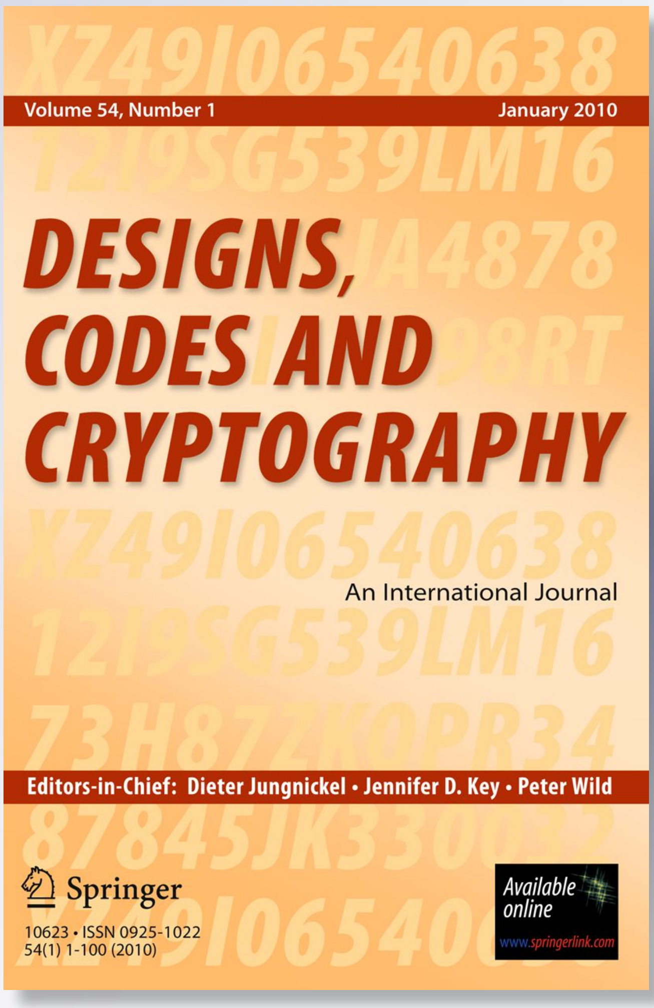
AES side-channel countermeasure using random tower field constructions

Alexis Bonnecaze, Pierre Liardet & Alexandre Venelli

Designs, Codes and Cryptography
An International Journal

ISSN 0925-1022

Des. Codes Cryptogr.
DOI 10.1007/s10623-012-9670-x



Des. Codes Cryptogr.
DOI 10.1007/s10623-012-9670-x

AES side-channel countermeasure using random tower field constructions

Alexis Bonnecaze · Pierre Liardet · Alexandre Venelli

Received: 23 January 2011 / Revised: 21 March 2012 / Accepted: 26 March 2012

Abstract Masking schemes to secure AES implementations against side-channel attacks is a topic of ongoing research. The most sensitive part of the AES is the non-linear SubBytes operation, in particular, the inversion in $GF(2^8)$, the Galois field of 2^8 elements. In hardware implementations, it is well known that the use of the tower of extensions $GF(2) \subset GF(2^2) \subset GF(2^4) \subset GF(2^8)$ leads to a more efficient inversion. We propose to use a random isomorphism instead of a fixed one. Then, we study the effect of this randomization in terms of security and efficiency. Considering the field extension $GF(2^8)/GF(2^4)$, the inverse operation leads to computation of its norm in $GF(2^4)$. Hence, in order to thwart side-channel attack, we manage to spread the values of norms over $GF(2^4)$. Combined with a technique of boolean masking in tower fields, our countermeasure strengthens resistance against first-order differential side-channel attacks.

Keywords AES · Side-channel attack · Countermeasure · Masking technique · Composite field arithmetic

Mathematics Subject Classification 94A60 · 11T71

Communicated by C. Cid.

A. Bonnecaze (✉)
Aix-Marseille University, IML, ERISCS, 13288 Marseille Cedex 09, France
e-mail: alexis.bonnecaze@univ-amu.fr

P. Liardet
Université de Provence, LATP, 13453 Marseille Cedex 13, France
e-mail: liardet@cmi.univ-mrs.fr

A. Venelli
Inside Secure, Avenue de la Victoire, Z.I. Rousset, 13790 Rousset, France
e-mail: avenelli@insidefr.com

1 Introduction

Securing cryptographic primitives on embedded devices is still a challenge today. One of the major threats in constrained environment is side-channel attacks introduced by Kocher et al. in [16]. Such attacks can be performed easily by an attacker with little knowledge about implementation details. Differential side-channel attacks exploit relationships between the processed data by the device and the side-channel leakage measured by an attacker. If we consider power consumption as the side-channel leakage, a power model can be assumed by the attacker [4,21]. Using this model, he can produce hypothetical values predicting the leakage information at several moments in time. These predictions are compared to the real power consumption of the device. The comparison is done using various statistical tests, for example the distance of means [16], the Pearson correlation factor [4] or, more recently, mutual information [13].

The advanced encryption standard (AES) is the standard for symmetric encryption [25], replacing the older data encryption standard (DES) [24]. It is used in many embedded systems and therefore its side-channel resistance has been studied in details over the years. Researchers have proposed different types of countermeasures, some more practical than others. The most general method to counter side-channel attacks is to randomize the intermediate values of the cryptographic algorithm. As the side-channel leakage is dependent on the values processed by the smart cards, the data is then de-correlated from the side-channel observations. In the case of the AES algorithm, several countermeasures have been proposed based on masking intermediate values of the AES. Most of them are concentrated on the SubBytes transformation which is the only non-linear transformation involved in the AES.

The most efficient SubBytes hardware implementation uses composite field arithmetic. Consequently, techniques introduced in [30,28,38] compute the SubBytes operation of the AES in a subfield of $GF(2^8)$. In these articles, the construction of the subfield is fixed arbitrarily whereas in [34] the authors propose to use a construction that minimizes the computation cost of composite field operations. In this work, we randomize the tower field construction (TFC) $GF(2) \subset GF(2^4) \subset GF(2^8)$ and study its impact on the side-channel resistance of the AES. When computing the inverse map in $GF(2^8)$, we have, in our case, to compute the norm in the field extension $GF(2^8)/GF(2^4)$. Hence, in order to thwart side-channel attack, the distribution of the masked norm values for a given element of $GF(2^8)$, by considering all representations in use, should spread uniformly over $GF(2^4)$. We introduce efficient methods to reach this requirement and analyze their efficiency from both the implementation and the side-channel resistance sides.

The paper is organized as follows. In Sect. 2, we give a brief description of the AES. Section 3 summarizes the major masking methods proposed for AES. Our proposition is based on a random TFC which is studied in Sect. 4. The effect of this randomness on norm values is analyzed in Sect. 5. A theoretical analysis of the security of our proposition against side-channel attacks leads to additional masking methods. In Sect. 6, we report the results of a differential power analysis attack on our propositions. We conclude this article in Sect. 7.

2 AES

We give a brief description of the AES round function, omitting the key schedule. More details can be found in [25]. The AES is defined for 128-bit blocks and key sizes 128, 192 and 256 bits. The 128-bit plaintext is viewed as a 4×4 byte matrix, called state, bytes corresponding in some way to elements of $GF(2^8)$.

The AES operates on states by iterating transformation rounds. The initial round consists in the AddRoundKey operation, the next rounds consist in applying successively the transformations SubBytes, ShiftRows, MixColumns and AddRoundKey, but the last round omits the MixColumns transformation. AddRoundKey is a bit-wise XOR operation between the state and the round key. The round keys are derived from the original key with the Key Expansion algorithm. ShiftRows is a cyclic shift operation on each of the four rows of the state. The first row is unchanged, the second is cyclically shifted by one byte to the left, the third by two bytes and the fourth by three bytes. MixColumns considers each column of the state matrix as coefficients of a degree three polynomial and multiplies them modulo $z^4 + 1$ with a fixed polynomial. SubBytes is the main building block of AES regarding the side-channel aspect. Each byte of the state matrix is replaced by its substitute in an SBox. This SBox is the composition of two transformations: an inversion in $GF(2^8)$ and an affine transformation.

