Improved Side-Channel Collision Attacks on AES

Andrey Bogdanov

Chair for Communication Security
Ruhr University Bochum, Germany
abogdanov@crypto.rub.de
www.crypto.rub.de

Abstract. Side-channel collision attacks were proposed in [1] and applied to AES in [2]. These are based on detecting collisions in certain positions of the internal state after the first AES round for different executions of the algorithm. The attack needs about 40 measurements and 512 MB precomputed values as well as requires the chosen-plaintext possibility.

In this paper we show how to mount a collision attack on AES using only 6 measurements and about $2^{37.15}$ offline computational steps working with a probability of about 0.85. Another attack uses only 7 measurements and finds the full encryption key with an offline complexity of about $2^{34.74}$ with a probability of 0.99. All our attacks require a negligible amount of memory only and work in the known-plaintext model. This becomes possible by considering collisions in the S-box layers both for different AES executions and within the same AES run. All the attacks work under the assumption that one-byte collisions are detectable.

Keywords: AES, collision attacks, side-channel attacks, generalized collisions, connected components, random graphs.

1 Introduction

An internal collision, as defined in [1] and [2], occurs, if a function f within a cryptographic algorithm processes different input arguments, but returns an equal output argument. As applied to AES, Schramm et al. [2] consider the byte transforms of the MixColumn operation of the first AES round as the colliding function f. To detect collisions, power consumption curves bytewise corresponding to separate S-box operations in the second round at a certain internal state position after the key addition are compared.

The key idea of our improved collision attacks on AES is that one can detect equal inputs to various S-boxes by comparing the corresponding power consumption curves. This turns out to be possible not only for the outputs of the same function f: Using this technique, it can be possible to detect whether two inputs to the AES S-box are equal within the same AES execution as well as for different AES runs.

C. Adams, A. Miri, and M. Wiener (Eds.): SAC 2007, LNCS 4876, pp. 84–95, 2007.

We introduce the notion of *a generalized internal collision* for AES that occurs within one or several AES runs, if there are two equal input bytes to the S-box operation in some (possibly different) rounds at some (possibly different) byte positions for one or several measurements. In other words, we take all applications of the S-box transform within a number of AES executions and compare them pairwise to each other. As the S-box operation is applied 16 times in each of the 10 rounds (160 varied S-box operations), this gives us about 40 generalized collisions within a single AES run or about 710 generalized collisions within just 6 AES executions.

Each of such collisions can be considered as a (generally) non-linear equation over $GF(2^8)$. The set of all detected collisions corresponds to a system of non-linear equations with respect to the key bytes. In this paper we explore the question of how to solve this large number of equations *linearly*. To be able to linearize, we restrict our consideration to the first two rounds. There are three most efficient attacks in this class we found. The first one requires 7 measurements and $2^{34.74}$ offline operations on average with a probability of 0.99. The second attack needs about 6 measurements and about $2^{37.15}$ offline operations with probability 0.854 or about $2^{44.3}$ operations with probability 0.927. The third one recovers the key with just 5 measurements and about $2^{37.34}$ simple offline operations with probability 0.372 or about $2^{45.5}$ operations with probability 0.548. This is to be compared to about 40 measurements required in the basic collision attack [2] on AES with some non-negligible post-processing, 29 measurements required for the AES-based Apha-MAC internal state recovery in [3] with about 2^{34} offline operations with a success probability > 0.5, and typically several hundred measurements for a DPA (differential power analysis) attack.

Our attacks work, as DPA and the collision attacks on Alpha-MAC in [3], in the *known-plaintext model*, while the attack in [2] is applicable in the *chosen-plaintext scenario* only. Moreover, as in [3], we do not need to know the output of the cryptographic transformation for the side-channel attack itself. However, our attacks mentioned above do need one plaintext-ciphertext pair for choosing the correct key from a set of key candidates in the offline post-processing stage. Note that this input-output pair does not have to be one of the those for which the measurements have been performed.

We use both theoretical and experimental tools for estimating the efficiency of our attacks. Linear systems of equations are rewritten in terms of associated undirected graphs. As the resulting equation systems never possess the full rank, combinatorial methods are applied to solve these systems. The complexity of these methods can be analyzed through connected components of those graphs. The expected number of edges in such a graph is computed theoretically. The number of connected components, which defines the overall complexity of the offline attack stage, is estimated using thorough computer simulations for the numbers of edges obtained theoretically.

