Do AES encryptions act randomly?

Anna Rimoldi (rimoldi@science.unitn.it)
Department of Mathematics, Univ. of Trento, Italy.

Massimiliano Sala (maxsalacodes@gmail.com)
Department of Mathematics, Univ. of Trento, Italy.

Enrico Bertolazzi (enrico.bertolazzi@ing.unitn.it)
Department of Mechanical and Structural Engineering, Univ. of Trento, Italy.

Abstract

The Advanced Encryption Standard (AES) is widely recognized as the most important block cipher in common use nowadays. This high assurance in AES is given by its resistance to ten years of extensive cryptanalysis, that has shown no weakness, not even any deviation from the statistical behaviour expected from a random permutation. Only reduced versions of the ciphers have been broken, but they are not usually implemented. In this paper we build a distinguishing attack on the AES, exploiting the properties of a novel cipher embedding. With our attack we give some statistical evidence that the set of AES-128 encryptions acts on the message space in a way significantly different than that of the set of random permutations acting on the same space. While we feel that more computational experiments by independent third parties are needed in order to validate our statistical results, we show that the non-random behaviour is the same as we would predict using the property of our embedding. Indeed, the embedding lowers the nonlinearity of the AES rounds and therefore the AES encryptions tend, on average, to keep low the rank of low-rank matrices constructed in the large space. Our attack needs 2^{23} plaintext-ciphertext pairs and costs the equivalent of 2^{48} encryptions.

We expect our attack to work also for AES-192 and AES-256, as confirmed by preliminary experiments.

Introduction

The Advanced Encryption Standard (AES) is widely recognized as the most important block cipher in common use nowadays [Nat01]. Its 256-bit version (AES256) can even be used at top secret level ([CGC05]). This high assurance in AES is given by its resistance to ten years of extensive cryptanalysis: AES has shown no weakness, not even any deviation from the statistical behaviour expected from a random permutation. Only reduced versions of the cipher have been broken, but they are not usually implemented (see e.g. [RST10], Section 2).

For a high-security cipher it is essential that nobody can distinguish its encryption functions from random functions. It is not enough that the encryption function associated to a key cannot be distinguished from a random map (single-key attack), or that the encryption functions associated to related keys cannot be distinguished from a set of random maps. A high security cipher must behave so randomly that it must be impossible to distinguish (a random sample of) the whole set of AES encryptions from (a random sample of) the set of random permutations.

In this paper we build a special kind of distinguishing attack on the AES. To be more precise, with our attack we give some statistical evidence that the set of AES-128 encryptions acts on the message space in a way different than that of the set of random permutations acting on the same space. In this paper we do not claim any other successful distinguishing attack, neither single-key nor related-keys.

Our attack has a subtle theoretical justification. We are able to embed the AES (and actually also other translation-based ciphers) in a larger cipher, as explained in full details in [RST10]. This embedding is designed to lower the non-linearity of the AES rounds. The decrease in the non-linearity should be noted by analysing the ranks of some matrices (similarly to a Marsaglia Die-Hard test [NIS00]). While we feel that more computational experiments by independent third parties are needed in order to validate our statistical results, we show that the non-random behaviour is the same as we would predict using the property of our embedding. Indeed, we observe that the AES encryptions tend, on average, to keep low the rank of low-rank matrices constructed in the large space. This holds true apparently for all standard AES versions.

Our attack needs 2^{23} plaintext-ciphertext pairs and costs the equivalent of 2^{48} encryptions, thanks to a highly specialized rank-computation algorithm.

The remainder of this paper is organized as follows. In Section 1 we sketch the internal structure of the AES, we explain our embedding and we treat some statistical models related to statistical attacks. In Section 2 we describe our attack strategy. In Section 3 we report our attack numerical results, including results on different AES versions. In Section 4 we discuss some computational matters, presenting a rank-computation algorithm. In Section 5 we provide our conclusions and several remarks.

1 Preliminaries

In this section we mainly follow the notations and the approach in [RST10], including viewing AES as a translation-based cipher.

1.1 An AES description

In this subsection we recall the essential structure of the AES cryptosystem viewed as translation-based (for a more traditional approach see [DR02]).

Let $V = (\mathbb{F}_2)^r$ with $r = 128$ be the space of all possible messages (plaintexts or ciphertexts). Let $\mathcal{K} = (\mathbb{F}_2)^\ell$ be the finite set of all possible keys (with $\ell = 128, 192, 256$). Any key $k \in \mathcal{K}$ specifies an encryption function ϕ_k. Let $x \in V$ be any plaintext. In order to obtain the corresponding ciphertext $y = \phi_k(x) \in V$, the encryption proceeds through $N = 10, 12, 14$ similar rounds, respectively (depending on ℓ), as described below.

