Location-Dependent EM Leakage
of the ATxmega Microcontroller

Thomas Korak[✉]

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
`thomas.korak@iaik.tugraz.at`

Abstract. Nowadays, low power microcontrollers are widely deployed in wireless sensor networks, also implementing cryptographic algorithms. These implementations are potential targets of so-called side-channel analysis (SCA) attacks which aim to reveal secret information, e.g. a secret key. In this work we evaluate the resistance of AES implementations on an *Atmel AVR XMEGA* microcontroller against SCA attacks using the electromagnetic (EM) emanation measured at different locations on the chip surface from the front side and the rear side. Results show that the exploitable leakage for correlation attacks of a software implementation is higher compared to the leakage of the AES crypto engine, a hardware accelerator implemented on the microcontroller. Further investigations show that front-side EM measurements lead to better results and the measurement location is crucial if the number of measurements is limited.

Keywords: Microcontroller · CEMA · Wireless sensor networks · Wireless sensor nodes · AES

1 Introduction

Microcontrollers are widely used in all kinds of applications nowadays. One reason for the exhaustive usage is the great amount of functionalities they provide as well as their flexibility compared to application-specific integrated circuits (ASICs). One popular field of application is that of wireless sensor networks (WSN). WSNs consist of several sensor nodes which communicate with each other over a wireless channel. Each WSN typically has one base station which acts as the master in the network and forwards the received data from the sensor nodes to a backend system. The data transmitted by the sensor nodes varies depending on the field of application. WSNs are employed e.g. in healthcare systems, for environmental monitoring, energy monitoring or building administration. In order to make attacks like eavesdropping or data alteration infeasible the data is encrypted before transmission. Due to the data encryption additional computational costs are introduced. The additional computational costs need to be minimized because sensor nodes are typically battery powered and the battery lifetime needs to be maximized. In order to achieve a long battery lifetime,

efficient cryptographic primitives need to be used and implemented in an efficient way. Besides energy, code size and RAM size are also limited resources on sensor nodes.

One popular cryptographic primitive is the Advanced Encryption Standard (AES), a standardized block cipher. AES supports three key lengths (128 bits, 192 bits, or 256 bits) depending on the required security level. Software implementations of AES exist for nearly every microcontroller platform. Some microcontrollers (e.g. ATxmega, TI MSP430) also have an integrated AES hardware accelerator (AES crypto engine). The usage of an AES hardware accelerator allows faster data encryption/decryption and also parallel execution of other tasks during encryption/decryption. Besides the efficiency of the data encryption/decryption, the leakage of secret information (e.g., the secret key) caused by side channels has to be analysed. Typical side channels with a correlation with secret information are the runtime of the algorithm, the power consumption or the electromagnetic (EM) emanation of the device.

1.1   Related Work

The Rijndael algorithm [6] was standardized by the US National Institute of Standards and Technology (NIST) as the Advanced Encryption Standard (AES) in 2001. Today, AES is among the most frequently used block ciphers. In wireless sensor networks, AES is used to encrypt the transmitted data in order to prevent eavesdropping or data alteration, meaning that the block cipher is used in a special operation mode [15].

AES is proven to be mathematically secure, but side-channel analysis (SCA) attacks are still feasible due to implementation-specific properties. These type of attacks use some key-dependent leakage in the power consumption (e.g., by Kocher *et al.* [13]), the EM emanation (e.g., by Gandolfi *et al.* [8]) or the runtime (e.g., by Kocher *et al.* [12]) of the cryptographic algorithm.

There have been several SCA attacks on implementations of the AES on various platforms in the past. In [18] the authors show an AES key extraction of an FPGA implementation in less than 0.01 s. In this work the authors point out that the signal-to-noise ratio on the attacked device is 30 dB to 40 dB lower than an implementation on an ATxmega microcontroller. Kizhvatov *et al.* [11] have performed a power-analysis attack on the hardware accelerator included in ATxmega microcontrollers. This article serves as the basis for our work.

The AES implementation is often also a building block for higher-layer cryptographic protocols. In [9] the authors use an AES hardware accelerator for an authenticated encryption algorithm. In [20], the authors take advantage of an AES hardware accelerator in order to create hash-based signatures.

Several works also deal with the implementation issues for cryptographic primitives, which arise because of the existing constraints for sensor nodes used in WSNs. In [4], the authors focus on the energy efficiency of security algorithms for WSN devices. They also compare software implementations with hardware accelerators from the point of view of energy efficiency. Rehman *et al.* [16] compare different encryption techniques for message authentication codes (MACs)

in WSNs. In [21] a security mechanism for WSNs, called *MoteSec-Aware*, is presented. *MoteSec-Aware* is based on the AES.

1.2   Our Contribution

When studying the related work above, it is clearly observable that the AES is the most popular encryption algorithm for sensor nodes, so this algorithm is the target for our attacks. Atmel's ATxmega microcontroller is frequently used for sensor-node applications as the list of sensor nodes [22] shows. It also has a AES crypto engine integrated, which makes it the perfect device for our evaluations. In contrast to [11] we use the EM emanation of the device for the SCA attacks on the ATxmega microcontroller. One advantage of the EM side channel when compared to the power-consumption side channel is that no modification of the attacked circuit is required (e.g. inserting a resistor in order to enable power consumption measurements).

