A Multi-Round Side Channel Attack on AES using Belief Propagation

Hélène Le Bouder[1], Ronan Lashermes[1], Yanis Linge[2], Gaël Thomas[3], and Jean-Yves Zie[4]

[1] LHS-PEC TAMIS, INRIA, Campus Beaulieu 35042 Rennes, France
[2] STMicroelectronics, 190 Avenue Coq 13106 Rousset, France
[3] Orange Labs, 44 Avenue de la République 92320 Châtillon, France
[4] CEA-TECH, Centre de Microélectronique de Provence 13120 Gardanne, France

Abstract. This paper presents a new side channel attack to recover a block cipher key. No plaintext and no ciphertext are required, no templates are built. Only the leakage measurements collected in many different rounds of the algorithm are exploited. The leakage is considered as a Hamming weight with a Gaussian noise. The chosen target is the Advanced Encryption Standard (AES). Bayesian inference is used to score all guesses on several consecutive round-key bytes. From these scores a Belief Propagation algorithm is used, based on the relations of the Key-Expansion, to discriminate the unique correct guess. Theoretical results according to various noise models are obtained with simulations.

Keywords: Side Channel Analysis, Hamming Weight, AES, Key Expansion, Multi-Round Attack, Bayesian Inference, Belief Propagation.

1 Introduction

Security is a key component for information technologies and communication. Even if an encryption algorithm is proved secure mathematically, cryptanalysis has another dimension: physicals attacks. These attacks rely on the interaction of the computing unit with the physical environment.

The Side Channel Analyses (SCA) are physical attacks based on observations of the circuit behavior. They exploit the fact that some physical values (timing, power consumption, electromagnetic emissions (EM)) of a device depend on intermediate values of the computation. This is the so-called leakage of information of the circuit.

The Advanced Encryption Standard (AES) has been chosen as a target because it is the most widespread block cipher. Yet, our approach would be the same for any other block cipher.

Motivations: Generally the SCA as in [1,2] links a text with a measurement. This induces that the attack is often on the first or last round. But the framework described in [3] suggests that other kinds of attack-path are possible.

Our first idea is to draw an attack-path linking a leakage measurement with another one. Another approach in SCA are template attacks [4,5] which compare traces from the targeted device with traces from a profiling device.

In this paper, the main motivation was to build an attack which uses only traces, no text and no template. We are in the case of an attacker who can just observe a leakage but has no access to the device's input/output.

The great majority of side-channel attacks published in the literature follows a divide and conquer strategy. In the case of AES, bytes of a round-key are attacked one at a time. So the other main motivation in our approach is to attack different round-keys of the AES and use links between them to improve the probability to find the correct guess.

Contribution: This paper presents a new side channel attack. It is a multi-round attack, where no template and no texts (neither plaintexts nor ciphertexts) are used; only leakage measurements are required. In our attack the leakage is considered as a Hamming weight with a Gaussian noise. Bayesian inference is used to obtain scores for the possible values of the different round-key bytes. Then, the main idea is to use a Belief Propagation (BP) algorithm to cross information between them, in order to have a key which respects the rules of KeyExpansion.

Organization of this paper: The paper is organized as follows. The general context is first introduced in section 2. Our attack is divided in two steps. A first analysis on each round is described in section 3. Then in section 4, the results of the analysis of this first step are linked using the BP algorithm. Results are presented in section 5. Finally the conclusion is drawn in section 6.

2 Preliminaries

2.1 The targeted encryption algorithm: AES

The algorithm: The Advanced Encryption Standard is a standard established by the NIST [6] for symmetric key cryptography. It is a block-cipher. The encryption first consists in mapping the plaintext T of 128 bits into a two-dimensional array of $4 \cdot 4 = 16$ bytes called the State. Rows and columns are respectively noted l and c. Then, after a preliminary xor (the bit-wise xor is noted $\oplus$) between the input and the key K_0, the AES executes 10 times a round-function that operates on the State. The operations used during these rounds are:

- **SubBytes**, composed of non-linear transformations: 16 S-boxes noted SB, working independently on individual bytes of the State.
- **ShiftRows** noted SR, a byte-shifting operation on each row of the State.
- **MixColumns** noted MC, a linear matrix multiplication on $GF(2^8)$, working on each column of the State.
- **AddRoundKey** a xor between the State and the round-key K_r, $r \in [\![0, 10]\!]$.

