Wide Collisions in Practice

Xin Ye and Thomas Eisenbarth

Department of Mathematical Sciences
Florida Atlantic University, Boca Raton, FL 33431, USA
{xye2,teisenba}@fau.edu

Abstract. Detecting collisions in the power consumption of a modern cryptographic engine is usually difficult due to small leakages and possible countermeasures. Wide collisions offer a much stronger leakage that significantly facilitates their detection. This is the first time wide collisions are exploited in a power analysis attack. In this work we introduce a collision detection method based on the detection of characteristic outliers. Detection results are compared to optimized subspace-based templates. We show that the outlier detection method, while not requiring a template building phase, is almost as effective in detecting collisions as the template-based approach.

1 Motivation

After several years of their exploration, side channel attacks remain a threat to embedded cryptographic implementations. Especially various power analysis attacks have been developed [8,9,10] and improved [1,7,12], as have countermeasures against them [11]. Power based collision attacks [13] are mostly disregarded because detection of collisions is usually more sensitive to noise and countermeasures than most other DPA attacks. However, the AES-specific wide collisions as described in [4] offer a huge advantage. Instead of trying to detect a single collision of a byte during an SubBytes operation of the AES algorithm, wide collisions result in a colliding column for a whole round in addition to two SubBytes byte collisions in the prior and anterior rounds. Hence, such collisions are much easier to detect due to increased leakage. This work is the first one to present results of applying wide collisions using power analysis.

Related Work. In 2004, Schramm *et al.* mounted the first collision attack on AES, pointing out the important fact that the collision attack significantly reduces the attack complexity by combining the analytical and side channel approach [13]. Bogdanov in 2007 [2] generalized the concept of internal collisions and improved the attack under the assumption that byte collisions are detectable. In the following year two multiple-differential methods – binary and ternary voting— were raised for collision detection in [6] and multiple-differential collision attacks MDCA were proposed in [3]. A differential cache-collision timing attack on AES was mounted on an ARM microprocessor in 2010 in [4]. This

F. Bao, P. Samarati, and J. Zhou (Eds.): ACNS 2012, LNCS 7341, pp. 329–343, 2012.

is a chosen plaintext attack based on the occurrence of wide collisions and its algebraic properties.

Another important approach, the template based attack, was firstly introduced in [8] in 2002. Archambeau *et al.* in 2006 [1] suggested a principal subspace-based template attack which overcomes the problem of selection of important points and specification of the minimal distance between points. Recently Bogdanov *et al.* [5] combined collision attack with divide-and-conquer attacks as DPA and template attack to further reduce the computational complexity.

Our Contribution. In this paper we propose a new practical method – the *outlier method* – for detecting wide collisions in AES with a high success probability. It is for the situation where the creation of templates is impossible. It has no requirement on any prior knowledge concerning the leakage model, but it requires knowledge of leaking time instances in the power traces. The attack is based on the assumption that two traces are more likely to form a wide collision pair if they are far away from the mean trace of all the measurements and, at the same time close to each other. The results of the outlier method attack are compared to the detection rates of the PCA based template attack proposed in [1]. The strength of the template-based approach lies in that the principal components are not computed from all the traces as a whole, but instead from the 256 bin average traces — each of which is the mean trace of a bin that has been trained. The idea is to magnify of Inter-Bins Variation through PCA. We extend this idea to an iterative PCA algorithm to make further separation amongst those bins that are still close to each other in a previous iteration.

The organization of this paper is as follows. In Section 2 we review the collision attacks, wide collisions and template attack. In Section 3 we describe the performed attacks. Section 3.1 gives a detailed description of the outlier method. In Section 3.2 the concepts of Inter-Bins Variation and Inner-Bin Variation are introduced. Sections 3.3 and 3.4 introduce the PCA based template attack followed by further improvements through repeated applications of PCA. In Section 4 we analyze the influencing factors for the outlier method and compare the results for the template based collision detection in conditions of reduced templates in the time domain, full templates in the time domain and the principal subspace.

2 Background

The following gives an overview on collision attacks in general and describes the properties of wide collisions. Furthermore, the template attack is introduced.