3 Related work on masking methods for AES

The goal of a side-channel countermeasure is to make the power consumption of a device as independent as possible of the intermediate values of a cryptographic algorithm. Masking techniques have been extensively studied in the literature. The general principle of a masked implementation is to replace intermediate values v by some combinations $C(v, m)$ of v and a random value m. Currently, v and m are binary strings and $C(v, m) = v \oplus m$ corresponds to the bit-wise XOR addition.

Higher-order differential power analysis (HODPA) is a class of side-channel attacks proposed to counter masking methods. When classical DPA analyzes the information of one point in time of a power consumption curve, the principle of HODPA is to combine different relevant points. For example, if an attacker is able to find the point in time when the mask value r is generated by the device and the point when v' is computed, he can use these informations to retrieve the correct value v. In response, higher-order masking techniques are proposed. However effectively counteracting n-order side-channel attacks is still a difficult task. In this study, we are only concerned with first-order attacks as they are the most practical.

The only non-linear part of the AES is the inversion over $GF(2^8)$ in the SubBytes operation. Using a masking method, usually we have to compute the inverse of the input $v + r_1$ such that we obtain $v^{-1} + r_2$ with r_1, r_2 two random values. We review in the following some of the main masking schemes. We first present masking techniques that apply to generic AES software implementations. Then, we consider the methods using tower fields. These methods are particularly suitable for efficient hardware implementations.

The transform masking method (TMM) [1]. The principle is to transform a boolean mask $v + r_1$ into a multiplicative mask $v.r'$, perform the inversion and transform back into a boolean mask $v^{-1} + r_2$. Trichina et al. [37] simplify the complexity of the TMM method by considering that the masks r_1 and r_2 are equal. This method is sensitive to the zero value side-channel attack. If $v = 0$ in $GF(2^8)$ then no multiplicative mask can conceal this special value.

Embedded multiplicative masking [14]. The authors propose a solution to the zero value problem. The idea is to embed the field $GF(2^8)$ into the ring

$$R_k = GF(2)[x]/(pq) \cong GF(2^8) \times GF(2^k)$$

where p is the eighth-degree AES polynomial and q is an irreducible polynomial, co-prime to p, of degree k. Consider the random mapping:

$$\begin{aligned} \rho : GF(2^8) &\to R_k \\ v &\mapsto v + rp \bmod pq, \end{aligned}$$

where r is a randomly chosen polynomial of degree less than k. Then the value $v = 0$ in $GF(2^8)$ is mapped into 2^k possible values in R_k and should be less noticeable for an attack.

Random-value masking method [20]. Let us consider the case when a precomputed lookup Sbox table is used to compute the SubBytes operation. Messerges's method consists in re-masking lookup tables with the current mask used with the value. As the mask needs to change in order to thwart DPA, the tables are recomputed within the AES algorithm. In [15], Itoh et al. simplify the previous idea and propose to use only limited sets of fixed precomputed mask values. This countermeasure is very costly in time.

Masked modular exponentiation [3]. The authors' idea is to compute the inverse of v in $GF(2^8)$ as v^{254} using a special square-and-multiply algorithm. The authors propose the algorithms *perfectly masked squaring* and *perfectly masked multiplication* in order to obtain, at the end, the inverse masked with a boolean random value. This method is particularly costly in time.

Masking using log tables [36]. Let γ be a generator of $GF(2^8)$. Then all pairs (α, i) such that $\alpha = \gamma^i$ for $0 \leq i \leq 255$ are precomputed and stocked into two tables defined such as

$$\log(\alpha) = i \quad \text{and} \quad \operatorname{alog}(i) = \alpha.$$

Operations in $GF(2^8)$ can be implemented using the log and alog tables. In particular, the propagation of the mask in the computation of the inverse is easier. Let $v' = v + r$ be the value v masked with a random r that has to be inverted. Then with $v = \gamma^i$ and $r = \gamma^j$ one has $v'^{-1} = (\gamma^i)^{-1}(\gamma^{j-i}+1)^{-1}$. Hence, the mask after the inversion becomes $(\gamma^{j-i}+1)^{-1}$. This method needs to store log tables in memory. This can be intractable in embedded systems.

Resistant Sbox based on Fourier transform [32]. First identify the $GF(2)$-vector space $GF(2)^n$ to $GF(2^n)$ from some base and then to $\{0, \ldots 2^n - 1\}$. Now, any element X in $GF(2^n)$ can be written as a column vector $X = {}^t(x_{n-1}, \ldots, x_0)$ and also identified to the integer $\operatorname{val}(X) = \sum_{0 \leq k < n} 2^k x_k$. Finally, any map $F : GF(2^n) \to GF(2^n)$ should be identified to the integer-valued map $X \mapsto \operatorname{val}(F(X))$ still denoted by F in spite of possible confusion. The classical integer-valued scalar product $A \cdot X = \sum_{0 \leq k < n} A_k x_k$ on $GF(2^n)$ allows to identify the additive group $GF(2^n)$ to its dual and consequently the discrete Fourier transform (DFT) $\widehat{F}$ of the function F is also defined on $GF(2^n)$ by

$$\widehat{F}(A) = \sum_{X \in GF(2^n)} F(X)(-1)^{A \cdot X}.$$

Notice that $\widehat{F}$ is $\mathbb{Z}$-valued and its DFT leads to the classical inversion formula

$$F(X) = \frac{1}{2^n} \sum_{A \in GF(2^n)} \hat{F}(A)(-1)^{A \cdot X}.$$

Prouff et al. [32] observe that if the function F denotes a SBox, the above relation can be used to compute a masked SubBytes operation. However, Coron et al. in [9] show a side-channel weakness in such computation and proposes the following masked transformation that uses four random masks R_1, R_2, R_3, R_4 from $GF(2^n)$ (identified to $GF(2)^n$). Let X be a sensitive vector and let $\widetilde{X} = X \oplus R_1$ be the masked vector by R_1, then the masked DFT takes the input $\widetilde{X}$ and gives the output $F' := (-1)^{(\widetilde{X} \oplus R_2).R_1} F(X) + R_3 \bmod 2^n$ computed from the equation

$$F' = \left\lfloor \frac{1}{2^n} \left(R' + \sum_{A \in GF(2^n)} \hat{F}(A)(-1)^{(A \cdot \widetilde{X}) + (R_1 \cdot (\widetilde{X} \oplus A \oplus R_2))} \bmod 2^{2n} \right) \right\rfloor$$

where $R' = 2^n \mathrm{val}(R_3) + \mathrm{val}(R_4)$. In [17], Li et al. show that there is still a flaw in Coron's SBox algorithm due to a biased mask.

Random isomorphisms on the AES field [33]. The authors suggest the use of random representations of elements in $GF(2^8)$ as a protection against side-channel attacks. There exist $256-16$ elements in $GF(2^8)$ which are of degree two over the subfield $GF(2^4)$ and so of degree eight over $GF(2)$. Therefore, there exist 240 possible representations of the field $GF(2^8)$ to be used in AES. The principle is to randomly choose, at the beginning of the encryption, one of these representations, map the input plaintext and adapt round functions to the new representation. The output of the encryption is then mapped in the original AES field. This method requires to change AES round function for each representation and hence is costly in time.

In the following, we consider methods that combine arithmetic of subfields. This method is efficient for hardware implementation since arithmetic on such smaller fields is easily implemented in hardware.

Boolean masking in tower field [7,29,30]. The tower field

$$GF(2) \subset GF(2^2) \subset GF(2^4) \subset GF(2^8)$$

was introduced as a speed improvement for the AES [38]. Computation of the inverse is transferred to a subfield of $GF(2^8)$. Protecting this operation is also assured at the lower level. In [30], Oswald et al. proposed a masked inversion technique into $GF(2^4)$ for hardware implementations. A software version of this method is presented in [28] with $GF(2^8)$ viewed as an extension $GF(2^4)[\theta]$ of $GF(2^4)$ where θ is quadratic over $GF(2^4)$ with irreducible polynomial of the form $z^2 + z + \lambda$. The inversion operation is computed by appropriated combinations of four lookup tables whose entries depend on masked values.

