Secure and Efficient Design of Software Block Cipher Implementations on Microcontrollers

Authors' names omitted for double-blind review

Abstract The vast diffusion of microcontrollers has led to their employment in security sensitive contexts, where the need for trusted implementations of cryptographic algorithms is paramount. These architectures are usually endowed with software and occasionally hardware implementation of ciphers, but in both cases, the price envelope is the first figure to be optimized. The vast diffusion and the tight budget to which these devices are bound, has pushed for the design of efficient engineering solutions in terms of tradeoff between attack resistance and cost. The strongest threat to microcontroller security has been proven to be represented by side channel attacks: power consumption analysis and Electro-Magnetic (EM) emissions analysis being the prime opportunities to retrieve the secret key embedded in the devices via commonly overlooked information leakage. We propose an efficient solution to the problem of compromising EM emissions from an embedded device, showing which are the design space parameters available to the designer, and how to appropriately tune the security margin with respect to the performances, obtaining an order of magnitude improvement over the state-of-the-art solutions.

Keywords: Side-Channel Attacks; Embedded Systems Security; Differential Electromagnetic Attacks; Applied Cryptography.

Reference

Biographical notes:

1 Introduction

Nowadays, the most pervasive computation devices are the microcontrollers. These small, single chip devices are employed in a wide variety of application scenarios, in such an extensive way that their presence often goes unnoticed, thanks to their deep integration degree. The most prominent trend in this respect is represented by the Internet of Things (**?**): the growing trend of embedding small computing platform in a variety of common objects, ranging from the common household appliances to e-health devices, which may benefit from the possibility of communicating their state via Internet to interested stakeholders (the house owner and the physician respectively). Another widespread application of low cost microcontrollers is their use as means of building remote keyless entry systems (**?**) such as garage doors and car openers, or electronic locks for reinforced doors. In this case the microcontrollers employ challenge-response schemes to provide a secure digital counterpart to the common physical keys. Following the same line of thought, also digital currency, in the form of both smart cards employed as credit cards (**?**), and contactless near field payment systems is implemented on small microcontrollers (**?**). In this case, the microcontrollers deal with providing the owner with the means to safely authenticate payment orders, thus effectively hindering the counterfeiting of them by malicious individuals. The security margin provided by such systems raises effectively the bar of technical skills needed to counterfeit a payment order, without impacting negatively on the usability of the systems. The use of microcontrollers is also widespread in industrial and environmental monitoring sensor networks. In this case, these devices match the low cost and low power consumption requirements to implement efficiently and in a cost-effective manner the "Smart Dust" paradigm: the vision of a very large amount of sensors monitoring all the aspects of the production process and reporting the results in realtime.

To meet the requirements for all the aforementioned scenarios, the microcontroller is bound to a tight budget for the final price of the device. The typical microcontroller is bound to be a single chip device, which incorporates some permanent storage in the form of Flash memory, holding the applications, and a small amount of SRAM to hold the temporary values being computed. The device usually has no memory management unit, thus lowering the costs and the power consumption. This simplification in the architecture is mutuated by the fact that these devices run without a full fledged operating system on board. Depending on the required features, the microcontroller is endowed with a variety of I/O buses, which range from the common RS-232 serial port to USB and SPI buses, and usually sports one or more ADC to acquire the environmental data.

The working environments for smart embedded systems are characterized by their paramount requirement regarding the security of the system. In fact, all the aforementioned scenarios involve possible economical losses or even life-threatening situations in case a malicious individual subverts the regular functioning of the microcontrollers regulating environmental hazard surveillance or biomedical equipment. Due to the resource constrained execution setting, it is common to implement the security services of confidentiality, integrity and availability employing only a symmetric block cipher, and embedding the shared secret key in the device at deploy time. Common block ciphers employed to this end are the Advanced Encryption Standard (AES), Triple DES, PRESENT[1] and KASUMI[2]. This strategy, although particularly cost effective, is founded on the assumption that the secret key cannot be extracted from the internal flash memory by any means. To this end, the devices are equipped with physical means (lockdown fuses on silicon) which allow the designer to impede the extraction of the secret key. Nonetheless, an entire class of attacks targeting these systems aims at extracting the secret key exploiting Side-Channels over which sensitive information is leaked unintentionally. Side-channel attacks (SCA) rely on the fact that observing parameters of a cipher implementation yields information on the values being computed by the device, and thus also on the secret key being employed. Common side channels exploited by attackers are the power consumption of the device, its electromagnetic (EM) emissions, the time taken to compute the cryptographic primitive, or the faulty results obtained through disturbing the computation process (**???????**). The key principle of these attacks is to predict the behavior of a small, key dependent part of the device, guessing the portion of the key: this results in building a number of models of the device, which are subsequently compared with the actual measurements through statistical tools. The comparison reveals which one among them is the correct model , thus revealing the correct key portion.

