A Collision-Attack on AES Combining Side Channel- and Differential-Attack

Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar

Horst Görtz Institute for IT Security
Ruhr-Universität Bochum, Germany
Universitätsstrasse 150
44780 Bochum, Germany
{schramm, cpaar}@crypto.rub.de,
{leander, felke}@itsc.rub.de,
WWW homepage: www.crypto.rub.de

Abstract. Recently a new class of collision attacks which was originally suggested by Hans Dobbertin has been introduced. These attacks use side channel analysis to detect internal collisions and are generally not restricted to a particular cryptographic algorithm. As an example, a collision attack against DES was proposed which combines internal collisions with side channel information leakage. It had not been obvious, however, how this attack applies to non-Feistel ciphers with bijective S-boxes such as the Advanced Encryption Standard (AES). This contribution takes the same basic ideas and develops new optimized attacks against AES. Our major finding is that the new combined analytical and side channel approach reduces the attack effort compared to all other known side channel attacks. We develop several versions and refinements of the attack. First we show that key dependent collisions can be caused in the output bytes of the mix column transformation in the first round. By taking advantage of the birthday paradox, it is possible to cause a collision in an output with as little as 20 measurements. If a SPA leak is present from which collisions can be determined with certainty, then each collision will reveal at least 8 bits of the secret key. Furthermore, in an optimized attack, it is possible to cause collisions in all four output bytes of the mix column transformation with an average of only 31 measurements, which results in knowledge of all 32 key bits. Finally, if collisions are caused in all four columns of the AES in parallel, it is possible to determine the entire 128-bit key with only 40 measurements, which a is a distinct improvement compared to DPA and other side channel attacks.

Keywords: AES, side channel attacks, internal collisions, birthday paradox

1 Introduction

An internal collision occurs, if a function within a cryptographic algorithm processes different input arguments, but returns an equal output argument. A typical example of subfunctions where internal collisions may occur are non-injective mappings, e.g., the S-boxes of DES, which map 6 to 4 bits. Moreover, partial collisions may also appear at the output of injective and non-injective transformations, e.g. in 3 bytes (24 bits) of a 4 byte (32 bit) output value. In the case of AES we will show that key dependent collisions can occur in one of the output bytes of the mix column transformation. We show that these internal collisions can be detected by power analysis techniques, therefore collision attacks should be regarded as a sub-category of Simple Power Analysis (SPA). The term internal collision implies itself that the collision cannot be detected at the output of the algorithm. In cooperation with Hans Dobbertin it was shown in [SWP03], that cross-correlation of power traces (or possibly EM radiation traces) makes it possible to detect internal collisions which provide information about the secret key. Furthermore, in [Nov03, Cla04] it is even claimed that internal collisions can be used to reverse-engineer substitution blocks of secret ciphers, such as unknown implementations of the A3/A8 GSM algorithm. Implementations which solely use countermeasures such as random wait states or dummy cycles will most probably succumb to internal collision attacks, since cross-correlation of power traces with variable time offsets will defeat these countermeasures. Also, in [Wie03] it was shown that the software countermeasure known as the duplication method [GP99] may not succeed against internal collisions. Another advantage of collision attacks over side channel attacks such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [KJJ99, KJJ98] is the fact that an internal collision will usually affect a sequence of instructions whereas SPA and DPA usually evaluate the power trace at a particular instance of time. For example, in the case of DES a collision in the output of the non-linear function f_k in round one will affect almost the entire second round. Detecting collisions by examining a sequence of instructions may be advantageous in terms of measurement costs. On the other hand, it must be noted that DPA based attacks against AES have the advantage of being known plaintext attacks whereas our proposed collision attack is a chosen plaintext attack.

The rest of this publication is organized as follows: in Section 2 the collision attack originally proposed in [SWP03] is applied against the AES. It is shown that partial collisions can occur in a single output byte of the mix column transformation and that these collisions depend on the secret key. In Section 3, an optimization of this attack is presented. It uses precomputed tables of a total size of 540 MB and on average yields 32 key bits with 31 encryptions (measurements). If the attack is applied in parallel to all four columns, 128 key bits can be determined with only 40 encryptions (measurements). In Section 4, we give information about our PC simulated attacks and practical attacks against a 8051 based microcontroller running AES in assembly. Finally, in Section 5, we summarize our results and give some conclusions.

2 Collisions in AES

2.1 Collisions in the Mix Column Transformation

In this section, we first briefly review the mix column transformation in AES. Then, we show how key dependent collisions can be caused in a single output byte of the mix column transformation.