The derived key: K denotes the master key. The size of the master key is 128 bits. K_r is the round-key used at round r, K_r is represented by a two-dimensional array of $4 \cdot 4$ bytes, like the State. $K_r^{l,c}$ is the round-key byte at row l and column c. The round-key K_{r+1} depends on the round-key K_r with $K_0 = K$. More precisely, the round-keys are computed with a KeyExpansion function described by the system of equations (1), where SB is the S-Box function and Rcon is a constant matrix of size $4 \cdot 10$.

$$\begin{cases} K_{r+1}^{l,0} = K_r^{l,0} \oplus Rcon(l,r) \oplus SB\left(K_r^{l+1 \bmod 4,3}\right) & \forall l \in [\![0,3]\!] \\ K_{r+1}^{l,c} = K_r^{l,c} \oplus K_{r+1}^{l,c-1} & \forall l \in [\![0,3]\!] \text{ and } c \in [\![1,3]\!] \end{cases} \quad (1)$$

2.2 Overview of our attack and state of the art

In this paper, we wanted to build an attack which uses only traces, no text and no template. In our attack, all the round-keys have already been precomputed, as is the case of most software AES implementations. So using the leakage of the KeyExpansion as done by Mangard in [7] is impossible, the attacker just observe leakage from AES round functions.

Actually in the state of the art, there are two kinds of approaches in SCA.

One consists in a divide and conquer strategy to attack one part (*e.g.* byte) at a time, as in classical attacks [1,2]. An attacker gives a score (for example a probability or a correlation) to each key byte guess. The difficulty is to enumerate all possible round-key guesses in a way that minimizes the rank of the correct round-key byte. This problem is indeed the focus of many papers as [8,9].

Recently in different works as [10,11,12,13,14], a new method consists in directly using links between variables of an algorithm. Information of each trace feeds a BP algorithm or a SAT-solver which usually converges to the correct key. The first time that BP was used in SCA on AES, was in the attack of Veyrat-Charvillon *et al.* [14]. They use BP on the whole AES algorithm to derive a global template attack.

Grosso *et al.* [15] compared both approaches and concluded that BP is a little bit better.

The presented work was made in parallel and independently from the recent works [14,15]. The strategy was chosen to be applied to a real experimental attack. Our attack presented is divided in two parts:

1. a divide and conquer attack focuses on key bytes of different rounds (§ 3);
2. linking the information obtained for each byte of every round-key (§ 4).

So, one contribution in our attack is to merge these two approaches and take advantage of both.

2.3 Attack-path

An attack-path is an exploitable relation between some observables (data or measurements) and the target, here the key K. Generally the attack-path in SCA

as [1,2] links a text with a measurement. In our attack, we want to link two EM or power leakage measurements. Finally the attack-path is between two rounds (AddRoundKey of round r and the S-boxes of round $r+1$); it is illustrated in Fig. 1. The attacker has no text at her disposal, she then needs an important leakage as it is her only source of information. More precisely, the most leaking functions in AES are MC and SB, so the leakages used in this attack are at the output of both computations.

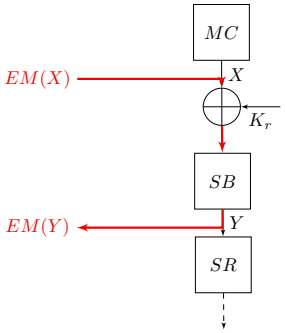

Fig. 1: Attack path

Hence this attack is possible on every rounds except rounds 0 and 10. Indeed there is no MC before the xor with K_0 and there is no SB after K_{10}.

In the following, $\mathcal{K}$ denotes the discrete random variable on a targeted key byte $K_r^{l,c}$, a guess is noted k and $\mathbb{K}$ is the set of guesses k, $\mathbb{K} = [\![0, 255]\!]$. The correct value is noted $\hat{k}$ (i.e. $K_r^{l,c} = \hat{k}$).

X denotes the discrete random variable on the byte at the input of AddRoundKey, an event is noted $X = x$ with $x \in [\![0, 255]\!]$. Likewise, Y denotes the discrete random variable on the byte at the output of SubBytes, an event is noted $Y = y$ with $y \in [\![0, 255]\!]$.

The mathematical model for the leakage is a Hamming weight (HW) with an additive Gaussian noise. This model is the classic model used in [1,2].

2.4 Theoretical attack-path

Before presenting our attack in practice, the relevance of the theoretical attack-path is proved; *i.e.* whether with only pairs (h_x, h_y) of Hamming weights (without noise) it is possible to deduce the value $\hat{k}$. This approach is similar to algebraic attacks [16,17,18].