We note that although Oswald et al. proved that, with their scheme, all intermediate values are masked, the study is only done at the algorithmic level. In a hardware implementation of this countermeasure Mangard et al. [18] show that it can be vulnerable to first-order side-channel attacks. The weakness is due to glitches in complementary metal oxide semiconductor (CMOS) circuits. In this paper, we do not treat this problem. Specific solutions are presented in [26,31].

4 Random TFC

First, we fix some useful notations. For a given irreducible polynomial $Q(z)$ over the field $GF(2)$ we set $GF_2(Q) := GF(2)[z]/(Q(z))$ and if $P(z)$ is an irreducible polynomial over $GF_2(Q)$ we set $GF_2(Q, P) := GF_2(Q)[z]/(P(z))$ in order to exhibit explicitly the tower construction.

4.1 The SubBytes transformation

The irreducible polynomial $R(z) = z^8 + z^4 + z^3 + z + 1$, specified in the AES, is used to create the Galois field $GF(2^8) := GF_2(R)$. This construction is referred as the standard definition of the field of 2^8 elements and we associate the basis $\Sigma := [\zeta^7, \zeta^6, \ldots, \zeta, 1]$, referred as the standard basis, where ζ is the class of z modulo $(P(z))$. Consequently, any element $x = x_7\zeta^7 + x_6\zeta^6 + \cdots + x_0$ of $GF(2^8)$ is represented in AES algorithm by the 8-bits column vector $X := {}^t[x_7, \ldots, x_0]$ (the transpose of the row vector $[x_7, \ldots, x_0]$), hence $x = \Sigma X$. Practically, x is identified to the integer $x_7 2^7 + x_6 2^6 + \cdots + x_0$. The Frobenius automorphism is intrinsically given in any field extension of $GF(2)$ by $\sigma : x \mapsto x^2$. It is represented in the standard basis by the matrix

$$S := \Sigma^{-1} \circ \sigma \circ \Sigma = \begin{bmatrix} 1&1&0&0&0&0&0&0\\ 0&0&1&0&1&0&0&0\\ 0&1&1&0&0&0&0&0\\ 1&0&0&1&0&1&0&0\\ 1&1&1&1&0&0&0&0\\ 0&0&1&0&0&0&1&0\\ 1&1&0&1&0&0&0&0\\ 0&1&0&1&0&0&0&1 \end{bmatrix}.$$

The SubBytes operation is the only non-linear round step of the cipher. It takes as an input a vector x from $GF_2(R)$ which is transformed by composing successively the following maps:

1. the multiplicative inverse in $GF_2(R)$, given $y = x^{-1}$, but fixing the inversion of $x = 0$ to $y = 0$,
2. the affine transformation

$$y \mapsto \omega(y) + \delta \tag{1}$$

defined in the standard basis by $\Omega Y + \Delta$ with:

$$\Omega = \begin{bmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{bmatrix} \quad \text{and} \quad {}^t\Delta = \begin{bmatrix} 0&1&1&0&0&0&1&1 \end{bmatrix}. \tag{2}$$

In the original description of AES the SubBytes is performed by means of a lookup table (the so-called Rijindael S-box). Here we do not use such a method of computation since it takes too much memory size for masking power consumption and so computation on-the-fly is preferred.

4.2 TFC of $GF(2^8)$ and inversion problem

Notice that calculation of the inverse in $GF_2(R)$ requires the inversion of a seventh-degree polynomial modulo a eighth-degree polynomial. This operation is very costly compared to the inversion of a first-degree polynomial modulo a second-degree one, even if the coefficients are taken from the subfield $GF(2^4)$. Consequently, the field $GF(2^8)$ is constructed using tower extensions $GF_2(Q, P)$ where Q denotes a biquadratic irreducible polynomial over $GF(2)$. Notice that there are three possible polynomials Q.

Let $P(z) := z^2 + \psi z + \lambda$ be any irreducible quadratic polynomial over the field $GF_2(Q)$. There are 120 possible such polynomials P. Now, $GF(2^8)$ is achieved as the quotient $GF_2(Q, P) = GF_2(Q)[z]/(P(z))$. As usual, each element of $GF_2(Q)$ is canonically identified in $GF_2(Q, P)$ so that $GF_2(Q)$ is viewed as a subfield of $GF_2(Q, P)$. Let α be the natural root of $P(z)$, that is to say the class of the monomial z modulo $(P(z))$. Then $[\alpha, 1]$ is a $GF_2(Q)$-basis and it is convenient to write elements of $GF_2(Q, P)$ as the sum $a\alpha + a'$. Recall that the multiplicative law (depending upon P, but not figured out) is given by the formula:

$$(a\alpha + a')\cdot(b\alpha + b') = (\psi ab + ab' + a'b)\cdot\alpha + (\lambda ab + a'b').$$

The efficiency of TFC of type $\mathbb{F}_{((2^2)^2)^2}$, in the AES case, has been already studied in detail. In [23], the authors adopt polynomials basis while Canright [6] uses normal basis. Recently, Nogami et al. [27] propose mixed bases. Our constructions lead to different basis. In fact, let θ be the natural root of $Q(z)$ in $GF_2(Q)$, that is to say θ is the class of z in $GF_2(Q) = GF(2)[z]/(Q(z))$, then, we consider the so-called natural $GF(2)$-basis

$$\Xi_{\theta,\alpha} := [\theta^3\alpha, \theta^2\alpha, \theta\alpha, \alpha, \theta^3, \theta^2, \theta, 1]$$

that sums up our construction of $GF_2(Q, P)$.

The Frobenius automorphism of the extension $GF_2(Q, P)/GF_2(Q)$ is σ^4. The conjugate over $GF_2(Q)$ of α being $\alpha + \psi$, the conjugate over $GF_2(Q)$ of $a\alpha + a'$ is $a\alpha + \psi a + a'$. The norm $N_P : GF_2(Q, P) \to GF_2(Q)$ is by definition the product of conjugates (over $GF_2(Q)$). An easy calculation gives

$$N_P(a\alpha + a') = \lambda a^2 + \psi aa' + a'^2. \tag{3}$$

Finally, the inverse of $a\alpha + a'$ can be written as

$$(a\alpha + a')^{-1} = (a\alpha + \psi a + a')N_P(a\alpha + a')^{-1}. \tag{4}$$

Once $GF_2(Q, P)$ is constructed, let

$$\mu : GF_2(Q, P) \to GF_2(R) \tag{5}$$

denote any field isomorphism. Let $y \in GF_2(R)$. Our goal is to compute y^{-1} from the composite field $GF_2(Q, P)$ by using μ, that is to compute $(\mu^{-1}(y))^{-1}$, so that the SubBytes function (Eq. 1) is given by

$$\texttt{SubBytes}(y) := \omega \circ \mu((\mu^{-1}(y))^{-1}) + \delta. \tag{6}$$

4.3 Choice of polynomials $Q(z)$ and $P(z)$

There are 3×120 possible pairs $(Q(z), P(z))$ that build $GF(2^8)$ as $GF_2(Q, P)$ but in practical applications we do not select all of them.

Choice of $Q(z)$. The biquadratic polynomial $Q(z)$ is chosen primitive. There are two such polynomials, namely $z^4 + z + 1$ and $z^4 + z^3 + 1$. We fix our choice of $Q(z)$ as the operations of multiplication, squaring and inverse in $GF(2^4)$ are generally either given by precomputed lookup tables or by hardware modules. With the above notations, by primitivity of $Q(z)$, all invertible elements of $GF_2(Q)$ are of the form θ^k with $k \in \{0, \ldots, 14\}$. Computation of multiplication, square and inversion in $GF_2(Q)$ depend on this polynomial.

Choice of $P(z)$. Irreducible polynomials $P(z) := z^2 + \psi z + \lambda$ are taken among the 64 (i.e., $\varphi(255)/2$) primitive polynomials. We will also restrict our choice to primitive polynomials with $\psi = 1$. There are exactly four such polynomials and they are conjugate by the Galois group action on λ. Since $z^2 + z + \theta^7$ is one of them, all others are $z^2 + z + \theta^{14}$, $z^2 + z + \theta^{13}$ and $z^2 + z + \theta^{11}$.

Remark 1 In [38], the authors select $Q(z) = z^4 + z + 1$ and $P(z) = z^2 + z + \theta^{11}$ where θ (in $GF_2(R)$) is a root of $Q(z)$, whereas in [34] the authors choose the same $Q(z)$ but with $P(z) = z^2 + z + \theta^{14}$. In our case, $P(z)$ is chosen randomly.

