J Cryptogr Eng (2013) 3:99–110
DOI 10.1007/s13389-012-0038-y

REGULAR PAPER

Improving cross-device attacks using zero-mean unit-variance normalization

David P. Montminy · Rusty O. Baldwin · Michael A. Temple · Eric D. Laspe

Received: 5 July 2012 / Accepted: 15 August 2012 / Published online: 29 September 2012

Abstract Template attacks are a very powerful form of side-channel analysis. It is assumed an adversary has access to a training device, identical to the device under attack, to build a precise multivariate characterization of the side-channel emissions. The training and test devices are assumed to have identical, or at least very similar, electromagnetic emissions. Often, when evaluating the effectiveness of a template attack, training and test data are from the same-device. The effectiveness of collecting training and test data from different devices, or cross-device attacks, are evaluated here using 40 PIC microcontroller devices. When the standard template attack methodology fails to produce adequate results, each step is evaluated to identify device-dependent variations. A simple pre-processing technique, normalizing the trace means and variances from the training and test devices, is evaluated for various test data set sizes. This step improves the success key-byte extraction rate for same part number cross-device template attacks from 65.1 to 100 % and improves attacks against similar devices in the same-device family. Additionally, it is demonstrated that due to differences in device leakage, minimizing the number of distinguishing features reduces the effectiveness of cross-device attacks.

Keywords Template attack · Cross-device · Zero mean unit variance · Normalization · PCA

D. P. Montminy · R. O. Baldwin (✉) · M. A. Temple · E. D. Laspe
Department of Electrical and Computer Engineering,
United States Air Force Institute of Technology, 2950 Hobson Way,
Wright-Patterson AFB, OH 45433, USA
e-mail: rusty.baldwin@afit.edu
URL: http://www.afit.edu

1 Introduction

Template attacks [1] are a form of two-stage profiling attack, with the initial stage obtaining 'a priori' knowledge of the side-channel leakage for a specific device. The profiling, or training stage estimates the multivariate probability densities of observable side-channels for the targeted key-dependent internal state of the cryptographic implementation. The estimated probability densities are used during the attack phase to determine the device's internal state. The key assumption for a profiling attack is that a powerful attacker has access to a training device, identical to the target device, over which he has full control. The training device is used to create a precise multivariate model of the device's side-channel leakage for each key dependency. Implicit in using a training device is that both devices produce similar side-channel emissions. This assumption was originally introduced in [1] and has since been repeatedly accepted without challenge [2–7].

It has recently been shown that in addition to operation- and data-dependent components of electromagnetic (EM) emissions, the emissions exhibit significant device-dependent characteristics [8]. This is likely due to random process variations introduced during fabrication and packaging [9]. Although the structural variations introduced in the manufacturing process are relatively small, and the devices produced meet the desired specifications, no two chips are exactly alike. Therefore, the emissions produced by similar devices are indeed similar to some degree but not identical. These variations are significant enough to allow a specific device to be uniquely identified based only on the devices EM emissions [10]. The work here examines the differences in cross-device emissions to determine if such differences are sufficient enough to prevent template attacks from being effective if *similar* devices are used for training and testing, versus using the *same* device for training and testing.

Template attack research has expanded the capabilities of template attacks to use multiple test traces [5], multiple side-channels [11], reduced the number of features required to build templates using heuristics [12] and systematic methods [2,6], and employed templates to defeat countermeasures [5,7]. Template attacks have been adopted as an attack methodology without evaluating the underlying assumption that the power consumption and EM emissions from two separate devices are sufficiently similar to make the attacks practical. In each of the papers cited above, the same smartcard or microprocessor was used to create both the training and test data.

Research here focuses on the EM side-channel of a device performing AES encryption operations. Unlike power consumption methods, the EM side-channel can be collected without physically modifying the device. This makes repeating the collection process more difficult. Instead of monitoring the voltage change across a single shunted resistor, careful consideration must be given to placing the EM probe in exactly the same position and configuration between collections on a device. Even template attacks performed using the same-device can fail if the probe is moved between the collection of training and test traces. Recently, differences in collection equipment, methodology, synchronization, and target device age where shown to reduce the effectiveness of template attacks even when attacking using templates created with the same device [13]. This paper explores the differences between devices.

The remainder of this paper is organized as follows: Background on the target cipher, the Advanced Encryption Standard (AES) [14], and template attacks are presented in Sect. 2. Differences in side-channel emissions between devices and development of the mean and variance normalization technique are explored in Sect. 3, followed by the experimental methodology in Sect. 4. Results are presented in Sect. 5 followed by the conclusion in Sect. 6.

2 Background

2.1 Advanced Encryption Standard (AES)

Devices running AES can be targeted using template attacks. AES-128 uses a 128-bit key to encrypt a 128-bit plaintext. Differential side-channel attacks, including template attacks, attempt to determine the secret key by measuring and analyzing the small statistical influence the computation of intermediate values has on the power or EM side-channel [15].

The first round of AES-128 performs the following computations:

1. *Initialization:* The 16-byte plaintext is loaded into the 4×4 byte state matrix.
2. *AddRoundKey:* The state matrix is XOR-ed with the original 16-byte key.
3. *SubBytes:* Each byte of the state matrix is substituted for another byte value based on the non-linear invertible substitution table (s-box) defined in [14].
4. *ShiftRows:* The last three rows of the state matrix are cyclically shifted column-wise using different offsets.
5. *Mixed Columns:* The state matrix is mixed column by column using a linear operation.

Outputs of the AddRoundKey and SubBytes operations are common targets for side-channel attacks. Since these are byte-wise computations, each byte can be considered separately. Let $P_{i,n}$ denote the nth byte of the ith input plaintext (which corresponds to the ith trace) and let K_n denote the nth byte of the secret key. Let $I_{i,n}$ represent the value of the intermediate value after the XOR operation, $I_{i,n} = P_{i,n} \oplus K_n$. Assuming $P_{i,n}$ is known and K_n is unknown, a differential attack tries to determine the most likely candidate K_n byte value based on the collected side-channel observation $I_{i,n}$. The input to SubBytes $I_{i,n}$ is the targeted intermediate value in this attack. This intermediate value is chosen because it allows for the highest same-device key extraction success rate in testing.

2.2 Template attack methodology

A profiling stage can be used to build multivariate statistical models of the device's side-channel leakage [1]. Incorporating a profiling stage allows the template attack to use all information present in a side-channel trace for classification, making them a strong attack even when only a single or few traces from the attacked device are available. Rather than try to eliminate or reduce noise, the noise present in the side-channel emission is assumed to be key dependent and precisely modeled. The profiling stage creates mean vectors and covariance matrices for each of the possible sub-keys. Agrawl et al. [16] expanded template attacks to differential power analysis and incorporated data to allow multiple traces from multiple side-channels. Oswald and Mangard [5] incorporated Bayes' theorem to allow multiple traces to be used during the classification phase.

Templates based on hypothesized intermediate values calculate each possible intermediate value using a hypothesized key-byte value and a known plaintext or ciphertext. The collected emissions used to create templates are referred to as *training* traces. During the attack phase of a template attack, the posterior probabilities, or the probability an observed trace or collection of traces comes from a specific class, is calculated using a classifier. The maximum likelihood (ML) decision rule is based on the notion that choosing the key guess with the highest posteriori probability will minimize the probability that an incorrect class is selected [17].

