A First-Order DPA Attack Against AES in Counter Mode with Unknown Initial Counter

Josh Jaffe

Cryptography Research, Inc.
575 Market Street, suite 2150, San Francisco, CA 94105, USA.
josh@cryptography.com

Abstract. Previous first-order differential power analysis (DPA) attacks have depended on knowledge of the target algorithm's input or output. This paper describes a first-order DPA attack against AES in counter mode, in which the initial counter and output values are all unknown.

Keywords: power analysis, SPA, DPA, HO-DPA, AES, counter mode.

1 Introduction

Previous first-order differential power analysis (DPA) attacks have depended on knowledge of the target algorithm's input or output [1][2]. This paper describes a first-order DPA attack against the Advanced Encryption Standard (AES) [3] in counter mode, in which the initial counter, input values, and output values are all unknown.

The attack proceeds as follows. Suppose the input data to an algorithm is unknown, but can be expressed as single secret constant summed with known, variable data. The known, variable part of the data is used to mount a DPA attack, and the secret constant is treated as part of the key to be recovered. The "key" recovered by the DPA attack is then a function of the actual key and the secret constant. The known input values are then combined with the recovered "key" to compute the actual intermediate values produced by the algorithm. The recovered intermediates are then used to carry the attack forward into later rounds, enabling additional DPA attacks to recover the real key.

The attack also addresses the challenges to DPA presented by block ciphers used in counter mode [4]. DPA attacks target secrets when they are mixed with known *variable* quantities. In counter mode only the low-order bits of the input change with each encryption. Hence there are few variable intermediates to target in the first round of a typical block cipher. We demonstrate a method for propagating the attack into later rounds in which more known, variable data is available.

Although counter mode presents additional challenges to DPA attacks, in certain respects it also makes the attack easier. Unlike most first-order DPA attacks, the sequential nature of the counter enables the attack to succeed with only knowledge of the power measurements. Knowledge of input, output, and initial counter values are not required to implement the attack.

1.1 Related Work

Simple power analysis (SPA) attacks have been used to extract portions of keys directly from power traces without requiring knowledge of input messages. Fahn and Pearson used inferential power analysis (IPA), an attack that exploits binary SPA leaks [5]. Mayer-Sommer presented attacks exploiting SPA leaks in high-amplitude power variations [6]. Mangard presented an SPA attack against the AES key expansion step [7]. Messerges et al described SPA attacks on Hamming weight and transition count leaks [8].

Side channel collision attacks were introduced by Dobbertin, and have traditionally targeted SPA leaks using chosen ciphertext [9] [10] [11]. Side channel collision attacks can be adapted to the case in which inputs are known to be successive values of a counter.

High-order differential power analysis (HO-DPA) [12] attacks target a hypothesized key-dependent relationship between data parameters in a computation. Previous work has noted that HO-DPA attacks can be applied to situations in which cipher input values are not known [13].

Fouque and Valette presented the "doubling attack" [14] which exploits the relationship between inputs in successive RSA decryptions to recover the exponent. The attack succeeds despite the fact that the input to the modular exponentiation step is masked by a blinding factor. Messerges presented a second-order DPA attack [15] that defeated a data whitening scheme.

Chari et al [16] and Akkar et al [17] also presented DPA attacks on block ciphers with a "whitening" step.

2 Preliminaries

2.1 Notation

Suppose X and Y are used to denote input and output data of a transformation. (Letters other than X or Y will also be used.) If the transformation is implemented as a sequence of rounds, the input and output of the i^{th} round are denoted by X_i and Y_i.

Within a round, data may be partitioned into bytes for processing. $X_{i,j}$ and $Y_{i,j}$ denote the j^{th} bytes of round data X_i and V_i.

K is used to denote input keys, K_i denotes the i^{th} round key derived from K, and $K_{i,j}$ denotes the j^{th} byte of round key K_i.

Symbols

The symbol '$\oplus$' denotes the bitwise XOR of two n-bit vectors.
The symbol '$+$' denotes the ordinary addition of two numbers.
The symbol '$\circ$' denotes multiplication between two elements of $GF(2^8)$.
The symbol '$||$' denotes the concatenation of two vectors.

2.2 Description of AES

Although most readers are no doubt familiar with AES, this section gives a brief review of its design. The round transformations are grouped differently than in the AES standard to facilitate presentation of the attack, but the algorithm described here is equivalent to AES. The review will also familiarize the reader with the notation and concepts used in this paper.