The contributions of this work can be summarized the following:

– We compare the EM leakage of a software AES implementation with the EM leakage of the AES crypto engine of the ATxmega microcontroller. Results show very similar leakage patterns for both implementations.
– EM signals are measured at different locations on the chip to find out at which point the maximum leakage appears. This information is helpful in scenarios where only a limited number of measurements is available and the approach can be applied on any device.
– We further compare two different power models for attacking the AES crypto engine, one based on the Hamming weight of an intermediate value and the other one based on the Hamming distance between two register values. The second power model has been used in [11].
– EM signals are measured from the front side and the rear side of the chip in order to compare the amount of exploitable leakage. Results show that front-side measurements lead to better results.
– We practically show that the number of required encryptions for a successful attack can be decreased by combining the EM signal in specific points.

1.3   Outlook

In Sect. 2 we introduce the microcontroller used in the experiments. A short introduction to the Advanced Encryption Standard as well as information about the software and hardware implementation is provided in Sect. 3. Section 4 explains the experimental setup as well as the results of the practical experiments. A discussion of the results can be found in Sect. 5. Some conclusions as well as future work are given in Sect. 6.

2   Used Microcontroller

In this section we introduce the microcontroller which was used for the evaluations. We decided to use an AVR XMEGA microcontroller, the ATxmega 256A3

to be exact. Due to the low power consumption, the high integration as well as the real-time performance this microcontroller is frequently used on wireless sensor nodes. Furthermore it has an AES hardware accelerator (in the following named as *AES crypto engine*) implemented.

In the following section we provide some more detailed facts about the ATxmega 256A3 microcontroller. The ATxmega 256A3 has 256 kB in-system self-programmable flash memory, 8 kB of boot-code section, 4 kB EEPROM, and 16 kB SRAM. The CPU is based on the AVR enhanced RISC architecture equipped with 32 general purpose working registers. The microcontroller can be operated with voltages between 1.6 V and 3.6 V and the maximum clock frequency is 32 MHz. Additional features are: One 16 bit Real-time counter, 16 bit timer/counter for PWM and compare modes, serial interface, ADCs, one DAC, 50 general-purpose I/O lines, and several other microprocessor-specific features. In addition to the AES crypto engine a DES accelerator is also included. The DES accelerator is out of the scope of this work, so no detailed description about this feature is provided. For more detailed information about the ATxmega 256A3 we refer to the datasheet [3]. Typical applications for this microcontroller are industrial control, factory automation, metering, and medical applications, to mention only a few. Although it is not explicitly noted as a high-security device we are sure that the cryptographic features the microcontroller provides are used frequently. So it makes sense to analyze possible weaknesses of these features.

3   AES Implementations

In this section, a short description of the AES algorithm in general is given followed by an introduction to the software AES implementation for the ATxmega microcontroller. The AES crypto engine is also introduced in this section.

3.1   Description of AES

The Advanced Encryption Standard (AES) is a symmetric block cipher. It supports a block size of 128 bits (equals state size) and key sizes of 128 bits, 192 bits and 256 bits. The cipher is round-based and generates one block of cipher text in 10, 12 or 14 rounds, depending on the used key length. Four transformations are performed on the state in each round: round-key addition (*add round-key*), byte substitution (*Sbox lookup*), byte permutation (*shift rows*), and a matrix operation (*mix column*). Further information about the AES can be found in [5,6].

For correlation attacks, which are performed in this work, an intermediate value (*IV*) of the attacked implementation has to be found which depends on a known input (e.g. plain text) and the secret key. In the case of AES the *IV* after the *Sbox lookup* in the first AES round is a popular choice. This value is calculated using $IV_i = Sbox(p_i \oplus k_i)$ where p_i is one byte of the known plain text and k_i is one byte of the secret key. For AES-128, both the plain text and the secret key have a length of 128 bits ($i = 1 \ldots 16$). The resulting cipher text

also has a length of 128 bits. The Hamming weight of IV_i, written as $HW(IV_i)$, serves as the power model. The Hamming weight of a value equals the sum of ones of its binary representation [14]. In the following we denote the key guesses for byte position i as $k_{g,i}$ and the correct key at byte position i as $k_{c,i}$.

3.2 Software AES Implementation

The software AES implementation is written in assembler. The implementation is optimized for fast encryption/decryption and no countermeasures against SCA attacks or fault attacks are implemented. The round key is calculated after each round and the byte substitution is implemented as a table lookup. It takes 4 054 clock cycles to encrypt one block of plain text. This is close to the 3 766 clock cycles given in [17].

3.3 AES Crypto Engine

The AES crypto engine requires 375 clock cycles to encrypt one block of plain text and it supports a key length of 128 bits. Comparing the run time of the AES crypto engine with the figures given for the software implementation it can be said that the AES crypto engine is approximately ten times faster. The usage of the AES crypto engine allows to perform other tasks in parallel to the encryption process.

