Optimizations of Side-Channel Attack on AES MixColumns Using Chosen Input

Aurelien Vasselle and Antoine Wurcker

eshard, France, surname.name@eshard.com

Abstract. Considering AES sub-steps that can be attacked with a small guess space, the most practicable is to target `SubBytes` of extremal rounds. For its contrast between candidates (non-linearity) and that the search space is reduced to 2^8-sized blocks. But when such point of interests are not available, `MixColumns` may be considered but involve search spaces of 2^{32}-sized blocks. This number of attacks to run being often considered as unrealistic to reach, published papers propose to attack using chosen inputs in order to reduce back search space to 2^8-sized blocks. Several sets of chosen inputs acquisition will then be required to succeed an attack.
Our contribution consists in an optimization of usage of gained information that allows to drastically reduce the number of set needed to realize such an attack, even to only one set in some configurations.

Keywords: AES · Advanced Encryption Standard · Side-channel · SCA · `MixColumns`

1 Introduction

Side-channels research field concerns the malicious usage of an involuntary leaked information during the execution of an algorithm, with the objective to retrieve a secret such as a cryptographic key. This was first introduced in [Koc96], where the processing time of operations, that was secret-dependent, was revealing the secret information. Once the side-channels potential discovered, numerous channels where used to extract secret information, such as: power consumption [KJJ98], electromagnetic emissions [GMO01], acoustic emissions [GST14] (extension of preliminary work of 2004), light emission [FH08]. The subject is wide, as in 2014 was shown a new leak source can be used: the ground of a laptop could leak sensitive information along cables (USB, Ethernet, ...) in [GPT14].

Side-channels might not leak directly the secret itself but be an information related to the secret, e.g. leakage model linear with the Hamming weight[1] of the processed data or with the Hamming distance[2] between two successive states are commonly considered. In order to exploit those kind of leakages, statistical methods where developed that can reveal the secret information from the leakage, such as: Differential Power Analysis (DPA) [KJJ99], Correlation Power Analysis (CPA) [BCO04], Mutual Information Analysis (MIA) [GBTP08], `SCATTER` [TGWC18]. All consist in using the variations of a known data (e.g. plaintext, ciphertext) to recover a constant secret data (e.g. key). The guesses of candidates for all, or a part of, the key are ranked from the most fitting to the leakages to the least one. These attacks needs to target an intermediate value of a computation that implies a know value varying and a constant secret value that will be be guessed. In order to keep this guessing phase complexity at a realistic level, the number of point of interest in an algorithm are limited as intermediate value of algorithm are melt with a raising number of

[1]Number of bits to "1" in a binary word.
[2]Number of bits that differ between two binary words.

secret bits during algorithm process.

AES, for Advanced Encryption Standard [Nat01], is the NIST[3] symmetric cipher standard. Established in 2001 after a contest to find a successor to the DES (Data Encryption Standard), being the previous standard established in 1977 [Nat77].

In this paper we extend work of [MS16] where authors where focusing on `MixColumns` operation leakage that is not an usual considered point of interest in AES algorithm due to its initial cost of 2^{32} guesses. Authors proposes a solution using chosen plaintexts in order to reduce the guess number back to 2^8. As a counterpart it will require 16 (4 with an optimization) sets of chosen input traces to be acquired, applying this factor to the number of traces necessary to perform an attack.

This paper show how unused information acquired during this attack can be exploited to reduce by a factor of 4 the number of set needed (thus the number of traces) to only 4 sets (1 set with optimization).

This paper is organized as follows. The Section 2 introduces the previous publications on the subject of attacking `MixColumns`. The Section 3 describes our contribution on how some information, unexploited during attacks on `MixColumns`, can be used to reduce the attack complexity, considering any AES sizes. The Section 4 gives a conclusion to this paper and further work perspectives.

2 State-of-the-Art

First Round AES bytes are given in first line of Figure 1, their color depict their dependency to key bytes (expressed here in bits):

- Output of `AddRoundKey` can be targeted but it is a xor selection function, the distinguishability is not the best one.
- Output of `SubBytes` can be targeted and it is a non linear selection function so the distinguishability is good.
- Output of `MixColumns` is a not a common choice as selection function because 4 key bytes are implied in each output byte computation and it then would require to explore 2^{32} guesses that is often not considered as a viable option.