Traces observed during the attack phase are referred to as *test* traces. In this article, template attacks performed with training and test data from the same-device are referred to as *same-device* attacks. Attacks performed with training and test data from different devices are referred to as *cross-device* attacks.

Since the introduction of template attacks in [1], a number of variations and improvements have been proposed. However, all template attacks fundamentally contain the following steps:

- *Identify classes*. The goal of a template attack is to correctly determine to which category or *class* an observed trace (or set of traces) from a target device belongs. The number and definition of the classes is determined by the attack scenario and the type of information leaked from target device. When attacking a microprocessor running AES, classes are commonly based on byte value (256 classes) [1,12], byte Hamming weight (9 classes) or bit value (2 classes) [7].
- *Data collection.* The training device and test device must both be observed performing encryption operations. It is assumed the attacker has complete control of the training device, can change the key and plaintext at will, and can associate a collected trace with the plaintext and key used to produce it. While the key on the target device is always unknown, some attack scenarios assume a powerful attacker is able to match observed test traces with corresponding plaintext or ciphertext.
- *Feature generation*. Preprocessing techniques may be applied to the traces or the extracted samples before they are used for training or classification. Examples of preprocessing techniques include Principle Component Analysis, Fourier transforms of the collected traces, or trace normalization.
- *Feature extraction.* The samples in the collected or preprocessed traces that distinguish between classes are identified and extracted from each trace.
- *Classifier Training.* Using the known plaintexts and keys from the training phase, the attacker can estimate the class from which the observed training trace belongs. One template is created for each class using the extracted distinguishing features from the training traces belonging to that class (Ref. to Sect.2.4).
- *Classification* Using distinguishing features generated from one or more test traces, the classifier estimates the class to which the test traces most likely belong. If the plaintexts or ciphertexts are known, hypothetical intermediate values may be used in this process.

The remainder of this section provides additional information for the more complicated template attack steps.

2.3 Class identification

To evaluate the effectiveness of cross-device attacks, a simple DPA template attack is performed on an unprotected implementation of AES-128 running on $N_D = 40$ different PIC microcontroller devices. The attacker has complete control over a training device, is able to collect EM emissions from the target device, and has knowledge of the plaintexts associated with each collected trace. This allows hypothetical intermediate values to be calculated based on key-byte guesses. Let s designate the key-byte class where $s \in \{0, 1, \ldots, K-1\}$ where K is the total number of classes. The attack considered here use $K = 256$ classes, one for each key-byte guess. Different templates are constructed for each intermediate value byte being attacked.

Although distinguishing features are generated and selected before constructing templates in an actual attack, it is more insightful to discuss the rationale for feature selection after explaining the mechanics of template attacks. Methods for selecting distinguishing features are discussed in Sect. 2.7.

2.4 Classifier training

The classifier is trained by constructing templates for each class. A fundamental assumption is that side-channel leakage for a particular operation follows a multivariate Gaussian distribution. This assumption has been shown to provide adequate performance in previous template attack research [1,2,5,7,17]. Let vector $\mathbf{x}$ be the list of γ distinguishing features. The probability density function of a γ-dimensional multivariate normal distribution is

$$p(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_{k_j}\right)^T \hat{\boldsymbol{\Sigma}}_{k_j}^{-1}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_{k_j}\right)\right)}{(2\pi)^{\gamma/2}\left|\hat{\boldsymbol{\Sigma}}_{k_j}\right|^{1/2}}, \tag{1}$$

where empirical mean vector $\hat{\boldsymbol{\mu}}_{k_j}$ and empirical noise covariance matrix, $\hat{\boldsymbol{\Sigma}}_{k_j}$, form the template of class k_j. One template is constructed for each of the $K = 256$ possible byte values, $k_j \in \{0, \ldots, 255\}$. The estimates are constructed using N_{k_j} distinguishing feature vectors that belong to class k_j. Each distinguishing feature vector is represented as $\mathbf{x}_i$ where $i \in \{1, \ldots, N_{k_j}\}$.

The empirical mean vector, $\hat{\boldsymbol{\mu}}_k$, and the $\gamma \times \gamma$ empirical noise covariance matrix, $\hat{\boldsymbol{\Sigma}}_k$, are

$$\hat{\boldsymbol{\mu}}_{k_j} = \frac{1}{N_{k_j}} \sum_{i=1}^{N_{k_j}} \mathbf{x}_{k_j,i}, \quad \text{and} \tag{2}$$

$$\hat{\boldsymbol{\Sigma}}_{k_j} = \frac{1}{N_{k_j}-1} \sum_{i=1}^{N_{k_j}} \left(\mathbf{x}_{k_j,i}-\hat{\boldsymbol{\mu}}_{k_j}\right)\left(\mathbf{x}_{k_j,i}-\hat{\boldsymbol{\mu}}_{k_j}\right)^T, \tag{3}$$

respectively.

2.5 Classifying observed traces

The template attack is performed for each of the 16 key-byte values in AES-128. There are $K = 256$ possible round key values, $k_j \in \{0, \ldots, 255\}$. For matrix $\mathbf{X}$, which contains the distinguishing features from one trace in each row, the probability that a key-byte guess is correct is [17]

$$p\left(k_j \middle| \mathbf{X}\right) = \frac{\left(\prod_{i=1}^{N_t} p\left(\mathbf{x}_i^T \middle| k_j\right) \cdot \left(k_j\right)\right)}{\sum_{l=0}^{K-1}\left(\left(\prod_{i=1}^{N_t} p\left(\mathbf{x}_i^T \middle| k_l\right)\right) \cdot \left(k_l\right)\right)}, \tag{4}$$

where i is the index of the N_t test traces.

Since AES key-bytes are uniformly distributed, it is initially assumed that $p\left(k_l = T_b^{r+1}\right) = 1/256$. The Bayesian classification process produces the probabilities $p\left(k_j \middle| \mathbf{X}\right)$ for all $j \in \{0, \ldots, 255\}$.

2.6 Class selection

After $p\left(k_j \middle| \mathbf{X}\right)$ is calculated for each possible round key value, the most likely key-byte value is

$$\hat{k}_j = \arg\max_{k_j} p\left(k_j \middle| \mathbf{X}\right). \tag{5}$$

2.7 Distinguishing feature selection

Device EM traces are typically collected at a very high sampling rate resulting in a large number of samples ($N_s > 10^4$). Building templates based on every sample is not feasible due to storage requirements of the covariance matrix and complexity of matrix inversion to calculate the observation probability [12].

The processing time and complexity of constructing the templates can be reduced by identifying n out of N points that provide the most information to the template attack. Since these samples must allow classes to be distinguished from each other, they are referred to as *distinguishing features* herein. They are also referred to as *points of interest* in related literature.

Previous research has focused on improving how distinguishing features are generated and selected. A number of heuristic approaches have been proposed, including selecting samples with the largest difference between mean traces [1], or the point at which the largest variance between the mean traces (for each class) occurs. Benefits of pre-processing using a Fast Fourier Transform before selecting the samples, with the highest cumulative difference between pairs of mean traces, were evaluated in [12]. Requiring a minimum number of samples between successive selected time samples has also been proposed as a way to reduce the number of distinguishing features by reducing redundant information [12].