AES is a block cipher that operates on 16-byte blocks of data. It is designed as a sequence of 10, 12, or 14 rounds, depending on whether the key K is 16, 24, or 32 bytes in length. The key is expanded by the AES key schedule into 16-byte round keys K_i.

The round structure of AES encryption. The following transformations are performed during each round of an AES encryption:

1. AddRoundKey
2. SubBytes
3. ShiftRows
4. MixColumns[1]

These operations are described below, using the following notation for intermediate round states:

X_i denotes the input to round i and the AddRoundKey transformation.
Y_i denotes the output of the AddRoundKey transformation and the input to the SubBytes transformation.
Z_i denotes the output of the SubBytes transformation and the input to the ShiftRows transformation.
U_i denotes the output of the ShiftRows transformation and the input to the MixColumns transformation.
V_i denotes the output of the MixColumns transformation and the input to the next round: $V_i = X_{i+1}$.

AddRoundKey Each byte of $Y_{i,j}$ is produced by computing the exclusive or (XOR) of a byte of incoming data $X_{i,j}$ with the corresponding byte of round key $K_{i,j}$:

$$Y_{i,j} = X_{i,j} \oplus K_{i,j} \ . \tag{1}$$

SubBytes Each byte of input data is transformed via an invertible non-linear 8-bit lookup table S:

$$Z_{i,j} = S[Y_{i,j}] = S[X_{i,j} \oplus K_{i,j}] \ . \tag{2}$$

[1] The MixColumns operation is not performed in the final round, and an additional AddRoundKey operation is performed after the final round.

ShiftRows ShiftRows permutes the bytes within the data vector:

$$U_i = \boxed{Z_{i,0}}\boxed{Z_{i,5}}\boxed{Z_{i,10}}\boxed{Z_{i,15}}\boxed{Z_{i,4}}\boxed{Z_{i,9}}\boxed{Z_{i,14}}\boxed{Z_{i,3}}\boxed{Z_{i,8}}\boxed{Z_{i,13}}\boxed{Z_{i,2}}\boxed{Z_{i,7}}\boxed{Z_{i,12}}\boxed{Z_{i,1}}\boxed{Z_{i,6}}\boxed{Z_{i,11}}$$

MixColumns The j^{th} column of U_i is defined to be the four bytes

$$\{U_{i,4j}, U_{i,4j+1}, U_{i,4j+2}, U_{i,4j+3}\} \ .$$

MixColumns is an invertible linear transformation over $GF(2^8)$ performed on the columns of U_i. The j^{th} column of output V_i is defined to be:

$$\begin{aligned}
V_{i,4j} &= (\{02\} \circ U_{i,4j}) \oplus (\{03\} \circ U_{i,4j+1}) \oplus (\{01\} \circ U_{i,4j+2}) \oplus (\{01\} \circ U_{i,4j+3}) \\
V_{i,4j+1} &= (\{01\} \circ U_{i,4j}) \oplus (\{02\} \circ U_{i,4j+1}) \oplus (\{03\} \circ U_{i,4j+2}) \oplus (\{01\} \circ U_{i,4j+3}) \\
V_{i,4j+2} &= (\{01\} \circ U_{i,4j}) \oplus (\{01\} \circ U_{i,4j+1}) \oplus (\{02\} \circ U_{i,4j+2}) \oplus (\{03\} \circ U_{i,4j+3}) \\
V_{i,4j+3} &= (\{03\} \circ U_{i,4j}) \oplus (\{01\} \circ U_{i,4j+1}) \oplus (\{01\} \circ U_{i,4j+2}) \oplus (\{02\} \circ U_{i,4j+3})
\end{aligned}$$

where $\{01\}$, $\{02\}$, $\{03\}$, and $U_{i,4j}$, $U_{i,4j+1}$, $U_{i,4j+2}$, $U_{i,4j+3}$ are considered 8-bit vectors representing elements in $GF(2^8)$.

The linearity of the AES MixColumns transformation will be exploited during the attack. Suppose that input data can be selected such that in round i, one or more input bytes to the MixColumns operation are unknown, but are known to remain constant across multiple invocations of the AES algorithm. Then the contribution of these constant bytes to V_i is equivalent to XORing with fixed constants.

