An Automated End-to-End Side Channel Analysis Based on Probabilistic Model

Jeonghwan Hwang and Ji Won Yoon *

Graduate School of Information Security, Korea University, Seoul 02841, Korea; ju.su.splab@gmail.com
* Correspondence: jiwon_yoon@korea.ac.kr

Received: 13 November 2019; Accepted: 23 March 2020; Published: 30 March 2020

Abstract: In this paper, we propose a new automated way to find out the secret exponent from a single power trace. We segment the power trace into subsignals that are directly related to recovery of the secret exponent. The proposed approach does not need the reference window to slide, templates nor correlation coefficients compared to previous manners. Our method detects change points in the power trace to explore the locations of the operations and is robust to unexpected noise addition. We first model the change point detection problem to catch the subsignals irrelevant to the secret and solve this problem with Markov Chain Monte Carlo (MCMC) which gives a global optimal solution. After separating the relevant and irrelevant parts in signal, we extract features from the segments and group segments into clusters to find the key exponent. Using single power trace indicates the weakest power level of attacker where there is a very slight chance of acquiring as many power traces as needed for breaking the key. We empirically show the improvement in accuracy even with presence of high level of noise.

Keywords: side channel attack; power analysis; markov chain Monte Carlo; change point detection

1. Introduction

Many Side Channel Analysis attacks have succeeded in breaking the secret keys analyzing power trace(s) generated from devices. However, there are many assumptions and limitations on these attacks. Some of them exploit as many power traces as needed to recover secrets. In addition, others have proposed methods to recover secrets from overall power trace(s) but not their exact location on the power trace from which each bit of secrets have been recovered.

One of the well known approaches is to find the reference window and apply a peak detecting algorithm [1,2]. However, the success of this approach heavily depends on the selection of a “good” reference window and the performance of peak detecting algorithms since we cannot search for all the windows in polynomial time. Therefore we need an automated approach that is feasible in polynomial time and does not require human intervention to succeed. Side channel analysis with machine learning has gained much interest [3–6].

Our work suggests a new approach to finding keys in a Bayesian approach. Our contributions on Side channel analysis and signal processing are as follows:

- We exploit only one single power trace and recover the secret.
- We suggest the methods to compute the probability of locations from which the secret came and find out global optimal solutions in a Monte Carlo approach.
- We suggest the methodology that is more robust than ad-hoc attacks in the presence of noise.

In correlation analysis [2,7–11], a set of power traces is used to find a correlation between the power trace and the key guess. Our method assumes weaker power of an attacker where a chance of
acquiring more than one power trace is slight. Analyses with clustering methods [12–14] are researched. Recently, horizontal attacks [7] have been succeeding with clustering algorithms [15,16]. These attacks exploiting clustering algorithms and horizontal attacks have so far fixed the dimension (length) of the segment/subsignal of power trace as \( \frac{\text{trace\_length}}{\text{key\_bits}} \) when treating segments. However, when treating time series, dimension of data segments have to be chosen with careful concern. There is no guarantee that operations are executed with an exact period of \( \frac{\text{trace\_length}}{\text{key\_bits}} \). Our work does not assume the operations are executed periodically but rather estimates the start and end points of the operation executions. Moreover, we suggest method to extract feature from time series segments in different lengths.

2. Notation and Problem Definition

2.1. Notations

- \( y_t \): The \( t \)-th signal (time location). Total signal length is \( T \), so we simplify \( y_{1:T} = (y_1, y_2, \cdots, y_T) \).
- \( r_t \): The \( t \)-th random variable which indicates whether \( t \)-th point is change point or not. That is, 
  \[
  r_t = \begin{cases} 
  1, & \text{if } t \text{ is a change point.} \\
  0, & \text{otherwise.}
  \end{cases}
  \]
- \( K \): The number of change points. That is, \( K_r = \sum_{t=1}^{T} r_t \)
- \( \tau_k \): The \( k \)-th change point. As \( K_r \) is the number of total change points, \( 1 \leq k \leq K_r \) and trivially \( \tau_1 = 1 \) and \( \tau_{K_r+1} = T + 1 \).
- \( \lambda \): The hyperparameter that controls the number of change points
- \( \sigma \): The hyperparameter that controls the shape of probability density function
- \( \mu \): The hyperparameter that controls the mean of segments.
- \( V \): The hyperparameter that controls the variance of segments.
- \( \theta \): The set of parameters. That is, \( \theta = [\lambda, \sigma, \mu, V] \).
- \( D \): The dimension of feature.
- \( \bar{\phi}_k \): The feature extracted from the \( k \)-th segment with dimension of \( D \).
- \( c_k \): The cluster assigned to \( k \)-th segment.