Principal Component Analysis (PCA). While heuristic methods for selecting distinguishing features have been effective, more systematic approaches have been developed. PCA can reduce the dimensionality of trace data using a linear transform that maximizes the inter-class variance between empirical mean traces $\{\hat{\boldsymbol{\mu}}_s\}_{s=1}^{K}$ for each class in the subspace [2]. To find this transform, PCA identifies the principal directions $\{\mathbf{w}_i\}_{i=1}^{N_p}$ such that $N_p \leq N_s$, which forms an orthonormal basis capturing the maximal variance of $\{\hat{\boldsymbol{\mu}}_s\}_{s=1}^{K}$ in an N_p-dimensional subspace. The principal directions are the eigenvectors $\mathbf{U}$ of the empirical covariance matrix

$$\bar{\boldsymbol{S}} = \frac{1}{K}\sum_{s=1}^{K}\left(\hat{\boldsymbol{\mu}}_s - \bar{\boldsymbol{\mu}}\right)\left(\hat{\boldsymbol{\mu}}_s - \bar{\boldsymbol{\mu}}\right)^{\mathrm{T}}, \quad \bar{\boldsymbol{S}} = \mathbf{U}\Delta\mathbf{U}^{\mathrm{T}}.$$

where $\bar{\boldsymbol{\mu}} = \frac{1}{K}\sum_{s=1}^{K}\hat{\boldsymbol{\mu}}_s$ is the average of the mean traces. The principal directions $\{\mathbf{w}_i\}_{i=1}^{N_p}$ are the columns of $\mathbf{U}$ that correspond to the N_p largest eigenvalues of Δ. The N_p-largest eigenvalues are denoted by the diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{N_p \times N_p}$ and the corresponding matrix of principal directions is denoted $\mathbf{W} \in \mathbb{R}^{N_s \times N_p}$. To perform an attack in the principal subspace, a Gaussian model after projection is assumed. The projected means $\{\boldsymbol{v}_s\}_{s=1}^{K}$ and projected covariance matrices $\{\boldsymbol{\Lambda}_s\}_{s=1}^{K}$ are given by

$$\boldsymbol{v}_k = \mathbf{W}^{\mathrm{T}}\hat{\boldsymbol{\mu}}_k \quad \text{and,} \tag{6}$$

$$\Lambda_k = \mathbf{W}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{W}. \tag{7}$$

A collection of traces from the test device, $\mathbf{X}$, is classified by

$$\hat{k}_j = \arg\max_{k_j} p\left(k_j \middle| \mathbf{W}^{\mathrm{T}}\mathbf{X}\right) \tag{8}$$

While PCA reduces the number of distinguishing features required for an attack, it may not improve the attack's classification performance. Therefore, PCA is implemented here and compared with a heuristic approach based on correlation analysis. The process used to identify distinguishing features using correlation analysis is outlined in Sect. 4.2. A comparison of PCA and correlation selected distinguishing features for each of the training devices is presented in Sect. 5.1.

3 Cross-device EM leakage

The EM side-channel can be divided into various components as was done for the power consumption side-channel in [17]. Each sample in the EM side-channel trace is made up of various components: a operation-dependent component S_{op}, a data-dependent component S_{data}, electronic noise $S_{\mathrm{el.\,noise}}$, and a constant component S_{const}. A sample in the total EM side-channel trace is a sum of these components, or

$$S_{\mathrm{total}} = S_{\mathrm{op}} + S_{\mathrm{data}} + S_{\mathrm{el.\,noise}} + S_{\mathrm{const}}. \tag{9}$$

The distribution of $S_{\text{el. noise}}$ in a microprocessor has been shown to be $S_{\text{el. noise}} \sim \mathcal{N}\left(0, \sigma^2_{\text{el.noise}}\right)$. The contribution of S_{data} is proportional to the Hamming weight of the data being processed and its distribution can be approximated with the normal distribution when the data are uniformly distributed, or $S_{\text{data}} \sim \mathcal{N}\left(0, \sigma_{\text{data}}\right)$. Note that $\sigma_{\text{el.noise}}$ and σ_{data} are device specific.

For a differential side-channel attack where only the data are changing between traces, it can be assumed that $Var\left(S_{\text{const}}\right) = Var\left(S_{\text{op}}\right) = 0$, and $E\left(S_{\text{op}}\right) = E\left(S_{\text{data}}\right) = E\left(S_{\text{el.noise}}\right) = 0$. Therefore, $E\left(S_{\text{total}}\right) = E\left(S_{\text{const}}\right) = \mu_{\text{const}}$, where μ_{const} is a constant contribution for the operation being performed at a specific time on a specific device.

Assuming S_{data} and $S_{\text{el.noise}}$ are statistically independent, $S_{\text{data}} + S_{\text{el.noise}} \sim \mathcal{N}\left(0, {\sigma^2}_{\text{data}} + {\sigma^2}_{\text{el. noise}}\right)$ and $S_{\text{total}} \sim \mathcal{N}\left(\mu_{\text{const}}, \sigma^2_{\text{data}} + \sigma^2_{\text{el. noise}}\right)$. For a cross-device template attack to be successful, μ_{const} and $\sigma^2_{\text{data}} + \sigma^2_{\text{el. noise}}$ must be consistent across devices at samples identified as distinguishing features.

Figure 1 uses a violin plot [21] to show the distributions of 5000 observations of sample (#972) for $N_D = 40$ different training devices. These 40 devices are from the same family of PIC microcontrollers, with devices within groups of 10 (denoted A, B, C and D) having identical part numbers. More information on the devices can be found in Sect. 4.1. It is shown in Sect. 5.2 that if the differences in means and variance between device groups are not compensated for some template attacks will fail.

3.1 Compensating for device differences

To compensate for the device-to-device differences in μ_{const} and ${\sigma^2}_{\text{data}} + {\sigma^2}_{\text{el. noise}}$, a transformation of variables is performed. The transformation ensures the test data have approximately the same distribution as the training data.

Using collected training and test data at a specific time sample collected across multiple traces, the mean and variance of that sample can be estimated for each data set. Let variable X_{train} represent the value of EM traces at a specific sample in the training data and let variable X_{test} represent the value of the test data at the same sample.

Let $\hat{\mu}_{\text{train}}$ and $\hat{\sigma}^2_{\text{train}}$ be the estimated mean and variance of X_{train}, and let $\hat{\mu}_{\text{test}}$ and $\hat{\sigma}^2_{\text{test}}$ represent the estimated distribution of the test data. X_{test} is used to calculate transformed X'_{test} having the same mean and variance as the training data using

$$X'_{\text{test}} = \frac{\left(X_{\text{test}} - \hat{\mu}_{\text{test}}\right)}{\hat{\sigma}_{\text{test}}} \hat{\sigma}_{\text{train}} + \hat{\mu}_{\text{train}}. \tag{10}$$

Test data transformation is performed for each sample selected as a distinguishing feature for the template attack.

