Efficient Template Attacks

Computer Laboratory, University of Cambridge
firstname.lastname@cl.cam.ac.uk

Abstract. Template attacks remain a powerful side-channel technique to eavesdrop on tamper-resistant hardware. They model the probability distribution of leaking signals and noise to guide a search for secret data values. In practice, several numerical obstacles can arise when implementing such attacks with multivariate normal distributions. We propose efficient methods to avoid these. We also demonstrate how to achieve significant performance improvements, both in terms of information extracted and computational cost, by pooling covariance estimates across all data values. We provide a detailed and systematic overview of many different options for implementing such attacks. Our experimental evaluation of all these methods based on measuring the supply current of a byte-load instruction executed in an unprotected 8-bit microcontroller leads to practical guidance for choosing an attack algorithm.

Keywords: side-channel attacks, template attack, multivariate analysis.

1 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 evaluating side-channel information is the *template attack* [4], which relies on a multivariate model of the side-channel traces. While the basic algorithm is comparatively simple (Section 2), there are a number of additional steps that must be performed in order to obtain a practical and efficient implementation.

In this paper we examine several problems that can arise in the implementation of template attacks (Section 3), especially when using a large number of voltage samples. We explain how to solve them in two steps: (*a*) using *compression techniques*, i.e. methods to reduce the number of samples involved, either by throwing away most, or by projecting them into a lower-dimensional space, using only a few linear combinations (Section 4); and (*b*) we contribute *efficient variants* of the template-attack algorithm, which can avoid numerical limitations of the standard approach, provide better results and execute faster (Section 5).

A. Francillon and P. Rohatgi (Eds.:): CARDIS 2013, LNCS 8419, pp. 253–270, 2014.
The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-08302-5_17

We evaluate all these methods in practice, against an unprotected 8-bit microcontroller, comparing their effectiveness using the guessing entropy (Section 6). We focus on gathering information about individual data values, *independent* of what algorithm these are part of. Other algorithm-specific attacks that use dependencies between different data values, e.g. to recover keys from a specific cipher, could be implemented on top of that, but are outside the scope of this paper. We show that PCA and LDA provide the best results overall, and that a previous guideline of selecting at most one point per clock cycle is not optimal in general. Based on these experiments and theoretical background, we provide practical guidance for the choice of template-attack algorithm.

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

During the *profiling* phase we record n_p leakage vectors $\mathbf{x}^\mathrm{r}_{ki} \in \mathbb{R}^{m^\mathrm{r}}$ from the profiling device for each possible value $k \in \mathcal{S}$, and combine these as row vectors $\mathbf{x}^\mathrm{r}_{ki}{}'$ in the leakage matrix $\mathbf{X}^\mathrm{r}_k \in \mathbb{R}^{n_\mathrm{p} \times m^\mathrm{r}}$.[1]

Typically, the *raw* leakage vectors $\mathbf{x}^\mathrm{r}_{ki}$ provided by the data acquisition device contain a 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 $m \ll m^\mathrm{r}$ of those samples, or by applying some other data-dimensionality reduction method (see Section 4). 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 $\mathbf{X}_k \in \mathbb{R}^{n_\mathrm{p} \times m}$. (Without any such compression step, we would have $\mathbf{X}_k = \mathbf{X}^\mathrm{r}_k$ and $m = m^\mathrm{r}$.)

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

$$\bar{\mathbf{x}}_k = \frac{1}{n_\mathrm{p}} \sum_{i=1}^{n_\mathrm{p}} \mathbf{x}_{ki}, \qquad \mathbf{S}_k = \frac{1}{n_\mathrm{p}-1} \sum_{i=1}^{n_\mathrm{p}} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)', \tag{1}$$

where the sample mean $\bar{\mathbf{x}}_k$ and the sample *unbiased*[2] covariance matrix $\mathbf{S}_k$ are the estimates of the true mean μ_k and true covariance $\mathbf{\Sigma}_k$. Note that a *sum of*

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

[2] Others [8,11,14] use $1/n_\mathrm{p}$ rather than $1/(n_\mathrm{p} - 1)$ in $\mathbf{S}_k$, thereby computing the *maximum likelihood estimator (MLE)* of $\mathbf{\Sigma}_k$. In theory, the correct estimator for

squares and cross products matrix such as $\sum_{i=1}^{n_\mathrm{p}}(\mathbf{x}_{ki}-\bar{\mathbf{x}}_k)(\mathbf{x}_{ki}-\bar{\mathbf{x}}_k)'$, from (1) can also be written as

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

where $\widetilde{\mathbf{X}}_k$ is $\mathbf{X}_k$ with $\bar{\mathbf{x}}_k'$ subtracted from each row.[3]

Side-channel leakage traces can generally be modeled well by a multivariate normal distribution [4], 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 [10, Chapter 4]. Then the probability density function (pdf) of a leakage vector $\mathbf{x}$, given $\bar{\mathbf{x}}_k$ and $\mathbf{S}_k$, is

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

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 $\mathbf{X}_{k\star} \in \mathbb{R}^{n_\mathrm{a}\times m}$. Then, for each $k \in \mathcal{S}$, we compute a *discriminant score* $\mathrm{d}(k \mid \mathbf{X}_{k\star})$. 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 [8,11,14], derived from Bayes' rule, is

$$\mathrm{d}(k \mid \mathbf{x}_i) = \mathrm{f}(\mathbf{x}_i \mid \bar{\mathbf{x}}_k, \mathbf{S}_k)P(k), \tag{4}$$

where the denominator from Bayes' rule 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

$$\mathrm{l}(k \mid \mathbf{x}_i) = \mathrm{d}(k \mid \mathbf{x}_i) = \mathrm{l}(\bar{\mathbf{x}}_k, \mathbf{S}_k \mid \mathbf{x}_i) = \mathrm{f}(\mathbf{x}_i \mid \bar{\mathbf{x}}_k, \mathbf{S}_k), \tag{5}$$

where the latter can be computed from (3). However, we do not need to compute a proper a-posteriori probability for each candidate k given a trace $\mathbf{x}_i$, but only a discriminant function that allows us to sort scores and identify the most likely candidates. Section 5 shows how the latter can be much more efficient.

3 Implementation Caveats

We now present several problems that can appear when implementing the template attack, especially when using a large number of samples m.

$\mathbf{\Sigma}_k$ is the *unbiased estimator* with $1/(n_\mathrm{p}-1)$; the MLE merely maximises the joint likelihood from the multivariate normal distribution. In practice, we found this choice made no significant performance difference (even down to $n_\mathrm{p} = 10, m = 6$).

