Principles on the Security of AES against First and Second-Order Differential Power Analysis*

Jiqiang Lu[1], Jing Pan[1,2], and Jerry den Hartog[1]

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands
lvjiqiang@hotmail.com, {j.pan,j.d.hartog}@tue.nl
[2] Riscure BV, 2628 XJ Delft, The Netherlands
pan@riscure.com

Abstract. The Advanced Encryption Standard (AES) is a 128-bit block cipher that is currently being widely used in smartcards. Differential Power Analysis (DPA) is a powerful technique used to attack a cryptographic implementation in a resource-limited application environment like smartcards. Despite the extensive research on DPA of AES, it seems none has explicitly addressed the fundamental issue: How many rounds of the beginning and end parts of an AES implementation should be protected in order to resist practical DPA attacks, namely first and second-order DPA attacks? Implementation designers may think that it is sufficient to protect the first and last one (or one and a half) rounds of AES, leaving the inner rounds unprotected or protected by simple countermeasures. In this paper, we show that power leakage of some intermediate values from the more inner rounds of AES can be exploited to conduct first and/or second-order DPA attacks by employing techniques such as fixing certain plaintext/ciphertext bytes. We give five general principles on DPA vulnerability of unprotected AES implementations, and then give several general principles on DPA vulnerability of protected AES implementations. These principles specify which positions of AES are vulnerable to first and second-order DPA. To justify the principles, we attack two recently proposed AES implementations that use two kinds of countermeasures to achieve a high resistance against power analysis, and demonstrate that they are even vulnerable to DPA. Finally, we conclude that at least the first two and a half rounds and the last three rounds should be secured for an AES implementation to be resistant against first and second-order DPA in practice.

Keywords: Side channel cryptanalysis, Advanced Encryption Standard, Differential power analysis.

1 Introduction

Nowadays, smartcards are widely used in many real-life security applications as a medium of authenticating the identity of a user and/or executing cryptographic

* This work was supported by the Dutch Sentinels project PINPASJC TIF.6687.

J. Zhou and M. Yung (Eds.): ACNS 2010, LNCS 6123, pp. 168–185, 2010.

operations. Power analysis [1] is well known as a practical threat to the security of a cryptographic implementation running in a resource-constrained application environment such as a smartcard. It is based on the fact that the power consumption during an execution of the implementation reveals some information about the data being processed. Primarily, there are two types of power analysis: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). An n-th order DPA attack extracts secret key information by analysing the correlation between the secret key and n points of the power consumption, and it works under the fundamental hypothesis [2]: *There exists a set of n intermediate variables that appear during execution of the algorithm, such that guessing a few key bits (in practice less than 32 bits) allows us to decide whether or not two inputs give the same value for a known function of these n variables.* Though the computing power has been increasing, guessing 32 or more bits is still considered to be unpractical in the context of DPA on smartcards. Up to now, only first and second-order DPA attacks have been experimentally demonstrated to be feasible and efficient in practice, see [3, 1, 4, 5]. Thus, it is a minimum requirement for a cryptographic implementation to be resistant against first and second-order DPA in practice.

The Advanced Encryption Standard (AES) [6] is a 128-bit block cipher with a user key of 128, 192 or 256 bits. It was released by NIST in 2001 as the new-generation data encryption standard for use in the USA, and was adopted as an ISO [7] international standard in 2005. Being used more and more widely in reality, AES is one of the most promising cryptographic algorithms for smartcards, and thus how to design a secure and efficient AES smartcard implementation is of great interest to both industry and academia. A variety of solutions have been presented during the past several years, for example, [10,11,12,8,13,14,9]. One of the most widely used software countermeasures is the Boolean masking method, as followed in [10,15,16,12,17,14]. The main idea of this method is to mask any sensitive intermediate value by XORing it with one or more randomly generated values called masks. Another kind of software countermeasures is hiding, such as randomizing the operations of an algorithm [18].

It has been known that an attacker can exploit the SubBytes operation of the first or last round of AES to conduct a first-order DPA attack. In 2006, Jaffe [19] described a first-order DPA attack exploiting the SubBytes operation of Round 2. Despite the extensive research on the protection of AES against DPA, to our best knowledge, it seems that none has explicitly addressed the fundamental issue: How many rounds of the beginning and end parts of an AES implementation should be protected in order to resist practical DPA attacks, namely first and second-order DPA attacks? To achieve a good tradeoff between security and efficiency, some implementation designers may think that it is enough to protect the first and last one (or one and a half) rounds of AES to thwart first and second-order DPA attacks, leaving the inner rounds unprotected or protected by simple countermeasures, e.g. [8, 9]. The reason that the beginning and end parts of AES are particularly protected is that they are more vulnerable to DPA attacks, because an intermediate value from there depends on a relatively small

fraction of the key, and an intermediate value from the inner rounds depends on a group of 32 or more key bits, due to the diffusion and confusion properties of the MixColumns operation.

In this paper, we focus on the security of the AES with 128 key bits against DPA. Taking advantage of several simple techniques, such as fixing certain plaintext/ciphertext bytes, we exploit some intermediate values from the inner rounds of AES to conduct first and second-order DPA attacks. We summarise them as five general principles for first and second-order DPA on unprotected AES implementations; some simple cases in the principles have already been known to the public, but more cases are presented for the very first time. We then derive several general principles for first and second-order DPA on protected AES implementations. The principles suggest that when securing an AES implementation against first and second-order DPA one should well protect at least the first two and a half rounds and the last three rounds, i.e. from the beginning until the MixColumns operation of the third round and from the beginning of the eighth round to the end. As examples, we apply the general principles to attack Herbst et al.'s and Tillich et al.'s AES software implementations [8,9], and show that they are not secure against first and/or second-order DPA attacks. An innovative point for Herbst et al.'s and Tillich et al.'s implementations is that they use two kinds of countermeasures, namely randomization (operation shuffling) and masking, to make themselves highly resistant against power analysis.