The second line of Figure 1 illustrates a generalization of the attack proposed in [MS16]. Authors suggest to choose as fixed three bytes of the four involved in a `MixColumns` matrix multiplication. This induces the fact that output of `MixColumns` are now dependent to only 8 bits of key and an unknown 8-bit constant. Finally the constant is ignored thanks to a mono-bit attack as described in details in Section 2.1. As described in Section 2.2, an optimization is also proposed, authors suggest to use the same set of chosen plaintext to target the four `MixColumns` matrix multiplication simultaneously, reducing the number of set to build.

One may remarks that the methods described below are a generalization we do from the method found in [MS16] that focused onto a specific case to solve: decryption of AES-256 with a particular leakage model.

2.1 Attack `MixColumns` Outputs

As depicted in second line of Figure 1, keeping all bytes constant except one in the plaintext would induce that 4 over the 16 `MixColumns` output bytes are then related to only one

[3]U.S. National Institute of Standards and Technology

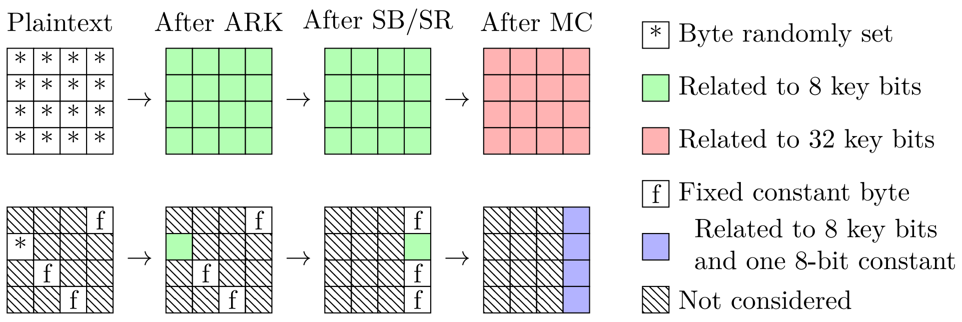

Figure 1: Illustration of attack path differences between two plaintext shape

key byte (same for all 4 bytes) and an unknown constant (different for each one of the 4 bytes). This phenomenon come from the linearity of `MixColumns` operation that consists in the matrix multiplication:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ v \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 2 \bullet f_1 \oplus 3 \bullet v \oplus 1 \bullet f_2 \oplus 1 \bullet f_3 \\ 1 \bullet f_1 \oplus 2 \bullet v \oplus 3 \bullet f_2 \oplus 1 \bullet f_3 \\ 1 \bullet f_1 \oplus 1 \bullet v \oplus 2 \bullet f_2 \oplus 3 \bullet f_3 \\ 3 \bullet f_1 \oplus 1 \bullet v \oplus 1 \bullet f_2 \oplus 2 \bullet f_3 \end{bmatrix} = \begin{bmatrix} 3 \bullet v \oplus \mathcal{F}_1 \\ 2 \bullet v \oplus \mathcal{F}_2 \\ 1 \bullet v \oplus \mathcal{F}_3 \\ 1 \bullet v \oplus \mathcal{F}_4 \end{bmatrix} \tag{1}$$

v being the variable output byte of `SubBytes` operation, we know it under a guess over one key byte. So the four blue boxes of Figure 1 are expressed here, each one being a linear combination of the same predictable variable byte v and an unknown constant $\mathcal{F}_i$. Attacking such an intermediate value by a classical way would require to guess the key byte and the constant implied resulting in a 2^{16} guesses, instead of original 2^{32}. As it still remain a lot of guesses, [MS16] proposes a methodology to reduce again the number of guesses to 2^8.

The method consists in avoiding the guess of the constant by attacking one single bit of v value[4]. The corresponding bit of constant $\mathcal{F}_i$ being, by definition, constant it will only have an impact on the sign of the correlation results. Considering absolute value of results will then remove its influence. Attacker can then recover one key byte on `MixColumns` leakage under a guess over only 2^8 candidates.