[3] The matrix form allows the use of fast, vectorized linear-algebra routines.

3.1 Inverse of Covariance Matrix

Several authors [15,14] noted that inverting the covariance matrix $\mathbf{S}_k$ from (1), as needed in (3), can cause numerical problems for large m. However, we consider it important to explain 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 $n_\mathrm{p} < m$. The constraint on $\widetilde{\mathbf{X}}_k$ to have zero-mean rows implies that it has dependent columns even for $n_\mathrm{p} = m$. Therefore, $n_\mathrm{p} > m$ is a *necessary* condition for $\mathbf{S}_k$ to be non-singular. See [10, Result 3.3] for a more detailed proof.

The restriction $m < n_\mathrm{p}$ is one main reason for reducing m through compression (see Section 4). However, it is not mandatory to compress m further than what is needed to keep the columns of $\widetilde{\mathbf{X}}_k$ independent. Note that in practice some samples can be highly correlated, in which case n_p needs to be somewhat larger than m (e.g., $n_\mathrm{p} \geq 3000$ for $m = 1250$ with our Section 6 data).

If we cannot obtain $n_\mathrm{p} > m$ then we can try the covariance estimator of Ledoit and Wolf [5], which gave us a non-singular $\mathbf{S}_k$ even for $n_\mathrm{p} < m$. However, a much better option is to use the *pooled* covariance matrix (see Section 5.2) when possible.

3.2 Floating-point Limitations

One 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 reach values that cause the subsequent exponentiation operation to overflow. For example, in IEEE double precision, $\exp(x)$ 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.

4 Compression Methods

A compression method can be used to reduce the length (dimensionality) of leakage vectors from m^r to m. As detailed in Section 3, this may be needed if we do not have enough traces for a full rank covariance matrix or to cope with computational or memory restrictions. Several approaches are described in the literature, which can be divided into two categories: (a) selecting some of the samples based on some criteria; (b) using some linear combinations of the leakage vectors, based on the principal components or Fisher's linear discriminant. All of the following techniques evaluate the differences $\bar{\mathbf{x}}_k - \bar{\mathbf{x}}$ where

$$\bar{\mathbf{x}} = \frac{1}{|\mathcal{S}|}\sum_{k \in \mathcal{S}} \bar{\mathbf{x}}_k. \tag{6}$$

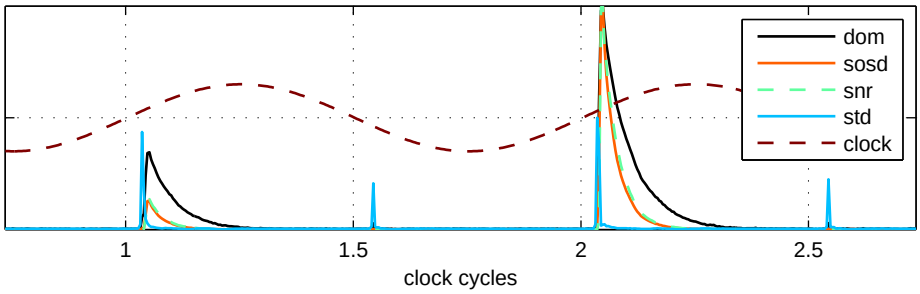

Fig. 1. Signal-strength estimates from DOM, SOSD and SNR (identical to SOST) for a LOAD instruction processing all possible 8-bit values, along with the average standard deviation (STD) of the traces and clock signal. We used 2000 traces per value. All estimates are rescaled to fit into the plot, so the vertical axis (linear) has no scale.

4.1 Selection of Samples

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

There are several proposals for producing $\mathbf{s}(t)$, such as *difference of means (DOM)* [4, Section 2.1], the *sum of squared differences (SOSD)* [9], the *Signal to Noise Ratio (SNR)* [15] and *SOST* [9]. All these are similar, with the notable difference that the first two do no take the variance of the traces into consideration, while the latter two do. We show the difference between these estimates for our experiments in Figure 1. 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 following value used by the *F-test* in the Analysis of Variance [10]:

$$\mathrm{F}(t) = \frac{\left(n_\mathrm{p} \sum_{k \in \mathcal{S}} (\bar{\mathbf{x}}_k(t) - \bar{\mathbf{x}}(t))^2\right) / (|\mathcal{S}| - 1)}{\left(\sum_{k \in \mathcal{S}} \sum_{i=1}^{n_\mathrm{p}} (\mathbf{x}_{ki}(t) - \bar{\mathbf{x}}_k(t))^2\right) / (|\mathcal{S}|(n_\mathrm{p} - 1))}. \tag{7}$$

$\mathrm{F}(t)$ 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 of which samples contain more information about the means.

In the second step of this compression method we need to choose m samples based on the signal-strength estimate $\mathbf{s}$. The goal is to select the smallest set of samples that contains most of the information about our target. An accepted guideline, by Rechberger and Oswald [7, Section 3.2], is to select at most one sample per clock cycle among the samples with highest $\mathbf{s}$. In Section 6 we evaluate several other options, and we show that this guideline is not optimal in general.

4.2 Principal Component Analysis (PCA)

Archambeau et al. [8] proposed the following method for using PCA as a compression method for template attacks. First compute the *sample between groups* matrix $\mathbf{B}$:

$$\mathbf{B} = n_{\mathrm{p}} \sum_{k \in \mathcal{S}} (\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}})(\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}})'. \tag{8}$$

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 eigenvectors $[\mathbf{u}_1 \dots \mathbf{u}_m] = \mathbf{U}^m$ are needed in order to preserve most of the information 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 template parameters $\bar{\mathbf{x}}_k \in \mathbb{R}^m$ and $\mathbf{S}_k \in \mathbb{R}^{m \times m}$:

$$\bar{\mathbf{x}}_k = \mathbf{U}^{m\prime} \bar{\mathbf{x}}_k^{\mathrm{r}}, \qquad \mathbf{S}_k = \mathbf{U}^{m\prime} \mathbf{S}_k^{\mathrm{r}} \mathbf{U}^m. \tag{9}$$

Choice of PCA Components. Archambeau et al. [8] propose 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* [6], requires manual inspection of the eigenvalues. Another technique, which does not require manual inspection of the eigenvalues, is known as the *Cumulative Percentage of Total Variation* [6]. It selects those m eigenvectors that retain at least fraction f of the total variance, by computing the score

$$\phi(m) = \frac{\sum_{1 \le j \le m} \delta_j}{\sum_{1 \le j \le m^{\mathrm{r}}} \delta_j}, \quad 1 \le m \le m^{\mathrm{r}}, \tag{10}$$

and selecting the lowest m for which $\phi(m) > f$.[5] We recommend trying both approaches, as "*there is no definitive answer [to the question of how many components to choose]*" [10, Chapter 8].

