ChipWhisperer

Smita Das

Secured Embedded and Architecture Laboratory (SEAL)

Department : Computer Science and Engineering

Bootcamp on Embedded Security and Trust (BEST 2025)

Overview

- Side Channel Infrastructure
- ChipWhisperer Setup
- Trace collection using ChipWhisperer-Lite
- CPA - Basics and implementation

Attack Model / Assumptions

- Consider a device capable of implementing the cryptographic function
- The key is usually stored in the device and protected
- Modern cryptography is based on Kerckhoffs's assumption.
- Any data other than the key is Public
- Attacker only needs to extract the key

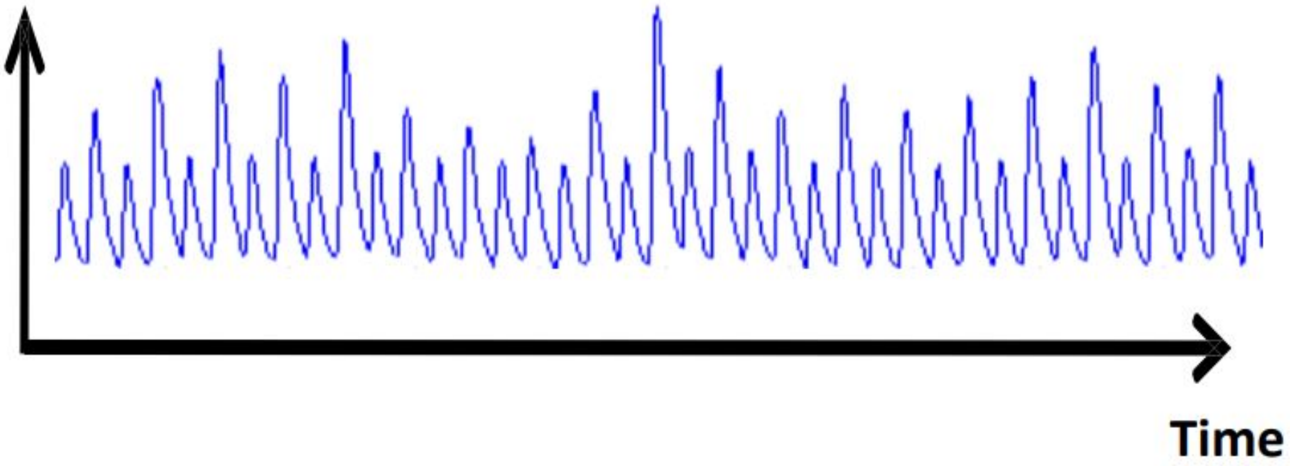
Attack Phases

- **Interaction phase**: interact with the hardware system under attack and obtain the physical characteristics of the device
- **Analysis phase**: analyze the gathered information to recover the key

Side Channel Analysis Infrastructure

Side-Channel attacks aim at side-channel inputs and outputs, bypassing the theoretical strength of cryptographic algorithms

- **Power Consumption :** Logic circuits typically consume different amounts of power based on their input data

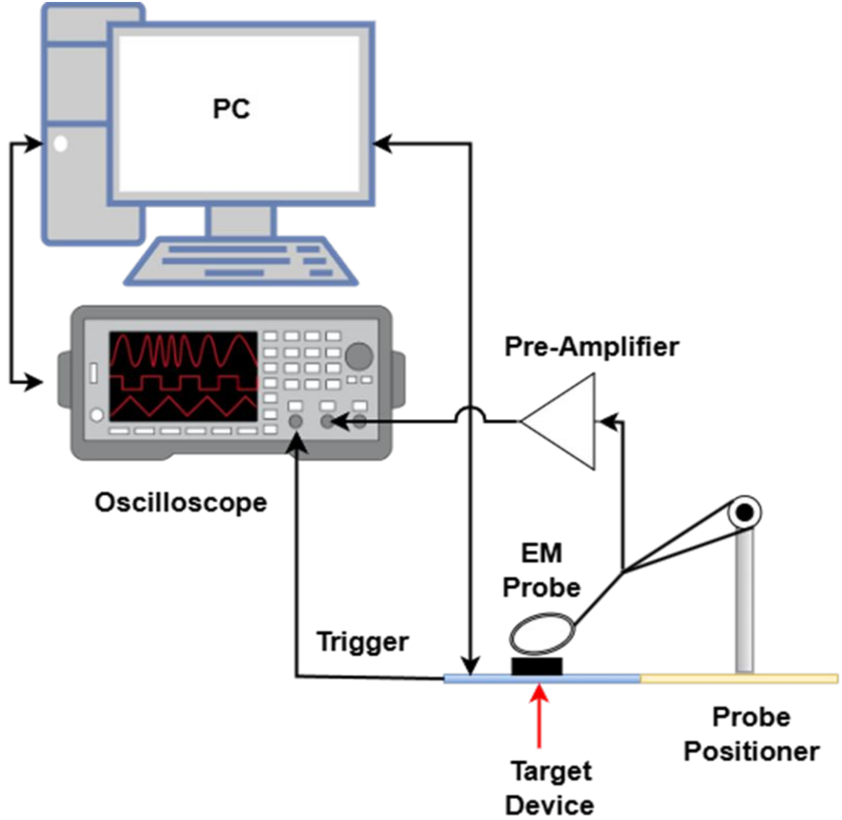


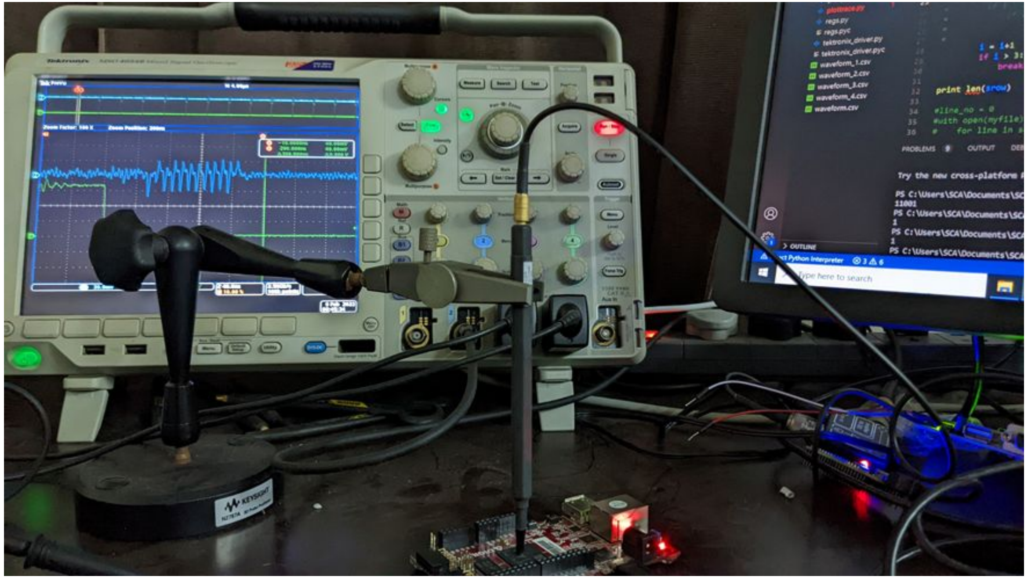
Side Channel Analysis Infrastructure

Electro-Magnetic:

- Physically measure the electromagnetic emanations from an electronic device and use analytical methods and leakage models to steal information from the data.
- EM attacks are non-invasive
- This makes EM attacks powerful because they are easy for the attacker to perform. It only requires the use of a near-field probe and an oscilloscope.