The remainder of this paper is organised as follows. In the next section we give the notation and briefly review the AES cipher. In Section 3, we present the general principles on first and second-order DPA of unprotected and protected AES implementations, and explain these principles in detail in Section 4. In Section 5, we provide justifications by applying the general principles to Herbst et al.'s and Tillich et al.'s AES implementations in practice. In Section 6 we compare our result with related work. In Section 7 we give several principles on the protection of AES against first and second-order DPA. Section 8 concludes the paper.

2 Preliminaries

In this section we describe some notation and the AES block cipher.

2.1 Notation

In the following descriptions, we assume that a number without a prefix is in decimal (base 10) notation, and a number with prefix $0x$ is in hexadecimal (base 16) notation. The sixteen bytes of a 4×4 byte array are numbered using the conventional row-column fashion, starting with (0,0) (i.e. (0,0), (0,1), $\cdots$, (3,3)). We use the following notation.

- $\oplus$ bitwise logical exclusive OR (XOR) of two bit strings of the same length
- $*$ polynomial multiplication modulo the polynomial $x^8 + x^4 + x^3 + x + 1$

2.2 The AES Block Cipher

The AES [6] cipher takes as input a 128-bit plaintext block P, represented as a 4×4 byte array, and has a total of 10 rounds. It uses the following four elementary operations to construct the round function.

- The AddRoundKey operation XORs a 4×4 byte array with a 16-byte subkey.
- The SubBytes operation applies the same 8×8-bit bijective S-box (denoted below by S) 16 times in parallel to a 4×4 byte array.
- The ShiftRows operation cyclically shifts the jth row of a 4×4 byte array to the left by j bytes, $(0 \leq j \leq 3)$.
- The MixColumns operation pre-multiplies a 4×4 byte array by a fixed 4×4 byte matrix $MC = (m_{i,j})$; see [6] for the specifications of the matrix MC and its inverse $MC^{-1} = (m'_{i,j})$.

The encryption procedure is, where X is a 16-byte variable, K_0, K_i and K_{10} are 16-byte round keys, and C denotes the ciphertext.

1. $X = \text{AddRoundKey}(P, K_0)$.
2. For $i = 1$ to 9:
   $X = \text{SubBytes}(X)$,
   $X = \text{ShiftRows}(X)$,
   $X = \text{MixColumns}(X)$,
   $X = \text{AddRoundKey}(X, K_i)$.
3. $X = \text{SubBytes}(X), X = \text{ShiftRows}(X), C = \text{AddRoundKey}(X, K_{10})$.

The ith iteration in Step 2 in the above description is referred to below as Round i, and the transformation in Step 3 is referred to below as the final round (i.e. Round 10). We write $K^i_{j,h}$ for byte (j, h) of K_i, $(0 \leq j, h \leq 3, 0 \leq i \leq 10)$.

3 Principles on First and Second-Order DPA of AES

In this section, we first give five general principles for DPA attacks on unprotected AES implementations, and then derive a few general principles on protected AES implementations. These principles are obtained under the fundamental hypothesis given in Section 1. Recall that a DPA attack is, under the fundamental hypothesis, considered to be unpractical if it requires guessing 32 bits or more. We remind the reader that more inner rounds could be similarly attacked if it was feasible to guess 32 or more bits in some environments.

3.1 Principles for Unprotected AES Implementations

Below are the five general principles for first and second-order DPA on plain AES implementations without countermeasure, which are categorized by the location of the intermediate byte(s) exploited. Details of the principles are given in Section 4.

i. Any intermediate byte before the MixColumns operation of Round 3 can be exploited to conduct a first-order DPA attack. The attack uses a number of plaintexts with 0, 3 or 15 bytes fixed, depending on the location of the exploited byte.
ii. Any intermediate byte after the AddRoundKey operation of Round 7 can be exploited to conduct a first-order DPA attack. The attack uses a number of ciphertexts with 0, 3 or 15 bytes fixed, depending on the location of the exploited byte.
iii. Any two intermediate bytes before the SubBytes operation of Round 3 can be exploited to conduct a second-order DPA attack, if their XOR value is plaintext-dependent[1]. The attack uses a number of plaintexts with 0, 1, 2, 3, 4, 7, 11 or 15 bytes fixed, depending on the location of the two exploited bytes.
iv. Any two intermediate bytes after the SubBytes operation of Round 8 can be exploited to conduct a second-order DPA attack, if their XOR value is ciphertext-dependent. The attack uses a number of ciphertexts with 0, 2, 3, 4, 7 or 15 bytes fixed, depending on the location of the two exploited bytes.
v. Any intermediate byte before the MixColumns operation of Round 2 and any intermediate byte after the MixColumns operation of Round 8 can be exploited to conduct a second-order DPA attack. The attack uses a number of plaintexts with 0, 3 or 4 bytes fixed and their ciphertexts with 0, 4 or 3 bytes fixed, depending on the location of the two exploited bytes.

To apply these DPA attacks in practice one needs to acquire the power traces of plaintexts and/or ciphertexts with the required properties, which depends on specific application environments. These DPA attacks work under four attack scenarios, namely, a ciphertext-only scenario, a known-plaintext scenario, a chosen-plaintext scenario, and an adaptive chosen-ciphertext scenario. The simple cases in the principles, like those exploiting the SubBytes operation of the first or last round, requires a known-plaintext or ciphertext-only scenario. Obtaining plaintexts with a few bytes fixed is simple, which can be easily done under a chosen-plaintext scenario. Obtaining ciphertexts with a few bytes fixed is a little sophisticated. It usually requires an (adaptive) chosen-ciphertext scenario. For instance, an attacker can first choose a number of ciphertexts with certain bytes fixed, and then decrypts them under the attacked key. This usually requires an AES decryption implementation available, and in this case the attacker may choose to attack the decryption implementation using the chosen ciphertexts. Anyway, as demonstrated by Bleichenbacher [20], an adaptive chosen-ciphertext attack is feasible in some real world applications. To obtain plaintexts with a few bytes fixed and their ciphertexts with certain bytes fixed, an attacker can use the following way under a chosen-plaintext scenario: first encrypt a number of plaintexts with certain bytes fixed, and then look for only those ciphertexts whose concerned bytes are identical (plaintexts not meeting this property can be discarded, which causes a high data complexity). Since a

[1] The notion 'plaintext-dependent' means that the XOR value involves some plaintext byte(s). Similar for the following notion 'ciphertext-dependent'.

large amount of computations are needed in order to obtain the useful texts, this attack may only be applicable to cryptographic devices that have a strong computation ability. In the rest of the paper we mainly focus on the possibility of launching an effective DPA attack when the required texts are available.

