

LAB INTRODUCTION

TTM4205 – Lecture 10

Caroline Sandsbråten

15.09.2026

Contents

Introduction

ChipWhisperer

Your Equipment Box

ChipWhisperer Models

Installation

Contents

Introduction

ChipWhisperer

Your Equipment Box

ChipWhisperer Models

Installation

Today

Lecture 9 was about what leaks and why. This lecture is about the equipment you will use to measure it yourself, and about getting it running on your machine.

The cryptography comes back afterwards: Lecture 11 is power analysis on AES, which is the SCA Lab, and Lecture 12 is timing and power attacks on RSA and ECC, which are the RSA and ECC Labs.

Bring your laptop. We install together at the end.

The Lab at a Glance

- ▶ Four parts: SCA (8 points), Fault (11), RSA (12), ECC (6)
- ▶ That is 37 points of work, and the score is capped at 30
- ▶ So you have 7 points of slack: you can skip a task that will not work
- ▶ Solved in groups of 2 to 3; register the group in Canvas first
- ▶ One joint report per group, written in $\text{\LaTeX}$, mandatory template
- ▶ Deadline: December 4th at 23:59

Everything here is stated in the assignment PDF; read it before you start.

What You Will Do

- ▶ **SCA Lab:** capture AES power traces with and without jitter, resynchronize them with SAD, and see why that makes CPA succeed
- ▶ **Fault Lab:** build a clock-glitch loop, then use it to bypass a password check, dump memory from a bootloader, and recover an AES key
- ▶ **RSA Lab:** capture power traces from an RSA decryption on the XMEGA target and recover the key bit by bit from the trace
- ▶ **ECC Lab:** using pre-recorded traces, recover the secret nonce in an ECC point multiplication one bit at a time

Each part is a notebook in the `jupyter/lab` folder of the lab repository; the assignment PDF states exactly what to hand in for each task.

Contents

Introduction

ChipWhisperer

Your Equipment Box

ChipWhisperer Models

Installation

What is ChipWhisperer?

- ▶ Open source platform for learning about side-channel attacks on embedded devices and hardware security research.
- ▶ Makes validating side-channel resistance easier.
- ▶ Designed specifically for side-channel analysis (SCA) and fault injection (FI).
- ▶ Built and maintained by NewAE Technology Inc.

Where It Came From

- ▶ Started as academic research: Colin O'Flynn showed that synchronous sampling let a cheap capture board match power-analysis results from lab oscilloscopes costing far more.
- ▶ Hardware, firmware, and software all stayed open source, so the whole attack chain can be inspected and reproduced, not just the parts that are convenient to publish.
- ▶ It is now the de facto open platform in the SCA/FI research community: a large share of published attacks and countermeasure evaluations are run on ChipWhisperer hardware.

This is also why we use it here: it is the same tool the field actually uses, not a simplified teaching version of it.

Who Uses It, and Why

- ▶ **Chip and device vendors:** test their own products for side-channel and fault weaknesses in-house, before sending them for formal, far more expensive, security certification.
- ▶ **Security researchers:** use it as a common, reproducible platform, so results from different labs can be compared and built on.
- ▶ **Universities:** use it to teach side-channel and fault attacks hands-on, since it is affordable enough to put real hardware in front of every student.

Why ChipWhisperer?

- ▶ Affordable.
- ▶ SCA and FI in one platform.
- ▶ Accessible for beginners and experts alike.
- ▶ No need for an advanced research lab with expensive equipment.

Main Features

- ▶ Captures power consumption or EM traces from microcontrollers.
- ▶ Performs differential power analysis (DPA) and correlation power analysis (CPA).
- ▶ Supports clock and voltage glitching to induce faults.
- ▶ Integrates with Python and Jupyter notebooks for experiments.

How is it Used?

1. Load cryptographic firmware onto a target device.
2. Capture side-channel signals during cryptographic operations.
3. Analyze traces to extract secret keys or detect vulnerabilities.
4. Test countermeasures against SCA and FI attacks.

Why Not Just Use an Oscilloscope?

A normal oscilloscope samples on *its own* clock, so every trace starts at a slightly different point in the target's clock cycle. You then need a very high sampling rate just to make that misalignment small enough to correct for.

ChipWhisperer instead **samples synchronously with the target's clock**: it drives the target clock, or locks onto it, and takes a fixed number of samples per cycle.

Every trace is therefore aligned with every other trace by construction, and attacks succeed at far lower sampling rates than a conventional scope would need. This is the single most important reason the boards are cheap and the attacks still work.

Four Layers

- ▶ **Hardware:** the scope board and the target board
- ▶ **Firmware:** what runs on each of them
- ▶ **Software:** the Python library that drives the scope and talks to the target
- ▶ **Notebooks:** the Jupyter labs you actually work in

ChipWhisperer Hardware

- ▶ **Scope board** – used to mount side-channel attacks. Yours is a ChipWhisperer Husky.
- ▶ **Target board** – the device under test, the thing you attack.
- ▶ **Adapter board** – the CW313 carries the small CW312-style target boards and provides their supporting circuitry.