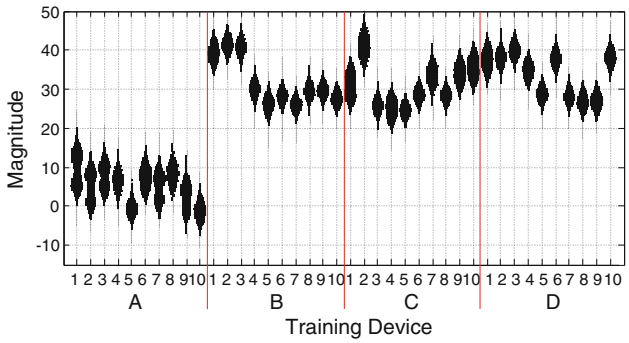

Fig. 1 Violin plots showing the distribution of 5,000 observations of sample #972 on an AES encryption operations with random plaintext and keys for $N_D = 40$ similar microcontroller devices

Alternatively, both X_{test} and X_{train} can be transformed to the standard normal via

$$X'_{\text{test}} = \frac{\left(X_{\text{test}} - \hat{\mu}_{\text{test}}\right)}{\hat{\sigma}_{\text{test}}}, \quad \text{and} \tag{11}$$

$$X'_{\text{train}} = \frac{\left(X_{\text{train}} - \hat{\mu}_{\text{train}}\right)}{\hat{\sigma}_{\text{train}}}. \tag{12}$$

In order to reduce attack time and eliminate the need to retain the training data, the classifier can be trained using (2) and (3) before collecting the test traces. Transforming both X_{test} and X_{train} eliminates the need to retain the training data or store $\hat{\mu}_{\text{train}}$ and $\hat{\sigma}_{\text{train}}$ for each distinguishing feature.

The remaining steps of the template attack are performed as usual using transformed test data. This process is referred to as the zero mean, unit variance normalization (MVN) technique herein. Although this technique was developed independently for this research, it was first published in [13] as a way to compensate for differences in the collected trace sets from the same device before and after device modifications and aging.

4 Experimental methodology

The experimental setup used to collect the data and the specific analysis methodology is described below.

Targeted devices. The following methodology was used to perform a template attack on 40 unprotected 16-bit PIC24F microcontrollers [18]. The devices include ten unique chips with four different part numbers, shown in Table 1. These devices were selected because they have similar device architectures. The ten chips from each part number were all manufactured in the same lot and are representative of low cost microcontrollers used in various embedded applications.

Although all devices have the same basic architecture, Part *A* devices have several on-board peripherals that are not included in the other three. Parts *B*, *C* and *D* devices

Table 1 Tested PIC micro-controller device classes

Part class	Device numbers	PIC part number
A	A1–A10	PIC24FJ64GA102 I/SP
B	B1–B10	PIC24FJ64GA002 I/SP
C	C1–C10	PIC24FJ48GA002 I/SP
D	D1–D10	PIC24FJ32GA002 I/SP

all have identical architectures with the exception of the amount of on-board flash RAM which is 32, 48, and 64 KB of RAM, respectively. Part *A* devices have 64 KB of RAM. Individual devices are referenced by an alphanumerical device number that includes part type and chip number, i.e., $A1, A2, \ldots, D9, D10$, as shown in Table 1.

The chips were fabricated using an unspecified 180 nm process. Since all ten chips for each part number were produced in the same lot, they contain identical architectural features. Uncontrolled manufacturing variations in the die fabrication and packaging process are believed to be the only physical differences between devices with the same part number.

4.1 Data collection

The collection process was designed to make measurements as repeatable as possible. Each microcontroller was mounted to a single evaluation board and programmed to respond to commands over a RS-232 serial interface. The C++ code used to implement the AES-128 algorithm on each device is identical, but the compiled versions vary slightly due to differences in part specific header files. To improve trace alignment, a trigger signal was programmed to go high immediately before the encryption operation started and to go low immediately after completion.

To find the best probe position, an X–Y scan is performed with the near-field probe as close to a reference device as possible without touching the packaging of the microcontroller. The point above the device yielding highest spectral intensity is chosen for collections. Lateral movement of the circuit board is minimized between signal collections using a custom made jig that fixed microprocessor position relative to the probe. A DC power supply (Agilent E3631A) minimizes variation in the supply voltage.

Training data for each of the 40 devices is generated by commanding 5000 AES-128 encryption operations using randomly chosen plaintexts and keys. EM emissions are captured using a Riscure probe [19] and a Lecroy 104-Xi-A digital oscilloscope. Similarly, test traces for each device are collected from 500 encryption operations using a fixed key and random plaintexts.

Traces are time aligned by shifting them based on the location of highest cross-correlation of a trace segment with a segment from the reference trace [19]. Traces are collected at a sampling rate of 2.5 GSa/s with a 1 GHz low-pass anti-aliasing filter inserted between the probe and the oscilloscope. The target device operates at 29.48 MHz so traces were decimated to 250 MSa/s. This was done by first filtering the traces with an eighth-order low-pass Chebyshev Type I filter having a cut-off frequency of 100 MHz and then properly decimating the filtered trace by 10 (every 10th sample retained and all others discarded).

4.2 Template attack methodology

This section explains the methodology used to implement template attacks for the target microcontroller. The initial step develops two highly effective same-device template attacks using both a heuristic method and PCA to select distinguishing features. If an attacker only has access to one training device, it is assumed that he would follow a similar process. For cross-device attacks, each device is used as a training device and used to attack all 40 devices. In all attacks, feature selection is performed using a single training device.

Class selection In the classification stage of the template attack, the classifier attempts to determine which class an observed side-channel emission most likely belongs to. The training traces are separated into $K = 256$ classes and templates are created for each class.

Correlation-based feature selection. Distinguishing features are identified using correlation analysis. Since both the plaintext, $\mathbf{t} = \left(t_1, t_2, \ldots t_{n_t}\right)^T$, and sub-keys, $\mathbf{k} = \left(k_1, k_2, \ldots k_{n_t}\right)^T$, are known for each of the n_t collected traces in the training data, the correct intermediate value $v_i = f_{(t_i,k_i)}$ can be calculated as well. The leakage model h_i based on the Hamming Weight of v_i is also easily calculated. A vector of the modeled leakage across all traces is $\mathbf{h} = \left(h_1, h_2, \ldots, h_{n_t}\right)^T$. For side-channel traces containing n_s samples, the collection of side-channel n_t traces is

$$S = \begin{pmatrix} s_{1,1} & \cdots & s_{1,n_s} \\ \vdots & \ddots & \vdots \\ s_{n_t,1} & \cdots & s_{n_t,n_s} \end{pmatrix} = \left(\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_{n_s}\right),$$

where $\mathbf{s_j}$ is a column vector containing the jth sample from each of the n_t traces.

The modeled EM leakage $\mathbf{h}$ is compared with observed leakage using the normalized correlation coefficient given by [20]

$$r_j = \frac{\sum_{i=1}^{n_t}\left(h_i - \bar{h}\right)\cdot\left(s_{i,j} - \bar{s}_j\right)}{\sqrt{\sum_{d=1}^{n_t}\left(h_i - \bar{h}\right)^2 \cdot \sum_{d=1}^{n_t}\left(s_{i,j} - \bar{s}_j\right)^2}}. \tag{13}$$

The correlation coefficient r_j is an indication of the linear relationship between the observed side-channel and modeled

leakage. High r_j indicates observed time samples are highly correlated with the modeled leakage.

