Project: Correlation Power Analysis (CPA) and Linear Regression Analysis (LRA) Against AES-128 Traces

Draft

Damien Marion

IRISA

Abstract. The project cover all the needed steps to mount CPA against Advanced Encryption Standard (AES) implementations, to recover secret keys using power traces.

Keywords: CPA, LRA, AES, python 3.7, Side-Channel Analysis

1 Materials

- extended version of [5], and the papers [4] and [2]
- Specification for the AES-128 [1],
- datasets:
  - dataset {traces, plaintext, ciphertext, key, source code} from a software implementation (CTF - CHES-2016 [6]) - Acquisition: 32MHz (traget: 32MHz - AVR XMEGA),
  - dataset {traces, plaintext, ciphertext, source code} from a software implementation (CTF - CHES-2016 [6]) - Acquisition: 32MHz (traget: 32MHz - AVR XMEGA),
  - dataset {traces, plaintext, ciphertext, key, source code} from a hardware implementation (DPA-contest V2 [3]) - Acquisition: 5GHz (traget: 24MHz - FPGA, SASEBO-GII),
  - dataset {traces, plaintext, ciphertext, source code} from a hardware implementation (DPA-contest V2 [3]) - Acquisition: 5GHz (traget: 24MHz - FPGA, SASEBO-GII),
- datasets (for the optional tasks):from a software implementations (CTF - CHES-2016 [6])- Acquisition: 32MHz (traget: 32MHz - AVR XMEGA):

Datasets-0x00 :

(a) "software_traces_k_known_countermeasure_0/ {traces, plaintext, ciphertext, key, source code} "

(b) "software_traces_k_unknown_countermeasure_0/ {traces, plaintext, ciphertext, source code} "

Datasets-0x01 :

(a) "software_traces_k_known_countermeasure_1/ {traces, plaintext, ciphertext, key, source code} "

(b) "software_traces_k_unknown_countermeasure_1/ {traces, plaintext, ciphertext, source code} "

Datasets-0x02 :

(a) "software_traces_k_known_countermeasure_2/ {traces, plaintext, ciphertext, key, source code} "

(b) "software_traces_k_unknown_countermeasure_2/ {traces, plaintext, ciphertext, source code} "

2 Notations

The notations use for the project differ from the one used in [5].

Whatever the target and regardless of the acquisition method, an attacker will record traces and additional data as inputs and/or outputs values. Matrix is the perfect mathematical object to store and manipulate these collected data. That is why we adopt in the whole project matrix notations. The traces are indexed by $q = 0, \ldots, Q-1$ where Q is the number the of traces, the samples in a given traces are indexed by $d = 0, \ldots, D-1$. In this way, let $X^{D,Q}$ denote a matrix containing Q side-channel traces of D samples:

$$X^{D,Q} = (X_{d,q})_{\substack{d<D,\\ q<Q}},$$

where $d = 0, \ldots, D-1$ is a row index and $q = 0, \ldots, Q-1$ is a column index. We also denote all dth samples for all traces as $(X_{d,q})_{q<Q} = X_d^Q$, and all the samples for the qth trace as $(X_{d,q})_{d<D} = X_q^D$. Thus, X_d^Q is a row vector and X_q^D is a column vector. Two matrices noted side-by-side are implicitly multiplied.

In the same way, let denote $\mathrm{P}^{\mathrm{B},Q}$ (resp. $\mathrm{C}^{\mathrm{B},Q}$) the matrix containing Q plaintext (resp. ciphertext) of B bytes. Where $\mathrm{P}_q^{\mathrm{B}}$ is the plaintext of B bytes, for the qth encryption.