The function HW is not invertible, the set $HW^{-1}(h)$, whose cardinal depends on the value of h, is the fiber of h by HW:

$$HW^{-1}(h) = \{x \text{ such that } HW(x) = h\} \quad .$$

For the good guess value $\hat{k}$, the following equation is verified:

$$SB(\hat{k} \oplus x) = y \quad .$$

The attacker has only pairs (h_x, h_y). For each pair only a subset of guesses k is possible. Let $\mathbb{K}_{(h_x,h_y)}$ be such a subset of guesses:

$$\mathbb{K}_{(h_x,h_y)} = \{k \text{ such that } \exists x \in HW^{-1}(h_x) \text{ and } HW\left(SB\left(k \oplus x\right)\right) = h_y\} \quad .$$

Let $\mathbb{K}(\hat{k})$ be the intersection of the sets $\mathbb{K}_{(h_x,h_y)}$ built with all 256 possible values of x, and the unknown and correct key $\hat{k}$:

$$\mathbb{K}(\hat{k}) = \bigcap_{x=0}^{255} \mathbb{K}_{(h_x,h_y)} \quad .$$

The correct guess $\hat{k}$ belongs to $\mathbb{K}(\hat{k})$. Thus, the first natural idea is to use a sieve to discriminate the wrong guesses.

We have studied all the cases for each key byte value $\hat{k}$. The sieve is not enough, because there exists some value $\hat{k}$ such that one wrong guess is not discriminated:

$$\mathbb{K}(\hat{k}) = \{\hat{k}, k\} \quad .$$

However it can be observed that the sets $\mathbb{K}(\hat{k})$ are all different, as illustrated in the following example. Besides they can be computed once and for all, for every possible value of the correct key $\hat{k}$; for example:

$$\begin{array}{l} \mathbb{K}(25) = \{25, 62\} \\ \mathbb{K}(62) = \{62\} \end{array} \quad .$$

If the attacker has used all possible pairs (h_x, h_y), she has computed the set $\mathbb{K}(\hat{k})$. Since all the sets $\mathbb{K}(\hat{k})$ are different, the attacker can discriminate the correct key $\hat{k}$. The attack-path is valid.

2.5 Leakage model

Our model for the leakage is a Hamming weight (HW) with an additive Gaussian noise. In this paper, for a given discrete random variable Z, the discrete random variable representing the Hamming weight of Z is noted H_Z, the event "the Hamming weight of Z is h_z" is denoted $H_Z = h_z$ for $h_z \in [\![0, 8]\!]$. H'_Z denotes the continuous random variable representing the "measured" Hamming weight; an event is noted $H'_Z = h'_z$, with $h'_z \in \mathbb{R}$ such as:

$$h'_z = h_z + \delta \quad ; \tag{2}$$

with δ an event of the Gaussian random variable $\mathcal{N}\left(0, {\sigma_Z}^2\right)$. For a continuous random variable H'_Z, $F_{H'_Z}$ denotes its probability density function. The probability density function associated to $\mathcal{N}\left(0, {\sigma_Z}^2\right)$ is given by:

$$\mathcal{F}_{\sigma_Z}(z) = \frac{1}{\sigma_Z \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{z}{\sigma_Z}\right)^2\right) \quad . \tag{3}$$

3 Attack on each round-key byte

In practice, the attacker does not have pairs of Hamming weights, but leakage measurements.

3.1 Points of interest

In order to use observed Hamming weights h'_z, the time when they can be observed in the trace must be identified. The points of interest, denoted PoI, are a set of points which correspond to the moments of information leakage. The detection of PoI is a critical point when performing an SCA attack but it is not the subject of this paper. We consider that PoI can be found using the method of [19], without a profiling phase.

3.2 Getting observed Hamming weights from physical measures

In this paragraph, the PoI are known and we assume that the noise follows a normal distribution $\mathcal{N}(0, {\sigma_Z}^2)$. Physical measures are obtained, as an example, with an oscilloscope.

The goal of our attack is to succeed without using a template approach, the attacker may not have a profiling device. In this case, the attacker has to guess the standard deviation σ_Z of the noise. A guess is noted σ_G. She considers the 9 theoretical Gaussian distributions $\mathcal{N}\left(h, \sigma_G^2\right)$ centered in the different Hamming weights $h \in [\![0, 8]\!]$. They are added to create a new distribution from which a new standard deviation σ_H is computed:

$$\sigma_H = std\left(\sum_{h \in [\![0,8]\!]} \binom{8}{h} \mathcal{N}\left(h, \sigma_G\right)\right) \quad .$$

She computes the mean $\overline{M}$ of the measured values $M = (m_i)_{1 \leq i \leq n}$ at a PoI. Then the observed hamming weights are computed as follow:

$$h'_i = \left(m_i - \overline{M}\right) \cdot \frac{\sigma_H}{std(M)} + 4 \quad .$$

If the attack succeeds, σ_G is a good approximation.