For example, suppose bytes $U_{1,4j+1}$, $U_{1,4j+2}$, and $U_{1,4j+3}$ are constant (but unknown) across multiple invocations of AES. Then the values

$$\begin{aligned}
E_{1,4j} &= (\{03\} \circ U_{1,4j+1}) \oplus (\{01\} \circ U_{1,4j+2}) \oplus (\{01\} \circ U_{1,4j+3}) \\
E_{1,4j+1} &= (\{02\} \circ U_{1,4j+1}) \oplus (\{03\} \circ U_{1,4j+2}) \oplus (\{01\} \circ U_{1,4j+3}) \\
E_{1,4j+2} &= (\{01\} \circ U_{1,4j+1}) \oplus (\{02\} \circ U_{1,4j+2}) \oplus (\{03\} \circ U_{1,4j+3}) \\
E_{1,4j+3} &= (\{01\} \circ U_{1,4j+1}) \oplus (\{01\} \circ U_{1,4j+2}) \oplus (\{02\} \circ U_{1,4j+3})
\end{aligned}$$

will be constant, and the MixColumns output can be expressed as

$$\begin{aligned}
V_{1,4j} &= (\{02\} \circ U_{1,4j}) \oplus E_{1,4j} \\
V_{1,4j+1} &= (\{01\} \circ U_{1,4j}) \oplus E_{1,4j+1} \\
V_{1,4j+2} &= (\{01\} \circ U_{1,4j}) \oplus E_{1,4j+2} \\
V_{1,4j+3} &= (\{03\} \circ U_{1,4j}) \oplus E_{1,4j+3} \ . \qquad (3)
\end{aligned}$$

As will be shown in Section 3, the constant, unknown terms E can then be incorporated into the round key of the next round, and effectively ignored.

2.3 Counter Mode

Counter mode is a standard mode of operation for block ciphers in which ciphertext is produced by encrypting a counter and XORing the result with the plaintext block. Let B be a block cipher using key K, C the initial counter value, and X_T the T^{th} block of plaintext to be encrypted. Then the T^{th} block of ciphertext Y_T is given by

$$Y_T = X_T \oplus B_{enc}(C + T, K) \ .$$

Ciphertext is decrypted by XORing it with same encrypted counter value:

$$X_T = Y_T \oplus B_{enc}(C + T, K) \ .$$

Since counter values are inputs to the first round only, C_j and T_j will be used to denote the j^{th} bytes of C and T respectively, and not their values at round j. See [4] for more information on counter mode.

Galois counter mode Galois counter mode (GCM) [18] is a draft counter mode protocol currently being studied by NIST. In GCM, the initial counter value is derived from a variable-sized initialization vector (IV). If the length of the IV is not exactly 96 bits, then the initial counter value C is derived from the IV using a secret key. In protocols where the IV is exactly 96 bits long, at least part of the initial counter value may be secret. For example, in RFC 4106 [19] the first four bytes of the IV are derived with the AES key and may remain secret. The attack described in this paper assumes that the entire initial counter value C is unknown.

3 The Attack on AES in Counter Mode

This section will present a first-order DPA attack against AES in counter mode with unknown initial counter value C.

To keep the index notation from getting too cumbersome, the symbol "T" is omitted from subscripts. When data is described as constant or variable, however, it means that the data is constant or variable with respect to T. For example, when we say that an attack recovers a variable such as $Z_{1,15}$, it means that it recovers each value the variable took for each value of T.

3.1 Overview

The main stages of the attack are as follows:

1. Perform data collection.
2. Use DPA against the first round to recover $Z_{1,15}$ and $Z_{1,14}$.

3. Derive the input to the second round, manipulating unknown values symbolically. Eight bytes of input to the second round are unknown constants, but the other eight can be expressed as

$$X_{2,j} = \tilde{X}_{2,j} \oplus E_{1,j}$$

where $\tilde{X}_{2,j}$ is known and variable, and $E_{1,j}$ is unknown, but constant.
4. Use DPA to determine the eight variable bytes of $Z_{2,j}$ corresponding to the 8 variable bytes $X_{2,j}$.
5. Derive the input to the third round, manipulating unknown constants symbolically. Each of the sixteen bytes of X_3 can be expressed as

$$X_{3,j} = \tilde{X}_{3,j} \oplus E_{2,j} \ ,$$

where $\tilde{X}_{3,j}$ is known and variable, and $E_{2,j}$ is unknown, but constant.
6. Use DPA to determine the sixteen variable bytes of Z_3.
7. Derive the input to the fourth round. There are no unknown or constant bytes in Z_3, so X_4 can be derived exactly.
8. Perform a standard DPA attack in the fourth round, using known input values X_4. Iterate the attack into subsequent rounds as necessary, recovering as many round keys as required to reverse the key schedule and obtain the key.