Contributions
Our contribution in this work regards the description of the design space options available to a designer in an applicative scenario to secure cryptographic implementations of block ciphers on microcontroller platforms, and the description of a new strategy to reduce the performance penalties by an order of magnitude with respect to the current state-of-the-art. We tackled the resistance to EM emissions side channel attacks, as it is hard to solve this problem without significantly increasing the price envelope of the device through adding EM shields. By contrast, it is possible to hinder side-channel attacks based on power consumption through the use of a current flattening circuit, which can be embedded on the microcontroller at a negligible production cost (**?**). We have chosen as a case study the software implementation of the Advanced Encryption Standard on ARM-based microcontrollers. We will provide an in-depth analysis, validation and comparison of the resistance of the alternative implementations against EM-based side-channel attacks.

Paper Contents
This work is organized as follows: Section 2 provides the background notions on EM emissions and the EM side-channel analysis techniques employed by the attackers. Section 3 provides a security analysis of the countermeasures employed in literature. Section 4 elaborates on the computational complexity of SCA attacks against block cipher implementations to derive the protection requirements followed by our proposal for a secure and efficient block cipher SW implementation. Section 5 introduces the case study and describes the design alternatives of the cipher implementations. Section 6 reports the experimental results on a widely popular ARM-based microcontroller. Finally, Section 7 provides a summary of the related literature and Section 8 draws our conclusions.

2 The Electro-Magnetic Side-Channel

Any electrical or electronic device creates a so called electromagnetic (EM) environment as the electrons flow around the circuit with a variable rate. The intensity of the EM emissions depend on the geometric area of the circuit and on the rate of change of the electric current (or of the electrical potential difference). Indeed, the emitted EM field is mostly due to the common-mode currents (i.e. current flows without any other close-by opposing current) on conductive tracks. In addition, also the interference produced by any other source emitter (e.g., another part of the circuit or an external source) influences the EM environment of the target device via coupling effects that highly depend on the specific device geometry. Hence, the EM environment of the device can be ascribed to either base-band signals from direct emanations or signals unintentionally modulated at higher frequencies, which are not necessarily related to the working clock frequency. As the emissions of the circuit are correlated with the current flow within it, it is possible to gain a high amount of information on the ongoing computations through measuring them. Although a high precision measurement setup for these emissions, such as the one used in electromagnetic compatibility testing (EMC) is expensive and requires strong technical knowledge to be operated, it is possible to devise a cheap and simple measurement setup with off-the-shelf components, and operate it in a rather simple fashion, while still obtaining satisfactory measurements. In particular, it is possible to build an EM probe out of a simple coil of copper wire and sample the voltage drop at its ends, while it is placed close to the emitting circuit, thus measuring what are commonly defined as near-field EM emissions of the circuit. The whole workbench for such a measure is constituted by the aforementioned EM probe and a digital oscilloscope to sample and store the voltage drop values at the end of the probe.

2.1 EM Emission Model and Correlation Attacks

The main cause of EM emissions for digital circuits at a close range is represented by the metal wiring which connects the transistors on the silicon die. Due to the very large number of such wires and active components, modeling the EM emissions of an integrated circuit cannot be done with a sufficient accuracy through the common methods (e.g. superimposition of the effects of multiple equivalent dipoles, equivalent radiating structures). Due to design time placement constraints, the longest wires on the die are usually the memory buses; moreover the bus wires are usually placed as a sheaf and thus there is no destructive interference in their magnetic radiation, making these wires one of the main contributors for the emissions. However, albeit the contribution is smaller in entity, also the wiring of the computational logic provides a contribution to the global emission, which is still related to the data being computed.

Considering only the wires carrying the sensitive data, it is possible to model the data dependent EM emission employing the Hamming Weight of the values being carried. This is justified by the fact that the intensity of the magnetic field radiated by a wire is directly proportional to the entity of the current flowing through it. As this field is sensed through measuring the voltage drops at the ends of a coil, the measured quantity is proportional to the Hamming Weight of the carried value (**??**).

This relation between the entity of the emissions and the data being processed is the one exploited by EM side channel attacks to deduce the values of the secret key embedded in the device. As a first step the attacker selects an operation of the cryptographic primitive combining a portion (e.g. one byte) of the key with a known piece of data (e.g. the ciphertext or any other predictable value). The attacker models the emitted EM radiation during the selected operation, for each possible value of the key portion, and evaluates the correlation of the models with the actual measured emissions. This procedure is usually performed through evaluating the Pearson's correlation coefficient between the models and the actual measurement for a large number of inputs of the circuit, thus resulting in only the correct model having non negligible correlation (**???**).

3 EM Analysis Countermeasures

Methods and principles employed to put into effect security countermeasures against EM attacks are split into two categories: damping of the emitted signals and concealing or invalidating of the relation between the emissions hypotheses made by the attacker and the emission values measured from the device (**?**).