3.3 Bayesian Inference

In this part, the goal is to build a probability for each guess k given a set of measurements of n pairs $(H'_X, H'_Y) = (h'_x, h'_y)$ that a round-key byte $\mathcal{K}$ equals k. The main idea is to study the joint probability. This kind of approach is used in stochastic attacks[20] or in the attack of Linge *et al.* [19].

Throughout the paper, the following relations are used. A set of n pairs (h_x, h_y) is denoted $\{(h_x, h_y)\}_n$, the i-th pair is denoted $(h_x, h_y)_i$. Likewise a set

of n pairs (h'_x, h'_y) is denoted $\{(h'_x, h'_y)\}_n$ and the i-th pair is denoted $(h'_x, h'_y)_i$.

The probability A_k for a guess k given the measurements (H'_X, H'_Y), (4) is defined as follow.

$$A_k = \Pr\left[\mathcal{K} = k | \{(h'_x, h'_y)\}_n\right] \quad . \tag{4}$$

The context can be represented with a belief network (as in [21]). It is a graph where the nodes are variables as illustrated in Fig 2.

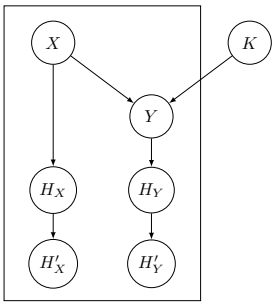

Fig. 2: Modeling the problem with a graph. An arrow means influence between two variables. The variables in the rectangle have a different value at each execution, while the value of the variables outside the rectangle is fixed throughout the attack.

At the start of the attack all guesses are equiprobables, the prior distribution is uniform:

$$\forall k \in \mathbb{K}, \quad \Pr\left[\mathcal{K} = k\right] = \frac{1}{256} \quad . \tag{5}$$

Probabilities $\Pr\left[(H_X, H_Y) = (h_x, h_y) | \mathcal{K} = k\right]$ can be precomputed once and for all by enumeration on the value x.

So the attacker wants to evaluate the probability of A_k, given by equation (4), i.e. the probability of $\mathcal{K} = k$ given a set of measurements. The Bayes theorem implies:

$$A_k = \frac{\overbrace{F_{(H'_X, H'_Y)}\left(\{(h'_x, h'_y)\}_n | \mathcal{K} = k\right)}^{A1_k} \cdot \Pr[\mathcal{K} = k]}{\underbrace{F_{(H'_X, H'_Y)}\left(\{(h'_x, h'_y)\}_n\right)}_{A0_k}} \quad .$$

The denominator $A0_k$ can be obtained by normalization, there is no need to compute it. The pairs $(H'_X, H'_Y)_i$ are independent and identically distributed; i.e. all the pairs have the same distribution of probabilities and all the pairs are mutually independent. It means that a pair $(H'_X, H'_Y)_1$ cannot be predicted with

the previous pair $(H'_X, H'_Y)_2$; thus:

$$A1_k = \prod_{i=1}^{n} \underbrace{F_{(H'_X,H'_Y)}\left((h'_x, h'_y)_i | \mathcal{K} = k\right)}_{A2_k} \quad .$$

Now, the probability of a single pair is needed.

$$A2_k = F_{(H'_X,H'_Y)}\left((h'_x, h'_y) | \mathcal{K} = k\right) \quad .$$

The law of total probability implies that:

$$A2_k = \sum_{(h_x,h_y)} \underbrace{F_{(H'_X,H'_Y)}\left((h'_x, h'_y) | (h_x, h_y)\right)}_{A3_k} \cdot \Pr\left[(h_x, h_y) | \mathcal{K} = k\right] \quad .$$

$$A3_k = F_{(H'_X,H'_Y)}\left((h'_x, h'_y) | (h_x, h_y)\right)$$

The pair (H_X, H_Y), the variable H'_X and the variable H'_Y are independent. For a fixed h_x, H'_X and H_Y are independent, thus:

$$F_{H'_X}\left(h'_x | (h_x, h_y)\right) = F_{H'_X}\left(h'_x | H_X = h_x\right) \quad .$$

Likewise, for a fixed h_y, H'_Y and H_X are independent, thus:

$$F_{H'_Y}\left(h'_y | (h_x, h_y)\right) = F_{H'_Y}\left(h'_y | H_Y = h_y\right) \quad .$$

Thus:

$$A3_k = F_{H'_X}\left(h'_x | H_X = h_x\right) \cdot F_{H'_Y}\left(h'_y | H_Y = h_y\right) \quad .$$

