Two New Techniques of Side-Channel Cryptanalysis

Alex Biryukov and Dmitry Khovratovich

University of Luxembourg
{alex.biryukov,dmitry.khovratovich}@uni.lu

Abstract. We describe two new techniques of side-channel cryptanalysis which we call the *impossible collision attack* and the *multiset collision attack*. These are inspired by the state-of-the-art cryptanalytic techniques of impossible differential attacks [BBS99] and partial-function collision attacks [GM00] respectively. Using these techniques on an example of the AES we show that one has to mask all the rounds of a 128-bit key AES in order to prevent such attacks. For example these attacks can be used to break a recent proposal by Schramm et al. [SP06] of high order masking for the AES, since it protects only 3 external rounds.

1 Introduction

This paper was motivated by a recent work by Handschuh-Preneel [HP06] in which they have shown how to break through 4 rounds of DES masked with unique masking method [AG03] using differential cryptanalysis. Another motivation comes from the fact that masking methods like [AG03] or algebraic masking bring a high toll on the performance of the underlying cryptographic algorithm (either in terms of high memory requirements for tables, or in terms of slowdown for table-recalculation or complicated algebraic masking). Thus it is a natural question for the designer: how many rounds of a cipher need to be masked so that side-channel leakage from internal unmasked rounds becomes useless to the attacker. For example in the case of the AES one may think that masking two rounds at the top and at the bottom would be enough, since two rounds of this cipher have complete diffusion (i.e. each output bit is influenced by all the input bits). A prudent implementation would add an extra round for the safety-margin, and in fact one such proposal which masks three rounds from each end of the cipher has been proposed recently [SP06].

In this paper we describe two new tools for side-channel cryptanalysis, namely the *impossible collision attack* and the *multiset collision attack*. As one may infer from their names these are variations of the collision side-channel attack [SLFP04, LMV04] and are inspired by the state-of-the-art techniques of block-cipher cryptanalysis such as impossible differential attacks [BBS99] and partial-function collisions attacks [GM00] respectively. In all our attacks we use a standard collision attack assumption that the attacker can detect identical calculations being performed in different executions from similarity of the power consumption patterns of these executions.

Before we describe the new techniques we show a very efficient collision attack which needs only 70-80 measurements in order to break AES implementations with perfect masking of two top and bottom rounds. This attack is surprisingly efficient if one compares it with the other advanced collision attacks on unmasked implementations of the AES that require about 40 measurements.

Then we show how to attack AES with three round perfect masking (ex. the one from [SP06]) using impossible collision attack. This attack requires $2^{19} - 2^{20}$ measurements and 2^{27} offline curve comparisons for a full key recovery. The main idea is to detect an *absence* of collisions, which leaks information about the propagation of differentials inside the cipher. Such information in its turn is very useful in order to filter out the wrong key guesses.

Finally we show attacks using the multiset collisions techniques on three and four rounds of masking. The distinguisher for three rounds of masking has complexity of about 2^{18} measurements. The attack on the four rounds is probably of theoretical interest only since it requires $2^{27} - 2^{28}$ measurements and 2^{50} offline computation steps, but it would be feasible for a determined attacker. The benefit of the multiset approach over the impossible collisions is that detection of collisions in sets is much more robust to errors than detection of single byte collisions/differences.

These results show that in the case of 128-bit key AES all the rounds have to be masked for high security level, for 192-bit key AES 10 out of 12 rounds have to be masked and for 256-bit 10 out of 14 have to be masked. For a medium level of security 8 rounds of masking in total seem to be a minimum. Complexities of our attacks (in terms of required measurements and off-line computations) are presented in Table 1.

This paper is organized as follows: in Section 2 we give a brief overview of the side-channel attacks and countermeasures. In Section 3 we describe a general framework for the collision side-channel attacks. In Section 4 we describe a simple collision attack on 2-rounds of masking and then we present an impossible collision attack on 3 rounds of masking. In Section 5 we present a multiset collision attack based on Gilbert-Minier distinguisher for AES. In Section 7 we give our conclusions.

2 Brief Overview of Side-Channel Attacks and Countermeasures

Power attacks are based on measuring the power consumption of a device which performs some computations with the secret values, which the attacker tries to find. Whenever the internal value depends on the secret key the measurement may leak information about it. Although different leakage models were proposed [Mes00, BCO04, SSI05] the one that is based on the Hamming weight of the computed data is the most widely used. One may assume that the power consumption at any given moment is a normally distributed random variable with a mean depending on the Hamming weight of the processed data. This assumption is the basis of a technique called differential power analysis (DPA) [KJJ99, Mes00].