Attacker may play this attack 16 times (requiring 16 sets) to recover a full AES-128 key. In case of AES-192 (resp. AES-256) the first half of (resp. the whole) second round key have to be recovered too: The knowledge of first round key allows to build second round key chosen input and do the same attack process, requiring 8 (resp. 16) more sets.

2.2 Optimization: Parallelization

Attack described in 2.1 require to acquire 16 chosen plaintext set, but an optimization described in [MS16] consists in parallelizing 4 bytes recovery and then reduce the number of set needed to 4.

As `MixColumns` is composed of 4 independent matrix multiplication, each of them implying only 4 of input byte, we can build plaintext in order to attack the 4 instance independently and simultaneously.

The best factor of gain is 4, but some limitations could reduce this gain/thwart this optimization in case of mutual perturbation of the 4 matrix multiplication in `MixColumns`. In case of such perturbations the number of traces per set will raise in order to allow the correlation to distinguish candidates. If the growth of the trace set size is lower than a factor of 4, we still gain to parallelize, otherwise parallelization is worthless.

[4]To be understand as v, $2 \bullet v$ or $3 \bullet v$ as needed.

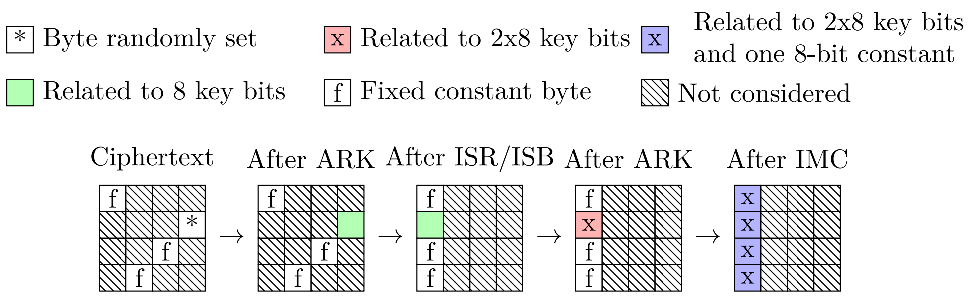

Figure 2: Illustration of key bytes and constants dependency for first rounds of AES decryption.

2.3 Attack on Decryption

[MS16] authors originally focused only on decryption of an AES-256. Even if `InvMixColumns` operation could be theoretically targeted the by the same way than `MixColumns` in encryption (by a chosen ciphertext attack), the attack of Section 2.1 cannot be applied as is, due to the first round of decryption that do not contain `InvMixColumns`.

We propose here a generalization of attack found in [MS16] that can be done by targeting the second round `InvMixColumns`. Without loss of generality we use here the numbering corresponding to the first (left most) `InvMixColumns` matrix multiplication but equations can be transposed to the other 3 matrix multiplications for the same effects. The Figure 2 shows how data is propagating.

We can see that, this time, two `AddRoundKey` operations are met before going through `InvMixColumns` of second round, inducing a dependency to 2 key bytes (one from K_{10} and one from K_9) and a constant byte. We can express equations related to the configuration depicted in Figure 2 after `InvMixColumns`:

$$\begin{bmatrix} 14 & 11 & 13 & 09 \\ 09 & 14 & 11 & 13 \\ 13 & 09 & 14 & 11 \\ 11 & 13 & 09 & 14 \end{bmatrix} \cdot \begin{bmatrix} f'_1 \\ v \\ f'_2 \\ f'_3 \end{bmatrix} = \begin{bmatrix} 14 \bullet f'_1 \oplus 11 \bullet v \oplus 13 \bullet f'_2 \oplus 09 \bullet f'_3 \\ 09 \bullet f'_1 \oplus 14 \bullet v \oplus 11 \bullet f'_2 \oplus 13 \bullet f'_3 \\ 13 \bullet f'_1 \oplus 09 \bullet v \oplus 14 \bullet f'_2 \oplus 11 \bullet f'_3 \\ 11 \bullet f'_1 \oplus 13 \bullet v \oplus 09 \bullet f'_2 \oplus 14 \bullet f'_3 \end{bmatrix} = \begin{bmatrix} 11 \bullet v \oplus \mathcal{F}'_1 \\ 14 \bullet v \oplus \mathcal{F}'_2 \\ 09 \bullet v \oplus \mathcal{F}'_3 \\ 13 \bullet v \oplus \mathcal{F}'_4 \end{bmatrix} \tag{2}$$