These attack stages are described in detail below.

3.2 Attack Details

Step 1: Data Collection. Encrypt 2^{16} consecutive blocks of data in counter mode, with unknown initial counter, and initial data blocks given by $X_1 = C + T$.

Record power traces covering the first four rounds of each encryption. Traces should cover the fifth round as well if the target key is longer than 16 bytes.

Step 2: Recover $Z_{1,15}$. The DPA attack in this step uses the known byte T_{15} as the input, and performs a 15-bit exhaustive search over the bits defined below.

Let $C_{15,lo}$ and $K_{1,15,lo}$ denote the low-order 7 bits of C_{15} and $K_{1,15}$, respectively. Let $C_{15,hi}$ denote the high-order bit of C_{15}, and b_{15} denote the XOR of $C_{15,hi}$ with the high-order bit of $K_{1,15}$. Let ϵ_{15} denote the outgoing carry of $C_{15} + T_{15}$. The reader can verify that

$$(C_{15} + T_{15}) \textbf{ mod } 256 = (2^7 * C_{15,hi}) \oplus ((C_{15,lo} + T_{15}) \textbf{ mod } 256) \ . \qquad (4)$$

Then Equations 1 and 4 imply that

$$\begin{aligned} Y_{1,15} &= K_{1,15} \oplus ((C_{15} + T_{15}) \ \textbf{mod } 256) \\ &= K_{1,15} \oplus (2^7 * C_{15,hi}) \oplus ((C_{15,lo} + T_{15}) \ \textbf{mod } 256) \\ &= (2^7 * b_{15}) \oplus K_{1,15,lo} \oplus ((C_{15,lo} + T_{15}) \ \textbf{mod } 256) \ . \end{aligned} \qquad (5)$$

Hence, the DPA search only depends on the 15 unknown bits in Equation 5: the bit b_{15}, seven bits of $K_{1,15,lo}$, and seven bits of $C_{15,lo}$. Also, note that the high-order bits of $K_{1,15}$ and C_{15} cannot be distinguished by this search.

The relationship between the parameters is shown in Figure 1. Note that because $Y_{1,15}$ is an eight bit quantity, it does not depend on the carry bit ϵ_{15}.

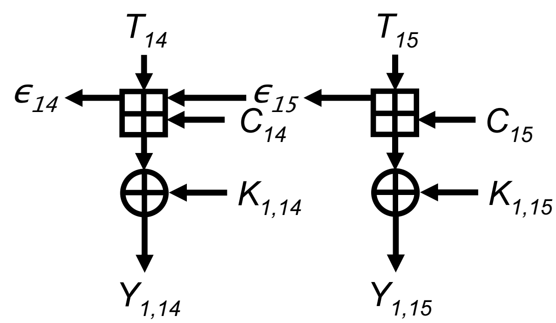

Fig. 1. Relationship between T, C, K, and Y for bytes 14 and 15 in Round 1 of the attack.

Step 3: Recover $Z_{1,14}$. The DPA attack in this step uses the known byte T_{14} as the input, and performs a 16-bit exhaustive searching over the following bits: the bit $C_{15,hi}$, the low-order 7 bits of C_{14}, the low-order 7 bits of $K_{1,14}$, and the bit b_{14}, defined as the XOR of the high-order bit of C_{14} with the high-order bit of $K_{1,14}$.

$Y_{1,14}$ is given by

$$Y_{1,14} = K_{1,14} \oplus ((C_{14} + T_{14} + \epsilon_{15}) \ \mathbf{mod}\ 256) \ .$$

$Y_{1,14}$ depends on ϵ_{15}, which iteself depends upon C_{15}. Hence $C_{15,hi}$ (the only bit of C_{15} not recovered in Step 2) is one of the bits searched for in this step. As in Step 2, the search recovers b_{14} but is unable to distinguish the high-order bits of $K_{1,14}$ and C_{14}, nor determine the value of the carry bit ϵ_{14}.

Step 4 Select those values of $T,\ 0 \le T < 2^{16}$ for which the bytes $X_{1,0} \ldots X_{1,13}$ remain constant.