Known techniques to achieve a reduction of the signal strength either shrink the etching technology employed to build the chip or redesign the circuit layout to reduce the amount of leaked information via coupling ef-

fects (**???**). Another measure is to add an extra metal layer on the top of the chip or a grid of live wires, but this significantly raises production costs and does not suppress completely the emissions. To further reduce the effectiveness of the measurements, it is possible to add photosensitive circuitry to the halt the chip if its casing is removed through a decapsulation process, thus forcing the attacker to acquire the data on a packaged chip. In principle, all of these countermeasures will reduce the information leakage via the EM side-channel, thus increasing the amount of measurements to be performed to lead a successful attack up to an practically unfeasible or non-economically advantageous point, at the cost of an increased unitary price per chip. It is common to assume that even devices equipped with anti-tampering solutions can be vulnerable to EM analysis threats, if the adversary is able to collect an infinite amount of EM measurements and has access to sufficient computational power. To prevent such a powerful attacker from succeeding additional techniques rely on the randomization of the circuit clock and/or secret key refreshing. The former ones aim at de-synchronizing the measured signals in such a way to raise the technical effort needed to lead an attack, since time alignment is a crucial factor for the success. The latter ones are performed with "key usage" counters pointing out when a re-keying is needed, limiting the number of measurements an attacker will be able to collect.

3.1 Software EM Countermeasures

Software countermeasures aim at either spreading the information leakage on the time axis (known as hiding schemes) or preventing the attacker from knowing the actual input value employed in the sensitive operations via random masking values (known as masking schemes) (**??**). Although this comes with a performance penalty, software countermeasures are of prime interest due to their low cost and the usual lack of tight throughput constraints on microcontrollers.

The simplest way to diffuse the leakage over the time axis is to insert random delays in the execution of the cipher. The entity of the delay is picked within a limited range of clock cycles, to prevent excessive performance penalties: the effect on the measured emissions is a misalignment in the time instant where the sensitive operation takes place. Since adding random delays impacts substantially on performances, an alternate strategy is to reschedule in a random order a set of operations without any mutual data dependencies. The typical case is the one of memory lookups performed on a cipher's S-Box, or the addition of the round-subkey to different portions of the state. This technique, commonly known as *shuffling*, achieves the same effect of inserting random delays , as the operation being attacked is executed in different time instants. The performance penalty of this technique can be limited implementing the shuffling of the lookups through introducing an indirect indexing of the table via a temporary array. Both the insertion of random delays and the rescheduling increase the amount of measurements which should be performed by the attacker to obtain a statistically significant correlation between the measured emissions and the key-dependent model. Considering that the operation under attack can be performed within a window of n instructions, and assuming that the power consumption of the other operations is independent from the one under attack, the resulting correlation value for the time instant where the protected target operation is performed, will be reduced to one n-th of the one of an unprotected implementation. It is possible for the attacker to reduce the protection factor of these countermeasures up to $\sqrt{n}$ through pre-computing a sliding window sum of the measured values over a n instructions wide time window and subsequently applying the usual correlation attack methodology. In this case, one of the windows will always contain the attacked instruction, thus the sum of its measurements will always be correlated with the prediction, although it will be affected by a stronger random noise (leading to a correlation coefficient reduction by $\sqrt{n}$) (**?**).

The other approach to software countermeasures involves the use of random masks to hide the actual values being computed in the cipher from an attacker. The core idea is to protect the sensitive operation through applying one or more random masks to its inputs and remove them once the operation has been performed. The most typical case is the masking of the round key additions, performed via an `xor` operation, in block ciphers. In this case, a random masking value is added via `xor` to both the cipher state and the round key, subsequently the combination of the two results, again via `xor`, removes the masks and perform the correct key addition. The attacker will not be able to perform a correlation analysis due to both the fact that he no longer knows the input values of the actual key addition operation and he cannot deduce the value of the mask as it changes at each run.

4 Secure Block Cipher Implementation

This section provides a precise evaluation of the computational effort which should be carried out by an attacker to breach the security of a protected software implementation and reports our countermeasure scheme.

4.1 Security Evaluation of Protected SW Implementations

To lead a successful attack against a masking-protected cipher the attacker needs to know all the power consumption values of the operations involved in the masking procedure and must find a combination of them which is independent from the value of the random mask. This attack strategy, known as "high-order correlation attack" (HO-CA) (where the term "order" refers to the number of masks m) (**?**) needs to guess the time instants when the aforementioned operations are performed. HO-CAs