The remainder of the paper is organized as follows. Section 2 outlines the basic collision attack on AES. Section 3 rigorously introduces the notion of

a generalized internal collision for AES as well as specifies and analyzes our enhanced collision attacks. In Section 4 we discuss the technical framework and practical feasibility of our attacks. We conclude in Section 5.

2 Basic Collision Attack on AES

Side-channel collision attacks were proposed for the case of the DES in [1] and enhanced in [4]. AES was attacked using collision techniques in [2]. This side-channel collision attack on AES is based on detecting internal one-byte collisions in the MixColumns transformation in the first AES round. The basic idea is to identify pairs of plaintexts leading to the same byte value in an output byte after the MixColumns transformation of the first round and to use these pairs to deduce information about some key bytes involved into the transformation.

Let $A = (a_{ij})$ with $i, j = \overline{0,3}$ and $a_{ij} \in GF(2^8)$ be the internal state in the first AES round after key addition, byte substitution and the ShiftRows operation. Let $B = (b_{ij})$ with $i, j = \overline{0,3}$ and $b_{ij} \in GF(2^8)$ be the internal state after the MixColumns transformation, $B = \text{MixColumns}(A)$, where the MixColumns transformation is defined for each column j as follows:

$$\begin{pmatrix} b_{0j} \\ b_{1j} \\ b_{2j} \\ b_{3j} \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \times \begin{pmatrix} a_{0j} \\ a_{1j} \\ a_{2j} \\ a_{3j} \end{pmatrix}. \tag{1}$$

Here all operations are performed over $GF(2^8)$. Let $P = (p_{ij})$ with $i, j = \overline{0,3}$, $p_{ij} \in GF(2^8)$, and $K = (k_{ij})$ with $i, j = \overline{0,3}$, $k_{ij} \in GF(2^8)$, denote the plaintext block and the first subkey, respectively. Then b_{00} can be represented as:

$$\begin{aligned} b_{00} &= 02 \cdot a_{00} \oplus 03 \cdot a_{10} \oplus 01 \cdot a_{20} \oplus 01 \cdot a_{30} = \\ &= 02 \cdot S(p_{00} \oplus k_{00}) \oplus 03 \cdot S(p_{11} \oplus k_{11}) \\ &\quad \oplus 01 \cdot S(p_{22} \oplus k_{22}) \oplus 01 \cdot S(p_{33} \oplus k_{33}). \end{aligned} \tag{2}$$

For two plaintexts P and P' with $p_{00} = p_{11} = p_{22} = p_{33} = \delta$ and $p'_{00} = p'_{11} = p'_{22} = p'_{33} = \epsilon$, $\delta \neq \epsilon$, one obtains the following, provided $b_{00} = b'_{00}$:

$$\begin{aligned} &02 \cdot S(k_{00} \oplus \delta) \oplus 03 \cdot S(k_{11} \oplus \delta) \oplus 01 \cdot S(k_{22} \oplus \delta) \oplus 01 \cdot S(k_{33} \oplus \delta) \\ &= 02 \cdot S(k_{00} \oplus \epsilon) \oplus 03 \cdot S(k_{11} \oplus \epsilon) \oplus 01 \cdot S(k_{22} \oplus \epsilon) \oplus 01 \cdot S(k_{33} \oplus \epsilon) \end{aligned} \tag{3}$$

Let $C_{\delta,\epsilon}$ be the set of all key bytes $k_{00}, k_{11}, k_{22}, k_{33}$ that lead to a collision (3) with plaintexts (δ, ϵ). Such sets are pre-computed and stored for all 2^{16} pairs (δ, ϵ). Each set contains on average 2^{24} candidates for the four key bytes. Actually, every set $C_{\epsilon,\delta}$ can be computed from the set $C_{\epsilon \oplus \delta, 0}$ using some relations between the sets. Due to certain dependencies within the sets, this optimization reduces the required disk space to about 540 megabytes.