The mix column transformation is linear and bijective. It maps a four-byte column to a four-byte column. Its main purpose is diffusion. Throughout this paper we follow the notation used in [DR02]. The mathematical background of the mix column transformation is as follows: all computations take place in $GF(2^8)$, represented by polynomials over $GF(2)$ modulo $m(x) = x^8 + x^4 + x^3 + x + 1$. Columns are interpreted as polynomials over $GF(2^8)$ and multiplied modulo $m(y) = y^4 + 1$. The input polynomial is multiplied with the fixed polynomial

$$c(y) = 03 \cdot y^3 + 01 \cdot y^2 + 01 \cdot y + 02$$

where 01, 02 and 03 refer to the $GF(2^8)$ elements 1, x and $x+1$, respectively. If we refer to the input column as $a(y)$ and to the output column as $b(y)$, the mix column transformation can be stated as

$$b(y) = a(y) \times c(y) \mod y^4 + 1$$

This specific multiplication with the fixed polynomial $c(y)$ can also be written as a matrix multiplication

$$\begin{pmatrix} b_{00} \\ b_{10} \\ b_{20} \\ b_{30} \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \times \begin{pmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \end{pmatrix}$$

If we look at the first output byte b_{00}, it is given by[1]

$$b_{00} = 02 \cdot a_{00} + 03 \cdot a_{10} + 01 \cdot a_{20} + 01 \cdot a_{30}$$

If we focus on the first round, we can substitute a_{00}, a_{10}, a_{20} and a_{30} with $S(p_{00} + k_{00}), S(p_{11} + k_{11}), S(p_{22} + k_{22})$ and $S(p_{33} + k_{33})$[2]. The output byte b_{00} can then be written as

$$b_{00} = 02 \cdot S(p_{00} + k_{00}) + 03 \cdot S(p_{11} + k_{11}) + 01 \cdot S(p_{22} + k_{22}) + 01 \cdot S(p_{33} + k_{33})$$

The main idea of this attack is to find two different plaintext pairs with the same output byte b_{00}. We are only considering plaintexts with $p_{00} = p_{11} = 0$ and $p_{22} = p_{33}$. If two plaintexts with $p_{22} = p_{33} = \delta$ and $p'_{22} = p'_{33} = \epsilon \neq \delta$ result in an equal output byte b_{00}, the following equation is satisfied:

$$S(\delta + k_{22}) + S(\delta + k_{33}) = S(\epsilon + k_{22}) + S(\epsilon + k_{33})$$

[1] The symbol + denotes an addition modulo 2, i.e. the binary exclusive-or operation.

[2] These are the diagonal elements of the plaintext and initial round key matrix due to the prior shift row transformation.

Suppose that an adversary has the necessary experience and measurement instrumentation to detect this collision in b_{00} (or any other output byte of the mix column transformation) with side channel analysis. First, he sets the two plaintext bytes p_{22} and p_{33} to a random value $\delta = p_{22} = p_{33}$. As next, he encrypts the corresponding plaintext, measures the power trace and stores it on his computer. He then keeps generating new random values $\epsilon = p'_{22} = p'_{33}$ unequal to previously generated values of δ,ϵ, and so on. He encrypt each new plaintext, measures and stores the corresponding power trace and cross-correlates it with all previously stored power traces until he detects a collision in output byte b_{00}. Once a collision has been found the task is to deduce information about k_{22} and k_{33}.

2.2 An Analysis of the Collision Function

To simplify the notation, we denote k_{00} (or k_{11}, k_{22}, k_{33}) simply by k_0 (or k_1, k_2, k_3) and output byte b_{00} by b_0. As described above, we are interested in values (δ, ϵ), such that for an unknown key the following equation is satisfied:

$$S(k_2 + \delta) + S(k_3 + \delta) + S(k_2 + \epsilon) + S(k_3 + \epsilon) = 0$$

Set

$$\mathcal{L}_{(a,b)} = \{(x,y) \in \mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \mid S(a+x) + S(b+x) + S(a+y) + S(b+y) = 0\}$$

The interpretation of this set is twofold. Given a key pair (k_2, k_3), the set $\mathcal{L}_{(k_2,k_3)}$ is the set of all pairs (δ, ϵ), which will lead to a collision in b_0. On the other hand, due to symmetry, the set $\mathcal{L}_{(\delta,\epsilon)}$ contains all possible key pairs, for which (δ, ϵ) will lead to a collision in byte b_0.

Note that if we measure a collision for δ and ϵ, the key (k_2, k_3) cannot be uniquely determined. This is due to the following properties of the set $\mathcal{L}_{(a,b)}$:

$$\begin{aligned}
\forall\ x \in \mathbb{F}_{2^8} \quad & (x,x) \in \mathcal{L}_{(a,b)} \\
(x,y) \in \mathcal{L}_{(a,b)} \Rightarrow & \ (y,x) \in \mathcal{L}_{(a,b)} \\
(x,y),(y,c) \in \mathcal{L}_{(a,b)} \Rightarrow & \ (x,c) \in \mathcal{L}_{(a,b)} \\
(x,y) \in \mathcal{L}_{(a,b)} \Rightarrow & \ (x, y+a+b) \in \mathcal{L}_{(a,b)}
\end{aligned}$$

Equations (1) to (3) establish an equivalence relation on $\mathbb{F}_{2^8}$.

