Software implementation of CRA and TRA to recover the AES-128 key using side-channel signals with Python3

Desislava Nikolova[1], Ivaylo Vladimirov[2] and Zornica Terneva[3]

Abstract – **In this scientific paper is presented a software implementation of two side-channel radio attacks: CRA – Correlation Radio Attack and TRA – Template Radio Attack. The main goal is to recover the AES-128 key from an unknown system. Both attacks are developed in Python3 using a Linux-based computer. The algorithm uses Probability Density Functions and the Hamming-weight model. The results are calculated by the Partial Guessing Entropy. The software is run on the CPU.**

Keywords – **Side-channel attacks – Correlation Radio Attack, Template Radio Attack, AES-128, Python3, Hamming-weight model, Partial Guessing Entropy.**

I. INTRODUCTION

Every type of system has sound, heat, light or other electromagnetic radiation while operating. They can be correlated with what is happening inside the system. These physical sources can be used as side channels if they leak out information (Figure.1). Side channel attacks are used to derive the key of a cryptosystem and use its weakness in the physical implementation [1].

In this paper, we are going to analyze two types of attacks - TRA and CRA. Template radio analysis relies on a multivariate model of the side-channel traces. Here we present a practical and efficient implementation. CRA correlation radio analysis is used to identify the parameters of the leakage model.

All systems used in these experiments are using AES-128(Advanced Encryption Standard). AES is a specification for the encryption of different electronic systems and its data.

One of the leading programming languages in data science is Python. Although it might be a little bit slow, it gives a lot of solutions to the most common problems. The best advantage of the language is the wide variety of libraries and already developed tools in it.

In this paper we are presenting a few python libraries, which are going to be used in the software implementation. The name of the package click comes from Command Line Interface Creation Kit. It gives the opportunity the command line interfaces to be written with as little code as possible in order to function correctly.

[1]Desislava Nikolova is with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria. E-mail: desislava.v.nikolova@gmail.com

[2]Ivaylo Vladimirov is with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mail: ivaylo.h.vladimirov@gmail.com.

[3]Zornitza Terneva is with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria. E-mail: z.terneva@abv.bg

Numpy, usually referred as np is the most common library for dealing with arrays and computing them fast in python. Pyplot from matplotlib is used mainly for visualization of the results. The package os in this approach is used only for finding the correct path and managing the files in the directory.

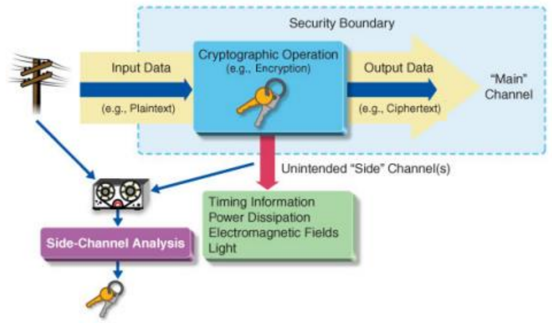


Fig. 1. Steps that lead to the radio transmission of the leak

The function multivariate_normal from scipy.stats is used for the normalization process in CRA. Pickle implements binary protocols for serializing and de-serializing a structure just like our configuration. The library itertools is used because it implements a number of iterator building blocks similar to Haskell. And finally binascii contains methods to convert from binary to various ASCII-encoded binary representations.

II. MATH MODEL OF THE ATTACKS

A. Correlation Radio Attack (CRA)

The first step of the CRA is to generate an intermediate value during the cryptographic operation, encryption or decryption. The selection function must be chosen. This is a function of a variable data value and part of the key. In the equation below I is the intermediate value. The variable data sample which is generally plain text or cipher text is d and k is a part of the key which will be called the subkey from this point on [2].

$$I = f(d, k) \tag{1}$$

Then measurements must be done while the device is doing the cryptographic operation.

In addition, the intermediate values for each of the used variable data samples and the power consumption for those intermediate values must be calculated using a power model such as the Hamming Distance Model. This calculation must be repeated for all possible values of the subkey.

These calculated data values are the hypothetical power consumption values.

The Hamming distance between D and R, also described by H(D $\oplus$ R), encloses the Hamming weight model which assumes that R = 0. The independent variable b encloses offsets, time dependent components and noise and a is a scalar gain between the Hamming distance and W the power consumed (2).

$$W = aH(D \oplus R) + b \qquad (2)$$

The last step is a comparison between the real and the theoretical power consumption for each subkey in order to find the one that is most likely to be used in encryption, based on the highest correlation coefficient [3].

