Channel Equalization for Side Channel Attacks

Colin O'Flynn and Zhizhang (David) Chen

Dalhousie University, Halifax, Canada
{coflynn, z.chen}@dal.ca

Revised: July 10, 2014

Abstract. This paper introduces the use of channel equalization as a method of reducing the computational complexity of side channel analysis (SCA), by effectively collapsing all points in a power measurement trace into a single random variable. This uses a simple Finite Impulse Response (FIR) linear equalizer, which has been studied extensively in communications systems. In addition the estimation of a channel model is used in developing the Channel Estimation Analysis (CEA), which is a generic attack requiring similar assumptions to the Correlation Power Analysis (CPA) attack. Both channel equalization and the CEA attack are straight-forward to apply to real systems, and Python examples are provided. Results of attacking unprotected AES-128 and protected AES-256RSM on a microcontroller are provided, and compared to a standard CPA attack along with a template attack.

Keywords: side-channel analysis, multivariate, equalization, channel estimation

1 Introduction

This work introduces the use of channel equalization to effectively collapse an *entire measured power trace* to a single point. This work is motivated in part by previous reports of the physical measurement channel combining measurements from many points into one, effectively breaking systems with univariate attacks, despite the algorithms being theoretically secure against such attacks[1]. This was mostly empirical observations, and does not for example demonstrate how to perform this optimally, or how to evaluate countermeasures under such assumptions.

After covering the related work, we will begin with a description of the model of a side channel analysis (SCA) attack as a communications system in Section 2. The use of Channel Estimation in communications systems will be covered in Section 3, which then applies this knowledge to the problem of side channel analysis. In Section 4 a proposal will be made about the use of channel equalization itself as a non-profiled attack. Section 5 will briefly cover details of improving the implementation performance, before finally bringing forth results of channel equalization in Section 6.

The attacks will be demonstrated on physical devices implementing unprotected software AES-128, along with protected software AES-256 using RSM (Rotating SBox Mask)[2]. It will be demonstrated that the proposed channel estimation is capable of breaking the AES-256 RSM *without* any knowledge of the implementation beyond the standard Hamming Weight assumption — i.e. without considering that the implementation is using masking.

Finally future work and conclusions follows in Section 7 and 8.

1.1 Related Work

The Stochastic Model [3] assumes that a physical observation t is composed of a data-dependant part related to the subkey s and data x, along with a random noise part. This can be seen as analogous to the channel model which will be introduced in Section 3.1. The stochastic method is still selecting points within the trace however, whereas the work in this paper is using the points to map back to an indicator of the original data.

The proposed channel estimator is a simple linear filter, implemented as a Finite Impulse Response (FIR) structure. Previous work in finding an optimal linear filter for a CPA attack is given in [4]. The authors use a FIR filter as a preprocessing step for a CPA attack, where the FIR filter is designed to maximize the ratio between the output of the CPA equation (discussed in Section 2) for correct and incorrect inputs.

Similarly in [5] the FIR structure is used, although instead of optimizing a metric based on the correlation equation as in [4], an attempt is made to directly optimize the FIR coefficients to improve the SNR. An advantage the method proposed in [5] is it doesn't require a profiling stage where a known key is being used. The work in [5] cannot however compensate for certain countermeasures as the proposal here does.

The work presented here instead attempts to use the same FIR structure to improve the fit of the leakage to a given hypothetical model. This will allow the FIR structure to determine all related points, including determining if it is possible to use combinations of points to break masking countermeasures. This work also has advantages when implementing analysis algorithms in real-time. The use of SCA for verifying device integrity has been presented in [6] for example. It is possible to implement the FIR structure in real time, for example if it is desired for an embedded system to use SCA as part of a verification system. Since the output requires comparison to only a single template, the computational complexity is minimized.

2 Correlation Power Analysis and the Matched Filter

As a precursor to the introduction of channel estimation, the well-known Correlation Power Analysis (CPA) [7] attack will be considered in lens of communications theory. The basic equation for a CPA attack, where $r_{i,j}$ is the correlation coefficient at point j for hypothesis i, the actual power measurement is $\boldsymbol{t_{d,j}}$ of

trace number d at point j, and $p_{d,i}$ is the hypothetical power consumption of hypothesis i for trace number d, with a total of D traces is given in equation (1). This equation is simply an application of the Pearson's correlation coefficient given in equation (2), where $X = \boldsymbol{p}$, and $Y = \boldsymbol{t}$.

$$r_{i,j} = \frac{\sum_{d=1}^{D} \left[\left(p_{d,i} - \overline{p_i} \right) \left(t_{d,j} - \overline{t_j} \right) \right]}{\sqrt{\sum_{d=1}^{D} \left(p_{d,i} - \overline{p_i} \right)^2 \sum_{d=1}^{D} \left(t_{d,j} - \overline{t_j} \right)^2}} \tag{1}$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[\left(X - \mu_X \right) \left(Y - \mu_Y \right) \right]}{\sqrt{E\left[\left(X - \mu_X \right)^2 \right]} \sqrt{E\left[\left(Y - \mu_Y \right)^2 \right]}} \tag{2}$$

The form given in these equations is referred to as the *normalized cross-correlation*, and frequently used in image processing applications for matching known templates to an image.

