Crypto Engineering (GBX9SY03)
TP — Square attack on $3^1/_2$ rounds of AES

2021-10-08

Grading

This TP is graded as part of the *contrôle continu*. You must send a written report (in a portable format) detailing your answers to the questions, and the corresponding source code **with compilation and execution instructions** by Friday in two weeks (2021-10-22T18:00+0200) to:

pierre.karpman@univ-grenoble-alpes.fr.

Working in teams of two is allowed and encouraged (but not mandatory), in which case only one report needs to be sent, with the name of both students clearly mentioned.

Introduction

The goal of this TP is to implement a simple, yet effective key-recovery attack on a reduced version of the AES. This attack (taking the many names of "square", "'saturation" or "integral" attack) is *structural*, in the sense that it does not depend on many details of the AES, but rather on its overall SPN structure. In fact, it was first developed for the SQUARE cipher, which is a predecessor of the AES (Daemen & al., 1997), and later generalized to even wider settings (Biryukov and Shamir, 2001).

The attack, like many other key-recovery attacks in symmetric-key cryptography, is first based on a *distinguisher*, i.e. a property that allows to decide if one is interacting with a specific algorithm (e.g. the AES) or a "random" one (e.g. a random permutation). In our case, the distinguisher works on 3 rounds of the AES and consists in the fact that for 256 well-chosen plaintexts $p_0, \ldots, p_{255}$, we have that $\mathrm{AES}_3(p_0) \oplus \ldots \oplus \mathrm{AES}_3(p_{255}) = 0$, i.e. the XOR of the 256 ciphertexts encrypted by 3-round AES is the all-zero 128-bit string. As this property is unlikely to hold for a random permutation, we can use it to distinguish the two cases.

We will use this distinguisher to recover the key for $3^1/_2$ rounds of AES (a $^1/_2$ round is a round without MixColumn, cf. below). The idea to do so is the following:

1. Make queries to the $3^1/_2$ oracle that would allow to observe the distinguisher (on 3 rounds), i.e. query $\mathrm{AES}_{3^1/_2}$ with an unknown key on $p_0, \ldots, p_{255}$ as above.
2. Partially decrypt the oracle answers by $^1/_2$ rounds, by making a guess on part of the key.
3. If the guess allows to observe the distinguisher on the partially decrypted ciphertexts, it is assumed to be correct; otherwise another one is made.

AES structure and a 3-round distinguisher

Recall that the AES round function is the composition of four functions: AddRoundKey (ARK), SubBytes (SB), ShiftRows (SR) and MixColumns (MC). The last round omits the MixColumn function and will be counted as a 1/2-round (however, this last round *does* include an extra ARK at the very end. It can thus be written as ARK ∘ SR ∘ SB ∘ ARK). The property which we exploit in the distinguisher derives from the fact that XORing all the 2^n n-bit values $0, 1, \ldots, 2^n - 1$ results in 0. In more details, we use the following facts:

1. We call λ the set $\{\texttt{0x00}, \ldots, \texttt{0xFF}\}$ of all 8-bit values from 0 to 255. If S is a permutation, we have that $\bigoplus_{x\in\lambda} \mathrm{S}(x) = \bigoplus_{x\in\lambda} x = 0$.

2. We consider a set λ' of 256 vectors of $\mathbb{F}_{2^8}^4$ for which one (w.l.o.g. the first) coordinate takes all possible values, and the three others are constant. We write such a set as $(\star, c, c, c)^t$; that is, a coordinate marked c takes the same value in all 256 elements of λ', and one marked $\star$ takes each possible value of $\mathbb{F}_{2^8}$ *exactly once*. Note that the three positions marked c do not have to take the *same* constant value. An example of a set having the property $(\star, c, c, c)^t$ is $\{(i, 0, 1, 2)^t, i \in \mathbb{F}_{2^8}\}$

   Now, as the MixColumn matrix $\boldsymbol{M}$ of the AES does not have any zero coefficient, each coordinate of the elements of the set $\{\boldsymbol{M} \cdot \boldsymbol{x},\ \boldsymbol{x} \in \lambda'\}$ takes each possible value of $\mathbb{F}_{2^8}$ exactly once, i.e. the output set is of the form $(\star, \star, \star, \star)^t$. Note that as for the "$c$" notation above, there is no requirement that all four coordinates of the vectors of this output set be the same. For instance, the set $\{(0, 0, 0, 0)^t\} \cup \{(\alpha^i, \alpha^{i+1}, \alpha^{i+2}, \alpha^{i+3})^t, i = 0, \ldots, 254\}$ for α a generator of the multiplicative group $\mathbb{F}_{2^8}^\times$ is of the form $(\star, \star, \star, \star)^t$.

3. We consider a set λ'' of 256 vectors of $\mathbb{F}_{2^8}^4$ of the form $(\star, \star, \star, \star)^t$. For any matrix $\boldsymbol{A} = (\boldsymbol{A}_{i,j})$ of $\mathcal{M}_4(\mathbb{F}_{2^8})$ (and thence for $\boldsymbol{M}$ in particular), the sum of all elements of $\mathcal{B} := \{\boldsymbol{A} \cdot \boldsymbol{x},\ \boldsymbol{x} \in \lambda''\}$ is the all-zero vector. Indeed, for any of the four coordinates i, the sum across all elements of $\mathcal{B}$ can be rewritten as:

$$\bigoplus_{\boldsymbol{x}\in\lambda} \boldsymbol{A}_{i,0}\boldsymbol{x} \oplus \bigoplus_{\boldsymbol{x}\in\lambda} \boldsymbol{A}_{i,1}\boldsymbol{x} \oplus \bigoplus_{\boldsymbol{x}\in\lambda} \boldsymbol{A}_{i,2}\boldsymbol{x} \oplus \bigoplus_{\boldsymbol{x}\in\lambda} \boldsymbol{A}_{i,3}\boldsymbol{x},$$

   with all $\boldsymbol{A}_{i,j}$s either zero or invertible, hence the above is equal to $0 \oplus 0 \oplus 0 \oplus 0$. We write this property of the output set $\mathcal{B}$ as $(\flat, \flat, \flat, \flat)^t$.

