



This work is distributed  
under the CC BY-SA license,  
unless stated otherwise.

# HardwareSecurity

## 8. SCA (tutorial)

Ján Jančár

Summer 2026

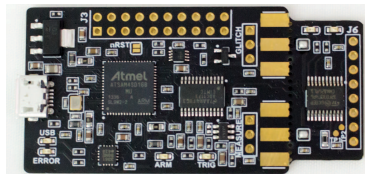
# Today

- ChipWhisperer hardware
- Breaking a password check with SPA
- Fixing a password check
- (Bonus) DPA

# ChipWhisperer

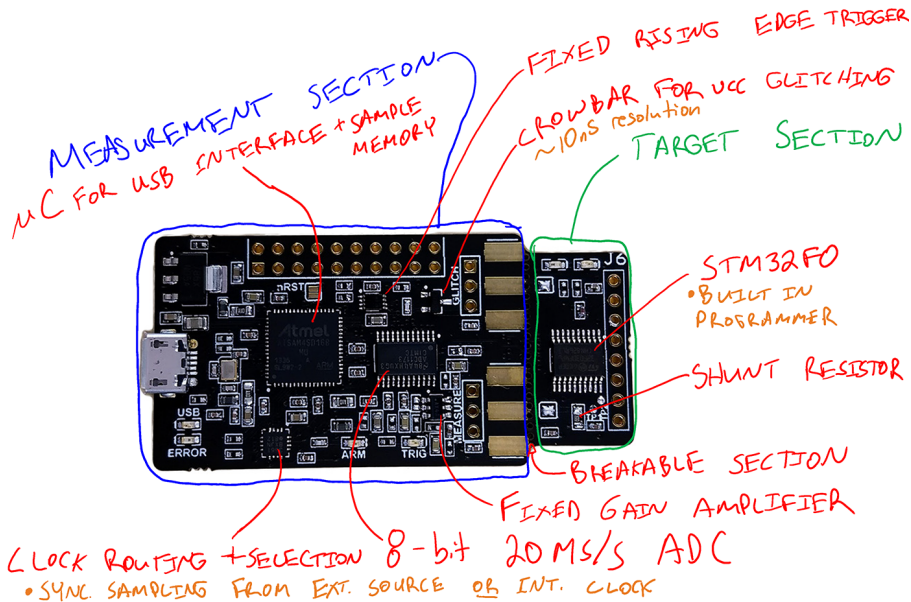
- **Open-source** toolchain for research and education by [NewAE](#)
  - Side-channel attacks
  - Fault injection
- **Hardware**
  - Schematics & PCB layout
- **Firmware**
  - Capture
    - ▶ USB controller + FPGA for captures
  - Target
    - ▶ Example targets
- **Software**
  - Python API for capture & analysis

# ChipWhisperer-Nano



- Low-cost power analysis and fault injection setup (50€)
- Capture
  - ATSAM4SD168 (microcontroller + memory)
  - ADC1173CIMTCX (Analog-to-Digital converter)
- Target
  - STM32F030F4P6 (microcontroller)

# ChipWhisperer-Nano



<https://chipwhisperer.readthedocs.io/en/latest/Targets/CW1101%20Target%20Board.html>