These bytes will remain constant if the carry bit ϵ_{14} remains constant. Let $(C_{14,lo}||C_{15})$ denote the 15-bit integer resulting from the concatenation of $C_{14,lo}$ and C_{15}. Even though the actual value of ϵ_{14} is unknown, the reader can verify that it remains constant for those values of T satisfying

$$2^{15} - (C_{14,lo}||C_{15}) \ \le\ T \ <\ 2^{16} - (C_{14,lo}||C_{15}) \ . \qquad (6)$$

This gives a subset of T values for which, after applying the AddRoundKey transformation to X_1 and SubBytes transformation to Y_1:

– The 14 bytes $\{Z_{1,0} \ldots Z_{1,13}\}$ are unknown, but constant.
– The bytes $Z_{1,14}$ and $Z_{1,15}$ are known, and varying.

The remainder of the attack proceeds using only the 2^{15} traces corresponding to this subset of T values.

Step 5: Apply the ShiftRows and MixColumns operation to Z_1 to compute $V_1 = X_2$, the input to Round 2, manipulating unknown values symbolically.

Using Equation 3 (discussed in §2.2), it can be shown that X_2 has the following properties:

– Bytes $X_{2,0} \ldots X_{2,7}$ have the form

$$X_{2,j} = \tilde{X}_{2,j} \oplus E_{1,j} \ , \tag{7}$$

where $\tilde{X}_{2,j}$ are known and vary with T, and the $E_{1,j}$ are unknown, but constant with respect to T.
– Bytes $X_{2,8} \ldots X_{2,15}$ are unknown, but constant.

Step 6: Apply the Round 2 AddRoundKey transformation to X_2 to compute Y_2, manipulating unknown values symbolically.

For $X_{2,0} \ldots X_{2,7}$, the results are

$$\begin{aligned} Y_{2,j} &= (\tilde{X}_{2,j} \oplus E_{1,j}) \oplus K_{2,j} \\ &= \tilde{X}_{2,j} \oplus (E_{1,j} \oplus K_{2,j}) \\ &= \tilde{X}_{2,j} \oplus \tilde{K}_{2,j} \ . \end{aligned} \tag{8}$$

For these bytes, the AddRoundKey transformation is equivalent to XORing known and varying input data $\tilde{X}_{2,j}$ with constant "key" bytes $\tilde{K}_{2,j}$.

Step 7: Use DPA to recover $\tilde{K}_{2,0} \ldots \tilde{K}_{2,7}$ using $\tilde{X}_{2,0} \ldots \tilde{X}_{2,7}$ as known inputs into the relationship:

$$Z_{2,j} = S[\tilde{X}_{2,j} \oplus \tilde{K}_{2,j}] \ . \tag{9}$$

This step displays one of the most crucial (and interesting) features of the attack. We cannot use the correct values for $X_{2,j}$ as input to the DPA attack, since they are unknown. Instead, we treat the known values $\tilde{X}_{2,j}$ as the input. They differ from the correct values by fixed error terms $E_{1,j}$. The keys recovered are not the correct keys, but differ from them by the same fixed error terms. As Equations 8 and 9 show, these error terms then cancel when $Y_{2,j}$ and $Z_{2,j}$ are computed, leaving us with the correct values for them.

Since bytes $X_{2,8} \ldots X_{2,15}$ are unknown but constant, the corresponding bytes $Z_{2,8} \ldots Z_{2,15}$ are also unknown, but constant.

At the end of this step,

– $Z_{2,0} \ldots Z_{2,7}$ are varying, and known exactly.
– $Z_{2,8} \ldots Z_{2,15}$ are unknown, but constant.

Step 8: As in step 5, apply the ShiftRows and MixColumns operation to Z_2 to compute $V_2 = X_3$, the input to Round 3, manipulating unknown values symbolically.

Again, the result can be expressed in terms of a known vector $\tilde{X}_3$ as:

$$X_3 = \tilde{X}_3 \oplus E_2 \ ,$$

where

– E_2 is a vector of 16 unknown, but constant bytes.
– Every byte of $\tilde{X}_3$ is known and variable.

Step 9: Use DPA to recover Z_3.

The attack uses $\tilde{X}_3$ as the known variable input, and recovers $\tilde{K}_3$, and all 16 correct bytes of Z_3.

Step 10: Given all 16 correct values of Z_3, apply the ShiftRows and MixCols operation to obtain $V_3 = X_4$.

Note that at this point all the error terms are gone, and X_4 is the *correct* input to round 4.