Alternative Computation of PCA Templates. Even though in [11, Section 4.1] the authors mention that PCA can help where computing the full covariance matrix $\mathbf{S}_k^{\mathrm{r}}$ is prohibitive (due to large m^{r}), their approach still requires the computation of $\mathbf{S}_k^{\mathrm{r}}$ (see (9)). Also, numerical artifacts during the double matrix

[4] Archambeau et al. [8] show a method for computing $\mathbf{U}$ that is more efficient when $m^{\mathrm{r}} \gg |\mathcal{S}|$, but in our experiments with $m^{\mathrm{r}} = 2500$ this direct approach worked well.

[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.

multiplication in (9) can make $\mathbf{S}_k$ non-symmetric. One way to avoid the latter is to use the Cholesky decomposition $\mathbf{S}_k^{\mathrm{r}} = \mathbf{C}'\mathbf{C}$ and compute

$$\mathbf{S}_k = \mathbf{U}^{m\prime}\mathbf{S}_k^{\mathrm{r}}\mathbf{U}^m = \mathbf{U}^{m\prime}\mathbf{C}'\mathbf{C}\mathbf{U}^m = (\mathbf{C}\mathbf{U}^m)'(\mathbf{C}\mathbf{U}^m) = \mathbf{V}'\mathbf{V}. \tag{11}$$

However, to avoid both the numerical artifacts and the computation of large covariance matrices, we propose an alternative PCA method, based on the following result: given the leakage matrix $\mathbf{X}_k^{\mathrm{r}}$ and the PCA projection matrix $\mathbf{U}^m$, it can be shown [10, Eq. (2-45)] that

$$\mathbf{S}_k = \mathrm{Cov}(\mathbf{X}_k^{\mathrm{r}}\mathbf{U}^m) = \mathbf{U}^{m\prime}\mathrm{Cov}(\mathbf{X}_k^{\mathrm{r}})\mathbf{U}^m = \mathbf{U}^{m\prime}\mathbf{S}_k^{\mathrm{r}}\mathbf{U}^m. \tag{12}$$

Therefore, instead of first computing $\mathbf{S}_k^{\mathrm{r}}$ and then applying (9) or (11), we can first compute the projected leakage matrix

$$\mathbf{X}_k = \mathbf{X}_k^{\mathrm{r}}\mathbf{U}^m \tag{13}$$

and then compute the PCA-based template parameters using (1). We use this method for all the results shown in Section 6.

4.3 Fisher's Linear Discriminant Analysis (LDA)

Given the leakage traces $\mathbf{x}_{ki}^{\mathrm{r}}$ (rows of $\mathbf{X}_k^{\mathrm{r}}$), Fisher's idea [2,10] was to find some coefficients $\mathbf{a}_j \in \mathbb{R}^{m^{\mathrm{r}}}$ that maximise the following ratio:

$$\frac{\sum_{k\in\mathcal{S}}(\bar{y}_{kj} - \bar{y}_j)^2}{\mathrm{Var}(y_j)} = \frac{\sum_{k\in\mathcal{S}}(\mathbf{a}_j{}'(\bar{\mathbf{x}}_k^{\mathrm{r}} - \bar{\mathbf{x}}^{\mathrm{r}}))^2}{\mathrm{Var}(\mathbf{a}_j{}'\mathbf{x})} = \frac{\mathbf{a}_j{}'\mathbf{B}\mathbf{a}_j}{\mathbf{a}_j{}'\mathbf{S}_{\mathrm{pooled}}\mathbf{a}_j}, \tag{14}$$

where the linear combinations $y_j = \mathbf{a}_j{}'\mathbf{x}$ are known as *sample discriminants*, $\mathbf{B}$ is the treatment matrix from (8) and $\mathbf{S}_{\mathrm{pooled}} = \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}}\mathbf{S}_k^{\mathrm{r}}$ is the common covariance of all groups (see also Section 5.2). Note the similarity between the left hand side of (14) and (7) which is used by the F-test, SNR and SOST. This allows us to make an interesting observation: while in the sample selection method we first compute (7) for each sample and then select the samples with the highest $\mathrm{F}(t)$, Fisher's method finds the linear combinations of the trace samples that maximise (14). The coefficients $\mathbf{a}_j$ that maximise (14) are the eigenvectors $[\mathbf{u}_1 \dots \mathbf{u}_{m^{\mathrm{r}}}] = \mathbf{U}$ corresponding to the largest eigenvalues of $\mathbf{S}_{\mathrm{pooled}}^{-1}\mathbf{B}$.[6]

As with PCA, we only need to use the first m coefficients $\mathbf{a}_1, \dots, \mathbf{a}_m$, which can be selected using the same rules discussed in Section 4.2. If we let $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_m] = \mathbf{U}^m$ be the matrix of coefficients, we can project each leakage matrix as:

$$\mathbf{X}_k = \mathbf{X}_k^{\mathrm{r}}\mathbf{A} = \mathbf{X}_k^{\mathrm{r}}\mathbf{U}^m \tag{15}$$

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

[6] 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 $\mathbf{B}$.

Several authors [11,14] 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 (Section 4.2), which does not depend on this condition, might be a better choice in some settings. Secondly, the coefficients that maximise (14) can be obtained using scaled versions of $\mathbf{S}_{\text{pooled}}$[7] or different approaches [11,14], which will result in a different scale of the coefficients $\mathbf{a}_j$. This difference has a major impact on the template attack: *only* when we scale the coefficients $\mathbf{a}_j$, such that $\mathbf{a}_j{}'\mathbf{S}_{\text{pooled}}\mathbf{a}_j = 1$, the covariance between discriminants becomes the identity matrix [10], i.e. $\mathbf{S}_k = \mathbf{I}$. That means the sample means in (1) suffice and we can discard the covariance matrix from the discriminant scores in Section 5, which greatly reduces computation and storage requirements.