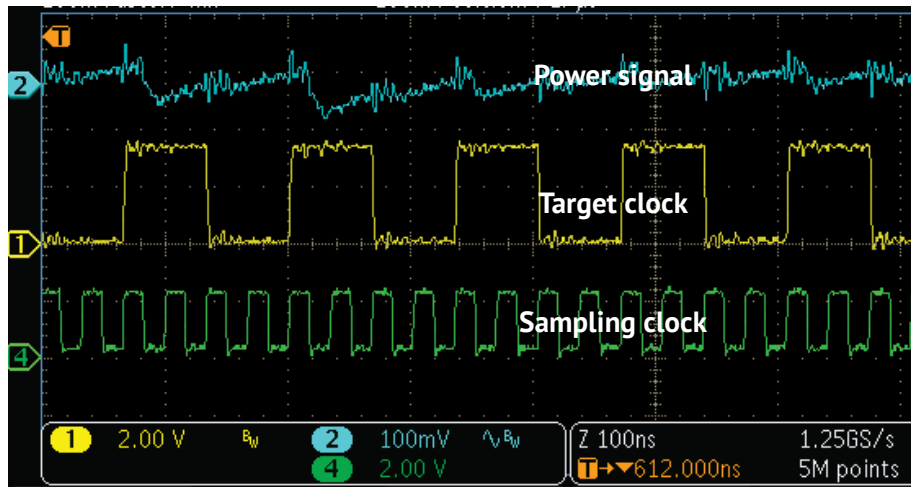
# ChipWhisperer-Nano

## Specs

| Feature                     | Notes/Range                                                            |
|-----------------------------|------------------------------------------------------------------------|
| ADC Specifications          | 8-bit ADC, 20 MS/s maximum sample rate.                                |
| ADC Sample Clock Source     | Selectable between internal generator or external input.               |
| Analog Input                | AC-Coupled, fixed gain.                                                |
| GPIO Types                  | Serial, clock, logic line (i.e., for reset pin). Fixed pin functions.  |
| GPIO Voltage                | 3.3V.                                                                  |
| Clock Options               | 3.75 MHz, 7.5 MHz, 15 MHz, 30 MHz , 60 MHz                             |
| Clock Output Type           | Generated by microcontroller, clock only (no clock glitching support). |
| Trigger Type (ADC + Glitch) | Rising edge only.                                                      |
| Glitch Width (min)          | ~20nS (depends on cabling used for routing glitch output).             |
| Glitch Offset               | ~200nS jitter, adjustable in 10nS increments.                          |
| Voltage glitch type         | Low-power crowbar circuitry.                                           |
| Crowbar pulse current       | 4A.                                                                    |
| USB Interface               | Custom USB firmware (full-speed USB 2.0 device).                       |
| Sample Buffer Size          | 50 000.                                                                |
| Target Device               | STM32F030F4P6 or STM32F070                                             |
| Programming Protocols       | STM32Fx Bootloader                                                     |

# ChipWhisperer-Nano

## Clock-synchronous sampling



**Allows lower sampling rates**

# Setup

- Content
  - **task.ipynb**: Task Jupyter notebook
  - **solution.ipynb**: Solution Jupyter notebook
  - **basic-passwdcheck/**: Target code
- Task
  - Run jupyter notebook and solve the task notebook
  - Requires a ChipWhisperer install

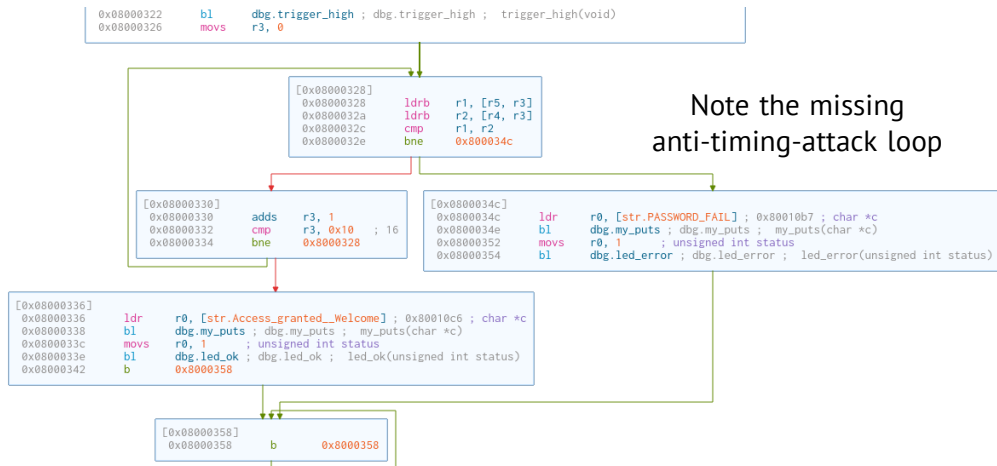
# Target

```
uint8_t passbad = 0;
trigger_high();

uint8_t i;
for (i = 0; i < MAX_PWLEN; i++) {
    if (correct_passwd[i] != passwd[i]) {
        passbad = 1;
        break;
    }
}

if (passbad) {
    // Finish the loop here to stop those pesky timing attacks
    for (; i < MAX_PWLEN; i++)
        ;
    my_puts("PASSWORD FAIL\n");
    led_error(1);
} else {
    my_puts("Access granted, Welcome!\n");
    led_ok(1);
}
```