A preliminary translation via (addition with) the first round key $k^{(0)}$ in $(\mathbb{F}_2)^r$ is applied to the plaintext to form the input to the first round (`Round 1`). Other N rounds follow.

Let $1 \le \rho \le N-1$. A typical round (`Round` ρ) can be written as the composition[1] $\gamma\lambda\sigma_{k^{(\rho)}}$, where the map γ is called SubBytes and works in parallel to each of the 16 bytes of the data (SubBytes is composed by two transformations: the inversion in $\mathbb{F}_{2^8}$ and an affine transformation over $\mathbb{F}_2$); the linear map $\lambda : V \to V$ is the composition of two linear operations known as ShiftRows and MixColumns; $\sigma_{k^{(\rho)}}$ is the translation with the round key $k^{(\rho)}$ (this operation is called AddRoundKey).

The last round (`Round` N) is atypical and can be written as $\gamma\bar{\lambda}\sigma_{k^{(N)}}$, where the affine map $\bar{\lambda}$ is the ShiftRows operation.

Obviously, we can see the linear operation λ as a matrix $\mathbf{M}$. We observe that the order of λ is quite small and equal to 8: $\lambda^8 = 1_V$.

1.2 The embedding we are using

We are interested in particular space embeddings where the vector space $V = (\mathbb{F}_2)^r$ and W is the vector space $(\mathbb{F}_2)^s$, with $s > r$. We want to embed V into W by an injective map α and to extend $\phi_k \in \mathrm{Sym}(V)$ to a permutation $\phi'_k \in \mathrm{Sym}(W)$ as shown in the following commutative diagram:

$$\begin{tikzcd} V \arrow[r, "\alpha"] \arrow[d, "\phi_k"] \arrow[dr, phantom, "\circlearrowleft"] & W \arrow[d, "\phi'_k"] \\ V \arrow[r, "\alpha"] & W \end{tikzcd}$$

In order to do this, we have to define the permutation $\phi'_k \in \mathrm{Sym}(W)$. We say that ϕ'_k is an extension of ϕ_k. Let $r = mb$, let $s = 2^m bt$. According to the setting described in [RST10, Section 4], the space embedding $\alpha : V \to W$ we consider is defined as follows

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^{t-1}v)) \qquad (1)$$

where:

[1] Note that the order of the operation is exactly: γ, λ, and then σ_k.

a) $\varepsilon : (\mathbb{F}_{2^m})^b \to ((\mathbb{F}_2)^{2^m})^b$ is a parallel map $\varepsilon(v_1,\dots,v_b) = (\varepsilon'(v_1),\dots,\varepsilon'(v_b))$;

b) the map $\varepsilon' : \mathbb{F}_{2^m} \to (\mathbb{F}_2)^{2^m}$ is defined by means of a primitive element η of $\mathbb{F}_{2^m}$ as

$$\varepsilon'(0) = (1, \underbrace{0,\dots,0}_{2^m-1}) \qquad \varepsilon'(\eta^i) = (0,\dots,0, \underset{\substack{\uparrow \\ i+1}}{1}, 0,\dots,0) \quad \forall 1 \le i \le 2^m - 1\,.$$

c) the matrix $\mathbf{M}$ in $\mathrm{GL}((\mathbb{F}_2)^{mb})$ has order t.

Moreover, for byte-oriented Mixing Layers, i.e. if $\mathbf{M} \in \mathrm{GL}((\mathbb{F}_{2^m})^b)$, the following bound has been proved as Proposition 4.2 in [RST10]:

$$\dim_{\mathbb{F}_2}\left(\langle \mathrm{Im}(\alpha)\rangle\right) \le 2^m bt - (bt-1) - mb(t-1).$$

Let $\mathbf{M} : V \to V$ be the MixingLayer of AES. The map $\alpha : V \to W$ we propose for AES is defined as follows

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^7 v)), \tag{2}$$

where: $\varepsilon : (\mathbb{F}_2)^{128} \to (\mathbb{F}_2)^{4096}$, $\varepsilon' : \mathbb{F}_{256} \to (\mathbb{F}_2)^{256}$, $\mathbf{M} \in \mathrm{GL}((\mathbb{F}_2)^{128})$.
We have $t = 8$, $b = 16$ and $m = 8$. In Fact 4.4 ([RST10]) we determined the dimension of $\langle \mathrm{Im}(\alpha)\rangle$, for α in (2), using the fact that $\mathbf{M} \in \mathrm{GL}((\mathbb{F}_{256})^{16})$:

$$\dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\alpha)\rangle) = 2^m bt - (bt-1) - mb(t-1) = 31745.$$

The encryption ϕ_k is the composition of AddRoundKey, Subbytes and MixingLayer. So the only part of ϕ'_k which is not linear is the SubBytes operation.

Remark 1.1. The goal of our ϕ'_k construction is to have the non-linearity of SubBytes decrease.