Continuing the steps that led to (15), we can compute the diagonal matrix $\mathbf{Q} \in \mathbb{R}^{m\times m}$, having the values $q_{jj} = (\frac{1}{\mathbf{a}_j{}'\mathbf{S}_{\text{pooled}}\mathbf{a}_j})^{\frac{1}{2}} = (\frac{1}{\mathbf{u}_j{}'\mathbf{S}_{\text{pooled}}\mathbf{u}_j})^{\frac{1}{2}}$ on its diagonal, to obtain the scaled coefficients $\mathbf{AQ} = \mathbf{U}^m\mathbf{Q}$, and replace (15) by

$$\mathbf{X}_k = \mathbf{X}_k^{\text{r}}\mathbf{AQ} = \mathbf{X}_k^{\text{r}}\mathbf{U}^m\mathbf{Q}. \tag{16}$$

An alternative approach is to compute the eigenvectors $\mathbf{u}_j$ of $\mathbf{S}_{\text{pooled}}^{-\frac{1}{2}}\mathbf{B}\mathbf{S}_{\text{pooled}}^{-\frac{1}{2}}$ and then obtain the coefficients $\mathbf{a}_j = \mathbf{S}_{\text{pooled}}^{-\frac{1}{2}}\mathbf{u}_j$, which leads directly to coefficients that satisfy $\mathbf{a}_j{}'\mathbf{S}_{\text{pooled}}\mathbf{a}_j = 1$.

5 Efficient Implementation of Template Attacks

In this section we introduce methods that avoid the problems identified in Section 3 and implement template attacks very efficiently.

5.1 Using the Logarithm of the Multivariate Normal Distribution

Mangard et al. [15, p. 108] suggested calculating the logarithm of (3), as in

$$\log \mathrm{f}(\mathbf{x} \mid \bar{\mathbf{x}}_k, \mathbf{S}_k) = -\frac{1}{2}\left(\log\left[(2\pi)^m|\mathbf{S}_k|\right] + (\mathbf{x}-\bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}-\bar{\mathbf{x}}_k)\right). \tag{17}$$

They then claim that *"the template that leads to the smallest absolute value [of (17)] indicates the correct [candidate]"*.

The first problem with this approach is that (17) does not avoid the computation of $|\mathbf{S}_k|$, which we have shown to be problematic. Therefore we propose to compute the logarithm of the multivariate normal pdf as

$$\log \mathrm{f}(\mathbf{x} \mid \bar{\mathbf{x}}_k, \mathbf{S}_k) = -\frac{m}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{S}_k| - \frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}-\bar{\mathbf{x}}_k), \tag{18}$$

[7] Instead of $\mathbf{S}_{\text{pooled}}$ we could use $\mathbf{W} = |\mathcal{S}|(n_{\text{p}}-1)\mathbf{S}_{\text{pooled}}$, known as a *sample within groups* matrix.

where we compute the logarithm of the determinant as

$$\log|\mathbf{S}_k| = 2 \sum_{c_{ii} \in \mathrm{diag}(\mathbf{C})} \log c_{ii}, \tag{19}$$

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

Secondly, it is incorrect to choose the candidate k that leads to the *"smallest absolute value"* of (17,18), since the logarithm is a monotonic function and preserves the property that the *largest value* corresponds to the correct k.[8]

We can use (18,19), dropping the first term which is constant across all k, to compute a discriminant score based on the log-likelihood:

$$\begin{aligned} \mathrm{d}_{\mathrm{LOG}}(k \mid \mathbf{x}_i) &= -\frac{1}{2}\log|\mathbf{S}_k| - \frac{1}{2}(\mathbf{x}_i - \bar{\mathbf{x}}_k)'\mathbf{S}_k^{-1}(\mathbf{x}_i - \bar{\mathbf{x}}_k) \\ &= \log \mathrm{f}(\mathbf{x}_i \mid \bar{\mathbf{x}}_k, \mathbf{S}_k) + \frac{m}{2}\log 2\pi = \log \mathrm{l}(k \mid \mathbf{x}_i) + \mathrm{const.} \end{aligned} \tag{20}$$

5.2 Using a Pooled Covariance Matrix

When the leakages from different candidates k have different means but the same covariance $\mathbf{\Sigma} = \mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \cdots = \mathbf{\Sigma}_k$, it is possible to *pool* the covariance estimates $\mathbf{S}_k$ into a *pooled* covariance matrix [10, Section 6.3]

$$\mathbf{S}_{\mathrm{pooled}} = \frac{1}{|\mathcal{S}|(n_{\mathrm{p}} - 1)} \sum_{k \in \mathcal{S}} \sum_{i=1}^{n_{\mathrm{p}}} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)', \tag{21}$$

an average of the covariances $\mathbf{S}_k$ from (1). The great advantage of $\mathbf{S}_{\mathrm{pooled}}$ over $\mathbf{S}_k$ is that it represents a much better estimator of the real covariance $\mathbf{\Sigma}$, since $\mathbf{S}_{\mathrm{pooled}}$ estimates the covariance using $n_{\mathrm{p}}|\mathcal{S}|$ traces, while $\mathbf{S}_k$ uses only n_{p}. This in turn means that the condition for a non-singular matrix (see Section 3.1) relaxes to $n_{\mathrm{p}}|\mathcal{S}| > m$ or $n_{\mathrm{p}} > \frac{m}{|\mathcal{S}|}$. Therefore the number of traces that we must obtain for each candidate k is reduced by a factor of $|\mathcal{S}|$, a great advantage in practice. Nevertheless, the quality of the mean estimate $\bar{\mathbf{x}}_k$ still depends directly on n_{p}. Also note that for Fisher's LDA (Section 4.3) we need to compute the inverse of $\mathbf{S}_{\mathrm{pooled}} \in \mathbb{R}^{m^{\mathrm{r}} \times m^{\mathrm{r}}}$, which requires $n_{\mathrm{p}}|\mathcal{S}| > m^{\mathrm{r}}$.

Several authors used $\mathbf{S}_{\mathrm{pooled}}$ with template attacks [12,16], but gave no motivation for its use. We would expect the assumption of equal covariances to hold for many side-channel applications, because $\mathbf{S}_k$ captures primarily information about how *noise*, that is variation in the recorded traces unrelated to k, is correlated across trace samples. After all, the data-dependent signal $\bar{\mathbf{x}}_k$ was already subtracted. As a result, we should not expect substantial differences between the $\mathbf{S}_k$ for different candidate values k, unless the targeted device contains a

[8] Note that a pdf, such as f from (3), unlike a probability, can be both larger or smaller than 1 and therefore its logarithm can be both positive *or* negative.

mechanism by which k can modify the correlation between samples (which we do not completely exclude).

