Improving Differential Power Analysis by Elastic Alignment

Jasper G. J. van Woudenberg[1], Marc F. Witteman[1], Bram Bakker[2]

Riscure BV, 2628 XJ Delft, The Netherlands
{vanwoudenberg,witteman}@riscure.com
University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
bram@science.uva.nl

Abstract. To prevent smart card attacks using Differential Power Analysis (DPA), manufacturers commonly implement DPA countermeasures that create misalignment in power trace sets and decrease the effectiveness of DPA. We design and investigate the *elastic alignment* algorithm for non-linearly warping trace sets in order to align them. Elastic alignment uses FastDTW, originally a method for aligning speech utterances in speech recognition systems, to obtain so-called warp paths that can be used to perform alignment. We show on traces obtained from a smart card with random process interrupts that misalignment is reduced significantly, and that even under an unstable clock the algorithm is able to perform alignment.
Keywords — Differential Power Analysis, unstable clock, random process interrupts, Elastic Alignment, time series analysis, dynamic time warping.

1 Introduction

Modern smart cards are devices designed for secure operation in an environment outside the control of the issuer. Because of this, they must be protected against a wide range of attacks, including side channel analysis. A powerful and well-studied technique is differential power analysis (DPA, [Koch99]). DPA analyzes the statistics of power measurements on a device while it is performing (a part of) its security function. Repeated measurements are taken of the same process, and by relating the power usage and the data values being processed, secret data may be revealed.

Smart card manufacturers are aware of these issues, and implement various countermeasures to reduce the effectiveness of attacks such as DPA. Entirely preventing DPA is often very hard, but a card can be considered secure if the resources needed for breaking it outweigh the resources an attacker has available.

Countermeasures are typically aimed at breaking the assumptions that underlie known attacks. DPA assumes that the cryptographic operations take place at exactly the same time in each power measurement. By using an internal clock with varying frequency, or randomly inserting dummy wait states into the execution of an algorithm, it is no longer time constant. Moreover, the cryptographic

operations do not take place at the same instant but are shifted in time with respect to each other. Inducing this misalignment is currently one of the common countermeasures used in secure devices. Although it does not completely prevent DPA, it can make it very costly in terms of the number of traces that need to be processed.

In cryptographic implementations that do not actively induce timing differences as a countermeasure, misalignment is typically caused by inaccuracies in triggering the power measurements. This means that traces can be aligned by determining the duration of the timing inaccuracies, and shifting the traces accordingly. This process is called *static alignment* [Mang07].

In contrast, when a cryptographic implementation actively induces random time delays or varying clock frequencies, static shifting cannot fully align the traces. *Dynamic alignment* is a general term for algorithms that match parts of several traces at different offsets, and perform nonlinear resampling of the traces. This is done such that afterwards, these parts are located at the same offsets.

The starting point for our work is the observation that there are parallels between time series analysis, in particular speech recognition techniques, and alignment of power traces. This paper describes an alignment algorithm called *elastic alignment*, which is based on dynamic time warping, a well established algorithm for time series matching. We also show our algorithm is practically applicable and can be used to recover secret key leakage from misaligned traces.

1.1 Previous work

DPA research is focused on performing an analysis that is less sensitive to misalignment. This contrasts our proposed method, which can be considered a preprocessing step before performing DPA: it directly modifies individual traces such that they are aligned. This is especially useful for the class of template attacks, which are very powerful given a correctly learned leakage model [Char03]. As the method does not explicitly take into account misalignment, its classification can be improved if it is applied to n correctly aligned traces versus n misaligned traces. This can be of significant importance, as in practice n is bounded. Also, the number of interest points may be lower for a correctly aligned trace.

There are generally three ways of dealing with misalignment when performing DPA [Mang07]: running static alignment on just the DPA target area, preprocessing traces (integration of samples, convolutions or FFT), or simply running DPA on unmodified traces. Note that running DPA on unmodified traces requires a much larger trace set, which may be infeasible.

Integration and other preprocessing methods can yield good results in terms of increasing the DPA peak; however, the increase can be limited due to the spectral components of the traces and the choice of algorithm parameters[Mang07].

In [Char05], a method is described to align traces based on the wavelet transform for denoising, and simulated annealing for resynchronization. The wavelet transform has the property that each trace can be viewed at different resolutions. It is shown that by performing DPA at lower resolutions, high frequency

noise is reduced and the DPA peaks become stronger. In addition, the simulated annealing algorithm is used to optimize a resynchronization function that relates a pair of wavelet-transformed traces. This further improves the DPA peak.