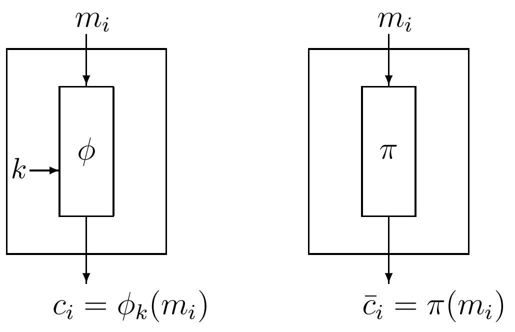
1.3 On randomness and statistical distinguishers

When a statistical test on data from a cryptographic algorithm is performed, we wish to test whether the data "seem" random or not. It is impossible to design a test that gives a decisive answer in all cases. However, there are many different properties of randomness and non-randomness, and it is possible to design tests for these specific properties. An example of a set of tests is the NIST Test Suite [NIS00]. It is a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. For example, the Marsaglia "Die Hard" test consists of determining whether the statistics of ranks of (32×32) matrices over $\mathbb{F}_2$, constructed with bits coming from the sequence, agrees with the theoretical distributions.

We are going to introduce three cryptanalitic scenarios where such test can be applied. They are called "distinguishing attacks" or "distinguishers".

Generally speaking, distinguishing attacks against block ciphers aim at determining whether a given permutation corresponds to a random permutation[2] or to a ϕ_k. Of course, there is always a distinguishing attack against any block cipher, since $|\mathcal{K}| < +\infty$, and so a brute-force key search will yield a distinguishing attack of average complexity $2^{\ell-1}$ (where ℓ is the key length), but we are interested in attacks costing significantly less.

Let $m_1, \dots, m_N$ be some plaintexts, let k be any key. We denote by π any random permutation in $\mathrm{Sym}((\mathbb{F}_2)^r)$ and by ϕ_k the encryption function for the key k; we have to consider the following situation, where one black box is involved and it contains[3] either ϕ_k or π.



A single-key distinguishing attack on a cipher $\mathcal{C}$ is any algorithm $\mathcal{A}$ able to distinguish the ciphertexts $\{c_i\}_{1\le i\le N}$ from the random ciphertexts $\{\bar{c}_i\}_{1\le i\le \mathbf{N}}$, using some information on the plaintexts. There are two main variants: the chosen-plaintext and the known-plaintext. In both, formally $\mathcal{A}$ takes as input a set of pairs $\{(m_1, \hat{c}_i), \dots, (m_N, \hat{c}_N)\}$ where

- either $\hat{c}_i = c_i \quad \forall 1 \le i \le N$,
- or $\hat{c}_i = \bar{c}_i \quad \forall 1 \le i \le N$,

and returns as output "true" or "false":

- $\mathcal{A}$ outputs "true" if and only if $\hat{c}_i = c_i$, $\forall i$ s.t. $1 \le i \le N$.
- $\mathcal{A}$ outputs "false" if and only if $\hat{c}_i = \bar{c}_i$, $\forall i$ s.t. $1 \le i \le N$.

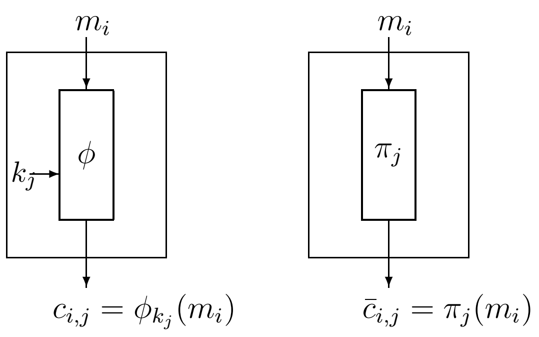
The difference between the two[4] variants:

- *Chosen-plaintext:* $\mathcal{A}$ can decide the plaintexts and obtain the corresponding ciphertexts. In this case, such plaintexts are often chosen "related", i.e. satisfying some additional prescribed mathematical relations;

[2] a permutation chosen uniformly at random from the set of all permutations.
[3] A weaker form of distinguisher assumes that the black box contain ϕ_k or π with the same probability [Luc96].
[4] There are other ways to consider the plaintexts, according to the possibilities and the capabilities of Eve.

- *Known-plaintext:* $\mathcal{A}$ cannot decide the plaintexts and we can only assume that $\mathcal{A}$ knows a certain amount of pairs (plaintext,ciphertext). In this case, the plaintexts are often supposed random.

If we have some **related** keys $k_1, \dots, k_\tau$, we can describe a second cryptanalitic scenario. A related-key distinguishing attack on a cipher $\mathcal{C}$ is any algorithm $\mathcal{A}$ able to distinguish the ciphertexts $\{c_{i,j}\}$ from the random ciphertexts $\{\bar{c}_{i,j}\}$, as in the following scheme.



