REASSURE Dataset AES-128 32bit FPGA Implementation (v1.0)

Davide Bellizia
Université Catholique de Louvain - Crypto Group
(davide.bellizia@uclouvain.be)

4^{th} October 2018

1 Introduction

The dataset included in this package has been collected for the T3.1 "Dataset for AES" of the REASSURE H2020 project by UCLouvain Crypto Group. The aim of this task is to provide datasets for researchers and developers to test their algorithms on real datasets.

2 Experiment

The dataset included in this package contains 1M of power consumption traces collected on AES-128 hardware implementation, divided in 50 files of 20k traces each. These traces have been collected adopting the following equipment:

- Oscilloscope: PicoTechnologies PicoScope 5244B;
- Probe: Tektronix CT-1, 1GHz inductive probe;
- Test Board: Sakura-G Board (R1);
- Device Under Test (DUT): Xilinx Spartan-6 LX75 (FPGA);
- Connectors: Rodhe&Schwarz HZ-22, BNC 50Ω adapter.

The CT-1 probe has been connected on the JP2 header of the SAKURA-G board, and through the 50Ω adapter to the input Channel A of the PicoScope. The power supply for the core of the FPGA has been set to the nominal voltage of $1.2V$ through the on-board trimmer VR1.The clock frequency has been set to 4MHz.

The traces have been collected by using a custom script in Python 2.6, adopting a custom script,and using the following libraries:

- picoscope;
- serial;
- numpy;
- scipy.io.

The communication between the PC and the DUT has been implemented through UART communication at $57.6baud/s$. The *trigger* signal is raised to level ‘high’ at the beginning of the encryption operation. It maintains level ‘high’ till the end of the cryptographic processing, to not alter the probing on the power supply. After the encryption, the *trigger* is signal is set to logic ‘low’. This signal has been routed to pin 1 of CN3 of the SAKURA-G board, and it is then connected to ”Ext” input of the oscilloscope through BNC probe cable, in order to provide synchronization.

2.1 Oscilloscope Parameters

The oscilloscope parameters used to produce this dataset are reported in Table 1.

Parameter	Value
Ch.A Range	$+/-20$mV
Ch.A Offset	0mV
Ch.A Coupling	DC
Trigger Ch.	EXT
Trigger Level	1V
Trigger Delay	0s
Trigger Direction	Rising Edge
Sampling Frequency	125MS/s
Resolution	14bit

Table 1: Oscilloscope parameters.

2.2 Input vectors

The 128bit key has been fixed at the beginning of the experiment, and it is contained in each file *traces_AES128_32bit_X.mat* of the dataset. The hexadecimal representation of the key is reported in the following:

$$key = CA08070605040302010A0B0C0D0E0FA0 \tag{1}$$

Regarding the plaintext vectors, they have been generated randomly from a uniform pool, by means of the function *numpy.random.randint* in the Python script.

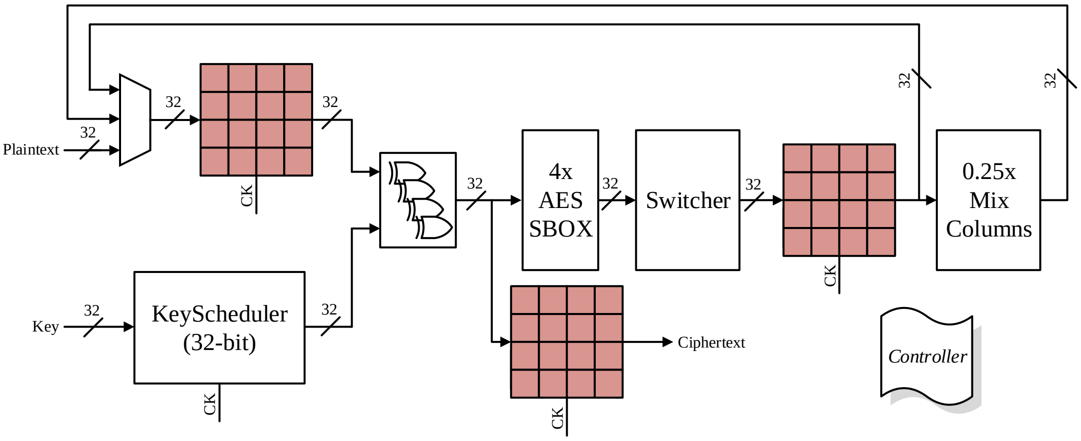

Figure 1: Target architecture.

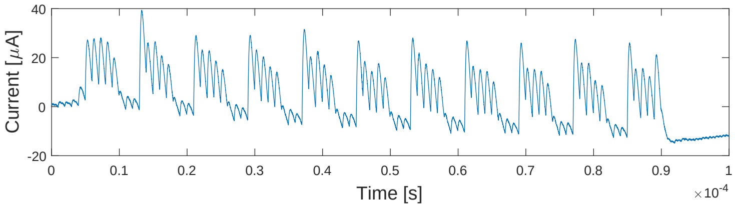

Figure 2: Example of power trace.

3 Implementation

In this experiment, we wanted to propose a very simple hardware implementation of the AES-128 algorithm for encryption, that is not meant to be optimized in speed nor in performance. The aim of this implementation is to get non-experts and developers familiar with power analysis and hardware implementation. The architecture is a iterative loop implementation of the AES-128 block cipher (encryption only) with a data-path width of 32bit, as shown in Fig. 2.
The combinational units implement byte-wise the three basic computation of the AES Rijndael algorithm: 32 XOR gates for the *AddRoundKey*, 4 Sboxes implement 14 of the original *SubsBytes* operation, and a *MixCol* unit implement 14 of the original *MixColumns* operation. The *ShiftRows* operation (along with the transposition to compute the *MixColumns*) is intrinsically performed in the sequence that the *Controller* load the data in the pipeline.
This implementation has not RTL-level optimization to reduce the memory and logic resources, since it aims at providing an easy-to-analyze example of hardware primitive for side-channel evaluation. Each round is performed in 8 clock cycles (it can be done in 4).

The clock frequency has been set to 4MHz.

4 How to read the data

As mentioned before, the dataset is divided in 50 files that contain 20k power traces each. The file has a format that is completely compatible with Matlab 2017, which is based on the HDF5 format. In the following, the data structure of each file:

- **traces**: 20000 × 2500, power traces raw data from the oscilloscope. Each row contains all samples collected on a single encryption (all rounds) of a plaintext. The data format is *int16*.
- **pt**: 20000 × 16, plaintext vectors. Each row contain 16 bytes encoded in *uint8*.
- **ct**: 20000 × 16, output ciphertext vectors. Each row contain 16 bytes encoded in *uint8*.
- **key**: 1 × 16, the secret key in Eq. 1, encoded in *uint8*.
- **Queries**: the number of power traces with the *.mat* file.

An additional file is provided, named *Configuration_AES128_32bit.mat*, which contains experiement parameters reported in Table 1.