2.2. Problem Definition

Our goal is to divide the power trace into operation-relevant segments and assign clusters to each segment so that we can figure out which operation was executed. In Figure 1 is shown whole process. Formally, we can define our goal as:

\[
\begin{align*}
    r_{1:T} &= \mathbb{E}_{p(r_{1:T}|y_{1:T},\theta)}[r_{1:T}] \quad \text{(goal of step 1)} \\
    r^M_{1:T} &= g(r_{1:T}, y_{1:T}) \quad \text{(goal of step 2)} \\
    \bar{\phi}_k &= f(y_{\tau_k;\tau_{k+1}-1}, r^M_{1:T}) \quad \text{(goal of step 3)} \\
    c^*_k &= \arg\min_{c \in \{\text{Idle, Square, Multiply}\}} \left\| \bar{\phi}_k - \sum_{k=1;K_r}^{k=K_r,M} \bar{\phi}_k \right\|_2^2 \quad \text{(goal of step 4)} \quad (1)
\end{align*}
\]

We defined each step, rather than one whole global model that segments time series and searches cluster (i.e., some function \( h(\cdot), c^* = h(y_{1:T}) \)) due to the high complexity of that model, if it exists.
2.2.1. Change Point Detection

We have found from the power trace that there is an idle period or a piece-wise constant between operations of binary exponentiation, square and multiplication. Exploring this period is the most important part of whole work since segments divided only by the exact locations of operations will have similar patterns. We can model solving change point detection problem (with their number unknown) [17]. However we do not adopt reversible jump MCMC [18,19]. In this subproblem embedding the change point detection algorithm, we only find idle periods and incomplete segments are made. We then make complete(operation-relevant) segments by combining the incomplete segments in Section 2.2.2. Reason behind dividing two steps is that first, we do not have any information about the key or the shape of power trace of each operation and change point detecting problem whose number of change points are not known is already complex enough to avoid putting detection of constants and unknown shapes together into model. For detecting piece-wise constants, we can define by Bayes’ theorem the posterior distribution of change points $r_{1:T}$

$$p(r_{1:T}|y_{1:T}, \theta) \propto p(y_{1:T}|r_{1:T}, \theta) p(r_{1:T}|\theta),$$

and our goal as:

$$\tilde{r}_{1:T} = E_{p(r_{1:T}|y_{1:T}, \theta)}[r_{1:T}].$$

2.2.2. Merging Segments

As mentioned above, in this subproblem, we merge incomplete segments to complete segments whose start and end points indicate the start and end points of the operation. The goal here is building the merging function $g(\cdot)$ as

$$r_{1:T}^M = g(r_{1:T}, y_{1:T}).$$

Only merging but not splitting is allowed in function $g(\cdot)$, so $K_r^M \leq K_r$ and \{1 \leq k \leq K_r^M \} \in \{1 \leq k \leq K_r \} holds.

2.2.3. Extracting Features from Time Series Segments

It is known that previous correlation power analysis attacks have used power traces or parts of power traces that are cut in the same length. However, the lengths of power traces (or their parts) are not guaranteed to have the same length. Therefore, the model or algorithm for extracting information relevant to each operation must be capable of treating subsignals in different length. In this part, given the segments in different lengths, we extract the features that will be the input to the clustering part which needs fixed dimension.

$$\Phi = [\phi_1, \cdots, \phi_k, \cdots, \phi_K]^T.$$
where each row $\hat{\phi}_k = [\phi_{k,1}, \ldots, \phi_{k,d}]^T$ is a feature vector of $k$-th segment. The goal of this subproblem is building the feature extractor $f(\cdot)$ that is capable of coping with the segments in different lengths $\hat{\phi}_k = f(y_{T_\tau:k-1:T_\tau+1}, r_{1:T}^M)$. (6)

2.2.4. Clustering Features

The last step is clustering all the segments with features extracted. Clustering is an unsupervised machine learning approach that assigns each data point a cluster based on similarity (or distance). After all the segments are assigned the clusters, we can find the key exponent. Our goal is identifying three clusters of features. In general, if the number of clusters is not known, optimizing the number of clusters is required. From this point of view, the number of clusters should be carefully considered to obtain highly accurate performance, but we can easily put three on it since there are only three patterns we look for, which are square, multiplication and the idle period between operations. After we identify the clusters, it becomes a trivial problem to decide which operation or period is related to one of clusters.

$$c_k^* = \arg \min_{c \in \{Idle, Square, Multiply\}} ||\hat{\phi}_k - \frac{\sum_{k=1, k \neq M} \hat{\phi}_k I(c_k = c)}{\sum_{k=1, k \neq M} I(c_k = c)}||^2_2.$$ (7)

3. Proposed Approach

3.1. Preprocessing

There are many countermeasures to side channel analysis including the random noise addition. In order to deal with the random noise addition, we sample each power trace with the median filter. The median filter will decrease the effect of noise and reflect the trend of power signal. We apply the median filter to the power trace. We use the stride size equal to the window length. As a result, we reduce the length of power trace to analyze. Since the magnitude of power trace is a positive value, we use the absolute value of the power trace:

$$y_{1:T} = \text{MEDSAMPLE}(|P_{1:N_p}|, N_w),$$ (8)

where $N_p$ is a length of the original power trace and $N_w$ is window size. Figure 2 shows the effect of preprocessing the power traces.
3.2. Change Point Detection

3.2.1. Posterior Distribution

As mentioned above, the model here detects the piecewise constant part from the time series. 