The principles also consider attacks where 15 bytes of the plaintexts or ciphertexts are fixed and the remaining byte is random. Thus there remain only 256 different plaintexts/ciphertexts available, and a remaining problem is how to obtain sufficient power traces. A simple way to do so is to encrypt/decrypt every available plaintext/ciphertext many times, and then to use the obtained power traces. This is applicable for some devices in practice, particularly considering that a first-order DPA attack can break an AES S-box implementation using as few as 100 or even 30 power traces (see [21, 22, 23]). The technique of fixing a few plaintext and/or ciphertext bits has been used in differential power analysis of some implementations for the DES [24] block cipher [16, 26, 25] and AES [19], as well as in block cipher cryptanalysis, [27] say.

3.2 Principles for Protected AES Implementations

From Principles (i)–(v), we can easily get the following two principles for a protected AES implementation:

vi. If a byte concerned by Principle (i) or (ii) is unprotected, then it can be exploited to conduct a first-order DPA attack, no matter how well the other parts are protected.
vii. If two bytes concerned by Principle (iii), (iv) or (v) are unprotected or protected only by the same mask, then they can be exploited to conduct a second-order DPA attack, no matter how well the other parts are protected.

Therefore, a protected AES implementation would be vulnerable to a first or second-order DPA attack if it has either of the above weaknesses, no matter how well the other parts are protected. Herbst et al.'s and Tillich et al.'s AES implementations are such examples as to be attacked in Section 5.

4 Principle Details

In this section we explain the principles in Section 3.1 in detail.

4.1 Explaining Principles (i) and (ii)

Simple cases in Principles (i) and (ii) have been well known, such as those exploiting a byte immediately before the SubBytes operation of Round 10. Here we will explain two representative cases: a moderate case in Principle (ii) which exploits a byte immediately before the AddRoundKey operation of Round 8, and the most difficult case in Principle (i) which exploits a byte immediately after the SubBytes operation of Round 3. The reasoning is similar for the other cases of Principles (i) and (ii).

Attacking the AddRoundKey Operation in Round 8. Let v be the exploited byte immediately before the AddRoundKey operation in Round 8. There is no MixColumns operation in the last round of AES, and a byte immediately before the AddRoundKey operation of Round 8 depends on a group of 9 key bytes (one from K_8, four from K_9, and the remaining four from K_{10}) when we express it using relevant ciphertext bytes. Thus, a naive first-order DPA attack exploiting the byte v would need to guess more than 32 secret key bits, which is infeasible under the fundamental hypothesis. Nevertheless, as described subsequently, we find that this limitation can be circumvented by using the technique of fixing a few ciphertext bytes.

The XOR of v and the corresponding byte of K_8, denoted by k_0, is input to an S-box in Round 9; and let u be the output of the S-box. Thus, we have $v = k_0 \oplus S^{-1}(u)$. The following ShiftRows operation does not change the value of u. Then, u is used as a part of the input to the MixColumns operation in Round 9, and we refer to the resulting output column (four bytes) as (u_0, u_1, u_2, u_3). Hence, u can be expressed as follows:

$$u = m'_0 * u_0 \oplus m'_1 * u_1 \oplus m'_2 * u_2 \oplus m'_3 * u_3 \,,$$

where m'_0, m'_1, m'_2, m'_3 are relevant elements from MC^{-1}. Subsequently, (u_0, u_1, u_2, u_3) is used to generate four ciphertext bytes, where four bytes of K_9 and four bytes of K_{10} are involved. We denote the involved four bytes of K_9 as (k_1, k_2, k_3, k_4), the involved four bytes of K_{10} as (k_5, k_6, k_7, k_8), and the resulting four ciphertext bytes as (c_0, c_1, c_2, c_3). Thus, the four bytes u_0, u_1, u_2, u_3 can be expressed as,

$$u_0 = S^{-1}(c_0 \oplus k_5) \oplus k_1, \qquad u_1 = S^{-1}(c_1 \oplus k_6) \oplus k_2,$$
$$u_2 = S^{-1}(c_2 \oplus k_7) \oplus k_3, \qquad u_3 = S^{-1}(c_3 \oplus k_8) \oplus k_4 \,.$$

To simplify our explanations, we write the value of $m'_0 * k_1 \oplus m'_1 * u_1 \oplus m'_2 * u_2 \oplus m'_3 * u_3$ as θ. Thus, we have $u = m'_0 * S^{-1}(c_0 \oplus k_5) \oplus \theta$. Therefore, the exploited byte v can now be represented as

$$v = k_0 \oplus S^{-1}(m'_0 * S^{-1}(c_0 \oplus k_5) \oplus \theta) \,. \tag{1}$$

Observe that θ is related to only three ciphertext bytes, namely (c_1, c_2, c_3). Thus, if we have a number of ciphertexts such that bytes (c_1, c_2, c_3) are fixed to arbitrary values (and the other bytes are random), then the value of θ will be constant for these ciphertexts. In order to use these ciphertexts to conduct a first-order DPA attack exploiting v, we need to guess only the 24 unknown bits for (k_0, k_5, θ) to predict the value of v. This is much less than the limitation of 32 bits in the fundamental hypothesis. As a consequence, the DPA attack can reveal the values of the two key bytes k_0 and k_5, guessing only 24 unknown bits. The whole round key K_8 or K_{10} can be revealed by performing additional 15 similar attacks.