More explicitly, if $(k_2, k_3) \in \mathcal{L}_{(\delta,\epsilon)}$, it follows that

$$(k_2 + \delta + \epsilon, k_3) \in \mathcal{L}_{(\delta,\epsilon)} \tag{1}$$

$$(k_2, k_3 + \delta + \epsilon) \in \mathcal{L}_{(\delta,\epsilon)} \tag{2}$$

$$(k_2 + \delta + \epsilon, k_3 + \delta + \epsilon) \in \mathcal{L}_{(\delta,\epsilon)} \tag{3}$$

$$(k_3, k_2) \in \mathcal{L}_{(\delta,\epsilon)} \tag{4}$$

$$(k_3 + \delta + \epsilon, k_2) \in \mathcal{L}_{(\delta,\epsilon)} \tag{5}$$

$$(k_3, k_2 + \delta + \epsilon) \in \mathcal{L}_{(\delta,\epsilon)} \tag{6}$$

$$(k_3 + \delta + \epsilon, k_2 + \delta + \epsilon) \in \mathcal{L}_{(\delta,\epsilon)} \tag{7}$$

and thus, we cannot hope to determine k_2 and k_3 completely given one collision. Let $(\delta, \epsilon) \in \mathcal{L}_{(k_2,k_3)}$ where we always assume that $\epsilon \neq \delta$. We have to discuss several cases:

case 1: If $k_2 = k_3$ then $\mathcal{L}_{(k_2,k_3)} = \mathbb{F}_{2^8} \times \mathbb{F}_{2^8}$, every choice of (δ, ϵ), i.e., every measurement will lead to a collision.

case 2: If $k_2 \neq k_3$ and if we furthermore assume that $\delta, \epsilon \notin \{k_2, k_3\}$ we obtain

$$0 = S(k_2 + \delta) + S(k_3 + \delta) + S(k_2 + \epsilon) + S(k_3 + \epsilon).$$

By expressing $S(x)$ as $L(x^{-1})$ and applying L^{-1} (where L is the affine transformation of the S-box) we conclude

$$0 = \frac{1}{k_2 + \delta} + \frac{1}{k_3 + \delta} + \frac{1}{k_2 + \epsilon} + \frac{1}{k_3 + \epsilon}$$

which finally yields

$$k_2 + k_3 = \delta + \epsilon.$$

case 3: If $k_2 = \delta$ and $k_3 = \epsilon$ or $k_3 = \delta$ and $k_2 = \epsilon$, we also conclude that $k_2 + k_3 = \delta + \epsilon$.

case 4: This case occurs if either $k_2 \in \{\delta, \epsilon\}$ or $k_3 \in \{\delta, \epsilon\}$. If $k_2 \in \{\delta, \epsilon\}$, we compute

$$p(k_3) = \frac{k_3^2}{(\delta + \epsilon)^2} + \frac{k_3}{\delta + \epsilon} + \frac{\delta\epsilon}{(\delta + \epsilon)^2} + 1 = 0 \tag{8}$$

This can be further simplified to

$$p(k_3) = \left(\frac{k_3 + \delta}{\delta + \epsilon}\right)^2 + \frac{k_3 + \delta}{\delta + \epsilon} + 1 = 0 \tag{9}$$

which shows that

$$\alpha = \frac{k_3 + \delta}{\delta + \epsilon} \in \mathbb{F}_4^* \backslash \{1\}$$

An analysis of the case $k_3 \in \{\delta, \epsilon\}$ yields a similar result. Combining both cases, we deduce the following possibilities for (k_2, k_3)

$$k_2 = \delta \text{ and } k_3 = \alpha(\delta + \epsilon) + \delta \tag{10}$$
$$k_2 = \epsilon \text{ and } k_3 = \alpha(\delta + \epsilon) + \delta \tag{11}$$
$$k_2 = \delta \text{ and } k_3 = \alpha(\delta + \epsilon) + \epsilon \tag{12}$$
$$k_2 = \epsilon \text{ and } k_3 = \alpha(\delta + \epsilon) + \epsilon \tag{13}$$
$$k_3 = \delta \text{ and } k_2 = \alpha(\delta + \epsilon) + \delta \tag{14}$$
$$k_3 = \epsilon \text{ and } k_2 = \alpha(\delta + \epsilon) + \delta \tag{15}$$
$$k_3 = \delta \text{ and } k_2 = \alpha(\delta + \epsilon) + \epsilon \tag{16}$$
$$k_3 = \epsilon \text{ and } k_2 = \alpha(\delta + \epsilon) + \epsilon \tag{17}$$

where $\alpha \in \mathbb{F}_4^* \backslash \{1\}$. In the case of the AES S-box, α can be chosen as $\alpha(x) = BC = x^7 + x^5 + x^4 + x^3 + x^2$. Note that solutions (10) to (16) correspond exactly to the seven additional possibilities (1) to (7).