Box's test [3] can be used to reject the hypothesis of equal covariances, although it can be misleading for large $|\mathcal{S}|$ or large m. In our experiments, with $|\mathcal{S}| = 2^8$, $m = 6$ and $n_\mathrm{p} = 2000$, Box's variable $C \sim F_{f1,f2}(\alpha)$ had the value 2.03, which was above the rejection threshold for any realistic significance level (e.g. $F_{f1,f2}(0.99) = 1.045$). Nevertheless, we found the different $\mathbf{S}_k$ to be visually similar (viewed as bitmaps with linear colour mapping), and we consider that our hypothesis was confirmed by the superior results from using the pooled estimate (Section 6).

Using $\mathbf{S}_\mathrm{pooled}$, we can discard the first two terms in (18) and use the generalized statistical distance

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

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 (18,22) we can derive the discriminant score

$$\mathrm{d}_\mathrm{MD}(k \mid \mathbf{x}_i) = -\frac{1}{2} d^2_\mathrm{M}(\mathbf{x}_i \mid \bar{\mathbf{x}}_k, \mathbf{S}_\mathrm{pooled}) = \mathrm{d}_\mathrm{LOG}(k \mid \mathbf{x}_i) + \text{const.}, \tag{23}$$

where the constant does not vary with k.

5.3 Linear Discriminant Score

When using the pooled covariance matrix $\mathbf{S}_\mathrm{pooled}$ we can rewrite the distance from (22) as:

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

and observe that the first term is constant for all groups k so we can discard it. That means, that we can now use the following *linear* discriminant score:

$$\mathrm{d}_\mathrm{LINEAR}(k \mid \mathbf{x}_i) = \bar{\mathbf{x}}'_k \mathbf{S}^{-1}_\mathrm{pooled} \mathbf{x}_i - \frac{1}{2} \bar{\mathbf{x}}'_k \mathbf{S}^{-1}_\mathrm{pooled} \bar{\mathbf{x}}_k = \mathrm{d}_\mathrm{MD}(k \mid \mathbf{x}_i) + \text{const.}, \tag{25}$$

which depends *linearly* on $\mathbf{x}_i$ (where const. does not depend on k). Although equivalent, the linear discriminant $\mathrm{d}_\mathrm{LINEAR}$ can be far more efficient to compute than the quadratic d_MD.

5.4 Combining Multiple Attack Traces

We have to combine the n_a individual leakage traces $\mathbf{x}_i$ from $\mathbf{X}_{k\star}$ into the final discriminant score $\mathrm{d}(k \mid \mathbf{X}_{k\star})$. We present two sound options for doing so:

Option 1: Average all the traces in $\mathbf{X}_{k\star}$ (similar to the mean computation in (1)) in order to remove as much noise as possible and then use this single mean trace $\bar{\mathbf{x}}_{k\star}$ to compute

$$\mathrm{d}^{\mathrm{avg}}(k \mid \mathbf{X}_{k\star}) = \mathrm{d}(k \mid \bar{\mathbf{x}}_{k\star}). \tag{26}$$

This option is computationally fast, requiring $O(n_{\mathrm{a}}m + m^2)$ time for any presented discriminant, but it does not use all the information from the available attack traces (in particular the noise).

Option 2: Compute the joint likelihood $\mathrm{l}(k \mid \mathbf{X}_{k\star}) = \prod_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathrm{l}(k \mid \mathbf{x}_i)$. By applying the logarithm to both sides we have $\log \mathrm{l}(k \mid \mathbf{X}_{k\star}) = \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \log \mathrm{l}(k \mid \mathbf{x}_i)$ and we obtain the derived scores:

$$\mathrm{d}^{\mathrm{joint}}_{\mathrm{LOG}}(k \mid \mathbf{X}_{k\star}) = -\frac{n_{\mathrm{a}}}{2} \log |\mathbf{S}_k| - \frac{1}{2} \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} (\mathbf{x}_i - \bar{\mathbf{x}}_k)' \mathbf{S}_k^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}_k), \tag{27}$$

$$\mathrm{d}^{\mathrm{joint}}_{\mathrm{MD}}(k \mid \mathbf{X}_{k\star}) = -\frac{1}{2} \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} (\mathbf{x}_i - \bar{\mathbf{x}}_k)' \mathbf{S}_k^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}_k), \tag{28}$$

$$\mathrm{d}^{\mathrm{joint}}_{\mathrm{LINEAR}}(k \mid \mathbf{X}_{k\star}) = \bar{\mathbf{x}}_k' \mathbf{S}^{-1}_{\mathrm{pooled}} \left(\sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) - \frac{n_{\mathrm{a}}}{2} \bar{\mathbf{x}}_k' \mathbf{S}^{-1}_{\mathrm{pooled}} \bar{\mathbf{x}}_k. \tag{29}$$

Given the n_{a} leakage traces $\mathbf{x}_i \in \mathbf{X}_{k\star}$, $\mathrm{d}_{\mathrm{LOG}}$ and d_{MD} require time $O(n_{\mathrm{a}}m^2)$ while $\mathrm{d}_{\mathrm{LINEAR}}$ only requires $O(n_{\mathrm{a}}m + m^2)$, since the operations $\bar{\mathbf{x}}_k' \mathbf{S}^{-1}_{\mathrm{pooled}}$ and $\bar{\mathbf{x}}_k' \mathbf{S}^{-1}_{\mathrm{pooled}} \bar{\mathbf{x}}_k$ only need to be done once, which is a great advantage in practice. As a practical example, our evaluations of the guessing entropy (see Section 6) for $m = 125$ and $1 \leq n_{\mathrm{a}} \leq 1000$ took about 3.5 *days* with $\mathrm{d}_{\mathrm{LOG}}$ but only 30 *minutes* with $\mathrm{d}_{\mathrm{LINEAR}}$.[9] We note that for $\mathrm{d}_{\mathrm{LINEAR}}$ the computation time is the same regardless of which option we use to combine the traces, and both give the same results for the template attack.

5.5 Unequal Prior Probabilities

In the previous descriptions we have assumed equal prior probabilities among the candidates k. When this is not the case, we only need to add the term $\log P(k)$ to the discriminant scores $\mathrm{d}^{\mathrm{avg}}_{\mathrm{LOG}}$, $\mathrm{d}^{\mathrm{avg}}_{\mathrm{MD}}$, $\mathrm{d}^{\mathrm{avg}}_{\mathrm{LINEAR}}$, or $n_{\mathrm{a}} \log P(k)$ to the discriminant scores $\mathrm{d}^{\mathrm{joint}}_{\mathrm{LOG}}$, $\mathrm{d}^{\mathrm{joint}}_{\mathrm{MD}}$, $\mathrm{d}^{\mathrm{joint}}_{\mathrm{LINEAR}}$.