Further research has shown that some devices may leak more than a Hamming weight. More precisely, it is sometimes possible to determine whether a computed byte is the same in two different measurements. This attack uses partial internal collisions and thus it is called a collision attack [SWP03, SLFP04, LMV04].

The main idea is to encrypt/sign/hash different plaintexts and look for pairs with colliding bytes. If, say, in the AES, there is a partial collision after MixColumns transformation then by detecting the collision we obtain information about a subkey of this round. The number of collisions needed for the attack depends on the structure of the set of plaintexts. Schramm et al. in their attack on the AES managed to recover the full key after only 40 measurements (compared to thousands in some DPA attacks). On the other hand they assumed that an attacker has perfect ability to detect collisions, i. e., they assumed that probability of errors in collision detection is zero.

Side-channel techniques described above work only with deterministic implementation by default. Any randomization added to the implementation may defeat these methods. Let us imagine that all internal values are masked by some previously computed random values, which will be removed before the final step. Then any leakage gives us information about a masked value but not about the real one. Formally, we still know nothing about the processed values.

If masking is not completely random (say, inputs of all S-boxes of a round are masked by the same mask), which is sometimes the case, then one becomes able to obtain information about internal values [KJJ99, GT02, ABG04, WW04] via higher order DPA attacks. Furthermore, the implementation of uniformly random independent masking for all the rounds and each S-box in a round is in most cases too expensive.

In the case of the AES several mathematical as well as engineering masking methods were proposed. One of the most efficient methods [OMPR05] is based on the algebraic representation of an AES S-box. The S-box function, which is usually described as a function on $GF(256)$, may be interpreted as a function on $GF(16) \times GF(16)$. This is called "using composite fields" while the masking method is known as "algebraic masking". The inversion operation, which is the crucial point of the S-box transformation, can be implemented much more efficiently whenever we deal with smaller fields.

The complexity of many masking methods is prohibitively expensive either in terms of required memory or in terms of slowdown effect compared to the unmasked version or both. A thorough research on this topic may be found in Schramm's PhD thesis [Sch06]. We just cite that the implementation of algebraic masking method would require either 50 times as many computations as unmasked AES requires or about 64KB ROM for saving precomputed masked tables. The latter method is not completely suitable because these tables should be recomputed quite often; otherwise this will lead to successful higher order DPA.

It is also pointed out [Sch06] that the best trade-off between the amount of memory and the number of operations is to compute the masked table on the fly whenever it is needed. Nevertheless, masking all the rounds of the AES is still too costly, which is the reason to mask only a few first and a few last rounds. In the high-order resistant AES implementation by Schramm and Paar only the first three and the last three rounds are masked [SP06].

3 General Framework of Collision Attacks

Overall. Let f be an arbitrary non-injective function. Then a pair of different inputs a and b such that $f(a) = f(b)$ (i.e. a pair that provides non-injectiveness) is called a *collision* for f.

Although the notion of collision in cryptography is most often related to the design of hash-functions, it has been recently used to describe a specific class of side-channel attacks. In addition to being quite different from the DPA-style attacks, this class of attacks benefits from the usage of the birthday paradox.

More precisely, let V be an intermediate variable of an algorithm A implemented in a device:

$$V = f(P, K),$$

where P is an input of A and K is a secret parameter. Let us also assume that information about V leaks via the measurement of power consumption. Say, it is easy to detect the case

$$f(P_1, K) = f(P_2, K), \tag{1}$$

which is exactly a collision for f.

As soon as an adversary detects such collisions he obtains information about K. This fact may be easily described in the language of probability theory.

At first collision attacks were proposed for the DES [SWP03], where one could use non-injectiveness of the S-boxes. Then they were extended to the AES and other ciphers [SLFP04, LMV04, HP06]. In the AES, the non-injectiveness of MixColumns subfunctions is used. More precisely, each output byte of a round is a function of four input bytes. Evidently, there exist collisions even if only two input bytes are varied and two others are kept constant.