Let us assume we detect a collision for a particular $(\delta,\epsilon) \in \mathcal{L}_{(k_2,k_3)}$. In order to deduce information about k_2 and k_3 we have to decide which case we deal with. We do not have to distinguish case two and case three, as the information we deduce about k_2 and k_3 is the same in both cases.
To distinguish case one, two or three from case four we use the following idea. Given a collision (δ,ϵ), we construct a new pair (δ',ϵ'), which will not lead to a collision if and only if (δ,ϵ) corresponds to case four. For this we need

Lemma 1. *Let*

$$\begin{aligned}\mathcal{L}_4 = \{&(k_2, \alpha(k_2+k_3)+k_2), (k_2, \alpha(k_2+k_3)+k_3),\\ &(k_3, \alpha(k_2+k_3)+k_2), (k_3, \alpha(k_2+k_3)+k_3)\\ &(\alpha(k_2+k_3)+k_2, k_2), (\alpha(k_2+k_3)+k_3, k_2),\\ &(\alpha(k_2+k_3)+k_2, k_3), (\alpha(k_2+k_3)+k_3, k_3)\}.\end{aligned}$$

Given an element $(\delta,\epsilon) \in \mathcal{L}_{(k_2,k_3)}$ the pair (δ',ϵ') with

$$\delta' \in \mathbb{F}_{2^8} \backslash \{\delta, \epsilon, \alpha(\delta+\epsilon)+\delta, \alpha(\delta+\epsilon)+\epsilon\}$$

and

$$\epsilon' = \delta' + \delta + \epsilon$$

is in $\mathcal{L}_{(k_2,k_3)}$ if and only if

$$k_2 = k_3$$

or

$$(\delta,\epsilon) \notin \mathcal{L}_4$$

i.e. if and only if (δ,ϵ) does not correspond to case four.

Proof.
"$\Leftarrow$:". If $k_2 = k_3$, the set $\mathcal{L}_{(k_2,k_3)} = \mathbb{F}_{2^8} \times \mathbb{F}_{2^8}$, so in particular $(\delta',\epsilon') \in \mathcal{L}_{(k_2,k_3)}$. If on the other hand $(\delta,\epsilon) \notin \mathcal{L}_4$, we see that $\forall \delta' \in \mathbb{F}_{2^8}$, the pair $(\delta', \delta'+\delta+\epsilon) \in \mathcal{L}_{(k_2,k_3)}$.

"$\Rightarrow$:"Assume $k_2 \neq k_3$ and $(\delta,\epsilon) \in \mathcal{L}_4$. W.l.o.g. let $\delta = k_2$ and $\epsilon = \alpha(k_2+k_3)+k_2)$. If $(\delta',\epsilon') \in \mathcal{L}_{(k_2,k_3)}$ we get

$$\frac{1}{k_2+\delta'} + \frac{1}{k_2+\epsilon'} + \frac{1}{k_3+\delta'} + \frac{1}{k_3+\epsilon'} = 0$$

If we substitute $k_3 = \alpha(\delta+\epsilon)+\epsilon$ and $\epsilon' = \delta+\epsilon+\delta'$, we conclude

$$\frac{1}{\delta+\delta'} + \frac{1}{\delta+\epsilon'} + \frac{1}{\alpha(\delta+\epsilon)+\epsilon+\delta'} + \frac{1}{\alpha(\delta+\epsilon)+\epsilon+\epsilon'} = 0$$

and due to the choice of δ' we finally get

$$\delta+\epsilon = \alpha(\delta+\epsilon)$$

a contradiction. □

Thus, with the pair (δ', ϵ') as constructed in the theorem, we can decide, if (δ, ϵ) corresponds to case four or not.
Now we are in a situation where we have to distinguish case one from cases two and three. If $k_2 \neq k_3$ we see that

$$D_{k_2,k_3} := \{a + b \mid (a, b) \in \mathcal{L}_{(k_2,k_3)}\}$$

contains only the values $k_2 + k_3$ in cases two and three and $\alpha(k_2 + k_3)$ and $(\alpha + 1)(k_2 + k_3)$ in case four. As a conclusion, we are able to exactly determine in which case we are in order to determine information about (k_2, k_3). In case one if $k_2 = k_3$ then $D_{k_2,k_3} = \mathbb{F}_{2^8}$. Thus if we are given a collision (δ, ϵ), we choose new values δ'' such that $\delta'' + \epsilon \notin \{\delta + \epsilon, \alpha(\delta + \epsilon), (\alpha + 1)(\delta + \epsilon)\}$. As argued above, such a pair (δ'', ϵ) will lead to a collision iff $k_1 = k_2$.