The attack on the single internal state byte b_{00} works as follows. The attacker generates random values (ϵ, δ) and inputs them to the AES module as described

above. The power consumption curve for the time period, where b_{00} is processed, is stored. Then the attacker proceeds with other random values (ϵ', δ'), measures the power profile, stores it and correlates it with all stored power curves. And so on. One needs about 4 collisions (one in each output byte of a column) to recover the four bytes involved into the MixColumns transformation. The probability that after N operations at least one collision $b_{00} = b'_{00}$ occurs in a single byte is:

$$p_N = 1 - \prod_{l=0}^{N-1} (1 - l/2^8). \tag{4}$$

Actually, the attack can be parallelized to search for collisions in all four columns of B in parallel. In this case the attacker needs at least 16 collisions, 4 for each column of B, so $p_N^{16} \geq 1/2$ and $N \approx 40$. Once the required number of collisions was detected, he uses the pre-computed tables $C_{\epsilon \oplus \delta, 0}$ to recover all four key bytes for each column by intersecting the pre-computed key sets corresponding to the collisions (ϵ, δ) detected. Thus, on average one has to perform about 40 measurements to obtain all 16 collisions needed and to determine all 16 key bytes. Note that since the cardinality of the intersections for the sets $C_{\epsilon,\delta}$ is not always 1, there are a number of key candidates to be tested using a known plaintext-ciphertext pair.

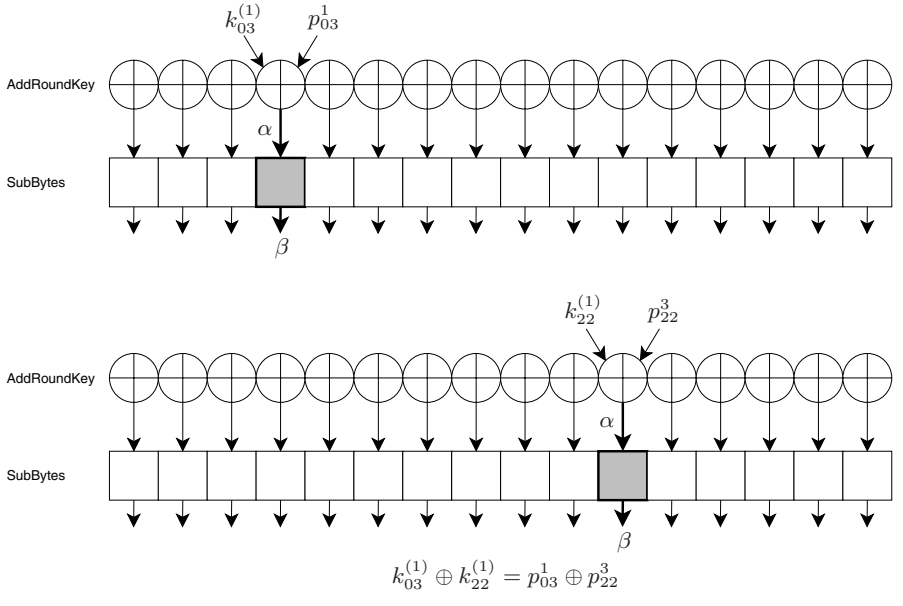

Fig. 1. Generalized internal collision within the first round of two AES runs

3 Our Improved Collision Attacks on AES

3.1 Generalized Internal Collisions

In round $i = \overline{1,10}$, AES performs the SubSytes operation (16 parallel S-box applications) on the output bytes of the previous round XORed with the i-th round subkey $K^{(i)}$. A generalized internal AES collision occurs, if there are two S-boxes within the same AES execution or within several AES runs accepting the same byte value as their input.

In Figure 1 a collision within the first round of two different AES executions (number 1 and 3) is illustrated. $p^j_{v,u}$, $v,u = \overline{0,3}$, are plaintext bytes for the jth measurement. $k^{(1)}_{v,u}$, $v,u = \overline{0,3}$, are the first subkey bytes remaining constant for the both executions. In the example of Figure 1, byte 03 in the first execution and byte 22 in the third execution collide.

A detected collision in the S-box layer of the first round in bytes (i_1, j_1) and (i_2, j_2) with $i_1, j_1, i_2, j_2 = \overline{0,3}$ corresponds to the following linear equation:

$$S(k^{(1)}_{i_1,j_1} \oplus p^x_{i_1,j_1}) = S(k^{(1)}_{i_2,j_2} \oplus p^y_{i_2,j_2}), \tag{5}$$

$$k^{(1)}_{i_1,j_1} \oplus k^{(1)}_{i_2,j_2} = \Delta^{(1)}_{(i_1,j_1),(i_2,j_2)} = p^x_{i_1,j_1} \oplus p^y_{i_2,j_2} \tag{6}$$

for some known plaintext bytes $p^x_{i_1,j_1}$ and $p^y_{i_2,j_2}$ with some positive integers x, y indicating measurement numbers. In the same way, one can rewrite equations resulting from collisions within some other round $i = \overline{2,10}$. In this case we have some unknown key- and plaintext-dependent byte variables instead of the plaintext bytes $p^x_{i_1,j_1}$ and $p^y_{i_2,j_2}$.