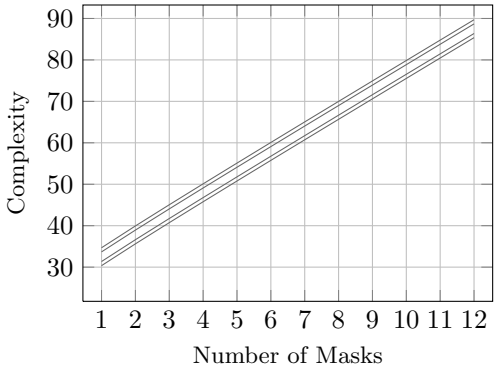


Figure 1 Computational cost (y-axis in base-2 logarithmic scale) to recover the whole key of a m-masked implementation of AES-128: $(4{\cdot}n_{tr}{\cdot}l{\cdot}2^8{\cdot}\frac{128}{8}){=}(16384{\cdot}n_{tr}{\cdot}l)$, with $n_{tr}{\in}\{100, 500, 1000, 5000\}$ and $l{=}\lceil 2.5{\cdot}(2m+1)\rceil$

derive a mask-independent value exploiting $2m+1$ measured values and combining them according to a chosen recombination function, and correlate it with the hypothetical values derived from the emission models. As the attacker does not know when the $2m+1$ operations are performed during the l samples long measurement, he needs to compute $\binom{l}{2m+1}$ possible outputs of the recombination function. As a correlation analysis must be conducted for each of the aforementioned choices, the computational effort will increase by a factor of $\binom{l}{2m+1}$.

Willing to provide a quantitative evaluation of the level of computational security against side channel attacks, we examine the amount of computations needed to perform a single correlation attack as a function of the number of measurements n_{tr}, and their length in samples l.We recall that, in case hiding schemes are in use, the number of traces will raise by a factor proportional to the square root of the delays introduced. Willing to consider the best case for an attacker we will dimension the security margin of our solution without the effect of the hiding. Assuming that the attacker is able to sample only the sensitive operations at 2.5 times the clock frequency of the device (to provide a safe margin above Nyquist's limit), he will obtain 2.5 samples per clock cycle (i.e. per instruction). Thus the effective length of a measurement for a m-mask implementation will be of $l{=}\lceil 2.5{\cdot}(2m+1)\rceil$ samples. Since the attacker will not need to recompute the hypotheses on the emission model for each of the correlation attacks to be led, the computational cost amounts to $(4{\cdot}n_{tr}{\cdot}l{\cdot}\texttt{num\_of\_key\_hypotheses}{\cdot}\texttt{portions\_of\_the\_key})$. Figure 1 shows computational complexity (in terms of the number of arithmetic operations), in base-2-logarithmic scale, of an attack against a m-masked implementation of the AES-128, employing a 8-bit key hypothesis, varying the number of masks from one to twelve. The four curves show the amount of computation assuming $n_{tr}{\in}\{100, 500, 1000, 5000\}$. From the figure it can be seen that, to obtain a computational complexity of 2^{80} for an attacker, 10 masks are required.

4.2 Proposed Design

Although masking techniques provide a quantifiable security margin, they suffer from the drawback of being computationally expensive when applied to nonlinear functions: open literature reports decreases in performance as high as 40× for a 4th-order masking on the AES S-box lookups (**?**). Since these operations are the most sensitive to EM-leakage, as the memory bus wires are particularly long and run parallel one to each other, the designer is forced to trade off a significant amount of performances to obtain a reasonable security margin.

We propose an alternative to the masking of the nonlinear functions which grants perfect protection under the presented EM emission model.

Proposition 4.1: *Assuming a Hamming Weight model for close range emissions, to protect the memory transfer operations of a software cipher implementation from EM based correlation attacks any key-driven memory access operation must assert on the memory buses only constant Hamming Weight values. To this end, an n-bit wide bus is employed to carry only $\frac{n}{2}$ bits of actual information, while the remaining bits must be set so that the total Hamming Weight is fixed to $\frac{n}{2}$.*

Proposed Scheme As it is usual for block ciphers to perform key-driven lookups of nonlinear functions tabulated in memory, an efficient way to meet the aforementioned requirements is to store the table values split into $\frac{n}{2}$-bit wide shares interleaved with $\frac{n}{2}$-bit wide compensation values. A choice for the compensation values to be employed is the bitwise negation of the share to be stored, as it always holds that $HW(\texttt{concat}(s, \neg s)) = \frac{n}{2}$.

The proposed scheme requires that two operations on the memory should be performed where the original cipher performed only one. This yields a worst-case performance penalty of 2×, which is far smaller than the one imposed by a masking scheme of order greater than 3. The doubling in size of the lookup tables of the cipher is not particularly taxing as the original size of the lookup tables is rather small (0.25 kiB–4 kiB) with respect to the current permanent memory sizes for microcontrollers (64 kiB–512 kiB). Note that the reconstruction of the actual value from the two shares must be performed in the CPU registers: this in turn implies that particular care must be exercised in order to prevent the compiler from spilling the values against the designer's will.

It is thus possible to design an EM-analysis protected block cipher implementation combining this technique and a high order masking of the round-key additions. This scheme allows the designer to obtain significant performance improvements over a pure masking-based implementation, while retaining the same security margin.

