Description of the masked AES of the DPA contest v4

Abstract

This note describes the implementation of the AES cipher that is executed for the DPA contest v4 on an ATMEL ATMega 163 smartcard.

Document identification:

`$Id: description_dpa_contest_v4.tex 15233 2013-10-17 16:51:35Z guilley $`

1 Introduction

The cipher to attack is AES-256 in encryption mode, without any mode of operation. It complies with the NIST FIPS standard [NIS01]. In the notations that will follow, some minor adjustments are done with respect to the standard; for instance, depending on the context, SubBytes (resp. MixColumns) can be considered on the whole state or on individual bytes (resp. individual columns). It is mainly coded in the `C` language, and is compiled by `avr-gcc`; only some constants to be stored in Flash memory are given in an assembly code. The implementation is protected against *univariate* side-channel attacks by a masking scheme called "Rotating Sbox Masking" (RSM), and described in [NSGD12].

It can be considered that the protection by masking is added on top of an unprotected AES. This "base AES" has those features:

- The key schedule is precomputed.
- The sixteen[1] substitution boxes (sboxes) are called in this order:

  `0x0, 0x2, 0x4, 0x6, 0x8, 0xa, 0xc, 0xe,` // Even sboxes first
  `0x1, 0x3, 0x5, 0x7, 0x9, 0xb, 0xd, 0xf.` // Odd sboxes second

- The MixColumns operation is computed on a byte-by-byte basis, using an xtime table.

The masking protection is an additive Boolean masked scheme, with statically masked sboxes (as introduced in [Mes00]). It adds to the "base AES" those features:

[1] The substitution boxes are labelled in hexadecimal, from `0x0` to `0xf`, because their index fits exactly on one nibble.

- Sixteen masks M_i, $i = [\![0, 15]\!]$, that are public information, are incorporated in the computation. They are precomputed as state-wide masks, called $\mathsf{Mask}_{\mathsf{offset}}$ and defined as:

$$\begin{aligned}\mathsf{Mask}_{\mathsf{offset}} = (&(M_{\mathsf{offset}+\mathtt{0x0}}, M_{\mathsf{offset}+\mathtt{0x1}}, M_{\mathsf{offset}+\mathtt{0x2}}, M_{\mathsf{offset}+\mathtt{0x3}}),\\ &(M_{\mathsf{offset}+\mathtt{0x4}}, M_{\mathsf{offset}+\mathtt{0x5}}, M_{\mathsf{offset}+\mathtt{0x6}}, M_{\mathsf{offset}+\mathtt{0x7}}),\\ &(M_{\mathsf{offset}+\mathtt{0x8}}, M_{\mathsf{offset}+\mathtt{0x9}}, M_{\mathsf{offset}+\mathtt{0xa}}, M_{\mathsf{offset}+\mathtt{0xb}}),\\ &(M_{\mathsf{offset}+\mathtt{0xc}}, M_{\mathsf{offset}+\mathtt{0xd}}, M_{\mathsf{offset}+\mathtt{0xe}}, M_{\mathsf{offset}+\mathtt{0xf}})) \ .\end{aligned}$$

  Notice that in the equation above, the layout of the bytes is *transposed* with respect to the canonical representation of the state (i.e., lines represent columns).

- A random offset, noted offset, is drawn randomly in $[\![0, 15]\!]$ at the beginning of the computation; it determines the allocation of the masks for each byte of the state. Explicitly, the state byte i is masked by mask $M_{\mathsf{offset}+i}$. In this equation, $\mathsf{offset} + i$ is to be understood "modulo 16". We will do the same assumption in the sequel concerning indices of bytes in a state.

- The sbox is replaced by sixteen masked sboxes, that are stored precomputed; their equation is $\mathsf{MaskedSubBytes}_i(X) = \mathsf{SubBytes}(X \oplus M_i) \oplus M_{i+1}$, where X is a byte. This means that the output mask of each sbox is the *successor* of the input mask. This also explains why sboxes are not called in the natural order; the goal is to prevent unfortunate demasking that might occur otherwise.

- To pass through the linear layer, the mask bytes are *compensated* (by exclusive-or), thanks to sixteen 128-bit precomputed constants, that are equal to:

$$\begin{aligned}&\mathsf{MaskCompensation}_{\mathsf{offset}} = \mathsf{Mask}_{\mathsf{offset}} \oplus \mathsf{MixColumns}(\mathsf{ShiftRows}(\mathsf{Mask}_{\mathsf{offset}})) =\\ &\mathsf{Mask}_{\mathsf{offset}} \oplus (\\ &\quad \mathsf{MixColumns}(M_{\mathsf{offset}+\mathtt{0x0}}, M_{\mathsf{offset}+\mathtt{0x5}}, M_{\mathsf{offset}+\mathtt{0xa}}, M_{\mathsf{offset}+\mathtt{0xf}}),\\ &\quad \mathsf{MixColumns}(M_{\mathsf{offset}+\mathtt{0x4}}, M_{\mathsf{offset}+\mathtt{0x9}}, M_{\mathsf{offset}+\mathtt{0xe}}, M_{\mathsf{offset}+\mathtt{0x3}}),\\ &\quad \mathsf{MixColumns}(M_{\mathsf{offset}+\mathtt{0x8}}, M_{\mathsf{offset}+\mathtt{0xd}}, M_{\mathsf{offset}+\mathtt{0x2}}, M_{\mathsf{offset}+\mathtt{0x7}}),\\ &\quad \mathsf{MixColumns}(M_{\mathsf{offset}+\mathtt{0xc}}, M_{\mathsf{offset}+\mathtt{0x1}}, M_{\mathsf{offset}+\mathtt{0x6}}, M_{\mathsf{offset}+\mathtt{0xb}})) \ .\end{aligned}$$