Further, we can conduct a single-bit (first-order) DPA attack using Eq. (1), by guessing only the 16 unknown bits for (k_5, θ). In such an attack, one bit of

the intermediate byte v is targeted, and to predict the target bit we first make a hypothesis on the value of the corresponding bit of k_0, and then guess for (k_5, θ). The difference between the average of the power traces for which the target bit is 1 and the average of the power traces for which the target bit is 0 is used to determine whether or not the guess for (k_5, θ) is correct. The difference will be the largest only when the guess for (k_5, θ) is correct, because the bit from k_0 does not affect the magnitude and just changes the sign of the difference. Therefore, the single-bit DPA attack can reveal k_5, guessing only 16 unknown bits (k_5, θ).

Attacking the SubBytes Operation in Round 3. The most difficult case in Principle (i) is to attack a byte immediately after the SubBytes operation in Round 3. Let v denote such a byte. Different from the intermediate byte exploited above, any intermediate byte between the MixColumns operation of Round 2 and the MixColumns operation of Round 8 depends on the whole 128 key bits and the whole 128-bit plaintext or ciphertext. Thus, we need more tricks to deal with this case. v is dependent on all the 16 plaintext bytes, due to the diffusion and confusion properties of the MixColumns and ShiftRows operations of Rounds 1 and 2. Following a reasoning similar to that described above, we learn that the intermediate byte v is an expression of the form:

$$v = S(m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus \delta)\,, \tag{2}$$

where m_0, m_1 are relevant elements from MC, p_0 is a plaintext byte, k_0 is the byte of K_0 XORed with p_0, θ is a function of 3 plaintext bytes and 4 key bytes, and δ is a function of 12 plaintext bytes and 13 key bytes. After a simple analysis, we can get the details of θ and δ.

If we choose a number of plaintexts with the fifteen bytes except p_0 fixed, then θ and δ are constant for these plaintexts. Thus, using these plaintexts, we can conduct a first-order DPA attack using Eq. (2), by guessing the 24 unknown bits for (k_0, θ, δ), and obtain the 8 key bits for k_0. Note that unlike in the previous attack, we now have to guess for all the 24 unknown bits even when a single-bit DPA is conducted, for each bit of v depends on all the 24 unknown bits.

4.2 Explaining Principles (iii) and (iv)

Second-order DPA exploits two intermediate values. We first explain a moderate case of Principle (iii), and then explain a most difficult case of Principle (iii). Similar for the remaining cases of Principles (iii) and (iv).

Attacking the SubBytes Operation in Round 2. The two exploited bytes are from the output of the SubBytes operation in Round 2. Let's first investigate how the targeted intermediate bytes are calculated. Let v be the output of one exploited S-box in Round 2, and let u be the input of the S-box, that is, $v = S(u)$. This input u equals the XOR of a byte of K_1, denoted by k_0, and an output byte of the MixColumns operation in Round 1. This output byte of the MixColumns

operation is calculated based on a column of 4 input bytes, and let (u_0, u_1, u_2, u_3) denote this column. Then, u can be expressed as:

$$u = k_0 \oplus m_0 * u_0 \oplus m_1 * u_1 \oplus m_2 * u_2 \oplus m_3 * u_3 \,,$$

where m_0, m_1, m_2, m_3 are relevant elements from MC. The four bytes u_0, u_1, u_2, u_3 are calculated based on 4 plaintext bytes and 4 bytes of K_0; we denote by (p_0, p_1, p_2, p_3) the 4 plaintext bytes and by (k_1, k_2, k_3, k_4) the 4 bytes of K_0. Hence, we have

$$u_0 = S(p_0 \oplus k_1), \quad u_1 = S(p_1 \oplus k_2), \quad u_2 = S(p_2 \oplus k_3), \quad u_3 = S(p_3 \oplus k_4).$$

Let $\theta = k_0 \oplus m_1 * u_1 \oplus m_2 * u_2 \oplus m_3 * u_3$, then v can be rewritten as $v = S(m_0 * S(p_0 \oplus k_1) \oplus \theta)$.

Now, given the other exploited output byte w of the SubBytes operation in Round 2, we have two situations to consider: (A) v and w are dependent on different sets of four plaintext bytes; and (B) v and w are dependent on the same set of four plaintext bytes. We first consider situation A. Compute the XOR of v and w as:

$$v \oplus w = S(m_0 * S(p_0 \oplus k_1) \oplus \theta) \oplus w \,. \tag{3}$$

It requires 4 plaintext bytes and 5 key bytes to calculate v or w. Hence, a straightforward second-order DPA attack needs to guess the 10 key bytes in order to predict the value $v \oplus w$, which is impossible in practice under the fundamental hypothesis. Nevertheless, observe that θ and w involve a total of 7 plaintext bytes. If we have a number of plaintexts with the seven bytes involved in θ and w fixed, then w and θ are constant for these plaintexts, and thus we can conduct a second-order DPA attack using Eq. (3), by guessing only the 24 unknown bits for (k_1, θ, w). Further, as in the single-bit first-order DPA attack in Section 4.1, if a single-bit second-order DPA attack is conducted using Eq. (3), we need to guess only the 16 unknown bits for (k_1, θ), and finally reveal the key byte k_1.

Next we consider situation B, where v and w are dependent on the same set of four plaintext bytes. Similarly, w can be expressed as $w = S(m_4 * S(p_0 \oplus k_1) \oplus \theta')$, where m_4 is an element from MC, and θ' is a function of the three plaintext bytes (p_1, p_2, p_3) and four key bytes. Thus, the XOR of v and w is

$$v \oplus w = S(m_0 * S(p_0 \oplus k_1) \oplus \theta) \oplus S(m_4 * S(p_0 \oplus k_1) \oplus \theta') \,. \tag{4}$$

As a result, if we have a number of plaintexts with the three bytes involved in θ and θ' fixed, then θ and θ' are constant for these plaintexts, and thus we can conduct a second-order DPA attack using Eq. (4) to obtain k_1, by guessing only the 24 unknown bits for (k_1, θ, θ').

