Physical Attacks on Secure Systems (NWI-IMC068)

Tutorial 3: DPA on Xoodyak

Goals: After completing these exercises successfully, you should understand the challenges in attacking a hardware implementation (compared to software) and you should have a better understanding of how Correlation Power Analysis (CPA) works.

Before you start: Look at the slides of Lecture 3 for an explanation of the Xoodyak algorithm and the attack that you are going to perform.

Instructions: Download the notebook and traces for this tutorial from Brightspace.

1 Get to know your traces

The target of our attack is a hardware implementation of Xoodyak[1] running on the Sakura-G FPGA board, a development board specifically designed for SCA.

1.1 Check dimensions

Load the power traces in the notebook and answer the questions below.

Q1. How many traces are present in the dataset?

Q2. How many samples does each trace have?

Q3. What are the dimensions of the nonce array? What does each dimension denote?

Q4. What is the correct key (for verification later)?

1.2 Visual examination of traces

Let's make a plot of Xoodyak to get an idea what the power traces look like.

Q5. Plot the average power trace. How many rounds of Xoodoo can you count?

[1] `https://github.com/KeccakTeam/Xoodoo/tree/master/Hardware/FPGA/AEAD/Xoodyak_R3`

1.3 Cut the traces

For this tutorial, we want to attack the first round of Xoodyak. Visually inspect where the first round starts and ends for selecting a time window. Plot the average truncated traces to verify your window.

Q6. What time window did you select?

Q7. What is the advantage of cutting the traces?

2 Getting to know Xoodoo - define some auxiliary functions

Xoodyak is a lightweight authentication encryption scheme with associated data. It uses the Xoodoo permutation using a duplex construction. The Xoodoo round function consists of 5 parts: linear mixing, plane shift, round addition, S-box and another plane shift. The sensitive intermediate variable for our side-channel attack will again be the s-box. In order to perform DPA on the hardware implementation of Xoodyak, we need to keep track of the internal state. We have provided you with the Xoodoo state and the Xoodoo step functions in the notebook to carry out the attack. Follow the instructions in the jupyter notebook and answer the questions below.

2.1 Extract bits from the state

Before we start the attack, let us define some helper functions to use later on. Implement the functions `get_bit()` and `get_column()` in the notebook.

Q8. What do we consider a *plane* in Xoodyak? How is this represented in our Python implementation?

Q9. What do we consider a *lane* in Xoodyak? How is this represented in our Python implementation?

Q10. What do we consider a *column* in Xoodyak? How is this represented in our Python implementation?

2.2 Load in Xoodoo state

We have provided you with an example key and nonce in the Jupyter notebook. Have a close look at how Xoodyak works. Load the key and nonce into the xoodoo-state.

Q11. In which plane is the key stored?

Q12. In which plane is the nonce stored?

Q13. What is the function of the plane with domain-specific bits?

2.3 Define Xoodoo step functions

In the Jupyter notebook, we have provided you with the step functions used in Xoodoo. Answer the questions below.

Q14. Have a look at the step functions provided. Match the greek letters ($\theta, \iota, \chi, \rho_{west}, \rho_{east}$ to their corresponding step functions: round addition, plane shift (right), s-box, plane shift (left) and linear mixing.

Q15. Print the Xoodoo state after absorbing the example key and nonce.

Q16. Using the Xoodoo step functions provided, calculate the Xoodoo-state after the s-box step.

Q17. Using the Xoodoo step functions provided, calculate the Xoodoo-state after a whole round.

3 CPA attack using the HD model

In this tutorial, you will attack Xoodyak using the Hamming Distance (HD) model and Pearson correlation. We adopt a divide-and-conquer approach, in which we recover one column at a time. In this tutorial, we will try to recover the value of column (0,0).

We summarize here briefly the steps of the attack for Xoodoo:

1. We take the three known bits of our Nonce-state. This is the column after the iota-operation $\mathtt{N}^{\lambda}$: (n_0, n_1, n_2), which we manually calculate from the known nonce.
2. We guess the three bits of $\mathtt{K}^{\lambda'}$: (k_0, k_1, k_2).
3. We compute the (hypothetical) intermediate value $\chi_3(n_0 + k_0, n_1 + k_1, n_2 + k_2)$ (output of the 3-bit S-box of Xoodoo).
   (a) Note: for computing χ, we need to supply three bits, which is the target column. Recall that the target column contains bits of the three planes: the key plane, the nonce plane, and the domain-specific plane. However, the key is unknown, so we need to guess this bit as well. This will be k_3.
4. We build the power-prediction matrix: we compute the Hamming Distance between the two states (the original state and the state after χ or our target column.
5. We compute the correlation between the hypothetical power consumption values and the real measurements.

You will be guided through the attack. To calculate our intermediate variable, we have provided you with two other auxiliary functions for χ and ρ. Complete the steps in the Jupyter Notebook and answer the questions below.

4 Build the states from the nonces

Now, we are going to prepare our attack. To calculate our intermediate sensitive variable for each trace/nonce pair, we need to load in the nonce into the Xoodoo-state. Complete the code in the Jupyter notebook.

Q18. To get our target state lambda, which step functions should we include in calculating the state?

5 Compute the power-prediciton matrix

Given all our xoodoo-states for each trace/nonce pair, we can compute the intermediate variable. Follow the instructions in the notebook to complete the functions `compute_intermediate_value()` `extract_original_bits()` and `compute_power_prediction()`. Answer the questions below.

Q19. How many bits do we need to guess? Why?

Q20. How many possible key guesses do we have?

Q21. What is the shape of our value-prediction matrix?

6 Comparison with real traces using CPA

Now, it is time to attack. Compute the correlation between the value-prediction matrix and the real power traces. Extract the key with the auxiliary functions given in the notebook and answer the questions below.

Q22. What is the shape of the correlation matrix?

Q23. What is the correct key guess for our target column (0,0)?

Q24. What does each key guess represent? i.e. Which bits denote the column bits and which one denotes the extra bit we had to guess?

Q25. Would the attack also work using the Hamming weight (HW) model for column (0,0)? Explain your answer.