\[ y_t = m_k + \epsilon, \]  

where random variable \( m_k \sim \mathcal{N}(\mu, V) \) is a mean of time series between \( \tau_k \) and \( \tau_{k+1} - 1 \) and \( \epsilon \sim \mathcal{N}(0, \sigma^2) \). The likelihood that \( y_{1:T} \) is observed given \( r_{1:T}, m_{1:K_r} \) and \( \theta \) is

\[
p(y_{1:T}|r_{1:T}, m_{1:K_r}, \theta) = \prod_{k=1}^{K_r} \prod_{l=\tau_k}^{\tau_{k+1}-1} \mathcal{N}(y_l; m_k, \sigma^2) 
\]

\[
= \left( \frac{1}{2\pi\sigma^2} \right)^{-\frac{T}{2}} \exp \left( -\frac{1}{2\sigma^2} \sum_{k=1}^{K_r} \sum_{l=\tau_k}^{\tau_{k+1}-1} (y_l - m_k)^2 \right). \tag{10}
\]

By conjugacy of the exponential family \([20]\), posterior distribution of \( m_{1:K_r} \) is also Gaussian. 

\[ m_k \sim \mathcal{N} \left( \frac{\sqrt{\nu_k + \sigma^2 \mu}}{\sqrt{\nu_k + \sigma^2}}, \frac{\nu_k}{\nu_k + \sigma^2} \right), \]

where \( \nu_k = \sum_{l=\tau_k}^{\tau_{k+1}-1} y_l \) and \( n_k = \tau_{k+1} - \tau_k \). If we plug the probability of \( m_{1:K_r} \) into Equation (10), we get by identification,
Therefore, we can get from Equation (11),

\[
p(y_{1:T} | r_{1:T}, \theta) = \left(\frac{1}{\pi \sigma^2}\right)^{-\frac{3}{2}} \left(\frac{V + \sigma^2}{\sigma^2}\right) \frac{2^\frac{K_r}{2}}{\lambda} \exp \left(-\frac{1}{2(V + \sigma^2)} \sum_{k=1}^{K_r} (y_t - \mu)^2 + \frac{V}{\sigma^2} \sum_{k=1}^{K_r} \sum_{t=\tau_k}^{\tau_k+1-1} (y_t - \bar{y}_k)^2\right). \tag{12}
\]

The trivial solution for this problem is to make every point as a change point, so that the sum of errors becomes least. In order to avoid this situation we have to model the number of change points \( K_r \). We modelled the prior distribution to control the number of change points as Bernoulli distribution.

\[
p(r_{1:T} | \lambda) = \lambda^{K_r} (1 - \lambda)^{n-K_r}. \tag{13}
\]

By Bayes’ theorem we can model the posterior distribution of random variable \( r_{1:T} \) by putting Equation (11) and (13) together as follows:

\[
p(r_{1:T} | y_{1:T}, \theta) = \frac{p(r_{1:T}, y_{1:T} | \theta)}{p(y_{1:T} | \theta)} \\
\propto p(r_{1:T}, y_{1:T} | \theta) \\
= p(y_{1:T} | r_{1:T}, \sigma) p(r_{1:T} | \lambda). \tag{14}
\]

Joint distribution of \( r_{1:T} \) and \( y_{1:T} \), \( p(r_{1:T}, y_{1:T} | \theta) \) is needed since the evidence of the observation \( p(y_{1:T} | \theta) \) is intractable to compute.

\[
p(r_{1:T}, y_{1:T} | \theta) = p(y_{1:T} | r_{1:T}, \sigma) p(r_{1:T} | \lambda) \\
= \left(\frac{1}{2\pi \sigma^2}\right)^{-\frac{3}{2}} \left(\frac{V + \sigma^2}{\sigma^2}\right) \frac{2^\frac{K_r}{2}}{\lambda} \exp \left(-\frac{1}{2(V + \sigma^2)} \sum_{k=1}^{K_r} (y_t - \mu)^2 + \frac{V}{\sigma^2} \sum_{k=1}^{K_r} \sum_{t=\tau_k}^{\tau_k+1-1} (y_t - \bar{y}_k)^2\right) \\
= Z(y_{1:T}, \theta) \exp(-H(r_{1:T})), \tag{15}
\]

where energy function of \( r_{1:T} \), \( H(r_{1:T}) = \frac{V}{2(\sigma^2+V)} E_{r_{1:T}} + \gamma K_r \), \( E_{r_{1:T}} = \sum_{k=1}^{K_r} \sum_{t=\tau_k}^{\tau_k+1-1} (y_t - m_k)^2, \gamma = \frac{1}{2} \log \left(\frac{\sigma^2+V}{\sigma^2}\right) + \log \left(\frac{1-\lambda}{\lambda}\right) \) and \( Z(y_{1:T}, \theta) \) is an intractable normalizing constant.