2.1 Basic Communications Principles

In communications theory, the most basic problem statement is how to receive a signal that has been corrupted by Additive White Gaussian Noise (AWGN). The continuous-time and discrete-time interpretations of this problem are given as follows:

$$r(t) = As(t) + n(t) \tag{3}$$

$$r[n] = As[n] + w[n] \tag{4}$$

The transmitted signal or sequence $s(t)$ or $s[n]$ is one of several valid signals, the specific signal depending on the system. The objective of the communication systems is for the receiver to determine which of the possible symbols $s_1(t), s_2(t) \cdots, s_N(t)$ was sent based on the received signal $r(t)$.

2.2 Correlation Implementation of the Matched Filter

The objective of receiving a known signal in Additive White Gaussian Noise (AWGN) has a well known solution, the matched filter (or 'North filter'), first described in 1943[8]. In the case of receiving the signal $s(t)$ given in (3), the impulse response $h(t)$ of the matched filter should be a time-reversed and shifted copy of the transmitted signal:

$$h(t) = s(T - t), 0 \leq t \leq T$$

Which will maximize the output of the filter at time $t = T$ when the transmitted signal is $s(t)$. Applying the filter to a received signal means convolving this impulse response with the received signal, and sampling the output of this

convolution at time $t = T$. If we are attempting to select which of the possible signals $s_1(t), \cdots, s_N(t)$ was sent, we would simply perform N convolutions, each for a candidate $s_n(t)$. Selecting the most likely candidate than becomes the arg max of the convolution over all candidates at $t = T$, which can also be written as the correlation of received signal $r(t)$ with all candidates $s_n(t)$ at $t = 0$, which would be:

$$\arg\max_n \left(r(t) \star s_n(t)|_{t=T}\right) = \arg\max_n \int_0^T y(\tau) s_n(\tau) d\tau \tag{5}$$

The forms given in equations (2) force both $r(t)$ and $s(t)$ to be zero-mean and normalized by standard deviation. This is necessary for us as we do not have proper scaling of the template $s(t)$ used at the receiver.

One critical difference between communications systems and side-channel power analysis is the definition of the argument of $s(t)$. In communications we are sending a known signal $s_n(t)$, which may be drawn from a set of 'allowed' signals $s_1(t), s_2(t), \cdots, s_N(t)$. Each of these signals is typically a finite-length signal as a function for time (or samples in the discrete case). At the receiver we can use the matched filter to determine which of the N possible signals was transmitted.

For side-channel analysis, our function $s(t)$ is actually defined over the number of cryptographic operations we observed. In equation (1) this was the 'trace index' d, and thus will be referred to as $s(d)$. Each of the possible functions $s_1(d), s_2(d), \cdots, s_N(d)$ reflects the hypothetical value for the byte of the secret key we are attacking. Thus the matched filter comparison is always done at the same sample (i.e. time point) in each power measurement trace $\boldsymbol{t_d}$.

3 Channel Estimation & Equalization

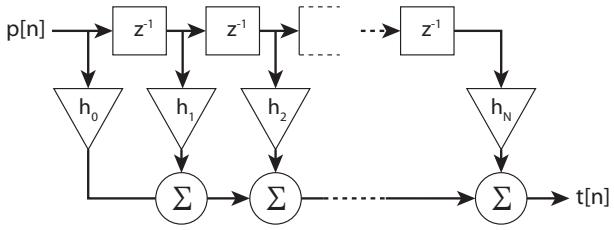

Fig. 1. Simple channel model, where noise can be added to the output if w[n] also needs to be modelled.

In equation (4) the received signal is corrupted only with AWGN, and possibly a fixed scaling factor. This does not account for realistic channels between the transmitter and receiver, which may include the signal coming from multiple

paths. Instead the model shown in Fig. 1 is used, which is described by equation (6). The objective of 'channel estimation' is to discover the value of the taps, given by the $\boldsymbol{h_{channel}}$ vector. Again p is the original transmitted data.

$$t[n] = \sum_{j=0}^{J-1} h_j p[n-j] + w[n] \tag{6}$$

Estimating the coefficients at the receiver requires that the transmitter sends a known 'training sequence'. In communications systems the channel estimation has several complicating factors: it must be performed in real-time to be useful, and the channel will change over time so one must track the channel. By comparison in side-channel analysis the computation must simply be possible in reasonable time, and the channel varies little over time since the measurement setup is fixed.