2.1 Collision Attack

In general, an internal collision in a cryptographic primitive occurs if some specific target function ϕ produces the same output value y for two different inputs x_1, x_2, that is, $\phi(x_1) = y = \phi(x_2)$. Internal collisions in AES were defined by [13] and generalized by [2]. Collisions occur in the output of the MixColumns

transformation in each round function of AES, which takes two different plaintexts (or internal states) as input and outputs the same byte value. For example, an internal collision at byte 0 in round 1 occurs for two plaintexts P, Q, where $P = (p_{ij})$, $Q = (q_{ij})$, $i, j = \{0, 1, 2, 3\}$ if

$$02 \cdot p'_{00} \oplus 03 \cdot p'_{10} \oplus 01 \cdot p'_{20} \oplus 01 \cdot p'_{30} = 02 \cdot q'_{00} \oplus 03 \cdot q'_{10} \oplus 01 \cdot q'_{20} \oplus 01 \cdot q'_{30}$$

where all the p'_{ij}, q'_{ij} refer to the byte values of the internal state before the MixColumns operation. Since each $p'_{i0} = S(k_{ii} \oplus p_{ii})$ and $q'_{i0} = S(k_{ii} \oplus q_{ii})$, the above equation contains information about a part of the key. The idea of side channel collision attack on AES is therefore to detect internal collisions from the side channel leakages and then to reduce the number of possible subkey candidates by making use of the above equations, as described in [13] and [2]. One of the main challenges is to get reliable detection of such collisions.

2.2 Wide Collisions

Wide collisions for AES are defined in [4]. They are a special case of internal collisions in the following way. Specific plaintexts are chosen to satisfy the condition that the bytes off the diagonal are pairwisely equal i.e. $p_{ij} = q_{ij}$ for all $i \neq j$. After such chosen plaintexts entering the encryption engine, one can track the byte of an internal collision occurring at the first round MixColumns. After the ShiftRows operation of the next round, the entire column where this collision byte was shifted to will collide. Therefore 4 more internal byte collisions can be observed after the second round MixColumns.

Consequently, this gives rise to one byte collision in Round 2 SubBytes and four additional byte collisions in Round 3 SubBytes, resulting in a total of five byte collisions. This phenomenon is referred to as *wide collision*. For example, if two plaintexts collide at byte 0 after Round 1 MixColumns ($p'_{00} = q'_{00}$), they will continue to collide in Round 2 SubBytes ($S(k'_{00} \oplus p'_{00}) = S(k'_{00} \oplus q'_{00})$). After ShiftRows, all the four bytes in the Column 0 are pairwisely equal and thus we get four additional collisions after Round 2 MixColumns ($p''_{i0} = q''_{i0}$, $i = 0, 1, 2, 3$), which will remain colliding in Round 3 SubBytes ($S(k''_{i0} \oplus p''_{i0}) = S(k''_{i0} \oplus q''_{i0})$).

The wide collision attack is described as a three stage algorithm [4]: an online stage where side channel leakage is measured for the chosen plaintexts, a collision detection stage which returns several pairs of plaintexts which most likely give wide collisions, and finally a key recovery stage. It is mentioned that every 4 wide collisions for each of the diagonally chosen plaintexts sufficiently reduce the subkey space with a remaining uncertainty of 2^8. Hence, a total of 16 correctly detected collisions reduces the number of remaining key candidates to 2^{32}, which can be exhaustively searched on an average PC within minutes.

2.3 Template Attack

Template attack is a powerful side channel attack.In [8] power traces are characterized with a multivariate Gaussian distribution model. The attack assumes

that each individual trace follows a multivariate normal distribution with the center and covariance that are identical to the ones of the correct bin. In other words, traces of the same bin form a Gaussian distribution. Therefore each bin can be characterized by a template $(\mathbf{m}, C)$, where $\mathbf{m}$ refers to the mean trace and the C is the covariance matrix.

A template attack consists of template building phase and template matching phase. In the building phase, one characterizes the device with a set of templates $(\mathbf{m}_i, C_i)$, each of which is created for one bin of traces. In the matching phase each analyzed trace $\mathbf{x}$ is matched to the template that it most likely belongs to. That is, it outputs the template which gives the highest probability density computed by

$$Pr\left(\mathbf{x}; (\mathbf{m}_i, C_i)\right) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{m}_i\right)^T C_i^{-1}\left(\mathbf{x}-\mathbf{m}_i\right)\right)}{\sqrt{(2\pi)^N \det\left(C_i\right)}}$$

where N is the length of $\mathbf{m}_i$, the number of points in the mean trace. This is called the full template matching.

A simplified approach is referred as the reduced template matching in which the dependency amongst different data points are disregarded. That is, for each template the covariance matrix is replaced with the identity matrix so that the computation of the probability density is simplified as

$$Pr\left(\mathbf{x}; \mathbf{m}_i\right) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{m}_i\right)^T \left(\mathbf{x}-\mathbf{m}_i\right)\right)}{\sqrt{(2\pi)^N}}$$

Such reduced model is equivalent to computing the Euclidean distance between each analyzed trace $\mathbf{x}$ and the bin average traces $\mathbf{m}_i$ of the templates. It determines as the correct matching the template $\mathbf{m}_j$ that is nearest to the analyzed trace $\mathbf{x}$, i.e. $\parallel \mathbf{x}-\mathbf{m}_j \parallel \leq \parallel \mathbf{x}-\mathbf{m}_i \parallel$, for all i.

