from now on.[2] Here, $\bar{\mathbf{x}}_k^{\mathrm{r}}$ are the mean vectors, as in (1a), but calculated from the raw leakage vectors $\mathbf{x}_{ki}^{\mathrm{r}}$.

In Figure 1, we show these estimates for the *Grizzly Beta* dataset (see Section IV). The methods SNR and SOST are in fact the same if we consider the variance at each sample point to be independent of the candidate k, which is expected in our setting. Under this condition SNR and SOST reduce to computing the F-score

$$F(t) = \frac{\left(n_{\mathrm{p}} \sum_{k \in \mathcal{S}} (\bar{\mathbf{x}}_k(t) - \bar{\mathbf{x}}(t))^2 \right) / (|\mathcal{S}| - 1)}{\left(\sum_{k \in \mathcal{S}} \sum_{i=1}^{n_{\mathrm{p}}} (\mathbf{x}_{ki}(t) - \bar{\mathbf{x}}_k(t))^2 \right) / (|\mathcal{S}|(n_{\mathrm{p}} - 1))} \tag{11}$$

as used in the Analysis of Variance (ANOVA) [12, Sec. 6.4]. $F(t)$ can be used to reject, at any desired significance level, the hypothesis that the sample mean values at sample point t are equal, therefore providing a good indication in which samples the variation is more related to k.[3]

In the second step of this compression method we need to choose m samples based on the signal-strength estimate s. The goal is to select the smallest set of samples that contains most of the information about our target. In our experiments we used the DOM estimate with different selections of samples (see Section IV) and concluded that we should select several samples per clock cycle [28], as long as we can use the pooled covariance matrix.

2) Principal Component Analysis (PCA): Archambeau *et al.* [9] proposed the following method for using PCA as a compression method for template attacks. First compute the *sample between groups* matrix $\mathbf{B}$:

$$\mathbf{B} = \sum_{k \in \mathcal{S}} (\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}})(\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}})'. \tag{12}$$

Next obtain the singular value decomposition (SVD) $\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{U}'$, where each column of $\mathbf{U} \in \mathbb{R}^{m^{\mathrm{r}} \times m^{\mathrm{r}}}$ is an eigenvector $\mathbf{u}_j$ of $\mathbf{B}$, and $\mathbf{D} \in \mathbb{R}^{m^{\mathrm{r}} \times m^{\mathrm{r}}}$ contains the corresponding eigenvalues δ_j on its diagonal.[4] The crucial point is that only the first $m \ll |\mathcal{S}|$ eigenvectors $(\mathbf{u}_1 \ldots \mathbf{u}_m) = \mathbf{U}^m$ are needed in order to preserve most of the variability from the mean vectors $\bar{\mathbf{x}}_k^{\mathrm{r}}$. Therefore, we can restrict $\mathbf{U}$ to $\mathbf{U}^m \in \mathbb{R}^{m^{\mathrm{r}} \times m}$. Finally, we can project the mean vectors $\bar{\mathbf{x}}_k^{\mathrm{r}}$ and covariance matrices $\mathbf{S}_k^{\mathrm{r}}$ (computed with (1) on the raw traces $\mathbf{x}_i^{\mathrm{r}}$) into the new coordinate system defined by $\mathbf{U}^m$ to obtain the PCA

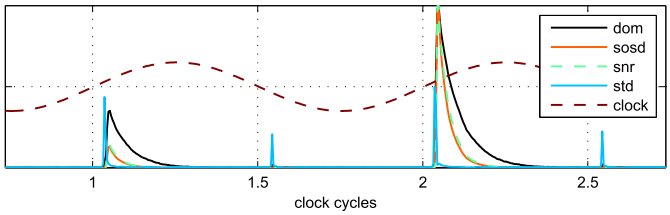

Fig. 1. Signal-strength estimates from DOM, SOSD and SNR (identical to SOST) on the *Grizzly Beta* dataset, along with the average standard deviation (STD) of the traces ($n_{\mathrm{p}} = 2000$) and clock signal. All estimates are rescaled to fit into the plot, so the vertical axis (linear) has no scale.

[2] Using SNR instead of DOM as the signal-strength estimate $s(t)$ provided very similar results.

[3] For an example of using the F-score with the *Grizzly* dataset, see Choudary's PhD thesis [31, p. 67].

[4] Since $\mathbf{B} = \mathbf{T}\mathbf{T}'$ with $\mathbf{T} = \left((\bar{\mathbf{x}}_1^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}}) \; (\bar{\mathbf{x}}_2^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}}) \; \cdots \; (\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}}) \right) \in \mathbb{R}^{m^{\mathrm{r}} \times |\mathcal{S}|}$ has rank less than $|\mathcal{S}|$, it is sufficient to compute what some numeric libraries call the "thin" or "economy size" singular-value decomposition $\mathbf{T} = \mathbf{V}\Delta\mathbf{W}'$ where $\mathbf{V} \in \mathbb{R}^{m^{\mathrm{r}} \times \upsilon}$ with $\upsilon = \min\{m^{\mathrm{r}}, |\mathcal{S}|\}$ contains the left-singular vectors of $\mathbf{T}$ as columns, which are orthonormal eigenvectors of $\mathbf{B}$, and where diagonal matrix $\Delta \in \mathbb{R}^{\upsilon \times \upsilon}$ contains the corresponding square roots of the eigenvalues of $\mathbf{B}$. (In our experiments with $m^{\mathrm{r}} = 2500$ the direct computation of the SVD on the matrix $\mathbf{B}$ also worked well.)

template parameters $\bar{\mathbf{x}}_k \in \mathbb{R}^m$ and $\mathbf{S}_k \in \mathbb{R}^{m \times m}$:

$$\bar{\mathbf{x}}_k = \mathbf{U}^{m\prime}\bar{\mathbf{x}}_k^{\mathrm{r}},\ \mathbf{S}_k = \mathbf{U}^{m\prime}\mathbf{S}_k^{\mathrm{r}}\mathbf{U}^m. \tag{13}$$

3) Choice of PCA Components: Archambeau *et al.* [9] proposed to select only those first m eigenvectors $\mathbf{u}_j$ for which the corresponding eigenvalues δ_j are a few orders of magnitude larger than the rest. This technique, also known as *elbow rule* or *Scree Graph* [7], requires manual inspection of the eigenvalues. A more algorithmic technique uses the *Cumulative Percentage of Total Variation* [7]. It selects those first m eigenvectors that retain at least fraction f of the total variance, by computing the score

$$\phi(m) = \frac{\sum_{1 \le j \le m} \delta_j}{\sum_{1 \le j \le m^{\mathrm{r}}} \delta_j}, \quad 1 \le m \le m^{\mathrm{r}}, \tag{14}$$

and selecting the lowest m for which $\phi(m) > f$.[5]

Eigenvector features can also help to chose the right projection axes. In Section V-B.1 we demonstrate the significance of the DC component (average voltage) of PCA eigenvectors $\mathbf{u}_j$. The idea that useful eigenvectors are likely to show localized peaks (see middle of Figure 9 later on) has also been explored to construct selection features [24], [33]. The *Explained Local Variance* (ELV) method proposed by Cagli *et al.* [33] calculates for each sample time t in each PCA eigenvector $\mathbf{u}_j$ the value

$$\mathrm{ELV}(\mathbf{u}_j, t) = \frac{\delta_j \mathbf{u}_j^2(t)}{\sum_{j'=1}^{m^{\mathrm{r}}} \delta_{j'}}, \tag{15}$$

to then compute the sum $\sigma_j = \sum_{t'=1}^{q} \mathrm{sort}_t\{\mathrm{ELV}(\mathbf{u}_j, t)\}(t')$ of the q largest ELV values of each eigenvector $\mathbf{u}_j$. The σ_j are essentially the eigenvalues δ_j weighed with a measure of the peakiness of the eigenvectors. We tried this measure on our datasets (with $q = 25$), but the top four σ_j values selected the same $\mathbf{u}_j$ as the top eigenvalues δ_j, so we were unable to confirm an improvement. Ultimately, "*there is no definitive answer [to the question of how many components to choose]*" [12, Ch. 8].