3.1 Applicability to Side Channel Analysis

Using a channel model for side-channel analysis means we assume that a single piece of data generated the entire power trace, via the channel model. If we use the inverse of this channel model at the receiver, we can thus generated a single point from each trace, this single point containing all of the relevant information from the entire trace for a specific subkey. Thus note that each subkey s requires a different channel estimate to be formed. Rather than forming the channel $\boldsymbol{h_{channel}}$, we will instead directly estimate the inverse. This inverse will be the required *linear equalizer* for our unknown channel. As we will generate a separate equalizer $\boldsymbol{h_s}$ for each subkey being attacked, but it is trivial to also use a $\boldsymbol{H}$ matrix instead by combining the $\boldsymbol{h_s}$ vectors, which would generate information for all attacked subkeys. This is analogous to Multiple Input Multiple Output (MIMO) systems, where the channel matrix is used to generate several independent communications channels. In our example the 'independent channels' means the different information about each subkey s. Extensions to generate a channel estimate for a single bit within a subkey are of course valid, and would just require additional channel estimates.

In communications systems a sequence at the transmitter is disrupted by the physical channel. We will be considering the 'known sequence' to instead be the leaked information, typically the hamming weight or hamming difference of sensitive data. The channel is considered everything in between the leaked information and the power measurement: thus we also group countermeasures into this channel and other details of the implementation.

The model used is given in equation (7). The information being leaked by the device is $p_{d,s}$ (e.g.: hamming weight or hamming distance) about subkey s related to trace d. For example using the classic HW CPA assumption, $p_{d,s} = HW(SubByte(b \oplus k))$, where b is the plaintext byte and k is a key value.

The equalization vector for subkey s is $\boldsymbol{h_s}$, and $\boldsymbol{t_d}$ is the vector of power measurements.

$$p_{d,s} = (\boldsymbol{t_d} - \mu_{t_d}) \cdot \boldsymbol{h_s} + \mu_{p,s} \tag{7}$$

Note in this form we make no assumption about $\boldsymbol{t_d}$ or p_d being zero-mean. If both are assumed to be zero-mean, this can simplify the notation by removing references to μ_{t_d} and μ_p. In which case (7) can simply be written as:

$$p_{d,s} = \boldsymbol{t_d} \cdot \boldsymbol{h_s} \tag{8}$$

The value of μ_{t_d} will not be known, and instead the estimate $\hat{\mu}_{t_d}$ is formed from the received data. The value of μ_p is known, which will simply be $\frac{1}{2}$ of the maximum hamming weight (e.g. $\mu_p = 4$ on our 8-bit microcontroller).

The vector $\boldsymbol{h_s}$ is the linear equalizer coefficients. Unlike in communications systems we have no control over the 'transmitter' and thus will always use the form in equation (8). Attempting to solve the form of equation (6) and then invert the matrix for use in equation (8) would be equivalent, however the form in (8) simplifies notation and computation in side channel analysis problems.

3.2 Equalizer Coefficients from Training Set

The equalizer coefficients will be built from leakage measurements on a device with a known secret key. Finding equalizer coefficients *without* a known key will be discussed in section 4. If the coefficients are required to be independent of the secret key, the training set should be generated using many different (known) keys, although the sensitivity to this is explored in Section 6.

For a traces with a known secret key, the expected leakage measurement is considered the known value of $p_{d,s}$. Each of the power measurements is $\boldsymbol{t_d}$, and then the error between the estimated value $\hat{p}_d$ and the 'known' p_d is:

$$e(d) = p_{d,s} - \hat{p}_{d,s} = p_{d,s} - \left(\hat{\boldsymbol{h_s}} \cdot \boldsymbol{t_d}\right) \tag{9}$$

For notational simplicity this uses the form of (8), if the zero-mean assumption is not made the form in (7) should instead be substituted. Two different options for minimizing this option are commonly used: the Least Square (LS) and the Mean Square Error (MSE). Results of experiments using both minimization options were almost identical, so only the LS will be described here.

For the LS cost function, the objective is to minimize the sum of square errors over all traces:

$$\sum_{d=0}^{D-1} e^2(d) \tag{10}$$

This can be accomplished with a least-squares (LS) error estimator (or 'solver'), with the solution $\hat{\boldsymbol{h}}_{\boldsymbol{s}}$. These solvers are frequently built into numeric packages

such as MATLAB, SciPy, etc. A faster method is to use the pseudoinverse to solve the LS problem, which has a known solution given by (11.12) in [9]:

$$\hat{\boldsymbol{h}}_{\boldsymbol{s}} = \boldsymbol{t}^{+} \cdot \boldsymbol{p}_{\boldsymbol{s}} \tag{11}$$

Where $\boldsymbol{t}^{+}$ is the pseudoinverse of $\boldsymbol{t}$ (also known as the Moore–Penrose pseudoinverse). Note that $\boldsymbol{t}^{+}$ needs to be calculated only once for any set of traces, and can then be reused for many steps in the algorithm. For generating $\hat{\boldsymbol{h}}_{\boldsymbol{s}}$ over $s = \{0, 1, \cdots, S-1\}$ the $\boldsymbol{t}^{+}$ is calculated once for example, instead of performing S least-square estimators.