4.4 Finding isomorphisms between $GF_2(Q, P)$ and $GF_2(R)$

In this subsection we explain the way to represent any isomorphism (Eq. 5) by matrices M and in the next subsection we compute one matrix M from a concrete example.

With the above choice of $P(z)$, its root α becomes a primitive element in $GF_2(Q, P)$. Consequently, a simple way to construct an isomorphism μ (Eq. 5) is to map α to a primitive element γ of $GF_2(R)$ such that the field isomorphism holds. In order to respect the multiplicative law, we introduce the map $f : GF_2(Q, P) \to GF_2(R)$ defined by $f(\alpha^i) = \gamma^i$. This map is linear if and only if α and γ are roots of the same primitive polynomial over $GF(2)$ or, equivalently, the Zech's logarithms (also called Jacobi's logarithms) L_α and L_γ corresponding to α and γ respectively are identical. This equality means that for all $i \in \{1, \dots, 255\}$ the equality $\alpha^i + 1 = \alpha^r$ (i.e., $r = L_\alpha(i)$) is sent to the equality $\gamma^i + 1 = \gamma^r$ (i.e., $r = L_\gamma(i)$). An algorithm based on this fact is proposed in [34]. But in the case of the Galois field $GF(2^8)$, it is enough to verify only that $\alpha + 1 = \alpha^a$ ($a = L_\alpha(1)$) is sent to $\gamma + 1 = \gamma^a$ resuming the proof that f is linear by verifying only the equality

$$f(\alpha + 1) = \gamma + 1.$$

This fact is a straightforward corollary of the following theorem.

Theorem 1 *Let α and γ be two primitive elements of $GF(2^8)$ and let L_α, L_γ be their corresponding Zech's logarithms. Then α and γ are conjugate if and only if $L_\alpha(1) = L_\gamma(1)$.*

Proof The proof is obtained by computer calculation. First recall that two conjugate primitive elements give rise to the same Zech's logarithm. Now, computation of the values $L_\zeta(1)$ when ζ runs in the 16 conjugate classes of primitive elements leads to 16 distinct values. □

Remark 2 The same theorem holds if we replace $GF(2^8)$ by $GF(2^n)$ with $1 \le n \le 9$ and $n = 11, 13$.

Once μ is constructed, any field isomorphism from $GF_2(Q, P)$ to $GF_2(R)$ is of the form $\mu_k := \sigma^k \circ \mu$ $(0 \le k \le 7)$ that maps α to the conjugate $\sigma^k(\gamma)$ of γ. In fact, one has the commutation formula

$$\sigma \circ \mu = \mu \circ \sigma\,, \tag{7}$$

where σ in the right member of the equality acts in $GF_2(Q, P)$. Therefore, $\mu_k = \sigma^i \circ \mu \circ \sigma^j$ with $i + j = k$.

4.5 Matrices representing isomorphisms

By definition, the matrix

$$M_{\mu(\theta),\mu(\alpha)} = \Sigma^{-1} \circ \mu \circ \Xi_{\theta,\alpha}$$

represents $\mu : GF_2(Q, P) \rightarrow GF_2(R)$ in the above basis $\Xi_{\theta,\alpha} : GF(2)^8 \rightarrow GF_2(Q, P)$ and $\Sigma : GF(2)^8 \rightarrow GF_2(R)$. Also $S^k M_{\mu(\theta),\mu(\alpha)}$ represents $\sigma^k \circ \mu$ in the same bases.

Let τ be an automorphism of $GF_2(Q, P)$. Replacing θ by $\tau(\theta)$ and $P(z) := z^2 + \psi z + \lambda$ by $P^\tau(z) := z^2 + \tau(\psi)z + \tau(\lambda)$ leads to the basis $\Xi_{\tau(\theta),\tau(\alpha)} = \tau \circ \Xi_{\theta,\alpha}$. Therefore, due to commutation formula (Eq. 7), the matrix representing μ in the bases $(\Xi_{\tau(\theta),\tau(\alpha)}, \Sigma)$ is $S^k M_{\mu(\theta),\mu(\alpha)}$ with $\sigma^k = \tau$. We may also keep θ fixed but replace α by its conjugate α' over $GF_2(Q)$. In that case, we obtain the basis $\Xi_{\theta,\alpha'} = \sigma^4 \circ \Xi_{\theta,\alpha}$ and the matrix representing μ after this change is $S^4 M_{\mu(\theta),\mu(\alpha)}$. Combining these changes of basis, we see that all matrices M that occur in the above representations of isomorphisms μ with all possible primitive polynomials $P(z)$ (or with restriction to only primitive polynomials of the form $P_{1,\lambda}$ if necessary) are practically constructed as the following. Choose in $GF_2(R)$ a root ξ of $Q(z)$, choose a quadratic primitive polynomial $P_{\psi,\lambda}(z) = z^2 + \psi z + \lambda$ and choose in $GF_2(R)$ a root γ of $P_{\psi,\lambda}(z)$. Now build the $GF(2)$-basis

$$L_{\xi,\gamma} := [\xi^3\gamma, \xi^2\gamma, \xi\gamma, \gamma, \xi^3, \xi^2, \xi, 1]$$

of $GF_2(R)$. Therefore, the associated conversion matrix M of the basis Σ to the new basis $L_{\xi,\gamma}$ is by definition

$$M_{\xi,\gamma} := \Sigma^{-1} \circ L_{\xi,\gamma}$$

and the linear map $\mu : GF_2(Q, P) \rightarrow GF_2(R)$ that sends the basis $\Xi_{\alpha,\theta}$ onto the basis $L_{\xi,\gamma}$ (with $\mu(\alpha) = \gamma$, $\mu(\theta) = \xi$) is, by construction, an isomorphism (which is represented by $M_{\xi,\gamma}$ in the bases $(\Xi_{\theta,\alpha}, \Sigma)$).

Since we have fixed the choice of $Q(z)$, by taking all possible quadratic primitive polynomials $P_{\psi,\lambda}(z) = z^2 + \psi z + \lambda$, we get 64×8 distinct conversion matrices M_i $(1 \leq i \leq 512)$ issued from the above constructions. Restriction to polynomials $P_{1,\lambda}(z)$ reduces this number to 32, which is quite enough for our purpose as we shall see after performing side-channel simulation attacks.

4.6 A worked example

Construction of extensions $GF_2(Q, P)$ and matrices M can be summarized into three steps that we exhibit through a typical example.

Step1: initiation. The field $GF_2(R)$ of the AES is created. Choose a root ξ of $Q(z) := z^4 + z + 1$ in $GF_2(R)$. For example:

$$\xi := \zeta^6 + \zeta^4 + \zeta^3 + \zeta^2 + 1.$$

Step 2: choice of $P(z)$. Take $P(z) := z^2 + z + \theta^{11}$ and find, according to Sect. 4.4, a root γ of $P(z)$ in $GF_2(R)$. For example:

$$\gamma := \zeta^4 + \zeta^3 + \zeta^2 + \zeta + 1.$$

Step 3: computation of $M_{\xi,\gamma}$. Compute in $GF_2(R)$ each element $\xi^\nu\gamma^\varepsilon$ $(0 \leq \nu \leq 3, 0 \leq \varepsilon \leq 1)$ of the basis $L_{\xi,\gamma}$. One gets the columns $\Sigma^{-1}(\xi^\nu\gamma^\varepsilon)$ of $M_{\xi,\gamma}$. In our example

$$M_{\xi,\gamma} = \begin{bmatrix} 1&0&1&0&1&1&0&0\\ 1&1&1&0&1&1&1&0\\ 0&0&1&0&1&1&0&0\\ 0&0&1&1&0&0&1&0\\ 1&1&0&1&1&0&1&0\\ 1&0&0&1&1&0&1&0\\ 1&1&0&1&0&0&0&0\\ 0&0&1&1&1&1&1&1 \end{bmatrix}.$$

Now, taking into account the standard basis $\Xi_{\theta,\alpha}$, Eq. 6 becomes

$$\texttt{SubBytes}(y) = \Sigma Y' + \delta \tag{8}$$

with $y = \Sigma Y$ and $Y' = \Omega M_{\xi,\gamma}\Xi_{\theta,\alpha}^{-1}\big([\Xi_{\theta,\alpha}M_{\xi,\gamma}^{-1}Y]^{-1}\big)$. The term $[\Xi_{\theta,\alpha}M_{\xi,\gamma}^{-1}Y]^{-1}$ corresponds to the computation of the inverse of $\mu^{-1}(y)$ in $GF_2(Q, P)$.