Attacking the AddRoundKey Operation in Round 2. The two exploited bytes v and w are from the output of the AddRoundKey operation of Round 2. Obviously, either of v and w is dependent on all the 16 plaintext bytes and 21

key bytes. After a simple analysis, we learn that v and w can be respectively expressed as the following form.

$$v = m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus \delta\,,$$
$$w = m_2 * S(m_3 * S(p_0 \oplus k_0) \oplus \theta') \oplus \delta'\,,$$

where m_0, m_1, m_2, m_3 are elements from MC, p_0 is a plaintext byte, k_0 denotes the byte of K_0 XORed with p_0, either of θ and θ' is a function of 3 plaintext bytes and 4 key bytes, and either of δ and δ' is a function of 12 plaintext bytes and 16 key bytes. Consequently, the XOR of v and w is

$$v \oplus w = m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus m_2 * S(m_3 * S(p_0 \oplus k_0) \oplus \theta') \oplus \delta \oplus \delta'\,. \quad (5)$$

Observe that θ' involves the same set of 3 plaintext bytes as θ, and δ' involves the same set of 12 plaintext bytes as δ. Thus, if we have a number of plaintexts with the 15 plaintext bytes except p_0 fixed, then θ, θ', δ and δ' will be constant for these plaintexts. Thus, by guessing the 24 unknown bits for (k_0, θ, θ') we can conduct a single-bit second-order DPA attack using Eq. (5) to obtain k_0.

Furthermore, $v \oplus w$ involves only a total of 12 plaintext bytes in a few special situations, and thus we need a number of plaintexts with only 11 bytes fixed. For example, v and w are the output bytes $(0, i)$ and $(1, i)$ of the AddRoundKey operation of Round 2, where $0 \leq i \leq 3$. As bytes (0,3) and (1,3) of the MC matrix are identical (i.e. $0x01$), XORing v and w will cancel the last input byte of the i-th column of the MixColumns operation of Round 2, and $v \oplus u$ involves a total of 12 plaintext bytes only. As a result, we only need to fix 11 plaintext bytes to get constant θ, θ', δ and δ'. Other special situations include bytes $(0, i)$ and $(3, i)$, bytes $(1, i)$ and $(2, i)$, and bytes $(2, i)$ and $(3, i)$, which can be easily spotted from the MC matrix. These special situations are because every column of the MC matrix contains two identical elements. Similar special cases do not hold for the MC^{-1} matrix, for the four elements of a column of MC^{-1} are different one another.

4.3 Explaining Principle (v)

We briefly explain a difficult case in Principle (v): one exploited byte v is immediately after the SubBytes operation of Round 2, and the other exploited byte w is immediately before the AddRoundKey operation of Round 8. We know that v is dependent on 4 plaintext bytes and 5 round key bytes, and w is dependent on 4 ciphertext bytes and 9 round key bytes. A similar analysis reveals that $v \oplus w$ can be expressed as the following form:

$$v \oplus w = S(m_0 * S(p_0 \oplus k_0) \oplus \theta) \oplus w\,, \quad (6)$$

where m_0 is an element from MC, p_0 is a plaintext byte, k_0 is the byte of K_0 XORed with p_0, and θ is a function of 3 plaintext bytes and 4 key bytes. If

we have a number of plaintexts and ciphertexts such that the 3 plaintext bytes involved in θ and the 4 ciphertext bytes involved in w are fixed, then θ and w are constant for these plaintexts and ciphertexts. Consequently, we can conduct a second-order DPA attack using Eq. (6) to obtain k_0, by guessing the 24 unknown bits for (k_0, θ, w). Similarly, a single-bit second-order DPA attack guessing only the 16 unknown bits for (k_0, θ) is feasible using Eq. (6).

Alternatively, $v \oplus w$ can also be expressed as the following form:

$$v \oplus w = S^{-1}(m'_0 * S^{-1}(c_0 \oplus k_0) \oplus \theta) \oplus \delta\,, \tag{7}$$

where m'_0 is an element from MC^{-1}, c_0 is a ciphertext byte, k_0 denotes the byte of K_{10} XORed with c_0, θ is a function of 3 ciphertext bytes and 7 key bytes, and δ is a function of 4 plaintext bytes and 6 key bytes. Therefore, if we have a number of plaintexts and ciphertexts such that the 4 plaintext bytes involved in δ and the 3 ciphertext bytes involved in θ are fixed, we can conduct a second-order DPA attack using Eq. (7) to obtain k_0 by guessing the 24 unknown bits for (k_0, θ, δ), and conduct a single-bit second-order DPA attack to obtain k_0 by guessing only the 16 unknown bits for (k_0, θ).

5 Experimental Results

In 2006, Herbst, Oswald and Mangard [8] presented an AES software implementation for 8-bit smartcards; see [8] for its specifications. Herbst et al.'s implementation uses randomization and masking for (roughly) the first round and the last one and a half rounds, and uses only masking for the inner rounds. Building on Herbst et al.'s idea, in 2007 Tillich, Herbst and Mangard [9] presented an AES implementation for 32-bit smartcards; see [9] for its specifications. Tillich et al.'s implementation uses both the countermeasures for (roughly) the first one and a half rounds and the last one and a half rounds, and uses no countermeasures for the inner rounds.

In this section, we take Herbst et al.'s and Tillich et al.'s AES implementations as examples, and conduct representative experiments to justify the principles in Section 3. We program the two implementation schemes in the EEPROM of an AVR-based 8-bit micro-controller. The micro-controller is clocked at 3.57 Mhz, and the power signals are sampled at a rate of 200M samples per second. Note that we use correlation to distinguish correct and incorrect key guesses in the experiments, but similar results hold when using the classical "difference of means" way [1].