So variable value v is now depending on 2 key bytes. The trick consists in moving K_9 byte into constant as it is a constant and is linearly is added by xor and then consider attacking variable v' instead of v:

$$\left.\begin{array}{rcl} v & = & v' \oplus K_{9,1} \\ v' & = & \text{Sbox}^{-1}(C_{13} \oplus K_{10,13}) \end{array}\right\} \Rightarrow \begin{bmatrix} 11 \bullet v' \oplus \mathcal{F}''_1 \\ 14 \bullet v' \oplus \mathcal{F}''_2 \\ 09 \bullet v' \oplus \mathcal{F}''_3 \\ 13 \bullet v' \oplus \mathcal{F}''_4 \end{bmatrix} \text{with} \left\{\begin{array}{l} \mathcal{F}''_1 = 11 \bullet K_{9,1} \oplus \mathcal{F}'_1 \\ \mathcal{F}''_2 = 14 \bullet K_{9,1} \oplus \mathcal{F}'_2 \\ \mathcal{F}''_3 = 09 \bullet K_{9,1} \oplus \mathcal{F}'_3 \\ \mathcal{F}''_4 = 13 \bullet K_{9,1} \oplus \mathcal{F}'_4 \end{array}\right.$$

v' can now be targeted under an 8-bit guess over $K_{10,13}$, classical monobit attack process would remove the influence of the constants $\mathcal{F}''_i$ and recover $K_{10,13}$.

The process can be repeated on each of the 16 bytes of K_{10} and then reveal the last key, requiring to acquire 16 sets of chosen ciphertext.[MS16] applied the same method onto previous round in order to recover penultimate round key and thus obtain the full AES-256 main key, requiring 32 sets in total.

3 Our contribution

3.1 Attack Optimization: Recovering Constants

Our contribution consists in using an information that was dropped in state of the art attacks: the bit value of the constants $\mathcal{F}_i$ change the sign of the correlation peak, so it can be recovered.

Then an attacker could do the attack as explained in 2.1 but not limited to only one bit of v and do it on the 8 available bits, revealing the value of $\mathcal{F}_i$ thanks to sign of correlations. Doing this for v, $2 \bullet v$ and $3 \bullet v$, the attacker should be able to recover the whole set $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$ additively to key byte related to v recovered during the attack. This is realized using only one chosen plaintext acquisition set.

Once the attacker knows $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$, the following equations system can be derived from Equation (1) :

$$\begin{aligned}
\mathcal{F}_1 &= 2 \bullet f_1 \oplus 1 \bullet f_2 \oplus 1 \bullet f_3 \\
\mathcal{F}_2 &= 1 \bullet f_1 \oplus 3 \bullet f_2 \oplus 1 \bullet f_3 \\
\mathcal{F}_3 &= 1 \bullet f_1 \oplus 2 \bullet f_2 \oplus 3 \bullet f_3 \\
\mathcal{F}_4 &= 3 \bullet f_1 \oplus 1 \bullet f_2 \oplus 2 \bullet f_3
\end{aligned}$$

This is a 4 linear equation system with 3 unknown values so it can be solved (with redundancy) and gives back the set of values $\{f_1, f_2, f_3\}$. Each of them being the output of one SBox, we know that:

$$\forall i \in \{1, 2, 3\}, \exists! j \in [0, 15], f_i = \mathrm{SBox}(P_j \oplus K_{0,j}) \Rightarrow K_{0,j} = P_j \oplus \mathrm{SBox}^{-1}(f_i),$$

Recovering the set $\{f_1, f_2, f_3\}$ is then equivalent to recover the 3 key bytes implied with the one recovered thanks to v in the targeted `MixColumns` matrix multiplication.

This optimization reduce attack of Section 2.1 cost by a factor of 4 (only 4 sets are needed instead of 16). If conditions are met, it can also be combined with optimization of Section 2.2 reducing the cost by a factor 16, only 1 set is needed instead of 16.

