Channel Equalization for Side Channel Attacks

Colin O'Flynn and Zhizhang (David) Chen

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

Abstract. This paper introduces the use of channel equalization as a method of simplifying side channel analysis attacks, 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.

Keywords: side-channel analysis, multivariate, higher order DPA, equalization

1 Introduction

An attack which uses a single point of data in each power trace is a univariate attack, since the statistics of a single random variable are considered. More powerful attacks consider the statistics of several points (i.e. several random variables) related to the leaked information, known as multivariate attacks, or alternatively as Higher-Order attacks. The number of points considered is declared the 'Order' of the attack, a 2^{nd} order attack using two points in the power trace for example as in [1].

Countermeasures that have been presented may protect against the univariate attack, but can be broken by multivariate attacks. This is generally accepted due to the added complexity of multivariate cases, making multivariate attacks a less practical threat. It has been previously reported how the use of physical effects of the measurement channel could combine measurements from many points into one, effectively breaking systems with univariate attacks, despite the algorithms being theoretically secure [2]. This was mostly empirical observations, and does not for example demonstrate how to perform this optimally, or how to evaluate countermeasures with such assumptions.

This work introduces the use of channel equalization to effectively collapse the *entire measured power trace* to a single point, which allows the simpler 'univariate' attack methods to be applied. Compared to existing multivariate attacks, the method presented here is considerably less complex, as it requires no special knowledge of the system under attack. There is no requirement to select points to apply the analysis, since the equalization procedure will generate a suitable matrix for point selection.

This work also approaches the problem from the background of a communications systems. Some background into communications will be given, with an analysis of the Correlation Power Analysis (CPA) attack in the lens of communications.

This background will motivate the use of a channel model along with equalization, which will generate a single data-point using a given power traces and known leakage assumption. This procedure requires a 'training set' to generate the equalizer. Once the equalizer is available for a given system setup, an extremely low-complexity selection criteria is available called minimum distance decoding. The combination of using a linear equalizer to generate a single data-point and minimum-distance decoding allows processing of extremely large datasets.

If a training set is not available, an attack called Channel Estimation Analysis (CEA) is also presented. The CEA attack requires similar assumptions to the CPA, but takes into consideration all data-points for the attack instead of being applied on a single data-point. This attack is fairly efficient, requiring only one complex operation (a pseudoinverse) on the trace set.

The attacks will be demonstrated on physical devices implementing unprotected software AES-128, along with protected software AES-256 using RSM (Rotating SBox Mask)[3].

2 Correlation Power Analysis and the Matched Filter

As a precursor to the introduction of channel estimation, the well-known Correlation Power Analysis (CPA) [4] 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 $Y = \boldsymbol{t}$.

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

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

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[5]. 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, which gives us:

$$y(t) = \int\limits_0^t y(\tau)s(T - t + \tau)d\tau \tag{5}$$

Where we will sample this result only 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:

$$\arg\max_n y(T) = \arg\max_n \int\limits_0^T y(\tau)s_n(\tau)d\tau \tag{6}$$

As $s_n(t)$ is only defined over the internal $0 \leq t \leq T$, this could also be written as the correlation of the received signal $r(t)$ with all candidates $s_n(t)$ at $t = 0$, which would be:

$$\arg\max_n \left(r(t) \star s_n(t)|_{t=0}\right) = \arg\max_n \int\limits_0^T y(\tau)s_n(\tau)d\tau \tag{7}$$

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

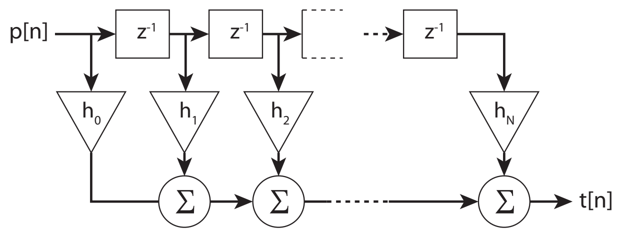

Fig. 1. Simple channel model, where noise can be added to the output if w[n] also needs to be modelled.

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 (8). The objective of 'channel estimation' is to discover the value of the taps, given by the $\boldsymbol{h_{channel}}$ vector.

$$t[n] = \sum_{j=0}^{J-1} h_j p[n-j] + w[n] \tag{8}$$

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 world the 'independent channels' means the different information about each subkey s.