3 Practical Collision Attacks

In classical collision attacks, bins are formed in the way that certain bytes of the intermediate state of the cipher collide to the same value. When the same value is processed by certain operations (e.g. MixColumns, SubBytes in AES) of the cryptographic primitive, the pattern of the power consumption of these operations should be highly similar. Consequently, all possible 256 values of a given byte give rise to 256 bins, hence 256 different patterns for each colliding byte position. In a wide collision scenario, traces of each bin provide at least 5 internal byte collisions, spanning from MixColumns in round one up to SubBytes in round three of an AES execution. This results in a highly similar power consumption over a relatively long time period, especially for serial implementations. Hence, detecting wide collisions should be much easier than detecting simple collisions.

Conventionally, leaking points of the power traces refer to a subset of samplings of measurement or its transformation which represents the characteristics of the pattern of the power traces. Locating these points usually requires knowledge of the implementation and leakage properties of the platform or profiling. As an example, the two plots in Figure 1 show a single trace (the upper plot) over the region from round one MixColumns to round three SubBytes and a differential trace (the lower plot) calculated as the absolute value of difference between an average of traces of bin 7 and the average of all traces obtained. It can be seen from this figure that peaks, which indicate the location of promising leaking points, are spread out all over this region.

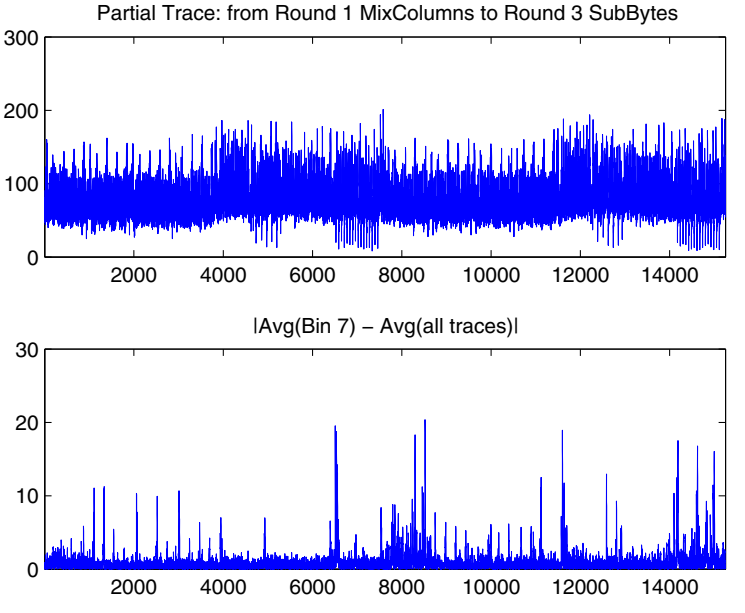

Fig. 1. Important Point Distributions

Another observation of leakages in the wide collision attack is that the positions of peaks are not invariant with respect to all wide collision bins. Some bins share one or more positions of leakages, while no pair of two different bins follows an identical pattern. Figure 2 gives an intuitive idea of the distribution of leakages for some bins.

Hence, picking only a single point from all important points for the purpose of collision detection is rather risky because only few bins leak at this point. In other words, if the closeness of two traces at one fixed point is the only criterion for wide collision detection, the detection can only succeed in rare cases because power traces of other bins that do not leak locally at this point are dominantly influenced by noise. Such traces make it difficult for a correct