3.2.2. Markov Chain Monte Carlo

We use Markov Chain Monte Carlo (MCMC) to find the global optimal solution [21,22] for the model we designed. In Reference [19], \( m_k \) was also the random variable to infer and apply reversible jump MCMC to cope with changing dimension (the number of \( m_k \)s to estimate), but our work considers only \( r_{1:T} \) and replace \( m_k \) with \( \bar{y}_k \) and simulate only for change points.
We detect the piecewise constants by computing the expectation of random variable $r_{1:T}$,

$$\hat{r}_{1:T} = \frac{1}{n} \sum_{s=1}^{n} r_{1:T}^s,$$

where $r_{1:T}$ is a true mean of $r_{1:T}$ and $\hat{r}_{1:T}$ is an Monte Carlo estimate of $r_{1:T}$ when $r_{1:T}^s \sim p(r_{1:T})$. The last part holds when $n \to \infty$.

Instead of generating samples from the intractable posterior distribution $p(r_{1:T}|y_{1:T}, \theta)$, we propose a function from which is easy to draw samples, namely proposal function (Metropolis-Hastings, Algorithm 1). We can think of two proposal functions regarding the property of change point detection problem. Given one sample, one of change points can be popped and two segments are merged or one change point is born and segment is split. Either case, this is reverting $r_t, 0 \to 1$, or $1 \to 0$. The other proposal function is a swap between $0 \leftrightarrow 1$.

**Algorithm 1 Metropolis Hastings**

```
procedure MH($p(\cdot), q(\cdot), r_{1:T}, n$)  \> posterior, proposal and initial sample, number of samples
  for $s = 1, 2, \ldots, n$ do
    $r_{1:T}^* \sim q(r_{1:T}|r_{1:T}^{s-1})$
    $\alpha \leftarrow \frac{p(r_{1:T}^*)q(r_{1:T}|r_{1:T}^{s-1})}{p(r_{1:T}^s)q(r_{1:T}^s|r_{1:T}^{s-1})}$
    $a \leftarrow \min(1, \alpha)$
    $u \sim \mathcal{U}(0, 1)$
    if $u < a$ then
      $r_{1:T}^s \leftarrow r_{1:T}^*$
    else
      $r_{1:T}^s \leftarrow r_{1:T}^{s-1}$
  return $r_{1:T}^1, r_{1:T}^2, \ldots, r_{1:T}^n$
```

The proposal function for MCMC is succession of reverting and swapping with probability of

$$q(r_{1:T}^s|r_{1:T}) = \begin{cases} \frac{1}{T}, & \text{if revert} \\ \frac{1}{K_{s-1}}, & \text{if swap.} \end{cases}$$

This proposal probability is reducible, when calculated in Metropolis-Hastings, to

$$\alpha = \frac{p(r_{1:T}^*)q(r_{1:T}^{s-1}|r_{1:T})}{p(r_{1:T}^s)q(r_{1:T}^s|r_{1:T}^{s-1})} = \frac{p(r_{1:T}^*)}{p(r_{1:T}^{s-1})},$$

since in reverting case the probability only depends on the length of the sample and in swapping case the number of change points does not change ($K_{s-1} = K_s$).

Figure 3 shows the $r_{1:T}$ obtained from MCMC section. However this model only detects the piecewise constants, so non-constant parts (i.e., drastic changes) are all high in probability of being change points.
3.3. Merging Segments

In Reference [23], it is shown that by controlling parameter and adopting various models, various goals can be achieved more than just detecting piecewise constants. However, we assume the least about the data that adopted merging segments part. The change points obtained from Section 3.2 indicate the locations where mean of each segment is changed. Therefore, when operations are executed, it is likely that the operation part is split into many segments and there are more than one change points. We detect whether segments are from the operation part or the idle part and merge segments from the operation part. We consider the following two properties of the idle part.

- Whether the length of segment is long enough to be a segment
- Whether the segments suspected as idle periods lie on similar level of power

Details for merging segments are on Algorithm 2 below. Figure 4 shows merged segments.

![Figure 3. Approximate expectation of \( r_t \).](image)

![Figure 4. Change points and mean of each segment based on \( r^M_{1:T} \).](image)
We visualize coefficients by reconstructing power traces in Figure 5.

Algorithm 2 Merge Segments

\textbf{procedure} MERGE($r_{1:T}, y_{1:T}$)

\begin{itemize}
  \item Initialize vector $\text{len\_seg}[1 : K_f]$ with $k$th element as $\tau_{k+1} - \tau_k$
  \item Initialize vector $\text{mean}[1 : K_f]$ with $k$th element as $y_k$
  \item $[\text{sorted\_mean}[1 : K_f], \text{Index}[1 : K_f]] \leftarrow \text{SORT}(\text{mean})$ \Comment*[r]{in descending order}
  \item $\text{best\_so\_far} \leftarrow -\infty$
  \item $TH \leftarrow 0$
  \item \textbf{for} $k = 1, 2, \ldots, K_f$ \textbf{do}
    \begin{itemize}
      \item $\text{mean\_iter} \leftarrow (T - \sum_{i=1}^{k} \text{len\_seg}[\text{Index}[i]]) \times k$
      \item \textbf{if} $\text{best\_so\_far} < \text{mean\_iter}$ \textbf{then}
        \begin{itemize}
          \item $\text{best\_so\_far} \leftarrow \text{mean\_iter}$
          \item $TH \leftarrow k$
        \end{itemize}
    \end{itemize}
  \item $\text{Index}^* \leftarrow \{\text{Index}[i] | TH \leq i \leq T\}$
  \item \textbf{for} $t = 1, 2, \ldots, T$ \textbf{do}
    \begin{itemize}
      \item $r_{M}[t] \leftarrow \begin{cases} 1 & \text{if } t \in \text{Index}^* \\ 0 & \text{otherwise.} \end{cases}$
    \end{itemize}
  \item \textbf{return} $r_{M}^{T} = r_{M}[1 : T]$
\end{itemize}