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

$$p_d = (\boldsymbol{t_d} - \mu_{t_d}) \cdot \boldsymbol{h_s} + \mu_p \tag{9}$$

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 (9) can simply be written as:

$$p_{d,s} = \boldsymbol{t_d} \cdot \boldsymbol{h_s} \tag{10}$$

The value of μ_{t_d} will not be known, and instead the estimate $\hat{\mu}_{t_d}$ be 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 (10). Attempting to solve the form of equation (8) and then invert the matrix for use in equation (10) would be equivalent, however the form in (10) 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 (or simulated leakages) on a device with a known secret key. Finding equalizer coefficients *without* a known key will be discussed in section 5. If the coefficients are required to be independent of the secret key, the training set should be generated using many different (known) keys.

For a traces with a given 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:

$$e(d) = p_{d,s} - \hat{p}_{d,s} = p_{d,s} - \left(\hat{\boldsymbol{h_s}} \cdot \boldsymbol{t_d}\right) \tag{11}$$

For notational simplicity this uses the form of (10), if the zero-mean assumption is not made the form in (9) should instead be substituted. Two different options for minimizing this option will be considered: the Least Square (LS) and the Mean Square Error (MSE).

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

$$\sum_{d=0}^{D-1} e^2(d) \tag{12}$$

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 [6]:

$$\hat{\boldsymbol{h}}_{\boldsymbol{s}} = \boldsymbol{t}^{+} \cdot \boldsymbol{p_s} \tag{13}$$

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)[6]. 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.

Minimum Mean Square Error The optimal Minimum Mean Square Error (MMSE) estimate of $p_{\hat{d,s}}$ for a given measurement t_d, under the linear model assumption given by (9), will be given by:

$$\hat{p}_{d,s} = \mu_p + \boldsymbol{C_{pt_s}} \boldsymbol{C_{tt_s}}^{-1} \left(\boldsymbol{t_d} - \mu_{t_d} \right) \tag{14}$$

For a derivation of this solution see e.g. equation (5.659) in [7]. We do not know the true values of the cross-covariance matrix $\boldsymbol{\hat{C}_{tt}}$ and $\boldsymbol{\hat{C}_{pt}}$, so will have to estimate them. The unbiased estimates of $\boldsymbol{\hat{C}_{tt}}$ and $\boldsymbol{\hat{C}_{pt}}$ will be calculated from the training set. If we have a total of D measurements of the data stored in $\boldsymbol{t}_d$, where each measurement corresponds to our known value of $p_{d,s}$, we can find:

$$\boldsymbol{\hat{C}_{tt_s}} = \frac{1}{D-1} \sum_{d=0}^{D-1} \left(\boldsymbol{t_d} - \mu_{t_d} \right) \left(\boldsymbol{t_d} - \mu_{t_d} \right)^T \tag{15}$$

$$\boldsymbol{\hat{C}_{pt_s}} = \frac{1}{D-1} \sum_{d=0}^{D-1} \left(p_{d,s} - \mu_p \right) \left(\boldsymbol{t_d} - \mu_t \right)^T \tag{16}$$

Where (14) – (16) can be directly solved. As $\boldsymbol{C_{tt_s}}$ is a square matrix the inverse *may* exist, unlike in LS where the $\boldsymbol{t}$ matrix must be inverted, which is likely not square and thus the pseudoinverse must be used.

3.3 Applying the Equalizer Information

Once the equalizer coefficients $\boldsymbol{\hat{h}}_s$ 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.

4 Minimum Distance Decoding

We can consider a sequence of D power traces $\boldsymbol{t_d}$, which having processed through all equalizers $\boldsymbol{\hat{h}}_s$ have an output of $\hat{p}_{d,s}$. Based on the known input to the system, we can generate hypothetical outputs of the system $p_{d,s,i}$ based on hypothesis i of subkey s for trace number d.