To calculate the pseudoinverse, we can use a singular value decomposition (SVD)[9]. If we perform the SVD on a matrix $\boldsymbol{A}$, we will have:

$$\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^{*}$$

Then we define the pseudoinverse as:

$$\boldsymbol{A}^{+} = \boldsymbol{V} \cdot \boldsymbol{\Sigma}^{+} \cdot \boldsymbol{U}^{*}$$

As $\boldsymbol{\Sigma}$ is a diagonal matrix, the pseudoinverse $\boldsymbol{\Sigma}^{+}$ is found by taking the inverse of each non-zero entry on the diagonal. In the implementation of this algorithm, there is a limit below which entries are considered to be zero, and any diagonal elements below this limit in $\boldsymbol{\Sigma}$ are replaced with zero.

3.3 Applying the Equalizer Information

Once the equalizer coefficients $\hat{\boldsymbol{h}}_s$ are acquired for each subkey s, it can be used to convert power traces into a single point containing all information linearly related to the leakage.

This single point can be processed per existing attack algorithms such as the CPA attack given in (1), although without the subscript j. The use of the equalizer results in an output with expected valid values, which allows a simplified selection algorithm discussed next.

3.4 Template and Minimum Distance

The objective of the equalizer is to generate outputs p, which is the supposed 'input' to the channel in (6). Again for our 8-bit HW assumption, this would mean $p \in \{0, 1, \cdots, 8\}$. This work assumes these values p are given by a leakage function, such as the HW of a sensitive value. In reality, the output $\hat{p}$ of the equalizer $\hat{\boldsymbol{h}}$ will *not* match the values given by the leakage function, or $\hat{p} \notin \{0, 1, \cdots, 8\}$.

Instead templates can be generated based on the output of the estimated channel for the training set. This output of the estimated channel is $\hat{p}_{s,i}$ for a given subkey s with leakage i. This simply means finding the value of $\mu_{\hat{p}_{s,i}}$ and $\sigma^2_{\hat{p}_{s,i}}$, and then matching the templates using a maximum likelihood hypothesis test.

As an alternative, the minimum distance approach is also considered. Here we consider the most likely template to be the one which minimizes the absolute difference between the received value and the template mean, or:

$$\hat{i_s} = \arg\min_i \sum_{d=0}^{D-1} |\hat{p}_{d,s} - \mu_{\hat{p}_{d,s,i}}| \tag{12}$$

In the case where the values of $\mu_{\hat{p}_{s,i}}$ are equally spaced and $\sigma^2_{\hat{p}_{s,i}}$ are all equal these two approaches (minimum distance and maximum likelihood) will produce the same results.

4 Equalizer Without Training Set

For evaluation of a specific cryptographic device, the most accurate equalizer coefficients will be generated with profiling (i.e. a training set). As a consideration of the use of these methods in practical 'attack' scenarios, it may be required to form the equalizer coefficients without knowledge of the secret key. This means that an attacker does not have a device they can characterize, and instead the problem is similar to a Correlation Power Analysis (CPA). This type of attack will be referred to as a Channel Estimation Analysis (CEA).

For CEA the attacker records D power traces, each trace $\boldsymbol{t_d}$ containing a number of points. The attacker also knows the input text (or cipher text) $\boldsymbol{b}_d$ for each power trace. The attacker than partitions the traces & texts into two arbitrary sets: a fitting set $\boldsymbol{t_d^f}$ with D^f elements, and a test set $\boldsymbol{t_d^t}$ with D^t elements. The majority of traces will belong to the fitting set, with a smaller number in the test set. Similarly the known text is split into $\boldsymbol{b}_d^f$ and $\boldsymbol{b}_d^t$ for the fitting set and test set.

We now solve equation (7) where $\boldsymbol{t_d}$ is the fitting set $\boldsymbol{t_d^f}$. We *do not* have the known leakage information p_s in this case, and instead a *hypothetical* leakage vector corresponding to the fitting set $\widetilde{\boldsymbol{p}}_{\boldsymbol{s}}^{\boldsymbol{f}}$ will be generated. If there are $i = \{0, 1, \cdots, N-1\}$ possible hypothetical values for each subkey, there will be N hypothetical $\widetilde{\boldsymbol{p}}_{\boldsymbol{s,i}}^{\boldsymbol{f}}$. The generation of leakage information is the same as in the CPA case, where the predicted leakage value depends on having a power model, the guess i, and the known text $\boldsymbol{b}_d^t[7]$.