4) Alternative Computation of PCA Templates: Standaert and Archambeau [15, Sec. 4.1] mention that PCA can help where computing the full covariance matrix $\mathbf{S}_k^{\mathrm{r}}$ is prohibitive, due to large m^{r}. However, their approach still requires the computation of $\mathbf{S}_k^{\mathrm{r}}$, see (13). Also, numerical artifacts during the double matrix multiplication in (13) can make $\mathbf{S}_k$ non-symmetric. Both problems can be avoided: rather than first computing $\mathbf{S}_k^{\mathrm{r}}$ and then applying (13), we can first compute the projected leakage matrix

$$\mathbf{X}_k = \mathbf{X}_k^{\mathrm{r}}\mathbf{U}^m \tag{16}$$

and then compute the PCA-based template parameters using (1). The result is equivalent since $\mathrm{Cov}(\mathbf{x}_k^{\mathrm{r}}\mathbf{U}^m) = \mathbf{U}^{m\prime}\mathrm{Cov}(\mathbf{x}_k^{\mathrm{r}})\mathbf{U}^m$ [12, eq. (3)–(45)].

[5]In our experiments, for $f = 0.95$ and $n_{\mathrm{p}} < 1000$ this method retained the $m = 4$ largest components, which correspond to the same components that we had selected using the elbow rule. However, when $n_{\mathrm{p}} > 1000$ the number of components needed for $f \ge 0.95$ decreased to $m < 4$, which led to worse results of the template attack.

5) Fisher's Linear Discriminant Analysis (LDA): Given a random vector of leakage traces $\mathbf{x}^{\mathrm{r}}$, applying Fisher's idea [2], [12] means to consider projections $y_j = \mathbf{a}_j{}'\mathbf{x}^{\mathrm{r}}$ and find directions (vectors of coefficients) $\mathbf{a}_j \in \mathbb{R}^{m^{\mathrm{r}}}$ along which the ratio

$$\frac{\sum_{k \in \mathcal{S}} \left(\mathrm{E}\left(y_{jk}\right) - \mathrm{E}\left(y_j\right)\right)^2}{\sum_{k \in \mathcal{S}} \mathrm{Var}\left(y_{jk}\right)} = \frac{\sum_{k \in \mathcal{S}} \left(\mathbf{a}_j{}'\left(\mathrm{E}\left(\mathbf{x}_k^{\mathrm{r}}\right) - \mathrm{E}\left(\mathbf{x}^{\mathrm{r}}\right)\right)\right)^2}{\sum_{k \in \mathcal{S}} \mathrm{Var}\left(\mathbf{a}_j{}'\mathbf{x}_k^{\mathrm{r}}\right)}$$

of the between-groups variance to the within-groups variance is maximised (where $\mathbf{x}_k^{\mathrm{r}}$ with $y_{jk} = \mathbf{a}_j{}'\mathbf{x}_k^{\mathrm{r}}$ is the same random vector within group k).

Given leakage traces $\mathbf{x}_{ki}^{\mathrm{r}}$ (rows of $\mathbf{X}_k^{\mathrm{r}}$), this ratio (times $|\mathcal{S}| - 1$) can be estimated as

$$\frac{|\mathcal{S}|(n_{\mathrm{p}} - 1)\sum_{k \in \mathcal{S}} (\mathbf{a}_j{}'(\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}}))^2}{\sum_{k \in \mathcal{S}} \sum_{i=1}^{n_{\mathrm{p}}} \mathbf{a}_j{}'(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)'\mathbf{a}_j} = \frac{\mathbf{a}_j{}'\mathbf{B}\mathbf{a}_j}{\mathbf{a}_j{}'\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\mathbf{a}_j}, \tag{17}$$

where $\mathbf{B}$ is the sample between-groups matrix from (12) and $\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}} = \frac{1}{|\mathcal{S}|}\sum_{k \in \mathcal{S}} \mathbf{S}_k^{\mathrm{r}}$ is the common covariance of all candidates (see also Section III-B). Note the similarity between the left hand side of (17) and the F-test (11), SNR and SOST. While in the sample selection method we first compute (11) for each sample and then select the samples with the highest $F(t)$, Fisher's method finds the linear combinations of the trace samples that maximise (17).

The coefficients $\mathbf{a}_j$ that maximise (17), subject to the constraint that $\mathrm{Cov}(y_{ik}, y_{jk}) = 0$ for $i \ne j$, are the eigenvectors of $\left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-1}\mathbf{B}$ with the largest eigenvalues.[6]

As with PCA, we only need to use the first m coefficients $\mathbf{a}_1, \ldots, \mathbf{a}_m$, which can be selected using the same rules as for PCA. If we let $\mathbf{A} = (\mathbf{a}_1 \ldots \mathbf{a}_m)$ be the matrix of coefficients, we can project each leakage matrix as:

$$\mathbf{X}_k = \mathbf{X}_k^{\mathrm{r}}\mathbf{A} \tag{18}$$

and compute the LDA-based template parameters using (1).

Several authors [15], [18] have used Fisher's LDA for template attacks, but without mentioning two important aspects. Firstly, the condition of equal covariances (known as *homoscedasticity*) may be important for the success of Fisher's LDA. Therefore, the PCA method, which does not depend on this condition, might be a better choice in some settings. Secondly, the coefficients that maximise (17) can be obtained using scaled versions of $\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}$ or different approaches [15], [18], which will result in a different scale

[6]Note that $\left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-1}\mathbf{B}$ is not necessarily symmetric, so we cannot directly apply singular-value decomposition to obtain orthonormal eigenvectors. Instead, we can first compute the eigenvectors $\mathbf{u}_j$ of the symmetric matrix $\left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-\frac{1}{2}}\mathbf{B}\left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-\frac{1}{2}}$, which has the same eigenvalues as $\left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-1}\mathbf{B}$ [12, Exercise 11.21], and from which we can then obtain the coefficients $\mathbf{a}_j = \left(\mathbf{S}_{\mathrm{pooled}}^{\mathrm{r}}\right)^{-\frac{1}{2}}\mathbf{u}_j$. There are a maximum of $s = \min(m^{\mathrm{r}}, |\mathcal{S}| - 1)$ non-zero eigenvectors, as that is the maximum number of independent linear combinations available in $\mathbf{B}$.

of the coefficients $\mathbf{a}_j$. This has a major impact on the template attack: *only* when we scale the coefficients $\mathbf{a}_j$, such that $\mathbf{a}_j{}'\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}\mathbf{a}_j = 1$, the covariance between discriminants becomes the identity matrix [12], i.e. $\mathbf{S}_k = \mathbf{I}$. In this case the sample means in (1) suffice and we can discard the covariance matrix from the discriminant scores in Section III-E, which greatly reduces computation and storage requirements. The method we presented above to obtain the coefficients $\mathbf{a}_j$ guarantees this condition.

D. Floating-Point Limitations

A second practical problem with (3) is that for large m the statistical distance

$$(\mathbf{x} - \bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x} - \bar{\mathbf{x}}_k)$$

can cause the subsequent exponentiation operation to overflow. For example, in IEEE double precision, e^x is only safe with $|x| < 710$, easily exceeded for large m.

Another problem is that for large m the determinant $|\mathbf{S}_k|$ can overflow. For example, considering that $|\mathbf{S}_k|$ is the product of the eigenvalues of $\mathbf{S}_k$, in some of our experiments the 100 largest eigenvalues were at least 10^6 and multiplying merely 52 such values again overflows the IEEE double-precision format.

E. The Linear Discriminant

Such numerical issues can be avoided by replacing the multivariate pdf (3) with an equivalent discriminant score.

A first step is to use the logarithm of the multivariate normal distribution:

$$\log f(\mathbf{x} \mid k) = -\frac{m}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{S}_k| - \frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}-\bar{\mathbf{x}}_k), \tag{19}$$

where we compute the logarithm of the determinant as

$$\log|\mathbf{S}_k| = 2\sum_{i=1}^{m}\log c_{ii}, \tag{20}$$

using the Cholesky decomposition $\mathbf{S}_k = \mathbf{C}'\mathbf{C}$ of the symmetric matrix $\mathbf{S}_k$. (Since $\mathbf{C}$ is a triangular matrix, its determinant is the product of its diagonal elements c_{ii}.)