3.4. Extracting Features from Time Series Segments

In this part, we extract features of fixed dimension from the segments. That is, given the segment $y_{\tau_k: \tau_{k+1} - 1}$, we compute feature of fixed dimension $\phi_k$. We applied two approaches to extract the features and this part is to be further researched.

3.4.1. Polynomial Least Square Coefficients

First approach we apply is polynomial fitting. Polynomial model has its form as below:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_{D-1} x_i^{D-1} + \epsilon = \beta^T x_i + \epsilon,$$

where $\beta_d$ is a $d$th parameter which describes the influence of $x^d$ on $y$ and $\beta = [\beta_0, \beta_1, \beta_2, \cdots, \beta_{D-1}]^T, x_i = [1, x_i, x_i^2, \cdots, x_i^{D-1}]^T$. The solution to polynomial fitting with minimum least squared error exists in closed form:

$$\hat{\beta} = (X^T X)^{-1} X^T y,$$

where $n$-by-$D$ matrix $X = [x_1, x_2, \cdots, x_n]^T$ is concatenation of $n$ data samples. Now that each segment is given, for segment $y_{\tau_k: \tau_{k+1} - 1}$, let $X_k = [x_{\tau_k}, x_{\tau_k+1}, \cdots, x_{\tau_{k+1} - 1}]^T$ and $x_i = [1, t, t^2, \cdots, t^{D-1}]^T, \tau_k \leq t \leq \tau_{k+1} - 1$. Then we can estimate $\hat{\beta}_k$ for each segment of time series and let it be a $D$-by-1 feature of each segment as follows:

$$\phi_k = \hat{\beta}_k = (X_k^T X_k)^{-1} X_k^T y_{\tau_k: \tau_{k+1} - 1}.$$

We visualize coefficients by reconstructing power traces in Figure 5.
3.4.2. Histogram

The second approach is making a histogram out of each segment. Each histogram shares the same scale level of bins. The size of one bin is \( \frac{\max y_{T_k} - \min y_{T_k}}{D} \). Once we normalize the histogram to sum up to 1, this gives a distribution of each segment.

\[
\bar{\phi}_k = \text{Histogram}(y_{T_k:T_{k+1}-1}, D)
\]

3.5. Clustering Features

In this part, we finally make clusters of the features so that we can match each segment to the operation which has been executed when generating the segment. We apply K-means clustering algorithm. Based on the pre-defined distance measure and number of clusters, the K-means algorithm repeats until convergence assigning data points to cluster based on distance and computing the mean of each cluster [20]. We assigned number of clusters \( K = 3 \) based on the number of operations (Square, Multiply and optionally idle period). For coefficients of Section 3.4.1, Euclidean distance \( \text{Distance}(\bar{\phi}_i, \bar{\phi}_j) = \sqrt{[(\bar{\phi}_i - \bar{\phi}_j)^T(\bar{\phi}_i - \bar{\phi}_j)]} \) is defined. For histogram coefficients of Section 3.4.2, both Euclidean distance and symmetrized divergence \( \text{Distance}(\bar{\phi}_i, \bar{\phi}_j) = KL(\bar{\phi}_i|\bar{\phi}_j) + KL(\bar{\phi}_j|\bar{\phi}_i) \) (only for normalized histogram) are defined as distance measures. The performance of K-means algorithm, however, is affected by the initial points. So we run K-means algorithm multiple times and evaluate each run [24]. Only with data we used should we evaluate each run so that any other information is not reflected and result of K-means is evaluated fairly. We adopt Davies-Bouldin index (DB index) to evaluate each run and choose best performing clusters. Desirable clusters, with high inter-cluster distance and low intra-cluster distance, produce low DB index [25]. Figure 6 shows recovering key exponent.

\[
\text{Figure 5. Time series and reconstruction with polynomial coefficients. (a) Time series; (b) Polynomially reconstructed time series.}
\]

\[
\begin{align*}
\text{(a)} & \quad \text{(b)}
\end{align*}
\]

\[
\text{Figure 6. Key exponent recovery process. (a) t-SNE visualized polynomial coefficients clustered; (b) Recovering key exponent with the clusters.}
\]
4. Experiments and Results

The experiments were conducted under the environment below:

- The window size $N_w$ is 1000, $N_p$ is dependent on input data, which in our experiment is 1,657,401.
- Hyperparameters were selected as follows: $\lambda = 48/T$, $\sigma = 9.644e-05$, $V = 7.377 - e07$, $\mu = \sum_{t=1}^{T} y_t / T$
- Binary exponentiation of RSA 16-bit algorithm was executed on Samsung Galaxy S3 smartphone. Electromagnetic wave was sampled by oscilloscope. Sampling rate was 500 MS/s.