Existing Side Channel Analysis Infrastructure



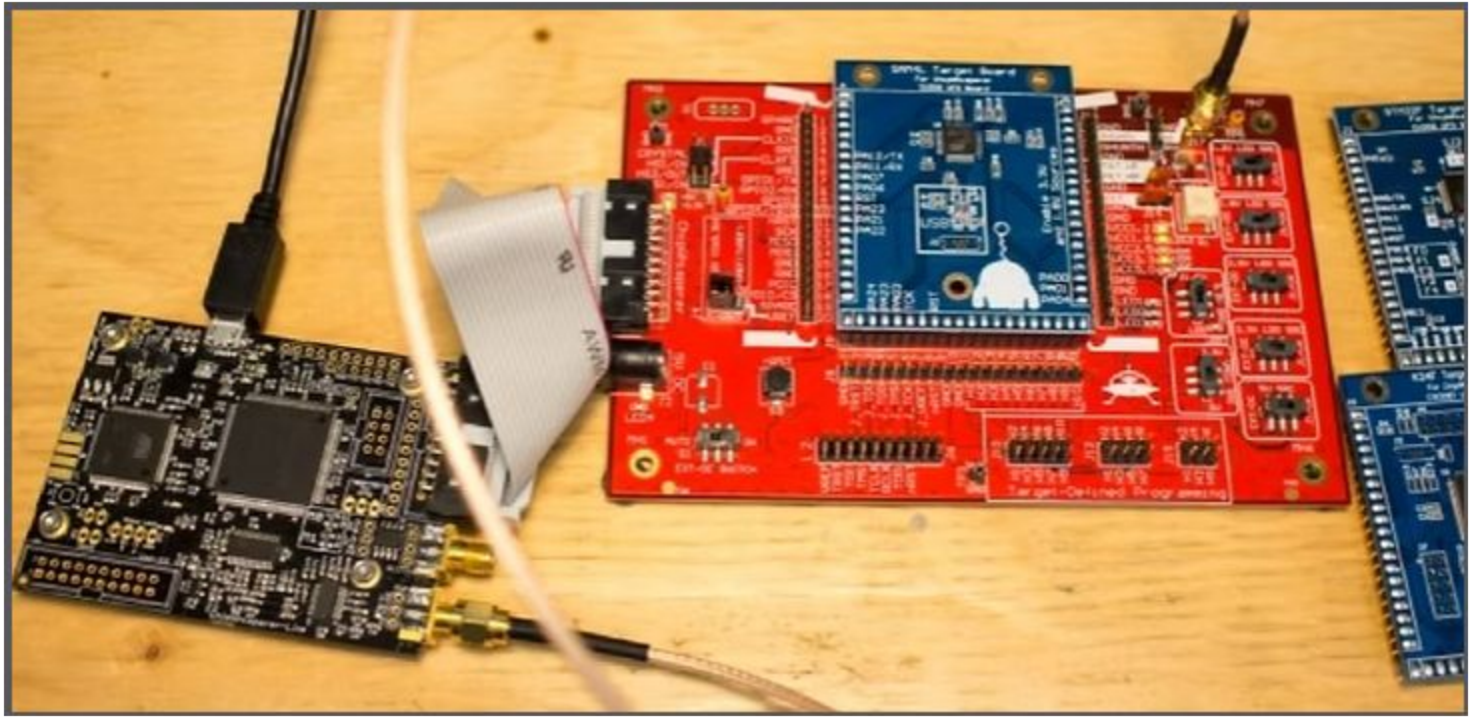


Process description:

- The Target Device executes some cryptographic algorithm
- The radiations emanated by the target device is collected using the EM probe
- The EM feeds the signal to an oscilloscope via a pre-amplifier
- The data from the oscilloscope is further transferred on the host PC for further analysis

Can we bring down these large setup into an integrated device?

ChipWhisperer



Capture Boards

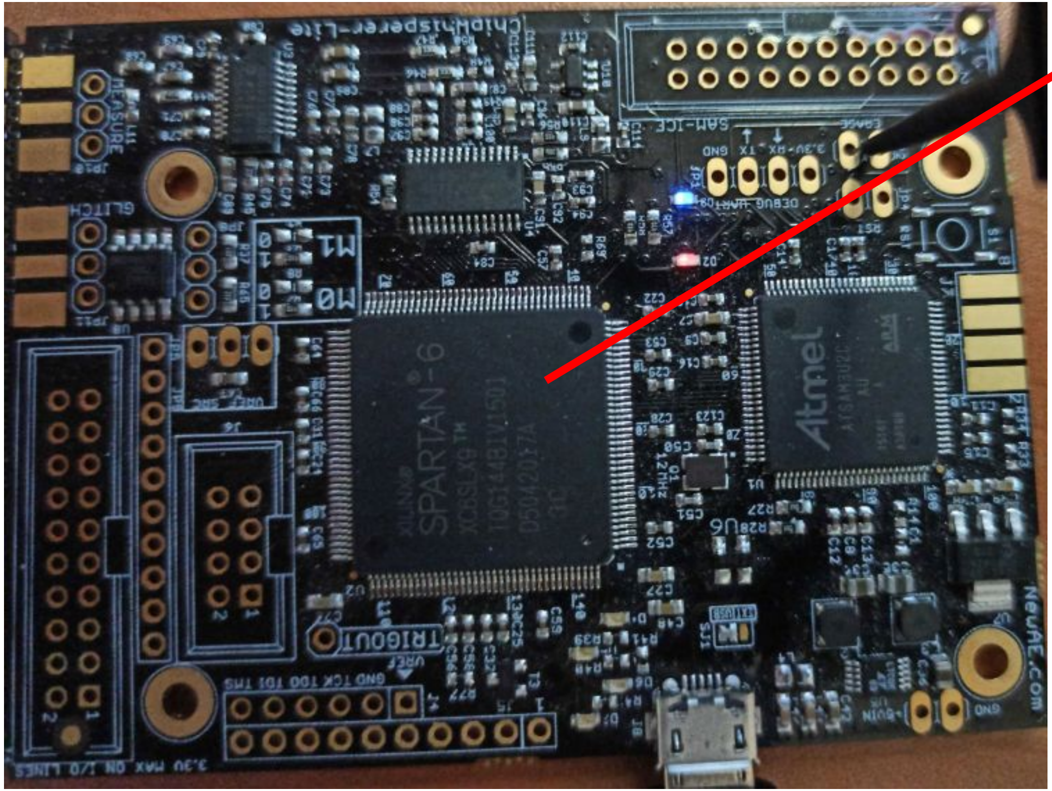
Five capture side hardware devices:

- ChipWhisperer-Husky,
- ChipWhisperer-Husky-Plus,
- CW1200 ChipWhisperer-Pro (CWPro),
- CW1173 ChipWhisperer-Lite (CWLite), and
- CW1101 ChipWhisperer-Nano (CWNano)

ChipWhisperer-Lite

- The ChipWhisperer-Lite typically comes with two main parts:
  - **a multi-purpose power analysis capture instrument,**
  - **a target board**
- Features, such as SMA connectors for trigger input and output, allowing for easy interfacing with lab equipment.
- Includes 8-bit Atmel XMEGA and 32-bit STM32F3 target devices

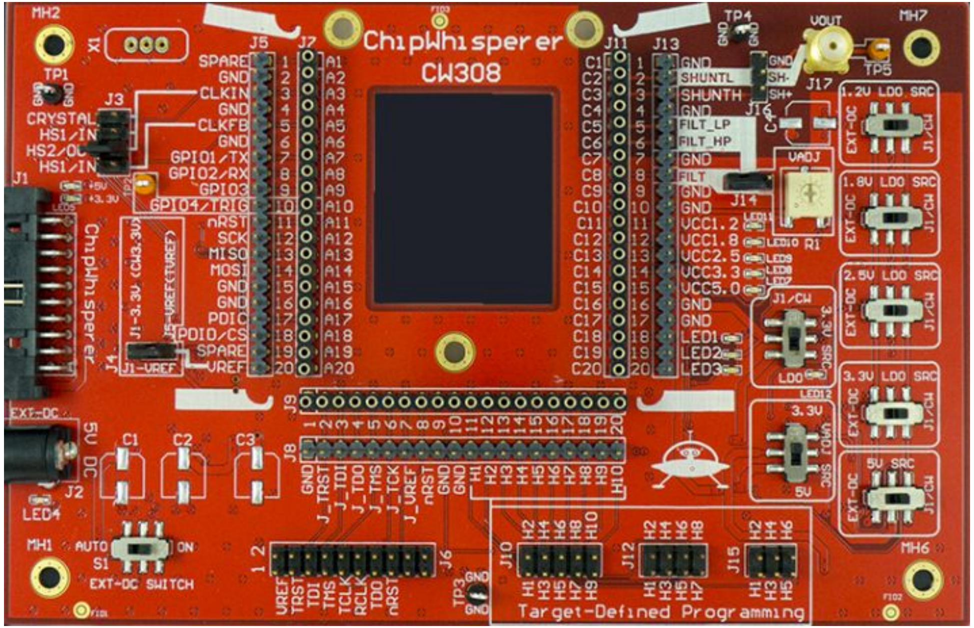
ChipWhisperer-Lite



Handles **triggering** for side-channel and fault injection attacks.

ChipWhisperer-Lite UFO board

Universal target board platform, providing power, clock, and trigger control for different target microcontrollers.

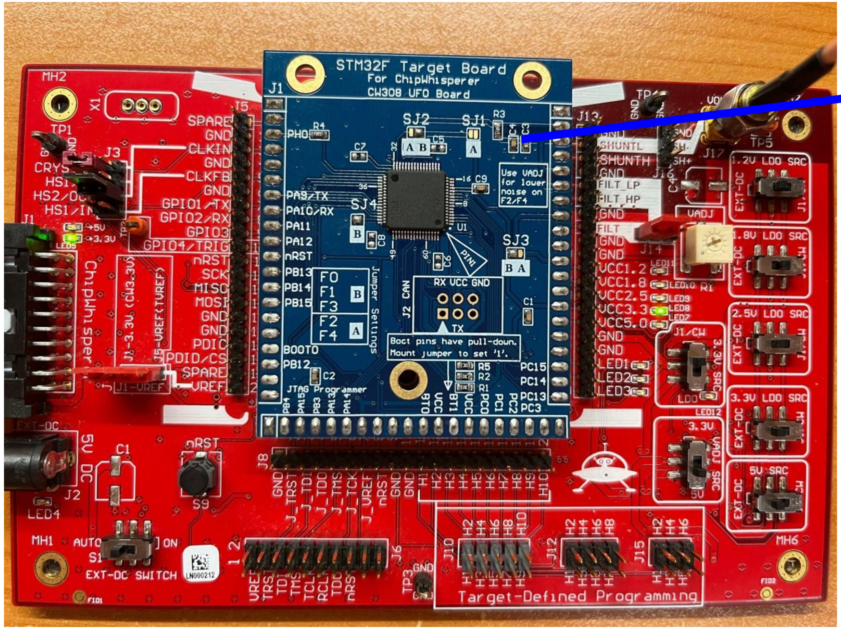


ChipWhisperer-Lite UFO board

- It is the starting point for side-channel power analysis attacks when combined with a ChipWhisperer Capture solution.
- The CW308 puts all the standard requirements onto one board (such as power supplies, oscillators) to make super-simple target victim boards.

ChipWhisperer-Lite UFO board

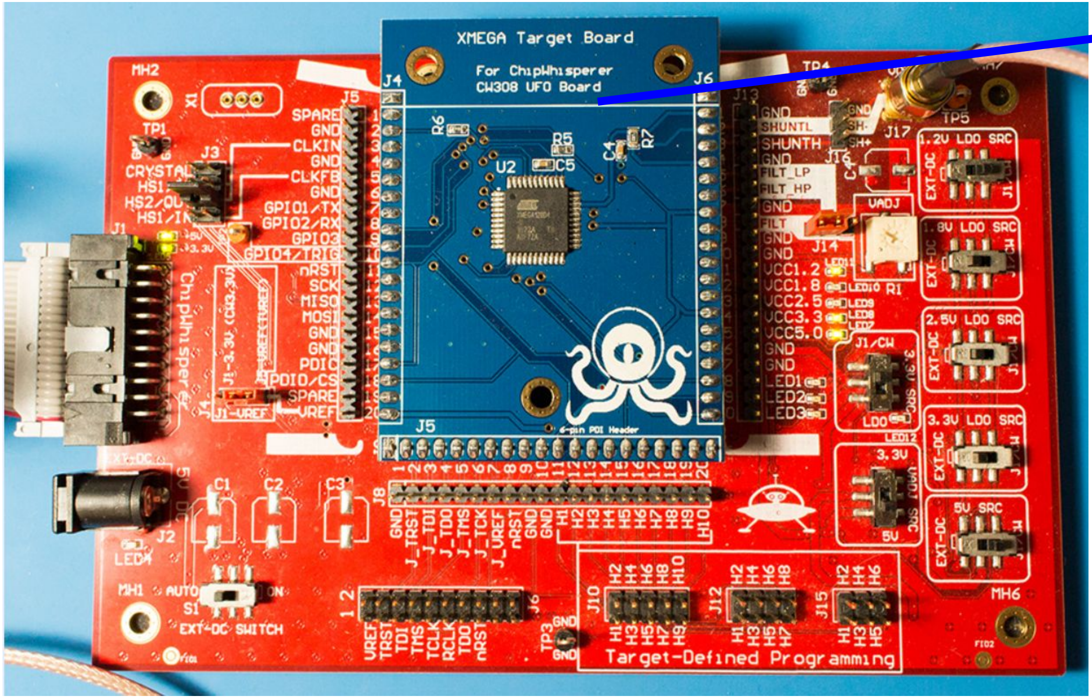
It is the main-board for attacking all sorts of embedded targets.