By dropping the first term, which is constant across all k, we obtain a discriminant score based on the log-likelihood:

$$\begin{aligned} D_{\log}(k \mid \mathbf{x}_i) &= -\frac{1}{2}\log|\mathbf{S}_k| - \frac{1}{2}(\mathbf{x}_i - \bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}_i - \bar{\mathbf{x}}_k) \\ &= \log f(\mathbf{x}_i \mid k) + \frac{m}{2}\log 2\pi \\ &= \log L(k \mid \mathbf{x}_i) + \text{const.} \end{aligned} \tag{21}$$

Using $\mathbf{S}_{\mathrm{pooled}}$, we can discard the first two terms in (19) and use the generalized statistical distance

$$d_{\mathrm{M}}^2(\mathbf{x}, \bar{\mathbf{x}}_k) = (\mathbf{x} - \bar{\mathbf{x}}_k)'\mathbf{S}_{\mathrm{pooled}}^{-1}(\mathbf{x} - \bar{\mathbf{x}}_k) \geq 0, \tag{22}$$

also known as the *Mahalanobis distance* [1], to compare the candidates k. The inequality in (22) holds because the covariance matrix is positive semidefinite. From (19,22) we can derive the discriminant score

$$\begin{aligned} D_{\mathrm{md}}(k \mid \mathbf{x}_i) &= -\frac{1}{2}d_{\mathrm{M}}^2(\mathbf{x}_i, \bar{\mathbf{x}}_k) \\ &= D_{\log}(k \mid \mathbf{x}_i) + \text{const.}, \end{aligned} \tag{23}$$

where the constant does not vary with k.

We can then rewrite the Mahalanobis distance (22) as

$$d_{\mathrm{M}}^2(\mathbf{x}, \bar{\mathbf{x}}_k) = \mathbf{x}'\mathbf{S}_{\mathrm{pooled}}^{-1}\mathbf{x} - 2\bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\mathbf{x} + \bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\bar{\mathbf{x}}_k, \tag{24}$$

and observe that the first term is constant for all candidates k, so we can discard it. That means we can now use the score

$$\begin{aligned} D_{\mathrm{linear}}(k \mid \mathbf{x}_i) &= \bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\mathbf{x}_i - \frac{1}{2}\bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\bar{\mathbf{x}}_k \\ &= D_{\mathrm{md}}(k \mid \mathbf{x}_i) + \text{const.}, \end{aligned} \tag{25}$$

which depends *linearly* on $\mathbf{x}_i$ (const. does not depend on k).

Although equivalent (for comparing single bytes), the linear discriminant D_{linear} avoids most of the numerical issues associated with the multivariate normal distribution. Furthermore, it can be far more efficient to compute than the quadratic log-likelihood discriminant D_{md}. To see this, compare the discriminants when used to combine n_{a} attack traces (essential for the success of many side-channel attacks) by computing the joint likelihood

$$L(k \mid \mathbf{X}_{k\star}) = \prod_{\mathbf{x}_i \text{ in } \mathbf{X}_{k\star}} L(k \mid \mathbf{x}_i) \tag{26}$$

or equivalently the joint log-likelihood

$$\log L(k \mid \mathbf{X}_{k\star}) = \sum_{i=1}^{n_{\mathrm{a}}} \log L(k \mid \mathbf{x}_i) \tag{27}$$

which leads us to these joint scores:

$$D_{\log}(k \mid \mathbf{X}_{k\star}) = -\frac{n_{\mathrm{a}}}{2}\log|\mathbf{S}_k| - \frac{1}{2}\sum_{i=1}^{n_{\mathrm{a}}}(\mathbf{x}_i - \bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}_i - \bar{\mathbf{x}}_k), \tag{28}$$

$$D_{\mathrm{md}}(k \mid \mathbf{X}_{k\star}) = -\frac{1}{2}\sum_{i=1}^{n_{\mathrm{a}}}(\mathbf{x}_i - \bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}_i - \bar{\mathbf{x}}_k), \tag{29}$$

$$D_{\mathrm{linear}}(k \mid \mathbf{X}_{k\star}) = \bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\left(\sum_{i=1}^{n_{\mathrm{a}}}\mathbf{x}_i\right) - \frac{n_{\mathrm{a}}}{2}\bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\bar{\mathbf{x}}_k. \tag{30}$$

Given n_{a} leakage traces $\mathbf{x}_i$ (the rows of $\mathbf{X}_{k\star}$), $D_{\log}$ requires time $O(n_{\mathrm{a}}m^2)$ while D_{linear} only requires $O(n_{\mathrm{a}}m + m^2)$, since the operations $\bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}$ and $\bar{\mathbf{x}}_k'\mathbf{S}_{\mathrm{pooled}}^{-1}\bar{\mathbf{x}}_k$ only need to be done once. This is a great advantage in practice: for example, our evaluations of the guessing entropy (see Section IV) for $m = 125$ and $1 \leq n_{\mathrm{a}} \leq 1000$ took about 3.5 days with $D_{\log}$ but only 30 minutes with D_{linear}.[7]

IV. Results on a Single Device

In this section, we present the framework used to evaluate the success of template attacks, comparing the methods (compression, linear discriminant) from Section III in the context of a single device. This framework, as well as the datasets presented here, will also be used in the context of different devices (Section V).

[7] MATLAB, single core CPU with 3794 MIPS.

A. Guessing Entropy

We are interested in evaluating the overall *practical* success of the template attacks. For this, we consider the number of guesses that an optimized brute-force search has to perform, which tries candidate values $k \in \mathcal{S}$ in order of decreasing score $D(k \mid \mathbf{X}_{k\star})$ until finding the correct value $k\star$. The expected value of this number can be used to calculate an upper bound for the Shannon entropy of the remaining uncertainty about the target value [3]. It is commonly called "guesswork" [5] or "guessing entropy" [4], [14], [17]. The lower the guessing entropy, the more successful the attack has been and the less search effort remains to find the correct $k\star$.

To estimate this guessing entropy, we average the rank of the score $D(k\star \mid \mathbf{X}_{k\star})$ of the correct candidate over all $k\star \in \mathcal{S}$. In detail, we compute the score $D(k \mid \mathbf{X}_{k\star})$ (either $D_{\log}$ or D_{linear}) for each combination of candidate value k and target value $k\star$, resulting in a score matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ with $\mathbf{M}(k\star, k) = D(k \mid \mathbf{X}_{k\star})$. Each row in $\mathbf{M}$ contains the score of each candidate value k given the traces $\mathbf{X}_{k\star}$ corresponding to a given target value $k\star$. Next we sort each row of $\mathbf{M}$, in decreasing order, to obtain a rank matrix $\mathbf{R} \in \{1, \ldots, |\mathcal{S}|\}^{|\mathcal{S}|\times|\mathcal{S}|}$ with

$$\mathbf{R}(k\star, k) = \text{position of } D(k \mid \mathbf{X}_{k\star}) \text{ in the sorted row of } \mathbf{M}(k\star, \cdot). \tag{31}$$

Using the rank matrix $\mathbf{R}$, we calculate the guessing entropy for a particular set $E = \{\mathbf{X}_{k\star}\}_{k\star\in\mathcal{S}}$ of $|\mathcal{S}| \cdot n_{\text{a}}$ evaluation attack traces as

$$G(E) = \frac{1}{|\mathcal{S}|} \sum_{k\in\mathcal{S}} \mathbf{R}(k, k). \tag{32}$$

As the guessing entropy is nicer to plot on a logarithmic scale, we usually state its binary logarithm $g = \log_2 G$, and indicate this with the suffix "bits". A random score would result in a guessing entropy of $g = \log_2(|\mathcal{S}| + 1) - 1$ bits. Below, we compute the guessing entropy for 10 random trace selections E and plot the average over these as a function of n_{a}.