The variance dependencies are: $\sigma_W^2 = a^2\sigma_H^2 + \sigma_b^2$. The correlation factor between the Hamming distance and the measured power to assess the linear model-fitting rate is ρ_{WH}. The covariance between both random variables is normalized by the product of their standard deviations. The dependences can be observed on equation (3).

$$\rho_{WH} = \frac{cov(W,H)}{\sigma_W\sigma_H} = \frac{a\sigma_H}{\sigma_W} = \frac{a\sigma_H}{\sqrt{a^2a_H^2+\sigma_b^2}} = \frac{a\sqrt{m}}{\sqrt{ma^2+4\sigma_b^2}} \qquad (3)$$

B. Template Radio Attack (TRA)

Many models can be inferred using a template attack. The first step is to make a calculation about the distance between the template probability density functions (PDFs). A template attack will give better results if the PDFs are far one from each other. As first order, it is proportional to the number of commutations of the gates. The energy dissipated by a portion of the hardware is thus measured by a Hamming distance. However, the best way to distinguish the templates is to build them in such a way that their dissipations are as different from each other as can be. As the dissipation is correlated to the Hamming distance, the best mapping is the Hamming distance between two successive values of the keys [4].

On average, it requires a lot of preparation and retrieval of many power traces, but the advantage of the SRA is that it is then necessary to collect very few traces of the attacked system to completely break the encrypted communication. It is even possible to extract the key from only one trace. The construction of a template consists in calculating the pair of parameters: $\mu_{K_j}^{|N|}$, $\Omega_{K_j}^{|N|}$ - the average value of the signal, noise distribution also known as the covariance matrix of noise. They are calculated using statistical formulas (4), (5) and (6) on all collected data for a given key. The final equation is (7).

$$\mu_{K_j}^{|N|} = \frac{1}{M_{K_j}}\sum_{i=1}^{M_{K_j}} \chi_{K_j}^{i}(t) \qquad (4)$$

$$\Theta_{K_j}[i,:] = \begin{pmatrix} \chi_{K_j}^{1}(t) & - & \mu_{K_j}^{|N|} \\ \vdots & \vdots & \vdots \\ \chi_{K_j}^{M_{K_j}}(t) & - & \mu_{K_j}^{|N|} \end{pmatrix} \qquad (5)$$

$$\Omega_{K_j}^{|N|} = \frac{1}{M_{K_j}}\left[\Theta_{K_j}^{T}\Theta_{K_j}\right] \qquad (6)$$

$$K^* = \mathrm{argmax}_{K_j}\left(\frac{1}{\sqrt{(2\pi)^N\left|\Omega_{K_j}^{|N|}\right|}}\exp\left(-\frac{1}{2}\left(\chi_{K^*}(t) - \mu_{K_j}^{|N|}\right)^T\cdot\left(\Omega_{K_j}^{|N|}\right)^{-1}\cdot\left(\chi_{K^*}(t) - \mu_{K_j}^{|N|}\right)\right)\right) \qquad (7)$$

III. Python Model of the Attacks

A. Correlation Radio Attack (CRA)

Correlation Radio Analysis runs a standard correlation attack against a data set, trying to recover the key used for the observed AES operations. The attack works by correlating the amplitude-modulated signal of the screaming channel with the power consumption of the SubBytes step in the first round of AES, using a Hamming-weight model.

Firstly we load the known key, the number of traces and the number of points. Then we create lists to store the information for the best guess, the PGE and the stored_cpas from maxcpa. The following steps are for each and every one of the key bytes:

The output of the CRA and its maximum value are creates as empty lists.

For every guessed subkey, its value is printed. Then we initialize arrays and variables to zero. We create a hypothesis using the HW model. The mean of the hypothesis and the mean of all points in the trace are performed for every subkey. And for every one trace of the guessed subkey, we are calculating the difference between the hypothesis and the difference in the traces. After this we calculate the output of the CRA and print the maximum value.

We find and print the PGE for every key byte in order to compare them easily.

At the end we present the final results for the best achieved guess for every key and the PGE.

B. Template Radio Attack (TRA) - Learning

The data set should have a considerable size in order to allow the construction of an accurate model. In general, the more data is used for template creation the less is needed to apply the template.

Creating the template is the so-called training phase consisting of measuring the electrical activity of a clone of the system that will be attacked during a large number of encryptions executed with K number of different keys on random data sets. To perform it, the attacker must have access to and full control over a copy of the system that will be attacked.