2.3 Probability Analysis of Collisions in a Single Output Byte

The probability that a collision occurs after n measurements is given by

$$P(n) = 1 - \prod_{i=0}^{n-1} \left(1 - \frac{i}{256}\right)$$

Table 1 lists various probabilities of a collision for a different number of measurements.

n	$P(n)$
1	0
10	0.1631
20	0.5332
30	0.8294
40	0.9599
50	0.9941

Table 1. Probability of a collision after n measurements

As a result, due to the birthday paradox an average of only 20 measurements are required in order to detect a collision in a single output byte of the mix column transformation.

3 Optimization of the Attack

In the last section, we described collisions which occur in a single output byte of the mix column transformation. This attack can be optimized by equally varying all four plaintext bytes which enter the mix column transformation while still focussing on collisions in one of the four output bytes, i.e., we now try to cause collisions with two pairs of plaintexts of the form $\delta = p_{00} = p_{10} = p_{20} = p_{30}$ and $\epsilon = p'_{00} = p'_{10} = p'_{20} = p'_{30}$. Moreover, we still look for collisions in a single output byte of the mix column transformation, however we observe all four outputs for collisions.

For example, a collision occurs in the first output byte of the mix column transformation whenever the following equation is fulfilled

$$\begin{aligned} C(\delta, \epsilon, k_0, k_1, k_2, k_3) &= 02S(k_0 + \delta) + 03S(k_1 + \delta) + S(k_2 + \delta) + S(k_3 + \delta) \\ &+ 02S(k_0 + \epsilon) + 03S(k_1 + \epsilon) + S(k_2 + \epsilon) + S(k_3 + \epsilon) \\ &= 0 \end{aligned}$$

We denote, for a known pair (δ, ϵ), the set of all solutions by

$$\mathcal{C}_{\delta,\epsilon} := \{(k_0, k_1, k_2, k_3) | C(\delta, \epsilon, k_0, k_1, k_2, k_3) = 0\}$$

Again, suppose that an adversary has the necessary equipment to detect a collision in any of the output bytes $b_{00}, \ldots, b_{30}$ with side channel analysis. In order to cause collisions in the outputs of the first mix column transformation, he sets the four plaintext bytes p_{00}, p_{11}, p_{22} and p_{33} to a random value $\delta = p_{00} = p_{11} = p_{22} = p_{33}$. As next, he encrypts the corresponding plaintext, measures the power trace and stores it on his computer. He then keeps generating new random values $\epsilon = p'_{00} = p'_{11} = p'_{22} = p'_{33}$ unequal to previously generated values of δ,ϵ, and so on. He encrypts each new plaintext, measures and stores the corresponding power trace and cross-correlates it with all previously stored power traces until he detects a collision in the observed output byte $b_{00}, \ldots, b_{30}$. Once a collision has been found the task is to deduce information about (k_0, k_1, k_2, k_3).

This equation can be solved by analysis or by using precomputed look-up tables which contain the solutions (k_0, k_1, k_2, k_3) for particular (δ, ϵ). However, this equation is much more complex than the one introduced in the previous section and an analog description is not trivial. An alternative solution to this problem is to create the sets $\mathcal{C}_{\delta,\epsilon}$ for every pair (δ, ϵ) by generating all possible values for (k_0, k_1, k_2, k_3) and checking $C(\delta, \epsilon, k_0, k_1, k_2, k_3) = 0$ for all pairs (δ, ϵ).

In our simulations we found that the resulting sets are approximately equal in size and on average contain $16,776,889 \approx 2^{24}$ keys, which corresponds to a size of 67 megabytes ($\approx 2^{26}$ bytes) per set. Multiplying this with the number of possible (δ, ϵ) sets, all sets together would require about $2,000$ gigabytes which is only possible with major efforts and distributed storage available. Reducing the amount of required disk space and still being able to compute all the necessary information is the purpose of the next section.

Moreover, it must be pointed out that there exist certain keys (k_0, k_1, k_2, k_3) for which no pair (δ, ϵ) will result in a collision. To our knowledge, there only exist three classes of keys (x, x, x, x), (x, x, x, y) and (x, x, y, y) which will not result in collisions for any pair (δ, ϵ). If the key (k_0, k_1, k_2, k_3) is an element of the key class (x, x, x, x), i.e. all four key bytes are equal, no collisions will occur in any of the four Mix Column output bytes for any pair (δ, ϵ) due to the overall required bijectivity of the Mix Column transform. The probability that this case occurs is $P = 2^8/2^{32} = 2^{-24}$. If the key (k_0, k_1, k_2, k_3) is an element of the key class (x, x, y, x) or (x, x, x, y), no collision will occur in the Mix Column output byte b_0. If the key (k_0, k_1, k_2, k_3) is an element of the key class (x, y, x, x) or (x, x, y, x), no collision will occur in the Mix Column output byte b_1. If the key (k_0, k_1, k_2, k_3) is an element of the key class (y, x, x, x) or (x, y, x, x), no collision will occur in the Mix Column output byte b_2. If the key (k_0, k_1, k_2, k_3) is an element of the key class (x, x, x, x) or (y, x, x, x), no collision will occur in the Mix Column output byte b_3. The probability that any of these cases occurs is $P = \frac{1}{256} \cdot \frac{1}{256} \cdot \frac{1}{256} \cdot \frac{255}{256} \approx 2^{-24}$. Our simulations showed that these are the only exceptional keys which will not result in collisions b_0, b_1, b_2 or b_3.