We now define a Λ-set as a set of 256 16-byte plaintexts with one byte taking all possible values ($\star$), and the fifteen other being constant (c, or in white in Figure 1). Thanks to the above facts, we can now graphically follow the propagation of a Λ-set over three rounds of the AES, in Figure 1.*

In words, the sum after 3 rounds of AES of all the 256 ciphertexts whose corresponding plaintexts form a Λ-set is zero.

Exercice 1: Warming up

Download the AES standard at

https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf

and the tarball

https://membres-ljk.imag.fr/Pierre.Karpman/cry_eng2021_tp_aessq.tar.bz2

*Figure slightly adapted from https://www.iacr.org/authors/tikz/.

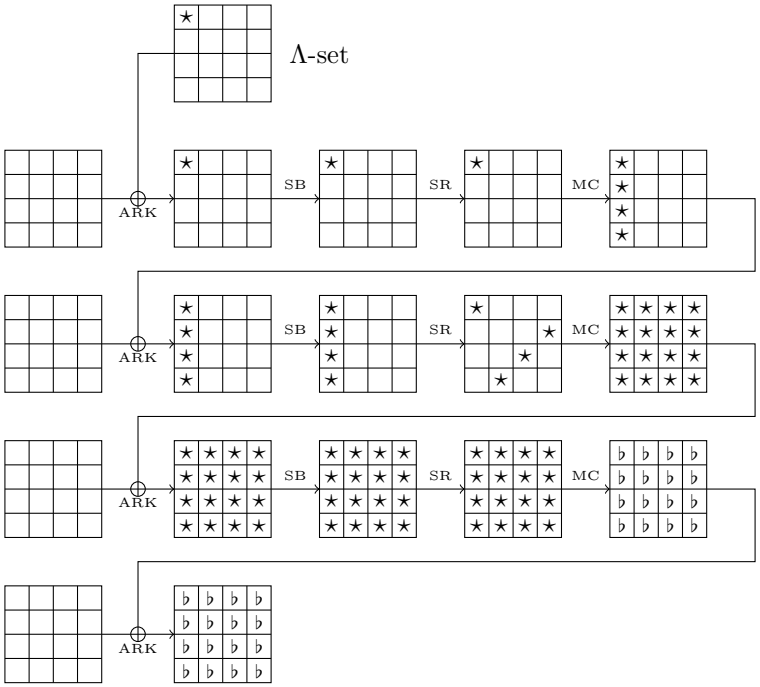

Figure 1: The 3-round Square distinguisher

Q.1 *This question is largely independent of the others.* Explain the role of the function `xtime` in the file `aes-128_enc.c`, and show that it is correct. Write your own variant of `xtime` for a different representation of $\mathbb{F}_{2^8}$ (for instance using the polynomial $X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$, which is irreducible over $\mathbb{F}_2[X]$).

Q.2 Implement the functions `prev_aes128_round_key` (you may look at the implementation of `next_aes128_round_key` for hints). Verify the correctness of your implementation by using the test values provided in the standard document.

✧

A good way to build a keyed function $\mathcal{F}$ from a block cipher $\mathcal{E}$ is to define $\mathcal{F}(k_1||k_2, x)$ as $\mathcal{E}(k_1, x) \oplus \mathcal{E}(k_2, x)$ (see Bellare & al., 1998; Lucks, 2000; and many others). Such a keyed function may then for instance be used to encrypt in CTR mode, or as the basis of a MAC.

Q.3 Implement the above construction with three (full) rounds of AES for $\mathcal{E}$. Why do you need to take $k_1 \neq k_2$ for $\mathcal{F}$ not to be trivial? Show that the 3-round square distinguisher for such an $\mathcal{E}$ also works for the corresponding $\mathcal{F}$, and write a test program to confirm this.

Exercice 2: Key-recovery attack for $3^1/_2$-round AES

Implement a key-recovery attack for $3^1/_2$-round AES (i.e. four rounds, with the last one omitting the MixColumn), using the square distinguisher.

Q.1 Implement the entire attack, and test it on randomly generated keys (e.g. obtained from `/dev/urandom`). That is, query your reduced AES on a Λ-set and iteratively find each byte of the last round key. This can simply be done by guessing the value for each key byte separately and discarding wrong guesses by partially decrypting and checking the 3-round distinguisher. However, don't forget to:

— Filter out false positives (if any) by using a few additional Λ-sets.

— Invert the key expansion to recover the original master key.

Q.2 Check that changing the representation of $\mathbb{F}_{2^8}$ as in (Exercice 1, Q.1) leads to a different cipher, but that the attack still works. Check the same when changing the S-box used in SubBytes and/or the MDS matrix used in MixColumn.

Note: The 3-round distinguisher shown in this exercice can be used in a key-recovery attack up to 6 rounds of AES-128, still exploiting the same kind of process (see Ferguson & al. (2000) for the original attack and Todo & Aoki (2014) for an alternative using FFT techniques). The cost of this 6-round attack is about 2^{32} chosen plaintexts and 2^{50} encryptions, which is expensive but manageable in practice.