The IDC ribbon cable connects the two. Over it the scope supplies the target's clock, resets and programs the device, and exchanges serial messages with the firmware you flashed. Supplying the clock is what makes the traces align.

Firmware

- ▶ **Scope firmware** is not something we look at in this course. The FPGA Verilog and USB microcontroller C code live on GitHub for those who are interested: [newaetech/chipwhisperer-husky](#) for the Husky, and [newaetech/chipwhisperer-capture](#) for the older scopes.
- ▶ **Target firmware** is written in C, and you will read and modify it. It is what decides what the target does and therefore what you measure.

In the lab repository the target sources are under `firmware/mcu/`, and the notebooks flash a compiled `.hex` with `cw.program_target(scope, prog, path)`.

Software

- ▶ NewAE maintain an open source Python library that controls the capture hardware and communicates with the target board. This is the bulk of the code you interact with.
- ▶ You drive it from Jupyter notebooks, which is also where you plot your traces.
- ▶ The SCA Lab additionally uses [Lascar](#), a side-channel analysis framework, for trace alignment and analysis.

Contents

Introduction

ChipWhisperer

Your Equipment Box

ChipWhisperer Models

Installation

What is in the Box

- ▶ 1× ChipWhisperer Husky (the scope)
- ▶ 1× CW313 adapter board for CW312 targets
- ▶ 1× SAM4S target board
- ▶ 1× XMEGA target board
- ▶ 1× CW308T to CW312 adapter board
- ▶ 1× 20-pin IDC ribbon cable
- ▶ 1× USB-C cable with adapter

Appendix A of the assignment shows a picture of every item.

The Scope and the Targets

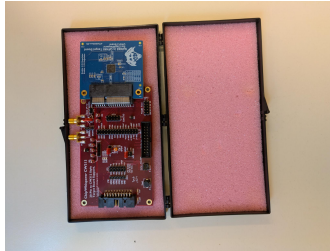


Figure: The ChipWhisperer Husky, the CW313 carrying the SAM4S target, and the XMEGA target board.

Which Board for Which Lab

- ▶ **SCA Lab:** AES on a microcontroller target, traces captured by you
- ▶ **Fault Lab:** glitching a microcontroller target until it misbehaves
- ▶ **RSA Lab: must** use the XMEGA target with the ChipWhisperer Husky
- ▶ **ECC Lab:** no hardware, it uses pre-recorded traces from a CW305 FPGA

Each notebook states the target it expects. If a capture looks weird, the first thing to check is if the right board is mounted and the ribbon cable is seated.

Rules for the Boxes

- ▶ The boxes live in the **CRYPTO-LAB**, Electro **A175**, and there are 42 of them
- ▶ Take the box whose **number matches your group number** in Canvas
- ▶ Every student in the course has card access, and the room is never booked for teaching, so you can work there whenever it suits you
- ▶ You may take the box home, but the *same* box must come back by the end of the semester
- ▶ Report anything broken or missing, so the next group gets working equipment

Use the CRYPTO-LAB computers or your own laptop: the software is not installed in other computer labs, and you may not be allowed to install it there.