## Target



# Task

- 1 Examine leakage of two different characters
- 2 Examine leakage of all one-character passwords
- 3 Automate one-character attack
- 4 Full attack against 12 characters (SPA)
- 5 Fix the leak
- 6 Examine the fix
- 7 (Bonus) Implement DPA-like attack against the fix

# Tools

- **matplotlib**: Plotting
- **numpy**: Trace manipulation
- **arm-none-eabi-gcc**: Building and binary examination

# Task

## Solution - SPA

```
guessed_pw = ""

for _ in range(12):
    ref_trace = capture_trace(guessed_pw + "\x00")
    for c in 'abcdefghijklmnopqrstuvwxyz0123456789':
        trace = capture_trace(guessed_pw + c)
        diff = np.sum(np.abs(trace - ref_trace))

        if diff > 40.0:
            guessed_pw += c
            print(guessed_pw)
            break
```

# Task

## Solution - Fix

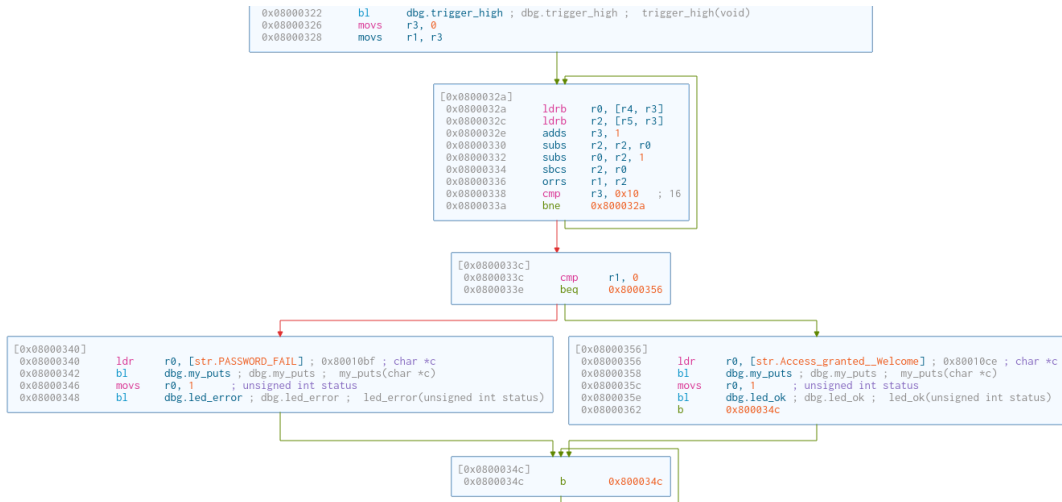
```
uint8_t passbad = 0;
trigger_high();

uint8_t i;
for (i = 0; i < MAX_PWLEN; i++) {
    if (correct_passwd[i] != passwd[i]) {
        passbad = 1;
        //break;
    }
}

if (passbad) {
    // Finish the loop here to stop those pesky timing attacks
    for (; i < MAX_PWLEN; i++)
        ;
    my_puts("PASSWORD FAIL\n");
    led_error(1);
} else {
    my_puts("Access granted, Welcome!\n");
    led_ok(1);
}
```

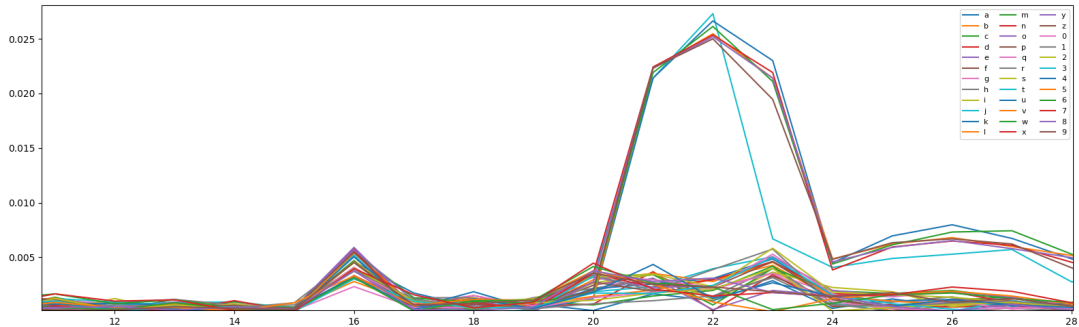
# Task

## Solution - Fix



# Task

## Solution - DPA



# Takeaways

- What are **you** taking away from this tutorial?
- What surprised you?