Detecting the difference. As soon as we are able to detect equality of values and thus collisions of type (1), we are likely to detect the inequality as well. Detection of impossibility of a collision in a certain byte resembles the cryptanalytic technique called "cryptanalysis with impossible differentials" [BBS99].

As only one of every 2^8 byte pairs collide, we expect to detect differences much more often than collisions. We will show in Section 4.2 and Appendix that mistaking differences for collisions might greatly reduce the probability of the attack to succeed.

Accuracy of the method. Schramm et al. in their attack on the AES [SLFP04] assumed perfect collisions detection. We expect that it is often not the case in the real world so in our attacks we allow some errors of both types (false positives and false negatives) to occur. Evidently, the number of measurements and the success of the attack depends on probabilities of these errors. We provide mathematical background on these issues in the Appendix and obtain formulas for the number of measurements and amount of data as the functions of the errors.

In order to eliminate the noise and thus reduce the probability of errors a standard approach would be to use averaging (by repeating the same measurement several times). The effect of variation in clock speed can be mitigated since it is often possible to distinguish rounds and even locate the moments of S-box computations. Furthermore we can compare several power consumption curves and find the best match using the FFT technique [WW04].

4 Impossible Collision Attack

In this section we show attacks on AES with two and three rounds of masking. We do not care which masking method is used in these rounds, we assume that it is perfect and no information is leaked from these rounds. We assume that the remaining inner rounds are unprotected, i.e. there are no software countermeasures in these rounds (the attack is robust against hardware countermeasures, such as random power noise (random peaks of power) or variable internal clock). These are the same assumptions as in [HP06].

4.1 Collision Attack on AES with 2 Perfectly Protected Rounds

If one considers unique masking method (UMM) or other costly masking technique for the AES at first sight it makes sense to mask only two top and two bottom rounds, since the AES (unlike the DES) has complete diffusion after just two rounds (i.e. change in a single bit influences all 128 bits). However in this section we describe a very efficient attack on the AES with only 2 top and 2 bottom round masking.

In this section we will perform a differential cryptanalysis [BS90] of the first two rounds of the AES, trying to predict differences at the input of the third round. Our goal is to make this as efficient as possible in order to minimize the number of queries to the encryption oracle and thus to minimize the number of measurements. As usual in differential cryptanalysis we consider only the propagation of differences and we call an S-box *active* if it gets non-zero input difference, otherwise we call an S-box *passive*. A pattern of difference propagation through several rounds of a cipher is called a *differential characteristic*, and a collection of characteristics with the same input and output differences, but with unrestricted behaviour in the intermediate rounds is called a *differential*.

We will use a differential which starts with four active S-boxes at the 1st round. We choose those active S-boxes to appear in positions which arrive in one column after the ShiftRows transformation. Then with probability 2^{-6} four active S-boxes will collapse to three (one byte out of four getting a zero difference). After the second round the three active bytes are expanded into 12 active bytes and there will still remain 4 passive bytes. This differential can be schematically described as $4 \to 3 \to 12$. We will use these bytes for the detection of collisions. This propagation is illustrated on the right in Figure 1.

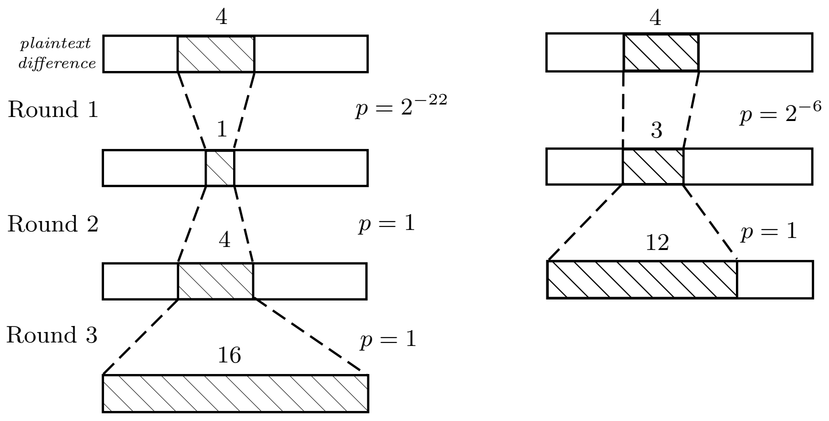


Fig. 1. Differentials for 3 and 2 rounds of the AES.