The result of the least-squares fitting will generate a hypothetical equalizer coefficients vector $\widetilde{\boldsymbol{h}}_{\boldsymbol{s,i}}$ for subkey s & hypothesis i. Note this fitting is computationally intensive, and accomplishing the attack in reasonable times will instead use a pseudoinverse discussed in section 5.2.

Finally we use the test set $\boldsymbol{t^t}$ of power traces, and again will generate hypothetical vector $\widetilde{\boldsymbol{p}}_{\boldsymbol{s,i}}^{\boldsymbol{t}}$ based on $\boldsymbol{b}_d^t$ and i. We will pass each test set trace $\boldsymbol{t_d^t}$ through the hypothetical equalizer, and compare the fit based on the test set hypothetical value. Conceptually, we are simply attempting to obtain the equalizer coefficients for each hypothetical key. The equalizer with the best fit is deemed to be the most likely key. It is required to partition the traces into a fitting set and a test

set to avoid being fooled by noise, which may have the smallest residuals from the least-squares with the original dataset.

Equation (13) shows the function $e(s,i)$ which should be minimized over i for every subkey s. The value of i which minimizes $e(s,i)$ is thus the most likely hypothetical value for subkey s.

$$e\left(s,i\right) = \sum_{d=0}^{D^t-1} \left(\left(\widetilde{\boldsymbol{h}}_{\boldsymbol{s,i}} \cdot \boldsymbol{t}_{\boldsymbol{d}}^{\boldsymbol{t}} \right) - \widetilde{p}_{s,d,i}^{t} \right)^2 \tag{13}$$

5 Implementation Performance

This section briefly mentions some practical considerations of implementing the algorithms from this paper. Of particular importance is the use of the pseudoinverse for the least-squares estimation of the channel.

5.1 Decoding with Known Equalizer Coefficients

If the equalizer coefficients $\hat{\boldsymbol{h}}$ are known, the application of minimum-distance decoding is a lightweight process. Each incoming trace is multiplied by the equalizer coefficients to produce the leaked information about each subkey. Trivially the sum in equation (12) can be converted to an update equation. Thus for each guess of each subkey only the value of the summation is stored. The memory requirements are such that implementation in an embedded system is simple. In addition it is also possible to implement the system in hardware (e.g. FPGA or ASIC) which can process the incoming trace measurement a point at a time, avoiding the need to store traces.

For general side-channel analysis, such improvements are of little value. If, however, once wishes to perform side channel analysis in real time as part of a validation system[6], the simplified processing requirements are of a great benefit.

In general, the use of equalization will greatly reduce computational requirements as the output of equation (7) is a single point for each trace, regardless of the length of each input trace. Applying equation (7) for a single subkey s, where the input trace has J points, over a total of D traces, would require $J \cdot D$ multiplications, and $(J-1) \cdot D$ additions. There is of course an additional cost to initially generate the equalizer.

5.2 Pseudoinverse for Solving for Equalizer Coefficients

As mentioned, solving the pseudoinverse greatly simplifies the least-squares problem. In particular, for the CEA attack given in equation (13), only a single $\boldsymbol{t}^+$ calculation is needed, which is reused for all key-guesses i across all subkeys s. For the CEA algorithm on byte-wise AES-128 for example, this means 1 pseudoinverse compared to 4096[1] least-squares estimation operations.

[1] 16×256

A variety of existing libraries for calculating the pseudoinverse exist which simplifies calculation of the equalizer coefficients, since equation (11) can almost directly be coded. Examples of packages implementing the pseudoinverse include MATLAB, NumPy, SciPy, LAPACK, and OpenCV.

6 Attack Results

6.1 Unprotected Software AES-128

An unprotected software AES implementation is used as the first example device. The code is the AVR-Crypto-Lib AES code in C[2], programmed into an AtMega328p microcontroller. The device runs at 7.3728 MHz, and power measurements are taken from a 50-ohm resistive shunt inserted into the VCC lines. Measurements are perfectly synchronized with a trigger generated by the device.

Two separate attacks are considered: the first is a profiled attack, which first solves the equalizer coefficients equation (7) using power measurements taken with a known plaintext and encryption key, both of which randomly change for every trace. Once the equalizer is known a different set of traces with a fixed key and randomly changing plaintext is used to generate attack statistics.

Each trace measurement with the unknown key is multiplied by the estimated $\hat{\boldsymbol{h}}_s$ to form a single datapoint, which is then processed by several different algorithms. The results of this compared to both a standard CPA attack along with a template attack is given in Fig. 2. Details of the various attacks are given in the follow subsections.