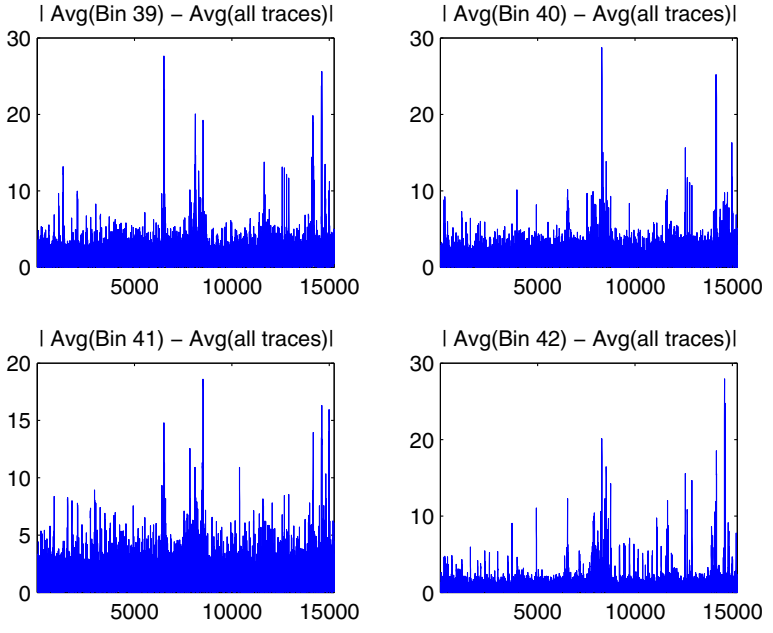

Fig. 2. Important Points for different Bins

collision detection, since, while randomly scattered, they are far more numerous than colliding traces.

We distinguish two different kinds of multiple-point-based approaches: One builds on a template-based detection, the other does not require templates. When generating templates is not possible, we propose an outlier method which assumes that a pair of traces forms a collision in the case they are close to each other and simultaneously far away from the average of all traces. As a comparison, we show that wide collisions can easily be detected using templates. The challenge in this case is to discover the characteristics of each pattern and to correctly recognize each individual power trace from all the patterns with high probability.

Furthermore, we introduce the concepts of Inner-Bin-Variation and Inter-Bins-Variation as two parameters determining the effect of collision detection. We propose methods with the application of Principal Component Analysis (PCA) and iterative PCA so that the idea of maximizing Inter-Bins-Variation is realized.

3.1 Outlier Method

Generally, the outlier method assumes that two traces in the outlier region – with distance sufficiently far away from the average of all traces – are more likely to form a collision pair if additionally they are sufficiently close to each other. It

includes one distance function $dist\,(\mathbf{x}, \mathbf{y})$ that gives a distance metric between two trace representatives $\mathbf{x}$ and $\mathbf{y}$. It also includes two distance parameters R and r, where R is the outlier lower bound ratio determining if one trace is inside the outlier region, and r is the mutual distance upper bound ratio determining if two traces are close by enough. Both R and r should be a number between 0 and 1. The procedure of the outlier method as follows:

Step 1: Use $\left(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(D)}\right)$ as the representation of D traces collected in the online stage and compute the average $\bar{\mathbf{x}}$ of all trace representation $\mathbf{x}^{(i)}$ at this point by $\bar{\mathbf{x}} = \sum \mathbf{x}^{(i)}/D$

Step 2: Compute the distances vector $\mathbf{d} = \left(d^{(1)}, ..., d^{(D)}\right)$ where each entry $d^{(i)} = dist\left(\bar{\mathbf{x}}, \mathbf{x}^{(i)}\right)$ is the distance between the trace $\mathbf{x}^{(i)}$ and the average trace $\bar{\mathbf{x}}$. Find the maximum element of the vector $\mathbf{d}$ by $maxd = \max(\mathbf{d})$.

Step 3: Find the set A of outliers by

$$A = \left\{\mathbf{x}^{(i)} \mid d^{(i)} \geq R \cdot maxd\right\}$$

It is a collection of trace representations with distance of no less than $R{\cdot}maxd$ from the average trace $\bar{\mathbf{x}}$. Figure 3 gives an example of the location of the outlier region.

Step 4: Find the list of pairwise distance

$$B = \left\{d_{ij} = dist\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \text{ where } \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in A, i \neq j\right\}$$

Note that if there are n outliers in set A, then the set B contains distances of $n\,(n-1)\,/2$ pairs of traces.

Step 5: Find the set C by

$$C = \left\{\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \mid d_{ij} \leq r \cdot maxd\ ,\ \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in A\right\}$$

This is a filteration from the set B of those pairs with mutual distance greater than $r{\cdot}maxd$. The set C is the output of the outlier method containing pairs of traces that are promising candidates for collisions.

Please note that in general the distance function $dist\,(\cdot)$ is a combination of all components. In practice, one could use a Euclidean distance for the *dist* function, viewing the trace representatives as elements in a finite dimensional vector space and assuming that components are independent from one another and each component contributes equally to the resulting distance.

Pros and Cons. The existence of leaking points is a necessary prerequisite for the wide collision detection. The points depend on the target device and implementation and should be chosen wisely by the attacker. Usually, either prior knowledge about the implementation or a profiling phase is needed. For detecting significant leakage points, SPA or DPA methods can be applied.