A naive implementation would require $2 \cdot 2^6$ measurements for 2^6 pairs of text until we detect a pair which has the differential property of 4 passive bytes (happens by chance with probability 2^{-32}). However we can use a standard trick from differential cryptanalysis and pack the texts into *structures*. The attacker will prepare a set of 24 texts which share the same random constant in the bytes corresponding to the passive S-boxes at the 1st round, and different randomly chosen values for the four active bytes. Then the attacker considers all possible $\sim 2^8$ pairs in this set. Thus in these structures of 24 texts there will be about four right pairs among about 256 wrong ones. If we can perfectly detect collisions at the input of the 3rd round then we will remain only with the right pairs (if we detect four collisions for the bytes corresponding to some column, this pair can only be a right pair). Each right pair reduces the key-space by 8 bits, and thus with four such pairs we know the 32-bit chunk of the key corresponding to the four active S-boxes of the first round. We can repeat this attack in four other locations completely recovering the subkey of the 1st round and thus the secret key of the AES as well. This attack needs less than 100 measurements. If we supplement the attack with partial exhaustive search of 2^{32} complexity the number of measurements becomes less than 75. Interestingly this complexity is close to the one obtained in collision attacks on unmasked AES implementations (40 measurements).

As a result of this attack it is clear that at least 3 rounds of AES have to be masked at both ends. It may also seem sufficient. Nevertheless in the next subsection we show how to penetrate through 3 rounds of masking for AES. However this requires a new side-channel attack technique.

4.2 Impossible Collision Attack on 3 Rounds of AES with Masking

Now we are in a scenario when three rounds of AES are masked at the top and at the bottom.

Let us see what happens if we use the differential as shown on the left side of Figure 1: $4 \rightarrow 1 \rightarrow 4 \rightarrow 16$. We see that by the properties of AES diffusion the four active bytes which enter the 3rd round, will be spread to four different columns and after application of the MixColumns all 16 bytes of the internal state of AES will be active. That means that we are guaranteed that there will be no collisions at the input to the 4th round. Thus it seems that we can not use the collision detection technique anymore, since there will be **no collisions**. However we can try to use precisely this property in order to build the attack. The pairs which do not follow the differential will occasionally have collisions in some bytes, at the input of the 4-th round. If we can reliably detect those byte collisions we can immediately discard all such pairs, since we know that there can be no collisions in a right pair. This idea is very similar to how wrong pairs are filtered in impossible differential cryptanalysis. Since we use not the collisions but absence of them as a distinguisher, we call this technique *an impossible collision attack*.

Probability that at least one byte out of 16 is the same (at the input of the 4th round) in a pair of AES encryptions is $1-(255/256)^{16} \approx 1/16$. This is clearly a very weak filter, 15 out of every 16 wrong pairs will go through and thus we need some way to amplify the filtration. For each pair that is being tested we can construct its *variation*, in which we keep the values that enter active S-boxes intact, and vary only the constants that enter the passive S-boxes. After 16 such trials the chance for a wrong pair to survive all the tests is about 0.36. We repeat the test for the remaining pairs again and again, etc. At the end we will be left only with the right pair which exhibits no collisions. Given a right pair we recover 22 bits of subkey information of the 1st round similar to an attack from the previous section. In order to use the measurements more efficiently we will again pack the data into structures, but this time of size 2^{12}, each structure providing 2^{23} pairs and thus about two right pairs (probability of the 1st round differential is 2^{-22}). We will need 2^7–2^8 such structures since filtration is much slower than in the previous section.

The total complexity of this attack is as follows. We will need about 2^7–2^8 times more structures than in the previous attack, since our filtration is weaker. Thus we will need a total of 2^{19}–2^{20} measurements. The analysis phase will have complexity of about $16 \cdot 2^{23} = 2^{27}$ curve comparisons.

This attack is relatively robust to the errors in which we miss collisions, since we still will have a chance to filter such pair in next trials. However this attack is sensitive to the errors of the second kind in which we miss the difference and detect a collision where there is none. Such error can filter out a right pair and thus we need to ensure that the set of pairs contains enough right pairs, so that at least one remains after the filtration process.

In the appendix we derive the formula which relates the error probability with the total number of required pairs and the probability of the differential.

5 Multiset Collision Attack on 4 Masked Rounds

5.1 Three round distinguisher

Our idea is to use an efficient 3-round distinguisher[1] for the AES proposed by Gilbert and Minier in [GM00].