6 Evaluation of Methods

We evaluated the efficiency of many template-attack variants on a real hardware platform, comparing all the compression methods from Table 1[10] and all

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

[10] We arbitrarily chose to use the DOM estimate, computed as the sum of *absolute* differences between the mean vectors. Using SNR instead of DOM as the signal strength estimate $\mathbf{s}(t)$ has provided very similar results, omitted due to lack of space.

Table 1. List of compression methods evaluated in this paper.

Name	**Description**	m
DOM 1ppc	DOM, 1 sample per clock at most	6–10
DOM 3ppc	DOM, 3 samples per clock at most	18–30
DOM 20ppc	DOM, 20 samples per clock at most	75–79
DOM allap	DOM, all samples above 95th percentile of $\mathrm{F}(t)$	125
PCA	Fixed selection of number of principal components	4
LDA	Fixed selection of number of coefficients	4

the implementation options from Section 5. We compare the commonly used high-compression methods, such as PCA, LDA and sample selection using the guideline [7] of 1 sample per clock at most (*1ppc*), against weak compressions providing a larger number of samples: the *3ppc*, *20ppc* and *allap* selections.[11]

6.1 Experimental Setup

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

For each candidate value $k \in \{0, \ldots, 255\}$ we recorded 3072 traces $\mathbf{x}^{\mathrm{r}}_{ki}$ (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 $m^{\mathrm{r}} = 2500$ 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. All the LOAD instructions require two clock cycles to transfer a value from RAM into a register, using indirect addressing. In all the experiments 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.[12]

[11] 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 $\mathrm{F}(t)$, which varies with n_{p} for each clock edge.

[12] A similar approach was used by Standaert et al. [11] and Oswald and Paar [16] to report results of template attacks on (part of) the key loading stage of a block cipher.

6.2 Guessing Entropy

We use the *guessing entropy* as the sole figure of merit to compare all methods. It estimates the (logarithmic) cost of any optimized search following a template attack to find the correct $k\star$ among the values k with the highest discriminant scores. It gives the expected number of bits of uncertainty remaining about the target value $k\star$. The lower the guessing entropy, the more successful the attack has been and the less effort remains to search for the correct $k\star$.

To compute the guessing entropy, we compute the score $\mathrm{d}(k \mid \mathbf{X}_{k\star})$ (see Section 5) 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 $\mathbf{M}(k\star, k) = \mathrm{d}(k \mid \mathbf{X}_{k\star})$. 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 depth matrix $\mathbf{D} \in \mathbb{N}^{|\mathcal{S}|\times|\mathcal{S}|}$ with

$$\mathbf{D}(k\star, k) = \text{position of } \mathrm{d}(k \mid \mathbf{X}_{k\star}) \text{ in the sorted row of } \mathbf{M}(k\star, \cdot). \tag{30}$$

Finally, using the matrix $\mathbf{D}$ we define the guessing entropy

$$g = \log_2 \frac{1}{|\mathcal{S}|} \sum_{k\in\mathcal{S}} \mathbf{D}(k, k). \tag{31}$$

Standaert et al. [13] also used this measure, but without the logarithm.

6.3 Experimental Results and Practical Guidance

We performed each attack 10 times for each combination of n_a, k and $k\star$, using a different random selection of $\mathbf{X}_{k\star}$ for each n_a and $k\star$. We plot in Figure 2 and 3 the averaged guessing entropy, resulting in highly reproducible graphs. The standard deviation across all experiments is around 0.1 bits.

These results, as well as the considerations discussed earlier, allow us to provide the following practical guidance regarding the choice of algorithm:

1. *Use Option 2 ($\mathrm{d}^{\mathrm{joint}}$) in preference to Option 1 ($\mathrm{d}^{\mathrm{avg}}$) to combine the discriminant scores for $n_\mathrm{a} > 1$ attack traces.* For $n_\mathrm{a} = 1$ or when using $\mathbf{S}_{\mathrm{pooled}}$, these options are equivalent. Otherwise, as the number n_a of attack traces increases and the covariance matrix is better estimated (e.g. due to a large number n_p of profiling traces or small number m of variables) $\mathrm{d}^{\mathrm{joint}}$ outperforms $\mathrm{d}^{\mathrm{avg}}$ for all compression methods.
2. *Try using a common covariance matrix $\mathbf{S}_{\mathrm{pooled}}$ with $\mathrm{d}_{\mathrm{LINEAR}}$* (unless differences between individual estimates $\mathbf{S}_k$ are very evident, e.g. from visual inspection). Failing a statistical test for homoscedasticity (e.g., Box's test) alone does not imply that using individual estimates $\mathbf{S}_k$ will improve the template attack. Using individual estimates $\mathbf{S}_k$ prevents use of the significantly faster and more robust discriminant $\mathrm{d}_{\mathrm{LINEAR}}$. Then:
   (a) *If your target allows you to acquire a large number of traces ($n_\mathrm{a} > 100$):* try the compression methods PCA, LDA and sample-selection with *large* m since they may perform differently based on the level of noise from the profiling traces $\mathbf{X}_k$.