Step 11: Use DPA to obtain K_4 using the known, variable Round 4 input X_4.

If 24 or 32-byte keys are used, repeat Steps 10 and 11 in Round 5 to recover another round key.

Step 12: After recovering enough round keys, reverse the key schedule and determine the original AES key.

4 Results

The attack was implemented against a smart card performing AES-128 in counter mode with unknown initial counter value.

Step 1 Power traces were collected during 2^{16} sequential encryptions.

Step 2 A custom program was written to search over $K_{1,15,lo}$, $C_{15,lo}$, and b_{15}, using the input values of T_{15} to generate predicted values of $Z_{1,15}$. Then DPA was used to evaluate the predictions. The analysis took about 11 minutes on a Dell workstation and revealed that $(K_{1,15,lo}, C_{15,lo}, b_{15}) = (\texttt{30h}, \texttt{42h}, \texttt{0})$.

Because evaluating 2^{15} difference traces individually is somewhat tedious, we measured and plotted the average square of the total amplitude of the differentials observed in each differential trace.

In addition to the primary spike at $(\texttt{30h}, \texttt{42h}, \texttt{0})$, this representation reveals secondary harmonics at $(\texttt{70h}, \texttt{02h}, \texttt{1})$, $(\texttt{10h}, \texttt{62h}, \texttt{0})$, and a few other locations as expected. Note that the AES substitution table is extremely flat, and does not contribute these harmonic peaks. Rather the spikes observed in this figure are due to relationships between the input parameters, stemming from the structure of the XOR and ADD combination.[2]

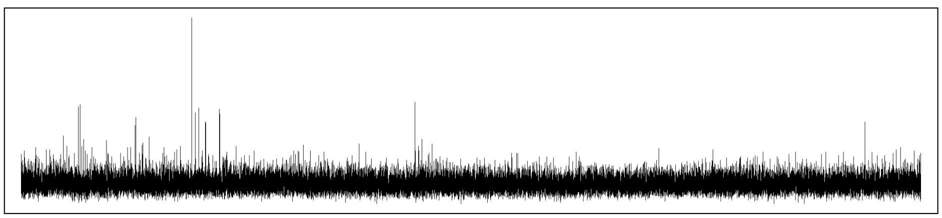

Fig. 2. DPA search results in compact form, showing primary spike for $K_{1,15,lo} = \texttt{30h}$, $C_{15,lo} = \texttt{42h}$, and $b_{15} = \texttt{0}$ at offset 6210 of 32678.

Step 3 A standard DPA attack was implemented to recover $C_{15,hi}$, $K_{1,14,lo}$, $C_{14,lo}$, and b_{14}, using T_{14} as the primary input, and T_{15} and $C_{15,lo}$ to derive the ϵ_{15}. This information was then used to compute $Z_{1,14}$.

The analysis revealed that $C_{15,hi} = 0$, $K_{1,14,lo} = \texttt{65h}$, $C_{14,lo} = \texttt{35h}$, and $b_{14} = \texttt{0}$.

Step 4 The recovered values

$$(C_{14,lo}, C_{15}) = (\texttt{35h}, \texttt{42h})$$

[2] Consider the eight-bit construction $Y = f(K, C, X) = K \oplus (C + X) \textbf{ mod } 256$. There are fifteen "harmonic" values of (K_i, C_i) for which $f(K_i, C_i, X) = f(\texttt{30h}, \texttt{42h}, X)$ for half of the possible values of X. With these related keys the input to the SubBytes would be correct exactly 50% of the time. At the output of the SubBytes operation, individual *bits* of $Z_{1,15}$ are correct about 75% of the time, leading to high-amplitude spikes in single-bit analysis. The Hamming weight of $Z_{1,15}$ is correct $57\% - 64\%$ of the time for these related keys. If correlation or other multi-bit analysis methods are used the statistical significance of the harmonic spikes will be observed with greater clarity.

were used to determine the values of T for which bytes 0..13 of $C + T$ remain constant. These values are given by

$$2^{15} - \texttt{3542h} \leq T < 2^{16} - \texttt{3542h} \qquad \Rightarrow \qquad 19134 \leq T < 51902 \; .$$

For the remainder of the attack, only those power traces for which T is in this range were used.[3]

Step 5 In this step we need to apply the ShiftRows and MixColumns to the Z_1 to compute X_2. Only $Z_{1,14}$ and $Z_{1,15}$ are known, however, and our analysis software is not configured to handle variables symbolically. As noted in Step 7, however, the DPA attack to recover the eight bytes $Z_{2,0} \ldots Z_{2,7}$ is unaffected by the actual values of the error terms $E_{1,j}$ derived from the unknown bytes of Z_1. Hence, to complete this step, we substituted the value 0 for each unknown constant byte. We were then able to perform the ShiftRows and MixColumns transformations using our existing software.