Remark 1.2. In this model, $\mathcal{A}$ knows additionally some mathematical relations between the keys used for encryption, but not the key values. Both the single key scenario and the related-key one describe a hypothetical situation, very difficult to reach in practice. Yet, a very secure block cipher must resist in both scenarios.

There is another scenario where a distinguishing attack can be mounted. This scenario is less common and we have not found an established name in the literature for it, so we will call it a random-key-sample distinguisher. As in the related-key scenario we consider some keys $k_1, \dots, k_\tau$, some plaintexts $\{m_1, \dots, m_N\}$ and the corresponding ciphertexts $\{c_{i,j}\}$. The difference is that now we consider the keys as a random sample from the keyspace. This is a realistic assumption, because when the session key is changed during transmissions a new (pseudo)-random key is negiotiated between the two peers. Formally, $\mathcal{A}$ behaves in the same way as the related-key algorithm, being able to distinguish the set of actual encryptions $\{c_{i,j}\}$ from a set of random vectors $\{\bar{c}_{i,j}\}$. Clearly, also for the random-key-sample scenario we could have a chosen-plaintext variant and a known-text variant, although it is rather unlikely that a known-key attack can be succesful (we would have both random keys and random plaintexts).

Our attack in the next section is of the third type. We use a strategy similar to that of the well-known Marsaglia test.

2 Strategy description

In this section, α is our embedding (2).
We recall that $\dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\alpha)\rangle) = 31745$. Let $a^1, \dots, a^{31745}$ be (not necessarily distinct) vectors in V. Let $\mathcal{D} = \{a^1, a^2, \dots, a^{31745}\}$, so $|\mathcal{D}| \le 31745$. We construct the (31745×2^{15})-matrix $\mathbf{D}$ such that the i-th row is the image of the map α applied to the plaintext a^i, as in (3).

$$\mathbf{D} = \begin{pmatrix} \alpha(a^1) \\ \alpha(a^2) \\ \vdots \\ \alpha(a^{31745}) \end{pmatrix} = \begin{pmatrix} \varepsilon(a^1) & \varepsilon(\mathbf{M}a^1) & \cdots & \varepsilon(\mathbf{M}^7 a^1) \\ \varepsilon(a^2) & \varepsilon(\mathbf{M}a^2) & \cdots & \varepsilon(\mathbf{M}^7 a^2) \\ \vdots & \vdots & \vdots & \vdots \\ \varepsilon(a^{31745}) & \varepsilon(\mathbf{M}a^{31745}) & \cdots & \varepsilon(\mathbf{M}^7 a^{31745}) \end{pmatrix}. \tag{3}$$

Let $\mathcal{M}$ be the set of all such matrices. Clearly, we have

$$|\mathcal{M}| = (|V|)^{31745} = (2^{128})^{31745}.$$

We note that the weight of any row is $bt = 128$.

What is the rank of $\mathbf{D}$ if $\mathcal{D}$ is random in $\mathrm{Im}(\alpha)$?

The probability that a $\nu \times n$ random matrix ($\nu < n$) with entries in $\mathbb{F}_2$ has rank exactly s is significantly greater than the probability of having rank equal to $\nu - 1$ or $\nu - 2$ or less. On the other hand, for a square $n \times n$ random matrix in $\mathbb{F}_2$ the rank $n-1$ is the most probable. However, the most likely rank for $\mathbf{D}$ as in (3) is not 31745, although $31745 < 2^{15}$, because our construction imposes specific constraints, for example on the row weight. Let $d_{\mathcal{M}}$ denote the total number of matrices in $\mathcal{M}$ and let d_{31743} denote the number of all matrices in $\mathcal{M}$ with rank less than or equal to 31743. In [RST10] we have computed the expected rank statistics for $\mathbf{D}$. In particular, a direct consequence of Theorem 3.19 in [RST10] is the following corollary:

Corollary 2.1.

$$\frac{d_{31743}}{d_{\mathcal{M}}} = 0.1336357\,.$$

Our attack is a <u>random-key-sample distinguisher with chosen plaintext</u>, as detailed in the remainder of this section.

We choose 2^{16} plaintexts obtained by taking

$$\bar{S} = \{\bar{v} = (\bar{v}_1, \dots, \bar{v}_{16}) \mid \bar{v} \in (\mathbb{F}_{256})^{16},\, \bar{v}_i = 0,\, 1 \le i \le 14\}.$$

Clearly, $|\bar{S}| = (2^8)^2 = 2^{16}$.
Remark 2.2. We order $(\mathbb{F}_2)^8$ following the order of the binary representation. For example, since $(00000010) \mapsto 2$ and $(00001100) \mapsto 12$ we have

(00001100) $>$ (00000010). We order $((\mathbb{F}_2)^8)^2$ using the lexicographic ordering, induced by the previous order: $(a, b) > (a', b')$ if and only if either $a > a'$ or $a = a'$, $b > b'$. Once chosen an irreducible polynomial $p \in \mathbb{F}_2[x]$, with $\deg(p) = 8$, we can identify $\mathbb{F}_{256}$ with $(\mathbb{F}_2)^8$ and so we can use the above orderings to order both $\mathbb{F}_{256}$ and $(\mathbb{F}_{256})^2$.