But $F_{H'_X}(h'_x | H_X = h_x)$ follows the normal distribution centred in h_x, so:

$$F_{H'_X}\left(h'_x | H_X = h_x\right) = \mathcal{F}_{\sigma_X}\left(h'_x - h_x\right) \quad .$$

Likewise:

$$F_{H'_Y}\left(h'_y | H_Y = h_y\right) = \mathcal{F}_{\sigma_Y}\left(h'_y - h_y\right) \quad .$$

Finally, the probability A_k, that a round-key byte $\mathcal{K}$ equals k for some given measurements of (H'_X, H'_Y), is proportional to the product [5]:

$$A_k \propto \prod_{i=1}^{n} \sum_{(h_x,h_y)} \mathcal{F}_{\sigma_X}\left(h'_{x,i} - h_x\right) \cdot \mathcal{F}_{\sigma_Y}\left(h'_{y,i} - h_y\right) \cdot \Pr\left[(h_x, h_y) | \mathcal{K} = k\right] \quad . \quad (6)$$

Note that, in the previous equation, the Gaussian noise hypothesis can be relaxed by replacing the Gaussian probability density functions ($\mathcal{F}_{\sigma_X}$ and $\mathcal{F}_{\sigma_Y}$) by whatever probability density function the attacker can come up with.

At the end of this part, the attacker has a probability for each guess k on every key byte of round 1 to 9.

[5] $h'_{x,i}$ is the i-th measurement h'_x.

4 Crossing information from round-key bytes with BP

4.1 Goal

The round-key bytes are linked by KeyExpansion relations (1). There are $16 \cdot 9$ round-key bytes linked together using $16 \cdot 8$ equations. It is supposed that the correct key is the one minimizing the ranks of its round-key byte values across all 9 rounds. In this part, this additional information is crossed with the estimations to improve the probabilities $\Pr[\mathcal{K} = k]$ and to have a key which respects the rules of KeyExpansion. To this end, in this part a technique known as Belief Propagation (or sum-product algorithm) [22] is used.

BP was first used by Gallager [23] for decoding low-density parity-check (LDPC) codes. It was then rediscovered by Tanner [24] and formalized by Pearl [25]. The first time that BP was used in SCA on AES, in the attack *et al.* [14], then it is studied in [15].

4.2 Factor Graph

The BP algorithm relies on a bipartite graph called a factor graph (or Tanner graph). To each node in the factor graph is associated some information.

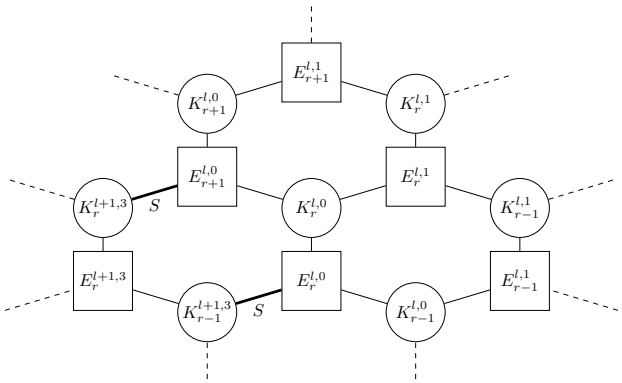

Fig. 3: Part of the factor graph associated with the AES KeyExpansion. Circles are variable nodes (round-key bytes) and squares are factor nodes (equations). Equations are labeled using the same indexes as the round-key byte they define, i.e. equation $E_r^{l,c}$ is the equation used to create $K_r^{l,c}$. The S-labeled edges remind the use of the S-box in the equation for that particular byte.

The nodes of a factor graph are of two kinds:

- variable nodes, in our case representing round-key bytes;
- factor nodes, in our case representing equations used in the KeyExpansion.

An edge links a variable node with a factor node, when the equation represented by the factor node involves the byte represented by the variable node. A part of the factor graph associated with the KeyExpansion is illustrated in Fig. 3.

In the following, the notation $N(\cdot)$ applied to a node is used to denote the set of neighbours of that node. Thus $N(\mathcal{K})$ is the set of equations involving round-key byte $\mathcal{K}$, and $N(E)$ is the set of round-key bytes composing equation E. Finally, a factor node E "is satisfied" when the corresponding KeyExpansion equation is satisfied.

4.3 Algorithm

Algorithm 1 Belief Propagation Algorithm.

Inputs: Experimental distributions A_k of every round-key byte $\mathcal{K}$; m the maximal number of iterations of BP.