5.1 Practical Attacks on Herbst et al.'s Implementation

From [8] we observe that the output bytes of the SubBytes operation of Round 2 or 9 are protected only by the same mask M' in Herbst et al.'s implementation, and the output bytes of the AddRoundKey operation of Round 1 or 8 are protected only by the same mask M. Thus, by Principle (vii) we learn that Herbst et al.'s implementation is vulnerable to second-order DPA attacks.

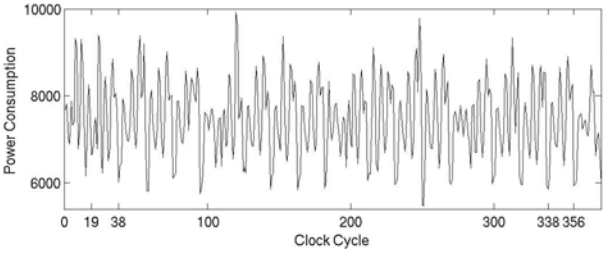

Fig. 1. The SubBytes operation in Round 2

We perform a second-order attack that exploit two output bytes of the SubBytes operation of Round 2. The attack recovers a key byte by guessing only 16 unknown bits. The two exploited bytes are the first byte $X_{0,0}$ and the last byte $X_{3,3}$. Following the descriptions in Section 4.2, we know that their XOR is,

$$X_{0,0} \oplus X_{3,3} = S(m_{0,0} * S(P_{0,0} \oplus K^0_{0,0}) \oplus \theta) \oplus X_{3,3}\,.$$

We choose 16384 plaintexts such that the seven bytes $(1,1)$, $(2,2)$, $(3,3)$, $(1,0)$, $(2,1)$, $(3,2)$ and $(0,3)$ are respectively fixed to 5, 10, 15, 1, 6, 11 and 12, and the other bytes are random, and we use the secret key $K = (K_{i,j})$ with $K_{i,j} = i + j \times 4$. Thus, $\theta = 115$ and $X_{3,3} = 94$ hold for every plaintext.

We then obtain the power traces for encrypting the 16384 plaintexts on the micro-controller. We use the attack strategy given in Chapter 10.3 of [21]. In order to reduce the computational workload of the attack, we compress the power traces by combining (adding up) all the signals during each clock cycle. We next make an educated guess for the time frames when $X_{0,0}$ and $X_{3,3}$ are computed during every execution by optically inspecting the power traces. Fig. 1 plots a power trace segment that corresponds to the SubBytes operation in Round 2, where the 16 sequential S-box substitutions are clearly distinguishable from each other. The first S-box substitution occurs in clock cycles 19–38, and the last S-box substitution occurs in clock cycles 338–356. Subsequently, for every power trace we calculate the absolute-of-difference of each signal in clock cycles 19–38 and each signal in clock cycles 338–356. The resulting traces, which contains 380 signals each, are referred to as the preprocessed power traces. At last, a first-order attack is applied using the preprocessed power traces. In this step, we correlate the power traces to the leftmost bit of the exploited variole $X_{0,0} \oplus X_{3,3}$, where all the possible values for $(K^0_{0,0}, \theta)$ are tested. In our experiment the advantages of a single-bit attack over a multi-bit attack are two-fold: (1) The testing space is reduced by $\frac{1}{3}$; and (2) The ghost peaks caused by incorrect hypotheses of $X_{3,3}$ in case of a multi-bit attack (due to the linearity of XOR) can be avoided.

The results of the second-order attack are plotted in Fig. 2, where the black curve corresponds to the correct guess (i.e., when $(K^0_{0,0}, \theta) = (0, 115)$), and the gray curves correspond to the incorrect guesses. The figure shows that the correct guess leads to the highest peak at point 300. This point corresponds to clock cycles 34 and 352 in the original traces, during which $X_{0,0}$ and $X_{3,3}$ are manipulated on the micro-controller.

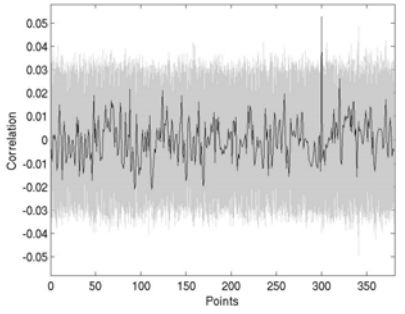

Fig. 2. Results for all the 65536 guesses in a second-order attack

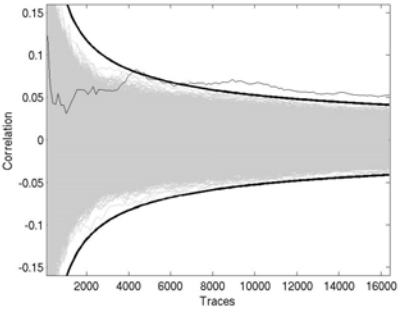

Fig. 3. Results at point 300 with an increasing number of traces

We analyze the number of power traces required by such an attack. We perform 128 attacks using 128, 256, 384, ..., 16384 power traces, respectively. Different from the above experiment, we only test the interesting point in time, i.e., point 300 in the preprocessed trace. Fig. 3 depicts the evolution of the results over an increasing number of power traces used, where the outer thicker black curves mark the expected region of the incorrect key hypotheses with a confidence of 0.9999. Again, the result for the correct guess is plotted in black, and the rest are plotted in gray. The point where the curve for the correct key hypothesis leaves this region gives an estimation of the number of traces required by a successful attack. From Fig. 3 we get that the attack can almost always succeed with approximately 8000 traces.