This is necessary because you need to create models - "templates" for each possible key that can be used. In practice, it requires a lot of preparation and retrieval of many power "traces", but the advantage of the TRA is that it is then necessary to collect very few "traces" of the attacked system to completely break the encrypted communication. It is possible to extract the key from only one "trace".

In the code for every byte in the key we are using a template S-Box from the S-Box, which is used as a lookup table in AES. Then we sort the traces by the Hamming-weigh (HW) model.

Make blank lists - one for each Hamming weight and fill them up with the traces. Then we have to check if there is at least a trace for each HW. They are switched to numpy arrays, where we find the average values and the sum of their differences.

The next step is to locate the points of interest, which is done by finding the maximum sum of their differences. In order to finish the training phase, we create the mean matrix and the covariance matrix for each HW.

We display the points of interest and the two matrixes for better understanding of the results.

C. Template Radio Attack (TRA) - Attacking

Template Radio Analysis applies the already prepared template to a new system with the intention to attack the key in a new data set. The template's directory must be the location of a previously created template with compatible settings as same trace length.

The real attack starts first with creating the lists for the scores, the best guess and the Partial Guessing Entropy (PGE) this saves a byte which indicates the distance between real and best guess. The closer it is to 0, the closer it is to the real value. It can have a value of 255 if it is completely wrong

For each and every one of the bytes in the key, we are doing the following steps:

In order to de-serialize the data stream of the point of interest, covariance matrix and mean matrix, we use the loads() option of the library pickle. To simplify and speed up the calculations, the number of reports is reduced, leaving only the points of interest.

Principal Component Analysis (PCA) is used for their selection. This is a statistical approach allowing the identification of patterns/trends/patterns in the data. In it, with the help of a covariance matrix, only the most significant values are selected. Creating these templates requires taking measurements in about 20 hours, while collecting attack data, with less than 1,000 traces, takes only 15 minutes. We create a ring buffer for keeping track of the last N best guesses using a window with start index 0 and 10 dimensions

The result of the total average values of the signal is the covariance matrix.

For each trace, we take the key points and put them in a new matrix. In addition, after this, each and every key is being tested. A Hamming-weight (HW) model is made of the S-Box. The HW is normalized and is transformed into a probability density function. The logarithm of this one is stored in a list together will all of the keys for a given trace.

The last list is sorted and sored into the main matrix. If a proper key is found, we display the subkey and the number of traces in which it was found.

The best guess for every key and the PGE are displayed at the end.

IV. RESULTS

A. Running an attack command

In order to run the two analysis in the Linux terminal there are a few parameters that you need to know. The most important for the attacks are the number of traces --num-traces (the maximum available if you pass 0), and the attack window defined by --start-point and --end-point (or the entire trace if you do not specify them). [5]

In order to choose the window one should plot (--plot) the traces. Use the two shown plots to manually identify the first round of AES, at the output of the S-Box. The attack works better this way, because we eliminate a lot of points which slow down the processing and add noise. The tool will first show all the traces aligned on the same plot (careful if there are many of them) and then the Sum of Absolute Differences (SOAD) - which should be higher where the traces differ. [6]

```
$ sc-attack --plot --data-path=../traces/mbedtls_1m_home –name =mbed_villa_500 --start-point=0 --end-point=0 --num-traces=1000 cra
```

Fig. 2. Example command

B. Running a template creation command

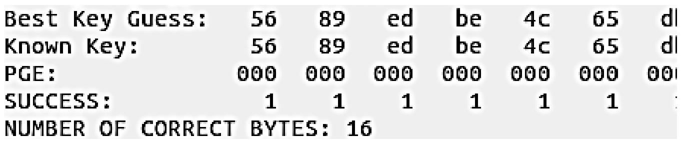
First, we use the template set to create a template. We first create a folder where we can store the template. Again, we could set the window, but we do not need it. The template attack first uses the SOAD to detect the points of interest that is the point of the trace that "leak". They are simply the peaks of the SOAD. One can configure how many peaks you want with --num-pois and the minimum distance between adjacent peaks with --poi-spacing. Note that the number of traces is very large, so it is better to use the plot option on a smaller subset.

```
$ sc-attack --data-path=../traces/tinyaes_anechoic_10m --start-point=0 --end-point=0 --num-traces=129661 tra_create tra_templates/example_template --num-pois=10 --poi-spacing=1
```