5 Case Study: AES encryption primitive

The most widespread block cipher implemented on microcontrollers is the AES, due to its standardization and the wide availability of efficient implementations. Moreover, its structure is a good representative of a large family of ciphers, thus making it a good candidate for a case study. The encryption primitive executes a number of round transformations on the input plaintext, where the output of each round is the input to the next one. The number of rounds r is determined by the key length: a 128-bit key uses 10 rounds, a 192-bit key uses 12 and a 256-bit key uses 14 (**?**). Each round is composed of the same steps, except for the first where an extra addition of a round key is inserted, and the last where the MixColumn operation is skipped. Each step operates on 16 bytes of data (referred to as the internal *state* of the cipher) generally viewed as a 4×4 matrix of bytes or an array of four 32-bit words, where each word corresponds to a column of the *state* table. The four round stages are: AddRoundKey (`xor` addition of a scheduled round subkey), SubByte (byte substitution by a lookup table (S-box)), ShiftRow (cyclical shifting of bytes), and MixColumn (linear transformation which mixes column state data). Given the cipher key k, the KeySchedule procedure outputs r+1 (16 byte wide) round subkeys.

The encryption procedure is amenable to several software implementations which trade-off memory and computational resources to obtain the best performance for the given architecture.

The SubByte operation is defined as the multiplicative inverse in the Galois field $\mathbb{F}_{2^8}$, followed by a constant affine transformation. The inverse of a 8-bit element $a{\in}\mathbb{F}_{2^8}$ is usually computed through either the extended Euclidean algorithm or a repeated "square & multiply" procedure to the following exponentiation: $a^{-1}{=}a^{254}{\in}\mathbb{F}_{2^8}$. The computational demands of a finite field inverse calculation make the fully computational implementation of the SubByte step inconvenient. The usual choice is to employ a pre-computed table (S-box) for all possible outputs of this step, thus implementing the operation through a simple table lookup.

Trading-off memory space vs. timing a little bit further, it is quite common to combine the SubByte, ShiftRow and MixColumn steps into a single set of tables lookups (**?**). Let us denote with $a_{i,\,j}$, $i,j{\in}\{0,1,2,3\}$ the generic 8-bit element of the state matrix $A[a_{i,j}]$, with $S[0,\ldots,255]$ the 256 bytes of the S-box table and with $\circ$ a $\mathbb{F}_{2^8}$ finite field multiplication (**?**). Let T_0, T_1, T_2 and T_3 be four lookup tables, each viewed as a sequence of 256 32-bit words indexed by a byte value a.

$$
\begin{aligned}
T_0[a] &= [\ S[a]\circ 02;\ S[a];\ S[a];\ S[a]\circ 03\ ]\\
T_1[a] &= [\ S[a]\circ 03;\ S[a]\circ 02;\ S[a];\ S[a]\ ]\\
T_2[a] &= [\ S[a];\ S[a]\circ 03;\ S[a]\circ 02;\ S[a]\ ]\\
T_3[a] &= [\ S[a];\ S[a];\ S[a]\circ 03;\ S[a]\circ 02\ ]
\end{aligned}
$$

These tables are used to compute the round operations as described by the following equation, where k_j is the j-th word of a given round subkey and $A_j{=}\langle a_{0,j}, a_{1,j}, a_{2,j}, a_{3,j}\rangle$ is the j-th column of the state table considered as a single 32-bit word (with the simplified notation: $A_j = A_{j \bmod 4}$, $a_{i,j} = a_{i,\,j \bmod 4}$):

$$A_j = T_0[a_{0,j}] \oplus T_1[a_{1,j-1}] \oplus T_2[a_{2,j-2}] \oplus T_3[a_{3,j-3}] \oplus k_j$$

The four tables T_0, T_1, T_2 and T_3 (called *T-tables* from now on) trade-off a larger memory requirement for a significant speedup in terms of computation time of the cipher. They use 4 KiB of storage space and their main goal is to avoid performing the MixColumn transformation as this operation, in the original definition of Rijdael algorithm, performs Galois Field multiplications by fixed constants which map poorly to general-purpose CPUs in terms of performance.

Since the T-tables may be derived also through rotating each word of T_0 by i bytes, $T_i[a] = \text{RotByte}(T_0[a], i)$, $i \in \{0,\ldots,3\}$, to reduce the active memory footprint used within each round, every column of the state table may also be computed as:

$$A_j = T_0[a_{0,j}] \oplus \text{RotByte}(T_0[a_{1,j-1}], 1) \oplus$$

$$\oplus \text{RotByte}(T_0[a_{2,j-2}], 2) \oplus \text{RotByte}(T_0[a_{3,j-3}], 3) \oplus k_j$$