Detailed information about the crypto engine can be found in [2]. In order to perform an encryption the plain text must be loaded into the *AES State Register* and the key into the *AES Key Register*. In the *AES Control Register* the decrypt bit has to be cleared and the encryption process is then started by setting the start bit. An interrupt as well as the status bit in the *AES Control Register* indicate when the encryption is finished. The cipher text equals the content of the *AES State Register*. The hardware crypto engine also supports the cipher block chaining (CBC) mode of operation in order to increase the efficiency when using CBC. For this purpose, it is necessary to set the XOR bit in the *AES Control Register*.

4 Practical SCA Experiments

In this section, the measurement setup used to record the EM traces is introduced. Then, a description of the performed experiments is given, followed by the results achieved for the software implementation and for the AES crypto engine, respectively.

4.1 Measurement Setup

In order to perform the measurements we have implemented a simple program on the microcontroller. After setting the AES key with an initial command, the plain text followed by a single control byte is sent to the microcontroller using the

serial interface. This control byte is used to select which implementation should be used for the encryption. The value *00h* indicates that the software implementation should be used and *01h* indicates that the AES crypto engine should be used. Just before the encryption process starts, one output pin (trigger pin) of the microcontroller is set to high. Setting the pin to low again indicates that the encryption is finished. In a last step the result is sent back to the control computer. This setup clearly indicates that we know the key of the attacked device. This fact simplifies the creation of two-dimensional EM-leakage landscapes.

A probe manufactured by *Langer EMV Technik* (model: *LF B 3*) has been used to measure the EM emanation of the chip. The signal of the probe was amplified with a 30 dB amplifier. The amplified signal was digitized using a *LeCroy WavePro 725Zi* oscilloscope. The sampling rate on the oscilloscope was set to 1 GS/s and the microcontroller was clocked with a frequency of 13.56 MHz.

Figure 1(a) shows the grid on the chip, where the probe has been placed in order to measure the EM signal, from the front side. The grid has a size of 9×9 points, leading to a total of 81 points. The distance between the points is 1 mm. Figure 1(b) shows the grid on the chip for the measurements from the rear side. Here, we have removed the package material in order to measure directly on the chip die. The chip die has a size of 5 mm $\times$ 5 mm. For the rear-side measurements, the focus was put on the die area, so a step size of 0.55 mm was used leading to a grid of 9×9 points again. In both cases, the probe has been moved using a stepper table. This allowed us to automate the measurement process up to a high degree. The amplitude of the measured EM signal varied for different points. Because of this observation, a calibration step in each point was performed before the traces were recorded. This calibration step ensures the same resolution of the voltage values in each point.

Figure 1(c) shows the board where the microcontroller is mounted on. Using this board it is hardly possible to perform power measurements without irreversible modifications. This is true for most real-world devices. Therefore, EM measurements are the best choice for measuring side-channel information. However, in order to perform the rear-side measurements, we had to develop a custom board which allows to mount the chip inverted.

4.2   Experiment Descriptions

Location-Dependent EM Leakage: For verification of the location-dependent EM leakage N EM traces were recorded in every point P ($P = 1 \ldots 81$). The measurement interval covered the first AES round and the same set of N random plain texts was used in each point. In total $81 \cdot N$ traces were recorded. A correlation EM analysis (CEMA) attack was performed for each of the points separately. In order to validate the success of the location-dependent correlation attacks the absolute value of the factor ρ_{border} (Eq. 1) was subtracted from the absolute maximum correlation coefficient of the correct key ($\rho_{c,P}$). ρ_{border} is a function of the number of used traces N and the derivation of ρ_{border} can be found in [14]. 99.99 % of all the correlation values are in this border. If $|\rho_{c,P}| > |\rho_{border}|$, the correct key can be distinguished from the remaining key guesses. The resulting vector $\bar{\rho}_{margin}$ has a

(a) EM measurement points, front side.

(b) Opened microcontroller, rearside.
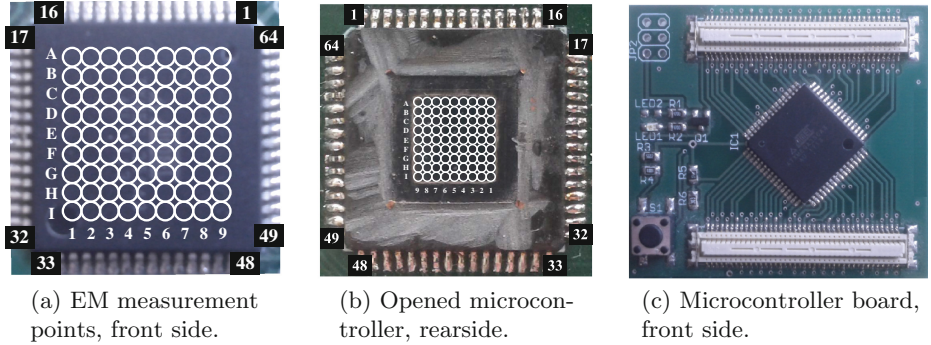
(c) Microcontroller board, front side.