Fig. 3. Example command

C. Numerical results

The numerical results that appear after using the attack command consist of a table with five lines. The first two lines show the bytes of the best guess and of the known key, respectively. The PGE is the Partial Guessing Entropy, which is the "distance" of our best guess from the known byte, and therefore it is 0 when the guess is correct. Our tool ranks the guesses, with the most likely at 0 and the least likely at 255, and the PGE is the rank of the known key.



```
Best Key Guess:   56   89   ed   be   4c   65   d
Known Key:        56   89   ed   be   4c   65   d
PGE:             000  000  000  000  000  000  00
SUCCESS:           1    1    1    1    1    1
NUMBER OF CORRECT BYTES: 16
```

Fig. 4. The result given in the terminal

D. Graphic results

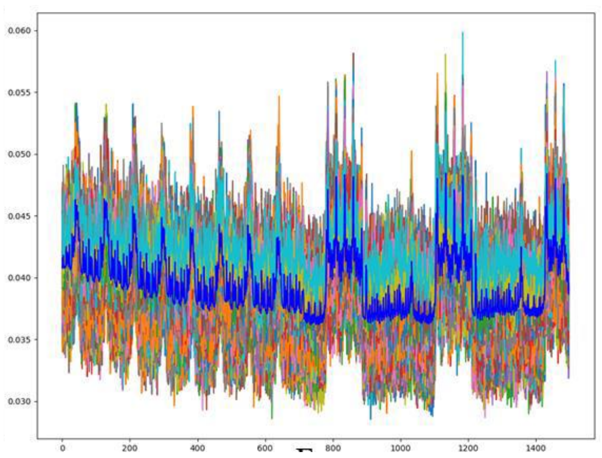


Fig. 5. Plot of all used traces aligned

The *–plot* tool of the *sc-attack command* will first show all the traces aligned (see Figure. 4) on the same plot and then the Sum of Absolute Differences (see Figure.5)(which should be higher where the traces differ, e.g. because of the key).

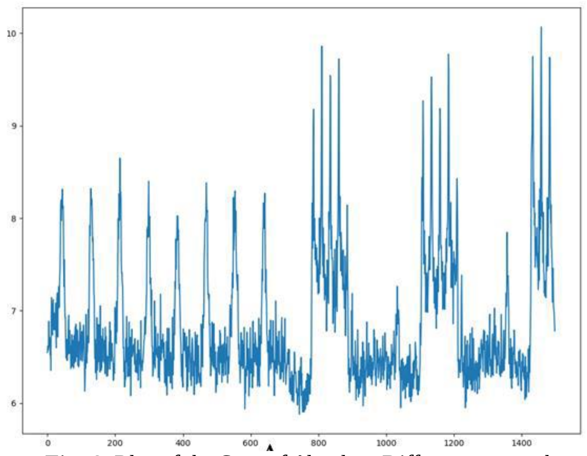


Fig. 6. Plot of the Sum of Absolute Differences graph

V. Conclusion

In this paper we presented an approach for software implementation of two side-channels attacks with the main goal of recovering the AES-128 key from an unknown system. CRA and TRA can be used in the field of security depending on the given model and its encryptions.

Although the opportunities for those systems are a lot, the programming realization is fairly similar and can be optimized in order to meet the required results. In our approach we are using the Probability Density Functions and the Hamming-weight model to calculate the Partial Guessing Entropy. The results are plotted in order to be easily understandable and analyzed. Because of its simplicity and variety of built-in functions Python3 was used as the main programming language.

The future of the field of security systems is not clear but can have different directions depending on our focus and research.

Acknowledgement

The research described in this paper is partialy realized within the Erasmus+ mobility program with the help of the profesors at SUPELEC University in Rennes, France.

The hardware implementation of the codes for the two side-channel power analysis techniques used in this paper are described in a scientific document: “Hardware implementation and comparison of CRA and TRA when trying to recover the AES-128 key” [7] that will also be presented at the ICEST 2020 conference.

References

[1] G.Camurati S.Poeplau M.Muench T.Hayes A.Francillon, „Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers”, 2018;

[2] V.K.Rai B.V.Reddy S.Tripathy, “Correlation power analysis and effective defense approach on light encryption device block cipher”, 2019;

[3] E.Brier C.Clavier F.Olivier „Correlation Power Analysis with a Leakage Model“, 2004

[4] M.E.Aabid S.Guilley, „Template Attacks with a Power Model“, 2007;

[5] H.Li A.T.Markettos S.Moore, „Security Evaluation Against Electromagnetic Analysis at Design Time – CHES 2005;

[6] G.Camurati,“Screaming_channels”,https://github.com/eurecom-s3/screaming_channels;

[7] I.Vladimirov D.Nikolova Z.Terneva, “Hardware implementation and comparison of CRA and TRA when trying to recover the AES-128 key”, 2020.