B. Results on 8-Bit Microcontroller Load Instruction

Our first target is the data bus of the Atmel XMEGA 256 A3U, an easily available 8-bit microcontroller without side-channel countermeasures, mounted on our own evaluation board to monitor the total current in all CPU ground pins via a 10 Ω resistor. We powered it from a battery via a 3.3 V regulator and supplied a 1 MHz sine clock (see Figure 2, left). We used a Tektronix TDS 7054 8-bit oscilloscope with P6243 active probe, at 250 MS/s, with 500 MHz bandwidth in SAMPLE mode. For this evaluation we used the same device for both the profiling and the attack phase.

For each candidate value $k \in \{0, \ldots, 255\}$ we recorded 3072 traces $\mathbf{x}^{\text{r}}_{ki}$ (i.e., 786 432 traces in total), which we divided into a *training* set (for the profiling phase) and an *evaluation* set (for the attack phase). Each trace contains $m^{\text{r}} = 2500$ samples, recorded while the target microcontroller executed the same sequence of instructions loaded from the same addresses: a MOV instruction, followed by several LOAD instructions (see Figure 2, right). We changed the candidate byte k in random order during these measurements, to avoid it correlating with any environmental changes, such as temperature. All the LOAD instructions require two clock cycles to transfer a value from RAM into a register, using indirect addressing. We shall refer to the traces from this experiment as the *Grizzly* dataset [29]. In our experiments with this dataset, our goal was to determine the success of the template attacks in recovering the byte k processed by the second LOAD instruction. All the other instructions were processing the value zero, meaning that in our traces none of the variability should be caused by variable data in other nearby instructions that may be processed concurrently in various pipeline stages.[8]

We compare in Figure 3 the guessing entropy remaining after template attacks using either the logarithm variant of the multivariate normal distribution discriminant with

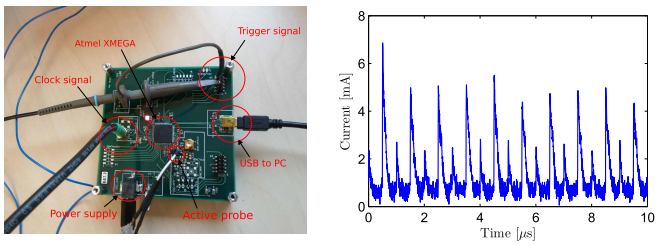

Fig. 2. Left: the device used during our experiments. Right: A single example trace $\mathbf{x}^{\text{r}}_i$ from our experimental setup.

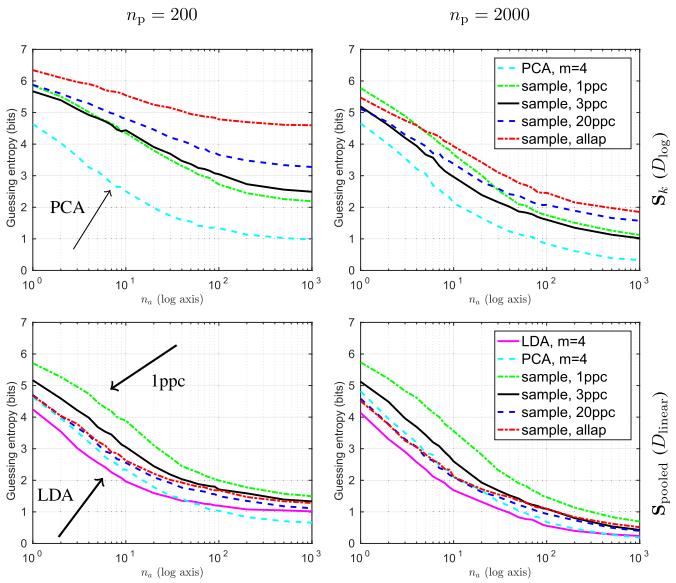

Fig. 3. Guessing entropy remaining after template attacks, with different compressions, for $n_{\text{p}} = 200$ (left) and $n_{\text{p}} = 2000$ (right) profiling traces, using individual covariances $\mathbf{S}_k$ with $D_{\log}$ (top) or a pooled covariance $\mathbf{S}_{\text{pooled}}$ with D_{linear} (bottom).

[8] A similar approach was used by Standaert and Archambeau [15] and Oswald and Paar [21] to report results of template attacks on (part of) the key loading stage of a block cipher.

the individual covariance matrices $\mathbf{S}_k$ ($D_{\log}$ – top) or the linear discriminant with the pooled covariance matrix $\mathbf{S}_{\text{pooled}}$ (D_{linear} – bottom). In each case we show the high-compression methods, such as PCA, LDA (only works with a pooled covariance matrix, so $D_{\log}$ was not applicable here) and the sample selections using one sample per clock at most (*1ppc*, $m \approx 8$), as well as a larger number of samples: the *3ppc* ($m \approx 25$), *20ppc* ($m \approx 77$) and *allap* ($m \approx 125$) selections.[9] We can see that the pooled covariance greatly improves the overall results[10] and that in general the methods PCA and LDA provide the best results.

Another important observation is that with the pooled covariance matrix and $n_{\text{p}} = 2000$, the guessing entropy drops to just 0.25 bits for $n_{\text{a}} = 10^3$, where in 85% of all attacks the maximum-likelihood candidate was the correct one. In other words, we are able to differentiate well between the power consumption of each individual bus line. These provide some of the best results of this kind to date, when comparing with other publications that mention or evaluate template attacks in a fixed-data (SPA) scenario [19], [25], [35].[11]

C. Results on a DPA-Style Attack

Targeting the byte value transferred by a simple LOAD instruction, as demonstrated above, is a substantially more difficult challenge than attacking a subkey byte in a cryptographic algorithm executed with known plaintext, which is what most other template-attack demonstrations reported in the literature do. To demonstrate the difference, we use the same *Grizzly* data set from above in order to simulate a DPA-style template attack on AES.

For this, we treat the values k used during profiling and attack as if they where actually the output $v = \text{Sbox}(p \oplus k)$ of the AES S-box. In other words, we generate a random key byte and for each existing trace then obtain the corresponding plaintext byte p such that $v = \text{Sbox}(p \oplus k)$ is the byte loaded. Using the same trace data all measurement parameters remain identical, making the results comparable.[12]

During the attack, we know for each trace $\mathbf{x}_i$ the plaintext byte p, and compute for each candidate key byte k the S-box output value $v = \text{Sbox}(p \oplus k)$. We then obtain the linear discriminant of each candidate key byte k as

$$D_{\text{linear}}(k \mid \mathbf{x}_i) = \bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \mathbf{x}_i - \frac{1}{2} \bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \bar{\mathbf{x}}_v. \tag{33}$$

[9]The selections *1ppc*, *3ppc* and *20ppc* provide a variable number of samples because of the additional restriction that the selected samples must be above the highest 95th percentile of $s(t)$, which varies with n_{p} for each clock edge.

[10]As was also observed in older [13], [21], as well as more recent [35] publications.

[11]In [19], the authors mention that in general the best we can hope from a SPA scenario is to obtain only the Hamming Weight of the value; in [25, Table 7.1], Oswald reports a guessing entropy above 1.8 bits, even with $n_{\text{p}} = n_{\text{a}} = 4000$ traces; in [35], when targeting the move of ciphertext (in a SPA scenario), the guessing entropy remains above 2 bits for $n_{\text{a}} = 1000$ traces.

[12]DPA attacks on real AES software implementations are usually less challenging: they can observe more than just one instruction handling v.

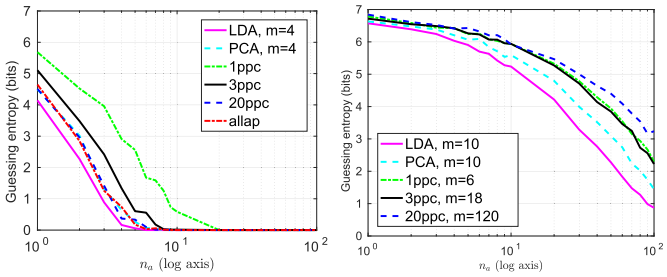

Fig. 4. Guessing entropy after template attack on the *Grizzly* dataset in an AES S-box scenario (left), and on the *Polar* dataset (right, AES engine), when using different compression methods.