Fig. 1. The attacked microcontroller.

length of 81 (Eq. 2). Positive values indicate that the attack is successful and negative values indicate that the attack fails in the specific point. For visualization reasons all negative values of $\bar{\rho}_{margin}$ were set to zero. In order to map the vector $\bar{\rho}_{margin}$ to the locations, the vector is transformed into a 9 x 9 matrix M according to Eq. 3. The 1st entry in the 1st row corresponds to point *A1*, and the 9th entry in the last row corresponds to point *I9*.

$$\rho_{border} = \pm \frac{4}{\sqrt{N}} \tag{1}$$

$$\bar{\rho}_{margin} = [|\rho_{c,1}|, |\rho_{c,2}|, \ldots, |\rho_{c,81}|] - |\rho_{border}| \tag{2}$$

$$M = \begin{bmatrix} \rho_{margin,1} & \rho_{margin,2} & \cdots & \rho_{margin,9} \\ \rho_{margin,10} & \rho_{margin,11} & \cdots & \rho_{margin,18} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{margin,73} & \rho_{margin,74} & \cdots & \rho_{margin,81} \end{bmatrix} \tag{3}$$

Relation Between EM Leakage and Number of Traces: In the next experiment the number of traces used for the correlation attacks was gradually reduced during several iteration. For each iteration the matrix M was visualized in a two-dimensional plot. The resulting plots visualize the leakage locations for different numbers of traces.

4.3   Results for Software Implementation

Location-Dependent EM Leakage: The first experiments targeted the AES software implementation. $N = 2000$ traces were recorded in each point, which leads to a total of $2000 \cdot 81 = 162\,000$ traces. This number was sufficient for the location-dependent EM leakage verification. For the correlation attack the Hamming weight of the output of the first substitution box, $HW(Sbox(p_i \oplus k_{g,i}))$, served as power model. Figure 2 shows the EM leakage landscape. The brighter the point is, the higher $\bar{\rho}_{margin}$ in that point. Black points indicate regions where
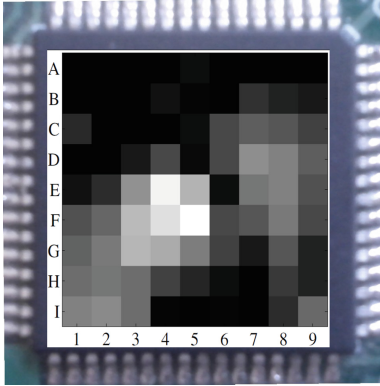
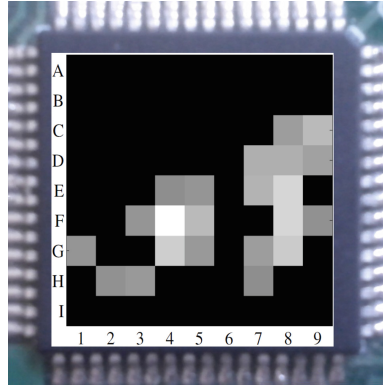**Fig. 2.** Landscape of the leakage of the AES software implementation using 2000 traces.

Fig. 3. Landscape of the leakage of the AES crypto engine using 10 000 traces and the Hamming weight power model.
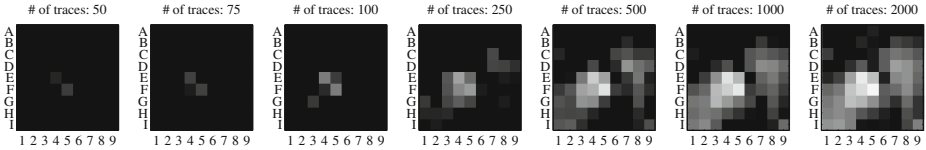

Fig. 4. EM-leakage landscape for different numbers of traces used for the CEMA attacks on the software AES implementation.

the attack yields a wrong key byte. The results discussed in this paragraph were achieved for key byte 1, the attacks targeting key bytes 2 to 16 lead to similar results. Due to the clear results, we did not further evaluate rear-side measurements for the software implementation.

Relation Between EM Leakage and Number of Traces: Figure 4 depicts the relation between CEMA success expressed by $\bar{\rho}_{margin}$ and the number of measurements used for the CEMA attack. With 50 measurements, the correct key can be extracted in two points. The fewer measurements are available, the more critical the location where the EM side-channel signal is measured.

4.4   Results for AES Crypto Engine

For the CEMA attacks targeting the AES crypto engine, two different power models were used. The first power model, which will be referred to as Hamming distance model in the rest of this paper, is the model used in [11]. This model takes into account two subsequent bytes of the plain text (p_i, p_{i+1}) and two bytes of the key $(k_{g,i}, k_{g,i+1})$: $HW((p_i \oplus k_{g,i}) \oplus (p_{i+1} \oplus k_{g,i+1}))$. The second model, named Hamming weight model, is the same model as is used for attacking
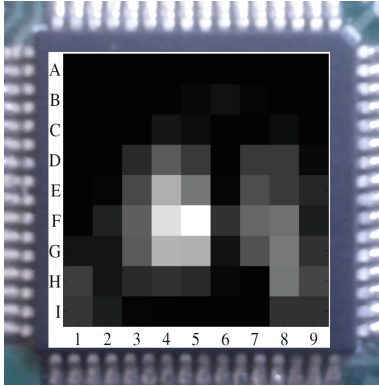
Fig. 5. Landscape of the leakage of the AES crypto engine using 10 000 traces, the Hamming distance power model and front-side measurements.
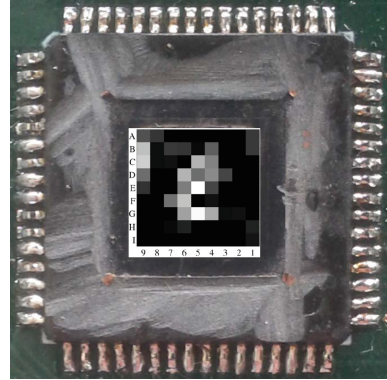
Fig. 6. Landscape of the leakage of the AES crypto engine using 10 000 traces, the Hamming distance power model and rear-side measurements.