Correlation Power Analysis (CPA) For this attack the standard CPA is used[7]. The most likely subkeys are ranked by the correlation coefficient given by equation (1), where the intermediate value attacked is the Hamming Weight at the output of the S-Boxes from the first round.

Template Attack For this attack a template[10] of the Hamming Weight (HW) at the output of the S-Boxes from the first round of AES is generated using 2500 traces with a known (random) encryption key & plaintext. A total of 16×9 templates are generated, each template targeting 3 Points of Interest (POI) in the trace for each subkey. The POI are selected based on the *sost* criteria as detailed in [11]. Template matching is done using a multivariate normal distribution, as given in [10].

Equalized Attacks For this attack the equalizer coefficients are generated from 2500 traces with a known (random) encryption key & plaintext. An attack is then performed on a different set of traces from the same setup, having passed those traces through a linear estimator $\hat{\boldsymbol{h}}_s$. The output of the linear estimator, a 1×1 random variable, is used in several different attacks.

[2] Available from: `http://avrcryptolib.das-labor.org/trac/wiki/AES`

The most basic attack uses that datapoint as the input to a CPA attack, which is exactly the same as the CPA attack used in Section 6.1. As in [4], this type of preprocessing should be more resilient to noise, since it's not dependant on specific templates (i.e. value of mean).

The CPA attack does not use all available information — we can use the existing training set to generate templates based on the output of the equalizer. Two versions as described in Section 3.4 are tested — one using a univariate normal distribution ('Equalizer Template') to generate probabilities against each candidate HW template, and one using a simple absolute difference between candidate HW template means.

The linear equalizer used here was found using the LS solver based on the pseudoinverse. Equalizers were also built with a regressive LS solver and the linear MSE solutions given in this paper – the resulting PGE was almost identical to that given in Fig. 2. To avoid cluttering the graph these have not been shown.

Channel Estimation Analysis The final attack graphed does not have a profiling phase, this is the Channel Estimation Analysis (CEA). The specified number of traces for each datapoint are split into half; one part becoming the fitting set, one part becoming the test set, as described in Section 4. The CEA appears to have poor performance, in part due to the partitioning of traces. The CEA achieves an average PGE < 10 in about 300 traces, although this is off-scale for the graph. The advantage of CEA will be demonstrated when instead attacking a masking scheme.

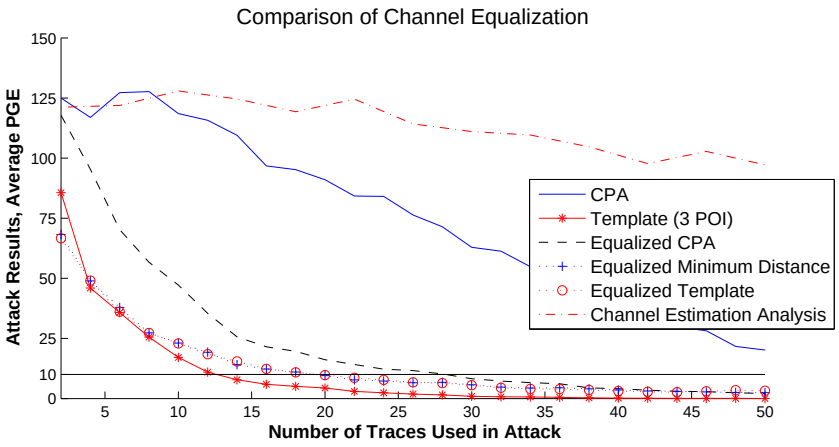

Fig. 2. Comparison of channel equalization attacks on AES-128.

6.2 Protected Software AES-256

A protected software AES implementation is used as another example device; the AES-256 code is protected by the Rotating SBox Mask [2]. This code is the same as used in the DPA Contest v4[3], but programmed into an AtMega328p microcontroller. The device runs at 7.3728 MHz, and power measurements are taken from a 50-ohm resistive shunt inserted into the VCC lines. Measurements are perfectly synchronized with a trigger generated by the device.

These examples will use both equalization with training set (i.e. profiling) and unprofiled. Note the profiling phase is *completely unaware* of any details of the implementation. The channel estimation procedure, including the power leakage model, is *exactly the same* as in the unprotected AES-128 case; these simple assumptions are the main advantage of linear equalization & CEA compared to other techniques. It will be appreciated that *considerably* better attack results on this specific implementation are published on the *DPAv4 Hall of Fame*[4], however they are normally aware of some details of the implementation.