Selecting distinguishing features based on correlation analysis with a Hamming weight model for the targeted intermediate value byte produces adequate results. However, classification is improved by performing correlation analysis separately for each bit of the intermediate value byte. This approach produces a vector of correlation coefficients for each of the 8 bits in the targeted byte. Samples with the highest correlation coefficient for each bit are added to the list of distinguishing features for the byte. If a sample has already been added because it was highly correlated with another bit, the sample with the next highest correlation coefficient for that bit is added. Only samples which have correlation coefficients significantly greater (5 dB) than the average correlation coefficient are added to the list of distinguishing features. It was determined empirically through experimentation that including up to ten points of interest for each bit produced very good results.

PCA-based feature selection. PCA is also used to generate and select distinguishing features in the principal subspace. Based on the samples selected using correlation analysis, only the first $N_{\text{PCA}} = 3500$ samples are used for each trace. For byte-wise analysis with $K = 256$ classes same-device PCA-based template attacks are not always successful unless approximately 80 components in the principal subspace are used as distinguishing features. This may be due to the relatively low number of traces ($\sim$19 on average) from each class used to construct the mean traces. Like the correlation-based feature selection process, the probability of correctly identifying the key-byte improves by performing bit-wise analysis.

PCA is performed for each bit of the target byte, with $K = 2$ classes for each bit. A PCA transformation matrix $\mathbf{W}_i \in R^{N_{\text{PCA}} \times 1}$ is constructed for each bit $i \in \{1, \ldots, 8\}$ retaining a single principal component. Rather than perform eight bit-wise template attacks, these eight transformation matrices are combined column-wise,

$$\mathbf{W} = [\mathbf{W}_1 \cdots \mathbf{W}_8] \tag{14}$$

where $\mathbf{W} \in R^{N_{\text{PCA}} \times 8}$. The new transformation matrix is used to perform byte-wise template attacks using (7) and (8).

4.3 Distinguishing feature data normalization

Motivation for the MVN technique can be seen in Fig. 1, which provided violin plots [21] for the distributions of the 5000 observations of sample #972 for each device. Sample #972 is one of the 16 samples chosen as a distinguishing feature for all 40 devices. The violin plots show that observation mean and variance changes from device to device.

For each sample selected as a distinguishing feature, the distributions of training and test data at that sample are normalized using (11) and (12), respectively. Normalization is performed independently on training and test data because the mean and variance at corresponding samples may not be the same for different devices. For simplicity when utilizing PCA the distribution for training and test, data are normalized for each sample across all traces before PCA transformation into the primary component subspace. The test data set must contain enough traces to estimate the distribution of each sample accurately.

5 Results

5.1 Selected features

The correlation-based feature selection methodology in Sect. 4.2 is repeated for each of the 40 devices. Each byte of the input to the SubBytes operation in round 1 of AES-128 is targeted separately with 256 templates constructed for each byte. The process is repeated to identify distinguishing features to build templates for each training device. Figure 2 is a graphical representation of the samples selected for each device for byte 1. Since only these samples are used as distinguishing features, and there are large gaps between them, the time axis is segmented multiple times to compress the plot.

Recall that part A devices have different peripherals than devices in groups B–D. There is some intra-device type variation in the samples for parts B–D devices but 19 of the samples are the same across the three part types. It is important to note that a number of samples which are consistently selected for devices in group A are not identified as distinguishing features for any of the devices in group B, C, and D. Likewise, some samples commonly selected in groups B, C, and D are not selected for group A.

When performing a PCA-based template attack, the transform matrix $\mathbf{W}$ that maps the trace samples into the principal subspace is generated separately for each set of training data. Plotting the eigenvectors is one way of visualizing contributions of the original samples in the principal subspace [6]. Since the magnitude of the eigenvector elements determine the weight of a sample's contribution in a component, a plot of samples which contribute most to one or more of the retained components can be generated by averaging the magnitude of the eigenvector elements for the retained components. Figure 3 shows the maximum value of the eigenvector element magnitude for each of the eight retained components found using bit-wise PCA. Since a majority of the samples contribute to one or more of the retained primary components the plot is not segmented as in Fig. 2. Note that different samples are weighted more heavily based on the training device. As is the case for the correlation-based feature selection, devices in groups B, C, and D are more similar to each other than they are to group A.

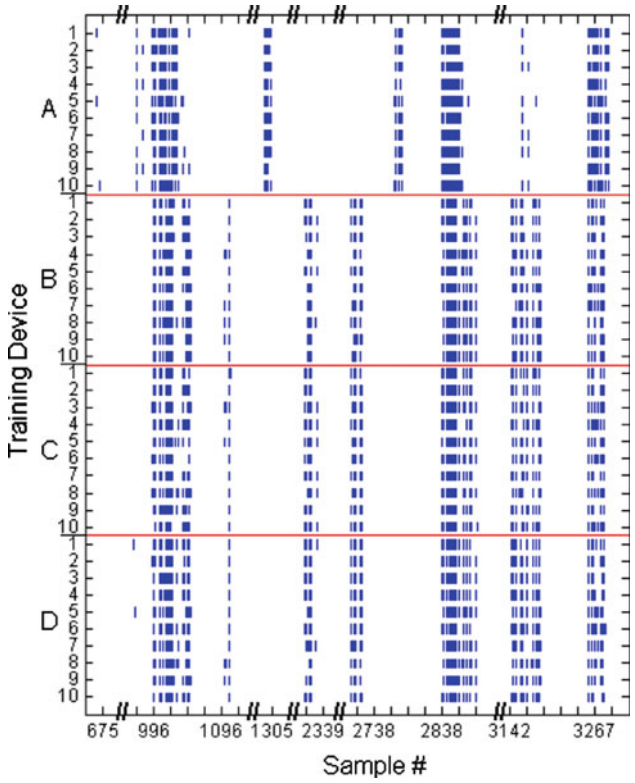

Fig. 2 Samples selected as distinguishing features for each of the 40 devices using the correlation-based feature selection process when attacking byte 1 of the input to the SubBytes operation. All *B*, *C* and *D* devices share 19 common features while only 6 of those features are identified for all of the type *A* parts