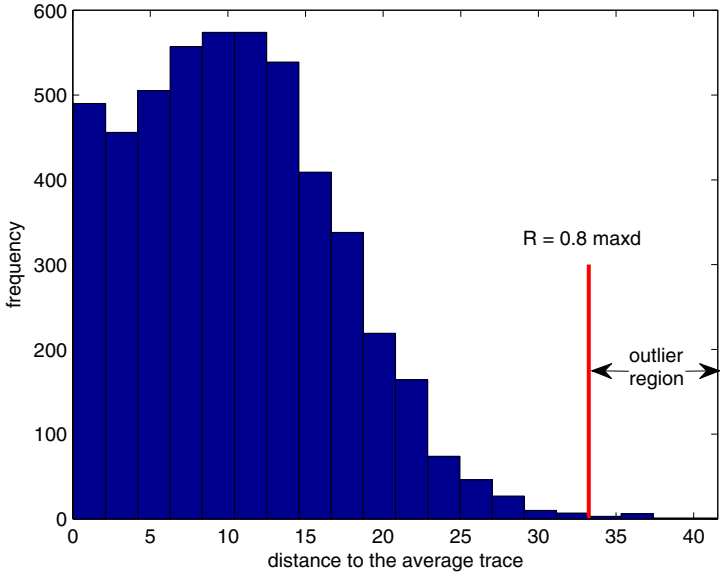

Fig. 3. outlier detection step 3

Choice of the parameters R and r is subjective. In practice, decreasing R while increasing r will eventually result in more non-colliding pairs that are detected as collisions. On the other side, decreasing r and increasing R will eventually increase to the set of detected collisions that does not contain enough pairs for key recovery.

The Euclidean distance is a convenient metric, since it is a straightforward combination of the influence of each leaking point. However, it is often weaker than a template attack on the same points. Using Euclidean distance makes two additional non-justified assumptions: that the points leak independently of one another and that they contribute equally to the output distance. For improved detection results, it can be replaced by some function $g(\cdot)$ such that the influence of different points can be more accurately reflected.

3.2 Inter-Bins Variation and Inner-Bin Variation

One way of selecting significant leakage points for collision detection is described in [5] and [6]. Both publications describe the *maxmin* function for selecting the most informative point of power traces and computing characteristics from traces at this point. Specific speaking, they first find for each fixed time point the lowest signal difference between all pairs of traces. They then find the time point that gives rise to the biggest one among all the lowest signal differences and consider that point to be the best choice. The logic of this method is that the lowest signal difference determines the level of difficulty of trace separtion at each time

point. The larger such lowest signal difference, the easier the separation of traces. However, this method makes use of a single point of the power trace, while ignoring all other remaining points. It yields for the attack a strong reliance on the single point that has been selected, which still suffers the risk of being influenced by the signal noise for each individual collision detection.

In contrast to single point selection, we propose an improved method. First, one should only include the leaking part of the power traces, i.e. the targeted round. For example, in wide collision attack, we only analyze the region starting from round 1 MixColumns to round 3 SubBytes. In this situation, two parameters – *inter-bins variation* (ITV) and *inner-bin variation* (INV) – determine the ease and probability of correct collision detection.

ITV describes the variation of the characteristics of the averaged traces $\mathbf{m}_i$ of each bin value. It is computed as a Euclidean distance as

$$\mathrm{ITV} = \sqrt{\sum_i \left(\mathbf{m}_i - \bar{\mathbf{m}}\right)^2}$$

where $\bar{\mathbf{m}} = \sum \mathbf{m}_i / 256$ is the mean trace of all the bin average traces. An increased ITV indicates an easier separation of the bins and more accurate pattern matching of each individual trace. Notice that the computation of ITV can also be applied to any representation of the traces. We refer the notion of maximizing ITV as computing the maximal ITV amongst all representations of the traces. Maximizing the ITV is therefore desired for the successful collision detection.

INV describes the variation of the characteristics of each individual trace $\mathbf{x}_j^{(i)}$ from that of the averaged trace $\mathbf{m}_i = \mathrm{avg}_j\left(\mathbf{x}_j^{(i)}\right)$ of a particular bin B_i. That is

$$\mathrm{INV}\,(i) = \sqrt{\sum_j \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i\right)^2}$$

INV can similarly be applied to any representation of the traces and we refer minimizing INV as computing the minimal of INV amongst all representations of the traces. Note that INV is a tuple of 256 entries, corresponding to the inner-bin variation of 256 bins, and minimizing one entry does not imply small values for the rest of entries. Hence, although minimizing INV is desired, it might not be practically feasible.

One should note that the existence of the bin average traces does not necessarily guarantee the feasibility of its computation. In fact, only if one can build up templates for the 256 bins, one can also obtain a raw representation of the average traces of bins. Since each representation gives a computational result for the ITV, the adversary would profit from a representation that maximizes the ITV.

In our experience, we realize the magnification of ITV through finding the representation of the average traces in the principal subspace, which are detailed in 3.3 and 3.4.