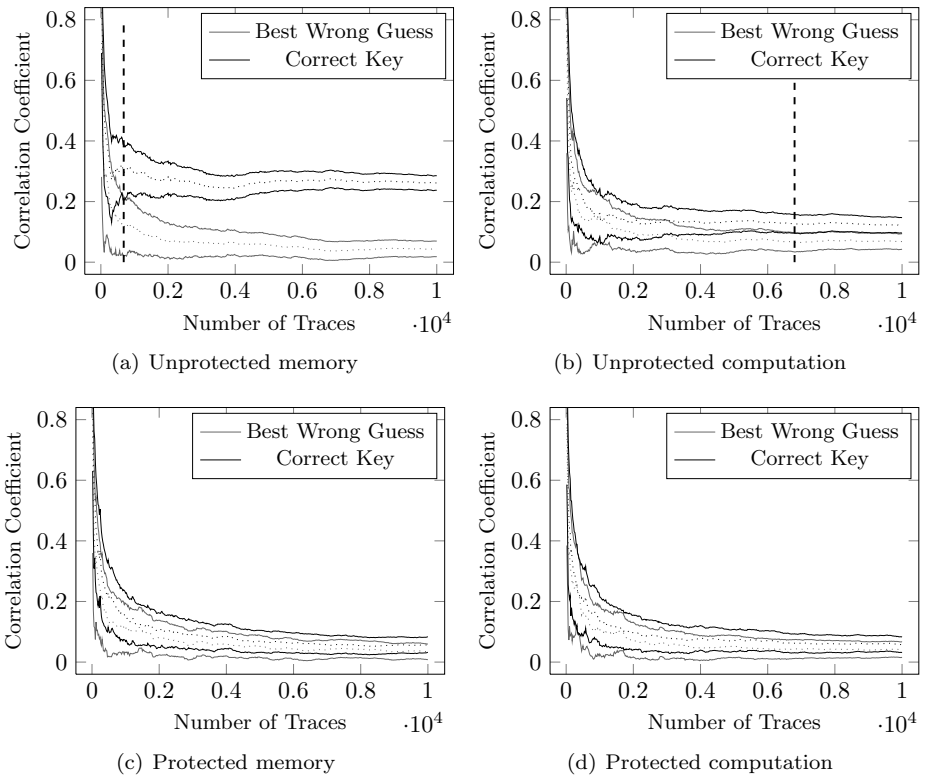
This single T-table variation reduces the lookup tables to a single 1 KiB one, while incurring a penalty of only three extra rotations per column per round with respect to the four T-tables implementation.

Finally, a further possibility to implement the SubByte step would be implementing an algorithm faster than the fully computational S-Box, without falling back to a pre-computed table. Canright investigated the definition of a very compact AES S-Box particularly suited for hardware implementations (**?**) which is also interesting for SW implementations that must be secured against side-channel attacks (like the EM-based ones) that exploits the information leakage of memory transfer operations. In a nutshell, Canright's contribution is based on a linear mapping for changing the representation base of elements in the Galois field $\mathbb{F}_{2^8} \cong \mathbb{F}_{((2^2)^2)^2}$ in such a way to perform the finite field inversion as a composition of operations in smaller (and more efficient) fields $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^2}$.

6 Experimental Results

6.1 Target Evaluation platform

The target platform is the STM32F4 Discovery board, a commercial grade development board based on the STM32F407VG Microcontroller (**?**). The microcontroller is based on the Cortex-M4 architecture from ARM and is endowed with 1 MiB Flash memory, and 128 kiB SRAM as well as a number of peripherals. During the measurements the microcontroller was clocked at the maximum frequency available (168 MHz) and the secret keys



(a) Unprotected memory

(b) Unprotected computation

(c) Protected memory

(d) Protected computation

Figure 2 Results of the attack on the 4 T-tables implementation. Figure 2(a) reports a successful attack on memory operations with 680 traces, while Figure 2(b) shows an attack to the computation of the first AddRoundKey with 6800 traces

were stored in its Flash memory, locked against readout through soft fuses. The measurements were gathered employing an Agilent DSO1012A digital oscilloscope sampling at 500 Msamples/s the voltage drop at the ends of the probe, using a vertical resolution of 5 mV/division of the 8-bit analogue to digital converter. A commodity probe was obtained out of a common 50 Ω coaxial cable, endowed with a BNC connector on one end for the connection to the oscilloscope channel. An 8-mm wide, two turns, copper wire coil was soldered on the opposite end of the coaxial cable and the coil was placed horizontally on top of the microcontroller. No amplification was needed as the measured signal consistently filled more than half of the vertical scale of the oscilloscope. Ten thousand traces for each AES implementation were acquired and stored on a support PC connected to the oscilloscope via USB. The time required to collect the traces is around two hours per batch of measurements. A triggering signal was generated internally by the microcontroller to ensure proper alignment of the traces. As a baseline reference, measurements were taken for the standard implementation of AES based on S-Box, the one present in PolarSSL (**?**), which employs four T-tables and an adaptation of it which uses only one T-table and bytewise rotations.