For each trace d, we aim to minimize $|\hat{p}_{d,s} - p_{d,s,i}|$ by selecting i. Thus the objective is simply to minimize this sum over all traces D, which means finding the value of i which minimizes $e(s,i)$, given in equation (17).

$$e\left(s,i\right) = \sum_{d=0}^{D-1} \left| \hat{p}_{d,s} - p_{d,s,i} \right| \tag{17}$$

Note this assumes that the equalizer has resulted in an unbiased estimator, i.e. $E[\hat{p}_s] = p_s$. If one plotted the distributions we would expect to see them centred around the 'valid' values of p_s, such as $0, 1, \cdots, 8$ for the 8-big hamming weight case. If the linear channel model was *not* valid, instead the distributions may be centred around other values, $p_s^{'}$. This requires the use of $p_s^{'}$ as the 'expected' value in (17), and mapping from p_s to $p_s^{'}$.

In communications systems, it is unreasonable to directly (i.e. via brute force) calculate the minimum distance. A 32-bit sequence for example requires a comparison between 2^{32} values, and is expected to run in close to real time every 32 bits of received signal. The number of comparisons grows exponentially with length of data. By comparison the use of minimum-distance decoding in side-channel analysis grows linearly with the length of traces used, since the size I depends only on the bit-width of the hypothesis. A byte-wise implementation of AES-128 means only $I = 2^8 = 256$ comparisons. Implementations may have larger values of I such as 2^{32}, resulting in large calculation complexity. The problem is still considerably less constrained than with general communications, since the real-time requirement is removed, and implementations may use large amounts of memory such as disk-based arrays.

4.1 Reduced-Memory Decoding

Algorithms exist for performing the minimum distance processing with considerably reduced memory requirements, which in applications with a large I may be necessary. The well known Viterbi decoder will be demonstrated here. First assume that using the channel model each trace has been reduced to $\hat{p}_{d,s}$, and the objective of this algorithm is to recover the vector $\hat{\boldsymbol{p}}_{\boldsymbol{s}} = [p_{0,s}, p_{1,s}, \cdots, p_{D,s}]$. If the device under attack leaked the hamming weight, $\hat{\boldsymbol{p}}_{\boldsymbol{s}}$ would be a vector representing the measured hamming weight corresponding to each input plaintext. Additional work will be required to recover the secret key from $\hat{\boldsymbol{p}}_{\boldsymbol{s}}$.

The general form of the Viterbi decoder is shown in Fig. 2, which corresponds to a 3-bit hamming weight (HW) leakage, with 4 traces measured. Initially, the system starts with a received value $\hat{p}_{0,s}$. The error is calculated for each possible value j of the HW, so that $e_{0,j} = |\hat{p}_{0,s} - j|$. In the general case of N bits, $j = \{0, 1, \cdots, N\}$.

We wish to calculate the sum of the absolute error over every valid path through the decoder, and will select the path which minimizes this error. The result will be the sequence $\hat{\boldsymbol{p}}_{\boldsymbol{s}}$ indicating the most likely value of the hamming weights used by the device under attack. The innovation of the Viterbi decoder is to realize that one does not need to keep a memory of the value of every possible path. Instead only the most likely incoming path at each node is kept. See for example the trace $d = 2$ in Fig. 2. The best error case is kept only, and the other paths in red are trimmed. This greatly limits the memory required, since we only need to store N elements. Note this assumes all paths are equally likely through the decoder: while true for communications theory, this is *not* the case for the side-channel attacks, where some paths may be impossible. Thus it may

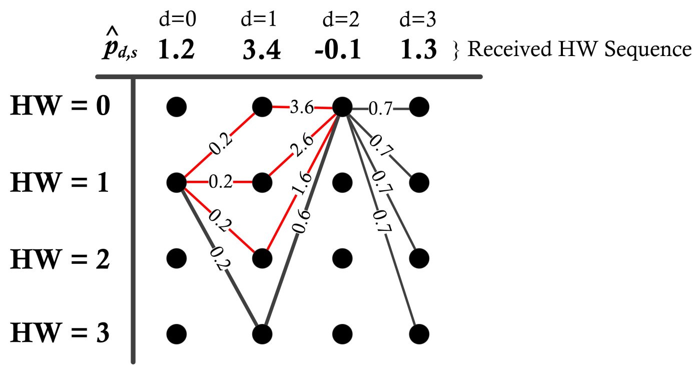

Fig. 2. Viterbi-style decoding trims the search path at each received signal to include only the most likely paths, i.e. the path with least error. This most-probable path is propagated forward. For clarity only the paths starting at HW=1 and passing through HW=0 are shown, the full search would include all possible paths.

be desired to either keep additional paths when trimming, or use a threshold that throws away paths only when their error grows too large.