STM32F
Target

ChipWhisperer-Lite UFO board

It is the main-board for attacking all sorts of embedded targets.



CW308T-XMEGA Target

CW308 Details

The CW308 comes with the following parts

- CW308 Main Board
- CW308T-XMEGA Target Board (Atmel 8-bit microcontroller)
- CW308T-STM32F3 Target Board (ARM Cortex M3)
- CW308T Prototyping Boards (2.54mm prototyping board)
- NPCB-CW308T-STM32F Blank PCB (Fits STM32F0, F1, F2, F3, F4)
- Target Removal Tool
- 7.37 MHz crystal
- 8x jumper wires
- 30 cm SMA cable

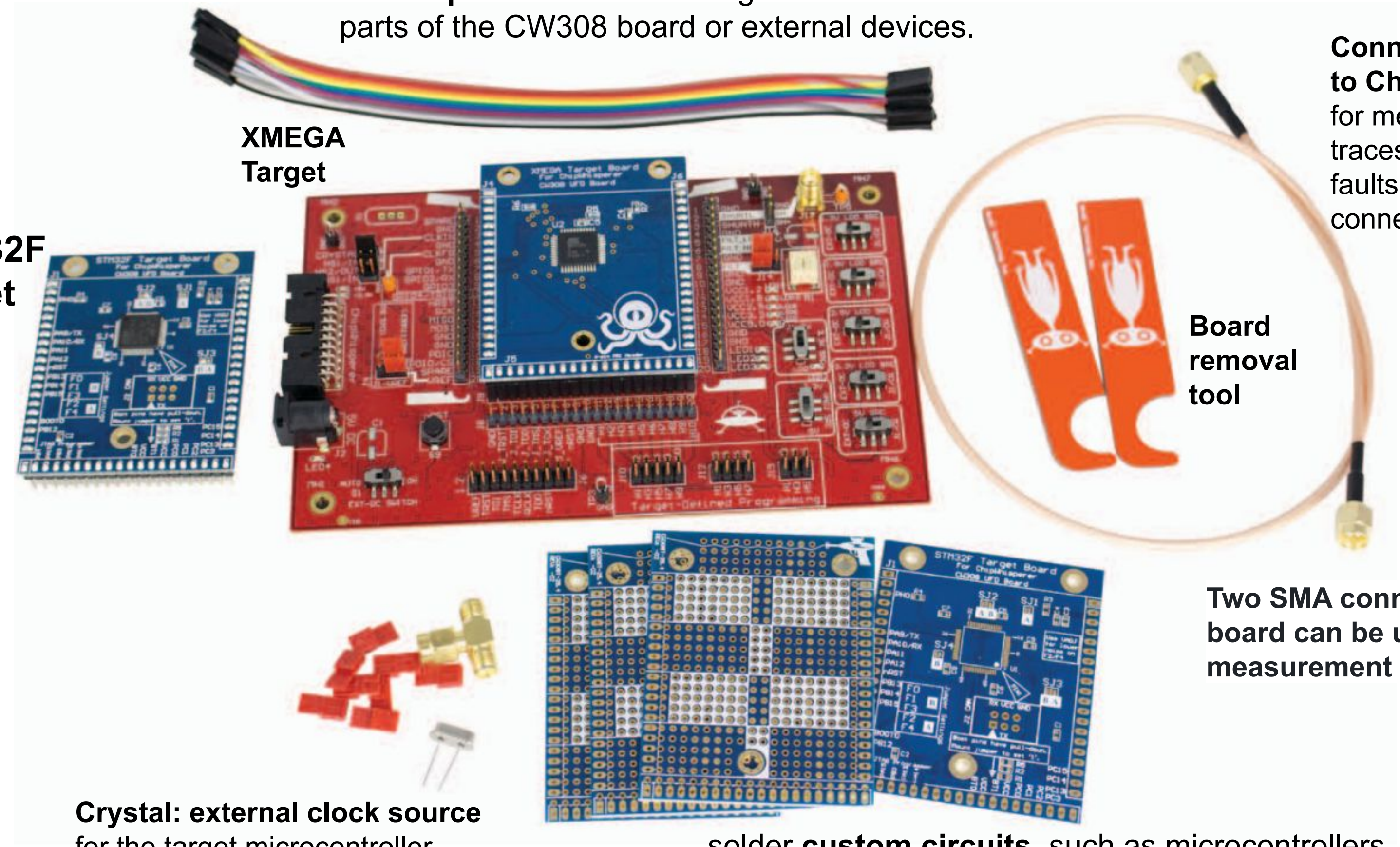
CW308 Details

8x Jumper Wires:connect signals between different parts of the CW308 board or external devices.

Connect the CW308 to ChipWhisperer for measuring power traces or injecting faults- SMA connector

XMEGA Target

STM32F Target



Board removal tool

Two SMA connectors on the board can be used for power measurement or glitch input