Therefore, when combining multiple attack traces we have to add the linear discriminant of *each* attack trace, obtaining

$$D_{\text{linear}}(k \mid \mathbf{X}_{k\star}) = \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \left(\bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \mathbf{x}_i - \frac{1}{2} \bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \bar{\mathbf{x}}_v \right). \tag{34}$$

While this now takes $O(n_{\text{a}} \cdot m^2)$ steps, we can precompute the fixed factors corresponding to the value v, and obtain a computation in $O(n_{\text{a}} \cdot m)$ operations. For the computation of the guessing entropy, we only used a single fixed key value k, since the target value v depends also on the plaintext bytes p.

Figure 4 (left) shows the guessing entropy achieved: in the *DPA* scenario targeting AES, the guessing entropy drops to almost zero within just $n_{\text{a}} = 5$ attack traces and $n_{\text{p}} = 200$, compared to the LOAD instruction scenario (Figure 3), where we need $n_{\text{a}} = 10^3$ traces for comparable results.

D. Results on AES Hardware Cryptographic Engine

Since the Atmel XMEGA A3U microcontroller also contains an AES hardware cryptographic module, we also implemented the above attack against that target. The module can perform one AES encryption with a 128-bit key in 375 CPU clock cycles. Each such AES encryption requires 10 encryption rounds. We recorded a dataset with a total of 384 000 traces, taken while running the hardware AES encryption module with the 16-byte fixed random key (in hexadecimal notation) "3c53eb11a470e4f7df71b49f2f7e72c6" and uniformly distributed 16-byte plaintexts (we shall refer to these traces as the *Polar* dataset). Each trace contains 5000 leakage samples, recorded at 500 MS/s using the HIRES mode of the Tektronix TDS 7054 oscilloscope, with 250 MHz bandwidth. The XMEGA PCB was powered from batteries via a 3.3 V linear regulator, and we provided the CPU with a sine wave clock signal at 2 MHz. These traces cover the first 20 CPU clock cycles of the AES encryption, which correspond only to part of the first encryption round of AES.

We implemented the template attack with the linear discriminant as above, using the byte $v = \text{Sbox}(p \oplus k)$ from the first AES encryption round as our target, where p and k represent the first byte of the plaintext and key, respectively. During the profiling step, we compute the value $v = \text{Sbox}(p \oplus k)$ corresponding to each plaintext byte p and compute the template parameters for each value of v instead of k.

Figure 4 (right) shows the guessing entropy remaining after using $n_{\text{p}} = 1000$ profiling traces per target value v with the

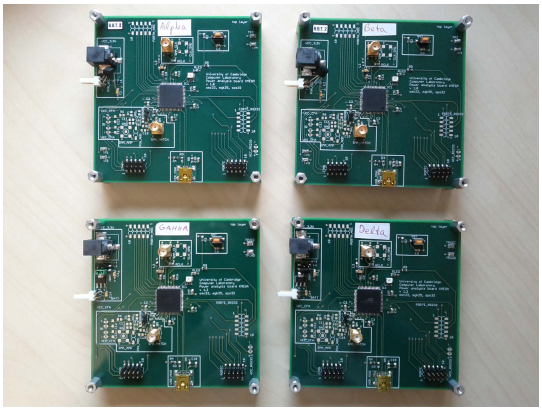

Fig. 5. The four XMEGA PCB devices used in our experiments.

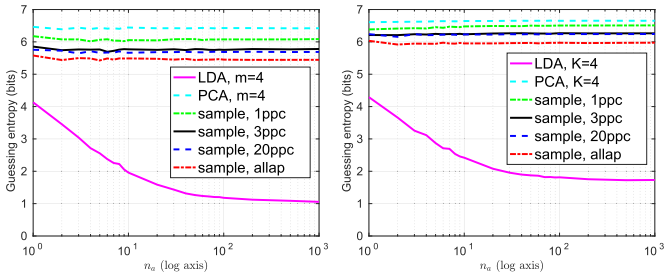

Fig. 6. Classic template attacks in different scenarios. Left: using *Alpha* for profiling and *Beta* for attack. Right: using same device (*Beta*) but different acquisition campaigns for profile (*Beta*) and attack (*Beta Bis*).

compression methods LDA, PCA, *1ppc*, *3ppc* and *20ppc*. Each line represents the average over 1000 experiments. We see that as n_a increases, LDA becomes the most efficient method (as expected, given the analysis of Bruneau *et al.* [34]), followed by PCA and *3ppc*, similarly to the results in Section IV-B. A notable difference is that, when attacking a LOAD instruction, the guessing entropy decreases abruptly within the first $n_a = 10$ traces, and then starts to level off, but here (as in Section IV-C) the guessing entropy steadily drops towards zero as n_a grows. This is because now we target different values v, which reduces the set of possible candidates with each new attack trace (e.g. there is a single value with Hamming weight 0) and helps the attack to converge faster.

V. Results Using Different Devices

Most publications on template attacks [6], [10], [15], [17], [28] used the same device (and most probably the same acquisition campaign) for the profiling and attack phases in their evaluation. The results of template attacks in this ideal case were shown in the previous section. However, attackers who can access a target device only briefly will be forced to use a different device for profiling. To explore the application of template attacks in this perhaps more realistic scenario, we produced four custom PCBs (see Figure 5) for the XMEGA microcontroller. We ran five acquisition campaigns: one for each of the devices, which we call *Alpha*, *Beta*, *Gamma* and *Delta* (the same name as the device,[13]) and another one at a later time for *Beta*, which we call *Beta Bis*. All these acquisition campaigns use the same setup as described in Section IV-B, hence are also part of the *Grizzly* dataset.

There have been some previous publications in this context. Renauld *et al.* [20] performed an extensive study on 20 different devices with 65 nm CMOS transistor technology, showing that the template attack may not work at all when the profiling and attack steps are performed on different devices. However, they only used a sample compression method with 1 to 3 samples. Elaabid *et al.* [22] showed that acquisition campaigns on the same device, but conducted at different times, also lead to worse template-attack results. They also only evaluated a single compression method (PCA with $m = 1$). Heuser *et al.* [23] used a variant of PCA that did not use information from the mean vectors (in contrast to how we use PCA and LDA here). Lomné *et al.* [26] also evaluated different devices, using electromagnetic leakage, again with just a single compression method. Below we compare many variants of compression methods.

As we show in Figure 6 (left), the overall efficacy of template attacks drops dramatically when using different devices for the profiling and attack steps. This confirms what Renauld *et al.* [20] observed. In Figure 6 (right) we find that, even when using the same device but different acquisition campaigns (same acquisition settings), we get results as bad or even worse as when using different devices, as was previously observed by Elaabid and Guilley [22]. However, a notable observation is that LDA is in fact able to produce good results even in this scenario. In the following we offer an explanation for these results and show how to use PCA and LDA efficiently.

A. Main Cause for Differences

Figure 7 shows the overall mean vectors $\bar{\mathbf{x}}^{\mathrm{r}} = \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} \bar{\mathbf{x}}^{\mathrm{r}}_k$ for each campaign, from which we removed the overall mean vector of *Beta* (hence the vector for *Beta* is 0). From this figure we see that all overall mean vectors $\bar{\mathbf{x}}^{\mathrm{r}}$ (except the one for *Beta*) are far outside the confidence region of *Beta* ($\alpha = 0.05$). Moreover, we see that the overall mean vector $\bar{\mathbf{x}}^{\mathrm{r}}$ for *Beta Bis* is the most distant from the overall mean vector of *Beta*. This leads us to the hypothesis that the main difference between our acquisition campaigns is an overall offset caused by campaign-dependent factors, such as temperature drift or battery charge, and not necessarily by the use of different devices. A similar observation was made by Elaabid and Guilley [22], however they used different setups for the different campaigns on the same devices. In our study we have used the exact same setup for the acquisition of data, while replacing only the tested device (evaluation board).