Finally we must map from $\hat{\boldsymbol{p}}_{\boldsymbol{s}}$ to the leaked information. As the hamming weight (HW) or hamming distance (HD) functions are a many-to-one mapping, there is no unique inverse, so additional work is needed. For the HW, one can see that a N-bit number with a HW of p has $\binom{N}{p}$ solutions. Assume we are again dealing with an 8-bit AES implementation that has a HW leakage. In which case the value $p_{d,s,i}$ indicating the hamming weight of subkey s for trace p with guess i, and having known input text $\boldsymbol{b}_d$ would be:

$$p_{d,s,i} = HW\left(SBox\left(i \oplus b_{d,s}\right)\right) \tag{18}$$

Thus for each value of $\hat{p}_{d,s}$ from the decoder, we can enumerate all possible outputs of the $SBox()$ which would generate the same HW. This enumeration function is called here $HW^{-1}(p)$ Finally we can generate a vector of guesses for the subkey s of the encryption key, $\boldsymbol{E}_s$, based on information in trace d:

$$\boldsymbol{E}_{s,d} = SBox^{-1}\left(HW^{-1}\left(\hat{p}_{d,s}\right)\right) \oplus b_{d,s} \tag{19}$$

Taking the intersection of all vectors should result in a single value for E_s common to all vectors. The number of possibilities enumerated by $HW^{-1}(p)$ will depend on the the value of p, where each value is a solutions to:

$$HW(HW^{-1}(p)) = p$$

The number of solutions will be given by $\binom{N}{p}$. The minimum number of solutions will be for $p = 0$ or $p = N$, with only N possibilities. For the most efficient calculation, one should first find the value of $\hat{p}_{d,s}$ resulting in the minimum number of solutions. This initial set can then be further reduced by removing guesses which cannot satisfy equation (18).

5 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 (9) 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 $\boldsymbol{b}_d^t$[4]. An example of the generation of $p_{d,s,i}$ is given in equation (18), when when generated over all values of d gives the vector $\widetilde{\boldsymbol{p}}_{\boldsymbol{s,i}}^{\boldsymbol{f}}$.

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 6.2.

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 (20) 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.

$$e\left(s,i\right) = \sum_{d=0}^{D^t-1} \left(\left(\widetilde{\boldsymbol{h}}_{\boldsymbol{s,i}} \cdot \boldsymbol{t_d^t} \right) - \widetilde{p}_{s,d,i}^{t} \right)^2 \tag{20}$$

6 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.

6.1 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 (17) 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) which can process the incoming trace measurement a point at a time, avoiding the need to store traces. Such improvements are of limited use, and probably only of interest if one wishes to verify the security of a target with an extremely high number of traces (>10E6), where the storage requirements and recording/processing time may become problematic.

In general, the use of equalization will greatly reduce computational requirements, as the output of equation (9) is a single point for each trace, regardless of the length of each input trace. Applying equation (9) 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.

6.2 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 (20), 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.

A variety of existing libraries for calculating the pseudoinverse exist which simplifies calculation of the equalizer coefficients, since equation (13) can almost directly be coded. Examples of packages implementing the pseudoinverse include MATLAB, NumPy, SciPy, LAPACK, and OpenCV.

7 Attack Results

7.1 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

[1] 16×256

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

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 (9) using power measurements taken with a known plaintext and encryption key. A number of the traces are used for generation of the equalizer, and once the equalizer is known a different set of traces (i.e. not the ones used for profiling) 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 datapoint, which is then ranked by the classic CPA attack algorithm.

Finally an example of the CEA attack is given, where no prior information is known.

Correlation Power Analysis (CPA) For this attack the standard CPA is used. The most likely subkeys are ranked by the correlation coefficient given by equation (1), where it is calculated for each datapoint. The resulting PGE is shown in Fig. 3.