As mentioned, $\lambda$ controls the number of change points. The best guess about the key exponents without any knowledge is a half of 1s and the other half of 0s. The number of change points will then be $48 (= 2 \times (16 + 8))$ as bit of 0 leads to only square operation and bit of 1 leads to square and multiplication. The best guess for $\mu$ is the mean of the time series. Though $\sigma$ and $V$ should be optimized for the best inference, we have experimentally chosen the values of $\sigma$ and $V$ among the candidates that were sampled during MCMC process.

Our approach is evaluated with criteria below:

- To which level of noise added and inserted to signal (Signal to Noise Ratio, SNR) does the approach work
- The quality of clusters with external information

The first criterion shows the robustness of the approach to the noise whereas adding noise is often suggested as a countermeasure to the side channel analysis. For the second criterion, the external information (information not used for making clusters) we adopt is the very key exponent we want to estimate.

4.1. Comparison with Naive Peak Detection

This experiment focuses on comparing the proposed approach to naive correlation peak detecting method. Figure 7a was obtained by computing covariance with the entire signal and sliding window size of 100. The most general sliding window is starting with the first part and sliding window size was selected as 100 in a ‘naive’ way. As seen in Figure 7b, peaks were found to satisfy two thresholds: covariance is positive and minimum distance between two nearest peaks is larger than 20 in time scale. Without these thresholds, too many irrelevant peaks were found. Our approach showed better performance in finding the locations of the executed operations and finding that of inserted noise.

![Cross-covariance and peaks](image)

**Figure 7.** Cross-covariance and peaks. (a) Cross-covariance with sliding window; (b) Peaks on cross-covariance.

Stems are drawn on $r_{i,j}^{MT}$ and peaks in Figure 8a,b. Noise inserted are located on time series from $t = 759$ to $t = 924$. The proposed method successfully finds out the location of noise as well as other operations whereas naive method does not successfully find out the location of operations nor the inserted noise.
4.2. Noise Level

We have experimented on 16 levels of noise. We incremented the ratio of the standard deviation of noise and the raw signal linearly by 0.2 up to 3.0. Table 1 shows how consistent our work is when a different level of noise is added. From the table, we see that when the ratio of standard deviation is 1.8, $K_{r_{1}T}$ has changed. From that, the number of change point keeps changing although with some level of noise it remained 44. When the noise is added, the change point changes and sometimes its number also changes.

Table 1. The number of change points with respect to noise level (True $K = 44$).

| std(noise)/std(signal) | 0   | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| SNR(db)                | 13.98 | 7.96 | 4.44 | 1.94 | 0.0 | -1.58 | -2.91 | |
| $K_{r_{1}T}$           | 44  | 44  | 44  | 44  | 44  | 44  | 44  | 44  |
| std(noise)/std(signal) | 1.6 | 1.8 | 2.0 | 2.2 | 2.4 | 2.6 | 2.8 | 3.0 |
| SNR(db)                | -4.07 | -5.09 | -6.01 | -6.84 | -7.60 | -8.29 | -8.94 | -9.53 |
| $K_{r_{1}T}$           | 44  | 45  | 47  | 44  | 47  | 44  | 48  | 58  |

For comparing $\tau_{1,K}$, we have chosen only a standard deviation ratio of 0.2–1.6 since besides those, the numbers of change points have already been changed. Average absolute error on $\tau_{1,K}$ compared to signal without noise is computed as $\frac{\sum_{k=1}^{K_{r_{1}T}} |\tau_{raw,k} - \tau_{noised,k}|}{K_{r_{1}T}}$. Table 2 shows that average absolute error is no more than 3, and especially for noise level 0.2, 0.4 and 0.6, the average absolute error is less than 1.
Table 2. Error on $\tau_{1,k}$ with respect to noise level.

| std(noise)/std(signal) | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| SNR(db)                | 13.98 | 7.96 | 4.44 | 1.94 | 0.0 | -1.58 | -2.91 | -4.07 |
| $\sum_{k=1}^{T} |\tau_{k}^{raw} - \tau_{k}^{noised}|/K_{1,T}$ | 0.5 | 0.5 | 0.89 | 1.25 | 1.23 | 1.77 | 2.34 | 2.39 |

4.3. External Information

Next we compared clusters with external information, the actual key exponent. This is not a part of our approach since the actual key exponent is used. This part rather evaluates our approach to compare our estimate with the label. Table 3 shows the the average accuracy of 100 runs of the K-means algorithm. We used polynomial coefficients to 3rd degree ($D = 4$) and for the histogram we used $D = 60$. Clustering polynomial least square coefficient features are not affected much by the noise level whereas clustering histogram features are relatively more affected by noise. We made a confusion matrix of actual key exponents and clusters assigned to segments. Accuracy of the cluster is defined as $\text{Accuracy(Cluster)} = \frac{\sum_{i,j} C_{i,j}}{\sum_{i,j} C_{i,j}}$, where $C_{i,j}$ is element of the confusion matrix.