We also plot in Figure 8 the distributions of our data for *Alpha* and *Beta*. We observe that the distributions are very similar (in particular the ordering of the different candidates k is generally the same) but differ mainly by an overall offset, as observed earlier. Therefore, for our experiments, this offset seems to be the main reason why template attacks perform

[13]Devices *Alpha* and *Beta* used a CPU with week batch ID 1145, while *Gamma* and *Delta* had 1230.

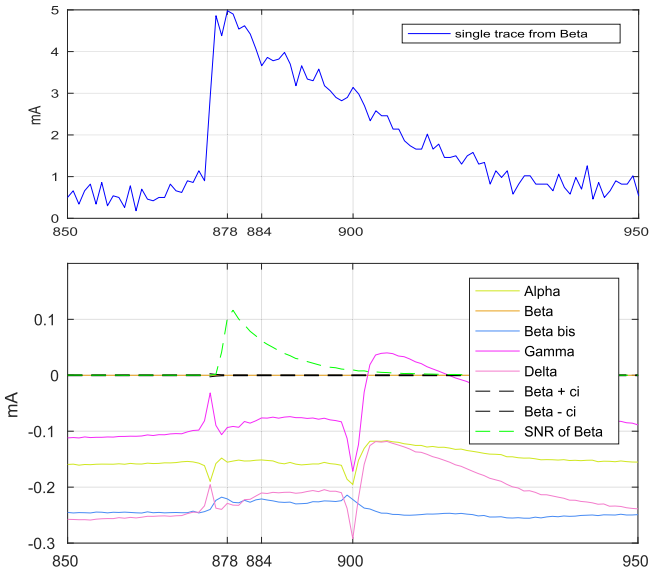

Fig. 7. Trace from *Beta* (top) and overall mean vectors $\bar{\mathbf{x}}^{\mathrm{r}}$ for all campaigns (bottom), from which the overall mean vector of *Beta* was subtracted. *Beta+ci* and *Beta−ci* represent the confidence region ($\alpha = 0.05$) for the overall mean vector of Beta. *SNR of Beta* is the Signal-to-Noise signal strength estimate of *Beta* (rescaled). Samples at first clock cycle of target LOAD instruction.

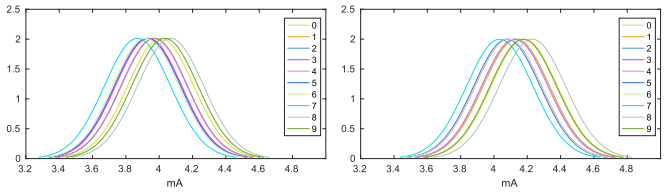

Fig. 8. Normal distribution at sample index $j = 884$ based on the template parameters $(\bar{\mathbf{x}}^{\mathrm{r}}_k, \mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}})$ for $k \in \{0, \ldots, 9\}$ on *Alpha* (left) and *Beta* (right).

badly when using different campaigns for the profiling and attack steps.

In some circumstances, traces might be misaligned, e.g. due to lack of a good trigger signal, or random delays introduced by some countermeasure. In such cases, one could first apply a resynchronisation method, such as those proposed by Homma *et al.* [11]. Our experiments used a very stable trigger, as shown by the exact alignments of sharp peaks in Figure 7.

B. Improved Attacks on Different Devices

There are several ways to improve template attacks when using different devices for the profiling and attack steps. One straight-forward method is to perform the profiling step over several devices (or campaigns) in order to capture some of the inter-campaign variation [20], [30]. Another approach is to compensate for the DC offset directly, e.g. by recording through a DC-block or high-pass filter, or normalizing the mean in the traces, by subtracting an estimate of the DC offset [27], [30], or indirectly by compressing only differences between attack traces [23]. Compensating a DC offset may improve the results of template attacks in some cases, but also carries a risk, in particular with longer recordings: DC filters add non-local effects (impulse response) that can spread the influence of localized noise (e.g., from variations in other data or control-flow several clock cycles away). Such noise can also affect DC-offset estimates, unless the samples used in these estimates have been carefully selected, and therefore spread its effects during mean normalization. Offset compensation can actually decrease the performance of PCA or LDA [30].

Therefore, rather than adding extra DC-compensating pre-processing steps, we explore below how to manipulate the PCA and LDA dimensionality-reduction techniques to make their output less sensitive to low-frequency variability.

1) Efficient Use of PCA and LDA: Remember from Section III-C that LDA takes into consideration the *raw* pooled covariance $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$. Also, we mention that we acquired traces for random permutations of all values k at a time and our acquisition campaigns took a few hours to complete. Therefore, the pooled covariance $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$ of a given campaign contains information about the drifting parameters that may have influenced the current consumption of our microcontrollers over the acquisition period. But one of the major sources of low-frequency noise is temperature variation (which can affect the CPU, the voltage regulator of our boards, the voltage reference of the oscilloscope, our measurement resistor; see also the study by Heuser *et al.* [23]), and we expect such temperatures to be as variable within a campaign as they are across campaigns if each acquisition campaign takes several hours. As a result, the temperature variation captured by the covariance matrix $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$ of one campaign should be similar across different campaigns. However, the mean vectors $\bar{\mathbf{x}}^{\mathrm{r}}_k$ across different campaigns can be different due to different DC offsets (even if the overall temperature variation is similar), and this is why the sample selection methods (e.g. *20ppc*, *allap*) perform poorly across different campaigns. Nevertheless, the LDA algorithm appears to be able to remove the DC component and use only the rest of the trace for the attack. This, combined with the fact that with LDA we no longer need a covariance matrix after compression, allows LDA to filter out temperature variations and other variability sources that are similar across campaigns, and provide good results even across different devices.

In order to show how LDA and PCA deal with the DC offset, we show in Figure 9 (top) the DC components (magnitude of mean) of the LDA and PCA eigenvectors. For LDA we can see that there is a peak at the fifth DC component, which shows that our choice of $m = 4$ avoided the component with largest DC offset by chance. For PCA we can see a similar peak, also for the fifth component, and again our choice $m = 4$ avoided this component. However, for PCA this turned out to be a disadvantage, because PCA does use a covariance matrix after projection, and therefore it would benefit from getting knowledge of the temperature variation from the samples. This variation will be encoded by the eigenvector with a high DC offset and therefore we expect that adding this eigenvector may provide better results. We also show in Figure 9 the first six eigenvectors of $\left(\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}\right)^{-1}\mathbf{B}$ (LDA), $\mathbf{B}$ (PCA) and $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$, along with the first 20 eigenvalues of LDA and PCA. A small DC offset can be seen in the fifth eigenvector for the PCA example, but is not visually obvious for the LDA

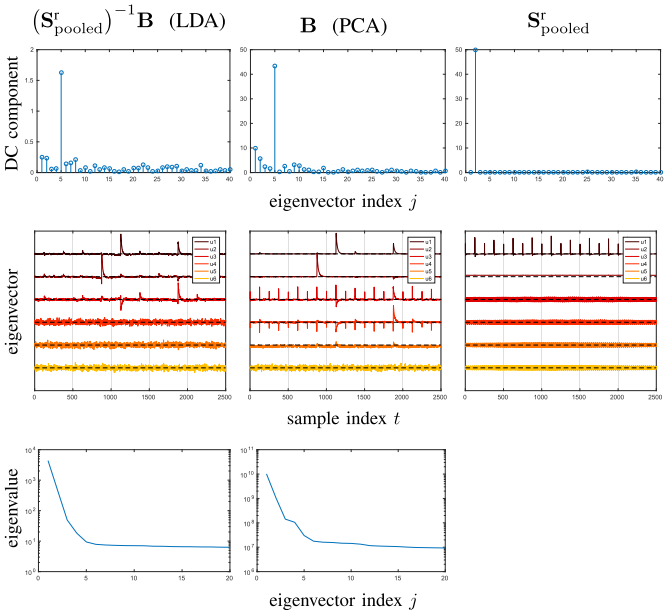

Fig. 9. Top: DC components of eigenvectors of $(\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}})^{-1}\mathbf{B}$ (LDA), $\mathbf{B}$ (PCA) and $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$. Middle: First six eigenvectors of these three matrices. Bottom: eigenvalues (log y axis) of LDA and PCA.