3.3 Template-Based Collision Detection

In cases where creating templates is possible, the attacker can build a template for each bin of collision in the time domain, as detailed in [11] and described in Section 2.3. Point selection can be automated by using *principal component analysis* (PCA) [1]. Template-based collision detection can achieve good detection rates, as shown in Section 4.2. PCA is a three step algorithm:

1. Finding the mean vector $\bar{\mathbf{x}}$ and the centered data matrix $\mathbf{X}' = (\mathbf{x}'_1, ..., \mathbf{x}'_D)^T$ of all the raw data record $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_D)^T$, where $\mathbf{x}'_i = \mathbf{x}_i - \bar{\mathbf{x}}$;
2. Computing the covariance matrix $\mathbf{S} = \frac{1}{D}(\mathbf{X}')^T(\mathbf{X}')$ and its d eigenvectors $(\mathbf{v}_1, ..., \mathbf{v}_d)$ corresponding to the largest d eigenvalues $(\lambda_1, ..., \lambda_d)$ of $\mathbf{S}$;
3. Projecting $\mathbf{X}$ into the subspace spanned by the d eigenvectors (also called components) $\mathbf{Y} = \mathbf{X}(\mathbf{v}_1, ..., \mathbf{v}_d)$.

PCA performs an orthogonal projection into a subspace called principal subspace. The projection maximizes the variance of the data. Hence, a point selection with minimal information loss becomes possible.

Constructing templates in principal subspace is only one additional step to the build-up of the templates in the time domain. That is, the raw traces need to be projected into the principal subspace before the construction of templates. For this step, the principal components could be obtained in two ways: from all the raw traces $\mathbf{X}$, or from the bin average matrix $\mathbf{M} = (\mathbf{m}_0, ..., \mathbf{m}_{255})^{\mathbf{T}}$, consisting bin average traces $\mathbf{m}_i$ of bin B_i where $\mathbf{m}_i = \text{avg}\{\mathbf{x}_j \mid \mathbf{x}_j \in B_i\}$. The consequence of applying PCA for the first choice is the maximization of the variance amongst individual traces and for the second approach amongst bin average traces. It is clear that the second method —computing principal components from bin averages— is desired because it achieves the goal of maximizing the ITV. After getting the projected traces in the principal subspace, the regular template building — the computation of bin averages and bin covariance — is performed as discussed in Section 2.3.

Finally, in the template matching phase, each analyzed trace is firstly projected onto principal components, then matched to the closest template through the evaluation of the probability densities. Every two different analyzed traces that are matched to the same template form a collision pair.

3.4 Template-Based Collision Detection Using Iterative PCA

A further improvement can be achieved by repeatedly conducting PCA. This gives rise to an iterative algorithm using projection. That is, in the template building phase, if two bins are too close or overlapping after the first PCA projection, one can repeat PCA projection (for which the computation of components only involves the average traces of that two bins) to further separate those two bins. The algorithm is given as follows:

Step 1: Use $\mathbf{M}^{(1)} = \{\overline{\mathbf{m}}_0, ..., \overline{\mathbf{m}}_{255}\}$ to compute the first set of principal components $\mathbf{V}^{(1)} = (\mathbf{v}_1, ..., \mathbf{v}_r)$ and the projection of each trace $\mathbf{P}^{(1)} = \mathbf{X} \cdot \mathbf{V}^{(1)}$.

Step 2: Partition the 256 bins into $C_\alpha^{(1)}$ and $C_\beta^{(1)}$ where bins from $C_\alpha^{(1)}$ can be clearly separated from other bins, while bins from $C_\beta^{(1)}$ are still clustered with some other bins. That is, if bin $B_i \in C_\beta^{(1)}$, then there exists bin $B_j \in C_\beta^{(1)}$ such that traces of bin B_i are not separable from traces in B_j.

Step 3: If $C_\beta^{(1)}$ is not empty, compute the set $\mathbf{M}^{(2)}$ which consists of the averages of projected traces of non-separable bins in $C_\beta^{(1)}$

$$\mathbf{M}^{(2)} = \left\{ \overline{\mathbf{m}}_i = \mathrm{avg}_j \left\{ \mathbf{p}_j \mid \mathbf{p}_j \in B_i \right\} \mid B_i \in C_\beta^{(1)} \right\}$$

Then based on $\mathbf{M}^{(2)}$, compute a second set of principal components $\mathbf{V}^{(2)}$ and obtain another set of projected traces $\mathbf{P}^{(2)} = \mathbf{P}^{(1)} \cdot \mathbf{V}^{(2)}$ from the previously projected ones.