the software implementation: $HW(Sbox(p_i \oplus k_{g,i}))$. The disadvantage of the Hamming distance model is that two 8 bit values $k_{g,i}, k_{g,i+1}$ are unknown and as a result the complexity of the attack increases. Instead of 2^8 key hypotheses, 2^{16} key hypotheses are required in theory. In [11] the authors show that the complexity can be decreased. To achieve this, they perform a full key recovery for all the 256 key values for the first key byte. Applying an exhaustive search, the correct key can be extracted from the 256 candidates in a final step. If the value of only one key byte is known (e.g., by using the Hamming weight model in a first step) all the other values can be revealed iteratively and the exhaustive search can be prevented. Furthermore, measurements captured from the front side and the rear side of the chip are analyzed.

Location-Dependent EM Leakage: For the CEMA attacks targeting the AES crypto engine, 10 000 EM traces were recorded in each point, once from the front side and once from the rear side. As mentioned in [11], 3 000 traces are sufficient to reveal the correct key bytes when using power measurements. Taking this as a reference point, 10 000 traces seemed to be a realistic value in order to achieve meaningful results.

First, CEMA attacks were performed in each point using the Hamming weight model. The landscape of the EM leakage in each point is plotted in Fig. 3 for front side measurements. It is clearly visible that there are several locations where the attack reveals the correct key byte value. With the Hamming weight model and rear-side measurements, no successful attacks could be performed.

Second, the experiments above were repeated using the Hamming distance model. Figure 5 shows the landscape of the EM leakage for front-side measurements and Fig. 6 shows the landscape of the EM leakage for rear-side measurements. Here, several locations can be detected where the attacks reveal the correct key byte for measurements from both sides of the chip. For these attacks

we assume that one key byte is known, resulting again in 2^8 key hypotheses. By comparing the results for the attacks using front-side measurements and rear-side measurements, the following observations can be made:

Attacks using the measurements from the front side lead to higher correlation values. Wires from the metal layers on top of the chip produce the exploitable EM signals. Therefore, front-side measurements seem to be better suited to capture these signals, although there is some package material between probe and chip surface. The fact, that measurements from the front side contain more exploitable leakage than rear-side measurements has also been observed by Heyszl *et al.* [10]. Here the authors target an FPGA.

Correlation values for the wrong key hypotheses are smaller when using rear-side measurements. Opening the chip from the rear side allowed us to minimize the distance between EM probe and chip die. This smaller distance minimizes the measurement noise leading to smaller correlation values for the wrong key hypotheses.

Relation Between EM Leakage and Number of Traces: Figure 7 depicts the relation between CEMA success expressed by $\bar{\rho}_{margin}$ and the number of measurements used for the CEMA attack. Bright points indicate a location yielding a successful attack ($\rho_{margin} > 0$). Black points indicate a location where the attack fails ($\rho_{margin} = 0$).

Plot (a) depicts the evolution when using the Hamming weight model. 4 000 traces are necessary in order to reveal the correct key byte. When less than 4 000 traces are used the attacks do not succeed, i.e. the correct key byte cannot be distinguished from the wrong key guesses. The more traces are used the more points leak key-dependent information. Plot (b) depicts the evolution when using the Hamming distance model together with front-side measurements. When applying this model, 1 000 traces are necessary in order to reveal the correct key byte. The more traces are used the more points leak key-dependent information. Plot (c) depicts the evolution when using the Hamming distance model together with rear-side measurements. Here, 1 000 traces are necessary in order to reveal the correct key byte. Increasing the number of traces decreases the influence of the location.