| | Idle | Cluster Idle | Cluster Square | Cluster Multiply |
|------------------------|-----|-------------|----------------|-----------------|
| Idle                   | $C_{1,1}$ | $C_{1,2}$ | $C_{1,3}$ |
| Square                 | $C_{2,1}$ | $C_{2,2}$ | $C_{2,3}$ |
| Multiply               | $C_{3,1}$ | $C_{3,2}$ | $C_{3,3}$ |

Table 3. Mean Accuracy of 100 runs.

| std(noise)/std(signal) | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 |
|------------------------|---|-----|-----|-----|-----|-----|-----|-----|-----|
| SNR(db)                | - | 13.98 | 7.96 | 4.44 | 1.94 | 0.0 | -1.58 | -2.91 | -4.07 |
| Polynomial coeffs      | 95.82 | 95.64 | 95.57 | 98.11 | 99.20 | 95.75 | 99.45 | 98.05 | 97.36 |
| Histogram              | 82.04 | 81.95 | 83.38 | 80.38 | 83.20 | 80.30 | 79.11 | 76.66 | 78.02 |

4.4. Recovering Key Exponent

Our approach should be feasible without the external information. That is, we should be able to distinguish more accurate runs of clustering from the others. Figure 9 shows the examples of desirable and undesirable runs of clustering. We sorted 100 DB indices in ascending order and picked 5, 10, and 15 lowest DB indices and corresponding clusters. Table 4 shows the the average accuracy of 5, 10 and 15 runs of the K-means algorithm. We can see huge improvement in most of the cases, especially in the histogram features. This means that even without the external information, we can distinguish the good and bad clusters and find the key exponents.

Table 4. Selecting desirable clusters with DB index.

| std(noise)/std(signal) | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 |
|------------------------|---|-----|-----|-----|-----|-----|-----|-----|-----|
| SNR(db)                | - | 13.98 | 7.96 | 4.44 | 1.94 | 0.0 | -1.58 | -2.91 | -4.07 |
| Polynomial coeffs - 5  | 93.18 | 100.00 | 100.00 | 98.18 | 97.73 | 95.45 | 97.73 | 91.36 | 96.36 |
| Polynomial coeffs - 10 | 94.77 | 100.00 | 100.00 | 99.09 | 98.86 | 97.73 | 97.73 | 90.91 | 93.18 |
| Polynomial coeffs - 15 | 96.52 | 100.00 | 100.00 | 99.39 | 99.24 | 98.48 | 97.73 | 93.03 | 91.97 |
| Histogram - 5          | 89.55 | 94.09 | 97.73 | 97.73 | 97.73 | 92.73 | 97.73 | 88.64 | 97.73 |
| Histogram - 10         | 93.64 | 95.91 | 97.73 | 97.73 | 97.73 | 95.23 | 98.86 | 93.18 | 97.73 |
| Histogram - 15         | 95.00 | 96.52 | 97.73 | 98.33 | 97.88 | 96.06 | 99.24 | 95.45 | 97.73 |
4.5. Data Scale

In this part, we empirically checked the time complexity of our approach. We have set the time series length \( T = \frac{N_p}{N_w} \) differently in each case. Then we checked each case 10 times and box plotted in Figure 10. It shows that time spent in each case is linear to the time series length \( T \). So if the raw data are in a larger sampling rate or a larger key bit size, our approach takes longer to that ratio.

5. Discussion

- For a longer key bit: If we can assume that the environment that generated the power trace is consistent during the whole time so that there exist certain patterns, we can apply our methods to the first part and extract some patterns from that part. For the rest, using the patterns we can find the key exponents faster. In this manner, we can solve the rest part of the problem in a supervised way, whereas our approach is a totally unsupervised way for finding a key. This will reduce the time spent on analysis even when the key length is relatively long.

- Weakness: One major weakness is that our approach is based on finding piece-wise constant parts. If the idle part changes drastically with a magnitude bigger than \( \sigma \) or the idle part has another specific shape, we shall adopt a different model.
Further application: the proposed approach of ours can be applied to other related applications. A discrete power system is one of the examples [26,27]. Problems of systems with different power generation models and different assumptions can be solved with the proposed approach.

6. Conclusions

In this work, we suggested a probabilistic model-based side channel analysis. We modelled a change point detection problem to detect piecewise constants which are not directly related to finding keys, and merged incomplete segments to key-relevant complete segments. We solved this problem with an MCMC approach to find a global optimal solution. From each segment, we extracted features of a fixed dimension and assigned a cluster to each segment. We showed that this cluster is highly related to key exponent of power trace and this approach consistently works even in the presence of the noise. We evaluated our approach with criteria of noise-level-robustness and accuracy of the key. The source code for our work is available at github.com/JeonghwanH/binEXP_CPD.