6.2 Experimental Evaluation

As a first result of the experimental evaluation, the evidence that the proposed scheme effectively hinders correlation attacks on the EM emissions of the chosen platform is provided. Figure 2 depicts the results of four attacks lead against the 4 T-tables implementation of the cipher, against both an unprotected (Subfigures 2(a) and 2(b)) and a protected implementation (Subfigures 2(c) and 2(d)) employing the proposed scheme (the employed masking is a 10-th order mask). The attacked operations were the first T-table look-up performed by the algorithm and a byte of the first AddRoundKey primitive respectively, considering an 8-bit key guess. The correlation metric used to determine which key guess is the correct one is Pearson's correlation coefficient, as since this is a well established method in open literature (**?**). As shown in the figures, the attacker is able to obtain a statistically sound[3] estimate of which key guess yields a correct emission model for the protected implementation, for both the T-tables lookups and the round key additions, albeit in our case the computational operation requires one order of magnitude more measures than the memory one. In both the protected cases the attacker is no longer able to obtain a statistically sound estimate as the confidence intervals for the correct key guess and the second best guess are always overlapping.

After ascertaining the effectiveness of the countermeasure, a comparison of the performances of the implementations of the AES is provided. Table 1 reports the execution times and memory footprints of both the protected and unprotected implementations. The first section of the table reports the unprotected implemen-

Table 1 Timing and memory fingerprints for the different implementations. The greyed out entry points to the best implementation for the target architecture

Implementation	Time [μs]	Tables Size [kiB]	Total Code Size [kiB]
S-box	57.2	0.25	2.45
One T-table	15.7	1	3.19
Four T-table	16.9	4	3.21
Compact S-box	708.0	0.03	4.53
Fully Computational S-box	1160.0	0	3.48
S-box compensated	64.4	0.5	2.84
S-box masked and compensated	159.0	0.5	2.85
One T-table compensated	34.3	2	4.92
One T-table masked+compensated	59.8	2	7.16
Four T-table compensated	36.2	8	5.19
Four T-table masked+compensated	59.6	8	9.00

tations, while the second reports the possible protection schemes for an S-Box based implementation. It is worth noting that suppressing completely the memory operations either by implementing the S-box through pure computation or through a partial computation over $\mathbb{F}_{2^4}$ incurs in performance penalties in the 12×-20× range and yields a net increase in the occupation of the permanent memory of the device, since the savings due to the elimination of the tabulated S-box are compensated by a higher increment in the code size, and does not provide protection against attacks on the computation. By contrast, the compensated S box implementation reports small performance penalties with respect to the unprotected one, which allow to obtain a 2.7× performance hit when employed in the proposed scheme. The third section of the table reports the results for the protected implementations employing T-tables: among these, of particular interest is the one employing the proposed protection scheme and a single T-table. This implementation has a 4% performance penalty with respect to the unprotected one based on a plain S-box (a 3.8× penalty with respect to the corresponding unprotected implementation), but it is fully protected against close range EM emissions analysis. This comes at a net increase of 4.71 kiB in the code size, which is less than 1% of the available permanent storage on our target platform. We also note that a 4 T-tables implementation does not yield any performance improvements due to a feature of the ARM architecture: ARM CPUs are endowed with a barrel shifter which is able to perform bitwise shifts and rotation on one of the operands of any arithmetical/logical instructions with no overhead. The overhead of performing the rotations on the values of the single T-table are reduced to zero, thus eliminating the need to employ 4 T-tables.

7 Related Work

Open literature on EM side channel attacks reports a number of attacks conducted on a wide range of devices. In particular, in (**??**), the authors demonstrate the viability of electromagnetic attacks (EMA) on an 8-bit processor running at 4 MHz in a smart-card, while in (**?**) the authors show that recording the emission traces over a particular spot of an FPGA programmed with an implementation of the AES block cipher. In (**?**) the authors target an even more complex device (a PDA) supporting mobile code applications running AES and elliptic curve cryptography. However, to the best of the author's knowledge, no software countermeasure scheme explicitly tailored for the EM side channel has been proposed in open literature. Nonetheless a relevant reference on masking schemes is (**?**), where the authors examine the performance and memory overheads of masking a full execution of the AES with up to 4 masks. With respect to the results presented in (**?**) we achieve a 10× reduction of the overhead in timing, and we raise the best case computational effort for the attacker from 2^{50} to 2^{80}, operations, thus rendering computationally unfeasible high-order correlation attacks. In (**??**), the authors demonstrate the viability of electromagnetic attacks (EMA) on an 8-bit processor running at 4 MHz in a smart-card, whilst in (**?**) the authors target a more complex device (a PDA) supporting mobile code applications running AES and elliptic curve cryptography.

8 Conclusions

The security and cost envelope requirements in microcontroller-based systems require efficient and cost effective countermeasures against side channel attacks. Since the power consumption side-channel can be tackled with cheap on-die solutions (**?**), the proposed countermeasure and security evaluation analysis against EM-base side-channels effectively improves the current state-

of-the-art achieving a tenfold performance improvement and an increase in the security margin by a factor of 2^{30} over previous solutions.