Step 4: Repeat steps 2 and 3 until after k iterations $C_\beta^{(k)}$ is empty so that all bins are sufficiently separated.

4 Experimental Results

All experiments have been performed on a smart card featuring an 8-bit microcontroller based software implementation of AES. The measurements have been performed using a Tektronix digital sampling oscilloscope with an 8 bit A/D converter. The sampling rate of 50MS/s provides about 12 sampling points per clock cycle.

We first evaluate the detection rate of the outlier method. We explore the impact on the detection rate of several parameters: the number of promising leaking points, the choices of the outlier lower bound ratio R and the mutual distance upper bound ratio r, as well as the number of traces being investigated. Detection results are shown in Tables 1 through 3. The first column contains the analyzed influencing factor. The second column is the average size of set A, i.e. the average number of the outliers, as described in Section 3.1. Again, if n traces were in the outlier region, $n\,(n-1)\,/2$ pairs are further analyzed by computing pairwise closeness. The third column shows the size of the set C, i.e. the average number of output pairs of promising collisions. The next column counts the number of correctly detected collisions, that is the detected pairs which actually form wide collisions. The last column is the ratio between the third and the fourth column, i.e. the ratio of correctly detected collisions.

For comparison we apply template-based collision detection in three different scenarios, (1) reduced templates in the time domain, (2) full template in time domain and (3) full template in the principal subspaces. Figure 4 shows how many traces per bin are necessary for training good templates and further indicates the asymptotic recognition rate for the three cases for our platform.

4.1 Results for the Outlier Method

We apply the outlier method as detailed in Section 3.1 to detect wide collision from the power traces. In our experiment, we define the distance function with the norm $\|\cdot\|_1$. That is,

$$dist\,(\mathbf{x},\mathbf{y}) = \| \mathbf{x}-\mathbf{y} \|_1 = \sum_i |x_i - y_i|$$

gives the distance from trace $\mathbf{x} = (x_1, ..., x_t)$ to $\mathbf{y} = (y_1, ..., y_t)$. In our experiments we use 3000 traces and fix the parameter R to 0.9 and r to 0.3. We locate between 1 to 8 promising leaking points to analyze the influence of the combination of leaking points. Table 1 confirms that using multiple points results in a better collision detection rate comparing to the case of locating only one important point per power trace.

Table 1. Collision Detection: Single point vs. multiple points

number of leaking points	number of outliers	number of detected pairs	number of correct detection	average rate of correct detection
1	19.6	127.7	21.9	23.0%
4	30.6	46.3	33.4	71.1%
6	110.7	126.3	105.4	86.2%
8	81.7	88.1	82.3	93.7%

Next, we explore different choices of the parameter pair (R, r) to analyze the effect on the collision detection rate. As explained in the algorithm in Section 3.1, R is the parameter determining which traces are in the region of "outliers" (the set A), sufficiently far away from the center of all traces. The larger R is, the fewer traces are considered as outliers. On the other hand r is the parameter that determines if two outliers are close enough to each other. The smaller r is, the fewer pairs of traces are detected as collisions, namely the smaller the cardinality of the set C . Our experiments use 3000 traces (the same as above) and fix 6 locations of promising leaking points. They confirm that the stricter the choice of (R, r), i.e. the larger choice of R and the smaller choice of r, the more accurate the detection is, as shown in Table 2. As a last analysis of the

Table 2. Collision Detection: Choices of R and r

choice of (R, r)	number of outliers	number of detected pairs	number of correct detection	average rate of correct detection
$(0.7, 0.2)$	382.1	807	551	68.4%
$(0.8, 0.2)$	110.7	126.3	105.4	86.2%
$(0.9, 0.2)$	19.9	8.3	7.7	89.6%
$(0.9, 0.3)$	19.9	16.1	12.9	81.3%
$(0.9, 0.4)$	19.9	22.9	13.9	60.8%

outlier method, we explore the relationship between the successful detection and the number of traces being used in the experiment. In this experiment, 6 leaking points are fixed, the parameter R is set to 0.8 and r is 0.2. It is found

that increasing the number of traces yields the increase in the number of outlier traces and the number of pairs being detected, meanwhile the detection rate does not significantly increase, as shown in Table 3.