3.2 Optimizations of Attack on Decryption

Recovering two round keys instead of one:

The resolution of equations described in Section 3.1 can also be applied with higher results, leading to the recovery of the two last round keys of simultaneously. Indeed, the 4 equations system generated by the gained knowledge of $\mathcal{F}''_i, i \in [1, 4]$ concerns 4 unknown constants (instead of 3 as seen in encrypt mode):

$$\begin{cases}
\mathcal{F}''_1 = 11 \bullet K_{9,1} \oplus 14 \bullet f'_1 \oplus 13 \bullet f'_2 \oplus 09 \bullet f'_3 \\
\mathcal{F}''_2 = 14 \bullet K_{9,1} \oplus 09 \bullet f'_1 \oplus 11 \bullet f'_2 \oplus 13 \bullet f'_3 \\
\mathcal{F}''_3 = 09 \bullet K_{9,1} \oplus 13 \bullet f'_1 \oplus 14 \bullet f'_2 \oplus 11 \bullet f'_3 \\
\mathcal{F}''_4 = 13 \bullet K_{9,1} \oplus 11 \bullet f'_1 \oplus 09 \bullet f'_2 \oplus 14 \bullet f'_3
\end{cases} \tag{3}$$

The resolution of the system (without redundancy this time) allows to recover $K_{9,1}$ additively to $K_{10,13}$. So, without taking care of f'_i knowledge, the 16 attack evoked in previous section would lead to full recovery of K_{10} and K_9 simultaneously. This allows to target AES-192 (resp. AES-256) without need to do a second 8 (resp. 16) sets attack onto next round as requested for encrypt.

Key Schedule Equations: AES-128

Furthermore, we can reduce the number of sets required by exploiting the gained f'_i knowledge. AES-128 key schedule uses 16 known equations to compute one round key from the previous one. Each equations relating three bytes from two consecutive round keys. We can then combine these 16 known equations to our gained knowledge of f'_i values that relates one byte of K_9 and one byte of K_{10}:

$$\forall i \in \{1,2,3\}, \exists!(j,\sigma()), f'_i = K_{9,\sigma(j)} \oplus \text{SBox}(P_j \oplus K_{10,j}) \tag{4}$$

Each acquisition set realized is then giving us knowledge of one byte of K_9, one byte of K_{10} and three f'_i equations, additively to the 16 equations known from key schedule. We then explored all starting byte choices and find what is the minimal number of attacked byte needed to solve the whole system of equation and find the AES-128 main key. The answer is 4 bytes, and 63 over the $\binom{16}{4} = 1820$ combinations lead to enough information to solve the equation system[5]. Any choice in the 63 valid 4-byte combination allows to gain a factor of 4 reducing the number of sets from 16 to 4.

Moreover, for 7 over the 63 choice, the 4 bytes can be parallelized (see optimization of Section 2.2) onto only 1 set to recover the whole key.

Key Schedule Equations: AES-192

AES-192 key schedule equations are different from AES-128 ones, and this time we gain 24 equations, each relating three bytes from three last round keys. Testing all combinations, we obtain that 25 occurences of 7-byte combinations are individually sufficent to solve the equation system[6] and recover the AES-192 key.

For 2 over the 25 choice, the 7 bytes can be parallelized onto only two sets to recover the key instead of 24 originally needed.

Key Schedule Equations: AES-256

AES-256 key schedule equations are different from AES-128 and AES-192 ones, and this time we gain 32 equations, each relating three bytes from three last round keys. Testing all combinations reveal that no further optimisation can be done by this mean and then 16 attack sets are required (4 in case of parallelization).