3.1 Reducing the Storage Requirements

First, note that the sets $\mathcal{C}_{\delta,\epsilon}$ also contain all the keys which will cause collisions in the output bytes b_1, b_2 and b_3. Since the entries in the mix column matrix are bytewise rotated to the right in each row, the stored 32-bit keys in the sets $\mathcal{C}_{\delta,\epsilon}$ must be cyclically shifted to the right by one, two or three bytes, as well, in order to cause collisions in b_1, b_2 and b_3.

Moreover, the amount of space can be further reduced by taking advantage of two different observations. First, we find some dependencies among the elements in a given set $\mathcal{C}_{\delta,\epsilon}$ and second we derive a relationship between two sets $\mathcal{C}_{\delta,\epsilon}$ and $\mathcal{C}_{\delta',\epsilon'}$.

The first approach uses an argument similar to an argument used in Section 2. If for a fixed pair (δ, ϵ) a key (k_0, k_1, k_2, k_3) is in $\mathcal{C}_{\delta,\epsilon}$, then the following elements are also in $\mathcal{C}_{\delta,\epsilon}$:

$$
\begin{aligned}
(k_0, k_1, k_2, k_3) &\in \mathcal{C}_{\delta,\epsilon} \\
&\Rightarrow \\
(k_0, k_1, k_3, k_2) &\in \mathcal{C}_{\delta,\epsilon} && (18) \\
(k_0 + \delta + \epsilon, k_1, k_2, k_3) &\in \mathcal{C}_{\delta,\epsilon} && (19) \\
(k_0, k_1 + \delta + \epsilon, k_2, k_3) &\in \mathcal{C}_{\delta,\epsilon} && (20) \\
(k_0, k_1, k_2 + \delta + \epsilon, k_3) &\in \mathcal{C}_{\delta,\epsilon} && (21) \\
(k_0, k_1, k_2, k_3 + \delta + \epsilon) &\in \mathcal{C}_{\delta,\epsilon} && (22)
\end{aligned}
$$

Combining these changes, we find 32 different elements in $\mathcal{C}_{\delta,\epsilon}$, given that $k_2 \neq k_3$ and $\delta + \epsilon \neq 0$. The case $\delta + \epsilon = 0$ is à priori excluded. If $k_2 = k_3$, we still find 16 different elements in $\mathcal{C}_{\delta,\epsilon}$. For the purpose of storing the sets $\mathcal{C}_{\delta,\epsilon}$, this shows that it is enough to save one out of 32 (resp. 16) elements in the $\mathcal{C}_{\delta,\epsilon}$ tables. This results in a reduction of required disk space by a factor of $(16+255*32)/256 \approx 32$. The second approach to save storage space is based on the following observation: an element (k_0, k_1, k_2, k_3) is in $\mathcal{C}_{\delta,\epsilon}$, if and only if $(k_0 + a, k_1 + a, k_2 + a, k_3 + a) \in \mathcal{C}_{\delta+a,\epsilon+a}$. Thus, every set $\mathcal{C}_{\delta,\epsilon}$ can be easily computed from the set $\mathcal{C}_{\delta+\epsilon,0}$. This shows that it is enough to store for all $\delta_0 \in \mathbb{F}_{2^8}$ the set $\mathcal{C}_{\delta_0,0}$.

Combining these two approaches reduces the required disk space by a factor of approx. $128 * 32 = 2^{12}$, and hence we only need approximately 540 megabytes which is no problem on today's PC. As a matter of fact, the sets $\mathcal{C}_{\delta+\epsilon,0}$ will fit on a regular CD-ROM.

3.2 Probability Analysis of the Optimized Attack

We analyze the functions, which map a value δ to an output b_i for a fixed key (k_0, k_1, k_2, k_3) as independent random functions from $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^8}$ in rows one to four. We want to determine the expected number of measurements until at least one collision has occurred in every output byte $b_0, \cdots, b_3$.
As stated in Section 2.3, the probability that after n measurements at least one collision occurs in a single output byte $b_0, \cdots, b_3$ is given by