5.2 Practical Attacks on Tillich et al.'s Implementation

From [9] we observe that there is no protection between the AddRoundKey operation in Round 2 and the MixColumns operation in Round 3 and between the MixColumns operation in Round 7 and the AddRoundKey operation in Round 8. Thus, by Principle (vi) we can conduct first-order DPA attacks on Tillich et al.'s implementation. Observe that all the output bytes of the AddRoundKey operation in Round 8 are concealed by the same mask M, and thus by Principle (vii), Tillich et al.'s implementation is vulnerable to second-order DPA attacks. The second-order DPA attacks are similar to that described above for Herbst et al.'s implementation.

We perform a first-order attack exploiting an input byte of the AddRoundKey operation in Round 8. The attack reveals a key byte of K_{10} by guessing only 16 unknown bits. The exploited byte is the first input $X_{0,0}$, and by the guideline in Section 4.1 we have,

$$X_{0,0} = K^8_{0,0} \oplus S^{-1}(m'_{0,0} * S^{-1}(C_{0,0} \oplus K^{10}_{0,0}) \oplus \theta).$$

We generate 16384 plaintexts which cause 16384 ciphertexts with bytes $(1, 3)$, $(2, 2)$ and $(3, 1)$ being equal respectively to 13, 10 and 7 when encrypted using

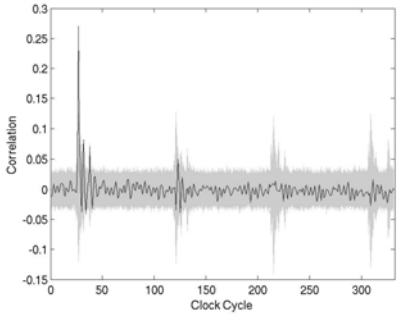

Fig. 4. Results for all 65536 guesses in a first-order attack

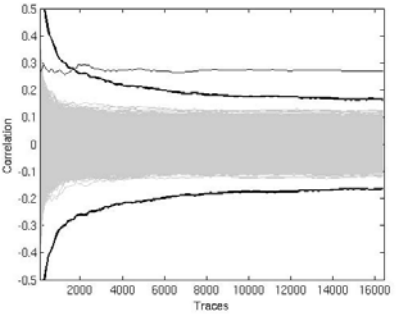

Fig. 5. Results at clock cycle 27 with an increasing number of traces

the secret key $K = (K_{i,j})$ with $K_{i,j} = i + j \times 4$. Hence, $\theta = 124$ for all the ciphertexts, and $K_{0,0}^{10} = 19$.

We collect the power traces for encrypting the plaintexts on the micro-controller. Again, to reduce the computational complexity of the subsequent analysis we perform the same compression of traces and optical inspection as in the attack in Section 5.1. Finally, we successfully find the key byte $K_{0,0}^{10}$ after performing an attack on the leftmost bit of $X_{0,0}$ by guessing for $(K_{0,0}^{10}, \theta)$. The results of this attack are depicted in Fig. 4, where the black curve represents the result for the correct guess of $(K_{0,0}^{10}, \theta)$ and the gray curves represent the results for the incorrect guesses. A distinctive peak happens at clock cycle 27, indicating that $X_{0,0}$ is manipulated at clock cycle 27 during the encryptions. To analyze the number of required power traces, we performe 128 such attacks using 128, 256, 384, ..., 16384 power traces respectively and calculate the results only for clock cycle 27; the results are shown in Fig. 5, which indicates 2000 traces are adequate for a successful attack.

6 Comparison with Related Work

As mentioned above, this paper concerns solely the security of an AES implementation against the primary type of power analysis — DPA. During the past few years, other types of power analysis have been proposed, including the template attack [28], the side-channel collision analysis [29] and those obtained by combining the existing power analytic techniques [30]. Compared with DPA, these attack techniques require more assumptions on the ability of an attacker and are much harder to conduct in practice.

Following the side-channel collision analysis approach, in 2006 Handschuh and Preneel [31] applied differential cryptanalysis [32] for power attacks on DES, which exploit intermediate values immediately after the first four rounds of DES. Handschuh and Preneel's result differs from ours in several aspects: (I) Their attacks require high probability differentials for reduced rounds of the cipher concerned, and are hard to apply to the first two and a half rounds of AES (i.e. from the beginning until the MixColumns operation of the third round). For DES there are 4-round differentials with large probabilities (e.g. 3.8×10^{-4}); but as the

maximum differential probability for the AES S-box is 2^{-6} and the MixColumns operation has a branch number of 5 [33], thus for AES a differential characteristic operating on the first two and a half rounds has a probability of at most 2^{-54}. Park et al. [34] gave the upper bound 1.144×2^{-111} for the probability of a 4-round AES differential. Therefore, these differentials are not useful for power analysis in practice because their probabilities are too small; (II) Handschuh and Preneel's attacks depend on a direct measurement of the Hamming weight of an intermediate value, and like the side-channel collision analysis, are harder to perform than DPA attacks, but our attacks are DPA attacks; and (III) Our result also includes attacks that work for protected implementations where the exploited intermediate state is masked or partially masked.

Motivated by Handschuh and Preneel's work, in 2007 Biryukov and Khovratovich [35] applied two other traditional symmetric-key cryptanalytic methods to obtain new techniques of side-channel cryptanalysis, namely the impossible collision attack and the multiset collision attack, which exploit intermediate values immediately after the first three or four rounds of AES. This is better than our result in terms of the numbers of attacked rounds. However, their attacks require more and stronger assumptions on the ability of an attacker than ours, and have much larger data and computation complexities, e.g., $2^{19} - 2^{32}$ measurements and $2^{27} - 2^{54}$ off-line computations; and another important difference is like that described in the above (III): our result can be applicable in some implementations with the exploited intermediate values being masked.

In 2008, using some sophisticated techniques [9] obtained by combining (high-order) DPA attacks and the windowing technique [3], Tillich and Herbst [36] presented several complicated power analysis attacks on Herbst et al.'s implementation. Their attacks aim to demonstrate how to break the countermeasures; they use the assumption that the attacker has the knowledge of the generating or storing time of a mask (as well as its power consumption), and use the windowing technique to deal with the effect of randomization. By contrast, we aim to demonstrate how far intermediate values can be exploited. Our attacks do not use the assumption, and do not need to deal with the effect of randomization, for they exploit intermediate values beyond the randomization zones; and they require much less plaintexts, and are very simple and easy to conduct in practice.