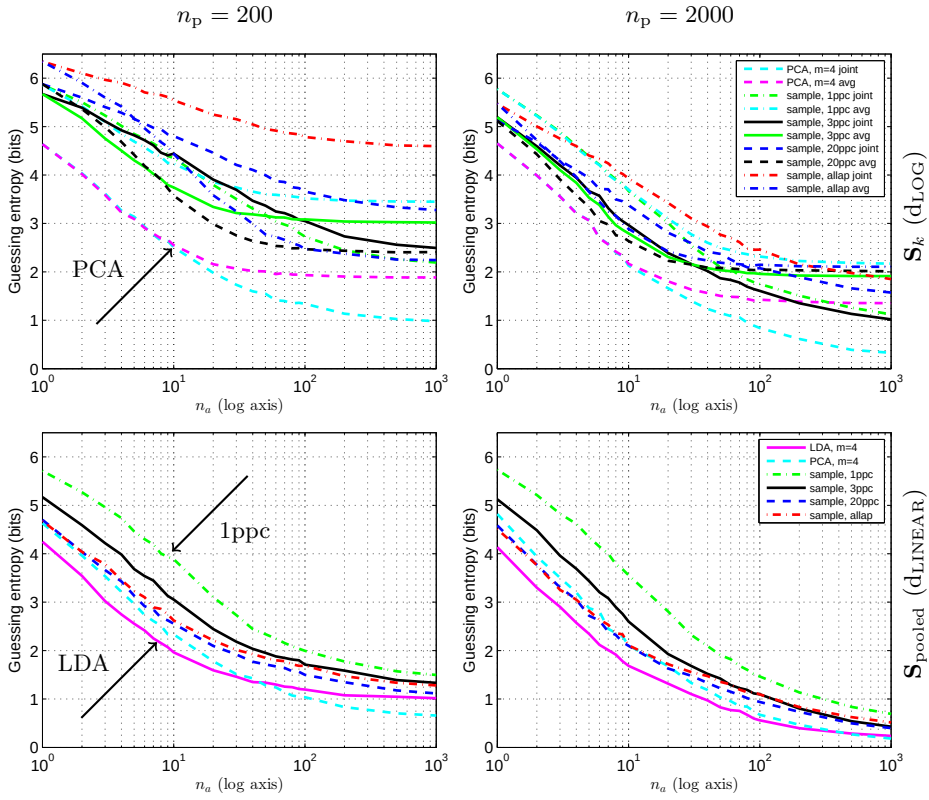

Fig. 2. Guessing entropy remaining after template attacks, with different compressions, for $n_\mathrm{p} = 200$ (left) and $n_\mathrm{p} = 2000$ (right) profiling traces, using individual covariances $\mathbf{S}_k$ with d_LOG (top) or a pooled covariance $\mathbf{S}_\mathrm{pooled}$ with $\mathrm{d}_\mathrm{LINEAR}$ (bottom).

(b) *If your target allows only acquisition of a limited number of attack traces ($n_\mathrm{a} < 10$):* use LDA. Note that in this case, as the covariance estimate improves due to large $|\mathcal{S}|n_\mathrm{p}$, performance increases with larger m (cf. *3ppc*, *20ppc*, *allap*). In particular, for $n_\mathrm{a} < 10$, we see in Figure 2 (bottom) that we got more than 1 bit of data from *20ppc* and *allap* compared to *1ppc*, which contradicts the claim [7, Section 3.2] that "additional [samples] in the same clock cycle do not provide additional information". In this setting, *20ppc* and *allap* can outperform PCA.

3. *If you cannot use the pooled covariance $\mathbf{S}_\mathrm{pooled}$, then use the individual covariances $\mathbf{S}_k$ with d_LOG and use PCA as the compression method.*

This guidance should work well in situations similar to our experimental conditions. Further research is needed to also consider pipelining, where other data in neighbour instructions can partially overlap in the side-channel.

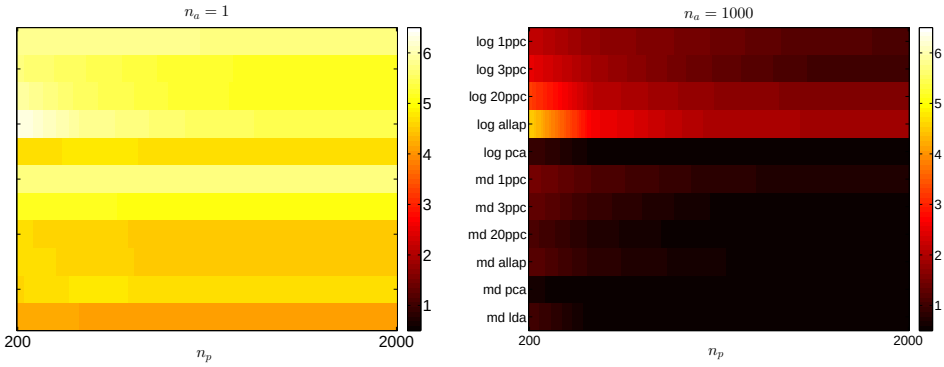

Fig. 3. Guessing entropy from the methods discussed, for $n_\mathrm{a} = 1$ (left) and $n_\mathrm{a} = 1000$ (right), using $\mathrm{d}^{\mathrm{joint}}$ (at $n_\mathrm{p} \in \{200, 500, 1000, 1500, 2000\}$, linearly interpolated).

7 Conclusions

In this paper, we have explored in detail the implementation of template attacks based on the multivariate normal distribution, comparing different compression methods, discriminant scores, and number of profiling and attack traces.

We explained why several numerical obstacles arise when dealing with a large number m of variables (e.g. when retaining a large part of the leakage vectors), and we presented efficient methods that can be used in this case, such as the discriminant $\mathrm{d}_{\mathrm{LOG}}$.

Based on the observation that the covariance matrices $\mathbf{S}_k$ of each candidate k are similar, we explained the use of the pooled covariance estimate $\mathbf{S}_{\mathrm{pooled}}$ and we showed how $\mathbf{S}_{\mathrm{pooled}}$ allows us to derive a *linear* discriminant $\mathrm{d}_{\mathrm{LINEAR}}$ which is much more efficient than $\mathrm{d}_{\mathrm{LOG}}$. For $n_\mathrm{a} = 1000$ attack traces and $m = 125$ samples, the computation of the guessing entropy remaining after the template attacks can be reduced from 3 days (using $\mathrm{d}_{\mathrm{LOG}}$) to 30 minutes (using $\mathrm{d}_{\mathrm{LINEAR}}$). This is a great advantage for the evaluation of template attacks, which is often a requirement to obtain Common Criteria certification.

We applied all the methods presented in this paper on real traces from an unprotected 8-bit microcontroller and we evaluated the results using the guessing entropy. Using the efficient methods presented in this paper we were able to obtain a guessing entropy close to 0, i.e. we are able to extract *all* 8 bits processed by a *single* LOAD instruction, not just their Hamming weight.

Based on these results and theoretical arguments, we proposed a practical guideline for the choice of algorithm when implementing template attacks.

Data and Code Availability: In the interest of reproducible research we make available our data and associated MATLAB scripts at:

`http://www.cl.cam.ac.uk/research/security/datasets/grizzly/`

Acknowledgement: Omar Choudary is a recipient of the Google Europe Fellowship in Mobile Security, and this research is supported in part by this Google Fellowship. The opinions expressed in this paper do not represent the views of Google unless otherwise explicitly stated.