3.2 Systems of Equations and Associated Graphs

First, we consider the structure of a random system of m linear equations of type (6) resulting from a number of collisions detected within the S-box layer in the first round:

$$S_m: \quad \begin{cases} k^{(1)}_{i_1,j_1} \oplus k^{(1)}_{i_2,j_2} = \Delta^{(1)}_{(i_1,j_1),(i_2,j_2)} \\ k^{(1)}_{i_3,j_3} \oplus k^{(1)}_{i_4,j_4} = \Delta^{(1)}_{(i_3,j_3),(i_4,j_4)} \\ \dots \\ k^{(1)}_{i_{2m-1},j_{2m-1}} \oplus k^{(1)}_{i_{2m},j_{2m}} = \Delta^{(1)}_{(i_{2m-1},j_{2m-1}),(i_{2m},j_{2m})}. \end{cases} \tag{7}$$

Note that this system has 16 variables (bytes of the first round subkey). In system (7) the key byte numbers and the variables are not necessarily pairwise distinct.

The following straightforward proposition holds for S_m:

Proposition 1. *The maximal rank of S_m is 15, $rank(S_m) \leq 15$.*

Proof. The maximal rank of 15 is attained, for instance, for

$$\begin{cases} k^{(1)}_{0,0} \oplus k^{(1)}_{0,1} = \Delta^{(1)}_{(0,0),(0,1)} \\ k^{(1)}_{0,1} \oplus k^{(1)}_{0,2} = \Delta^{(1)}_{(0,1),(0,2)} \\ \dots \\ k^{(1)}_{3,1} \oplus k^{(1)}_{3,2} = \Delta^{(1)}_{(3,1),(3,2)} \\ k^{(1)}_{3,2} \oplus k^{(1)}_{3,3} = \Delta^{(1)}_{(3,2),(3,3)}. \end{cases} \tag{8}$$

It is easy to see that the XOR of any other pair of variables can be obtained as a sum of two of the 15 equations in (8). Thus, 15 is the maximal rank for S_m □

We use the graph representation of S_m for our analysis.

Definition 1. *A random graph $G_m = \langle V, E\rangle$ is associated with the random system S_m of linear equations, where $V = \{k^{(1)}_{0,0}, k^{(1)}_{0,1}, \dots, k^{(1)}_{3,3}\}$ is the set of 16 vertices of G_m and the edge $(k^{(1)}_{i_1,j_1}, k^{(1)}_{i_2,j_2})$ belongs to the edge set E iff the binomial equation*

$$k^{(1)}_{i_1,j_1} \oplus k^{(1)}_{i_2,j_2} = \Delta_{(i_1,j_1),(i_2,j_2)}$$

belongs to the system S_m, $|E| = m$.

Among others, the associated graph possesses the following obvious properties:

Proposition 2. *The system S_m is of the maximal rank 15 iff its associated graph G_m is connected.*

Proposition 3. *Let $G = \langle V, E\rangle$ be a non-directed graph with n vertices, $|V| = n$. If*

$$|E| > \binom{n-1}{2},$$

the graph G is connected.

For G_m Proposition 3 implies that if $|E| > 105$, G_m is necessarily connected and, thus, S_m has the maximal rank of 15. A system of type (7) having the maximal rank can be solved by assigning a byte value to some variable $k^{(1)}_{i,j}$ (which is equivalent to adding a further, linearly non-dependent equation $k^{(1)}_{i,j} = \Delta^{(1)}_{i,j}$ to the system) and uniquely solving the system

$$S_m \cup \left\{k^{(1)}_{i,j} = \Delta^{(1)}_{i,j}\right\}$$

of rank 16. Then another byte value is assigned to that variable. The correct key is identified on the basis of a known plaintext-ciphertext pair.