Crystal: external clock source for the target microcontroller

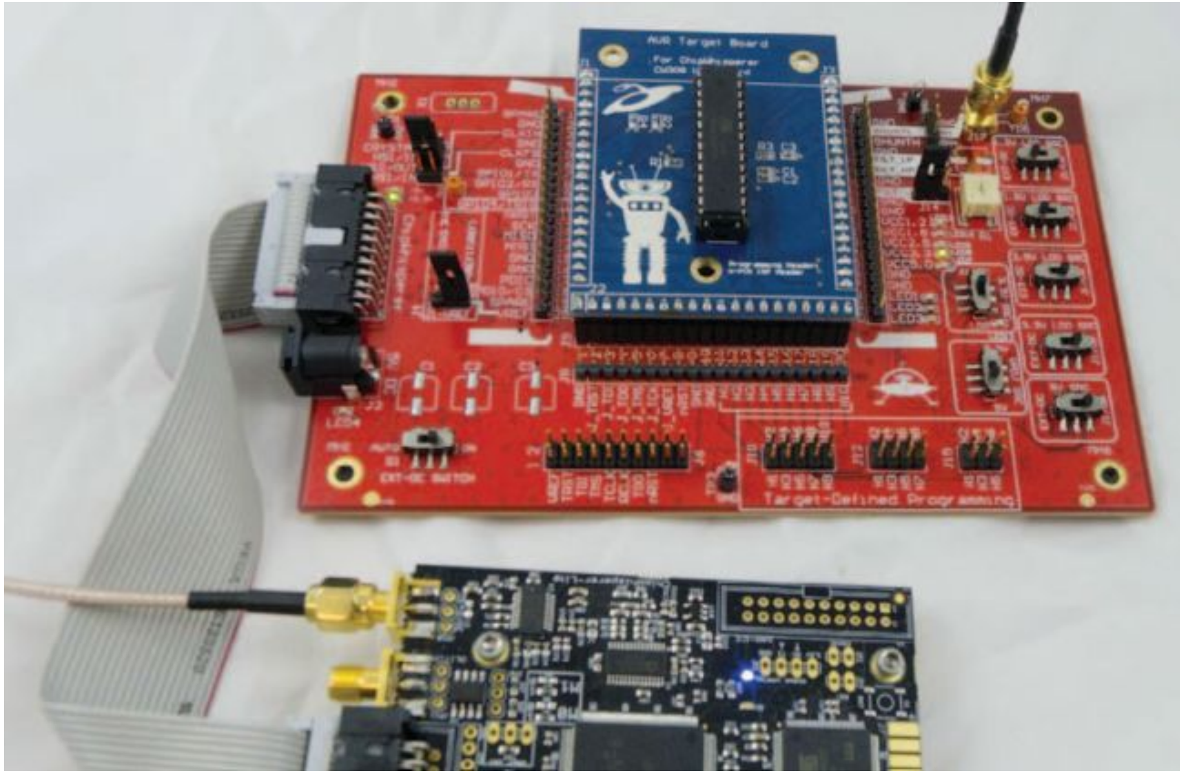
solder **custom circuits**, such as microcontrollers,

ChipWhisperer Capture

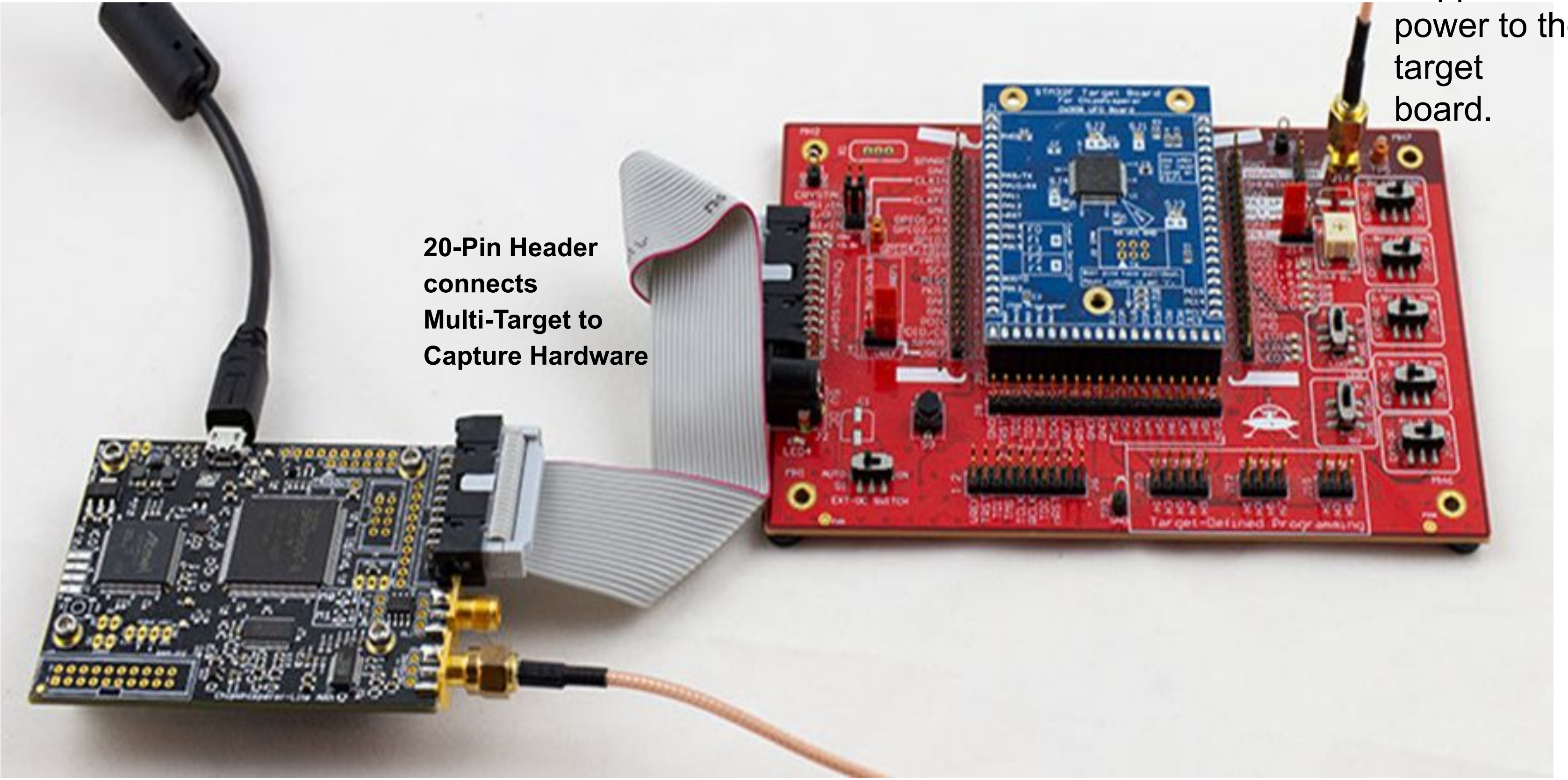
The capture hardware provides

- power,
- serial communication,
- clock,
- shunt resistor monitoring,
- voltage fault injection capability.

Select devices (XMEGA, and STM32Fx) targets can be programmed from the ChipWhisperer, avoiding the need for an external programmer.



ChipWhisperer-Lite



Supplies power to the target board.

Scope API

- Control the capture/glitch portion of the ChipWhisperer device.
- Create a scope object via the **chipwhisperer.scope()** function, which will connect to a ChipWhisperer device and return a scope object of the correct type

```
import chipwhisperer as cw
scope = cw.scope()
```

There are currently two types of scopes:

- OpenADC Scope (Lite, Pro, Husky)
- ChipWhisperer Nano Scope (Nano)

This function allows any type of scope to be created.