5 Masking norm distribution

We analyze the distribution of the norm values occurring in Eq. 3 and its consequence on the side-channel leakage from a theoretical point of view.

5.1 Random TFC and distribution of norms

We leave out of our study the element $0 \in GF(2^8)$ as our technique provides no randomizing effect for this value. Hence this proposal needs to be applied jointly with an additive masking method.

The most sensitive part of the SubBytes using Eq. 4 is the inversion in $GF_2(Q)$ that corresponds to the inverse of the norm of elements x in $GF_2(Q, P)$. Consequently, we have to study the distribution of $N_P(\mu^{-1}(y))$ $(y \in GF_2(R))$ in all mappings μ or, equivalently, to study the distribution of $N(\tau y)$ where $N(\cdot)$ is the norm from $GF(2^8)$ to the subfield $GF(2^4)$ and τ belongs to the Galois group of $GF(2^8)$. To this aim we recall the following classical but important algebraic properties. The norm map $N(\cdot)$ is a surjective homomorphism with $\#N^{-1}(1) = 17$ and $N(y) = y^{17}$. Moreover, $N(\cdot)$ commutes with any automorphisms τ. It follows that the partition of $GF(2^8)^*$ in sets $N^{-1}(x)$ $(x \in GF(2^4)^*)$ is itself partitioned into orbits of the Galois group action on $GF(2^4)^*$, which are the so-called conjugacy classes. There are five such classes denoted $S_1, \ldots, S_5$. More precisely, let χ be any primitive element of $GF(2^4)$, then

- $S_1 = \{1\}$ and $N(y) = 1$ for 17 values y in $GF(2^8)$,
- $S_2 = \{\chi^5, \chi^{10}\}$ and $N(y) \in S_2$ for 34 values y in $GF(2^8)$,
- $S_3 = \{\chi, \chi^2, \chi^4, \chi^8\}$ and $N(y) \in S_3$ for 68 values y in $GF(2^8)$,
- $S_4 = \{\chi^3, \chi^6, \chi^9, \chi^{12}\}$ and $N(y) \in S_4$ for 68 values y in $GF(2^8)$,
- $S_5 = \{\chi^7, \chi^{11}, \chi^{13}, \chi^{14}\}$ and $N(y) \in S_5$ for 68 values y in $GF(2^8)$.

The distribution of $N(y)$ for all $y \in GF(2^8)$ and for all mappings is presented in Fig. 1.

These results imply that given an element $y \in GF(2^8)$, we can expect, at most, four different values for its norm. While the number of representations of y grows with the number of considered conversion matrices M, the bottleneck is placed in the maximal size of the

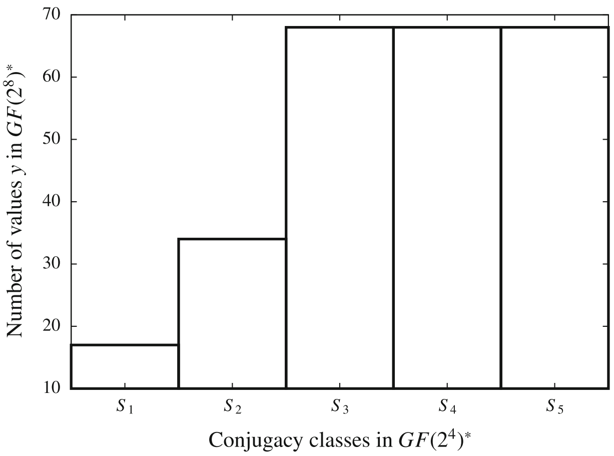


Fig. 1 The figure represents the number of elements of $GF(2^8)$ in each conjugacy class $S_1, \dots, S_5$ of values of the norm considering all possible mappings

sets S_i. Furthermore, this maximal size can be achieved with each one of the 64 polynomials $P(x)$ and, due to the fact that σ^4 is the identity on $GF(2^4)$, only four of the eight primitive elements γ give the maximal sets S_i for any given $P(x)$. These observations are visualized in Sect. 6 with the help of an experimental analysis of the side-channel leakages.

5.2 Improving the distribution of the norm

From a given y in $GF_2(R)$, our aim is to maximize the number of norm values computed from various representations of y in fields $GF_2(Q, P)$. We have just exposed that randomizing the TFC only gives, at most, four different norm values issuing from y, i.e., the norms $N_P(\mu^{-1}(y))$ belong, independently of $P(z)$ and μ, to one of the above sets S_i that has a maximal size of four elements. We would like to increase the set of possible outputs so that norm values better spread over $GF_2(Q)^*$. To this goal in mind, we propose two masking techniques and explain their respective advantages.

Method using the order of field elements

We start from the fact that $GF(2^8)^*$ is equal to the direct product of its cyclic subgroups C_3, C_5 and C_{17}, of order 3, 5 and 17 respectively. In addition, $\ker(N(\cdot)) = C_{17}$ and $GF(2^4)^*$ is the direct product of its subgroups of orders 3 and 5. The Galois group of $GF(2^8)$ acting on $GF(2^8)^*$ lets the cyclic groups invariant and the conjugacy classes are in one-to-one correspondence with the orbits of the Galois group action on the set of orders $\text{ord}(y)$ ($y \in GF(2^8)^*$). Finally, the restriction of the norm map $N(\cdot)$ on $GF(2^4)^*$ corresponds to the Frobenius automorphisms. It follows the following description of the above sets S_i in terms of orders:

- $S_1 = \{y \in GF(2^8)\,;\ \text{ord}(y) \in \{1, 17\}\}$ $(= C_{17})$,
- $S_2 = \{y \in GF(2^8)\,;\ \text{ord}(y) \in \{3, 51\}\}$,
- $S_3 = \{y \in GF(2^8)\,;\ \text{ord}(y) \in \{15, 255\}\}$,
- $S_4 = \{y \in GF(2^8)\,;\ \text{ord}(y) \in \{5, 85\}\}$,
- $S_5 = \{y \in GF(2^8)\,;\ \text{ord}(y) \in \{15, 255\}\}$.

In order to modify the norm $N_P(\mu^{-1}(y))$ to be in different sets S_i, we modify $x = \mu^{-1}(y)$ to change the order. In fact, just before the computation of the norm of x, if we knew its order, we could multiply it by an element w of an adequate order so that the resulting norm belongs to another class of conjugacy. However in practice, we do not easily get the order of x. Moreover, this method may not be adequate as the choice of w would be dependent on the value of x, hence of y, so that side-channel information could be exploited. It is then better to choose w at random. Let O_i denote the set of elements of order i in $GF(2^8)$. Let O' be the union of O_3 with a set of two elements in O_{17}, hence $|O'| = 4$. Notice that $|O_5| = 4$. Now the mask value w will be taken of the form uv with (u, v) chosen at random in $O' \times O_5$.

Our proposition for the inversion step in $GF_2(Q, P)$, with $P(z)$ fixed, consists in the following operations.

```
Inversion Masking Algorithm

  Initialization

  1. Precompute a matrix $M_0 = M_{\xi,\gamma}$, the matrices $M_k = S^k M_0$, $k = 1, 2, 3$ and their
     inverses; each matrix $M_k$ determines an isomorphism $\mu_k : GF_2(Q, P) \to GF_2(R)$
     with $\mu_k = \sigma^k \circ \mu_0$.

  2. Precompute $N_P(u)$ and $N_P(v)$ for all $u \in O'$ and $v \in O_5$.

  Randomization

  3. Randomly choose a conversion matrix $M_r$ from $\{M_0, \dots, M_3\}$.

  4. Randomly choose $u$ in $O'$ and $v$ in $O_5$.

  Input $y$ output $y^{-1}$

  5. Compute $x = \mu_k^{-1}(y) = \Xi_{\theta,\alpha} M_r^{-1} \Sigma^{-1} y = a\alpha + a'$ and compute $x' := xuv$ in
     $GF_2(R)$.

  6. Compute the norm $N_P(x')$, then its inverse $N_P(x')^{-1}$ in $GF_2(Q)$.

  7. Compute the right norm inverse $N_P(x)^{-1} = N_P(x')^{-1} N_P(u) N_P(v)$.

  8. Once the norm inverse is computed, compute the inverse $x^{-1}$ in $GF_2(Q, P)$ using
     Eq. 4.

  9. Output $y^{-1} = \Sigma M_r \Xi_{\theta,\alpha}^{-1}(x^{-1})$.
```

Remark 3 In the case of an actual AES implementation, matrices and norm values computed during the initialization phase are stored once and for all before starting any AES computation.