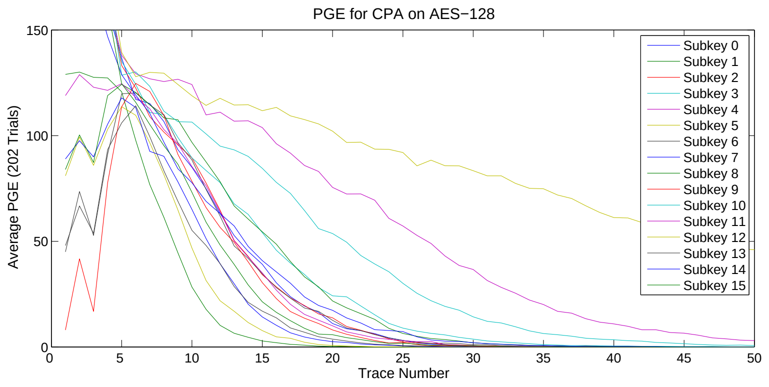

Fig. 3. CPA performed on unprotected AES-128.

CPA with Equalization For this attack the equalizer coefficients are generated from 5000 traces with a known 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 based on $\hat{\boldsymbol{h}}_s$. The results are ranked by the output of the correlation coefficient given by (1). The resulting PGE is shown in Fig. 4.

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. 4. To avoid cluttering the graph these have not been shown.

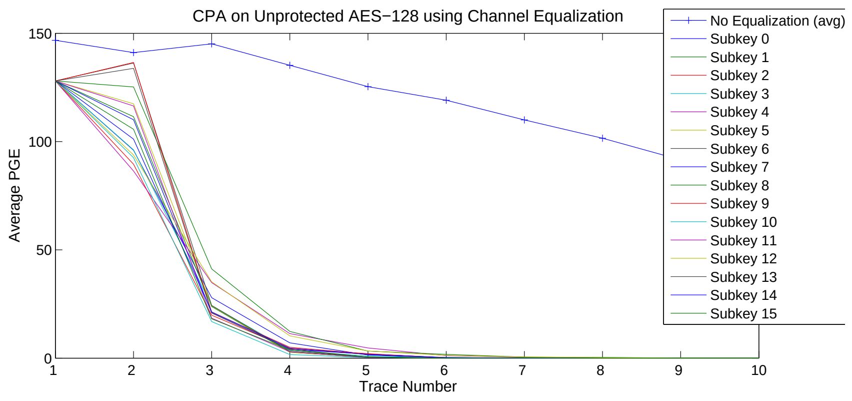

Fig. 4. CPA performed on unprotected AES-128, where traces have been preprocessed by the linear equalizer. The average PGE across all subkeys in shown for an attack without equalization, corresponding to the data in Fig. 3.

Channel Estimation Analysis For this attack no prior knowledge is assumed beyond the assumption about the device leaking the hamming weight. The specified number of traces for each datapoint are split into half; one part becoming the fitting set, one part becoming the test set. The resulting PGE is given in Fig. 5, which can be compared with a CPA attack in Fig. 3. Due to the partitioning of traces, the CEA attack requires more traces compared to the CPA attack.

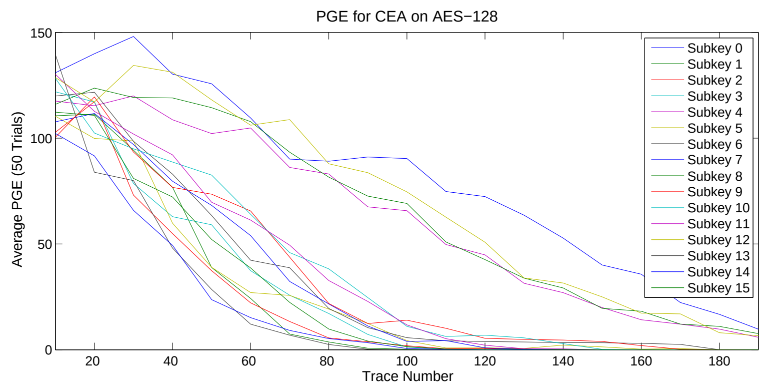

Fig. 5. CEA performed on unprotected AES-128.

7.2 Protected Software AES-256

An protected software AES implementation is used as another example device. The AES-256 code protected by Rotating SBox Mask [3]. This code is the same as used in the DPA Contest v4, 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.

In these examples only the first 16 bytes of the AES-256 key are attacked.

Correlation Power Analysis (CPA) For this attack the standard CPA is used. The most likely subkeys are ranked by the correlation coefficient given by equation (1), where it is calculated for each datapoint. The results are shown in Table 1. There is not movement of subkey PGE with increasing traces, even with 14 000 traces.