By default, the object created is based on the attached hardware (OpenADC for CWLite/CW1200, CWNano for CWNano)Scope Types:

scopes.OpenADC (Pro and Lite)

scopes.CWNano (Nano)

OpenADC scope object: ChipWhisperer scope submodules

This class contains the public API for the OpenADC hardware, including the ChipWhisperer Lite. To connect to one of these devices

```
import chipwhisperer as cw
scope = cw.scope(scope_type=cw.scopes.OpenADC)
```

- **scope.gain**
- **scope.adc**
- **scope.clock**
- **scope.io**
- **scope.trigger**
- **scope.glitch (Lite/Pro)**
- **scope.default_setup**
- **scope.con**
- **scope.dis**
- **scope.arm**
- **scope.get_last_trace**

Setup scope to begin capture/glitching when triggered.

OpenADC scope object: ChipWhisperer scope submodules

Scope Attributes

- **scope.gain**
  - Controls the gain (amplification) of the signal captured by the ADC.
- **scope.adc**
  - Represents the analog-to-digital converter (ADC) settings.
  - Includes parameters like the sampling rate, offset, and triggering settings.
- **scope.clock**
  - Handles clock settings for the target and glitch module.
- **scope.io**
  - Manages I/O pins on the ChipWhisperer device.
- **scope.trigger**
  - Configures the trigger module, which determines when the scope starts capturing traces.
  - Triggers can be based on rising edges, falling edges, or specific signal levels.
- **scope.glitch**
  - Manages glitching settings, including width, offset, and trigger conditions.
  - Used for voltage or clock glitching attacks.

OpenADC scope object: ChipWhisperer scope submodules

Scope Methods

- **scope.default_setup()**
  - Resets the scope to default settings.
- **scope.con()**
  - Connects to the ChipWhisperer hardware.
- **scope.dis()**
  - Disconnects from the ChipWhisperer hardware.
- **scope.arm()**
  - Arms the scope to start capturing power traces once a trigger event occurs.
- **scope.get_last_trace()**
  - Retrieves the last captured power trace after a successful capture.

OpenADC scope object: ChipWhisperer scope submodules

```
scope.default_setup()
```

Scope Default setup

- Sets the scope gain to 45dB
- Sets the scope to capture 5000 samples
- Sets the scope offset to 0 (i.e., it will begin capturing as soon as it is triggered)
- Sets the scope trigger to rising edge
- Outputs a 7.37MHz clock to the target on HS2
- Clocks the scope ADC at 4*7.37MHz. Note that this is *synchronous* to the target clock on HS2
- Assigns GPIO1 as serial RX (allowing data receiving from the target)
- Assigns GPIO2 as serial TX (allowing data transmission to the target)

Building and Uploading Firmware

```
%%bash
cd ../firmware/mcu/simpleserial-base/
make PLATFORM= CRYPTO_TARGET=NONE
```

```
make PLATFORM=CWLITEARM CRYPTO_TARGET=TINYAES128C
```

// Uses TinyAES for encryption

- **Attacking a target** :

  Get some firmware built and uploaded onto it.

- Fill in your platform, re-run the *build* command, and firmware should be successfully built.

  `PLATFORM=CW308_STM32F3`

Communication with the Target

Done through the `SimpleSerial target` object.

Grouped into two categories:

1. Raw serial via `target.read()`, `target.write()`, `target.flush()`, etc.
2. SimpleSerial commands via `target.simpleserial_read()`, `target.simpleserial_write()`, `target.simpleserial_wait_ack()`, etc.

The firmware we uploaded uses the simpleserial protocol

Communication with the Target

- **SimpleSerial** is a lightweight communication protocol used by **ChipWhisperer** to send and receive data between a **host PC** and a **target device** (XMEGA or STM32 microcontroller)
- Uses **UART (serial communication)** for data exchange
- HEX File Generated After Make Firmware
- Contains **machine instructions** for the microcontroller

Communication with the Target

Encryption Application

The encryption application provides a simple method to encrypt a plaintext into a ciphertext. The following operations are performed:

1. Load encryption key with '**k**' command (

   Eg: **k2b7e151628aed2a6abf7158809cf4f3c\n** sets key to **2b7e151628aed2a6abf7158809cf4f3c**).
2. Set input text to encryption module with '**p**' command.

   Device encrypts input text, and toggles the I/O trigger line during the encryption operation.
3. The ciphertext is returned with the '**r**' command.

Capturing Traces

- Arm the ChipWhisperer with `scope.arm()`
    - It will begin capturing as soon as it is triggered
- `scope.capture()`
    - read back the captured power trace, blocking until either ChipWhisperer is done recording, or the scope times out.
- Read back the captured power trace with `scope.get_last_trace()`

Capturing Traces

simpleserial_base will trigger the ChipWhisperer when we send the '**p**' command. Try capturing a trace now

```
scope.arm()
target.simpleserial_write('p', msg)
## fill in the rest...
```

ChipWhisperer also has a capture_trace() convenience function that:

1. Optionally sends the '**k**' command
2. Arms the scope
3. Sends the '**p**' command
4. Captures the trace
5. Reads the return '**r**' as response
6. Returns a Trace class that groups the trace data, '**p**' message, the '**r**' response, and the '**k**' key.