Using wavelets for denoising is in our opinion a viable method; however, as we are purely considering an alignment algorithm, we do not wish to reduce the information in a trace.

In [Clav00] a method called *sliding window DPA* (SW-DPA) is introduced. SW-DPA targets random process interrupts (RPIs) by replacing each clock cycle in the original traces with an average of itself and a number of previous cycles. Its two parameters are the number of cycles to average, and the length of a cycle. Effectively, SW-DPA integrates leakage spread out over a few clock cycles back into one place, such that DPA peaks are restored.

There are two aspects of SW-DPA that need to be taken into account: the clock cycles are assumed be of fixed length, and the number of cycles to average must be specified carefully based on the number of RPIs. Our aim is to be able to deal with targets with an unstable clock as well as automatically overcome RPIs.

1.2 Organization of this paper

This paper is organized as follows. Section 2 describes the background and design of the dynamic time warping algorithm and its linear-time successor FastDTW. In Section 3 we first introduce elastic alignment as our approach to trace set alignment, and how it is based on the warp paths, a side effect of FastDTW. Next, we analyze the alignment performance in Section 4. Final conclusions and ideas for further work are given in Section 5, and supplementary information is present in Appendix A.

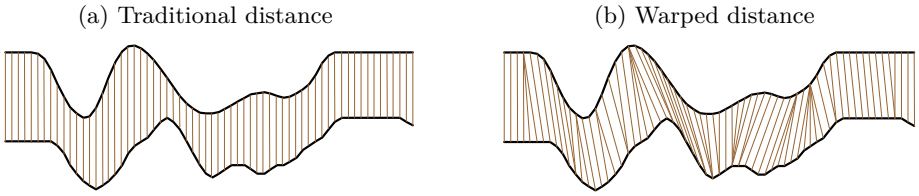
2 Dynamic Time Warping

The dynamic time warping (DTW) algorithm originates from speech recognition research [Sakh78]. Matching spoken words to a database containing prerecorded words is a nontrivial problem, as words are always spoken with variances in timing. Traditionally, calculating distances between two word utterances is performed by using a measure that compares recorded words sample by sample. These are based on, e.g., the sum of squared differences between the samples or the correlation of the sample values.

However, in cases where we have two similar utterances with differences in timing, the distance under such measures will be larger than if the utterances were 'aligned'. This follows from the property that these sample-by-sample measures do not explicitly consider unaligned utterances.

Being confronted with this problem, Sakoe et al. introduced a dynamic programming approach to match utterances using nonlinear time paths [Sakh78]. DTW measures the distance between two utterances by 'elastically' warping

Fig. 1: Dynamic time warping distance calculation (from [Chu02]).



them in time (see Figure 1), and then measuring the distance. Warping is performed based on the *warp path* the algorithm produces, which gives the alignment under which the signals have a minimum distance. DTW thereby allows utterances from processes with variable timing to be matched more accurately.

Traditionally, DTW is used for calculating a distance between two speech utterances. However, we are interested in trace alignment. We note that the warp path internally produced by DTW for measuring distances represents a matching between the time axes of two utterances. In this paper we use the same principle to align measured power traces from smart cards. Note that the DTW algorithm can only align two traces, so like other alignment algorithms we will be dependent on a reference trace.

The remainder of this section explains the original DTW algorithm, the improved FastDTW algorithm and how to apply the algorithm to trace pair alignment.

2.1 Obtaining the warp path

For our alignment we are interested in the *warp path.* The warp path is a list of indexes in both traces that represents which samples correspond to each other. Formally, if we have two traces X and Y, we define a warp path F

$$F = (c(1), c(2), \ldots, c(K)) \tag{1}$$

with $c(k) = (x(k), y(k))$ indexes in X and Y respectively. Figure 2 gives an example of a warp path.

There are several constraints on the warp path:

- **Monotonicity**: $x(k-1) \leq x(k)$ and $y(k-1) \leq y(k)$.
- **Continuity**: $x(k) - x(k-1) \leq 1$ and $y(k) - y(k-1) \leq 1$.
- **Boundary**: $x(1) = y(1) = 1$, $x(K) = T$ and $y(K) = T$, with T the number of samples in X and Y.

Fig. 2: Example warp path [Keog99] for traces X and Y. The warp path shows the optimal matching of the two traces by index pairs i and j.