References

Agosta, G., Barenghi, A., and Pelosi, G. (2012). A code morphing methodology to automate power analysis countermeasures. In Groeneveld, P., Sciuto, D., and Hassoun, S., editors, *DAC*, pages 77–82. ACM.

Agrawal, D., Archambeault, B., Rao, J. R., and Rohatgi, P. (2002). The EM Side-Channel(s). In Jr., B. S. K., Çetin Kaya Koç, and Paar, C., editors, *CHES*, volume 2523 of *LNCS*, pages 29–45. Springer.

Barenghi, A., Pelosi, G., and Teglia, Y. (2010). Improving first order differential power attacks through digital signal processing. In Makarevich, O. B., Elçi, A., Orgun, M. A., Huss, S. A., Babenko, L. K., Chefranov, A. G., and Varadharajan, V., editors, *SIN*, pages 124–133. ACM.

Barenghi, A., Pelosi, G., and Teglia, Y. (2011). Information Leakage Discovery Techniques to Enhance Secure Chip Design. In Ardagna, C. A. and Zhou, J., editors, *WISTP*, volume 6633 of *LNCS*, pages 128–143. Springer.

Brainspark (2012). PolarSSL Library. `http://polarssl.org/`.

Canright, D. (2005). A Very Compact S-Box for AES. In Rao, J. R. and Sunar, B., editors, *CHES*, volume 3659 of *LNCS*, pages 441–455. Springer.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer.

Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., and Shalmani, M. T. M. (2008). On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In Wagner, D., editor, *CRYPTO*, volume 5157 of *LNCS*, pages 203–220. Springer.

Gandolfi, K., Mourtel, C., and Olivier, F. (2001). Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, Naccache, D., and Paar, C., editors, *CHES*, volume 2162 of *LNCS*, pages 251–261. Springer.

Garcia, F. D., van Rossum, P., Verdult, R., and Schreur, R. W. (2009). Wirelessly Pickpocketing a Mifare Classic Card. In *IEEE Symposium on Security and Privacy*, pages 3–15. IEEE-CS.

Gebotys, C. H. and White, B. A. (2008). EM analysis of a Wireless Java-based PDA. *ACM Trans. Embedded Comput. Syst.*, 7(4).

Hui, J. and Culler, D. (2010). IPv6 in Low-Power Wireless Networks. *Proceedings of the IEEE*, 98(11):1865 –1878.

Kocher, P. C. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Koblitz, N., editor, *CRYPTO*, volume 1109 of *LNCS*, pages 104–113. Springer.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential Power Analysis. In Wiener, M. J., editor, *CRYPTO*, volume 1666 of *LNCS*, pages 388–397. Springer.

Laohavaleeson, E. and Patel, C. (2010). Current Flattening Circuit for DPA Countermeasure. In Plusquellic, J. and Mai, K., editors, *HOST*, pages 118–123. IEEE-CS.

Mangard, S., Oswald, E., and Popp, T. (2007). *Power analysis attacks - revealing the secrets of smart cards*. Springer.

Messerges, T. S., Dabbish, E. A., and Sloan, R. H. (1999). Investigations of power analysis attacks on smartcards. In *In USENIX Workshop on Smartcard Technology*, pages 151–162. Usenix Association.

Murdoch, S. J., Drimer, S., Anderson, R. J., and Bond, M. (2010). Chip and PIN is Broken. In *IEEE Symposium on Security and Privacy*, pages 433–446. IEEE-CS.

Peeters, E., Standaert, F.-X., and Quisquater, J.-J. (2007). Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integr. VLSI J.*, 40(1):52–60.

Quisquater, J.-J. and Samyde, D. (2001). ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Attali, I. and Jensen, T. P., editors, *E-smart*, volume 2140 of *LNCS*, pages 200–210. Springer.

Réal, D., Valette, F., and Drissi, M. (2009). Enhancing Correlation Electromagnetic Attack Using Planar Near-Field Cartography. In *DATE*, pages 628–633. IEEE.

Rohatgi, P. (2009). Electromagnetic Attacks and Countermeasures. In Koç, c. K., editor, *Cryptographic Engineering*, pages 407–430. Springer US.

Schramm, K. and Paar, C. (2006). Higher Order Masking of the AES. In Pointcheval, D., editor, *CT-RSA*, volume 3860 of *LNCS*, pages 208–225. Springer.

STMicroelectronics (2012). STM32F40x family Product Data Sheet. `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/DM00037051.pdf`.

Note

[1]ISO/IEC standard 29192-2:2012 - *Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers*

[2]ETSI TS 135 202 V10.0.0 (2011-04) - *UMTS LTE - Specification of the 3GPP confidentiality and integrity algorithms - Document 2: Kasumi specification*

[3]Confidence intervals with a confidence level $\gamma = 99\%$ for the sample correlation coefficients are plotted on the figures