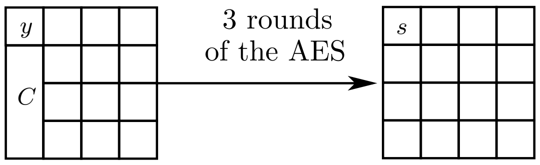
We use the following notation for bytes of the AES internal state:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Let us denote by y the byte a_{00} in the first round input, and by C the block of three bytes a_{10}, a_{20}, a_{30} in the same column. Now fix the other bytes to any 12-tuple of constant values. Let us also denote by s the byte a_{00} in the input to the fourth round. Then the following proposition holds.

Proposition 1 ([GM00]). *The value of s is fully determined by a secret key K, the byte y, the set C, and 12 fixed bytes. Furthermore, its value may be expressed as a key-dependent function of one byte-value function of y and four byte-value functions of C:*

$$s = F_K(f_K(y), g_K^1(C), g_K^2(C), g_K^3(C), g_K^4(C)). \tag{2}$$



Due to the birthday paradox, a pool of 2^{16} distinct sets C contains triplets C' and C'' such that $g_K^i(C') = g_K^i(C'')$, $i = \overline{1,4}$, which lead to the following

[1] It is natural to try to apply Square attack distinguisher to side-channel attacks, however one quickly discovers that detection of a *balanced* property requires very high precision from the attacker in the course of many measurements, and thus is not robust to even small amounts of noise.

equality:

$$s^{C'}[y] \equiv s^{C''}[y]. \tag{3}$$

Gilbert and Minier also noted that much fewer than 256 different y is usually enough for verifying this property. For our purposes only 6 different y's will suffice. Indeed, a wrong pair of triplets collides in one point (byte) of the function s with probability 2^{-8} so 6 y's will filter all wrong pairs, except for a fraction of 2^{-48}.

Thus we obtain a *distinguisher* for three rounds of the AES that works as follows. An adversary chooses a set of $6 \cdot 2^{16} \approx 2^{18.5}$ plaintexts with 2^{16} different C's and 6 y's for each set C. Then he encrypts those texts and checks for collisions in the input of fourth round. A right pair provides 6 collisions while on average no wrong pair (out of 2^{31}) will survive this test. Complexity of this distinguisher is comparable to the complexity of our impossible collision attack, however it seems more robust, since it detects collisions rather than the absence of them and can be easily strengthened by increasing the size of the set. This flexibility can be very useful in the presence of the countermeasures. Now we will show how to use this distinguisher to construct an attack on the four masked rounds.

5.2 Attack

To pass through four masked rounds we add a round in the beginning. Now we have to guess 4 bytes of the first subkey that affect y and C. We propose two approaches to detect whether a tested subkey is the right subkey. The first one requires 2^{32} measurements and $2^{44.5}$ offline steps. The second one requires fewer measurements but the overall complexity is substantially higher.

Attack 1. We encrypt all the 2^{32} plaintexts that vary only in bytes a_{00}, a_{11}, a_{22}, a_{33} of the first round and save measurements. Thus before the second round we have 2^{32} intermediate states that vary only in the first column. Let us denote this set by D.

As we are able to detect collisions, we divide D by 2^8 distinct groups (let us denote them in B_i, $i = \overline{1, 256}$) with regard to corresponding power consumption curves. A pair of plaintexts from the same group is highly likely to collide in s.

Next for each key guess of bytes $k_{00}, k_{11}, k_{22}, k_{33}$ we do the following.

– Choose from D 2^{16} distinct sets $D_c = (P_1, P_2, \ldots, P_6)$ such that c is the common C-triplet for P_i, and y byte is equal to i. Thus every D_c is exactly a verifying set that we use for 3-round distinguisher with 6 different y's.
– For each D_c fill the vector $V_c = (j_1, j_2, \ldots, j_6)$ such that $P_i \in B_{j_i}$.
– Search for vectors $V_{c'} = V_{c''}$. Such a pair implies a functional collision of type (3). A right key guess will lead to at least one such pair.

Complexity 1. The first two steps require $6 \cdot 2^{16}$ 1-round AES transformations if we do them straightforwardly. To reduce the complexity we first decrypt the $y \circ C$ columns ($6 \cdot 2^{16}$ ones) using the zero subkey. Thus we obtain D_c for the zero subkey.