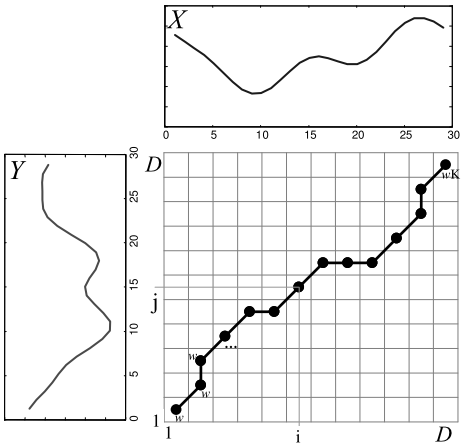
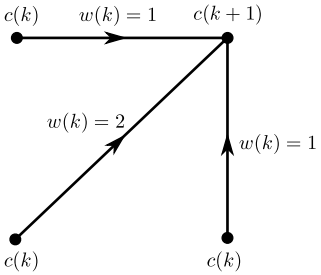


Fig. 3: Warp path steps and cost factor $w(k)$.



Combined these constraints restrict the warp path to three possible steps (see Figure 3):

$$c(k+1) = (x(k+1), y(k+1)) \tag{2}$$

$$= \begin{cases} (x(k), y(k)+1) & \text{or} \\ (x(k)+1, y(k)) & \text{or} \\ (x(k)+1, y(k)+1) \end{cases} \tag{3}$$

Furthermore, the length of the warp path can be deduced to be bounded by

$$T \leq K < 2T \tag{4}$$

The monotonicity and continuity constraints are a natural choice for our problem domain: we do not allow going back in time, nor skipping any samples. The choice for the boundary constraint is based on the assumption that traces start at the same phase, and end at the same phase of a measured process.

Starting at the same phase can usually be realized in side channel acquisition by timing the acquisition or static alignment. However, traces are usually of fixed length and, due to the introduction of time variations, do not necessarily end at the same phase of a process. This implies that matching the end of two traces can be difficult. DTW can overcome this by the inherent property that segments found in only one trace can be 'skipped' by matching the entire segment to a one or few samples in the other trace. Practical experience with the algorithm confirms the neither beginning nor the end of the traces need to be exactly aligned for DTW to find a good match.

Cost matrix In order to find the minimum cost warp path, the DTW algorithm calculates a *cost matrix d*. This matrix contains the distances between all samples of X and Y, and is calculated as

$$d(i,j) = |X[i] - Y[j]| \tag{5}$$

The length of a warp path F depends on how the distances between samples combined with the path through them translate into a distance between X and Y. The measure L giving the distance for the minimum length warp path is:

$$L(X,Y) = \frac{1}{2T} \min_{F} \left[\sum_{k=1}^{K} d(c(k))w(k) \right] \tag{6}$$

where $w(k)$ is a weighting factor and T the length of the traces. The weighting factor was introduced to construct a measure with flexible characteristic. We use the symmetric measure from [Sakh78], which implies $w(k)$ is the number of steps made in each dimension:

$$w(k) = [x(k) - x(k-1)] + [y(k) - y(k-1)] \tag{7}$$

So, if we make a step only in X or only in Y then $w(k) = 1$, for diagonal step $w(k) = 2$ (see Figure 3).

Finding the minimum cost path Having defined the distance $L(X,Y)$, we need an algorithm to find the minimum distance path. This corresponds to the optimized way of warping the two traces such that they are aligned.

A simple dynamic programming algorithm is implied by rewriting $L(X,Y)$ in recursive form:

$$g_1(c(1)) = d(c(1)) \cdot w(1) \tag{8}$$

$$g_k(c(k)) = \min_{c(k-1)} [g_{k-1}(c(k-1)) + d(c(k))w(k)] \tag{9}$$

$$L(X,Y) = \frac{1}{2T} g_K(c(K)) \tag{10}$$

If we only apply steps from Eq. (2), substitute for $w(k)$ (Eq. (7)), and drop the subscript k for simplicity, we obtain:

$$g(1,1) = 2d(1,1) \tag{11}$$

$$g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \end{bmatrix} \tag{12}$$

$$L(X,Y) = \frac{1}{2T} g(T,T) \tag{13}$$

The algorithm first calculates matrix d, and then starts at (T,T) to trace the minimum warp path according to Eq. (12). To avoid boundary problems, we define $d(0,j) = d(i,0) = \infty$. This procedure yields both the distance measure and the minimum distance warp path. An example calculation is found in Appendix A.