Disconnect from the hardware

```
scope.dis()
target.dis()
```

Useful Links

https://rtfm.newae.com/Capture/ChipWhisperer-Lite/

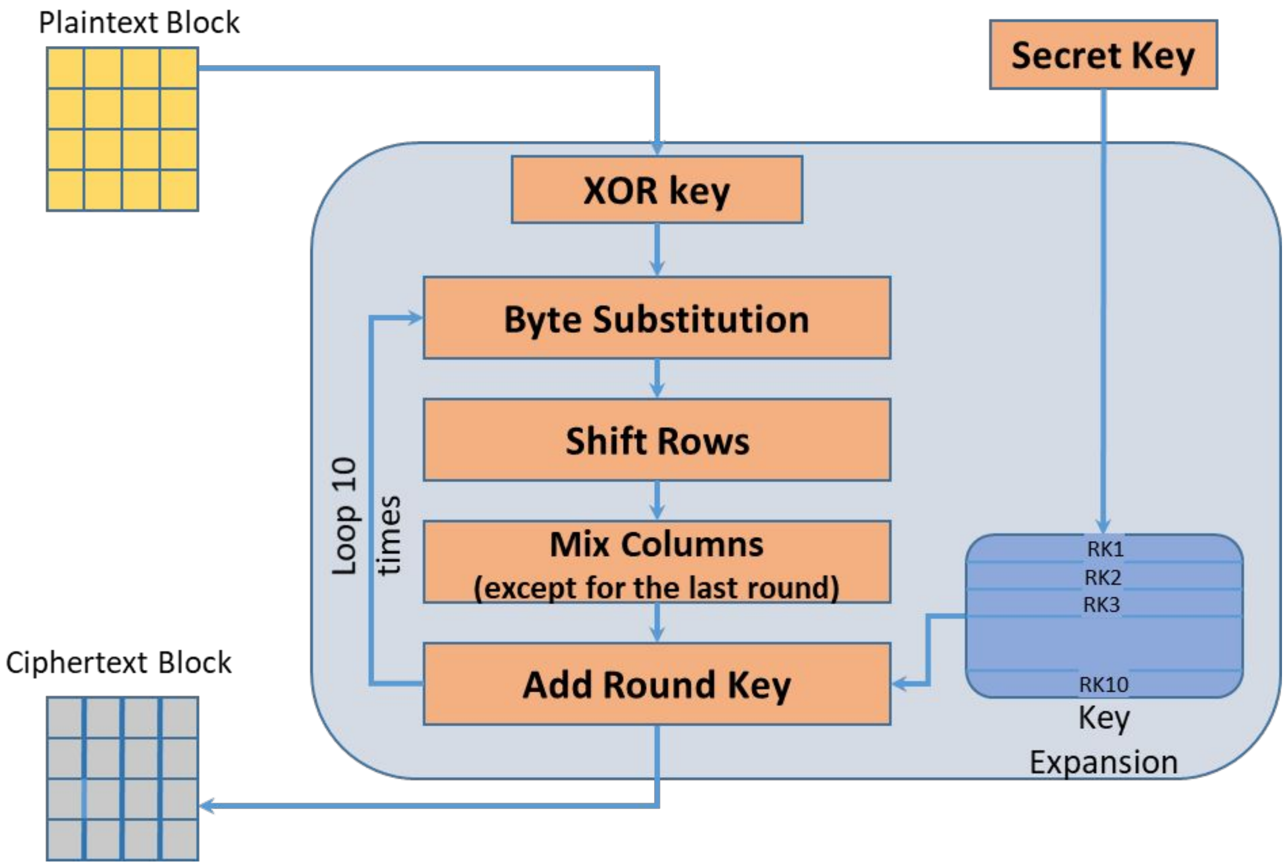
https://www.newae.com/products/nae-cw1173

https://chipwhisperer.readthedocs.io/en/latest/windows-install.html

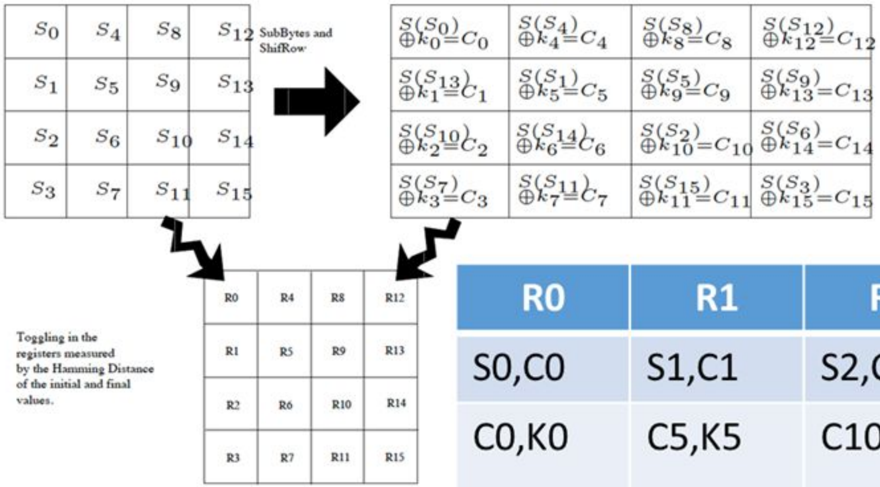
- Anaconda
- ARM GCC Compiler
- Visual Studio Code

Let’s try to capture some trace and perform Correlation Power Attack (CPA)

AES-128 Encryption



Calculating the Hypothetical Power



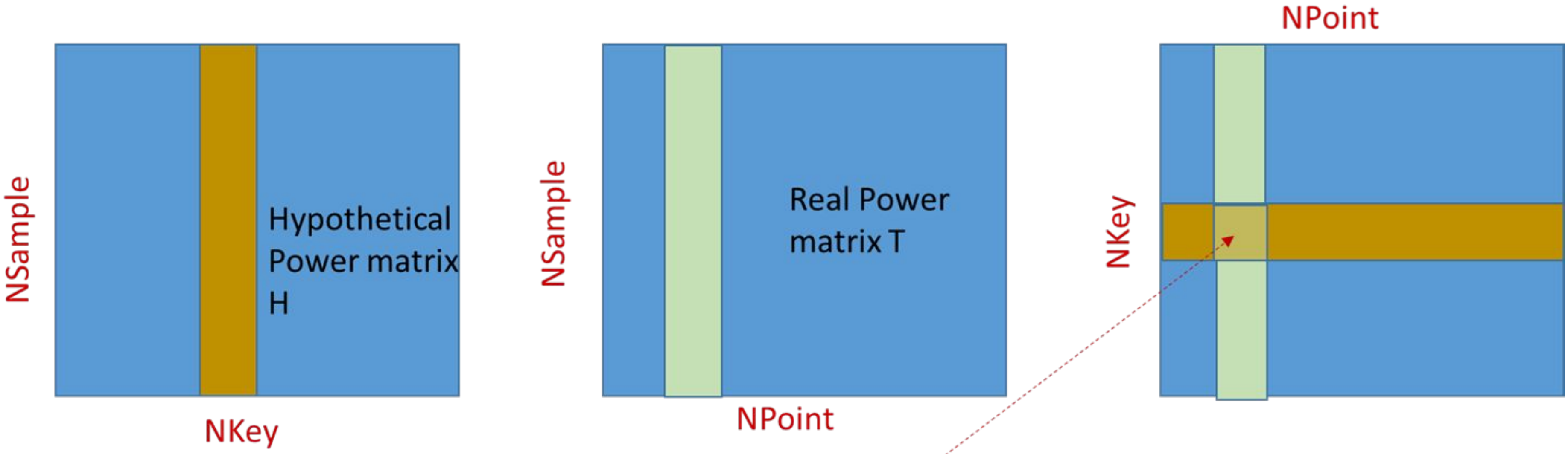
R0	R1	R2	R3	R4	R5	R6	R7
S0,C0	S1,C1	S2,C2	S3,C3	S4,C4	S5,C5	S6,C6	S7,C7
C0,K0	C5,K5	C10,K10	C15,K15	C4,K4	C9,K9	C14,K14	C3,K3

R8	R9	R10	R11	R12	R13	R14	R15
S8,C8	S9,C9	S10,C10	S11,C11	S12,C12	S13,C13	S14,C14	S15,C15
C8,K8	C13,K13	C2,K2	C7,K7	C12,K12	C1,K1	C6,K6	C11,K11