Nevertheless, the gathered equations f'_i may offer a last optimization for AES-256. When we target a first byte in a `MixColumns` matrix multiplication we obtain one byte of K_{10}, one byte of K_9 and three f'_i equations, each relating one of the three other bytes of K_{10} involved, one associated byte of K_9 and a constant byte of ciphertext (see Equation 5). Even if, as stated above, we can't obtain direct byte recovery from thoses equations, we can still select different input constants when targeting the second byte leading to a second occurrence of equations with different constants but same key byte involved:

$$\begin{aligned} \mathcal{C}^1 &= K_{9,\sigma'(j)} \oplus \text{SBox}^{-1}(C_j^1 \oplus K_{10,j}) \\ \mathcal{C}^2 &= K_{9,\sigma'(j)} \oplus \text{SBox}^{-1}(C_j^2 \oplus K_{10,j}) \end{aligned} \tag{5}$$

This system is not linear so it cannot be solved directly but, having tested all possible combinations, we state that in 98.4% of such equation system lead to only 2 solutions for the involved $(K_{9,\sigma'(j)}, K_{10,j})$ byte pair, the others lead always to 4 solutions. So after 8 attacks (2 in case of parallelization), it remains a number of AES-256 key candidate

[5] All these 63 possible solutions are listed in Table 2 in Appendix A.
[6] All these 25 possible solutions are listed in Table 3 in Appendix B.

between a minimum of 2^8 ($\sim 88\%$ of occurrence) and a maximum of $4^8 = 2^{16}$ candidates ($\sim 10^{-16}\%$ of occurrence). The AES-256 key candidates obatined can then be checked against a plaintext/ciphertext pair.

If such an exploration is not possible (e.g. no access to a plaintext/ciphertext pair), an attack onto a third byte gives a third equation for the last remaining byte and lead the equation system to be solved in more than 99.99% of configurations. So 12 sets (3 in case of parallelization) are (almost) always enough to reveal the key without need of further exploration.

3.3 Attacks Results Summary

Our unit of measure of attack complexity is the number of set of chosen plaintext needed to retrieve the key. In Table 1, we summarize how many set are needed in each configuration considered in this paper.

Table 1: Number of set needed to perform AES-128/192/256 key recovery (Bold values can be considered if a small exhaust is possible)

		Parallelization	
		Yes	No
Target Constant	Yes	1/2/2	4/8/8
	No	4/6/8	16/24/32

Encryption

		Parallelization	
		Yes	No
Target Constant	Yes	1/2/3(**2**)	4/7/12(**8**)
	No	4/6/8	16/24/32

Decryption

3.4 Additive Tricks

Reduce Uncertainty Thanks to Redundancy

One can remark that, in Section 3.1, we target twice the value v in order to recover constant $\mathcal{F}_3$ and $\mathcal{F}_4$ and then the role of each of these constant may remain unclear for attacker. Moreover, we can object that 5 over 8 bits remains equals between any byte value X and $2 \bullet X$, implying a potential confusion of constant bit recovered:

$$X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \Rightarrow 2 \bullet X = (\underline{x_6}, \underline{x_5}, \underline{x_4}, x_3 \oplus x_7, x_2 \oplus x_7, \underline{x_1}, x_0 \oplus x_7, \underline{x_7})$$

underlined bits being the ones common to X and $2 \bullet X$.

First, the temporal positionning may be clear enough to differentiate constant bit/byte role, and, if not, we propose to use the redundancy induced by the "3 unknown constants and 4 equations" system. This redundancy could indicates the proper role of each bit/byte.

For decryption mode, the probelm does not even exists as we target $09 \bullet X, 11 \bullet X, 13 \bullet X$ and $14 \bullet X$

Intermediate Result in Unproperly Masked `MixColumns` Computation

Classical 8-bit masking scheme for AES is using the same mask on every SBox output (M_o). The `MixColumns` being linear, the mask at output of this operation is:

$$3 \bullet M_o \oplus 2 \bullet M_o \oplus M_o \oplus M_o = M_o$$

So one can consider that the masking is scheme is secure and implement the multiplication as described in official AES specificiation. Doing so, an error is done as, during intermediate computation of some of the output bytes, the mask is removed and may then allow an intermediate computation result to leak, e.g. $\{2, 3, 1, 1\}$ is not a proper order of execution when $\{3, 1, 1, 2\}$ is a good one:

$$\underbrace{\underbrace{\underbrace{2 \bullet M_o \oplus 3 \bullet M_o}_{M_o} \oplus M_o}_{\varnothing} \oplus M_o}_{M_o} \qquad \underbrace{\underbrace{\underbrace{3 \bullet M_o \oplus M_o}_{2\bullet M_o} \oplus M_o}_{3\bullet M_o} \oplus 2 \bullet M_o}_{M_o}$$