2.2 FastDTW: Iterative reduction and bounding

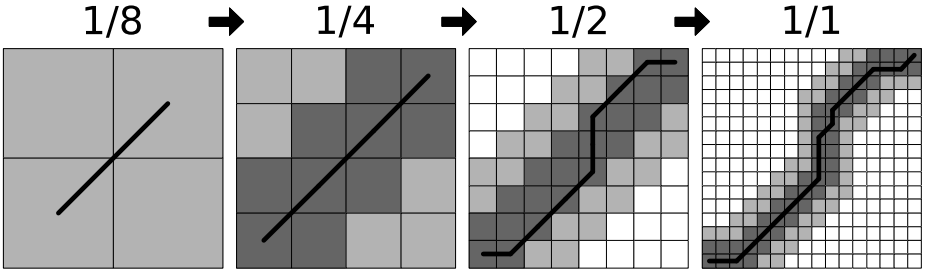
The time and space complexity of DTW can be restrictive in practice. Calculating $g(i,j)$ in Eq. (12) requires calculating $d(r,s)$ for all $1 \leq r \leq i$ and $1 \leq s \leq j$, and thus the complexity of calculating $g(T,T)$ is quadratic: $O(T^2)$. There are several approaches to overcome this problem which abstract the data or restrict the search space. We can constrain the search for an optimal warp path, but this should be based on knowledge (or assumptions) about the data. To balance the rigidity of constraints and algorithm performance, [Salv04] proposes the *FastDTW* algorithm.

The FastDTW algorithm restricts the warp path by bounding which elements of matrix d are calculated. This bounding is determined by the warp path at different resolutions of the traces, and, if the bounding is not too tight, produces the same results as DTW but with $O(T)$ complexity.

FastDTW uses a multilevel approach with three key operations. *Coarsening* reduces the size of a trace by averaging adjacent pairs of samples (see Figure 4). The resulting trace is a factor of two smaller than the original trace. Coarsening is performed several times to produce many different resolutions of the trace. At the lowest resolution, the original DTW algorithm is used to generate a warp path.

Projection takes a warp path calculated at a lower resolution and determines what cells the warp path passes through in the next higher resolution traces. This projected path is then used as a heuristic to bound the warp path at the higher resolution. *Refinement* takes this projected path and increases the bounds by a *radius*, which controls the space DTW can search beyond the projected path. Next, it finds the optimal warp path within these extended bounds by executing a bounded version of DTW. This bounded version of DTW can be understood as the original DTW, with all elements of d outside the search bounds set to ∞.

Fig. 4: Iterative DTW warp path refinement in FastDTW [Salv04]. Each iteration the resolution for both axes is doubled, and a new warp path is determined within the light gray bounds.



The difference between this method and the other approximations to full DTW is that the FastDTW algorithm does not rigidly determine the bounds a priori without consulting the data; the bounds are set based on the local shape of the warp path. This way a much closer approximation to the original DTW algorithm is achieved. The radius parameter affects the approximation: the higher the radius parameter, the higher the accuracy of representation of the original DTW algorithm is, but also the slower FastDTW runs. The radius parameter should not be set too high: if it is in the order of T the algorithm reduces to DTW speed, as the bounds always include all cells. Otherwise, [Salv04] shows FastDTW is in $O(T)$. In our experiments we need to find a reasonable value for this parameter.

3 Elastic alignment using FastDTW

In this section, we propose the *elastic alignment* algorithm. By using FastDTW on two traces, we obtain a sample-by-sample match of these traces. In order to align the traces, matching samples need to be projected onto new traces. This implies the warp path $F = (c(1), c(2), \ldots, c(K))$ between two traces X and Y gives rise to two projections onto aligned traces $\dot{X}$ and $\dot{Y}$.

Under the restriction not to increase the length of two aligned traces, we use the following asymmetric projections:

$$\dot{X}[i] = X[i] \tag{14}$$

$$\dot{Y}[j] = \frac{1}{|\{k \mid x(k) = j\}|} \sum_{x(k)=j} Y[y(k)] \tag{15}$$

with the minimal length warping path $c(k) = (x(k), y(k))$, $1 \leq k \leq K$, $1 \leq i \leq T$ and $1 \leq j \leq T$. These projections can be understood as elastically aligning Y to X by averaging and duplicating samples of Y based on the minimal length warping path. The length of the traces remains T.

Algorithm 1 Elastic Alignment

1. Obtain a reference trace X and a trace set $\mathcal{Y}$; all traces of length T
2. For each trace $Y \in \mathcal{Y}$
   (a) Calculate warp path $c(k) = (x(k), y(k))$, for X and Y using FastDTW
   (b) Calculate $\dot{Y}[j]$ for $1 \leq j \leq T$, output $\dot{Y}$

The projections as described are only capable of aligning pairs of traces. However, the chosen asymmetric projections allow for a reference trace to be chosen. This reference trace can be used as a basis to elastically align an entire trace set by aligning each trace to this reference trace, as described in Algorithm 1. Note this reference trace can be included in the trace set to align, but this is not a necessity.