Computing the Correlation Matrix

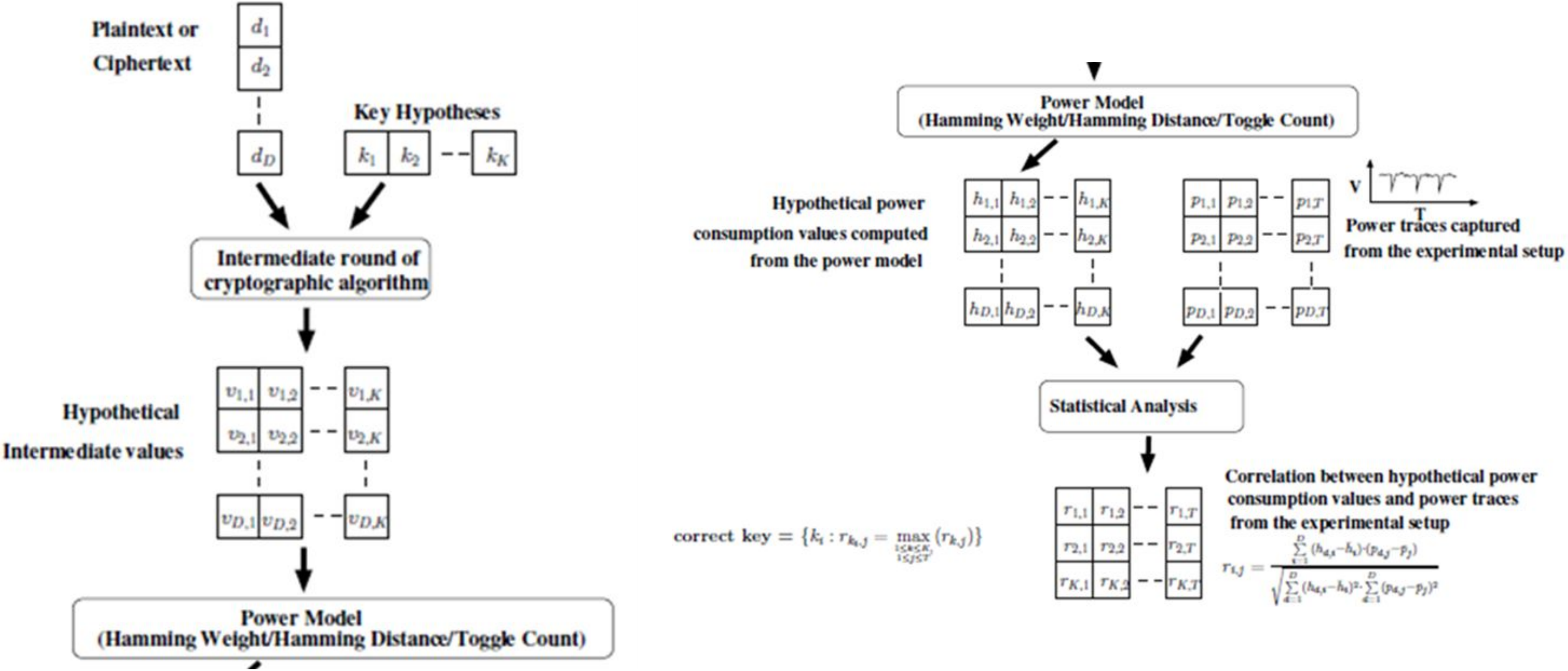
- Actual Power values for all the NSample encryptions are stored in the array trace[NSample][NPoint].
- Attacker first scans each column of this array and computes the average, and stores in meanTrace[NPoint].
- Likewise, the hypothetical power is stored in the array hPower[NSample][NKey].
- Attacker scans each column and stores in the array meanH[NKey]

Correlation Matrix



$$C[i][j] = \frac{\Sigma_{k=0}^{NSample}(hPower[i][k] - meanH[i])(trace[j][k] - meanTrace[j])}{\Sigma_{k=0}^{NSample}(hPower[i][k] - meanH[i])^2 \Sigma_{k=0}^{NSample}(trace[j][k] - meanTrace[j])^2}$$

Correlation Matrix



Fault Attack

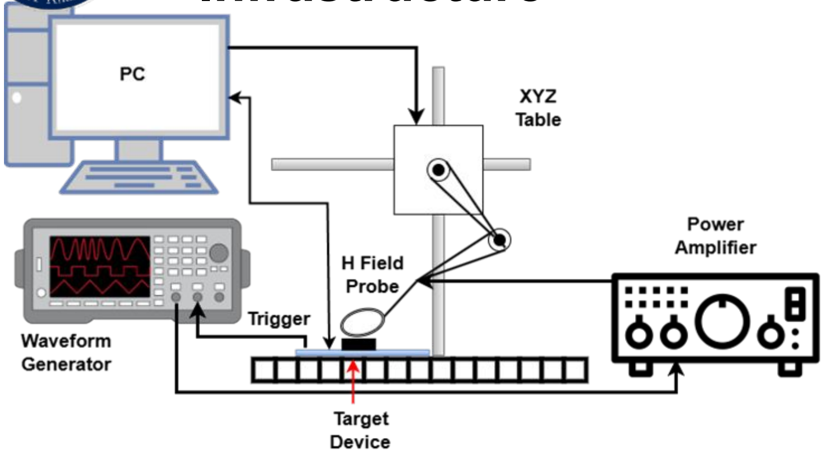
Electromagnetic Fault Injection:

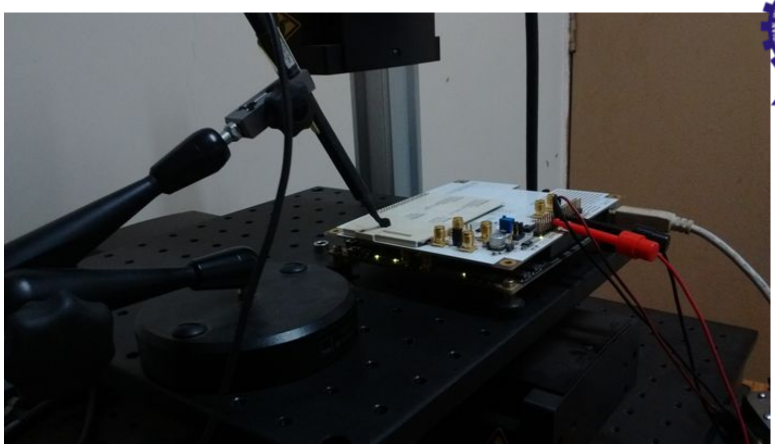
- Emission of electromagnetic pulses on a specific location of the device can induce a fault.

Laser Fault Injection:

- Emitting a laser pulse to a specific location of a device can cause a fault.
- The control of the attack is very high compared to the previous ones.
- This attack is invasive, since the packaging of the target chip must be at least partially removed (decapsulation).

Existing Fault Attack Infrastructure





EM fault injection setup

EM Fault injection along with side channel observation setup

Components in the setup:

- Arbitrary waveform generator: Keysight 81160A
- Constant-gain power amplifier: Teseq CBA 400M-260
- High-frequency near field H-probe: Rigol Near-field Probe 30MHz-3GHz
- XYZ table: Thorlabs SMC100

Thank You