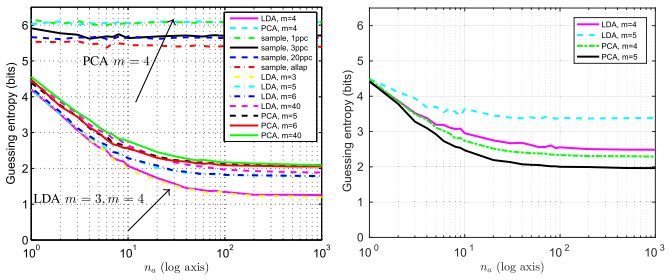

Fig. 10. Template attack on different campaigns (profiling on *Alpha*, attack on *Beta*). Left: using various compressions with the classic method. Right: using PCA and LDA after adding random DC offset.

one. Also, we see that the division by $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$ in LDA has removed much of the noise found in the PCA eigenvectors, and it appears that LDA has reduced the number of components extracting most information from four (in PCA) down to three.

To confirm the above observations, we show in Figure 10 (left) the results of template attacks when using PCA and LDA with different values of m. We see that for LDA there is a great gap between using $m = 4$ and $m = 5$, no gap between $m = 3$ and $m = 4$, while the gap between $m = 5$ and $m = 40$ is very small. This confirms our previous observation that with LDA we should ignore the eigenvector containing a strong DC coefficient. Also, we see that for PCA there is a huge gap between using $m = 4$ and $m = 5$ (in the opposite sense as with LDA), but the gap between $m = 5$ and $m = 40$ is negligible. Therefore, PCA can work well across devices if we include the eigenvectors containing the DC offset information. These results provide an important lesson for implementing template attacks across different devices or campaigns: the choice of components should consider the DC offset contribution of each eigenvector. This suggests that previous studies may have missed important information, by using only sample selections with one to three samples [20] or only the first PCA component [22].

2) Add DC Offset Variation to PCA: Renauld *et al.* [20] mentioned that *"physical variability makes the application of PCA irrelevant, as it cannot distinguish between inter-plaintext and inter-chip variances"*. While it is true that the standard PCA approach [9] is not aimed at distinguishing between the two types of variance, we showed earlier that PCA can actually provide good results if we select the eigenvectors carefully. Starting from this observation, we can try to enhance the PCA algorithm by deliberately adding DC noise, in the hope of concentrating the DC sensitivity in one of the first eigenvectors, thereby making the other eigenvectors less DC sensitive (as all eigenvectors are orthogonal). This method can be implemented using the following steps:

1) Obtain the *raw* leakage traces $\mathbf{X}^{\mathrm{r}}_k$ from the profiling device, for each k.
2) Obtain the *raw* pooled covariance matrix $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}} \in \mathbb{R}^{m^{\mathrm{r}} \times m^{\mathrm{r}}}$.
3) Pick a random offset c_k for each mean vector $\bar{\mathbf{x}}^{\mathrm{r}}_k$.[14]
4) Compute the between-groups matrix as $\mathbf{B} = \sum_{k \in \mathcal{S}} (\bar{\mathbf{x}}^{\mathrm{r}}_k - \bar{\mathbf{x}}^{\mathrm{r}} + \mathbf{1}^{\mathrm{r}} \cdot c_k)(\bar{\mathbf{x}}^{\mathrm{r}}_k - \bar{\mathbf{x}}^{\mathrm{r}} + \mathbf{1}^{\mathrm{r}} \cdot c_k)'$.
5) Use PCA (uses $\mathbf{B}$ only) or LDA (uses both $\mathbf{B}$ and $\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$) to compress the *raw* leakage traces and obtain $\mathbf{X}_k$ for each k.
6) Compute the template parameters ($\bar{\mathbf{x}}_k$, $\mathbf{S}_{\mathrm{pooled}}$) using (1) and (7).
7) Obtain the compressed leakage traces $\mathbf{X}_{k\star}$ from the attacked device.
8) Compute the guessing entropy (see Section IV-A).

The results of this method are shown in Figure 10 (right). We see that now PCA provides good results even with $m = 4$, whereas LDA now gives worse results with $m = 4$. Figure 11 shows the eigenvectors from LDA and PCA, along with their DC component. We can see that, by using this method, we managed to push the eigenvector having the strongest DC component first, and this was useful for PCA. However, LDA does not benefit from including a noise eigenvector into $\mathbf{B}$, so we propose this method only for use with PCA.

3) Applicability: Note that the attack evaluated in Figure 6 targets an 8-bit LOAD instruction directly, just like in Section IV-B and Figure 3, but on different devices. This is a very challenging attack and therefore particularly sensitive to inter-campaign differences. If instead we repeat this attack in the much easier scenario of Section IV-C and Figure 4 (left), against a (simulation of a) software-implementation of AES, these differences weigh much less and the attack performs well without special measures, such as those proposed above or the clustering technique proposed by Whitnall and Oswald [32]: the guessing entropy still drops to zero with just $n_{\mathrm{a}} = 20$ traces, for LDA even with just $n_{\mathrm{a}} = 5$ traces.

[14] We have chosen c_k uniformly from the interval $[-u, u]$, where u is the absolute average offset between the overall mean vectors shown in Figure 7.

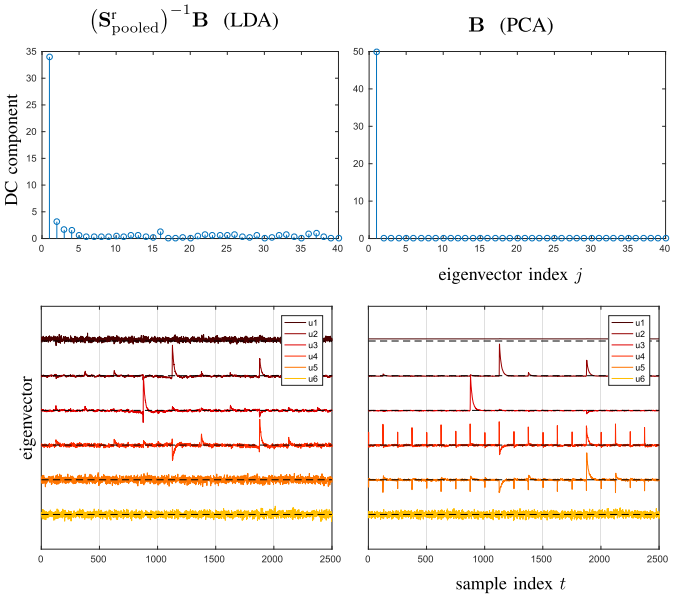

Fig. 11. Top: DC components of eigenvectors of $(\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}})^{-1}\mathbf{B}$ (LDA) and $\mathbf{B}$ (PCA) after adding random DC offsets to $\bar{\mathbf{x}}^{\mathrm{r}}_k$. Bottom: The first six of these LDA and PCA eigenvectors.

VI. Conclusions

In this paper, we have explored in detail some of the main problems that may arise in practice when implementing template attacks based on the multivariate normal distribution. These problems appear when using a large number of samples, or different devices for the profiling and attack steps, both of which are most likely to happen in a practical template attack.

We presented several methods that can help dealing with these problems, such as the use of pooled covariance matrices ($\mathbf{S}^{\mathrm{r}}_{\mathrm{pooled}}$, $\mathbf{S}_{\mathrm{pooled}}$) and the compression methods PCA and LDA, as well as the use of a linear discriminant D_{linear}, which can be much more efficient than the logarithm of the multivariate normal distribution $D_{\log}$ (in some of our experiments we reduced the evaluation time from 3.5 days to 30 minutes). We also showed that there are ways to use these compression methods in the context of different devices to increase the performance of template attacks in this scenario.