The difference between this algorithm and FastDTW is that the latter focuses on distance calculation between two traces, producing a warp path as byproduct. Elastic alignment uses this warp path to resynchronize traces.

3.1 Computational complexity

The complexity of elastic alignment is per trace $O(T)$ for FastDTW, and $O(K)$ for the resynchronisation step. Because $T \leq K < 2T$, this makes the total complexity linear in the trace length and the number of traces: $O(T \cdot |\mathcal{Y}|)$.

FastDTW has a radius parameter that trades off computation time and time series matching optimality. For DPA this means it is possible to tune the alignment quality and the computation time. The radius has an optimum value at which increasing it does not affect the alignment, but only increases the computation time. This is the point at which the optimal warp path is fully contained within the radius for each FastDTW iteration.

As this optimal value depends on the characteristics of the traces under analysis, it is non-trivial to give a general formula for the exact value. By starting with a low radius, and continuously increasing it until the alignment of a pair of traces becomes stable, we usually find the optimal radius value lies between 100 and 150.

3.2 Usage for DPA

One of the more practical problems encountered when performing DPA, is that of misalignment due to unstable clocks and random process interrupts. Because of the continous trace matching, elastic alignment synchronizes to the reference clock and process. Unstable clocks are therefore automatically accounted for.

However, random process interrupts (RPIs), an active countermeasure introducing misalignment, can also be overcome. If the RPI is present in the reference trace, the other trace can be stretched in order to 'skip' the RPI. Conversely, if

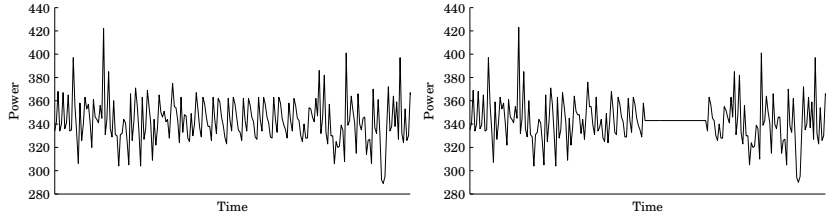


Fig. 5: Two traces with random length interrupt aligned by elastic alignment

the RPI is present in the other trace, it can be compressed down to a one or a few samples. An example of the effect of elastic alignment is shown in Figure 5.

Continuous synchronisation is important when a DPA attack on multiple serially executed targets is mounted. In e.g. an AES software implementation, there are 16 consecutive S-box lookups that need to be targeted. With elastic alignment, a trace set with an unstable clock needs to be aligned only once, whereas with static alignment 16 different alignments need to be performed.

3.3 Elastic alignment considerations

Although appointing one trace from a set as reference trace seems arbitrary, in practice reference traces (or parts thereof) are used for trace alignment. We have experimented with elastically aligning trace sets without a reference trace by using a hierarchical alignment scheme that iterates trace pair alignment using symmetric projections. We found that even though the alignment works, the (linearly bounded) increase in trace length implied by this method is less practical and does not necessarily outperform elastic alignment with a reference trace.

We choose not to dispose of samples, but to merely compress or stretch areas. This is because we cannot say which amount of local misalignment is actually an RPI that can be disposed of, and which misalignment is caused by an unstable clock or slightly differing instruction paths. By not disposing any samples, we decrease the possibility of removing interesting information.

4 Experiments

In our experiments we test to what degree elastic alignment increases the effectiveness of a power attack when countermeasures are present that induce misalignment: random process interrupts, and an unstable clock. We compare elastic alignment with sliding window DPA (SW-DPA,[Clav00]), a technique that targets RPIs.

We target a smart card on which we implement the first round of the DES cipher in software, and introduce random process interrupts. These interrupts randomly halt the card for 0 or 1 cycles, before each of the 8 S-box lookups. This process causes misalignment in the acquired traces.

Unstable clocks are typical for cards with an internal clock. Our sample does not have an internal clock, and we are unaware of programmable cards with an internal clock that are vulnerable to DPA within a few thousand traces. Therefore, we choose to process the obtained traces and introduce an unstable clock by duplicating or removing a sample at the end of only a small fraction of clock cycles. This is the second set of traces we will be analysing.

Note that in our set with a stable clock, SW-DPA can exactly average consecutive cycles. With the unstable clock, it is not possible for SW-DPA to correctly average clock cycles, as it assumes a fixed clock length. We therefore expect SW-DPA to perform better with the stable clock than with the unstable clock. Elastic alignment should be able to deal with both scenarios, as it automatically synchronizes to the clock.