Steps 6,7 DPA was used to recover "key" bytes

$$\tilde{K}_{2,0} = K_{2,0} \oplus E_{1,0} \quad \ldots \quad \tilde{K}_{2,0} = K_{2,7} \oplus E_{1,7} \; ,$$

using input data

$$\tilde{X}_{2,0} = X_{2,0} \oplus E_{1,0} \quad \ldots \quad \tilde{X}_{2,7} = X_{2,7} \oplus E_{1,7} \; .$$

The 8 bytes of $\tilde{K}_{2,0} \ldots \tilde{K}_{2,7}$ recovered were:

$$\tilde{\mathrm{K}}_2 = \texttt{30451E9FD1923450----------------h}$$

Given the $\tilde{K}_{2,j}$ and $\tilde{X}_{2,j}$ we calculated the correct values $Z_{2,0} \ldots Z_{2,7}$ by:

$$Z_{2,j} = S[\tilde{X}_{2,j} \oplus \tilde{K}_{2,j}]$$

Step 8 At this point, bytes $Z_{2,0} \ldots Z_{2,7}$ are known and variable, and the remaining $Z_{2,j}$ are unknown but constant. As in Step 5, the unknown $Z_{2,j}$ are set to zero, and the second round completed. All 16 bytes of $V_2 = X_3$ have the form

$$X_{3,j} = \tilde{X}_{3,j} \oplus E_{2,j}$$

where the $\tilde{X}_{3,j}$ is known and variable, and $\tilde{E}_{2,j}$ are unknown but constant.

Step 9 As in Step 7, the $\tilde{X}_{3,j}$ were used as known input to a DPA attack to recover $\tilde{K}_{3,j}$ and $Z_{3,j}$. All 16 bytes of Z_3 were recovered, as was the entire key $\tilde{K}_3$.

$$\tilde{\mathrm{K}}_3 = \texttt{7A610872DE8FE299708A89A85DD9914Dh}$$

[3] The signal-to-noise levels observed in this dataset were sufficiently high that we actually performed the attack on round two using only 2^{13} traces.

Step 10 With all 16 values of Z_3 known, we simply completed the round to compute $V_3 = X_4$, the correct input to Round 4.

Step 11 We performed standard DPA using correct, known variable inputs X_4 The following key was recovered:

$$\mathrm{K}_4 = \texttt{52438AAD476E016D31EAE1CDAE8E0F3Dh}$$

Step 12 Since the target of this attack was performing AES-128, at this point we had sufficient material from the key schedule to compute the correct input key. Running the key schedule backwards gave:

$$\begin{aligned}
\mathrm{K}_4 &= \texttt{52438AAD476E016D31EAE1CDAE8E0F3Dh}\\
\mathrm{K}_3 &= \texttt{156B0676152D8BC07684E0A09F64EEF0h}\\
\mathrm{K}_2 &= \texttt{F6C0556800468DB663A96B60E9E00E50h}\\
\mathrm{K}_1 &= \texttt{CC8D5116F686D8DE63EFE6D68A496530h}
\end{aligned}$$

Therefore, the 128-bit AES key recovered by this attack is equal to

$$\mathbf{K} = \texttt{CC8D5116F686D8DE63EFE6D68A496530h}$$

and the attack is complete.

5 Concluding Remarks

In this paper we described a first-order DPA attack against AES in counter mode with an unknown counter. We introduced a technique to shift unknown constant data onto round keys such that they could be effectively ignored. This compensates for the unknown counter value, as well as the counter mode property that only the low-order bytes of the input change.

The techniques presented here were used to mount an attack against a smart card implementation of AES in counter mode. The attack required only 2^{13} traces pulled from a set of 2^{16} sequential operations. The same technique might still succeed using 2^8 or fewer sequential traces, if the leakage rates are sufficiently high.

Countermeasures that defend against first-order DPA attacks should be effective against this attack as well. Also, modifying the method by which the counter updates (using a linear feedback shift register, for example) would present a challenge to this attack.