We applied all the methods presented in this paper to real traces from an AVR XMEGA 8-bit microcontroller. We targeted both its CPU and hardware AES engine. Firstly, we eavesdropped directly on the CPU transferring an unknown byte in a LOAD instruction, independent of any cryptographic algorithm. Secondly, we simulated an attack on a software implementation of the AES cipher, reconstructing a subkey byte from known plaintext and targeting a first-round Sbox output. Finally, we did the same to the hardware AES engine. We compared our methods in all three scenarios by means of the guessing entropy. Even for the particularly demanding, algorithm-neutral attack on the LOAD instruction, where we had to distinguish the power signatures of each individual data-bus wire, we were able to reduce the guessing entropy to near zero bits, i.e. we are able to extract all 8 bits processed by a single LOAD instruction (i.e., not just their Hamming weight). We also showed how to use PCA and LDA to achieve similar results when using different devices for profiling and attack.

Data and Code Availability: In the interest of reproducible research we make available our data and associated MATLAB scripts [29].

References

[1] P. C. Mahalanobis, "On the generalised distance in statistics," *Proc. Nat. Inst. Sci.*, vol. 2, no. 1, pp. 49–55, 1936.

[2] R. A. Fisher, "The statistical utilization of multiple measurements," *Ann. Hum. Genet.*, vol. 8, no. 4, pp. 376–386, 1938.

[3] J. L. Massey, "Guessing and entropy," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun./Jul. 1994, p. 204.

[4] C. Cachin, "Entropy measures and unconditional security in cryptography," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 1997.

[5] J. O. Pliam, "On the incomparability of entropy and marginal guesswork in brute-force attacks," in *Progress in Cryptology—INDOCRYPT* (Lecture Notes in Computer Science), vol. 1977. Cham, Switzerland: Springer, 2000, doi: https://doi.org/10.1007/3-540-44495-5_7.

[6] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 2523. Cham, Switzerland: Springer, 2003, pp. 51–62, doi: https://doi.org/10.1007/3-540-36400-5_3.

[7] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Wiley, 2005.

[8] C. Rechberger and E. Oswald, "Practical template attacks," in *Information Security Applications* (Lecture Notes in Computer Science), vol. 3325. Cham, Switzerland: Springer, 2005, pp. 440–456, doi: https://doi.org/10.1007/978-3-540-31815-6_35.

[9] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 4249. Cham, Switzerland: Springer, 2006, pp. 1–14, doi: https://doi.org/10.1007/11894063_1.

[10] B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. stochastic methods," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 4249. Cham, Switzerland: Springer, pp. 15–29, doi: https://doi.org/10.1007/11894063_2.

[11] N. Homma, S. Nagashima, Y. Imai, T. Aoki, and A. Satoh, "High-resolution side-channel attack using phase-based waveform matching," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 4249. Berlin, Germany: Springer, 2006, pp. 187–200, doi: https://doi.org/10.1007/11894063_15.

[12] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 6th ed. Harlow, U.K.: Pearson, 2007.

[13] P. Karsmakers *et al.*, "Side channel attacks on cryptographic devices as a classification problem," Dept. COSIC, KU Leuven Univ., Leuven, Belgium, Tech. Rep. 07/36.

[14] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 286–296.

[15] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 5154. Cham, Switzerland: Springer, 2008, pp. 411–425, doi: https://doi.org/10.1007/978-3-540-85053-3_26.

[16] L. Batina, B. Gierlichs, and K. Lemke-Rust, "Comparative evaluation of rank correlation based DPA on an AES prototype chip," in *Information Security* (Lecture Notes in Computer Science), vol. 5222. Berlin, Germany: Springer, 2008, pp. 341–354, doi: https://doi.org/10.1007/978-3-540-85886-7_24.

[17] F.-X. Standaert, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 5479. Berlin, Germany: Springer, 2009, pp. 443–461, doi: https://doi.org/10.1007/978-3-642-01001-9_26.

[18] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a side channel based disassembler," in *Transactions on Computational Science X*, vol. 6340. Berlin, Germany: Springer, 2010, pp. 78–99, doi: https://doi.org/10.1007/978-3-642-17499-5_4.

[19] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. 1st ed. New York, NY, USA: Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-0-387-38162-6

[20] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6632. Berlin, Germany: Springer, 2011, pp. 109–128.
[21] D. Oswald and C. Paar, "Breaking Mifare DESFire MF_3ICD40: Power analysis and templates in the real world," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 6917. Berlin, Germany: Springer, 2011, pp. 207–222.
[22] M. A. Elaabid and S. Guilley, "Portability of templates," *J. Cryptogr. Eng.*, vol. 2, no. 1, pp. 63–74, 2012.
[23] A. Heuser, M. Kasper, W. Schindler, and M. Stöttinger, "A new difference method for side-channel analysis with high-dimensional leakage models," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 7178. Berlin, Germany: Springer, 2012, pp. 365–382.
[24] D. Mavroeidis, L. Batina, T. van Laarhoven, and E. Marchiori, "PCA, eigenvector localization and clustering for side-channel attacks on cryptographic hardware devices," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 7523. Springer, 2012, pp. 253–268.
[25] D. Oswald, "Implementation attacks: From theory to practice," Ph.D. dissertation, Ruhr-Unive. Bochum, Bochum, Germany, 2013.
[26] V. Lomné, E. Prouff, and T. Roche, "Behind the scene of side channel attacks," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 8269. Berlin, Germany: Springer, 2013, pp. 506–525.
[27] D. P. Montminy, R. O. Baldwin, M. A. Temple, and E. D. Laspe, "Improving cross-device attacks using zero-mean unit-variance normalization," *J. Cryptogr. Eng.*, vol. 3, no. 2, pp. 99–110, 2013.
[28] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *Smart Card Research and Advanced Applications* (Lecture Notes in Computer Science), vol. 8419. Berlin, Germany: Springer, Nov. 2013, pp. 253–270.
[29] M. O. Choudary and M. G. Kuhn. *Grizzly: Power-Analysis Traces for an 8-Bit Load Instruction*. Accessed: Oct. 2017. [Online]. Available: http://www.cl.cam.ac.uk/research/security/datasets/grizzly/
[30] O. Choudary and M. G. Kuhn, "Template attacks on different devices," in *Constructive Side-Channel Analysis and Secure Design* (Lecture Notes in Computer Science), vol. 8622. Paris, France: Springer, Apr. 2014, pp. 179–198.
[31] M. O. Choudary, "Efficient multivariate statistical techniques for extracting secrets from electronic devices," Ph.D. dissertation, Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-878, Sep. 2015.
[32] C. Whitnall and E. Oswald, "Robust profiling for DPA-style attacks," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 9293. Berlin, Germany: Springer, 2015, pp. 3–21.
[33] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in *Smart Card Research and Advanced Applications* (Lecture Notes in Computer Science), vol. 9514. Cham, Switzerland: Springer, 2015, pp. 15–33. [Online]. Available: https:/doi.org/10.1007/978-3-319-31271-2_2
[34] N. Bruneau, S. Guilley, A. Heuser, D. Marion, and O. Rioul, "Less is more—Dimensionality reduction from a theoretical perspective," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 9293. Springer, 2015, pp. 22–41.
[35] M. Wagner, Y. Hu, C. Zhang, and Y. Zheng, "Comparative study of various approximations to the covariance matrix in template attacks," International Association for Cryptologic Research, Tech. Rep. 2016/1155, 2016. [Online]. Available: http://eprint.iacr.org/2016/1155.pdf
[36] H. Zhang and Y. Zhou, "How many interesting points should be used in a template attack?" *J. Syst. Softw.*, vol. 120, pp. 105–113, Oct. 2016.

Marios O. Choudary received the Diploma engineer degree from the University Politehnica of Bucharest in 2008 and the M.Phil. and Ph.D. degrees, in 2010 and 2014, in computer science from the University of Cambridge. He is currently a Lecturer in computer science with the University Politehnica of Bucharest. His research interests include authentication, security protocols, and side-channel analysis.

Markus G. Kuhn received the Diplom-Informatiker degree from the University of Erlangen-Nürnberg, Germany, in 1996, the M.Sc. degree from Purdue University, IN, in 1997, and the Ph.D. degree from the University of Cambridge, U.K., in 2002, all in computer science. He is currently a Senior Lecturer in computer science with the University of Cambridge. His research interests include hardware and signal-processing aspects of computer security, in particular compromising emanations, side-channel attacks, distance bounding protocols, and the security of RFID and navigation systems.