4.1 Measuring DPA success rate

To analyze the results, we perform correlation power analysis (CPA, [Brie04]) on trace sets of different number of traces, and calculate the first order success rate graph [Stan08]. This graph displays for increasing trace set size what the estimated probability is of finding the correct key as first candidate.

We know our target implementation strongly leaks the Hamming weight of the processed intermediates. For CPA, we therefore use the Hamming weight power model and correlate with the output of the DES S-boxes. We are targeting the first round of DES for efficiency reasons. Because of this, the total key we recover has 48 bits. As elastic alignment and SW-DPA are signal processing techniques, there is no reason to assume they perform differently if more rounds or even another cryptographic algorithm is used.

4.2 Trace acquisition and processing

We acquire one set of 100000 traces. All traces are acquired by measuring the instantaneous power consumption of the card by an oscilloscope sampling at 50MHz, using an analog low pass filter at 11MHz. The clock of the card runs at 4MHz, and we compress the traces by averaging consecutive samples resulting in one sample per clock period. The number of samples per trace is 5600.

From this original trace set we generate two derived trace sets: one with a stable cycle, and one with an unstable cycle. From Fourier transforms of measurements on various cards with unstable clocks we know the clock to be strongly centered around its base frequency, with a sharp dropoff in both tails. This sharp dropoff indicates the instability to be small, and we therefore choose to create the derived trace sets such that the instability is small as well: the stable cycle has 5 samples per clock, and the unstable cycle length is determined by a rounded Gaussian distribution: $\lfloor L + 0.5 \rfloor, L \sim N(5, 0.2)$, which yields about 98.7% cycles of 5 samples, and only 1.3% cycles of 4 or 6 samples. Each cycle is represented by one sample with the measured value, followed by samples that are the average

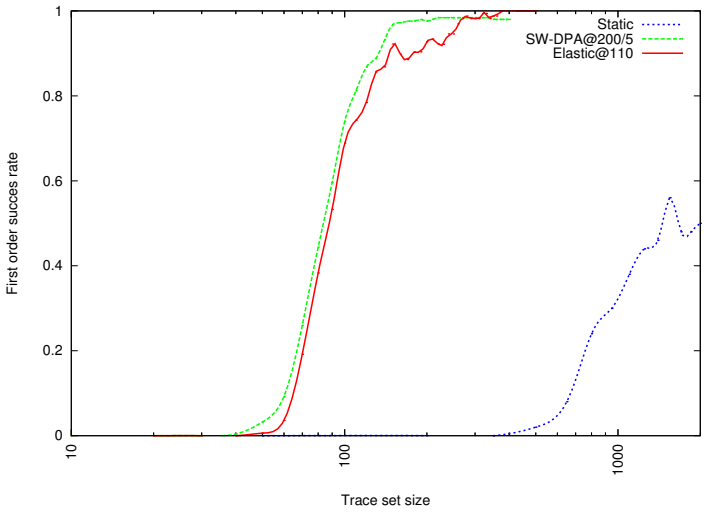


Fig. 6: CPA success rate for stable cycle length

of the previous sample and the minimum sample value of the trace. This corresponds with the observation that leakage is typically concentrated in a specific peak of the cycle.

For the experiments with elastic alignment, we set the radius parameter to 110. This is an experimentally derived value that balances computational performance and output quality. We start with a low value and increase it until it does not significantly improve the alignment quality. With this value for the radius parameter, aligning each trace takes about 2 seconds on a current 2.4GHz processor.

For SW-DPA, we choose to use the average length of a clock cycle in our measurements as the distance parameter, in these measurements 5 samples. The number of consecutive cycles to average is set to 200, which is the experimentally determined width of the distribution of the widest CPA peak (for the last S-Box in the calculation).

4.3 Results

For the experiments with fixed cycle length (Figure 6) we first see that the random process interrupts thwart DPA with static alignment: at about 1400 traces we only obtain a success rate around 0.5. For SW-DPA, we observe the effect of a perfect match between countermeasure and analysis technique: the averaging of fixed length clock cycles has restored the DPA peak in the face of random process interrupts. A success rate of close to 1 is already obtained at 160 traces.

Elastic alignment shows the same success rate around 270 traces. This is likely due to the fact that elastic alignment is an adaptive method, and noise may be

affecting the matching of trace sections. However, compared with ordinary DPA it is within the same order of magnitude as SW-DPA.