Table 1. A standard CPA attack on a AES-256 RSM Implementation, only first 16 key bytes shown.

Traces	Subkey Partial Guessing Entropy															
Used	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4000	3	25	55	167	230	57	118	117	0	36	71	35	85	72	5	71
6000	3	31	56	161	192	80	139	132	0	80	74	37	84	47	8	54
8000	4	27	46	153	169	60	132	112	1	61	72	59	84	39	9	67
10000	2	32	39	145	176	82	134	96	2	55	80	67	81	80	17	103
12000	2	33	41	135	175	77	141	79	4	59	73	51	79	78	26	109
14000	3	25	45	119	171	104	131	79	4	47	72	48	83	80	26	138

CPA with Equalization Using Least-Squares Estimator For this attack the equalizer coefficients are generated from 5 000 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. 6. Three different tests are performed: in the first test, the equalizer was generated from traces where the encryption key changes on each trace. This equalizer should work well with any secret key, which is shown in Fig. 6A. The second equalizer is used in both Fig. 6B & Fig. 6C, and is generated from 5 000 traces with a fixed key. When this equalizer is used on test traces generated from the same secret key, it performs better than the random-key equalizer, as in Fig. 6B. However using this equalizer on test traces generated with a different secret key, as in Fig. 6C, it performs poorly.

Thus the most realistic attack is one where the channel equalizer is generated from a randomly changing 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.

Channel Estimation Analysis 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 Table 2 for each subkey — the cost of the pseudoinverse on large trace sets makes calculating many datapoints a lengthy process, which is why this data is shown in tabular forms instead of a graph. Whilst a complete attack was unsuccessful, the PGE of subkeys is considerably better compared to the CPA attack given in Table 1. Both the CPA & CEA attack are using the same assumptions — the device leaks the Hamming Weight at the output of an SBox.

Table 2. A standard CEA attack on a AES-256 RSM Implementation, only first 16 key bytes shown.

Traces	Subkey Partial Guessing Entropy															
Used	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4000	6	29	10	57	14	34	2	0	2	11	66	5	3	25	0	1
6000	39	1	180	29	7	70	0	0	2	84	104	0	0	0	0	0
8000	37	0	49	22	3	13	0	0	0	82	3	0	0	0	0	7
10000	26	0	15	0	7	2	2	2	0	49	1	1	0	0	0	6
12000	25	0	20	0	15	2	0	1	0	14	19	2	0	1	0	7
14000	26	0	13	0	25	5	0	0	0	4	12	1	3	6	0	1

8 Future Work

8.1 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 [9]. 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.