$$P(n) = 1 - \prod_{i=0}^{n-1} \left(1 - \frac{i}{256}\right)$$

For $n = 20$, $P(20) = 0.5332 \geq 1/2$, which means that on average 20 measurements are required in order to detect a collision. In the optimized attack, we want to determine the number of required measurements such that at least one collision occurs in all the values $b_0, \cdots, b_3$ with a probability greater than 1/2. Therefore, we have to compute the minimum value n such that $P(n) \geq (1/2)^{1/4}$. As a result, we obtain $n = 31$, thus after an average of 31 measurements collisions will be detected in all four rows of the mix column transformation.
Every collision (δ, ϵ) will yield possible key candidates (k_0, k_1, k_2, k_3), which can be looked up in the stored tables $\mathcal{C}_{\delta+\epsilon,0}$. Our thorough simulations show that every new collision will decrease the intersection of all key candidates by approximately 8 bit. As a result, we are able to determine the entire 32-bit key (k_0, k_1, k_2, k_3) after collisions have been detected in all four output bytes $b_0, \cdots, b_3$.
Furthermore, it is possible to apply the optimized attack in parallel against all four columns. If we do not only consider the values $b_0, \cdots, b_3$, but also the output bytes $b_4, \cdots, b_{15}$ of the remaining columns, we have to compute the minimal value n such that $P(n) \geq (1/2)^{1/16}$. As a result, we get $n = 40$, thus after an average of 40 measurements at least one collision will be detected in each of the 16 outputs $b_0, \cdots, b_{15}$. These values are verified by our simulations. Thus, on average we only need 40 measurements to determine the whole 128-bit key.

4 Simulation and Practical Attack

As a proof of concept, the AES collision attack was simulated on a Pentium 2.4 GHz PC and results were averaged over 10,000 random keys. As stated above, whenever a collision occurs, all possible key candidates can be derived from the sets $\mathcal{C}_{\delta+\epsilon,0}$ and every further collision will provide an additional set of key candidates. The intersection of all sets of key candidates must then contain the real key. As shown in table 2, our simulations made clear that the number of key candidates in the intersection decreases by approximately 8 bit with each new collision.

no. of collisions in b_0, b_1, b_2 and b_3	0	1	2	3	4
no. of key candidates	2^{32}	16,777,114	65492	256.6	1.065

Table 2. Average no. of key candidates after one or more collisions have occured.

In order to check the practicability of the attack, an 8051 based microcontroller running an assembly implementation of AES without countermeasures was successfully compromised using the proposed collision attack. In our experiments, the microcontroller was running at a clock frequency of 12 MHz. At this frequency it takes about 3.017 ms to encrypt a 128-bit plaintext with a 128-bit key[3]. A host PC sends chosen plaintexts to the microcontroller and thus triggers new encryptions. In order to measure the power consumption of the microcontroller a small shunt resistance ($R_s = 39\Omega$) was put in series between the ground pad of the microcontroller and the ground connection of the power supply. Moreover, we replaced the original voltage source of the microcontroller with a low-noise voltage source to minimize noise superimposed by the source.

A digital oscilloscope was used to sample the voltage over the shunt resistance. We focused on collisions $S(k_{22})+S(k_{33}) = S(\delta+k_{22})+S(\delta+k_{33})$ in output byte b_{00} of the mix column transformation in the first round. Our main interest was to find out which measurement costs (sampling frequency and no. of averagings per encryption) are required to detect such a collision. Within the 8051 AES implementation the following assembly instructions in round two are directly affected by a collision in byte b_{00}:

```
mov     a, 30h      ;(1) Read round 1 mix column output byte b_00
xrl     a, 40h      ;(1) X-Or b_00 with round 2 key byte k_00
movc    a, @a+dptr  ;(2) S-box lookup
mov     30h, a      ;(1) Write back the S-box output value
```

The number of machine cycles per instruction is given in parentheses in the remarks following the assembly instructions. Since the microcontroller is clocked at 12 MHz which corresponds to a machine cycle length of 1 μs, this instruction sequence lasts about 5 μs. We began our experiments at a sampling rate of 500 MHz and one single measurement per encyption, i.e. no averaging of power traces was applied. In order to examine collisions, plaintext bytes $p_{22} = p_{33} = \delta$ were varied from $\delta = 1...255$ and compared with the reference trace at $p_{22} = p_{33} = 0$ by applying the least-squares method:

[3] using on-the-fly key scheduling

$$R[\delta] = \left(\sum_{t=t_0}^{t_0+N-1} (p[t,0] - p[t,\delta])^2 \right)^{-1} \tag{23}$$

At a sampling rate of 500 MHz the number of sampling points N is 2500. Figure 1 shows the deviation $R[\delta]$ of power traces with $\delta = 1...255$ from the reference trace with $\delta = 0$. Our AES implementation used the key bytes $k_{22} = 21$ and $k_{33} = 60$, therefore we expected a distinct peak at $\delta = k_{22} \oplus k_{33} = 41$ as shown in Figure 1. It is interesting to note that no averaging of power traces was applied, therefore, it would be possible to break the entire 128-bit key with only 40 measurements[4].