CPA with Equalization Using Least-Squares Estimator For this attack the equalizer coefficients are generated from 5000 traces with a known key & plaintext. The coefficients are found using a least-squares estimator using the pseudoinverse. An additional 10 000 traces are recorded and used for generation of the statistics. The results are ranked by the output of the correlation coefficient given by equation (1). The resulting PGE is shown in Fig. 3. Note the most useful results are generated from a randomly changing training key, since the equalizer will be agnostic of the secret key. For test purposes the equalizer generated with a fixed key performs similarly, but this equalizer will be very sensitive to changes in the key.

It is important to note that an equalizer generated with a single fixed secret key does still provide some useful information about the channel. Consider that an attacker is unable to change keys but *does* know the value of a key. This could be the case where there is a default key on a device; the key is not used for any trusted operations, but is processed with the same operations as a trusted key. Using this key for training the channel equalizer, and then attacking the unknown secret key, provides improved results compared to a standard CPA attack.

Channel Estimation Analysis and Comparison to First-Order Attacks Two first-order attacks are mounted, which would be expected to fail on the AES-256RSM implementation. The first is the standard CPA using the Hamming Weight (HW) assumption, where the output of the S-Box is targeted. The most likely subkeys are ranked by the correlation coefficient given by equation (1), where it is calculated for each datapoint. Next, a template attack is used. The template attack again targets the first-round S-Box output, where templates

[3] `http://www.dpacontest.org/v4/`

[4] See `http://www.dpacontest.org/v4/hall_of_fame.php`

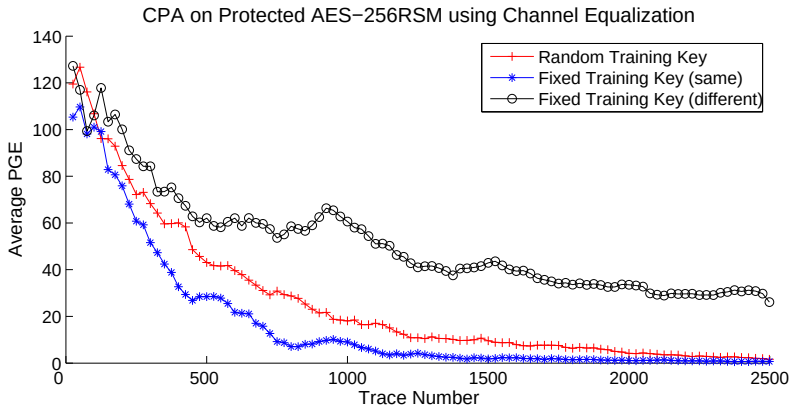

Fig. 3. CPA performed on AES-256 protected with RSM, where traces have been pre-processed by Channel Equalization. Three different training key setups are compared: a randomly varying key, a fixed key that is the same as the secret key (i.e. as someone testing the system), and a fixed key that differs from the secret key (i.e. as an attacker that cannot control the key).

are generate for the Hamming Weight (HW) of this value (i.e. the same leakage assumption being used for the channel estimation attack and CPA attack). These results are shown in Fig. 4, note after 10 000 traces no changes in the average PGE are found.

Note that it is possible for first-order attacks to succeed, as given in [12]. This required changes to the leakage model, and it was also noted that a standard HW leakage model of the S-Box input or output failed to recover the secret key.

Next, we will consider the CEA attack. For this attack no prior knowledge is assumed beyond the assumption about the device leaking the hamming weight, i.e., the same assumption made in the unprotected case. The specified number of traces are again split into two groups: a training set and a test set. The results of this attack are shown in Fig. 4. Whilst a complete attack was unsuccessful, the PGE of subkeys is considerably better compared to the other attacks, and a progression of the PGE towards zero is present.

7 Future Work

7.1 Classifying Countermeasure Effectiveness

The use of equalization can also be used to quantify the effectiveness of countermeasures. This is particularly useful for the simulated environment — it was previously reported for example how a physical measurement setup which 'blended' several measurements together resulting in an attack on a physical device being far more successful than the simulation predicted [1]. If instead the channel estimation and equalization process is used for the simulated environment, the equalized attack algorithm will combine all linearly related leakage points.

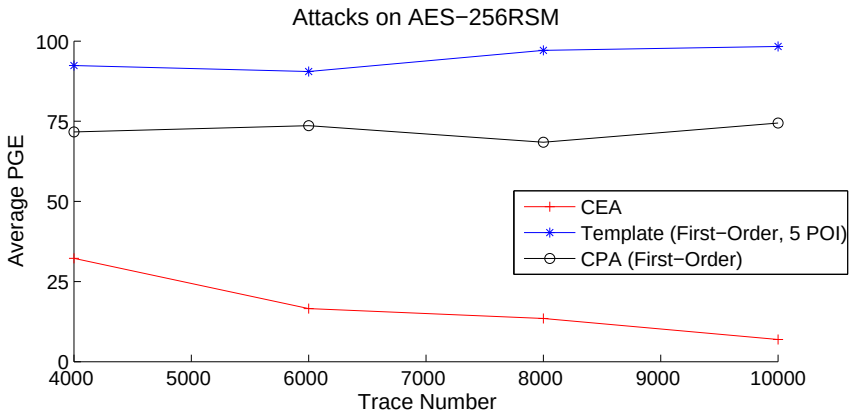

Fig. 4. Comparing the CEA to two first-order attacks.