Outputs: Final distributions B_k deduced using KeyExpansion relations.

for all round-key byte $\mathcal{K}$ **and** value k **and** equation E **do**

$$\mu_{\mathcal{K}\to E}(k) = A_k.$$

end for

for $j = 1$ **to** m **do**

for all round-key byte $\mathcal{K}$ **and** value k **and** equation E **do**

$$\mu_{E\to\mathcal{K}}(k) = \sum_{(k_1,k_2)\in\mathbb{K}^2} E(k,k_1,k_2)\cdot\mu_{\mathcal{K}_1\to E}(k_1)\cdot\mu_{\mathcal{K}_2\to E}(k_2)$$

with $N(E)\setminus\{\mathcal{K}\} = \{\mathcal{K}_1,\mathcal{K}_2\}$.

end for

for all round-key byte $\mathcal{K}$ **and** value k **and** equation E **do**

$$\mu_{\mathcal{K}\to E}(k) \propto A_k \prod_{E_1\in N(\mathcal{K})\setminus\{E\}} \mu_{E_1\to\mathcal{K}}(k).$$

end for

end for

for all round-key byte $\mathcal{K}$ **and** value k **do**

$$B_k \propto A_k \prod_{E\in N(\mathcal{K})} \mu_{E\to\mathcal{K}}(k).$$

end for

The BP algorithm in the general case is summed up in Algorithm 1.

In our case, the input of BP are the probabilities A_k found in (6). For a key byte $\mathcal{K}$, BP computes B_k a better belief , from the initial value A_k. As already stated, nodes in the factor graph exchange information messages with their

neighbours. More precisely, since the graph is bipartite, two types of messages are exchanged:

- variable to factor messages between a variable node (key byte) $\mathcal{K}$ and a factor node (equation) E, denoted $\mu_{\mathcal{K}\to E}(k)$;
- factor to variable messages between a factor node E and a variable node $\mathcal{K}$, denoted $\mu_{E\to\mathcal{K}}(k)$.

B_k is computed according to the input probability A_k and to the probabilities $\Pr[\mathcal{K}=k|E]$ conditional on factor node E in $N(\mathcal{K})$ to be satisfied using the following equation:

$$B_k \propto A_k \prod_{E\in N(\mathcal{K})} \Pr[\mathcal{K}=k|E] \quad . \tag{7}$$

$N(E)\setminus\{\mathcal{K}\}=\{\mathcal{K}_1,\mathcal{K}_2\}$. Thanks to the law of total probability, $\Pr[\mathcal{K}=k|E]$ can the be obtained by summing $\Pr[\mathcal{K}_1=k_1]\cdot\Pr[\mathcal{K}_2=k_2]$ over all the possible values for k_1 and k_2 such that factor node E is satisfied. Thus, the following equation holds:

$$\Pr[\mathcal{K}=k|E] = \sum_{(k_1,k_2)\in\mathbb{K}^2} E(k,k_1,k_2)\cdot\Pr[\mathcal{K}_1=k_1]\cdot\Pr[\mathcal{K}_2=k_2] \quad . \tag{8}$$

$\Pr[\mathcal{K}_1=k_1]$ and $\Pr[\mathcal{K}_2=k_2]$ are needed to compute $\Pr[\mathcal{K}=k|E]$ which depends on:

$$E\in N(\mathcal{K})\cap N(\mathcal{K}_1)\cap N(\mathcal{K}_2) \quad .$$

Hence, using Equation (7) directly on $\mathcal{K}_1$ and $\mathcal{K}_2$ would create a self-convincing loop for node $\mathcal{K}$. To avoid that problem, the factor corresponding to node E is removed from the product in Equation (7) in that case:

$$\Pr[\mathcal{K}_1=k_1] \propto A_{k_1} \prod_{E_1\in N(\mathcal{K}_1)\setminus\{E\}} \Pr[\mathcal{K}_1=k_1|E_1] \quad . \tag{9}$$

However, it can be shown [22,23] that the equations (7), (8) and (9) do not hold in general because they require an independence assumption on the probabilities used in the different products. In [26], authors show that in practice, the equation can be replaced by approximation, the BP gives excellent results. So, the equations (7), (8), and (9) are respectively replaced by:

$$B_k \propto A_k \prod_{E\in N(\mathcal{K})} \mu_{E\to\mathcal{K}}(k) \tag{10}$$

$$\mu_{E\to\mathcal{K}}(k) = \sum_{(k_1,k_2)\in\mathbb{K}^2} E(k,k_1,k_2)\cdot\mu_{\mathcal{K}_1\to E}(k_1)\cdot\mu_{\mathcal{K}_2\to E}(k_2) \tag{11}$$

$$\mu_{\mathcal{K}\to E}(k) \propto A_k \prod_{E_1\in N(\mathcal{K})\setminus\{E\}} \mu_{E_1\to\mathcal{K}}(k) \tag{12}$$

To complete the description of the BP algorithm, an initialization step is done before applying the above equations. The variable to factor messages $\mu_{\mathcal{K}\to E}(k)$ are initialized with the prior probabilities A_k corresponding to round-key byte $\mathcal{K}$.