Contents

Introduction

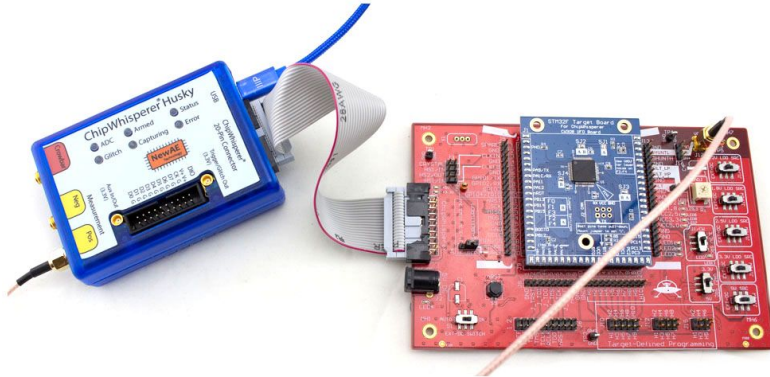
ChipWhisperer

Your Equipment Box

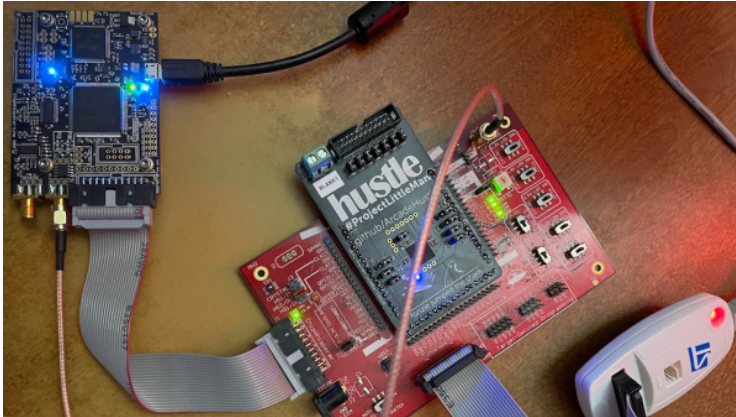
ChipWhisperer Models

Installation

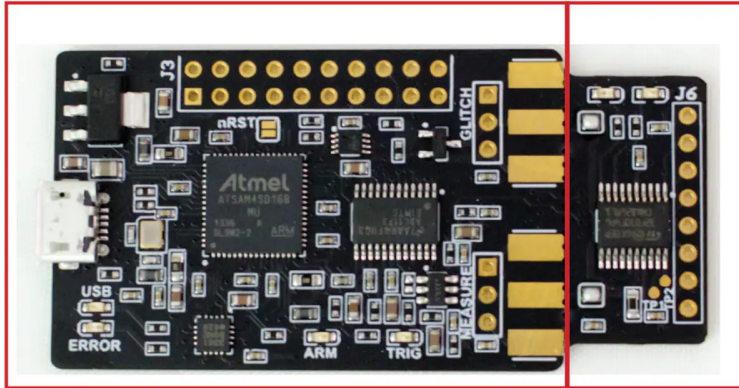
ChipWhisperer Husky



ChipWhisperer Lite



ChipWhisperer Nano



CAPTURE Section

TARGET Section

What the Differences Mean

	Husky	Lite	Nano
ADC	12-bit, 200 MS/s	10-bit, 105 MS/s	8-bit, 20 MS/s
Samples per capture	up to 204 796	24 573	50 000
Streaming capture	yes	no	no
Analog input	differential	single-ended	fixed gain
Clock glitching	yes	yes	no (voltage only)
Analog / SAD trigger	yes	no	no

More bits and more samples mean less averaging to see the same signal, and the Husky's glitch resolution does not depend on the target clock. Use Husky.

Contents

Introduction

ChipWhisperer

Your Equipment Box

ChipWhisperer Models

Installation

Three Routes

- ▶ “Vanilla” installation: download everything manually, optionally inside a Python virtual environment.
- ▶ The ChipWhisperer installer (Windows only).
- ▶ Virtual machine with Vagrant and VirtualBox.

The virtual machine is the **recommended** route if you work on your own laptop, and especially on Windows. On the CRYPTO-LAB machines everything is already in place.

Prerequisites

- ▶ Git
- ▶ Compilers and build tools for the target firmware
- ▶ Nice to have: a Python version manager, `pyenv` or `anaconda`
- ▶ For the virtual machine: [Vagrant](#) and [VirtualBox with the Extension Pack](#)

NewAE Installation Guides

Start from the overview, then pick your platform:

- ▶ chipwhisperer.readthedocs.io/en/latest/getting-started.html
- ▶ Windows Linux macOS Virtual machine

Follow these guides, but with one change, on the next slide.

Use the Course Repository

Wherever the NewAE guide tells you to clone `newaetech/chipwhisperer`, clone **our** repository instead:

git.ntnu.no/ie-iik/chipwhisperer-lab-ttm4205

It contains everything the lab needs, including the target firmware and the graded notebooks. The assignment files are in the `jupyter/lab` sub-folder.

This is the single most common thing to get wrong: with NewAE's repository, the notebooks in the assignment simply are not there.

Virtual Machine with Vagrant

Our Vagrant configuration: git.ntnu.no/ie-iik/chipwhisperer-v6-vagrant

1. Install Vagrant and VirtualBox with the Extension Pack.
2. Clone the configuration repository.
3. Run `vagrant up` in a terminal or PowerShell inside the `chipwhisperer-v6-vagrant` folder.

The Extension Pack is what lets the virtual machine see the Husky over USB, so do not skip it.

First Notebook: `Setup.ipynb`

Before any of the graded work, run `Setup.ipynb`.

It checks that your installation works and that the ChipWhisperer `Husky` is detected, and it is the fastest way to find out whether your environment is correct.

The ECC Lab has its own preparation notebook, `ECC_lab_setup.ipynb`, which you run once before 5-ECC_lab.

Notebooks are not graded. The report is.

When Something Breaks

- ▶ **Lascar missing:** `pip install lascar` inside your Python environment
- ▶ **File or folder not found:** usually a path fixed in a later commit
- ▶ **Getting the fixes:** `git pull --autostash` in the lab folder
- ▶ **Dynamic Time Warping is slow:** fewer traces, smaller radius, or lower `scope.adc.samples`

Appendix B of the assignment lists these and their solutions. Beyond that: ask the teaching assistants in the exercise classes or lab sessions, bring your setup and show what you tried, or ask on the Canvas forum.

Questions?

Let's install ChipWhisperer together!