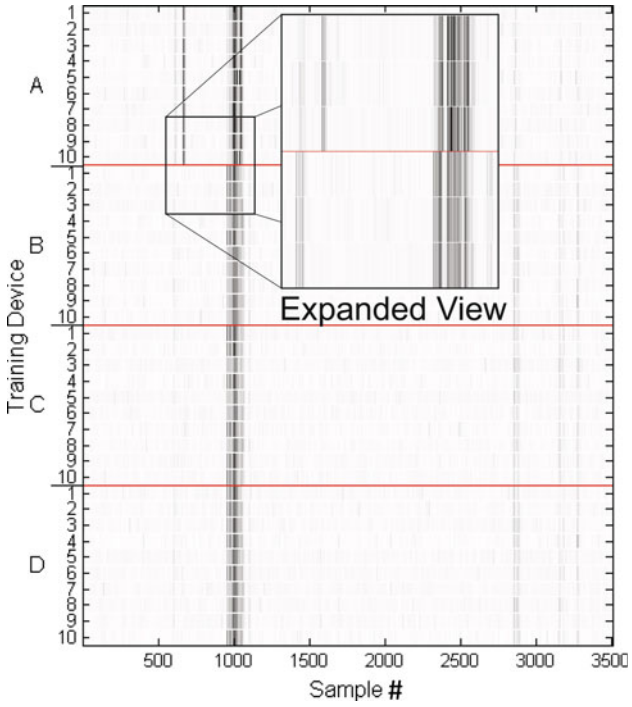

Fig. 3 Plot of the maximum magnitude of the eigenvector elements for the eight retained components found using bit-wise PCA. *Darker points* have higher maximum eigenvector element magnitude, indicating they contribute more to one or more of the retained components in the principal subspace

5.2 Baseline standard template attack

A standard template attack assumes a multivariate Gaussian distribution for each sample and assumes the distributions for corresponding training data samples and test data samples are identical [1]. Figure 4 shows the percent of key-bytes correctly extracted for 1600 template attacks, one for each device used as training device and as a test device. Each attack is repeated using 100 randomly chosen sets of 30 test traces from the 500 available test traces for each test device. The same 100 trace sets for each test device are used in the attack performed in Sects. 5.3 and 5.4. When the same-device is used to generate both the training and test data, the template attacks identify each of the 16 key-bytes correctly in all trials. Same-device attacks are found on the diagonal of the chart. The overall percent of successful key-byte extractions using one device from groups A–D to attack another device from groups A–D is shown in Table 2. The cross-device success rates do not include same-device attacks.

The reduced number of correct key-bytes when training using devices from group A to attack devices from groups B, C, or D can be explained in part by the difference in the distinguishing features used to build templates. More surprising

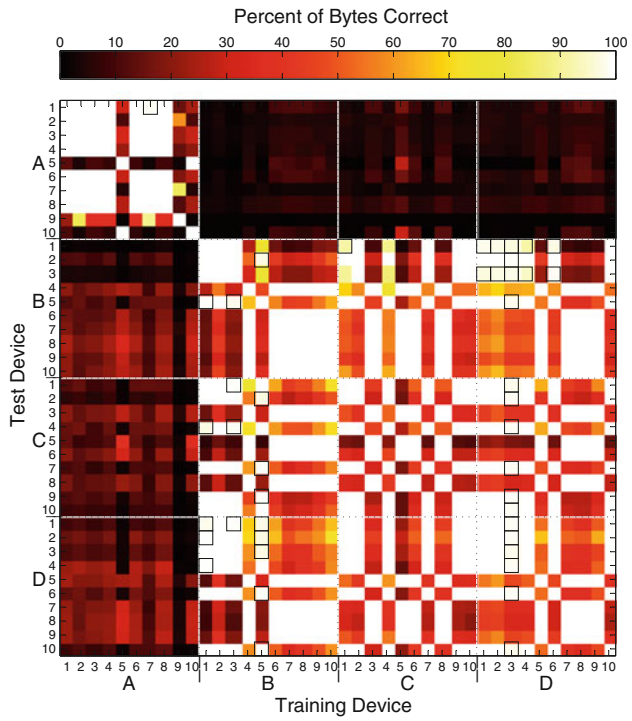

Fig. 4 Standard attack results using same- and cross-device templates without the MVN technique. The percentage of correctly extracted key-bytes in 100 trials is indicated by the *color* of the block. Percentages ≥90 and <100 % are *highlighted* with a box

Table 2 Standard template attack cross-device key-byte extraction success rate using correlation-based selection of distinguishing features (without the MVN technique)

Test device	Training device			
	A (%)	B (%)	C (%)	D (%)
A	59.7	4.3	5.2	5.1
B	13.1	63.0	68.3	70.2
C	11.6	67.9	63.7	69.5
D	14.7	69.8	69.2	70.3

is the poor results for intra-group attacks which construct templates using many of the same samples as distinguishing features. The poor results are due to location and spread differences in the distributions for collected side-channel emissions at the points used as distinguishing features for training and test data. For example, Fig. 1 shows the distribution of sample #972 for 5,000 observations in each set of training traces. Since the test data collected for each device is consistent with the distribution of the training data from that device, it is not shown separately.

5.3 MVN technique results

The template attack results can be improved by transforming the test data to match the distribution of the training data or by mapping both the training and test data to the standard normal. This transformation is performed separately for the data from each distinguishing feature before templates are built. These template attacks are performed with 5,000 training traces and 30 test traces. Unlike the attacks in [13] which normalized the data using 50,000 measurements, only the 30 test traces in each trial are used to estimate the distribution of the test data. Since the main benefit of template attacks is the low number of test traces required, limiting the number of traces used for normalization is more realistic. Using this simple pre-processing step, the successful byte extraction rate is improved for cross-device attacks for both the same part number and similar devices.

The correlation-based template attack is repeated after pre-processing the training data and each of the 100 test trace sets for each device. The results are shown in Fig. 5 and Table 3. With the MVN technique, any device from groups B, C, or D can be used to successfully attack any test device in groups B, C, or D. Any device in group A can be used to attack any device within that group. The worst same-part-number performance was using A9 to attack A5 where the correct key-byte was returned in only 94.4 % of the attacks. Device D3 has the "poorest" results as a training device when attacking similar devices with only 98.7 % of bytes-correct for devices in groups B–D; however, in practice such an attack would likely be successful. The MVN technique also dramatically improves the percentage of bytes correctly identified when attacking between groups B–D and group A.

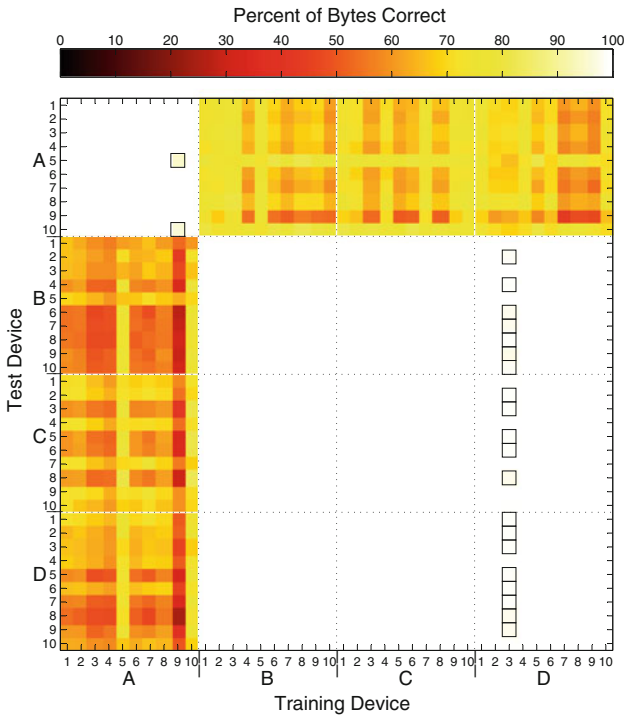

Fig. 5 Results from same- and cross-device template attacks using MVN technique with correlation-based distinguishing features. The percentage of correctly extracted key-bytes is indicated by the *color* of the block. Percentages ≥90 and <100 % are *highlighted* with a box

Table 3 MVN-enhanced cross-device key-byte extraction success rate for matched distribution correlation-based template attack

Test device	Training device			
	A (%)	B (%)	C (%)	D (%)
A	99.9	70.0	70.3	67.8
B	58.8	100.0	100.0	99.9
C	64.8	100.0	100.0	99.9
D	61.5	100.0	100.0	99.9

A same-device attack can be performed successfully for all 40 devices using 1–4 training traces. For same-device attacks, normalizing the training and test data reduces the posteriori probability of the correct key-byte guess found during the classification step for an equivalent number of traces, but the correct key-byte is still chosen based on the ML decision rule. The additional traces required for cross-device attacks allow the distribution of the test data to be estimated accurately. It is important to note that plaintexts only need be known for traces used in the classification process and not for all traces used to estimate the distribution. In this case, 30 traces were used for both distribution estimation and classification.