In summary, after an initialization phase, BP works by alternatively applying equations (11) then (12) for every edge $(\mathcal{K}, E)$ in the graph. At the end of the execution, the returned value B_k is computed using equation (10). The number of iterations is not precisely defined but BP converges rapidly.

At the end, the attacker deduces from the BP outputs B_k, 9 probable round-keys. The attack succeeds if one of these 9 probable round-keys is an actual round-key, i.e. it is derived from the correct master key using the KeyExpansion.

Finally it is interesting to note that using BP for enhancing the probabilities on the different round-keys would work better as the number of rounds increases. Indeed, each new round brings independent information on the key that can be crossed with all other rounds.

5 Results

5.1 Simulation results

The simulations are done with the programming language julia (v0.4). They would correspond to an attack against a typical unprotected 8-bit software implementation of AES. Plaintexts are randomly generated. Measured Hamming weights are simulated with a noise according to $\mathcal{N}(0, \sigma^2)$ for various standard deviations σ. To facilitate the simulation the noise is supposed to be the same on X and Y: $\sigma = \sigma_X = \sigma_Y$.

We emphasize that to overcome floating point arithmetic issues, the normalization steps in both the Bayesian attack and the BP algorithm are not performed. As such, we work with scores corresponding to the logarithm of probabilities instead of probabilities directly.

Simulation to retrieve a key byte from pairs of noisy Hamming weights: First, at the level of a single byte using Bayesian inference (§ 3.3). For different noise standard deviation σ and different numbers of traces n, the average rank of the good key byte has been computed, for 100 simulated attacks for each possible value of the key $\hat{k}$. The results are displayed in Table 1.

Table 1: Average rank of the good key byte $\hat{k}$ according to the noise standard deviation σ and the number of traces n, for 100 simulated attacks for each possible value of the key $\hat{k}$.

$n \setminus \sigma$	0.1	0.2	0.3	0.5	1.0	1.5	2.0	3.0
100	1.2	1.3	2.3	14	66	96	107	119
1000	1	1	1	1	7.1	35	66	97
10000	1	1	1	1	1	2.2	12	48
100000	1	1	1	1	1	1	1.1	7.3

Using BP on simulation results: Now, the attack on the whole master key is simulated to see the additional benefit of BP. The algorithm returns 9 round-keys. The measure used here is then the minimum of the Hamming distances between the guessed round-keys and the correct round-keys, where the minimum is taken over the nine round-keys. The results are summarized in Table 2. As it

Table 2: Hamming distance between the best key found by BP and the correct master key K according to the noise standard deviation σ and the number of traces n, estimated over 100 simulated attacks.

$n \setminus \sigma$	0.1	0.2	0.3	0.5	1.0	1.5	2.0	3.0
100	0	0	0	0	59	51	53	54
1000	0	0	0	0	0	39	46	51
10000	0	0	0	0	0	0	0	40
100000	0	0	0	0	0	0	0	0

can be seen, the results are sharply separated, either the attack always succeeds or it always fails completely. Nonetheless, the number of traces required for the attack to succeed using BP is an order of magnitude below of what is required without BP. Finally the improvement of BP on the the attack is illustrated in the Table 3.

Table 3: Success of the attack according to the noise standard deviation σ and the number of traces n, for 100 simulated attacks. ✓ indicates the attack always succeeds even if not using BP, ✓ indicates the attack succeeds only with using BP and × indicates the attack fails.

$n \setminus \sigma$	0.1	0.2	0.3	0.5	1.0	1.5	2.0	3.0
100	✓	✓	✓	✓	×	×	×	×
1000	✓	✓	✓	✓	✓	×	×	×
10000	✓	✓	✓	✓	✓	✓	✓	×
100000	✓	✓	✓	✓	✓	✓	✓	✓

Using the BP algorithm considerably improves the success rate. It makes it possible to reduce the number of traces required for the attack to succeed. For example, when σ is equal to 0.1, only 100 traces are required thanks to BP, as opposed to 1000 traces without BP.

6 Conclusion

This paper presents a new side channel attack targeting the AES key. The first motivation for this paper was to realize an attack without texts and without templates, using only leakage measurements. The leakage is considered as a Hamming weight with an additive Gaussian noise. On each round 1 to 9 of the AES, two points of leakage are required to define the attack path without any text.