Following the previous remark, we can write $\bar{S} = \{\bar{v}^1, \ldots, \bar{v}^{2^{16}}\}$ where $\bar{v}^{i+1} > \bar{v}^i$ for all i. In other words, $\bar{S}$ is an ordered set of 2^{16} vectors.

We now describe an algorithm, that we call $\mathcal{B}$, that takes in input an ordered set $S = \{v^1, \ldots, v^{2^{16}}\}$ of 2^{16} vectors in V and that outputs a list of natural numbers $r_0, \ldots, r_{31745}$ computed as follows. We construct a first matrix $\mathbf{D}$ starting from $\{v^1, \ldots, v^{31745}\}$. We compute the rank of $\mathbf{D}$ and we store the value. We repeat this procedure with $\{v^2, \ldots, v^{31746}\}$ and so on with $\{v^{k+1}, \ldots, v^{k+31745}\}$, where $2 < k \leq 33791$. In total, we compute the rank of $2^{16} - 31745 + 1 = 33792$ matrices. We define r_j as the number of these matrices with rank j (for $0 \leq j \leq 31745$).

We applied algorithm $\mathcal{B}$ to $\bar{S}$ and, since the rows of these matrices are strongly related (they share the same first 14 bytes), we expect the corresponding ranks to be significantly lower than the most probable ones (see Subsection 3.1 for details).

We can apply algorithm $\mathcal{B}$ to $\phi_k(\bar{S})$ and to $\pi(\bar{S})$, where π is any random map. We would like to use the two output lists to distinguish between ϕ_k and π, but we are not able to do it. Instead, we choose a number τ and we do two different operations. In one case, we apply $\mathcal{B}$ to $\phi_{k_i}(\bar{S})$ for τ random keys $k_1, \ldots, k_\tau$. In the other case, we apply $\mathcal{B}$ to $\pi_i(\bar{S})$ for τ random maps $\pi_1, \ldots, \pi_\tau$. In practice, we apply $\mathcal{B}$ to τ random ordered sets, each containing 2^{16} distinct vectors.
Finally, we use the output lists to distinguish between $\{\phi_{k_i}\}$ and $\{\pi_i\}$.

We expect that the behaviour of the ranks coming from the encrypted matrices is distinguishable from the theoretical distribution, and in particular that these ranks are lower. On the other hand, we expect that the ranks coming from the random matrices follow the theoretical distribution. The results are reported in Section 3.

3 Numerical results

To mount the attack successfully we need to choose τ small enough to allow for a practical computations *and* large enough to overcome the effects induced by the variance in the random distribution.

Since we computed bunches of 10 random keys (and random maps), we observed that the values coming from the random maps may be distinguishable (from the expected distribution) if we consider up to 50 maps. However, with 70 maps (or more) the random maps become undistinguishable, especially compared to the drastic values obtained by the encryptions.

Starting from a sample of 70 matrices, we report the obtained rank values corresponding to $r_0 + \ldots + r_{31743}$ and $r_{31744} + r_{31745}$:

Rank	Random	AES	Expected
> 31743	2049671	2047430	2049333
≤ 31743	315769	318010	316107