Generally speaking, it is not necessary for S_m to have the maximal rank. If there are several isolated subsystems within S_m, then each of them can be solved independently as described above. If there are q independent subsystems $SS^1_m, \dots, SS^q_m$ in S_m, then S_m can be represented as a union of these subsystems:

$$S_m = SS^1_m \cup \dots \cup SS^q_m,\ SS^i_m \cap SS^j_m = \varnothing,\ i \neq j.$$

To solve S_m in this case, one has to assign q byte values to some q variables in the subsystems $\{SS_m^i\}_{i=1}^q$. At the end there are 2^{8q} key candidates. The correct key is identified using a known plaintext-ciphertext pair as outlined above.

It is clear that the independent subsystems $\{SS_m^i\}_{i=1}^q$ of S_m correspond to the q connected components of the associated graph G_m.

The number of connected components of a random graph has the following asymptotic behaviour:

Proposition 4. *Let G be a random graph with n labeled vertices and $N = \lfloor \frac{1}{2} n \log n + cn \rfloor$ for some constant c. Let $q = q_{n,N}$ be the number of connected components in G. Then:*

$$\lim_{n\to\infty} \Pr\{q = i+1\} = \frac{(e^{-2c})^i}{i!} \exp\left\{-e^{-2c}\right\}.$$

Proof. See Theorem 2.3 in [5] □

Unfortunately, the estimate of Proposition 4 for the number of connected components cannot be directly applied for S_m, since its associated graph has only 16 vertices.

3.3 Expected Number of Random Binomial Equations

The number of edges in the associated graph G_m can be estimated using the following

Proposition 5. *If generalized byte collision in AES are always detectable, the expected number $E(m)$ of edges in G_m (equivalently, the expected number of binomial equations in S_m) for the first round of AES after $t \geq 1$ measurements is*

$$E(m) = 120 \cdot \left(1 - \left(\frac{119}{120}\right)^{16t-256+256\cdot\exp\left\{16t\cdot\ln\frac{255}{256}\right\}}\right).$$

Proof. The expected number of generalized collisions within the first round after t measurements can be estimated as:

$$N_{1\mathrm{R}} = 16t - 256 + 256 \cdot \left(\frac{255}{256}\right)^{16t}, \tag{9}$$

where $16t$ is the number of S-box operations in one AES round within t measurements. This equation is a reformulation of the birthday paradox.

The expected number of edges in a random graph with n labeled vertices after N_{1R} random selections of edges (after N_{1R} generalized collisions) can be interpreted as the expected number of filled boxes after N_{1R} random shots in the classical shot problem, which was studied e.g. in Chapter 1 of [6]. In the case of a graph,one deals with $\binom{n}{2}$ boxes (possible graph edges) and the expected number of edges after N_{1R} collisions is

Table 1. Number of collisions and edges in G_m according to Proposition 5

Measurements, t	4	5	6	7	8	9	11	29
1R collisions, N_{1R}	7.27	11.18	15.82	21.14	28.12	33.70	48.55	249.64
Edges, $E(m)$	7.09	10.72	14.88	19.46	24.36	29.49	40.07	105.14

$$E(m) = \binom{n}{2}\left(1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^{N_{1R}}\right). \tag{10}$$

As $n = 16$ for the case of AES, one obtains the claim of the proposition by combining (9) and (10) □

Table 1 contains theoretical estimations for the numbers of 1R-collisions N_{1R} and edges $E(m)$ depending on the number of measurements t for some interesting t's.

Note that according to Proposition 3, it is expected that after 29 measurements one obtains 105 edges which provide the maximal rank of S_m. However, on average a lower number of edges are sufficient for the G_m to be connected with a high probability (see Section 3.4).

3.4 Number of Connected Components in Associated Graphs

In order to estimate the number q of connected components for G_m accounting for the offline complexity, statistical simulation was applied consisting of generating a random graph on 16 vertices corresponding to t measurements as well as counting the number of connected components q using a variation of Karp and Tarjan's algorithm [7] for finding connected components of a graph. Note that the expected complexity of this algorithm in $O(n)$, that is, linear in the number of vertices. For each number of measurement we performed 2^{16} simulations with random graphs.