To avoid the most of recomputations we arrange the set of subkeys such that the computation of new D_c's requires fewer operations. A good example is provided by the Gray code [Knu04] where the next codeword differs only in a single bit. Thus we need only $6 \cdot 2^{16}$ bit flips for each new subkey. The usage of specific data structures might give us 2^{16} byte complexity.

In the last step we have to find a colliding pair of vectors V. We sort them and then search for the pairs. Overall we need about 2^{16} simple operations for each key guess, i.e. 2^{48} offline steps or about 2^{43} one AES round steps to recover the 32-bit chunk of the key. In order to complete the key recovery we can repeat the attack for other byte positions 3 times, thus finding 96 bits of the first round subkey and complete the attack by 2^{32} exhaustive search steps. Thus the complexity of the off-line part is about $2^{44.5}$ one AES round steps. For the online part we need about 2^{32} measurements. Due to high on-line and off-line complexity this attack is mainly of theoretical interest, though it is on the verge of practicality for a determined attacker.

Attack 2. Now we encrypt $2^{29.5}$ (1/6th of the number of measurements used by the previous attack) distinct plaintexts (we follow the notations for the first attack) that vary only in bytes $a_{00}, a_{11}, a_{22}, a_{33}$ of the first round.

For each key guess of bytes $k_{00}, k_{11}, k_{22}, k_{33}$ we do the following.

– Choose also 2^{16} distinct sets D_c but of different cardinality. On average each D_c contains $m = 2^{5.5}$ plaintexts. The problem is that the set of y's depends on a D_c. Nevertheless, we expect that for each pair c', c'' there are about $m^2/256 = 8$ common y's.
– For each $D_c = (P_1, P_2, \ldots, P_m)$ fill the vector

$$V_c = \big(\langle y_1, j_1\rangle, \langle y_2, j_2\rangle, \ldots, \langle y_m, j_m\rangle\big)$$

  such that
  - If we encrypt P_i using subkey k then the value of y-byte (input to the second round) is y_i;
  - $P_i \in B_{j_i}$.

– Search for pairs such that common y's imply common j's. This means we found a functional collision of type (3). A right key guess will lead to at least one such pair.

Complexity 2. Now we estimate the number of operations for each key guess.

We use the same technique to reduce the complexity of the first two steps. The only difference is that we first have to decrypt all the possible columns because we do not know what y's we will need further. We also have to filter this set of $2^{16} \cdot 2^8 = 2^{24}$ diagonals by our predefined set of plaintexts (D), since only 1/6 of the diagonals will be present in our measurement sets. Thus the complexity for the zero subkey is about 2^{30} byte-operations while for the other ones we need 2^{24} bit operations or 2^{21} byte ones.

Unfortunately, the problem of V_c vector matching is somewhat more complicated. For each value of c only about 40 plaintexts would be present in our measurement set (i.e. we get 2^{16} V_c vectors with 40 random entries "turned on" in each). Due to the birthday paradox any pair of vectors would typically have about 6 common indexes. If all the y_i's at common locations coincide — this is a multiset collision that we are looking for. If at least one of the coordinates differs — the pair is a wrong pair. The problem is that common index locations differ for different pairs. Thus a naive algorithm would be to try all possible $(2^{16})^2 = 2^{32}$ pairs of vectors which would bring the total complexity of the attack to about 2^{64} steps.

A faster approach would be as follows: for the pair c', c'' which produce a multiset collision there exist at least 6 common index locations in which the y_i's are the same. Thus if we try about 40 index locations at random one of them will likely be the index location in which the right pair has a common index and is colliding. At a specific index location i only about $2^{16}/6 = 2^{13.5}$ constants c will be active (i.e. will have a corresponding measurement). We can thus distribute the measurements in location i into 2^8 buckets according to the y_i. Each value will appear about $2^{5.5}$ times. Only the measurements that fell in one bucket need to be further checked for the presence of collisions at other common locations. Here we can simply check all possible pairs in a bucket in 2^{11} vector comparisons which require about 8 simple operations till we get the next common location at which in most cases there will be a c ontradiction. We have 2^8 buckets to check, thus the total complexity for one key guess is $40 \cdot 2^8 \cdot 2^{11} \cdot 8 = 2^{27}$ simple comparisons, or 2^{22} one AES round steps. Due to a high off-line complexity this attack is mainly of theoretical interest.