The choice of the sets O' and O_5 seems relevant. We compute for each $y \in GF_2(R)$ all possible norm values using our randomization technique and consider the number of distinct norms we can obtain for each given y. The results are presented in Fig. 2. We clearly see an improvement. Indeed, using only the randomization between four mappings elements of $GF_2(R)$ to $GF_2(P, Q)$ we have at most four possible norms. If we add the multiplication by elements of O' and O_5, for each $y \in GF_2(R)$, these values belong to $GF_2(Q)$ and are

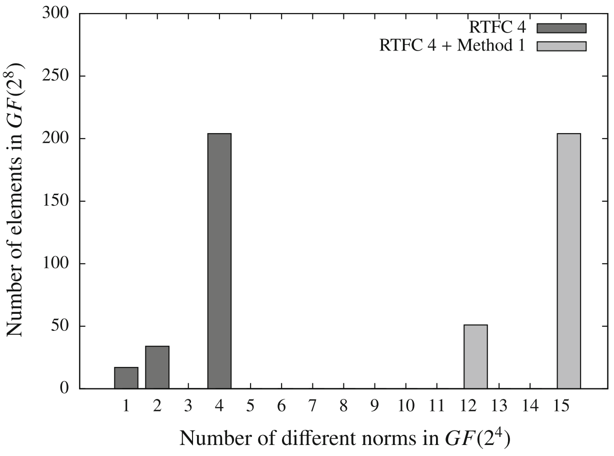


Fig. 2 Repartition of the values $N_P(\mu_k^{-1}(y))$ and $N_P(\mu_k^{-1}(y)uv)$ $y \in GF(2^8)$ for k, u and v taken from appropriated sets. We compare a random TFC with four mappings (RTFC 4) and the same construction using in addition the masking method 1 (RTFC 4 + Method 1)

given by the formula $N_P((\sigma \circ \mu_0)^{-k}(y)uv)$ with $(k, u, v) \in \{0, \dots, 3\} \times O' \times O_5$. The sets O_3, O_5 and O_{17} are invariant under the Frobenius automorphism which also commutes with N_P. Moreover $N_P(O_{17}) = \{1\}$, $N_P(O') = C_3$ and $N_P(O_5) = O_5$. These facts imply that the set of norm values we obtain for each y is given by

$$E(y) := \{\sigma^k(N_P(x))uv\,;\ (k, u, v) \in \{0, \dots, 3\} \times C_3 \times O_5\}$$

with $x = \mu_0^{-1} \circ \sigma^4(y)$. If $N_P(x) \in C_3$ (order 1 or 3) then $E(y)$ is the set $C_3.O_5$, product in $GF_2(Q)^*$ of the elements in C_3 by the elements of order 5, consequently $\#E(y) = 3 \times 4$ and going back to $GF_2(R)$ we get 3×17 elements y that give rise to 12 norm values. If $N_P(x)$ is of order 5 or 3×5 then $E(x)$ is the subgroup $C_3.C_5$ $(= GF_2(Q)^*)$. Hence $\#E(y) = 15$ and there are $(\varphi(5) + \varphi(15)) \times 17 = 204$ elements y which are involved.

A full masking method

In the above proposition, if we replace O' by C_3 and O_5 by C_5 then the norm $N_P(\mu_k^{-1}(y)uv)$ can take all the values of $GF_2(Q)^*$. This advantage is not significant with respect to our experimentation performing side-channel attacks whose results are given in the next section. In addition, during the random choices we have to prevent a possible bias due to the fact that the cardinality of C_3 and C_5 are not a power of 2.

5.3 Implementation

As previously mentioned, both our methods are based on multiplicative maskings. It is well known that this kind of maskings does not thwart zero input attacks. In order to solve this problem, one should find a conversion function that maps additive maskings into multiplicative ones. We propose to use a Dirac function to treat this issue. This solution has recently been introduced and analysed by Genelle et al. [11,12].

If we consider a classical implementation of the SubBytes using the tower field technique, the memory overhead of our method is then very small. For the two masking methods proposed we need to store four conversion matrices M_k and four corresponding inverse matrices M_k^{-1}. A matrix is stored in 8 bytes, hence those eight matrices are stored in 64 bytes. The elements of the sets O' and O_5, for the first method, and of C_3 and C_5 for the second, consist in 8 bytes. We also need to precompute $N_P(u)$ and $N_P(v)$ for $u \in O'$, $v \in O_5$ for the first method, and $u \in C_3$, $v \in C_5$ for the second. As the polynomial P is modified by the choice of a matrix M_k, we have to precompute $N_P(u)$ for the four polynomials P fixed implicitly. The first method requires to store $4 \times 4 = 16$ bytes for $N_P(u)$, $u \in O'$ and also 16 bytes for $N_P(v)$, $v \in O_5$, hence 32 bytes in total. The second method needs $3 \times 4 = 12$ bytes for $N_P(u)$, $u \in C_3$ and $5 \times 4 = 20$ bytes for $N_P(v)$, $v \in C_5$, hence also 32 bytes in total. Both methods also have a computational overhead of two multiplications in $GF(2^8)$ and two multiplications in $GF(2^4)$.

6 Experimental analysis of the countermeasure

Let K be a random variable representing a part of the secret. Let X be a random variable representing a part of the input, or output, of the cryptographic algorithm. In our context, K and X are binary strings. Suppose an attacker wants to target an intermediate value computed with the function $F(\cdot)$ that takes as parameters (X, K). Let L be a random variable representing the side-channel leakage generated by the computation of $F(X, K)$. In practice, the attacker is only able to obtain n realizations of the random variable L, denoted $V_L = (l_1, \dots, l_n)$, as he inputs n different values of X, denoted $V_X = (x_1, \dots, x_n)$. Using a distinguisher function D, he combines these two vectors plus an hypothesis on the value of the secret k'. If the distinguisher D is relevant and if the leakage vector V_L brings enough information on $F(X, K)$, then the correct value k taken by K can be recovered. In the literature, a certain amount of work has been done for creating a model for $F(X, K)$. For example, it has been proposed to consider the Hamming weight of the output of F [22], the Hamming distance [5] or simply its value [13]. The choice of a leakage model should be decided depending on the considered platform attacked. Other researches were conducted on the distinguisher function D that plays a fundamental role in the attack. Depending on its choice, the function is able to extract more or less information from the side-channel leakages. We cite some of the most used functions: the simplified T-Test proposed by Kocher et al. [16], the Pearson correlation factor [5], the mutual information [13]. It has been shown by Mangard et al. [19] and Doget et al. [10] that univariate side-channel attacks are equivalent in regard to the choice of distinguisher, given that they are provided with the same information about the leakages. We consider here the Pearson correlation factor as the distinguisher function D [5]. It is one of the most used attack on most embedded devices. The side-channel attack using this distinguisher is called correlation power analysis (CPA). The power consumption of most devices was observed to be closely linear in the Hamming weight of the processed data at a given time [5,21]. Hence the use of the Pearson factor is particularly well suited as it records linear relationships between variables.