The results of our simulations are shown in Table 2. The first and second rows of the table represent measurement numbers and average numbers of edges in G_m according to Proposition 5 (see also Table 1), respectively. The offline

Table 2. Offline complexity and success probabilities

Measurements, t	4	5	6	7	8	9	11	29
Number of edges in G_m, m	7.09	10.72	14.88	19.46	24.36	29.49	40.07	105.14
Connected components of G_m, q	8.81	5.88	3.74	2.20	1.43	1.15	1.04	1.00
Offline complexity ≤ 40 bit	34.70	37.34	37.15	34.74	30.32	21.36	12.11	8
Success probability ≤ 40 bit	0.037	0.372	0.854	0.991	0.999	1.000	1.000	1.000
Offline complexity ≤ 48 bit	43.90	45.50	44.30	41.14	30.32	21.36	12.11	8
Success probability ≤ 48 bit	0.092	0.548	0.927	0.997	0.999	1.000	1.000	1.000

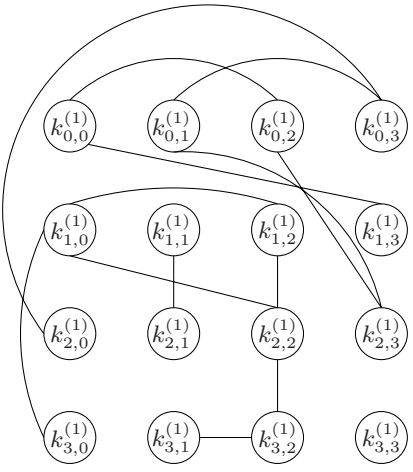

Fig. 2. Typical random graph with 13 edges and 4 components

complexity is given for two cases: ≤ 40 bit and ≤ 48 bit. In the first case, only offline complexities $\leq 2^{40}$ are considered in the computation of the average offline complexity value. For each number of measurements the probability is provided that the overall offline complexity is $\leq 2^{40}$. In the second case, the upper bound for the offline complexities taken into account is 2^{48}. The corresponding success probabilities are also provided in the table.

A low-complexity offline stage of the attack becomes probable after 5 measurements ($2^{45.5}$ simple steps with a probability of 0.548). Practically all instances of linear systems resulting from 7 measurements are easily solvable with an average complexity of $2^{34.74}$ steps (with a probability of 0.99). After 11 measurements the expected offline attack complexity is about $2^{12.11}$.

Figure 2 shows a typical random graph G_m associated with a random system S_m of linear equations with $m = 13$ and 4 independent subsystems (4 connected components): $\{k^{(1)}_{0,0}, k^{(1)}_{2,3}, k^{(1)}_{0,1}, k^{(1)}_{0,2}, k^{(1)}_{0,3}, k^{(1)}_{1,3}, k^{(1)}_{2,0}\}$, $\{k^{(1)}_{2,2}, k^{(1)}_{3,0}, k^{(1)}_{3,1}, k^{(1)}_{3,2}, k^{(1)}_{1,0}, k^{(1)}_{1,2}\}$, $\{k^{(1)}_{1,1}, k^{(1)}_{2,1}\}$, $\{k^{(1)}_{3,3}\}$.

3.5 Optimization of the Attack

In this subsection we propose an optimization of the attack described in the previous subsections. It consists in generating additional collisions for the first round by considering the second round and key schedule.

The basis of this optimization is the fact that there are also generalized byte collisions within the second AES round as well as between the first and the second AES rounds. However, as opposed to those in the first round, the values of inputs to the second round are not known and depend on the key and plaintext bytes in a non-linear way.

Suppose after N_{1R} collisions have been detected, the graph G_m consists of q connected components, but their number is too high to allow for an efficient solution of the corresponding system (e.g. $q = 7$). Let 2 or 3 connected components of this graph contain at least two diagonals of the first subkey $K^{(1)}$. Then we can test all 2^{16} or 2^{24}, respectively, possible candidates for these diagonals. Each subkey diagonal in the first round corresponds to a column in the second round. Thus, at least two columns of the input to the second round can be considered as known. Now we assume that a number of other bytes of the first round subkey also lie in the same 2 or 3 connected components of G_m. This can allow for the recovery of some of the second round subkey bytes corresponding to the known input columns. Thus, the corresponding inputs to the S-box layer of the second round can be assumed as known.

Now we have a number of variables in the second round virtually belonging to the 2 or 3 connected components of G_m. Note that adding the vertices corresponding to the second round subkey bytes described above does not increase the number of connected components. These can be seen as further reference points for the recovery of the remainder of the first subkey bytes by reducing the number of connected components in the new, larger graph.