Note that whist the channel equalizer is linear, the channel itself may be non-linear. Thus the real system may be able to combine non-linear effects to break the countermeasures, even though in the simulated channel the system is perfectly secure. Work in comparing simulated to physical channels is required to understand these effects. Using non-linear channel models may also improve the performance, but solving the LS or MSE cost functions for non-linear channels is more complex.

7.2 Multiple Leakage Combining

The issue of combining multiple leakages from the same device (i.e.: shunt measurement with electromagnetic probe) may also be solvable via the equalizer. In the original case we had a power trace measurement $\boldsymbol{t}_d$ corresponding to a plaintext input $\boldsymbol{b}_d$. Assume instead we record the power across a shunt resistor into vector $\boldsymbol{t}_d^{shunt}$, and the EM Field emitted into vector $\boldsymbol{t}_d^{em}$. Both of these measurements occurred for the same input $\boldsymbol{b}_d$. We now create a unified vector $\boldsymbol{t}_d$, where $\boldsymbol{t}_d = [\boldsymbol{t}_d^{shunt}, \boldsymbol{t}_d^{em}]$. If both vectors have some relation to the leaked value, the equalizer $\boldsymbol{h}$ will combine these into a single point.

8 Conclusions

Using channel equalization is a simple method of compensating for all disruptions to the leaked data of a device. With proper selection of the channel, even intended disruptions such as countermeasures can be compensated for. This work has used a simple linear FIR equalizer, where the equalizer is found using least squares (LS) or Mean Square Error (MSE) metrics. The improvement in attack performance for unprotected AES 128 is demonstrated, along with proving the ability of channel estimation to attack protected implementations. While channel equalization requires a profiling phase, some initial work using the channel

estimation without the profiling phase was also demonstrated, under the name of the Channel Estimation Attack (CEA).

The advantage of both the equalizer with profiling and the CEA is they require minimal assumptions about the device being attacked. In the case of CEA the attack requires no more information than a CPA attack, for example being unaware of any countermeasures inserted into the device. Compared to typical multivariate or higher-order DPA attacks, this is a considerable reduction in attack complexity. It was demonstrated how the channel equalization & CEA could both be used to break a specific protected AES implementation.

Complete code for implementation of all attacks, including measurements, is available at [link removed as would violate request for anonymous submissions].

References

1. Moradi, A., Mischke, O.: On the Simplicity of Converting Leakages from Multivariate to Univariate. In Bertoni, G., Coron, J.S., eds.: Cryptographic Hardware and Embedded Systems - CHES 2013. Volume 8086 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 1–20
2. Nassar, M., Souissi, Y., Guilley, S., Danger, J.L.: Rsm: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In Rosenstiel, W., Thiele, L., eds.: DATE, IEEE (2012) 1173–1178
3. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In Rao, J., Sunar, B., eds.: Cryptographic Hardware and Embedded Systems - CHES 2005. Volume 3659 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 30–46
4. Oswald, D., Paar, C.: Improving Side-Channel Analysis with Optimal Linear Transforms. In Mangard, S., ed.: Smart Card Research and Advanced Applications. Volume 7771 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 219–233
5. Hajra, S., Mukhopadhyayd, D.: On the Optimal Pre-processing for Non-Profiling Differential Power Analysis. In: International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE). (2013)
6. Mehari Msgna, K.M., Mayes, K.: Verifying Software Integrity in Embedded Systems: A Side Channel Approach. In: International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE). (2013)
7. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. Cryptographic Hardware and Embedded Systems - CHES 2004 (2004) 135–152
8. North, D.: An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. RCA Labs. (1943)
9. Trefethen, L.N., Bau III, D.: Numerical linear algebra. Number 50. Siam (1997)
10. Chari, S., Rao, J., Rohatgi, P.: Template attacks. In Kaliski, B., Koç, e., Paar, C., eds.: Cryptographic Hardware and Embedded Systems - CHES 2002. Volume 2523 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 13–28
11. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In Goubin, L., Matsui, M., eds.: Cryptographic Hardware and Embedded Systems - CHES 2006. Volume 4249 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 15–29

12. Moradi, A., Guilley, S., Heuser, A.: Detecting hidden leakages. In Boureanu, I., Owesarski, P., Vaudenay, S., eds.: Applied Cryptography and Network Security. Volume 8479 of Lecture Notes in Computer Science. Springer International Publishing (2014) 324–342