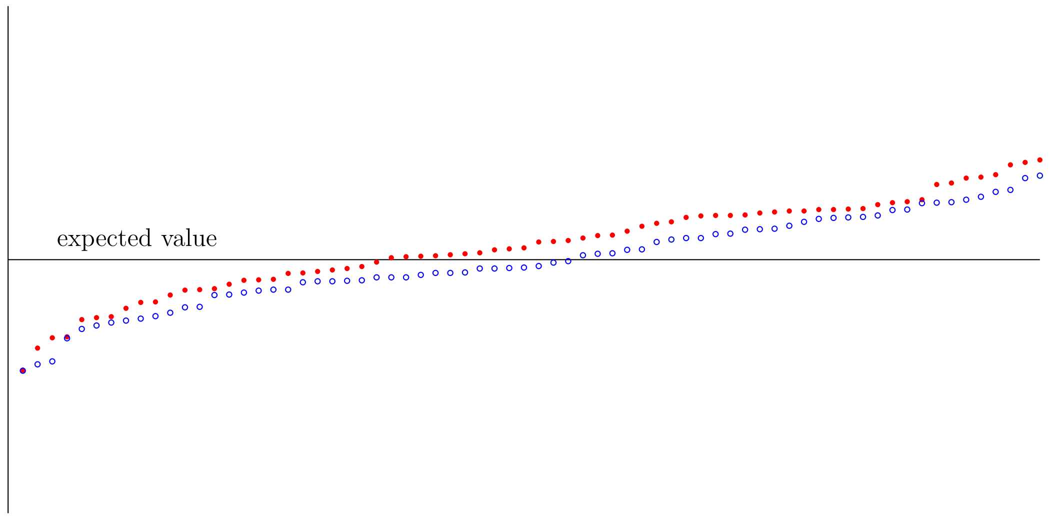
Now, we apply the χ^2 test between

(1) Random and Expected, $\rightarrow$ P value equals **0.51**;

(2) AES and Expected, $\rightarrow$ P value equals **0.0003**.

The lower the P-value, the higher the probability that the observed data do not come from the theoretical distribution. It is customary in Statistics to consider 0.05 as a threshold. Since the value for random data is 0.51 and that for AES-128 is 0.0003, we may safely assume that, with high probability, the ranks observed for AES-128 do *not* come from a random sample.
Besides, apart from the threshold, the ratio between the two P values is remarkable. And the difference between the AES-128 ranks and the theoretical distribution is exactly where we expect it to lie: in a significantly higher number of low-rank matrices.

In the following figure, we report the results of two samples coming from $\phi_k(\bar{S})$ (the 70 red dots) and from $\pi(\bar{S})$ (the 70 blue circles). First, we ordered our samples according to the number of low-rank matrices: on the left the samples with a smaller number and on the right those with a larger number. Then we plotted vertically this number. The horizontal line corresponds to the expected value for low-rank matrices. It should be apparent from the picture how the two groups of values are separated.



3.1 Furher remarks

In this subsection we report on special applications of our algorithm $\mathcal{B}$. Some results are predictable:

- since our plaintext set $\bar{S}$ contains highly correlated vectors, we would expect that $\mathcal{B}$ outputs very low ranks when the input is $\bar{S}$ itself; indeed, in this case the output is $r_{4690} = 33792$, that is, we get exactly 33792 ranks equal to 4690. Actually, it is not difficult to prove that the dimension of the vector space generated by $\alpha(\bar{S})$ is 4821, with arguments similar to those of the proof of Propostion 4.2 in [RST10]. Therefore, we would expect our 31745 vector sample to form a matrix with a lower rank $(4690 < 4821)$;
- when we apply *one* round of AES (with any key) to $\bar{S}$, algorithm $\mathcal{B}$ outputs again $r_{4690} = 33792$. This may come as a surprise, but it is easily explained in our framework. One round[5] means, in order, one key addition, one S-Box, one λ and another key addition. Thanks to properties of the embedding α, all the above operations are linear, except for the S-Box (see Proposition 4.5 in [RST10]). However, the S-Box in this case does not change the type of subspace. Indeed, after the fist key-addition we have all vectors sharing the first 14 coordinates and the last two are free to be any pair. Since the S-Box acts in parallel, it does not change this situation and so the dimension of the whole space and the ranks of our matrix remain unchanged;
- things change when we apply *two* rounds of AES; the reason is that the MixColumns changes the structure, since it intermixes four bytes at a time; it is true that the MixColumns in the first round does not change the rank, but the change in the structure is fatal to the rank when the S-Box of the *second* round is applied; indeed, our experiments shows that $\mathcal{B}$ outputs in this case $r_{20548} = 33792$. Again, this lower value is justified by the dimension of the 2-round encryption of $\bar{S}$, which is 20679;
- similarly, the rank raises with the application of *three* rounds: $\mathcal{B}$ outputs in this case $r_{31661} = 33792$ and the dimension of the 3-round encryption of $\bar{S}$ is 31681;
- when we apply *four* rounds or more, we get values near to the random setting (and the dimension of the subspace is 31745, since it coincides with the whole space $\langle Im(\alpha)\rangle$); in this sense, we could say that the diffusion of AES is working from 4 rounds and above, as it is usually agreed.

[5] In the translation-based notation we are performing `Round` 0 and `Round` 1

Some results are largely unexpected. For example:

- let us consider two random maps $\pi_1(\bar{S})$, $\pi_2(\bar{S})$ and two encryption functions $\phi_{k1}(\bar{S})$ and $\phi_{k2}(\bar{S})$ ($k_1 \neq k_2$). Now, we apply algorithm $\mathcal{B}$ to the four corresponding sets and we obtain the following rank distributions.