6 Attacks on the bottom rounds

As soon as an adversary is allowed to decrypt ciphertexts he may perform an attack on the bottom round of AES. Although the last round does not contain MixColumns transformation, the number of S-box layers we pass through is the same.

Futhermore, AES might (and is encouraged to) be implemented such that the decryption steps are exactly the same as those of encryption. This is gained by swapping MixColumns $\leftrightarrow$ AddRoundKey and swapping SubBytes $\leftrightarrow$ ShiftRows. Such an implementation is of the same security against a decrypting adversary as well as encrypting one.

Thus all the listed attacks may be succesfully applied also to the bottom rounds with the same complexity.

7 Conclusion

In this paper we have shown two new techniques of side-channel cryptanalysis based on collision detection: an *impossible collision attack* and a *multiset collision attack*. Using these techniques we show how to break AES implementations with 2, 3 and 4 round masking at both ends. One example of such implementation is [SP06]. This shows that in total at least 10 AES rounds out of 10, 12, or 14 (depending on the key-size) have to be masked in order to prevent such attacks. The overall complexities (in terms of required measurements and off-line computations) are presented in Table 1. The off-line complexities of single and impossible collision attacks are measured in curve comparisons while those of the other attacks are measured in one AES round executions.

Method	**Complexity**		**Masked**	**Attack**
	Measurements	Off-line	**rounds**	
Simple collisions	2^6–2^7	2^7	2	Full key recovery
Impossible collisions	2^{19}–2^{20}	2^{27}	3	Full key recovery
MultiSet collisions	$2^{18.5}$	2^{20}	3	Distinguisher
MultiSet collisions	2^{32}	$2^{44.5}$	4	Full key recovery
MultiSet collisions	2^{29}–2^{30}	2^{54}	4	32 key bits recovery

Table 1. Complexities of our attacks on partially masked AES implementations.

Acknowledgements

We wish to thank Jean-Sébastien Coron for fruitful discussions and anonymous referees for their helpful comments. Dmitry Khovratovich is supported by PRP "Security & Trust" grant of the University of Luxembourg.

References

[ABG04] Mehdi-Laurent Akkar, Régis Bevan, and Louis Goubin, *Two Power Analysis Attacks against One-Mask Methods*, FSE 2004, LNCS, vol. 3156, Springer, 2004, pp. 332–347.

[AG03] Mehdi-Laurent Akkar and Louis Goubin, *A Generic Protection against High-Order Differential Power Analysis*, FSE 2003, LNCS, vol. 2887, Springer, 2003, pp. 192–205.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, EUROCRYPT 1999, LNCS, vol. 1592, Springer, 1999, pp. 12–23.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier, *Correlation Power Analysis with a Leakage Model*, CHES 2004, LNCS, vol. 3156, Springer, 2004, pp. 16–29.

[BS90] Eli Biham and Adi Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, CRYPTO 1990, LNCS, vol. 537, Springer, 1990, pp. 2–21.

[GM00] Henri Gilbert and Marine Minier, *A Collision Attack on 7 Rounds of Rijndael*, AES Candidate Conference, 2000, pp. 230–241.

[GT02] Jovan Dj. Golic and Christophe Tymen, *Multiplicative Masking and Power Analysis of AES*, CHES 2002, LNCS, vol. 2523, Springer, 2002, pp. 198–212.

[HP06] Helena Handschuh and Bart Preneel, *Blind Differential Cryptanalysis for Enhanced Power Attacks*, Selected Areas of Cryptology 2006, in appear, 2006.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun, *Differential Power Analysis*, CRYPTO 1999, LNCS, vol. 1666, Springer, 1999, pp. 388–397.

[Knu04] Donald E. Knuth, *The Art of Computer Programming, Volume 4A: Enumeration and Backtracking*, ch. Generating all n-tuples, 2004.

[LMV04] Hervé Ledig, Frédéric Muller, and Frédéric Valette, *Enhancing Collision Attacks*, CHES 2004, LNCS, vol. 3156, Springer, 2004, pp. 176–190.