References

1. P.C. Mahalanobis, "On the Generalised Distance in Statistics". In: Proceedings National Institute of Science, India, vol. 2, pp 49–55, 1936.
2. R.A. Fisher, "The Statistical Utilization of Multiple Measurements", in Annals of Eugenics, 8, pp 376–386, 1938.
3. G.E.P. Box, "Problems in the Analysis of Growth and Wear Curves", in Biometrics, 6, pp 362–389, 1950.
4. S. Chari, J. Rao, and P. Rohatgi, "Template Attacks", CHES 2002, Springer, 2003, LNCS 2523, pp 51–62.
5. O. Ledoit, M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices", in Journal of Multivariate Analysis, 2004.
6. I. Jolliffe, "Principal Component Analysis". John Wiley & Sons, Ltd, 2005.
7. C. Rechberger and E. Oswald, "Practical Template Attacks", in Information Security Applications, Springer, 2005, LNCS 3325, pp 440–456.
8. C. Archambeau, E. Peeters, F. Standaert, and J. Quisquater, "Template Attacks in Principal Subspaces", in CHES 2006, Springer, 2006, LNCS 4249, pp 1–14.
9. B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. Stochastic Methods", in CHES 2006, Springer, LNCS 4249, pp 15–29.
10. R. Johnson and D. Wichern, "Applied Multivariate Statistical Analysis", 6th ed. Pearson, 2007.
11. F.-X. Standaert and C. Archambeau, "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages", in CHES 2008, Springer, LNCS 5154, pp 411–425.
12. L. Batina, et al. "Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip", Information Security, 2008, LNCS 5222, pp 341–354.
13. F.-X. Standaert, et al. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks", Eurocrypt 2009, LNCS 5479, pp 443–461.
14. T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a Side Channel Based Disassembler", Trans. on Computational Science X, 2010, vol. 6340, pp 78–99.
15. S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards", 1st ed., Springer, 2010.
16. D. Oswald and C. Paar, "Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World", in CHES 2011, LNCS 6917, pp 207–222.

A Evaluation Board

For our experiments, we built a custom PCB for the Atmel microcontroller (see Figure 4, left). This 4-layer PCB has inputs for the clock signal and supply voltage, a USB port to communicate with a PC, and a 10-ohm resistor in the ground line for power measurements. The PCB connects all the ground pins of the microcontroller to the same line, which leads to the measurement resistor.

B Executed Code

During all our experiments we recorded traces with 2500 samples, covering the execution of several instructions, as shown in Figure 4 (right). The executed instruction sequence is

```
5a5c:   00 00   nop             ; several previous NOPs ommited in this listing
5a5e:   fc 01   movw  r30, r24  ; 1 clock cycle, recorded traces start here
5a60:   81 90   ld  r8, Z+      ; 2 clock cycles per ld instruction
5a62:   91 90   ld  r9, Z+      ; this is our target instruction (2 clock cycles)
5a64:   a1 90   ld  r10, Z+     ; we want to infer the data loaded in r9
5a66:   b1 90   ld  r11, Z+
5a68:   c1 90   ld  r12, Z+     ; recorded trace ends after first clock cycle of this ld
```

The load instructions use the Z pointer (which refers to registers r31:r30) for indirect RAM addressing. The initial value of registers r8–r12 before the load operations is zero. The initial value of Z before the first load instruction is 2020.

C Some Proofs

In Section 5.3 we rewrote (22) as (24). This is possible because

$$\bar{\mathbf{x}}_k' \mathbf{S}_{\text{pooled}}^{-1} \mathbf{x} = (\bar{\mathbf{x}}_k' \mathbf{S}_{\text{pooled}}^{-1} \mathbf{x})' = \mathbf{x}' \mathbf{S}_{\text{pooled}}^{-1}{}' \bar{\mathbf{x}}_k = \mathbf{x}' \mathbf{S}_{\text{pooled}}^{-1} \bar{\mathbf{x}}_k. \tag{32}$$

In Section 5.4 we state that $\mathrm{d}_{\text{LINEAR}}$ provides the same results for both options of combining the traces (from average trace and based on joint likelihood). This happens because if we let $c_k = -\frac{1}{2} \bar{\mathbf{x}}_k' \mathbf{S}_{\text{pooled}}^{-1} \bar{\mathbf{x}}_k$ for any k, then we have

$$\mathrm{d}_{\text{LINEAR}}^{\text{joint}}(k \mid \mathbf{X}_{k\star}) = \bar{\mathbf{x}}_k' \mathbf{S}_{\text{pooled}}^{-1} \left(\sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + n_{\text{a}} c_k, \tag{33}$$

$$\mathrm{d}_{\text{LINEAR}}^{\text{avg}}(k \mid \mathbf{X}_{k\star}) = \bar{\mathbf{x}}_k' \mathbf{S}_{\text{pooled}}^{-1} \left(\frac{1}{n_{\text{a}}} \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + c_k, \tag{34}$$

and therefore for any $u, v \in \mathcal{S}$ it is true that

$$\begin{gathered}
\mathrm{d}_{\text{LINEAR}}^{\text{avg}}(u \mid \mathbf{X}_{k\star}) > \mathrm{d}_{\text{LINEAR}}^{\text{avg}}(v \mid \mathbf{X}_{k\star}) \Leftrightarrow \\
\bar{\mathbf{x}}_u' \mathbf{S}_{\text{pooled}}^{-1} \left(\frac{1}{n_{\text{a}}} \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + c_u > \bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \left(\frac{1}{n_{\text{a}}} \sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + c_v \Leftrightarrow \\
\bar{\mathbf{x}}_u' \mathbf{S}_{\text{pooled}}^{-1} \left(\sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + n_{\text{a}} c_u > \bar{\mathbf{x}}_v' \mathbf{S}_{\text{pooled}}^{-1} \left(\sum_{\mathbf{x}_i \in \mathbf{X}_{k\star}} \mathbf{x}_i \right) + n_{\text{a}} c_v \Leftrightarrow \\
\mathrm{d}_{\text{LINEAR}}^{\text{joint}}(u \mid \mathbf{X}_{k\star}) > \mathrm{d}_{\text{LINEAR}}^{\text{joint}}(v \mid \mathbf{X}_{k\star}).
\end{gathered}$$

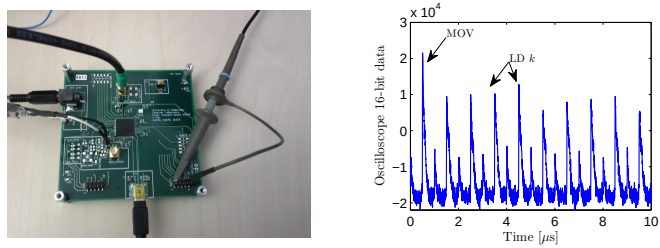

Fig. 4. Left: the device used during our experiments. Right: A single example trace $\mathbf{x}_i^{\text{r}}$ from our experimental setup.