Table 3. Collision Detection: Impact of number of traces used

number of traces	number of outliers	number of detected pairs	number of correct detection	average rate of correct detection
1000	37.1	13.4	12.1	93.6%
3000	81.7	88.1	82.3	93.7%
5000	118.7	217.1	200.1	93.7%
7000	127.3	277	256.9	94.3%

4.2 Results for Template-Based Detection

In the preceding outlier method, it is assumed that one can locate several leaking points and these points are independent of each other and contribute equally to the computation of distance function. The same assumptions hold if a reduced template attack is mounted in the time domain. But if a full template attack is applied, these assumptions do not need to be fulfilled. If the templates are built in the time domain, one only needs to locate good leaking points. While templates built in principal subspaces, as described in Section 3.3, even locating leaking points is no longer necessary. This is because the attacker can make use of all the region of power traces that corresponds to wide collisions operations. Our experiment compares the method using reduced template and the full template to see how the dependency amongst leaking points helps with assigning an analyzed trace to its collision bin. We also compare the recognition rate between templates in the time domain and in the principal subspace through which we can verify that magnifying ITV enhances the recognition rate.

Our experiments use 8 leaking points in the time domain. The counterpart in PCA is 8 principal components that correspond to the 8 largest eigenvalues computed as specified in Section 3.4. We use 2560 to 5632 traces to build up templates so that each template makes use of 10 to 22 traces. We test 1000 traces to match to the templates and interpret #(*correct* recognition)/1000 as the recognition rate. From Figure 4 we can draw the following conclusions:

1. Using reduced templates in the time domain gives very stable recognition rate for a wide range of the number of used training traces for the templates. The asymptotic rate is at approx. 0.8. This is lower than the result for the two full models when sufficient training traces are provided. Therefore, the assumption of independence of leaking points cannot provide a strong collision distinguisher.
2. Full templates with PCA gain better recognition results comparing to the full templates in the time domain. Full templates in principal subspace gain close to 0.95 recognition rate given more than 20 training traces per bin. While

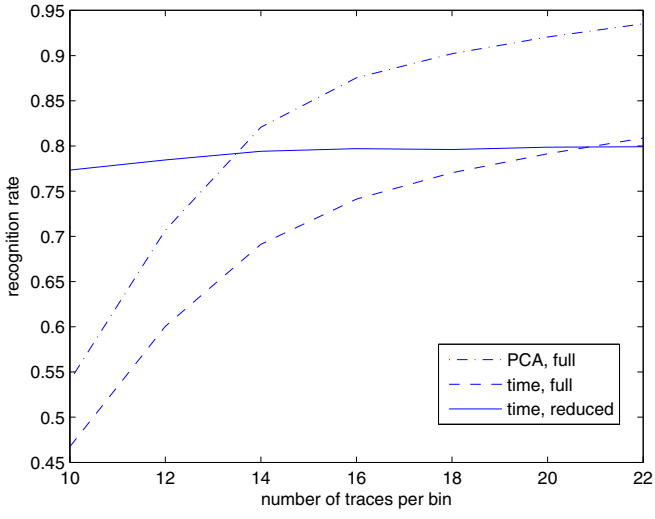

Fig. 4. Templates in the time domain and in the principal subspace

full model in the time domain achieves only around 0.8 to 0.85. This confirms the contribution of PCA for maximizing ITV in terms of recognition.

3. Full models have stricter requirements on the number of training traces. In particular, if the attacker chooses n leaking points in the time domain or n principal components in the principal subspace, then the number of training traces per bin cannot be less than n, otherwise a singular covariance matrix C is an unavoidable result and this makes the computation of probability density infeasible. Even when the least number of training traces is satisfied, the computation of the covariance matrix can still be remarkably impacted by the noise in the side channel. That is why the recognition rate for both of the full templates is low when fewer than 14 traces per bin are used.

5 Conclusion

This paper presents the first wide collision based power analysis attack. One major finding is the outlier collision detection method. It shows that power traces that are close to each other in the outlier region have a highly increased chance of forming a collision pair. It is a strong method for detecting wide collisions. Unlike template attacks, it does not require extensive profiling. However, it is shown that template-based approaches are a great method for detecting collisions if template building is possible. Different ways of building collision-detecting templates are compared. Using PCA for finding independent strong leaking points seems to be better than hand-picking points in the time domain. However, sufficiently many measurements for each bin must be available during template building.

Acknowledgment. The work described in this paper has been supported in part by the National Science Foundation under Grant No. 1054776.

References

1. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
2. Bogdanov, A.: Improved Side-Channel Collision Attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
3. Bogdanov, A.: Multiple-Differential Side-Channel Collision Attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008)
4. Bogdanov, A., Eisenbarth, T., Paar, C., Wienecke, M.: Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 235–251. Springer, Heidelberg (2010)
5. Bogdanov, A., Kizhvatov, I.: Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. IEEE Transactions on Computers PP(99), 1 (2011)
6. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)
7. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
8. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
9. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smartcards. Springer (2007)
12. Rechberger, C., Oswald, E.: Practical Template Attacks. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 440–456. Springer, Heidelberg (2005)
13. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)