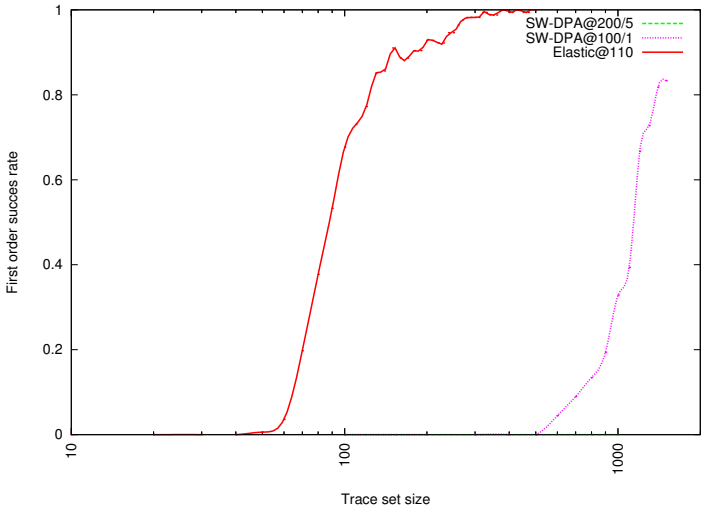


Fig. 7: CPA success rate for stable cycle length

The results become very different when unstable clocks are introduced. The success rate of SW-DPA, seen in Figure 7, goes to 0 for all experimented trace set sizes up to 1000. The same holds true for static alignment. In fact, we have attempted to perform DPA using this set at 100000 traces, and with SW-DPA the key could not be found; with static alignment it was found at around 67000 traces. So, even at 1.3% cycles with a different length than average, SW-DPA gets desynchronized and harms the correlation peak.

We have been able to get SW-DPA somewhat back by tweaking the parameters: setting the clock cycle length to 1 and setting the number of clocks to average to 100, we could get a 50% success rate at about 1150 traces, as seen in Figure 7. Note SW-DPA then acts more like a moving average filter: it does not have the 'comb' effect normally used to accumulate specific samples within a clock cycle.

Elastic alignment is by design able to overcome these desynchronized traces. The results shows that it is relatively unaffected by the unstable clock: the smallest trace set size with a success rate close to 1 increases marginally. Preliminary experiments show this also holds for a wider distribution of 2.5%, 47.5%, 47.5% and 2.5% cycles of length 2, 3, 4 and 5 respectively.

These experiments show that elastic alignment is able to deal with random process interrupts, and is very suited to dealing with unstable clocks due to its ability to continuously adapt alignment to the reference trace. This adaptativeness does imply it may also 'adapt' to noise that is present. We have some

ideas and preliminary experiments showing how to overcome this, as described in section 5.1.

5 Conclusions

In this paper we described and experimented with elastic alignment, an algorithm for aligning trace sets when misalignment is present. The elastic alignment is designed to be applicable practically in the context of performing side channel analysis in the presence of random process interrupts and unstable clocks.

We use FastDTW, a linear complexity variant of the dynamic time warping algorithm, for alignment. Dynamic time warping measures the distance between two traces and produces a warp path that describes a nonlinear time matching of the two. We use this warp path to align a pair of traces. By selecting one trace as a reference, we can thereby iterate this process to elastically align an entire trace set.

By design, elastic alignment attempts to globally optimize the alignment. This implies that at every resolution, be it process, instruction, clock or sub-clock, a good alignment can be found. For side channel analysis this is helpful as the traces may be analyzed at different resolutions.

The only parameter input to the algorithm is the FastDTW radius, which acts as a speed versus quality trade-off. This contrasts sliding window DPA, which requires knowledge of two target specific parameters: the clock cycle length, and the spread of the DPA peak over different clock cycles. Having fewer and easily selectable parameters makes the effectiveness of the side channel analysis less dependent on the user performing it.

Experiments were done based on traces obtained from a card with a fixed clock and the random process interrupt countermeasure enabled. These show sliding window DPA is moderately better than elastic alignment at dealing with only RPIs as countermeasure. This is probably due to noise affecting the dynamic adaptation of elastic alignment. However, as soon as even a slightly unstable clock is introduced, elastic alignment is much better at recovering the DPA peak due to its dynamic synchronization with the reference trace.

The experiments conform with our experiences on other cards, which are mostly implementations with hardware DES engines. Unfortunately we do not fully control these implementations, and they are therefore less suitable for structured experimentation. Elastic alignment also appears to work when traces are compressed down to the frequency of the internal clock, and, with proper signal integration, also on EM traces. In a number of cases, elastic alignment has played a key role in breaking a card using CPA within a bounded number of traces.