First, with a Bayesian inference approach a score is assigned to each round-key byte for all rounds from 1 to 9. Then, the second step is to use the KeyExpension rules to aggregate the knowledge on the round-key bytes to discriminate the correct key. A belief propagation is used for that purpose.

Simulation results have shown that the attack is effective, using the BP algorithm is a very good way to enhance the chances to recover the key. Even in the presence of a strong noise the attack can succeed. The BP algorithm approach can be used in combination with any other attack able to score all round-key bytes on several consecutive rounds. Additionally, it shows that increasing the number of rounds in a crypto-algorithm in order to make it resist classical cryptanalysis can weaken it with respect to our attack.

Finally, we would like to explore if masked implementations are effectively protecting against this attack.

Acknowledgment

The authors would like to thank Christophe Clavier (University of Limoges, France), Guillaume Reymond (Tiempo, France), Assia Tria (CEA-TECH, France), and Antoine Wurker (Eshard, France) for their valuable contributions to the development and understanding of the issues discussed in the paper. This work was initiated while Hélène Le Bouder was at École des Mines de Saint-Étienne. This work was partially funded by the French National Research Agency (ANR) as part of the program Digital Engineering and Security (INS-2013), under grant agreement ANR-13-INSE-0006-01 and by the French DGCIS (Direction Générale de la Compétitivité de l'Industrie et des Services) through the CALISSON 2 project.

References

1. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer.
2. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.
3. Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Bruno Robisson, and Assia Tria. A Unified Formalism for Physical Attacks. *IACR Cryptology ePrint*, 2014.
4. Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown Plaintext Template Attacks. In *WISA 2009*, volume 5932 of *LNCS*, pages 148–162. Springer.
5. Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template Attacks in Principal Subspaces. In *CHES 2006*, volume 4249 of *LNCS*, pages 1–14. Springer.
6. NIST. Specification for the Advanced Encryption Standard. *FIPS PUB 197*, 2001.
7. Stefan Mangard. A simple power-analysis (spa) attack on implementations of the aes key expansion. In *ICISC 2002*, pages 343–358. Springer.
8. Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *SAC 2012*, volume 7707 of *LNCS*, pages 390–406. Springer.
9. Sonia Belaïd, Jean-Sébastien Coron, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, and Emmanuel Prouff. Improved Side-Channel Analysis of Finite-Field Multiplication. In *CHES 2015*, volume 9293 of *LNCS*, pages 395–415. Springer.
10. Daniel P Martin, Jonathan F O'Connell, Elisabeth Oswald, and Martijn Stam. Counting Keys in Parallel After a Side Channel Attack. In *ASIACRYPT 2015*, pages 313–337. Springer.
11. Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and memory-efficient key recovery in side-channel attacks. *IACR Cryptology ePrint*, 795, 2015.
12. Benoît Gérard and François-Xavier Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting. In *CHES 2012*, pages 175–192. Springer.
13. Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? How to determine whether limited side channel information enables key recovery. In *Smart Card Research and Advanced Applications*, pages 215–232. Springer, 2014.
14. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *ASIACRYPT 2014*, pages 282–296.
15. Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with Enumeration: Which One Beats the Other and When? In *ASIACRYPT 2015*, pages 291–312. Springer.
16. Nicolas Courtois. How Fast can be Algebraic Attacks on Block Ciphers? In *Symmetric Cryptography*, volume 07021 of *Dagstuhl Seminar Proceedings*, 2007.
17. Harris Nover. Algebraic Cryptanalysis of AES: An Overview. *University of Wisconsin, USA*, 2005.
18. Nicolas T. Courtois and Gregory V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard. In *Cryptography and Coding 2007*, volume 4887 of *LNCS*, pages 152–169. Springer.
19. Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the Joint Distributions of a Cryptographic Function in Side Channel Analysis. In *COSADE 2014*, pages 199–213. Springer.

20. Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *CHES 2005*, volume 3659 of *LNCS*. Springer, 2005.
21. David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 04-2011 edition, 2011.
22. Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519, 2001.
23. Robert G. Gallager. Low-density parity-check codes. *IRE Trans. on Information Theory*, 8(1):21–28, 1962.
24. Robert M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. on Information Theory*, 27(5):533–547, 1981.
25. Judea Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *National Conference on Artificial Intelligence 1982*, pages 133–136. AAAI Press.
26. Sae-Young Chung, G. David Forney Jr., Thomas J. Richardson, and Rüdiger L. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 5(2):58–60, 2001.