5   Discussion of the Results

The results presented in Sect. 4 show that a software AES implementation on a microcontroller which is not secured against SCA attacks, can be successfully attacked within seconds even without modifying the device. Measuring the EM emanation does not require any modification. The attack reveals the correct key byte with only 50 traces. For attacks with this low number of traces, the location where the EM side-channel signal is measured is critical. From Fig. 8, it can be seen that attacks lead to correct results in only two points. The higher the number of traces, the less critical the influence of the location is. When using 2 000 traces, attacks at 62 of the 81 locations lead to correct results.
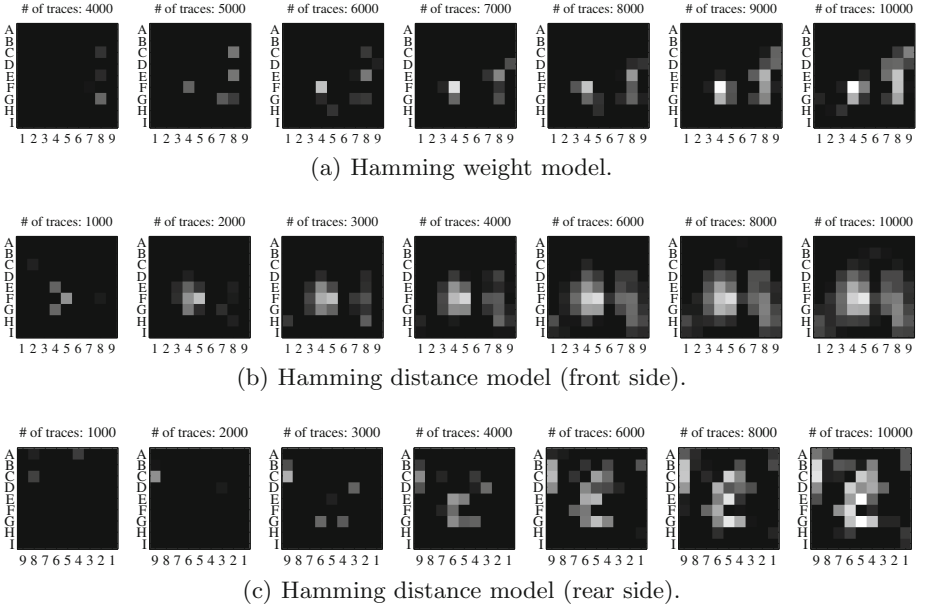
(a) Hamming weight model.

(b) Hamming distance model (front side).

(c) Hamming distance model (rear side).

Fig. 7. EM-leakage landscape for different numbers of traces used for the CEMA attacks on the AES crypto engine. Note that for (a) and (b) the covered area is $9\times9\,\mathrm{mm}$, while for (c) the covered area equals $5 \times 5\,\mathrm{mm}$.

The results achieved for the AES crypto engine clearly show that this hardware part is also not secured against SCA attacks. The leakage of the key-dependent signal is smaller, so more traces for a successful attack are required compared to the software implementation. Nevertheless, the factors of $\frac{1000}{50} = 20$ (Hamming distance model) and $\frac{4000}{50} = 80$ (Hamming weight model) do not indicate a security improvement compared to the software implementation. The additional measurement effort is negligible when keeping in mind the power of today's measurement equipment. As Fig. 8 shows, when using only 1 000 traces for the Hamming distance model, the attack only succeeds in six locations. The situation for the Hamming weight model is similar; here at least 4 000 traces are required and with this minimum number only the measurements at four out of 81 locations lead to successful attacks.

There are two important observations when comparing the results achieved for front-side and rear-side experiments. First, less measurements are required for a successful key recovery when front-side measurements are used although the chip for the rear-side measurements was opened to minimize the distance between probe and chip die. Interconnects in the metal layers on top of the chip create the EM signals and it seems that the silicon between probe and wires attenuates these signals stronger (rear-side measurements) than the package material (front-side). Second, the Hamming weight power model does not yield any successful attack when applied on the rear-side measurements for the AES crypto engine. Here, the assumption is that more than 10 000 traces are
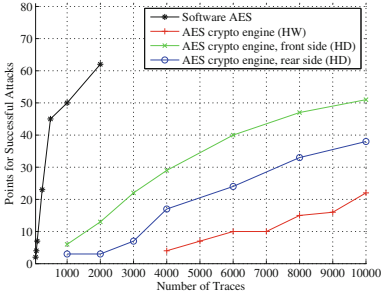
Fig. 8. Points for successful attack as function of number of traces used.
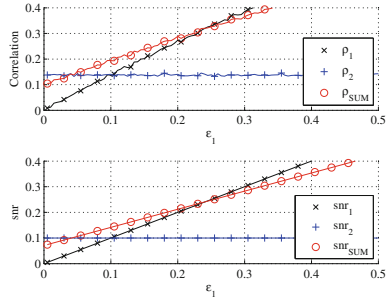
Fig. 9. Simulated leakage model.

required to succeed because also for the Hamming distance model double the amount of traces is required for a successful attack.

To draw a fair comparison between the four attack approaches, we give the number of traces, where at least 20 of the 81 locations (equals 24.7 %) lead to successful attacks. For the software implementation, 220 traces are required to achieve this result. Attacks on the AES crypto engine, based on the Hamming distance model, required 2 780 traces (4 800 traces for rear-side measurements, respectively) to succeed in at least 20 locations. When using the Hamming weight model, attacks in at least 20 locations succeed when more than 9 700 traces are used. Table 1 summarizes the depicted results from Fig. 8.

In order to further improve the attacks (i.e., reduce the number of traces required for a successful attack) we present an approach where the EM signals of two measurement points are combined. A comparable multi-channel SCA attack is presented in [1]. In this work the authors show how to combine power and EM measurements attacking a DES implementation. As a result the number of measurements can be significantly reduced. Elaabit *et al.* [7] present two combined attacks: In the first attack, a four-bit model and a mono-bit model are combined resulting in a higher success rate. The second attack uses a combination of samples at different time instances to improve the success rate for

Table 1. The minimum number of traces required in order to reveal the correct key byte value in at least *Nr. of locations* locations.