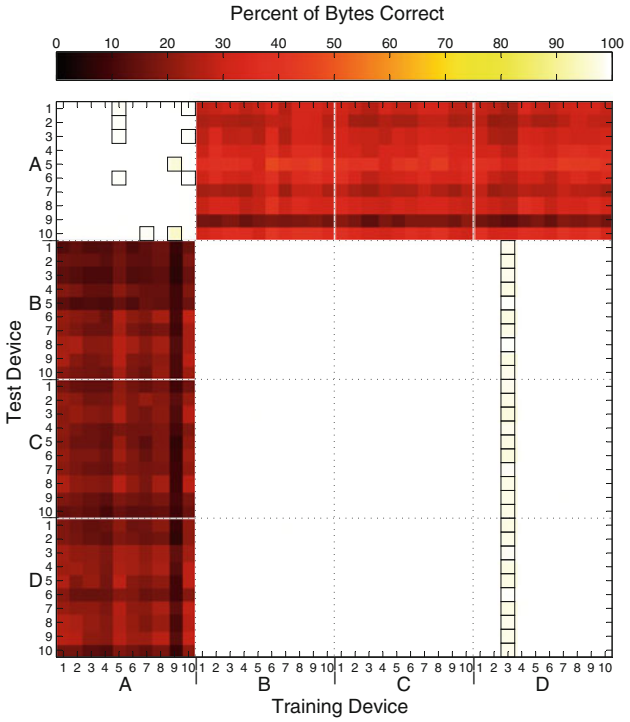

Fig. 6 Results from same+ and cross-device template attacks using MVN technique with bit-wise PCA performed on the distinguishing points vectors found using correlation analysis. The percentage of correctly extracted key-bytes is indicated by the *color* of the block. Percentages ≥90 and <100 % are *highlighted* with a box

5.4 PCA-based attack

The PCA-based attack incorporates the MVN processing step. Normalizing the mean and variance is a common PCA pre-processing step when data are collected using various scales for different dimensions. It is not clear if this step is performed in [2,6], as testing showed it is not necessary for same-device template attacks.

The PCA-based attack uses eight distinguishing features generated by transforming the training data and test data using **W** found using (14) and performing a byte-level template attack. The PCA-based template attack is repeated 100 times for each training and test pair and the success rate is shown in Fig. 6. Cross-device byte extraction success rate is shown in Table 4. Although all same-device attacks are successful, a small number of the attacks within a group only correctly recover 15 of 16 bytes. The number of bytes correctly extracted when training using devices in group A to attack devices in group B–D is significantly lower than for attacks using the same test traces with the correlation-based distinguishing feature selection process. The PCA-based attack can be improved by increasing the number of principal components retained for each byte (or bit) or by performing PCA only on the points identified using the correlation-based selection process.

Table 4 Cross-device key-byte extraction success rate for MVN PCA-based template attack

Test device	Training device			
	A (%)	B (%)	C (%)	D (%)
A	99.6	30.4	29.9	30.0
B	16.7	100.0	100.0	99.7
C	17.3	100.0	100.0	99.7
D	19.5	100.0	100.0	99.7

These numbers do not include the same-device attacks

5.5 Comparison of attacks

This section examines how the MVN technique affects the probability of successful key-byte extraction for a single training and test device pair. The correlation-based and PCA-based template attacks are performed using A1 as the training device and A5 as the test device. This pair is chosen because the MVN technique improves the results for both the correlation and PCA-based attacks for 30 traces. The attacks in Figs. 4, 5 and 6 are performed using sets of 30 traces from the 500 collected test traces for each device. The number of traces required to perform each type of attack is evaluated below.

When using a Bayesian classifier, the order in which traces are processed theoretically does not matter. In practice, traces may be ignored if they cause the denominator of (9) to be zero. This occurs frequently for cross-device attacks without the MVN technique when the test data samples have different distributions than the training data.

To randomize the order traces are added to the template attacks, 500 permutations of the 500 test trace indices are generated to specify trace order. The percent of bytes correct in Figs. 7 and 8 are the percentage of bytes correct across all 16 key-bytes using the template attack for the 500 randomly generated trace orders. The same 500 trace orders are used for each template attack methodology.

Same-device template attacks are very effective using only a small number of traces. All results in Fig. 7 are performed using only the indicated number of test traces for both normalization and classification. As expected, for a low number of traces, i.e., less than 15, where the distribution of each sample cannot be accurately estimated, using MVN for same-device attacks dramatically reduces the effectiveness of the attack. However, for a same-device attack the standard template attack methodology can be used.

For A1–A5 cross-device correlation-based attacks, a relatively poor successful byte extract rate of approximately 28 % is achieved after 15 traces without the MVN technique. The PCA-based template attack gradually improves to 51.3 % for 30 traces. The A1-A5 cross-device attacks, however, are greatly enhanced by the MVN technique with 99.6 %

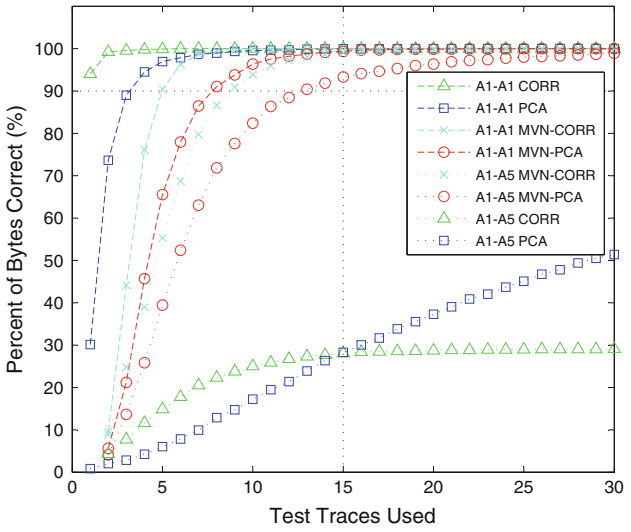

Fig. 7 Comparison of same-device (A1–A1) and cross-device (A1–A5) template attacks using the baseline standard template attack and attacks using the MVN technique. CORR and PCA indicates whether correlation analysis or PCA was used to identify/generate distinguishing features