- For the last round, the compensation is slightly different, because there is no $\mathsf{MixColumns}$. Instead of $\mathsf{MaskCompensation}_{\mathsf{offset}}$, the following constant is added by exclusive-or to the state:

$$\mathsf{MaskCompensationLastRound}_{\mathsf{offset}} = \mathsf{Mask}_{\mathsf{offset}} \oplus \mathsf{ShiftRows}(\mathsf{Mask}_{\mathsf{offset}}) \ .$$

The protected AES can thus be represented by the algorithm 1. The unprotected version of this algorithm can be recovered by erasing the lines in blue, and by trading $\mathsf{Masked}\mathsf{SubBytes}$ for $\mathsf{SubBytes}$.

Algorithm 1: AES-256 used for the DPA contest v4 [TEL14].

Input : Plaintext X, seen as 16 bytes X_i, $i \in [\![0, 15]\!]$,
Key schedule, 15 128-bit constants $\mathsf{RoundKey}[r]$, $r \in [\![0, 14]\!]$

Output: Ciphertext X, seen as 16 bytes X_i, $i \in [\![0, 15]\!]$

Draw a random offset, uniformly in $[\![0, 15]\!]$

$X = X \oplus \mathsf{Mask}_{\mathsf{offset}}$ /* Plaintext blinding */

/* All rounds but the last one */

for $r \in [\![0, 12]\!]$ **do**

  $X = X \oplus \mathsf{RoundKey}[r]$ /* AddRoundKey */

  for $i \in [\![0, 15]\!]$ **do**

    $X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+i+r}(X_i)$

  end

  $X = \mathsf{ShiftRows}(X)$

  $X = \mathsf{MixColumns}(X)$

  $X = X \oplus \mathsf{MaskCompensation}_{\mathsf{offset}+1+r}$

end

/* Last round */

$X = X \oplus \mathsf{RoundKey}[13]$

for $i \in [\![0, 15]\!]$ **do**

  $X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+13+r}(X_i)$

end

$X = \mathsf{ShiftRows}(X)$

$X = X \oplus \mathsf{RoundKey}[14]$

/* Ciphertext demasking */

$X = X \oplus \mathsf{MaskCompensationLastRound}_{\mathsf{offset}+14}$

Acknowledgements

The authors sincerely thank Amir Moradi (RUB, Bochum, Germany) for helping improve this document in particular, and positive feedback about the DPA contests in general. We are also grateful to Ofir Weisse for pointing out errors in formulas.

References

[BCG13] Shivam Bhasin, Claude Carlet, and Sylvain Guilley. Theory of masking with codewords in hardware: low-weight dth-order correlation-immune Boolean functions. Cryptology ePrint Archive, Report 2013/303, 2013. http://eprint.iacr.org/2013/303/.

[CG13] Claude Carlet and Sylvain Guilley. Side-Channel Indistinguishability. In *HASP*, pages 9:1–9:8, Tel-Aviv, Israel, June 23-24 2013. ACM. Extended version: http://hal.archives-ouvertes.fr/hal-00826618/en.

[Mes00] Thomas S. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms*. PhD thesis, University of Illinois at Chicago, USA, 2000. 468 pages.

[NIS01] NIST/ITL/CSD. Advanced Encryption Standard (AES). FIPS PUB 197, Nov 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[NSGD12] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean-Luc Danger. RSM: a Small and Fast Countermeasure for AES, Secure against First- and Second-order Zero-Offset SCAs. In *DATE*, pages 1173–1178, March 12-16 2012. Dresden, Germany. (TRACK A: "Application Design", TOPIC A5: "Secure Systems"). On-line version: http://hal.archives-ouvertes.fr/hal-00666337/en.

[TEL14] TELECOM ParisTech SEN research group. DPA Contest (4^{th} edition), 2013–2014. http://www.DPAcontest.org/v4/.

A The 16 masks

The sixteen masks $M_{i \in [\![0,15]\!]}$ are devised such that they form a code of length $n = 16$ and of size 16 of maximal dual distance, i.e., 4 [BCG13]. This guaranties that the function

$$x \in \mathbb{F}_2^n \mapsto \mathbb{E}\left[\mathscr{L}(X \oplus M) | X = x\right] \quad ,$$

where:

- X represents any sensitive random variable,

- M is a random variable uniformly distributed on the code (i.e., amongst the 16 masks),
- $\mathbb{E}$ is the expectation,

is *constant* (i.e., does not depend on x) for all pseudo-Boolean function $\mathscr{L} : \mathbb{F}_2^n \to \mathbb{R}$ of algebraic degree $d < 4$.

Specifically, the code is chosen linear [CG13], of parameters $[8, 4, 4]$. It is equal to

$$M_{i \in [\![0,15]\!]} = \{\texttt{0x00}, \texttt{0x0f}, \texttt{0x36}, \texttt{0x39}, \texttt{0x53}, \texttt{0x5c}, \texttt{0x65}, \texttt{0x6a}, \\ \texttt{0x95}, \texttt{0x9a}, \texttt{0xa3}, \texttt{0xac}, \texttt{0xc6}, \texttt{0xc9}, \texttt{0xf0}, \texttt{0xff}\} \ .$$

It is auto-dual, hence of identical minimal distance and dual distance.