A side-channel attack, given a vector V_L of size n, outputs a vector containing the key candidates sorted according to the test result given by D. Let $R_n = \left[k'_1, \dots, k'_{|K|}\right]$ being this sorted vector, the most likely key candidate being k'_1. A success rate [35] of order ν is the probability that the correct key is among the ν-th first candidates found by the side-channel attack. A success function of order ν can then be defined as: $S_\nu(R_n) = 1$ if $k \in \left[k'_1, \dots, k'_\nu\right]$, else $S_\nu(R_n) = 0$. The success rate of order ν of a side-channel attack $\mathscr{A}$ is then defined as

$$\mathrm{Succ}^{\nu}_{\mathscr{A}}(n) = Pr\left[S_\nu(R_n) = 1\right].$$

In practice, this probability is only estimated by performing m times a side-channel attack $\mathscr{A}$ and computing the mean. We can only consider a success rate of order 1 as a meaningful metric of the efficiency of a side-channel attack. Another convenient metric is the guessed entropy [35]. It measures, in our context, the average number of key candidates to test, after a side-channel attack has been performed, in order to find the secret key. If we keep the same notations as before, let $G(R_n)$ be the function that outputs the rank of the correct key in the sorted vector of key candidates R_n. The guessed entropy of a side-channel attack $\mathscr{A}$ is then defined as:

$$GE_{\mathscr{A}}(n) = E(G(R_n)),$$

where E denotes the expectation. As with the previous metric, in practice, it is evaluated by performing m times the side-channel attack.

We evaluate our propositions on a software AES implementation on 8-bit architecture and place the attacker in the best possible scenario. We use a simulator of power consumption to

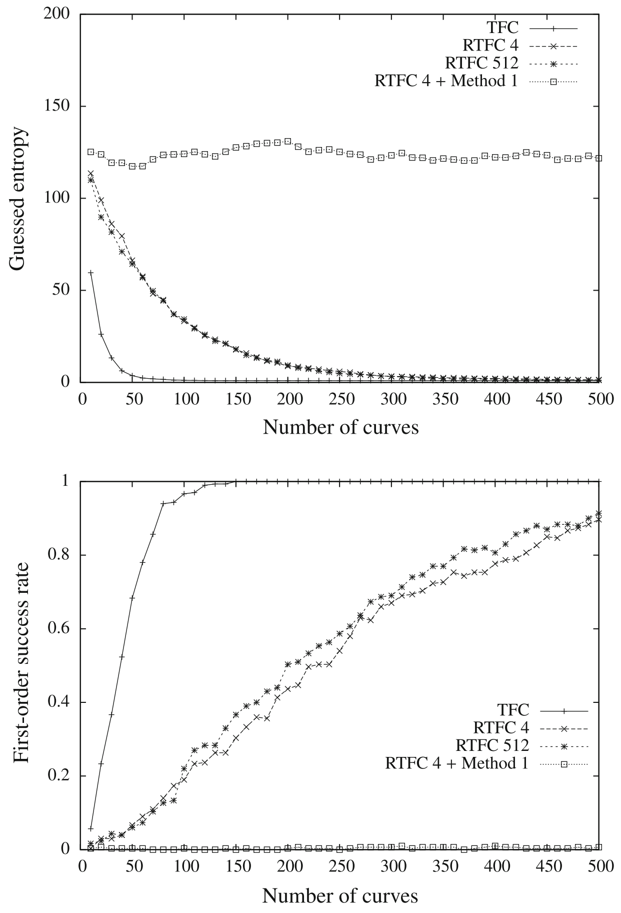


Fig. 3 Side-channel attacks results using the guessed entropy and first-order success rate metrics from simulated curves. Comparison between AES implementations: basic using a TFC, using a randomized construction with 4 mappings (RTFC 4), using a randomized construction with 512 mappings (RTFC 512) and a randomized construction with 4 mappings combined with the masking method 1 (RTFC 4 + Method 1)

obtain the power curves. This simulator outputs, at each cycle, the Hamming weight of the processed data. The power consumption is then perfectly linear in the Hamming weight of the data. The possible measurement noise is eliminated.

As previously mentioned, the sensitive part of the SubBytes operation in a TFC is the inversion, hence, the computation of $N_P(\mu_k^{-1}(y)uv)$ (see Inversion Masking Algorithm, step 6). In order to compare the effect of our propositions, we first consider a TFC with a fixed primitive polynomial $P(z)$ of the form $z^2 + z + \lambda$ and we arbitrarily fix a mapping. This is simply denoted 'TFC' in Fig. 3. Then, we choose four mappings constructed with four of the eight primitive elements in $GF(2^8)$ such that we only take elements without their conjugate over $GF_2(Q)$. This implementation is denoted 'RTFC 4' (Random TFC with four mappings). The observation of Sect. 5.1 is tested practically with an implementation using all possible mappings for a total of 512. This implementation is denoted 'RTFC 512'. Finally,

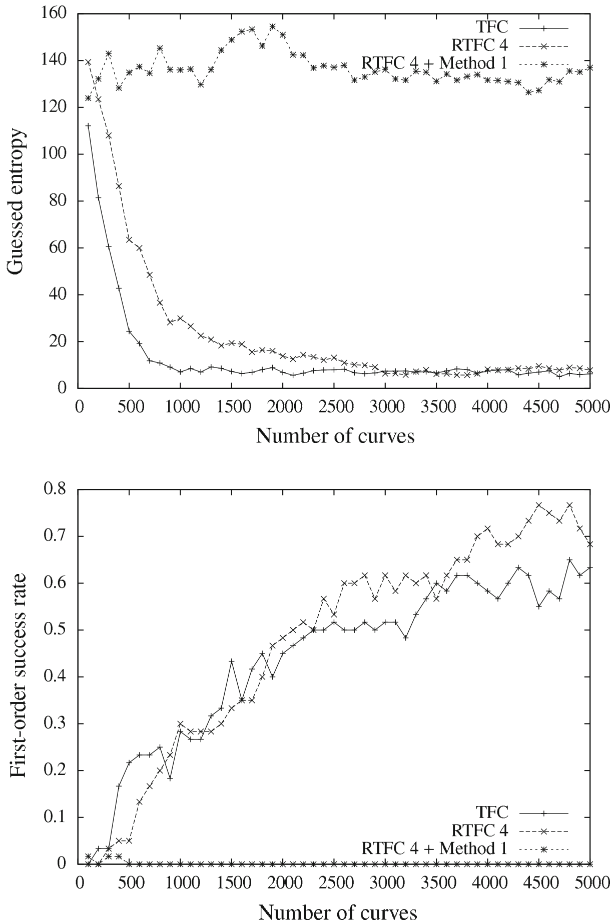


Fig. 4 Side-channel attacks results using the guessed entropy and first-order success rate metrics from implementations on an AVR 8-bit ATmega 2561 processor. Comparison between AES implementations: basic using a TFC, using a randomized construction with four mappings (RTFC 4) and a randomized construction with four mappings combined with the masking method 1 (RTFC 4 + Method 1)

we examine a random TFC combined with our first masking method (see Inversion Masking Algorithm) which is denoted 'RTFC 4 + Method 1'.

The side-channel resistance of each implementation is evaluated by applying the metrics first-order success rate and guessed entropy defined previously. In order to have metrics estimated properly, we perform $m = 20$ times each side-channel attack on $n = 500$ measurements of power consumptions. The results are presented in Fig. 3.

We first notice that the AES implementation using composite field arithmetic without masking is broken with around 50 power curves. A random TFC gives a clear improvement of the side-channel resistance. Both guessed entropy and success rate metrics indicate that an attacker would need six times more curves compared to the unprotected AES. We remark that a random construction using only four mappings is as resistant as one using every possible mappings. It confirms the properties of the norm described in Sect. 5.1. Finally, the implementation using our masking method combined with a random TFC clearly gives the best results. We note that both masking methods proposed in Sect. 5.2 give similar attack results. For a small memory and computational overhead our propositions provides a clear improvement of the side-channel resistance of a AES using TFC.

We also evaluate different AES implementations on an AVR 8-bit ATmega 2561 processor [2] running at 16 MHz. We perform $m = 4$ times each side-channel attacks on $n = 5{,}000$ power consumption measurements. In Fig. 4, we compare an AES using basic TFC with our propositions 'RTFC 4' and 'RTFC4 + Method 1'. The practical implementation confirms our simulated results and clearly shows the gain obtained from our last solution.