Our thorough simulations show that such methods do increase the expected number of edges in the original graph G_m. Note that this improvement of our attack is not enough to decrease the number of measurements needed, though it increases the success probability of all our attacks for a fixed number of measurements.

4 Practical Feasibility

To make the detection of byte collisions during the S-box applications within AES possible, the AES implementation has to satisfy the property that all instances of the AES S-box are implemented in a similar way. The requirement is not necessary for [2] or [3]. This is the only difference of our technical framework with respect to that in [2] or [3]. Note that this requirement is very likely to be fulfilled in low-end real-world embedded systems, which are the main target of such attacks, since AES implementations in these systems are deliberately simplified by reducing diversity in order to save code size in software and area in hardware. On constrained 8-bit microcontrollers, the implementation of the AES S-box transform is likely to be a separate routine, thus, being exactly the same for all S-box applications.

As in standard collision attacks on AES, the attacker has to precisely know the times when the S-boxes leak. Note that this is not the case for DPA or similar differential techniques. Another advantage of the DPA method is that it works for the absolute majority of AES implementations including software and hardware ones. At the same time, the collision attacks on AES are mainly constrained to 8-bit software implementations on simple controllers.

However, the practical feasibility of collision attacks for AES was shown in [3] for a PIC16F687 microcontroller and in [2] for an i8051-type controller. To detect

a collision, the attacker compares the corresponding power curves using such basic techniques as correlation functions or more advanced wavelet methods [2].

Measurements of high accuracy are required to detect byte collisions. The usage of averaging techniques can improve the probability of correct collision detection in the cases where the implementation and the measurement setup do not allow for a reliable byte collision detection using a single power curve for each input. In this case, one cannot speak of a known-plaintext model any more, since the same plaintexts have to be input several times to increase the signal-to-noise ratio.

Note that our collision attack, as any other power analysis attack, can be significantly hampered or even made impossible by minimizing the signal-to-noise ratio, using sound masking techniques [8], [9] or advanced clock randomizing methods [10]. However, the collision attack is likely to break through basic time randomization countermeasures such as simple random wait states, which can be detected using SPA or alignment techniques.

5 Conclusions

In this paper we proposed and analyzed several improved side-channel collision attacks on AES. The first one requires 7 measurements and $2^{34.74}$ offline operations on average with a probability of 0.99. The second attack needs about 6 measurements and about $2^{37.15}$ offline operations with probability 0.854 or about $2^{44.3}$ operations with probability 0.927. The third one recovers the key with just 5 measurements and about $2^{37.34}$ simple offline operations with probability 0.372 or about $2^{45.5}$ operations with probability 0.548. This is to be compared to about 40 measurements required in the basic collision attack [2] on AES with some non-negligible post-processing, 29 measurements required for the AES-based Apha-MAC internal state recovery in [3] with about 2^{34} offline operations with a success probability > 0.5, and typically several hundred measurements for a classical DPA attack.

Acknowledgements. The author would like to thank Oxana Radetskaya for fruitful discussions during the work on this paper, Timo Kasper for providing some technical background about collision detection while working on another paper about collision attacks, the Horst-Görtz Institute for IT Security at the Ruhr-University of Bochum for financial support, and the anonymous referees for their comments that helped him to improve the paper.

References

1. Schramm, K., Wollinger, T.J., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
2. Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on AES: combining side channel- and differential-attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)

3. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision Attacks on Alpha-MAC and Other AES-based MACs. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727. Springer, Heidelberg (2007)
4. Ledig, H., Muller, F., Valette, F.: Enhancing collision attacks. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 176–190. Springer, Heidelberg (2004)
5. Sachkov, V.N.: Probabilistic Methods in Combinatorial Analysis. Encyclopedia of Mathematics and Its Applications, vol. 56. Cambridge University Press, Cambridge (1997)
6. Kolchin, V.F., Sevastyanov, B., Chistyakov, V.P.: Random Allocations. V. H. Winston & Sons (1978)
7. Karp, R.M., Tarjan, R.E.: Linear extected-time algorithms for connectivity problems. J. Algorithms 1 (1980)
8. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, Springer, Heidelberg (2005)
9. Oswald, E., Schramm, K.: An Efficient Masking Scheme for AES Software Implementations. In: Song, J., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, Springer, Heidelberg (2006)
10. Herbst, C., Oswald, E., Mangard, S.: An AES implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, Springer, Heidelberg (2006)