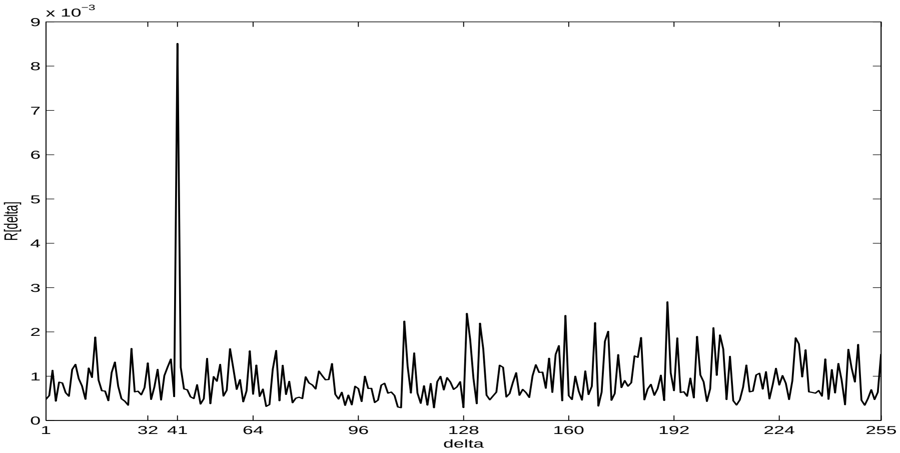

Fig. 1. Deviation of power traces with $\delta = 1...255$ from the reference trace with $\delta = 0$.

We also investigated other signal analysis methods such as computation of the normalized Pearson correlation factor [MS00] and continous wavelet analysis in order to detect internal collisions. As a result, we concluded that computation of the Pearson correlation factor does only seem to be an approriate method when focussing on very particular instances of time within a machine cycle, e.g. when bits on the data or address bus are switched. We achieved very good collision detection using wavelet analysis, however, when compared with the least-squares method, its computational costs are much higher. We will further introduce wavelet analysis in the field of simple power analysis and related reverse-engineering in the future, since it is far beyond the scope of this paper.

[4] provided that the attacker knows the instances of time when mix column outputs are processed in round two

5 Results and Conclusions

We proposed two new variants of the collision attack which use side channel analysis to detect internal collisions in AES. Typical methods to recognize internal collisions are computation of square differences, cross-correlation of power consumption curves or application of more advanced methods used in signal analysis theory, such as wavelet analysis. We showed that partial collisions can be detected in the output bytes of the mix column transformation in the first round of AES and each collision typically provides 8 bits of the secret key.
When compared with Differential Power Analysis (DPA) our proposed collision attack has the advantage of requiring less power trace measurements. However, DPA has the advantage of being a known plaintext attack whereas the collision attack is a chosen plaintext attack. A DPA against AES which yields the correct key hypothesis typically requires between 100 and 1,000 measurements depending on the presence of superimposed noise. Our collision attack on the other hand takes advantage of the birthday paradox. As a result, we are able to determine the entire 128-bit key with only 40 measurements.

6 Acknowledgements

We would like to thank Markus Bockes for pointing out that there exist certain keys which will not result in a collision in a particular Mix Column single output byte when the optimzed collision attack is applied.

References

[Cla04] C. Clavier. Side Channel Analysis for Reverse Engineering (SCARE). http://eprint.iacr.org/2004/049/, 2004. Cryptology ePrint Archive: Report 2004/049.

[CRR02] S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, pages 13–28. Springer-Verlag, 2002.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag, Berlin, Germany, 2002.

[GP99] L. Goubin and J. Patarin. DES and differential power analysis: the duplication method. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 1999*, volume LNCS 1717, pages 158–172. Springer-Verlag, 1999.

[KJJ98] P. Kocher, J. Jaffe, and B. Jun. Introduction to Differential Power Analysis and Related Attacks. http://www.cryptography.com/dpa/technical, 1998. Manuscript, Cryptography Research, Inc.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology — CRYPTO '99*, volume LNCS 1666, pages 388–397. Springer-Verlag, 1999.

[MDS99] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology*, pages 151–162, 1999.

[MS00] R. Mayer-Sommer. Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smart Cards. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume LNCS 1965, pages 78 – 92. Springer-Verlag, 2000.

[Nov03] R. Novak. Side-Channel Attack on Substitution Blocks. In *ACNS 2003*, volume LNCS 2846, pages 307–318. Springer-Verlag, 2003.

[SWP03] K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and its Application to DES. In Thomas Johansson, editor, *Fast Software Encryption — FSE '03*, volume LNCS 2887, pages 206 – 222. Springer-Verlag, February 2003.

[Wie03] A. Wiemers. Partial Collision Search by Side Channel Analysis. Presentation at the Workshop: Smartcards and Side Channel Attacks, January 2003. Horst Goertz Institute, Bochum, Germany.