Let $\{Y_b^Q(k_b)\}_{b<B}$ denote the matrix containing the modelled targeted intermediate values dependant of the secret k_b. Where $Y_{q,b}(k_b)$ contains the modelled values of the bth byte of the targeted intermediate value for the qth execution that dependant of the secret k_b. $Y_{q,b}(k_b)$ is function of a known variable, $\mathrm{P}_{q,b}$ or $\mathrm{C}_{q,b}$, and a secret one, k_b. We distinguish two cases, when the secret byte is known using is denoted $k_b^\star$ while when is unknown (or guessed) is denoted k_b.

$$\begin{array}{rcrl} \phi : (\mathbb{Z}/2\mathbb{Z})^8 & \times & (\mathbb{Z}/2\mathbb{Z})^8 \rightarrow & \mathbb{N} \\ \mathrm{P}_{q,b} \| \mathrm{C}_{q,b} & , & k_b \rightarrow & Y_{b,q}(k) = \phi(\mathrm{P}_{q,b} \| \mathrm{C}_{q,b}, k_b) \end{array}$$

3 Steps

1. Read one of the following paper [5], [4] or [2] (one paper per student)
2. python implementation of an AES-128 ,
3. implementation of two leakage models:

– Hamming Weight (HW) of the output of the S-boxat the first round:

$$Y^{\mathrm{B},Q}(k) = \{Y_{b,q}(k_b) = \mathrm{HW}(\text{S-box}(\mathrm{P}_{b,q} \oplus k_b))\}_{\substack{\mathrm{B}<b \\ q<Q}}$$

– Hamming Distance (HD) between the ciphertext and the input of the S-boxat the last round:

$$Y^{\mathrm{B},Q}(k) = \{Y_{b,q}(k_b) = \mathrm{HW}(\text{S-box}^{-1}(\mathrm{C}_{b,q} \oplus k_b) \oplus \mathrm{C}_{b,q})\}_{\substack{\mathrm{B}<b \\ q<Q}}$$

4. implementation of the CPA, LRA,
5. mount a CPA and a LRA against the dataset: dataset {traces, plaintext, ciphertext, key, source code} from a software implementation (CTF - CHES-2016 [6]) and dataset {traces, plaintext, ciphertext, key, source code} from a hardware implementation (DPA-contest V2 [3]) , verifying that your are able to recover the provided key.
6. mount a CPA and a LRAagainst the dataset: dataset {traces, plaintext, ciphertext, source code} from a software implementation (CTF - CHES-2016 [6]) and dataset {traces, plaintext, ciphertext, source code} from a hardware implementation (DPA-contest V2 [3]) , verifying that your are able to recover the known the right key using pairs of plaintext and ciphertext.
7. *(optional)* mount a CPA or a LRA against the Datasets-0x00b, the Datasets-0x01b and the Datasets-0x02b using pre-processings in order to defeat the embedded countermeasure. To test your attack you can use the datasets with the datasets with the known key (Datasets-0x00b, Datasets-0x01b, Datasets-0x02b).

References

[1] Federal Information Processin Standards Publication 197, ed. *Specification for the Advanced Encryption Standard (AES)*. link to the document: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`. 2001.

[2] Eric Brier, Christophe Clavier, and Francis Olivier. *Correlation Power Analysis with a Leakage Model*. Ed. by Marc Joye and Jean-Jacques Quisquater. eprint version `https://www.iacr.org/archive/ches2004/31560016/31560016.pdf`. 2004. DOI: `10.1007/978-3-540-28632-5_2`. URL: `http://dx.doi.org/10.1007/978-3-540-28632-5_2`.

[3] Christophe Clavier et al. *Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest*. link to the DPA-contest V2 website: `http://www.dpacontest.org/v2/`. 2014. DOI: `10.1007/s13389-014-0075-9`.

[4] Julien Doget et al. *Univariate side channel attacks and leakage modeling*. 2011. DOI: `10.1007/s13389-011-0010-2`. URL: `https://doi.org/10.1007/s13389-011-0010-2`.

[5] Victor Lomné, Emmanuel Prouff, and Thomas Roche. *Behind the Scene of Side Channel Attacks*. Ed. by Kazue Sako and Palash Sarkar. Extended version freely available: https://eprint.iacr.org/2013/794.pdf. 2013. DOI: 10.1007/978-3-642-42033-7_26. URL: https://doi.org/10.1007/978-3-642-42033-7_26.

[6] Colin O'Flynn. *CHES 2016 Capture the Flag - (NewAE)*. http://ctf.newae.com/flags/. 2016.