7 Conclusion

In this paper, we propose a masking technique for SubBytes operations in AES using a TFC. The resulting implementation of AES is particularly well suited for hardware. The SubBytes is then performed in a subfield of the original field of AES for efficiency and security reasons. This TFC can be randomly chosen in order to improve security against side channel attacks. Our study shows that the different representations of an element of the field produce at most four distinct norms. We then analyze the relation between the order of an element and its norm and extend the method so that the number of norms may be optimal for field elements. Experimental analysis shows that our method is both efficient and resistant against first-order side-channel attacks.

Acknowledgments Pierre Liardet was partially supported by CNRS (LATP-UMR 6632).

References

1. Akkar M., Giraud C.: An implementation of DES and AES, Secure against some attacks. In: CHES 2001. Lecture Notes in Computer Science, vol. 2162, pp. 309–318. Springer, Heidelberg (2001).
2. ATMEL: ATmega 2561 data sheet (2011). http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf.
3. Blömer J., Guajardo J., Krummel V.: Provably secure masking of AES. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 3357, pp. 69–83. Springer, Heidelberg (2005).
4. Brier E., Clavier C., Olivier F.: Correlation power analysis with a leakage model. In: CHES 2004. Lecture Notes in Computer Science, vol. 3156, pp. 135–152. Springer, Heidelberg (2004).
5. Brier E., Déchène I., Joye M.: Unified point addition formulæ for elliptic curve cryptosystems. In: Nedjah N., Mourelle L.M. (eds.) Embedded Cryptographic Hardware: Methodologies and Architectures, pp. 247–256. Nova Science, New York (2004).

6. Canright D.: A very compact S-box for AES. In: CHES 2005. Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer, Heidelberg (2005).
7. Canright D., Batina L.: A very compact "perfectly masked" S-Box for AES. In: ACNS 2008, pp. 446–459. Springer, Heidelberg (2008).
8. Coron J., Kizhvatov I.: Analysis of the split mask countermeasure for embedded systems. In: Proceedings of the 4th Workshop on Embedded Systems Security, Grenoble, pp. 1–10 (2009).
9. Coron J., Giraud C., Prouff E., Rivain M.: Attack and improvement of a secure S-box calculation based on the Fourier transform. In: CHES 2008. Lecture Notes in Computer Science, vol. 5154, pp. 1–14. Springer, Heidelberg (2008).
10. Doget J., Prouff E., Rivain M., Standaert F.-X.: Univariate side channel attacks and leakage modeling. J. Cryptogr. Eng. **1**(2), 123–144 (2011).
11. Genelle L., Prouff E., Quisquater M.: Secure multiplicative masking of power functions. In: ACNS 2010. Lecture Notes in Computer Science, vol. 6123, pp. 200–217. Springer, Heidelberg (2010).
12. Genelle L., Prouff E., Quisquater M.: Montgomery's trick and fast implementation of masked AES. In: AFRICACRYPT 2011. Lecture Notes in Computer Science, vol. 6737, pp. 153–169. Springer, Heidelberg (2011).
13. Gierlichs B., Batina L., Tuyls P., Preneel B.: Mutual information analysis—a generic side-channel distinguisher. In: CHES 2008. Lecture Notes in Computer Science, vol. 5154, pp. 426–442. Springer, Heidelberg (2008).
14. Golić J., Tymen C.: Multiplicative masking and power analysis of AES. In: CHES 2002, Lecture Notes in Computer Science, vol. 2535, pp. 31–47. Springer, Heidelberg (2002).
15. Itoh K., Takenaka M., Torii N.: DPA countermeasure based on the "masking method". In: ICISC 2001. Lecture Notes in Computer Science, vol. 2288, pp. 440–456. Springer, Heidelberg (2002).
16. Kocher P., Jaffe J., Jun B.: Differential power analysis. In: CRYPTO 1999. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer, Heidelberg (1999).
17. Li Y., Sakiyama K., Kawamura S., Komano Y., Ohta K.: Security evaluation of a DPA-resistant S-Box based on the Fourier transform. In: Information and Communications Security, Lecture Notes in Computer Science, vol. 5927, pp. 3–16. Springer, Heidelberg (2009).
18. Mangard S., Pramstaller N., Oswald E.: Successfully attacking masked AES hardware implementations. In: CHES 2005. Lecture Notes in Computer Science, vol. 3659, pp. 157–171. Springer, Heidelberg (2005).
19. Mangard S., Oswald E., Standaert F.-X.: One for all–all for one: Unifying standard DPA attacks. IET Information Security, Cryptology ePrint Archive, Report 2009/449 (in press) (2009).
20. Messerges T.: Securing the AES finalists against power analysis attacks. In: Fast Software Encryption. Lecture Notes in Computer Science, vol. 1978, pp. 293–301. Springer, Heidelberg (2001).
21. Messerges T.S., Dabbish E.A., Sloan R.H.: Investigations of power analysis attacks on smartcards. In: USENIX Workshop on Smartcard Technology, Chicago, pp. 151–162 (1999).
22. Messerges T.S., Dabbish E.A., Sloan R.H.: Power analysis attacks of modular exponentiation in smartcard. In: CHES 1999. Lecture Notes in Computer Science, vol. 1717, pp. 144–157. Springer, Heidelberg (1999).
23. Morioka S., Satoh A.: An optimized S-Box circuit architecture for low power AES design. In: CHES 2002. Lecture Notes in Computer Science, vol. 2523, pp. 271–295. Springer, Heidelberg (2002).
24. National Institute Standards and Technology: Data encryption standard (DES). Publication 46–2 (1993).
25. National Institute Standards and Technology: Advanced encryption standard (AES). Publication 197 (2001).
26. Nikova S., Rijmen V., Schläffer M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptol. **24**(2), 292–321 (2011).
27. Nogami Y., Nekado K., Toyota T., Hongo N.Y.M.: Mixed bases for efficient inversion in $\mathbb{F}_{((2^2)^2)^2}$ and conversion matrices of SubBytes of AES. In: CHES 2010. Lecture Notes in Computer Science, vol. 6225, pp. 234–247. Springer, Heidelberg (2010).
28. Oswald E., Schramm K.: An efficient masking scheme for AES software implementations. In: Information Security Applications. Lecture Notes in Computer Science, vol. 3786, pp. 292–305. Springer, Heidelberg (2006).
29. Oswald E., Mangard S., Pramstaller N.: Secure and efficient masking of AES-A mission impossible. Cryptology ePrint Archive, Report 2004/134 (2004).
30. Oswald E., Mangard S., Pramstaller N., Rijmen V.: A side-channel analysis resistant description of the AES S-Box. In: Fast Software Encryption. Lecture Notes in Computer Science, vol. 3557, pp. 413–423. Springer, Heidelberg (2005).

31. Prouff E., Roche T.: Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In: CHES 2011. Lecture Notes in Computer Science, vol. 6917, pp. 63–78. Springer, Heidelberg (2011).
32. Prouff E., Giraud C., Aumônier S.: Provably secure S-box implementation based on Fourier transform. In: CHES 2006. Lecture Notes in Computer Science, vol. 4249, pp. 216–230. Springer, Heidelberg (2006).
33. Rostovtsev A., Shemyakina O.: AES side channel attack protection using random isomorphisms. Cryptology ePrint Archive, Report 2005/087 (2005).
34. Rudra A., Dubey P., Jutla C., Kumar V., Rao J., Rohatgi P.: Efficient Rijndael encryption implementation with composite field arithmetic. In: CHES 2001. Lecture Notes in Computer Science, vol. 2162, pp. 171–184. Springer, Heidelberg (2001).
35. Standaert F.X., Malkin T., Yung M.: A unified framework for the analysis of side-channel key recovery attacks. In: EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer, Heidelberg (2009).
36. Trichina E., Korkishko L.: Secure and efficient AES software implementation for smart cards. In: Information Security Applications. Lecture Notes in Computer Science, vol. 3325, pp. 425–439. Springer, Heidelberg (2005).
37. Trichina, E., De Seta D., Germani L.: Simplified adaptive multiplicative masking for AES. In: CHES 2002. Lecture Notes in Computer Science, vol. 2523, pp. 71–85. Springer, Heidelberg (2003).
38. Wolkerstorfer J., Oswald E., Lamberger M.: An ASIC implementation of the AES SBoxes. In: CT-RSA 2002. Lecture Notes in Computer Science, vol. 2271, pp. 29–52. Springer, Heidelberg (2002).