The techniques in this paper can be applied to other cryptographic algorithms. In general, when an unknown constant is mixed with known variable data, DPA can be used to mount an attack if the mixing function is nonlinear. As we have shown, if the mixing function is linear, evaluation of the secret constant can often be postponed until an attack is possible.

References

1. Paul Kocher, Josh Jaffe, and Benjamin Jun. *Differential Power Analysis.* In Advances in Cryptology - CRYPTO 1999, LNCS **1666**, Springer-Verlag, 1999, pp. 388–397.
2. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, Pankaj Rohatgi. *Towards Sound Approaches to Counteract Power-Analysis Attacks* In Advances in Cryptology - CRYPTO 1999, LNCS **1666**, Springer-Verlag, 1999, pp. 398–412.
3. National Institute of Standards and Technology. *Advanced Encryption Standard (AES) (FIPS PUB 197).* National Institute of Standards and Technology. Federal Information Processing Standards Publication 197 (FIPS 197), November 2001.
4. M. Dworkin. *Recommendation for Block Cipher Modes of Operation:Methods and Techniques* National Institute of Standards and Technology. Special Publication 800-38A, December 2001.
5. Paul N. Fahn, Peter K. Pearson. *IPA: A New Class of Power Attacks* In Cryptographic Hardware and Embedded Systems - CHES 1999, LNCS **1717**, Springer-Verlag, 1999, pp. 173–186.
6. Rita Mayer-Sommer. *Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards* In Cryptographic Hardware and Embedded Systems - CHES 2000, LNCS **1965**, Springer-Verlag, 2000, pp. 78–92.
7. Stefan Mangard. *A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion* In ICISC 2002, LNCS **2587**, Springer-Verlag, 2002, pp. 343–358.
8. Thomas Messerges, Ezzy Dabbish, Robert Sloan. *Investigations of Power Analysis Attacks on Smartcards.* In Proc. USENIX Workshop on Smartcard Technology, 1999, pp. 151–162.
9. Hervé Ledig, Frédéric Muller, Frédéric Valette. *Enhancing Collision Attacks* In Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS **3156**, Springer-Verlag, 2004, pp. 176–190.
10. Kai Schramm, Thomas Wollinger, and Christof Paar. *A New Class of Collision Attacks and its Application to DES* In Fast Software Encryption - FSE 2003, LNCS **2887**, Springer-Verlag, 2003, pp. 206–222.
11. Kai Schramm, Gregor Leander, Patrick Felke, Christof Paar. *A Collision-Attack on AES Combining Side Channel- and Differential- Attack* In Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS **3156**, Springer-Verlag, 2004, pp. 163–175.
12. Paul Kocher, Josh Jaffe, Benjamin Jun. *Introduction to Differential Power Analysis and Related Attacks (Technical Report)*, `http://cryptography.com/resources/whitepapers/DPATechInfo.pdf` and `http://web.archive.org/web/19990504025809/www.cryptography.com/dpa/technical/index.html` via `http://tinyurl.com/244azs` and `http://tinyurl.com/2zgfc3`, 1998.
13. Josh Jaffe, Benjamin Jun, Paul Kocher. *Advanced Topics 1*, Presentation for the DPA Workshop, Chicago IL, Cryptography Research, May 14-15 1999.
14. Pierre-Alain Fouque, Frédéric Valette, *The Doubling Attack – Why Upwards Is Better than Downwards* In Cryptographic Hardware and Embedded Systems - CHES 2003, LNCS **2779**, Springer-Verlag, 2003, pp. 269–280.
15. Thomas Messerges. *Using Second-Order Power Analysis to Attack DPA Resistant Software* In Cryptographic Hardware and Embedded Systems - CHES 2000, LNCS **1965**, Springer-Verlag, 2000, pp. 238–251.

16. Suresh Chari, Charanjit Jutla, Josyula R. Rao, Pankaj Rohatgi. *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards* AES Second Candidate Conference, `http://csrc.nist.gov/encryption/aes/round1/conf2/papers/chari.pdf`, February 1999.
17. Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, Didier Moyart. *Power Analysis, What Is Now Possible*, In ASIACRYPT 2000, LNCS **1976**, Springer-Verlag, 2000, pp 489–502.
18. David A. McGrew, John Viega. *The Galois/Counter Mode of Operation (GCM)* National Institute of Standards and Technology. Draft Special Publication 800-38D. May 31, 2005.
19. Viega, J. and D. McGrew, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)*, RFC 4106, June 2005.