Taking assumption that the intermediate bytes of computation are leaking and that the `MixColumns` operations computing order is not properly coded, we could obtain some unmasked intermediate data:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ v \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 2 \bullet f_1 \oplus 3 \bullet v \oplus 1 \bullet f_2 \\ 1 \bullet f_1 \oplus 2 \bullet v \oplus 3 \bullet f_2 \\ 1 \bullet f_1 \oplus 1 \bullet v \\ \varnothing \end{bmatrix} = \begin{bmatrix} 3 \bullet v \oplus \mathcal{F}_1 \\ 2 \bullet v \oplus \mathcal{F}_2 \\ 1 \bullet v \oplus \mathcal{F}_3 \\ \varnothing \end{bmatrix} \tag{6}$$

v can still be targeted (and the associated key byte recovered by CPA) but we can see that, contrarilly to Equation (1), the constants $\mathcal{F}_i$ do not contain each one the three elements f_1, f_2, f_3. Moreover constant $\mathcal{F}_4$ cannot be recovered anymore.

By the same methodology as in Section 3.1 we can solve this 3 equations system with 2 unknown values and then obtain f_1 and f_2 leading to the recovery of two more key bytes. A total of 3 of the 4 key bytes involved in a `MixColumns` matrix multiplication are recovered, the last byte remaining unreachable as it never appears in uncorreclty masked intermediate values.

Reproducing this on the 4 `MixColumns` matrix multiplications (parallelization could help reduce the cost once again), 12 bytes over 16 are then recovered, allowing to brute force a negligible remaining 2^{32} candidates for an AES-128 key.

4 Conclusion

In this paper we show an extension of a chosen input attack on AES `MixColumns`, using previously ignored information. This information allowing to find other key bytes by equations resolution, reducing the number of traces set to build (thus trace number required). We show that the factor of gain is around 4 times less trace set needed, in most of configurations for AES encryption. The methodology was first shown as less effective on AES decryption but we introduces some tricks allowing to reach same reduction factor.

We also show how an easy-made error in masking schemes would unmask some states only inside `MixColumns` operation, allowing an attacker to realize an first order chosen plaintext attack, that can be sped up with our methodology.

As a further work, we suggest to consider different leakage models in order to check which equations are underlying, and if resolutions can be performed to fasten the key recovery.

References

[BCO04] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES '04*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, 2004.

[CR17] Christophe Clavier and Léo Reynaud. Improved Blind Side-Channel Analysis by Exploitation of Joint Distributions of Leakages. In Wieland Fischer and

Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES '17*, volume 10529 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2017.

[CRW18] Christophe Clavier, Léo Reynaud, and Antoine Wurcker. Quadrivariate improved blind side-channel analysis on boolean masked AES. In *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, pages 153–167, 2018.

[FH08] Julie Ferrigno and Martin Hlavác. When AES blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES '08*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '01*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.

[GPT14] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES '14*, volume 8731 of *Lecture Notes in Computer Science*, pages 242–260. Springer, 2014.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2014.

[KJJ98] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Introduction to Differential Power Analysis and Related Attacks. Technical report, Cryptography Research Inc., 1998. http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[lB14] Hélène le Bouder. *Un formalisme unifiant les attaques physiques sur circuits cryptographiques et son exploitation afin de comparer et rechercher de nouvelles attaques.* PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne, 2014.

[LDL14] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the Joint Distributions of a Cryptographic Function in Side Channel Analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design – COSADE '14*, volume 8622 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2014.

[MS16] Amir Moradi and Tobias Schneider. Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, pages 71–87, 2016.

[Nat77] National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standard #46, 1977.

[Nat01] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standard #197, 2001.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 61–81, 2016.

[TGWC18] Hugues Thiebeauld, Georges Gagnerot, Antoine Wurcker, and Christophe Clavier. SCATTER: A new dimension in side-channel. In *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, pages 135–152, 2018.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 390–406, 2012.

A 63 Ciphertext Byte Choices to Optimize AES-128 Decipher Key Recovery

The table 2 shows the exhaustive list of 63 possible choices (over $\binom{16}{4} = 1820$) of 4 bytes of ciphertext to target in order to reach full recovery of AES-128 key by the mean of key schedule equations and equations obtained during attacks.