Nr. of locations	Percentage %	SW AES	AES crypto engine		
			HW	HD,fs	HD,rs
10	12.3	125	6 000	1 570	3 300
20	**24.7**	**220**	**9 700**	**2 780**	**4 800**
30	37.0	330		4 180	7 400
40	49.4	440		6 000	
50	61.7	1 000		9 500	
60	74.1	1 830			

correlation power-analysis attacks. Souissi *et al.* [19] propose two methodologies for combined attacks. In the first approach commonly used side-channel distinguishers are combined. In the second approach EM signals caused by two specific decoupling capacitors are combined. Both approaches lead to a significant improvement of the attacks. In our attack the assumption is that the attacker has two similar EM probes measuring the EM emanation on the chip surface. This fact enables the attacker to measure the EM emanation at two different points in parallel. This assumption is realistic as most modern oscilloscopes provide four independent channels which allow for the recording of up to four signals in parallel. If the traces of two points with a high leakage are combined, the result of the attack can be improved. The sum of the two traces has been used as combination function. The traces participating to the sum were recorded at two different locations during the encryption process of the same plain text.

Before we present the practical results, we want to give a theoretical explanation. The leakage in two points can be expressed mathematically as L_1 and L_2, according to Eq. 4. The values c_1 and c_2 are constants, V_1 and V_2 are functions of the correct key byte k_c and the known plain-text byte p ($V = f(p, k_c)$). r_1 and r_2 are normal distributed random values with zero mean and standard deviation one ($r \sim \mathcal{N}(0,1)$, $\sigma \cdot r = \mathcal{N}(0, \sigma^2)$), representing the noise. We further assume that the leakage functions in the two points are similar ($V_1 = V_2 = V$). The signal-to-noise ratio (SNR) is calculated as given in Eq. 5.

$$L_1 = c_1 + \epsilon_1 \cdot V_1 + \sigma_1 \cdot r_1 \qquad L_2 = c_2 + \epsilon_2 \cdot V_2 + \sigma_2 \cdot r_2 \qquad (4)$$

$$snr_i = \frac{\epsilon_i}{\sigma_i}; \quad i = 1, 2 \qquad (5)$$

Adding two traces in order to improve the attack, equals adding L_1 and L_2. The result of this summation is given in Eqs. 7 and 8, respectively. Equation 6 shows how to calculate the sum of two normal distributions with different values for the standard deviations. Equation 8 allows us to calculate the SNR for L_{SUM}, snr_{SUM} (Eq. 9). This result allows us to come to the conclusion, that the attack can only be improved by adding the leakage in two different points, if Eq. 9 holds.

$$\mathcal{N}(0, \sigma_1^2) + \mathcal{N}(0, \sigma_2^2) = \mathcal{N}(0, \sigma_1^2 + \sigma_2^2) = \sqrt{(\sigma_1^2 + \sigma_2^2)} \cdot \mathcal{N}(0, 1) \qquad (6)$$

$$L_{SUM} = L_1 + L_2 = (c_1 + c_2) + (\epsilon_1 + \epsilon_2) \cdot V + \sigma_1 \cdot r_1 + \sigma_2 \cdot r_2 \qquad (7)$$

$$L_{SUM} = c_{SUM} + \epsilon_{SUM} \cdot V + \sqrt{(\sigma_1^2 + \sigma_2^2)} \cdot r_{SUM} \qquad (8)$$

$$snr_{SUM} = \frac{\epsilon_{SUM}}{\sqrt{(\sigma_1^2 + \sigma_2^2)}}; \qquad snr_{SUM} > max(snr_1, snr_2) \qquad (9)$$

In order to visualize the relation given in Eq. 9, we have performed a simulation using MATLAB. We have varied ϵ_1, which influences snr_1, while snr_2 has been kept constant for the whole simulation run. Figure 9 depicts the results of the simulation. The lower plot shows the SNR and the upper plot shows the evolution of the correlation coefficients for the two points and the sum, respectively. In the range where Eq. 9 holds, the summation leads to an improvement.

Table 2. Results of the CEMA attacks before and after the combination of traces recorded at two different locations.

Software AES, (Hamming weight, front side)			Gain / %
Location	E4	F5	E4 and F5	
N_{min}	43	40	29	27.5
AES crypto engine, (Hamming weight, front side)			**Gain / %**
Location	F4	F8	F4 and F8	
N_{min}	3086	4444	2215	28.2
AES crypto engine, (Hamming distance, front side)			**Gain / %**
Location	F5	G4	F5 and G4	
N_{min}	396	683	295	25.5
AES crypto engine, (Hamming distance, rear side)			**Gain / %**
Location	E5	G5	E5 and G5	
N_{min}	1890	1850	1093	40.9

By improvement, we mean $\rho_{SUM} > max(\rho_1, \rho_2)$. In practice, the SNR values are typically not known. To circumvent this, we recommend the combination of only two points where the single attacks yield to similar correlation coefficients ρ_1 and ρ_2, respectively. From Fig. 9, it can be seen that the combination yields the best result if $\rho_1 = \rho_2$.