Author Contributions: Writing—original draft, J.H.; writing—review and editing, J.W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported as part of Military Crypto Research Center(UD170109ED) funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Power analysis attacks of modular exponentiation in smartcards. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, 12–13 August 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 144–157.
2. Witteman, M.F.; van Woudenberg, J.G.; Menarini, F. Defeating RSA multiply-always and message blinding countermeasures. In Proceedings of the Cryptographers’ Track at the RSA Conference, San Francisco, CA, USA, 14–18 February 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 77–88.
3. Picek, S.; Heuser, A.; Jovic, A.; Ludwig, S.A.; Guilley, S.; Jakobovic, D.; Mentens, N. Side-channel analysis and machine learning: A practical perspective. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 4095–4102.
4. Lerman, L.; Bonnetp, G.; Markowitch, O. Power analysis attack: An approach based on machine learning. IJACT 2014, 3, 97–115. [CrossRef]
5. Benadjila, R.; Prouff, E.; Struliu, R.; Cagli, E.; Dumas, C. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. In ANSSI, France & CEA, LETI, MINATEC Campus, France. 2018; Volume 22. Available online: https://eprint.iacr.org/2018/053.pdf (accessed on 20 February 2020).
6. Hettwer, B.; Gehrer, S.; Güneysu, T. Applications of machine learning techniques in side-channel attacks: A survey. J. Cryptogr. Eng. 2019, 1–28. [CrossRef]
7. Clavier, C.; Feix, B.; Gagnerot, G.; Roussellet, M.; Verneuil, V. Horizontal correlation analysis on exponentiation. In Proceedings of the International Conference on Information and Communications Security, Barcelona, Spain, 15–17 December 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 46–61.
8. Bauer, A.; Jaulmes, E.; Prouff, E.; Wild, J. Horizontal and vertical side-channel attacks against secure RSA implementations. In Proceedings of the Cryptographers’ Track at the RSA Conference, San Francisco, CA, USA, 25 February–1 March 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–17.
9. Bauer, A.; Jaulmes, É. Correlation analysis against protected SFM implementations of RSA. In Proceedings of the International Conference on Cryptology in India, Mumbai, India, 7–10 December 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 98–115.
10. Clavier, C.; Feix, B.; Gagnerot, G.; Giraud, C.; Roussellet, M.; Verneuil, V. ROSETTA for single trace analysis. In Proceedings of the International Conference on Cryptology in India, Kolkata, India, 9–12 December 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 140–155.
11. Walter, C.D. Sliding windows succumbs to Big Mac attack. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Paris, France, 14–16 May 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 286–299.
12. Specht, R.; Heyszl, J.; Kleinsteuber, M.; Sigl, G. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design, Berlin, Germany, 13–14 April 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–19.
13. Heyszl, J.; Ibing, A.; Mangard, S.; De Santis, F.; Sigl, G. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Berlin, Germany, 27–29 November 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 79–93.
14. Batina, L.; Gierlichs, B.; Lemke-Rust, K. Differential cluster analysis. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Lausanne, Switzerland, 6–9 September 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 112–127.
15. Nascimento, E.; Chmielewski, Ł. Applying horizontal clustering side-channel attacks on embedded ECC implementations. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Lugano, Switzerland, 13–15 November 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 213–231.
16. Perin, G.; Chmielewski, Ł. A semi-parametric approach for side-channel attacks on protected RSA implementations. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Bochum, Germany, 4–6 November 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 34–53.
17. Lavielle, M.; Moulines, E. Least-squares estimation of an unknown number of shifts in a time series. J. Time Ser. Anal. 2000, 21, 33–59. [CrossRef]
18. Green, P.J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 1995, 82, 711–732. [CrossRef]
19. Lavielle, M.; Lebarbier, E. An application of MCMC methods for the multiple change-points problem. Signal Process. 2001, 81, 39–53. [CrossRef]
20. Bishop, C.M. Pattern Recognition and Machine Learning; Springer Science+ Business Media: Berlin/Heidelberg,Germany, 2006.
21. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equation of state calculations by fast computing machines. J. Chem. Phys. 1953, 21, 1087–1092. [CrossRef]
22. Hastings, W.K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications; Oxford University Press: Oxford, UK, 1970.
23. Lavielle, M. Optimal segmentation of random processes. IEEE Trans. Signal Process. 1998, 46, 1365–1373. [CrossRef]
24. Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–9 January 2007; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2007; pp. 1027–1035.
25. Davies, D.L.; Bouldin, D.W. A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. 1979, PAMI-1, 224–227. [CrossRef]
26. Dassios, I.K.; Szajowski, K.J. A non-autonomous stochastic discrete time system with uniform disturbances. In Proceedings of the IFIP Conference on System Modeling and Optimization, Sophia Antipolis, France, 29 June–3 July 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 220–229.
27. Dassios, I.K.; Szajowski, K.J. Bayesian optimal control for a non-autonomous stochastic discrete time system. Appl. Math. Comput. 2016, 274, 556–564. [CrossRef]