Rank	Random$_1$	Random$_2$	AES$_1$	AES$_2$	Expected
31745	9782	9467	9765	9554	9759
31744	19482	19765	19525	19569	19517
≤ 31743	4528	4560	4502	4669	4516

  As before, we apply the χ^2 test between

  (1) Random$_1$ and Expected, $\rightarrow$ P value equals 0.928;
  (2) Random$_2$ and Expected, $\rightarrow$ P value equals 0.002;
  (3) AES$_1$ and Expected, $\rightarrow$ P value equals 0.97;
  (4) AES$_2$ and Expected, $\rightarrow$ P value equals 0.008.

  We note that case (1) and case (3) are statistically indistinguishable from the expected distribution, while case (2) and (4) appear statistically distinguishable from the expected. In other words, if we apply our test *only to one key*[6], it fails badly, because it would distinguish with probability 0.5, that is, it outputs a random value! The reason for the single-key failure lies in the large variance among our samples. This is why, in order to overcome this problem, we had to consider more keys: the right τ, large enough to highlight the statistical differences and still small enough to compute efficiently the corresponding ranks.

- When we mount our attack we have the freedom to consider for the χ^2 test whatever combination of the ranks r_j we like, as long as random samples are not distinguishable and AES samples are. We tuned our test to consider only two values (ranks lower than 31743 and those higher). Two other choices are possible.
  One would be to look only at even smaller ranks, such as "ranks lower than 31740" (and those higher). We have discarded this option because smaller ranks are very infrequent and we would then need a very large sample in order to validate our tests.
  The second choice is to consider more ranks, for example three ranks, as in the following table. We considered a total sample having 70 elements

[6] that is, if we try to mount a single-key attack.

Rank	Random	AES	Expected
31745	684191	682317	683111
31744	1365480	1365113	1366222
≤ 31743	315769	318010	316107

According to the χ^2 test we have that:

(1) Random and Expected, $\rightarrow$ P value equals **0.29**;
(2) AES and Expected, $\rightarrow$ P value equals **0.0013**.

So again we would distinguish between random and AES-128, but with more difficulty. This can actually be explained a posteriori: it is true that our embedding would induce less maximum-rank matrices (and the numbers confirm this: $682317 < 683111$), but they might become 31744-rank matrices and so add to the most noisy value[7] , and indeed the 31744-rank matrices in the AES-128 sample are only slighty less than the expected.

We did not report on experiments on the other standard versions of AES (AES-256 and AES-192), but our preliminary tests seem to indicate that our test works well also in those cases, with only a slight worsening of the P value ratio. Indeed, our strategy is independent from the key-length of AES, since our approach is actually independent from the key-schedule and only marginally dependent on the round numbers.

4 Computational effort

The algorithm developed to compute the ranks for the attacks is specialized for $\mathbb{F}_2$ and is described in [BR10]. Here we provide a sketch.

Since the matrix is rather large (circa $2^{15} \times 2^{15}$), we must keep the heaviest part of the computation within the cache. However, some steps on the whole matrix cannot be avoided, so we need a strategy that keeps these to a minimum. In particular, we may need both column and row permutations. In the below description, we assume that we do not need them.

Remark 4.1. To be able to avoid permutations is equivalent to having performed a preprocessing such that each upper-left square block is square and non-singular. Of course this cannot be done a priori, but we stick to this for clarity's sake.

Without the technicalities pertaing permutations, our algorithm reduces to a variant of a recursive LU decomposition ([GV96]). Let the matrix M be in $(\mathbb{F}_2)^{R\times n}$. If the whole M does not fit into the cache, we partition M into four blocks of approximately the same size.

[7] random variable r_{31744} has the largest variance.

Let M_1 be the left-top block. If it does not fit, then we partition M_1 similarly in four blocks and so on. Let A be the smallest block that does not fit. We will have

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}, \tag{4}$$

where A_{11} fits into the cache. Thanks to Remark 4.1, we can assume that block $\boldsymbol{A}_{11}$ is square and non-singular. We can build the LU decomposition of $\boldsymbol{A}_{11} = \boldsymbol{L}_{11}\boldsymbol{U}_{11}$ and consider the following equality

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{U}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{A}}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{U}_{11} & \boldsymbol{L}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tag{5}$$

where $\tilde{\boldsymbol{A}}_{22} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{A}_{12}$ is the Schur complement. Even if block $\tilde{\boldsymbol{A}}_{22}$ is singular we can compute the LU decomposition $\tilde{\boldsymbol{A}}_{22} = \boldsymbol{L}_{22}\boldsymbol{U}_{22}$ and the final LU decomposition of matrix $\boldsymbol{A}$, as follows:

$$\begin{aligned} \boldsymbol{A} &= \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{U}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{L}}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{U}}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{U}_{11} & \boldsymbol{L}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} = \\ &= \underbrace{\begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{U}_{11}^{-1} & \tilde{\boldsymbol{L}}_{22} \end{pmatrix}}_{\boldsymbol{L}} \underbrace{\begin{pmatrix} \boldsymbol{U}_{11} & \boldsymbol{L}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \tilde{\boldsymbol{U}}_{22} \end{pmatrix}}_{\boldsymbol{U}}. \end{aligned}$$

Once we have the LU decomposition of A, we use it to recursively compute the LU decomposition of larger blocks containing A, until we reach a global LU decomposition for the whole M. From it, the rank determination is trivial, because it is enough to count the nonzeros in the diagonal of L.