5.1 Discussion and future work

Besides the basic elastic alignment algorithm as presented in this paper, we have implemented a number of other experimental features. One is the possibility of decoupling warp path detection and application: we allow detecting the warp

paths on one set of traces, and applying them to another set of traces. If the other set of traces has a different number of samples, the warp path is scaled to accommodate this. This allows us to e.g. calculate an alignment at the level of one sample per clock, while repairing misalignment in traces with multiple samples per clock. The experiments and results are preliminary, but show interesting potential.

When performing elastic alignment, we implicitly violate the preconditions that the first and last samples of the trace pair are aligned. Elastic alignment can accommodate for this by matching the initial or final part of one trace to only a few samples in the other; however, we envision the alignment can be improved if both the first and last samples of the trace pair are aligned. This can be implemented by allowing variable length traces and using pattern matches to find the locations of the first and last samples.

A way to reduce the effect of noise on the alignment is to change the way FastDTW measures the distance between individual samples. Now this is done by their absolute difference, but one could consider taking more samples into account. This is accomplished by for instance using the weighted average distance of neighboring samples, or by correlation of trace fragments. This effectively 'smoothes' the distance function and potentially cancels some of the effects of noise in the individual samples.

6 Acknowledgments

We would like to thank Fred de Beer for helpful comments and discussions regarding elastic alignment. Also, we are grateful for the helpful comments about the previous version of this paper from the anonymous reviewers. We also thank Nick Towner for reviewing an earlier version this paper. Finally, we are indebted to Federico Menarini and Eloi Sanfelix Gonzalez for providing us with the trace sets used in the experiments.

References

Brie04. Eric Brier, Christophe Clavier, Francis Olivier *"Correlation Power Analysis with a Leakage Model"*, CHES 2004, Springer-Verlag, LNCS, 3156, pp. 16–29, 2004.

Char03. Suresh Chari, Josyula R. Rao and Pankaj Rohatgi, *"Template Attacks"*, CHES 2003, Springer-Verlag, LNCS, 2523, pp. 13–28, 2003.

Char05. Xavier Charvet, Herve Pelletier, *"Improving the DPA attack using Wavelet transform"*, NIST Physical Security Testing Workshop, 2005.

Chu02. S. Chu, E. Keogh, D. Hart and M. Pazzani, *"Iterative deepening dynamic time warping for time series"*, proceedings 2 SIAM International Conference on Data Mining, 2002.

Clav00. Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. *"Differential power analysis in the presence of hardware countermeasures"*, CHES 2000, Springer-Verlag, LNCS, 1965, pp. 252–263, 2000.

Keog99. Eamonn Keogh and M. Pazzani, "*Scaling up Dynamic Time Warping to Massive Datasets*", 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99), Vol. 1704, pp. 1–11, 1999.

Koch99. Paul Kocher, Joshua Jaffe and Benjamin Jun, *"Differential Power Analysis"*, Lecture Notes in Computer Science, Vol. 1666, pp. 388–397, 1999.

Mang07. Stefan Mangard, Elisabeth Oswald and Thomas Popp, "*Power Analysis Attacks: Revealing the Secrets of smart Cards*", Springer, 2007.

Sakh78. H. Sakoe, and S. Chiba, "*Dynamic programming algorithm optimization for spoken word recognition*", IEEE Trans. Acoustics, Speech, and Signal Processing., Volume 26, pp 143–165, 1978.

Salv04. S. Salvador and P. Chan. "*FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*", in Proc. KDD Workshop on Mining Temporal and Sequential Data, 2004. Java implementation: `http://cs.fit.edu/~pkc/FastDTW/FastDTW.zip`

Stan08. François-Xavier Standaert, Benedikt Gierlichs, Ingrid Verbauwhede, "Partition vs. Comparison Side-Channel Distinguishers", in proc. ICISC 2008, Springer-Verlag, LNCS, vol 5461, pp. 253–267, December 2008.

A DTW calculation example

Traces

X	1	1	2	3	4
Y	1	2	3	4	4

$d(i,j)$

	3	3	2	1	0
	3	3	2	1	0
$i \uparrow$	2	2	1	0	1
	1	1	0	1	2
	0	0	1	2	3

$j \rightarrow$

$g(i,j)$

	9	9	5	2	0
	6	6	3	1	0
$i \uparrow$	3	3	1	0	1
	1	1	0	1	3
	0	0	1	3	6

$j \rightarrow$

Warp path

$$F = ((1,1),(2,1),(3,2),(4,3)(5,4),(5,5))$$