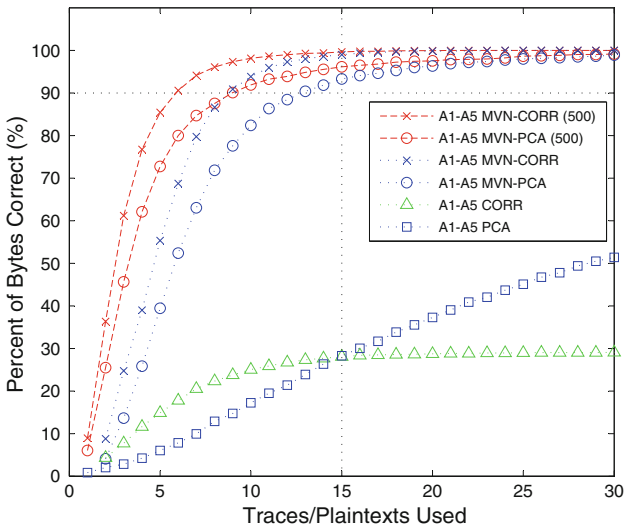

Fig. 8 Using additional traces to estimate the distributions improves the results for MVN technique-based attacks. When 500 traces are used for the MNV process (denoted 500), with the indicated number of plaintexts, the results for both the correlation-based (CORR) and PCA-based MVN attacks improve

successful byte extraction by 15 traces for the correlation-based attack and 99.1 % successful byte extraction by 30 traces for the PCA-based attack.

Figure 8 shows the benefit of using more traces to estimate the distributions before normalization. By performing the MVN technique on all 500 traces before using the traces for classification, fewer traces are needed to achieve the same percentage of correct bytes for both correlation- and PCA-based cross-device template attacks. Using 500 traces for the MVN processing technique, the correlation-based attack reaches 90 % successful extraction in six traces. It takes nine traces to reach the same byte extraction rate when only traces with plaintexts are used to estimate the distribution. Using 500 traces for MVN, the PCA-based attack reaches 90 % at nine traces. When only using the traces with plaintexts, 13 traces are required before the 90 % extraction rate is reached.

6 Conclusion

This research explores whether *similar* devices can be used as effective training devices for a template attack. It was shown that while template attacks based on mean and covariance matrices work well for attacking the same device on which the training traces are collected, the slight differences in emissions from similar devices may be sufficient to cause a template attack to fail. However, if the zero mean, unit variance normalization (MVN) technique is used to pre-process both the test and training data before building templates, the effectiveness of cross-device template attacks is significantly improved. These results are consistent with the benefit of the MVN technique utilized in [13] for differences in measurement conditions and device age for training and test data.

Additionally, the distinguishing features selected may be different from device to device, even for devices with the same part number produced in the same lot. If enough distinguishing features are different between two devices, the template attack will fail. To improve the effectiveness of a template attack the attack should try to get an identical device. Even small changes such as internal peripherals are sufficient to degrade the byte extraction success rate. While the goal for a same-device attack is to reduce the number of distinguishing features to make the templates easier to create, increasing the number of distinguishing features improves the cross-device attack success rate.

Ultimately, the original assumption that training and target devices have sufficiently similar side-channel emissions in [1] is validated with an added caveat that device-dependent differences in sample means and variances must be compensated for before performing the template attack.

One limitation of the MVN technique is that sufficient traces from the target device must be available to estimate sample distributions accurately. This is a drawback of the MVN technique, but nevertheless it allows for successful attacks that would fail otherwise. Furthermore, even if a relatively large number of traces are required to estimate the distribution, only a small number of plaintext or ciphertext must be known for the classification process.

References

1. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S. Jr., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2002. 4th International Workshop, Redwood Shores, CA, USA, August 13–15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer, Berlin (2003)
2. Archambeau, C., Peeters, E., Standaert, F., Quisquater, J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2006. Lecture Notes in Computer Science, vol. 4249, pp. 1–14. Springer, Berlin (2006)
3. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2006. Lecture Notes in Computer Science, vol. 4249, pp. 15–29, Springer, Berlin (2006)
4. Renauld, M., Standaert, F.-X.: Algebraic side-channel attacks. Cryptology ePrint Archive, Report 2009/279. http://eprint.iacr.org/2009/279
5. Oswald, E., Mangard, S.: Template attacks on masking-resistance is futile. In: Abe, M. (ed.) Topics in Cryptology—The Cryptographers Track at the RSA Conference 2007-CT-RSA. Lecture Notes in Computer Science, vol. 4377, pp. 243–256. Springer, Berlin (2007)
6. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2008. Lecture Notes in Computer Science, vol. 5154, pp 411–425. Springer, Berlin (2008)
7. Agrawal, D., Rao, J., Rohatgi, P.: Templates as master keys. In: Rao, J., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2005. Lecture Notes in Computer Science, vol. 3659, pp. 15–29. Springer, Heidelberg (2005)
8. Cobb, W., Garcia, E., Temple, M., Baldwin, R., Kim, Y.: Physical layer identification of embedded devices using RF-DNA fingerprinting. In: Proceedings of the 2010 Military, Communication Conference, MILCOM2010 (2010)
9. Maes, R., Tuyls, P.: Process Variations for Security: PUFs. In: Verbauwhede, I. (ed.) Secure Integrated Circuits and Systems. Springer, New York (2010)
10. Cobb, W., Laspe, E., Baldwin, R., Temple, M., Kim, Y.: Intrinsic physical layer authentication of integrated circuits. IEEE Trans. Inf. Forensics Sec. **7**, 14–24 (2012)
11. Agrawal, D., Rao, J.R., Rohatgi, P.: Multi-channel attacks. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2003. Lecture Notes in Computer Science, vol. 2779, pp. 2–16. Springer, Heidelberg (2003)
12. Rechberger, C., Oswald, E.: Practical template attack. In: Lim, C., Yung, M. (eds.) Information Security Applications. Lecture Notes in Computer Science, vol. 3325, pp. 440–456. Springer, Berlin (2005)
13. Elaabid, M., Guilley, S.: Portability of templates. J. Cryptogr. Eng. **2**(1), 63–74 (2012)
14. National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES), FIPS publication 197. http://csrc.nist.gov/encryption/aes/index.html (2001)
15. Kocher, P., Jaffe, J., Jun, B., Rohatgi, R.: Introduction to differential power analysis. In: Koç, Ç.K., Paar, C. (eds.) J. Cryptogr. Eng. **1**, 5–27 (2011)
16. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM side-channel(s). In: Kaliski, B., Koç, Ç., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2002. Lecture Notes in Computer Science, vol. 2523 pp. 29–45. Springer, Berlin (2002)
17. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, New York (2007)
18. Microchip: PIC24F Family Reference Manual (2010). http://www.microchip.com/stellent/idcplg?IdcService=SSGETPAGE&nodeId=%2575
19. Riscure: Inspector Data Sheet. http://www.riscure.com/benzine/documents/EMProbeStation.pdf
20. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2004. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
21. Hintze, J.L., Nelson, R.D.: Violin plots: a box plot-density trace synergism. Am. Stat. **52**(2), 181–184 (1998)