Table 2: All the 63 existing combinations of 4 ciphertext bytes to target in order to fully recover an AES-128 key

0, 5, 10, 12	2, 3, 4, 14	2, 5, 10, 12	2, 6, 11, 14	**2, 7, 11, 12**	**3, 5, 10, 14**	3, 7, 10, 13
0, 6, 10, 12	2, 4, 5, 12	2, 5, 11, 12	2, 6, 11, 15	2, 7, 11, 13	3, 5, 10, 15	3, 7, 10, 14
0, 6, 11, 14	2, 4, 6, 12	2, 5, 11, 13	2, 7, 8, 14	2, 7, 11, 14	3, 6, 8, 14	3, 7, 10, 15
0, 7, 10, 14	2, 4, 7, 14	2, 5, 11, 14	2, 7, 8, 15	2, 7, 11, 15	3, 6, 9, 14	
0, 7, 10, 15	**2, 4, 10, 12**	2, 5, 11, 15	**2, 7, 9, 14**	3, 4, 6, 14	3, 6, 10, 12	
1, 5, 10, 12	2, 4, 11, 14	2, 6, 8, 12	2, 7, 9, 15	3, 4, 10, 12	3, 6, 10, 13	
1, 6, 10, 12	2, 4, 11, 15	2, 6, 9, 12	2, 7, 10, 12	3, 4, 10, 14	3, 6, 10, 14	
1, 6, 11, 14	2, 4, 12, 14	2, 6, 10, 12	2, 7, 10, 13	**3, 4, 10, 15**	3, 6, 10, 15	
1, 7, 10, 15	2, 5, 8, 12	2, 6, 11, 12	2, 7, 10, 14	3, 5, 10, 12	3, 6, 11, 14	
1, 7, 10, 15	2, 5, 9, 12	**2, 6, 11, 13**	2, 7, 10, 15	3, 5, 10, 13	3, 7, 10, 12	

The 7 bold combinations are the only ones that can be fully parallelized, others are not because using at least two bytes in a set of 4 bytes that are related.

B 25 Ciphertext Byte Choices to Optimize AES-192 Decipher Key Recovery

The table 3 shows the exhaustive list of 25 possible choices of 7 bytes of ciphertext to target in order to reach full recovery of AES-192 key by the mean of key schedule equations and equations obtained during attacks.

Table 3: All the 25 existing combinations of 7 ciphertext bytes to target in order to fully recover an AES-192 key

0, 1, 3, 4, 5, 6, 10, 15	0, 3, 4, 6, 9, 10, 13, 15	**1, 3, 4, 6, 8, 10, 13, 15**	3, 4, 5, 6, 8, 10, 13, 15
0, 1, 3, 4, 6, 9, 10, 15	0, 3, 5, 6, 8, 10, 13, 15	1, 3, 4, 6, 9, 10, 12, 15	3, 4, 5, 6, 10, 12, 13, 15
0, 1, 3, 4, 6, 10, 13, 15	0, 3, 6, 8, 9, 10, 13, 15	1, 3, 4, 6, 10, 12, 13, 15	3, 4, 6, 8, 9, 10, 13, 15
0, 1, 3, 5, 6, 8, 10, 15	1, 3, 4, 5, 6, 8, 10, 15	1, 3, 5, 6, 8, 10, 12, 15	3, 4, 6, 9, 10, 12, 13, 15
0, 1, 3, 6, 8, 9, 10, 15	1, 3, 4, 5, 6, 10, 12, 15	1, 3, 6, 8, 9, 10, 12, 15	3, 5, 6, 8, 10, 12, 13, 15
0, 1, 3, 6, 8, 10, 13, 15	1, 3, 4, 6, 8, 9, 10, 15	1, 3, 6, 8, 10, 12, 13, 15	3, 6, 8, 9, 10, 12, 13, 15
0, 3, 4, 5, 6, 10, 13, 15			

The 2 bold combinations are the only ones that can be parallelized into 2 attack sets (instead of 7), others requiring 3 or 4 attack sets if parallelized.