7 Principles on the Protection of AES against First and Second-Order DPA

From the principles in Section 3, we can easily get the following principles for protecting AES against first and second-order DPA.

(1) Any one or two bytes concerned by Principles (i)–(v) should not be left unprotected.
(2) Any two bytes concerned by Principles (iii)–(v) should not be protected only by the same mask; in other words, if they are protected by the same mask, other countermeasure(s) should be present.

How far should we protect the beginning and end parts of AES in order to resist first and secon-order DPA attacks (under the fundamental hypothesis)? Now the general principles answer this question to some extent: In summary, in order to make an AES implementation secure against first and second-order DPA, one should sufficiently protect at least the first two and a half rounds and the last three rounds of AES, i.e. from the beginning until the MixColumns operation of Round 3 and from the beginning of Round 8 to the end. Note that this result is towards a general application environment, and aims to provide a necessary security level against first and second-order DPA, and sophisticated attacks are not taken into account. In addition, it is based on the techniques we have known currently. In practice, implementation designers can decide the protected rounds according to application environments, for example, if it is not possible for an attacker to obtain plaintexts then the first two and a half rounds might be left unprotected.

Various countermeasures or their combinations can be applied to make an AES implementation secure against first and second-order DPA. If using the randomization and masking techniques as followed in [8, 9], one should apply masking to those operations that are vulnerable to first-order DPA, and should apply both countermeasures to those operations that are vulnerable to second-order DPA. If the sophisticated attacks were considered, enhanced protection mechanisms should be adopted.

8 Conclusions

In this paper, we have extensively studied the security of AES against first and second-order DPA. A few general principles have been presented for attacking an AES implementation. We have discovered that some values from the inner rounds of AES can be exploited to conduct a first or second-order DPA attack. As examples, we have demonstrated that Herbst et al.'s and Tillich et al.'s AES implementations are even vulnerable to first and second-order DPA attacks, although they are designed to achieve a high protection against power analysis using two kinds of countermeasures. We have given several principles for protecting an AES implementation against first and second-order DPA attacks. In general, for the time being, the first two and a half rounds and the last three rounds of an AES implementation should be well protected in order to thwart first and second-order DPA in practice.

Acknowledgments. The authors are very grateful to Jaap de Vos for providing the power traces and to Jasper van Woudenberg and the anonymous referees for their comments.

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
2. Goubin, L., Patarin, J.: DES and differential power analysis — the "duplication" method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)

3. Clavier, C., Coron, J.S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
4. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
5. Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical second-order DPA attacks for masked smart card implementations of block ciphers. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)
6. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), FIPS-197 (2001)
7. International Organization for Standardization (ISO), ISO/IEC 18033-3:2005: Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers (2005)
8. Herbst, C., Oswald, E., Mangard, S.: An AES smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
9. Tillich, S., Herbst, C., Mangard, S.: Protecting AES software implementations on 32-bit processors against power analysis. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 141–157. Springer, Heidelberg (2007)
10. Akkar, M.-L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
11. Blömer, J., Guajardo, J., Krummel, V.: Provably secure masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
12. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
13. Oswald, E., Schramm, K.: An efficient masking scheme for AES software implementations. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 292–305. Springer, Heidelberg (2006)
14. Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)
15. Akkar, M.-L., Goubin, L.: A generic protection against high-order differential power analysis. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 192–205. Springer, Heidelberg (2003)
16. Akkar, M.L., Bévan, R., Goubin, L.: Two power analysis attacks against one-mask method. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 332–347. Springer, Heidelberg (2004)
17. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
18. Daemen, J., Rijmen, V.: Resistance against implementation attacks: a comparative study of the AES proposals. In: Proceedings of The Second Advanced Encryption Standard Candidate Conference, NIST(1999)
19. Jaffe, J.: Introduction to differential power analysis. In: ECRYPT Summer School on Cryptographic Hardware, Side Channel and Fault Analysis (2006)
20. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)

21. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: revealing the secrets of smart cards. Springer, Heidelberg (2007)
22. Pan, J., den Hartog, J., de Vink, E.: An operation-based metric on CPA resistance. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC 2008. International Federation for Information Processing, vol. 278, pp. 429–443. Springer, Boston (2008)
23. Prouff, E., Ciraud, C., Aumonier, S.: Provably secure S-box implementation based on Fourier transform. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 216–230. Springer, Heidelberg (2006)
24. National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), FIPS-46 (1977)
25. Lv, J., Han, Y.: Enhanced DES implementation secure against high-order differential power analysis in smartcards. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 195–206. Springer, Heidelberg (2005)
26. Lv, J.: On two DES implementations secure against differential power analysis in smart-cards. Information and Computation 204(7), 1179–1193 (2006)
27. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: Proceedings of The Third Advanced Encryption Standard Candidate Conference, NIST (2000)
28. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
29. 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)
30. Pan, J., den Hartog, J., Lu, J.: You cannot hide behind the mask: Power analysis on a provablely secure S-box implementation. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 178–192. Springer, Heidelberg (2009)
31. Handschuh, H., Preneel, B.: Blind differential cryptanalysis for enhanced power attacks. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 163–173. Springer, Heidelberg (2007)
32. Biham, E., Shamir, A.: Differential cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
33. Daemen, J., Rijmen, V.: AES proposal: Rijndael. In: Proceedings of The First Advanced Encryption Standard Candidate Conference, NIST (1998)
34. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)
35. Biryukov, A., Khovratovich, D.: Two new techniques of side-channel cryptanalysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 195–208. Springer, Heidelberg (2007)
36. Tillich, S., Herbst, C.: Attacking state-of-the-art software countermeasures—A case study for AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 228–243. Springer, Heidelberg (2008)