In order to prove that the theoretical assumptions also hold true in practice, we have combined points with similar correlation coefficients from the experiments above and analysed the results. For the attack targeting the software implementation $\rho_{c,P}$ could be increased from 0.630 to 0.740. For the attacks targeting the AES crypto engine, $\rho_{c,P}$ could be increased from 0.072 to 0.085 (Hamming weight power model). For the Hamming distance power model, $\rho_{c,P}$ could be increased from from 0.201 to 0.233 (front-side measurements) and from 0.093 to 0.121 (rear-side measurements) respectively. Table 2 lists the achieved results for the combination of measurements at two different locations. Furthermore the value N_{min} is given. This value equals the minimum number of traces required for a successful attack when the correlation value $\rho_{c,P}$ is given. N_{min} is calculated according to Eq. 10 and the deviation can be found in [14].

$$N_{min} = (\frac{4}{\rho_{c,P}})^2 \tag{10}$$

6   Conclusion

In this work we have shown the location-dependent EM leakage of the ATxmega microcontroller. We have compared the leakage of a software AES implementation with the leakage of the AES crypto engine, a hardware accelerator for AES encryptions. For the software implementation, less than 50 traces are sufficient

to reveal the AES key if the EM signal is measured at an appropriate position. Attacks targeting the AES crypto engine require about 700 EM measurements if captured from the front side and about 1 900 EM measurements if captured from the rear side. Again, an appropriate measurement position is a prerequisite. In all scenarios, the number of required measurements can be decreased if leakage information from two points are combined. Here, it is important that the signal-to-noise ratio (SNR) in the points which are combined is similar. This is shown in theory and verified in practical experiments.

The presented attacks pose a serious threat for sensor nodes, as the required number of EM measurements can be recorded in a few minutes and no modification of the circuit is required.

References

1. Agrawal, D., Rao, J.R., Rohatgi, P.: Multi-channel attacks. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 2–16. Springer, Heidelberg (2003)
2. Atmel. AVR1318: Using the XMEGA built-in AES accelerator (2008) (accessed 5 November 2013)
3. Atmel. 8/16-bit AVR XMEGA A3 Microcontroller (2013) (accessed 5 November 2013)
4. Botta, M., Simek, M., Mitton, N.: Comparison of hardware and software based encryption for secure communication in wireless sensor networks. In: Telecommunications and Signal Processing (TSP), pp. 6–10. IEEE (2013)
5. Paar, C., Pelzl, J.: Understanding Cryptography. Springer, Heidelberg (2010)
6. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. NIST AES Algorithm Submission (September 1999). http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
7. Elaabid, M.A., Meynard, O., Guilley, S., Danger, J.-L.: Combined side-channel attacks. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 175–190. Springer, Heidelberg (2011)
8. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
9. Gouvêa, C.P.L., López, J.: High speed implementation of authenticated encryption for the MSP430X microcontroller. In: Hevia, A., Neven, G. (eds.) LatinCrypt 2012. LNCS, vol. 7533, pp. 288–304. Springer, Heidelberg (2012)
10. Heyszl, J., Merli, D., Heinz, B., De Santis, F., Sigl, G.: Strengths and limitations of high-resolution electromagnetic field measurements for side-channel analysis. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 248–262. Springer, Heidelberg (2013)
11. Kizhvatov, I.: Side-channel analysis of AVR XMEGA crypto engine. In: Proceedings of the 4th Workshop on Embedded Systems Security, p. 8. ACM (2009)

12. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
13. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer (2007). ISBN 978-0-387-30857-9
15. National Institute of Standards and Technology (NIST). Special Publication 800–38A 2001 ED, Recommendation for Block Cipher Modes of Operation - Methods and Techniques (December 2001). http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
16. Rehman, S.U., Bilal, M., Ahmad, B., Yahya, K.M., Ullah, A., Rehman, O.U.: Comparison Based Analysis of Different Cryptographic and Encryption Techniques Using Message Authentication Code (MAC) in Wireless Sensor Networks (WSN) (2012). arXiv preprint arXiv:1203.3103
17. Rinne, S., Eisenbarth, T., Paar, C.: Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers (June 2007). http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/publications/conferences/lw_speed2007.pdf
18. Skorobogatov, S., Woods, C.: In the Blink of an Eye: There Goes your AES Key. IACR Cryptology ePrint Archive 2012:296 (2012)
19. Souissi, Y., Bhasin, S., Guilley, S., Nassar, M., Danger, J.-L.: Towards different flavors of combined side channel attacks. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 245–259. Springer, Heidelberg (2012)
20. Eisenbarth, T., von Maurich, I., Ye, X.: Faster hash-based signatures with bounded leakage. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 223–244. Springer, Heidelberg (2014)
21. Tsou, Y.-T., Lu, C.-S., Kuo, S.-Y.: MoteSec-Aware: a practical secure mechanism for wireless sensor networks. IEEE Trans. Wireless Commun. **12**(6), 2817–2829 (2013)
22. Wikipedia. List of Wireless Sensor Nodes – Wikipedia, The Free Encyclopedia (2013) (accessed 4 November 2013)