[Mes00] Thomas S. Messerges, *Using Second-order Power Analysis to Attack DPA Resistant Software*, CHES 2000, LNCS, vol. 1965, Springer, 2000, pp. 238–251.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen, *A Side-Channel Analysis Resistant Description of the AES S-Box*, FSE 2005, LNCS, vol. 3557, Springer, 2005, pp. 413–423.

[Sch06] Kai Schramm, *Advanced Methods in Side Channel Cryptanalysis*, Ph.D. thesis, University of Bochum, 2006.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar, *A Collision-Attack on AES: Combining Side Channel- and Differential-Attack*, CHES 2004, LNCS, vol. 3156, Springer, 2004, pp. 163–175.

[SP06] Kai Schramm and Christof Paar, *Higher Order Masking of the AES*, CT-RSA 2006, LNCS, vol. 3860, Springer, 2006, pp. 208–225.

[SSI05] Daisuke Suzuki, Minoru Saeki, and Tetsuya Ichikawa, *DPA Leakage Models for CMOS Logic Circuits*, CHES 2005, LNCS, vol. 3659, Springer, 2005, pp. 366–382.

[SWP03] Kai Schramm, Thomas J. Wollinger, and Christof Paar, *A New Class of Collision Attacks and Its Application to DES*, FSE 2003, LNCS, vol. 2887, Springer, 2003, pp. 206–222.

[WW04] Jason Waddle and David Wagner, *Towards Efficient Second-Order Power Analysis*, CHES 2004, LNCS, vol. 3156, Springer, 2004, pp. 1–15.

A Amount of data as a function of error probability

Let us denote by p the probability of the differential ($= 2^{-22}$ in our case) and by M the number of tested pairs. Thus we can approximate the number of right pairs by pM. After a test about αpM good pairs and βM bad pairs survive, where α is the probability of a right pair to survive while β is that of a wrong pair (we will show how to estimate them later).

Now recall that we do the variation so each pair has, say, t sisters, which has the same right/wrong property. After a test we have about $\alpha^t pM$ good pairs and $\beta^t M$ bad pairs survived. A simple sufficient condition for the attack to succeed is the elimination of the wrong pairs[2] while at least one right pair

$$\begin{cases} \beta^t M < 1; \\ \alpha^t pM = 1. \end{cases} \tag{4}$$

[2] Note that one may not need complete elimination of the wrong pairs. As soon as the faction of right/wrong pairs doubles we can claim that we have gained a single bit of key information.

Exponentiating the second expression by the $\log_\alpha \beta$ and dividing it to the first one, one may easily obtain the condition on M:

$$M > \left(\frac{1}{p}\right)^{\log_{\beta/\alpha} \beta}.$$

It implies simple conditions on α and β:

$$\beta < \alpha \le 1, \tag{5}$$

which has an obvious interpretation. There is no information gain if the inequality does not hold.

A formulae for the number of measurements (t) may be easily obtained from (4):

$$t = \max(\log_{1/\beta} M,\, \log_{1/\alpha}(pM)).$$

Now we show how to estimate α and β a priori. Imagine we evaluated a priori errors of detecting collisions in a single byte. We are interested in two of them:

α_B — the probability of a difference in a single byte to be recognized; (6)

β_B — that of a collision in a single byte to be missed. (7)

Schramm et al. in [Sch06] have the first one to be 1 while the second one is zero. We show how to deal with a more complicated case. Let c_i of all wrong pairs collide in i bytes ($i = \overline{1,16}$). To survive, no collisions should be detected while all differences should be recognized. Thus every wrong pair with i collisions survives with the probability $\beta_B^i \cdot \alpha_B^{16-i}$. Finally, $\sum c_i \beta_B^i \alpha_B^{16-i}$ of wrong pairs survive. This coefficient is exactly β so we have the following equality:

$$\beta = \sum c_i \beta_B^{16} \alpha_B^{16-i} = \sum (1-1/256)^i (1/256)^{16-i} \beta_B^i \alpha_B^{16-i} = \left(\frac{255}{256}\beta_B + \frac{\alpha_B}{256}\right)^{16}.$$

As a right pair survives with the probability α_B^{16}, (5) may be rewritten as follows:

$$\left(\frac{255}{256}\beta_B + \frac{\alpha_B}{256}\right)^{16} < \alpha_B^{16} \iff \beta_B < \alpha_B.$$

As a result, the sufficient condition on the attack to succeed is the fact that the probability of a difference in a single byte to be recognized is more than that of a collision to be missed.