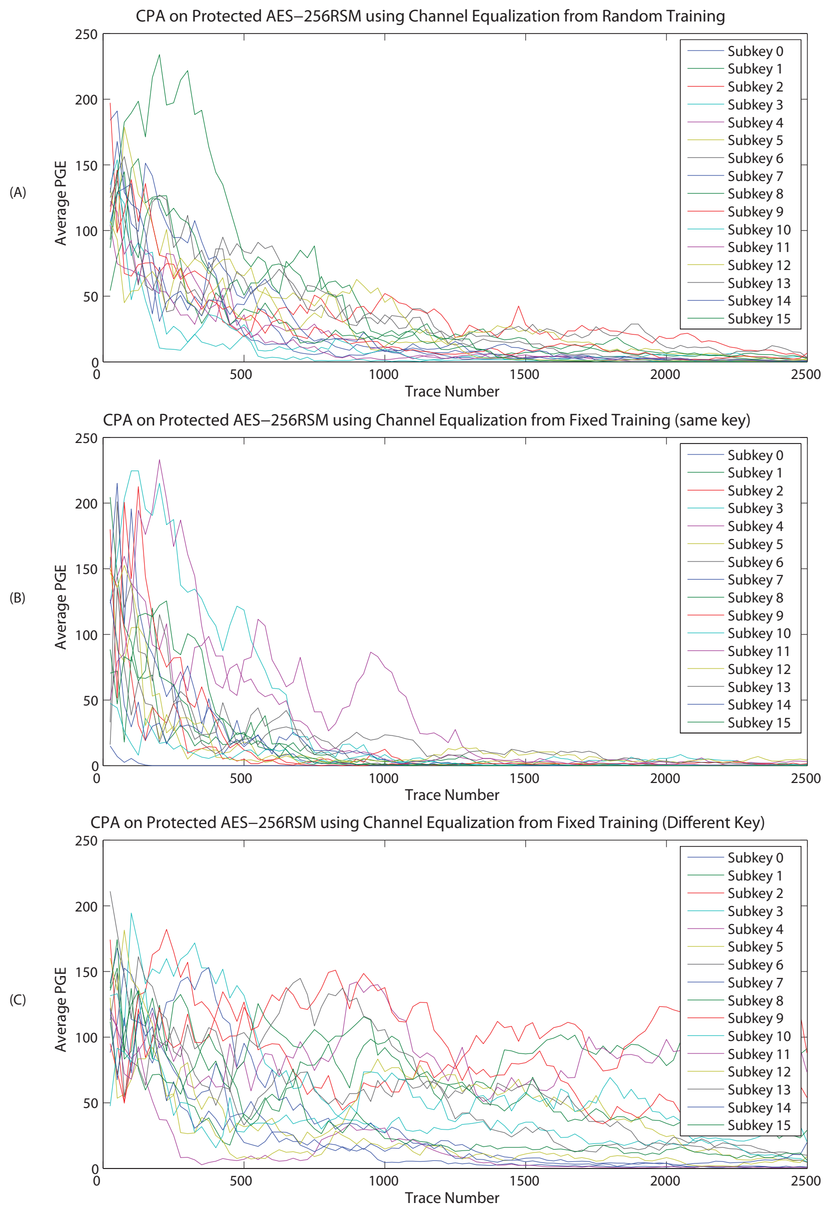

Fig. 6. CPA performed on AES-256 protected with RSM, where traces have been preprocessed by Channel Equalization. Three different systems are tested: in (A) the equalizer is generated from traces with a random key changing on each trace, in (B) and (C) the equalizer is generated from traces with a fixed key.

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.

In addition as the output of the linear equalizer results in a single datapoint, this simplifies comparison of statistical properties such as the variance of the measurement, which is linked to the Signal to Noise Ratio (SNR). It thus becomes possible to discuss the specific effect on the SNR a countermeasure has on the system.

8.2 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.

8.3 Online Calculation of CEA

For security analysis, it is often desired to determine the progression of a metric over increasing number of observed traces. The results section for example shows the Partial Guessing Entropy (PGE) vs. observed traces. This requires a calculation of the pseudoinverse for $1, 2, 3, \cdots, D$ traces. An improvement would be to use update equations, which takes an existing matrix with a known pseudoinverse, and appends a row/column to this matrix.

8.4 Other Ranking Metrics

The ranking metrics used here are very basic — either the CPA attack or the minimum-distance rank. Ideas such as variance based distinguishers[10] could be extended to analyse higher order moments for the output of equation (9).

9 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 completely agnostic to 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 protected AES implementation.

Complete code for implementation of all attacks, including measurements, is available at [Removed from Anonymous].

Acknowledgements Removed from anonymous version.

References

1. Messerges, T.: Using second-order power analysis to attack dpa resistant software. In Koç, e., Paar, C., eds.: Cryptographic Hardware and Embedded Systems - CHES 2000. Volume 1965 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2000) 238–251
2. 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
3. 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
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. Cryptographic Hardware and Embedded Systems - CHES 2004 (2004) 135–152
5. North, D.: An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. RCA Labs. (1943)
6. Trefethen, L.N., Bau III, D.: Numerical linear algebra. Number 50. Siam (1997)
7. Van-Trees, H., Bell, K.: Detection, Estimation, and Modulation Theory: Part 1. 2nd edn. Wiley (2013)
8. Liavas, A., Tsipouridou, D.: On the performance of the mismatched mmse and the ls linear equalizers. Trans. Sig. Proc. **55**(7) (July 2007) 3302–3311
9. Moradi, A., Mischke, O.: On the simplicity of converting leakages from multivariate to univariate - (case study of a glitch-resistant masking scheme). In: CHES, Springer (2013) 1–20
10. Standaert, F.X., Gierlichs, B., Verbauwhede, I.: Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In Lee, P., Cheon, J., eds.: Information Security and Cryptology - ICISC 2008. Volume 5461 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 253–267