The three most expensive steps in the above algorithm are:

(1) the LU decomposition of block $\boldsymbol{A}_{11}$.

(2) the LU decomposition of block $\tilde{\boldsymbol{A}}_{22}$.

(3) the computation of the Schur complement $\tilde{\boldsymbol{A}}_{22}$.

All three steps cost at most $O(n^3)$, with standard matrix multiplication, however the actual constants differ and depend on the matrix structure and sparsity. The cost of matrix multiplication can be lowered theoretically with the Strassen method ([Str69]), but our matrices are too small to take advantage of it. However, they are large enough to entice the use of the famous *four Russian algorithm* ([Ang76]). We refer to [BR10] for our exact strategy.

4.1 Cost of the attack

The attack works very well with 128 keys, although 70 are usually enough. We draw a very conservative estimate of its cost in this subsection.

The attacker needs to collect 2^{16} pairs encrypted with the same key. Since we are requiring 128 keys, it means that the total number of pairs is

$$2^{16} \cdot 2^7 = \mathbf{2^{23}}\,.$$

For any key, the attacker must compute about 2^{15} matrix ranks. Any rank computation costs[8] 2^{26} ecnryptions. Therefore, the total cost of the attack is

$$2^{26} \cdot 2^{15} \cdot 2^7 = \mathbf{2^{48}}\,\text{encryptions}$$

5 Conclusions

Reduced-round versions of AES-128, AES-192, AES-256 are known to be weak, although none of these attacks come close to the actual number of rounds. The best-known attacks use advanced differential cryptanalysis and depend heavily on the key-scheduling algorithm. Our distinguishing attack is independent of the key-schedule and depends only on the round structure. Therefore, it may be successful even if a huge number of rounds is used.
We strongly invite anyone to try our attack, with any number of rounds, and we put our software freely usable at

http://www.science.unitn.it/˜ sala/AES/

The more statistical evidence we collect, the more confidence we will grow in our results. Of course, it is possible that a refinement of our approach might lead to a key-recovery algorithm. Yet, we have not been able to see how, since the link between the key and the rank statistics is still unclear.

Acknowledgments

A preliminary attack with the same underlying philosophy is contained in the first author's Ph.D thesis. The first author would like to thank the second author (her supervisor). The computational part contributed by the third author (with very fast implementations) has been essential to the success of this attack.

We thank G. Naldi and G. Aletti for the use of the computer cluster Ulisse (University of Milan).

These results have been presented in a few talks (2009: Trento; 2010: Marseille, Torino) and several scientific discussions with colleagues. The authors would like to thank all the people involved in our discussions, for their

[8] using our algorithm, this is an experimental estimate.

valuable comments and suggestions. In particular, we would like to thank T. Mora, L. Perret and C. Traverso. Some discussions took place during the Special Semester on Groebner Bases (2006), organized by RICAM, *Austrian Academy of Sciences* and RISC, Linz, Austria.

Part of this research has been funded by: `Provincia Autonoma di Trento grant` *"PAT-CRS grant"*, `MIUR grant` *"Algebra Commutativa, Combinatoria e Computazionale"*, `MIUR grant` *"Rientro dei Cervelli"*.

References

[Ang76] D. Angluin, *The four Russians' algorithm for Boolean matrix multiplication is optimal in its class*, SIGACT News **8** (1976), 29–33.

[BR10] E. Bertolazzi and A. Rimoldi, *Fast matrix-rank computation in GF(2)*, Tech. report, arxiv, 2010.

[CGC05] *NSA Suite B Cryptography*, NSA web site, 2005.

[DR02] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer, 2002.

[GV96] G. H. Golub and C. F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.

[Luc96] S. Lucks, *Faster luby-rackoff ciphers*, Proc. of FSE 1996, LNCS, vol. 1039, Springer, 1996, pp. 189–203.

[Nat01] National Institute of Standards and Technology, *The Advanced Encryption Standard*, Federal Information Processing Standards Publication (FIPS) 197, 2001.

[NIS00] *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, Special Publication SP 800-22, NIST, 2000.

[RST10] A. Rimoldi, M. Sala, and I. Toli, *A possible intrinsic weakness of AES and other cryptosystems*, Tech. report, arxiv, 2010, http://arxiv.org/abs/1006.5894.

[Str69] V. Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik **13** (1969), 354–356.